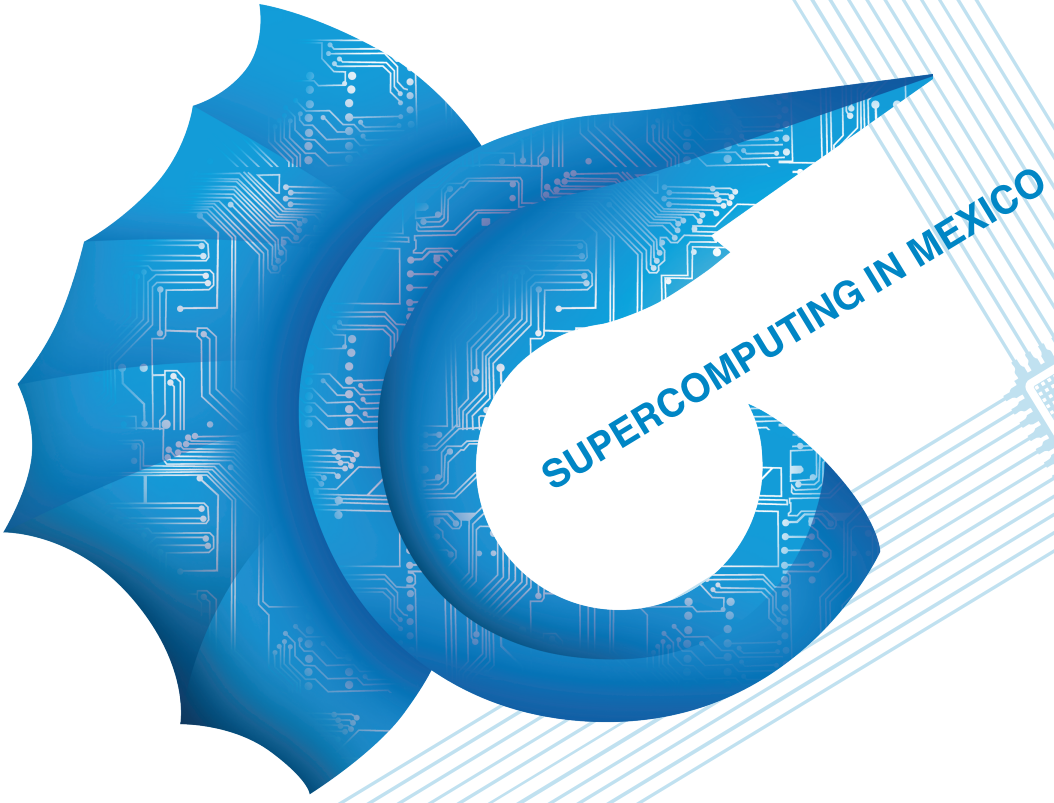
Supercomputing in México
A Navigation Through Science and Technology

Volume 4
1st Edition



THEMES:

Applications | Arquitectures | Parallel Computing | Scientific Visualization

UNIVERSIDAD DE GUADALAJARA
Red Universitaria de Jalisco

Editor: Dr. Moisés Torres Martínez

Supercomputing in México

A Navigation Through Science and Technology

4th INTERNATIONAL SUPERCOMPUTING CONFERENCE IN MÉXICO 2013

ISUM 2013 Conference Proceedings

ISUM 2013 Conference proceedings are published by the *Coordinación General de Tecnologías de Información*, Volume 4, 1st edition, December 19, 2013. Printed in Guadalajara, Jalisco México, December 19, 2013 in los *Talleres Gráficos de Transición*.

SUPERCOMPUTING IN MÉXICO

4th INTERNATIONAL SUPERCOMPUTING CONFERENCE IN MÉXICO 2013

Where Supercomputing, Science and Technologies Meet

Volume Editor

Dr. Moisés Torres Martínez

ISBN: 978-607-450-919-9

Universidad de Guadalajara
Ave. Juárez No. 976, Piso 2
Col. Centro, C.P. 44100
Guadalajara, Jal., México

Volume IV:
1st Edition

Obra Completa: 978-607-450-347-0

Printing: 700 Issues/700 ejemplares
Printed in México

CONTENTS

Architectures

Parallel Computing

Scientific Visualization

Preface

The theme Navigation through Science and Technology describes the state of supercomputing in Mexico in its voyage to making significant strides in its uses of supercomputing in research and development. This voyage took place in the beautiful port of Manzanillo, Colima México where the 4th International Supercomputing Conference in México was hosted. The conference as it has been doing for the past four years had over 500 participants throughout the three days of the event and more than 100 presentations, 7 industry speakers, and 6 keynote speakers. The talent that came from across the country and the international research community was vital to the success of this research conference.

The conference had some of the most internationally renowned researchers in High Performance Computing (HPC) who contributed to the knowledge shared during the three days of the conference. They set the conference momentum by sharing some of the most contemporary topics in HPC. For example, Dr. Carl Kesselman from the University of Southern California in the U.S.A. described the work of an Institute for Empowering Long Tail Research with collaborators from the University of Chicago, the University of Washington, the University of California at Los Angeles and the University of Arizona. He shared the work they´ve done to address the problems of big-data and small teams and illustrated solutions for a number of different application domains.

Similarly, his compatriot Dr. Jack Dongarra from the University of Tennessee spoke about how high performance computing has changed over the last 10-years and shared a look toward the future in terms of trends. He also shared that some of the software and algorithm challenges have already been encountered, such as management of communication and memory hierarchies through a combination of compile—time and run—time techniques, but the increase scale of computation, depth of memory hierarchies, range of latencies, and increased run—time environment variability will make these problems much harder. His talk stimulated much interest from the conference attendees and created a rich discussion on the software and algorithm challenges in HPC.

The conference also had the most renowned supercomputing guru from Europe, Dr. Mateo Valero from the Barcelona Supercomputing Center, who spoke about the Killer mobiles: the way towards energy efficient High Performance Computers. He shared that the Mont-Blanc project aims to build an alternative approach towards Exascale based on aggregating parts from the embedded and mobile market, which offer a better FLOPS/Watt ratio and a lower unit cost, at the expense of lower peak performance per chip. He reviewed the design philosophies of several vendors, including HPC compute accelerators, and ARM-based mobile application processors in terms of peak performance, memory bandwidth, and energy efficiency: and reviewed how the OmpSs programming models exploits the benefits of the Mont-Blanc approach while overcoming the drawbacks. Without saying, Dr. Mateo Valero contributed to the conference immensely with his talk and interaction with conference participants throughout the three days. He demonstrated that his willingness to work with young researchers from México is one of his highest priorities.

From Europe to Australia, we had the pleasure of having Dr. Rajkumar Buyya from the University of Melbourne in Australia join us and share his pioneering work on cloud computing covering a 21st century vision of computing and identified various IT paradigms promising to deliver the vision of computing utilities and architecture for creating market-oriented Clouds by leveraging technologies such as VMs. His

presentation stimulated much interest among conference participants due to the extensive work Dr. Buyya has done in the area of cloud computing.

Latin America was represented by Dr. Carlos Jaime Barrios from the Industrial University of Santander in Bucaramanga, who shared a discussion of a roadmap on advanced computing facing the challenges of the Exascale questions and how Latin America responds to these challenges observing energy-aware impact, environment, usability and scientific/technological realities. His presentation provided lessons learned and an interesting perspective from Latin America which was rather enriching for supercomputing center directors in México.

We also had the pleasure of having Dr. Jaime Klapp from the National Nuclear Sciences Research Institute and CINVESTAV represent México at the conference. He shared the evolution of High Performance Computing (HPC) in México during the past three decades. He pointed out the challenges the country faces in connectivity and the NIBA project to ameliorate that situation. He also shared the development of the ABACUS supercomputing Center under development in the State of México, which is expected to be the most powerful supercomputing center in the country. His presentation stimulated much discussion among the audience and throughout the three days of the conference among the technologists and supercomputing center directors who were particularly interested in the methodology used to build the ABACUS center.

In addition to these keynote speakers, the conference had rich discussions and presentations from the conference sponsors who shared emerging technologies and products from industry. Their perspective added a great deal of interest from all the young researchers who are always seeking to learn the new industry tendencies in new technology development. ISUM is well aware of the importance of industry partners to continue to foster young researchers interest in the uses of supercomputing to solve some of the most complex problems in our society today.

This book is a sample of the research work being done throughout the country and Latin America. There is no doubt that the works presented in this book are inspired by the excellent keynote speakers that ISUM presented and the innate interest of researchers in continuing to advance supercomputing in México through the use of supercomputing to solve the myriad of research problems the country is facing and affecting our society. The nineteen articles with more than eighty authors from México, United States of America, Europe and Latin America are a real testament of the research collaboration being conducted to advance research with the uses and in supercomputing.

It is certainly an honor to congratulate all the authors for contributing their work to this 4th edition of this book and to thank you for the confidence instill me to be the editor of this important work, which is one of its kind in the country. Also, I thank the ISUM National Committee for all the work that it has done in making this publication a reality because without their support ISUM and this publication would not be possible.

I invite you to browse through this publication and find the articles that are of your interest and read it to advance your own work in your respective area. At the same time, I encourage you to contribute your own work in future publications.

Dr. Moisés Torres Matínez
ISUM National Committee, Chair

Acknowledgements

This publication could not be possible without the contributions of participants representing institutions from México, Latin America, United States, Asia and European Union whom participated in this "4th International Supercomputing Conference in México 2013" with presentations and paper submittals. It is a great honor to have had the participation of the many authors who contributed to this publication and conference attendees for their participation, presentations, questions, and interaction making this conference a success.

In addition, this conference was also possible due to the many important contributions from the following people who made this event a success. We gratefully thank everyone for their individual contribution.

Universidad de Guadalajara

Mtro. Itzcóatl Tonatiuh Bravo Padilla,	Rector General
Dr. Miguel Ángel Navarro Navarro,	Vicerrector Ejecutivo
Lic. José Alfredo Peña Ramos,	Secretario General
Mtra. Carmen Enedina Rodríguez Armenta,	Coordinadora, Coordinación General Administrativa
Dr. Luis Alberto Gutierrez Diaz de Leon,	Coordinador, Coordinación General de Tecnologías de Información
Mtro. Manuel Moreno Castañeda,	Rector, Sistema de Universidad Virtual (SUV)

Executive Committee in University of Colima

Juan Manuel Ramírez Alcaraz	José Rubén Tejeda Santa Ana
Carlos Alberto Flores Cortés	Ricardo Acosta Díaz
Fernando Rodriguez Haro	María Andrade Aréchiga
Juan Antonio Guerrero Ibáñez	Sara Sandoval Carrillo
Jorge Enrique Preciado Velasco	Pedro Damián Reyes
Fermín Estrada González	Jorge Rafael Gutiérrez Pulido
Francisco Vaca Gutiérrez	Erika Margarita Ramos Michel
María Estela González Arellano	Martha Elizabeth Evangelista Salazar
Joaquín Carrillo Hidalgo	José Luis Álvarez Flores

Special recognition to the ISUM National Committee 2013 because without their participation and dedication in the organization of this event, the ISUM 2013 could not had been possible.

Andrei Tchernykh	(CICESE)	Lizbeth Heras Lara	(UNAM)
César Carlos Diaz Torrejón	(IPICyT-CNS)	Mauricio Carrillo Tripp	(CINVESTAV)
Cynthia Lynnette Lezama Canizales	(IPICyT-CNS)	Manuel Aguilar Cornejo	(UAM)
Jaime Klapp Escribano	(ININ)	Ma. Del Carmen Heras Sanchez	(UNISON)
René Luna García	(IPN)	Moisés Torres Martínez	(UdeG)
José Lozano	(CICESE)	Salma Jalife	(CUDI)
Juan Carlos Chimal Enguía	(IPN)	Salvador Botello Rionda	(CIMAT)
Juan Carlos Rosas Cabrera	(UAM)	Salvador Castañeda	(CICESE)
Juan Manuel Ramirez Alcaraz	(UCOLIMA)	Verónica Lizette Robles Dueñas	(UdeG)

Introduction

A Review of Research Studies in Supercomputing in México

By
Dr. Moisés Torres Martínez
University of Guadalajara
moises_torres10@hotmail.com

This work presents an overview of the research presented on this 4th edition of Supercomputing in México: A Navigation through Science and Technology. It gives a summary of the studies presented in this book covering the areas of Applications, Architectures, Parallel Computing and Scientific Visualization. It also provides a perspective on the advancement of supercomputing and its uses to conduct high quality research in México.

Introduction

México is making significant leaps in undertaking some of the most complex problems the country is facing in science and technology today and making use of the power of supercomputing to achieve faster results. This progress is not only reflected on the growth of supercomputing centers throughout the country and their individual evolution to meet the computing power needs of scientists from across the nation. It is also reflected on the research works conducted in the myriad of academic and research institutions nationally. The more than 100 studies presented in the 4th International Supercomputing Conference in México (ISUM) is a sample of the interesting and important work being conducted in academic and research institutions represented at this important meeting of the minds in supercomputing. Although, the works presented were of superb quality and gave the conference attendees a view of the works being done nationally and internationally, it is acknowledged that many of the high profile research conducted around the nation continues to be presented abroad and seldom shared in conferences held in México.

Presenting research work abroad continues to be of higher benefits to the researcher in México than to present in national conferences like ISUM or others that are held in the country. It is not to say that presenting abroad is not important and shouldn´t be done. Clearly, sharing the research work abroad is important to the growth of any researcher and to obtain validation from the international community on the work that one is doing. At the same time it does represent the country as making significant leaps in science and technology in the eyes of the international community. These benefits are unquestionable. What is equally important is that these works are also presented in the many forums that exist in the country to continue to foster the interest of young scientists in their quest to define the areas of study that are of most interest. Sharing the important work that every researcher does with the scientific and technological community through the many forums that exist

is invaluable to the growth of scientist in the country. With this spirit in mind, the International Supercomputing Conference in México (ISUM) was designed to create a space for young and seasoned scientists using the power of High Performance Computing (HPC) to share their research work with the national and international community within the country.

The past four years ISUM has taken the leadership in creating a space for scientists to have a broader forum beyond colloquiums within their institutions to share their research work with colleagues. At the same time this space has served to bring a greater awareness nationally of the types and quality of science and technology research work conducted making use of the many supercomputing centers around the nation. Building these types of events is one way of fostering and advancing the use of supercomputing in science and technology research. Thus, the importance for all research work conducted in the country to be shared at these events is what will continue to foster the growth of using the power of supercomputing to advance science and technology research in México.

This book presents the works of more than 36 academic and research institutions in México, Colombia, Argentina, China and Germany representing the best works presented in the ISUM 2013 in Manzanillo, Colima México. These works fall in four themes: applications, architectures, parallel computing and scientific visualization. A summary of the works in this publication is presented below:

Applications

In this section the authors share a series of articles that discuss the uses of high performance computing and its application in broad areas of study. For example Martinez Gutierrez and Ibarra Bracamontes in their article a Model for Synovial Articular Cartilage under Compression share a system modeled numerically by using the dynamic mesh technique; it allows a constant remeshing in the system due to deformations. The results show that at higher viscosities, the system can support higher pressures. In a similar area Jardón Valadez and García Arellano discuss in their article the Modulation of the enzymatic activity in the dTDP-L-rhamnose biosynthesis pathway, in which they performed exploratory MD simulations to test the stability and the conformational dynamics of the RmlA enzyme in aqueous solutions. Using HPC, they were able to run benchmarks on a relatively large system including ≈ 60 thousand atoms, and using a classical force field.

In addition, the article Assessing the Portability of JavaFX-based rich internet applications is a unique application of HPC where the authors show how they depend on the platform from which those applications are accessed. From all the platforms evaluated in this work, they found that Windows is the only one with full support for JavaFX as either an application running on the computer equipment (.jar applications) or an application hosted on a web server.

In the article Electroencephalographic High-Performance Cloud Data Processing System, the authors Ortega Magaña et al. presented an on-line processing platform that processes a high number of signals acquired by many sensors (multivariable data) on a computer cluster. This tool allows the user to upload to the cluster datasets acquired from EEG (Electroencephalographic) experiments

and characterize them with their own attributes in the main database, so the user can share that information with other users, request common scientific data from other users, or process the datasets with the modules included in the platform.

Lopresti, Piccoli & Reyes in their article Solving Efficiently Multiple Approximate Similarity Queries, where they present a solution to solve many queries in parallel taking advantage of GPU´s architecture. In this work they show an implementation that uses a Permutation Index to solve approximate similarity search on a database of words. Their results showed improvements in every different GPU architecture considered. Both obtained speed up and throughput are very good, showing better performance when the load work is hard.

A neighbour search code for galaxies by Ortega-Minikata, Torres-Papai & Andernach presents preliminary results of applying a neighbour search code to a large sample of galaxies drawn from the Sloan Digitial Sky Survey (SDSS). They draw a sample of target galaxies from the spectroscopic catalogue of the SDSS, which has redshift measurements for all its ~ 8 x 105 galaxies. One of their preliminary results shows the relation between N, the inferred morphology and the dominant activity type of the target galaxies, as well as the relation between N and the star formation history of the target galaxies. Their environment measurement is able to reproduce the morphology-local density relation, and is independent of the morphology-activity relation. We also show a clear sequence of SFH with N, showing that galaxies in denser environments tend to be older than galaxies in less dense environments.

The study Optimization of a distributed repository of multimedia files using High Performance Hardware and Software by Cristobal-Salas et al. present the details of optimizing a computer system for retrieving multimedia teaching materials by introducing multi-thread and distributed computing techniques to separate the system's layers to be processed by different software agents and hardware servers. Preliminary results show that the new system improved its performance by reducing the most time consuming operations which are to find relevant files associated or related to the original user request for information.

In a second article by Cristobal-Salas et al. they present a Project-based Learning Approach to Design Performance Microprocessors using Low Cost Hardware. In this paper they present a didactic strategy where undergraduate students design a hardware simulation of a microprocessor using a micro-controller. This simulation is used as a teaching tool in a course about high performance computer organization and architecture. This teaching tool helps electronics engineering and computer engineering students to be familiarized, in the practice, with hardware issues and challenges involved in the HPC area.

Architectures

Garcia-Gomez et al. in their article Autonomous Recovery Technology for Fault Tolerance in Distributed Service-Oriented Mission Critical Systems, they briefly present the Autonomous Decentralized Service Oriented Architecture (ADSOA), which has been proposed as a service-oriented architecture for designing MCS, which has been mainly utilized in financial sector

applications. They proposed a cloning mechanism to recover quickly and efficiently the operational service level when a decrease on it is detected. We have built a prototype to verify the feasibility of this technology.

Parallel Computing

The article Suffix Array Performance Analysis for Multi-core Platform presented by Ochoa, Gil Costa & Pintista, analyzes the performance of the suffix array index by means of the Perf and PAPI tool on a multi-core environment. Parallel codes were implemented with the OpenMP library. Experiments were performed with different index size and different patterns length over a 32-core platform. They ran experiments to evaluate hardware features directly aimed to parallelize computation. Results show that read-only operations performed over shared data structures affect performance, specially running time. They also propose an optimization scheme which allows increasing the number of hit cache and tends to improve running time.

In an interesting study, Modeling Rock Failures using Particles and Potential Energy Functions under a GPU Parallelized Scheme, the authors Cardoso-Nugaray and Botello-Rionda present an interesting study in which they use a GPU to perform an approach that uses particles with associated energy potentials to allow the interaction between linked particles and unliked particles. They found a high efficiency in using parallelized approach with GPU´s.

In the work by Tinetti et al., Optimization and Parallelization Experiences Using Hardware Performance Counters, the authors examined the uses HPC to take advantage of its processing power to analyze two problems 1) updating and parallelizing legacy (HPC/numerical) software, and 2) analyzing different problems and approaches to optimization and parallel processing in clusters. They found that raw hardware event counters do not always directly provide useful information and they found some guidelines for evaluating performance using those counters in the context of optimization and parallelization.

The work presented by Tapia et al, Calculation of Vibration Modes and Optimization of Structures subject to Seismic Actions, they share a parallelized seismic analysis algorithm, applied to the optimal design of structures. The method proposes different structure configurations, varying location, quantity, material and cross sectional type of elements that compose it, and searches for the design with the least amount of material that meets current structural building regulations. The objective of this work is to achieve better solutions for sophisticated calculations in a shorter time and taking advantage of the parallel computing infrastructure available, ensuring a correct application of numerical methods.

Gonzalez Mendez & Alvarado Nava´s work on Parallel Encoding of Audio in Architectures Multi-Core, they present the adaptation of the encoder LAME for the encoding process WAV to MP3 performed in parallel to obtain acceleration proportional to the number of cores in the architecture without losing the audio quality. The uses of multi-core architectures speed up the response time of encoders.

In an experimental study in the Comparative of Sorting Methods using OpenMP, Guerrero et al.

share a comparative analysis of the computational cost of sorting algorithms. The results provide information to choose a sorting method suitable for the type of application and available computing power. They suggest applications that require efficient sorting methods for large data volumes.

Scientific Visualization

Ramirez Rivera, Chapa Vergara & Meneses Viveros in their work of a DVO Model Using Mask for Data Distribution on Tiled Display, share a proposal for the data distribution of the DVO model CBTDs driver. The data distribution is one of the general functions defined in the DVO model; which are data distribution, display distribution, event handler and synchronization. They proposed and algorithm strategy to crop images using a mask. The proposed model allows the creation of a distributed window manager. One of the most important characteristics in the model definition is the data distribution. It shows an algorithm based on mask position to split an image, the division is performed on each node to distribute the work.

The article Generating Large Varied Animated Crowds in the GPU by Rudomin et al. discusses several steps in the process for simulating and visualizing large and varied crowds, in real time, for consumer-level computers and graphic cards (GPUs) and examining cost efficient approaches.

The work conducted by Wan, Huang & Song in their work on Molecular Simulation of Hidration Layers on Hydrophilic Mineral Surfaces in Water, they highlight the molecular simulation of hydration layer on mineral surfaces in water. It includes the structure of the hydration layers, density and viscosity in the layers, hydration layer thickness, hydrogen bonding, and adsorption sites, etc. Also, they summarized the commonly used software in the molecular and suggest that the MD simulation plays an important role to study hydration layer and reveal the effect of hydration layer in many fields.

In the article Speeding-up the Rendering Process of High Definition Animations using Four Quad-Core Intel Xeon Microprocessors, Cristobal-Salas et al. present the experience of improving the rendering process of high definition animations using Blender 2.64a and the native option for rendering called Blender-Render. A 3D simulation of mechanical equipment was used to test the improvements in a 2x2.4GHz Quad-Core Intel Xeon MACPRO workstation. Also, the impact in energy consumption in each optimization is analyzed. Their results show that energy issues could be a key point to consider when rendering large 3D animations.

Conclusion

The summary of these studies gives us a brief lens of the work presented in this 4th volume of Supercomputing in México: A Navigation through Science and Technology. However, the contents of each article provide the essence of the work conducted, which is not reflected in this summary. This collection of studies is a real testament of the quality of works presented in ISUM 2013, giving us a brief snap shot of some the works done in and out of the country in supercomputing. It is breathtaking to see the growth of supercomputing in México and the course that is taking. It

is evident in the multiple studies presented in this book that scientists are making greater use of supercomputing to achieve their results. We know that as supercomputing continues to evolve in México and Latin America we will see studies that are focused on the evolution of HPC and not so much on the uses of HPC, and contribute significantly to the progress of these systems.

It is with much gratitude that I thank the many authors who contributed to this publication, and the review committee who contributed their time to make this book a collection of quality work. On behalf of the National Committee, I invite you to read about this pioneering work and to participate in the upcoming International Supercomputing Conference in México and share your research work with the scientific community by presenting and submitting your work for publication.

Applications

A Project-based Learning Approach to Design High Performance Microprocessors using Low Cost Hardware

Alfredo Cristóbal-Salas, Efrén Morales-Mendoza, Neiel Israel Leyva Santes, Carlos Azuara Meza
Facultad de Ingeniería en Electrónica y Comunicaciones, Universidad Veracruzana
Poza Rica de Hidalgo, Veracruz, México 93390
{acristobal, efmorales}@uv.mx, {nleyvas, carlos8905}@gmail.com

Abstract

This paper presents a didactic strategy where undergraduate students design a hardware simulation of a microprocessor using a micro-controller. This simulation is used as a teaching tool in a course about high performance computer organization and architecture. This teaching tool helps electronics engineering and computer engineering students to be familiarized, in the practice, with hardware issues and challenges involved in HPC area. Students are stimulated to design their own instruction set considering: energy consumption, heat problems and the space needed to implement their design. Also, students are encouraged to detect, design and implement strategies to achieve high-performance.

Keywords: Problem-based learning, Computer organization, Computer architecture, ASM programming, micro-controller, PIC16F84A

I. Introduction

Computer organization and architecture is a common course that is offered at universities throughout the world. Traditionally, teaching such a course to electronics engineering (EE) or computer engineering (CE) students can be considered as insufficient if lectures only consider theoretical didactic materials. If this is the case, then students have to rely on their imagination to understand hardware-related concepts. In most universities software tools are used as a tool to practice theoretical concepts; however, this type of simulators hide from students all hardware related details. For instance, EE and CE students need to learn about power consumption and optimization, heat control, and hardware alterations to achieve high-performance in microprocessors. These topics are better understood when dealing directly with hardware components instead of using software simulators. There have been several attempts to implement computer architecture courses where students deal with simulators such as: [2, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 18, 19].

On the other hand, working with hardware could be time and money consuming; that is why, this paper proposes a didactic tool based on low-cost and well-known hardware that may help students to implement their design easily. That is why, a micro-controller PIC16F84A is used as part of this didactic tool. Micro-controller technology offers the potential of simulating high performance systems at low cost and they have been used for many computational and application-oriented tasks. In particular, the PIC16F84A micro-controller is6 has a simple design, it can be considered as low cost hardware (under US$5 in Mexico, at

the time of writing) it has only 35 single word instructions; all instructions are single cycle except for program branches which are two-cycle. This micro-controller has 1024 words of program memory, 68 bytes of data RAM, 64 bytes of data EEPROM, 14-bit wide instruction words, 8-bit wide data bytes, 15 special function hardware registers, eight-level deep hardware stack, direct, indirect and relative addressing modes, I/O pins with individual direction control.

II. Didactic tool design and implementation

The idea on this design is to bring to EE and CE students a basic tool in which they can design their own instruction set taking advantage of PIC16F84A micro-controller characteristics. This didactic tool has to introduce the single-cycle computer features to students, from which, they have to create their own computer. In Figure 1, the two-address instruction format used to implement a simple computer on the PIC16F84A micro-controller is shown. As can be seen in Figure 1, our design considers 8-bit operations working on 8-bit data registers transferring data trough 8-bit data/operation bus. This design correspond directly with 8-bit registers in the micro-controller.

Our simulator contains two units (see Figure 2b): control (CU) and arithmetic-logic (ALU); it also has three cycles (see Figure 2a): Fetch, Execution and Store. The CU contains the following components: a clock (CK), and instruction register (OP), a decoder (DC), and a program counter (PC). The ALU contains the two data registers (D1, D2), a flag register (FL) containing the flags integer carry (C), zero (Z), result ready (RL), instruction ready (OL), a result register (R) and an instructional unit (OU). Out basic architecture has also a set of registers for code memory and data memory as seen in Figure 2c.

All previous registers are mapped into PIC16F84A registers as shown in Figure 2c in the range of addresses 0x0c to 0x12; addresses from 0x13 to 0x34 are reserved for code memory and addresses from 0x35 to 0x4F are reserved for data memory.

As part of hardware training, students are instructed to construct their own circuit according to the design shown in Figure 3 and the implementation in proto-board shown in Figure 4.

The previous design constitutes the basic computer implementation from which students have to design a new design.

III. Project-based learning (PBL)

In order to understand PBL it is necessary to review the Problem-based learning didactic strategy. Problem-based learning [1, 3, 15, 10] is a student-centered didactic methodology where students are challenged to learn about a specific topic through the experience of real-life or controllable problem solving. This strategy develops both problem solving general strategies and disciplinary knowledge bases. It also introduces students in an active role (as problem solvers) where they strengthen their skills. Problems, in PBL, have to describe ill-structured situations similar to the kind of problems students might confront in real-life.

Barrows (Barrows, 1996) defines the Problem-Based Learning Model as:

1. Student Centered Learning
2. Learning is done in Small Student Groups, ideally 6-10 people
3. Facilitators or Tutors guide the students rather than teach

4. A Problem forms the basis for the organized focus of the group, and stimulates learning

5. The problem is a vehicle for the development of problem solving skills. It stimulates the cognitive process.

6. New knowledge is obtained through Self-Directed Learning(SDL)

Unlike problem-based learning, the PBL [3] is "a systematic teaching method that engages students in learning knowledge and skills through an extended inquiry process structured around complex, authentic questions and carefully designed products and task". PBL is a variation of problem-based learning where the projects are meant to be solved inside school where several problems' variables can be controlled and during the academic period.

In [4], some key features that projects need to have in order to be used as part of PBL: (a) they are intended to teach significant content, (b) require critical thinking, problem solving, collaboration, and various forms of communication, (c) require inquiry as part of the process of learning and creating something new, (d) are organized around an open-ended Driving Question, (e) create a need to know essential content and skills, (f) allow some degree of student voice and choice, (g) include processes for revision and reflection, (h) involve a public audience. PBL can be engaging and motivating because constitutes the kind of problems students are willing to solve as professional engineers. This emotional component is an additional tool to enhance learning while promotes effective reasoning and self-directed learning.

It is important to apply PBL within the context of authentic tasks/problems where real-world restrictions and concerns can be easily detected and analyzed. In a PBL course, students and the instructor become colearners, coplanners, coproducers, and coevaluators as they design, implement, and continually refine their curricula. PBL is unique in that it fosters collaboration among students is more active, as you are engaged as a problem-solver, decision-maker, and meaning-maker, rather than being merely a passive listener and note-taker. [17]. PBL helps students to be in charge of their own learning in time and quantity because they learn what they need in order to solve the problem and they do it as fast as they are willing to finish the task.

The PBL strategy for EE and CE students is explained next, but, it is important to remember that the 8-bit operation microprocessor design, explained above, is given to students as a starting point for their project.

Project #1. Simulation of 8-bit instructions for x86 architecture. In this project, students have to implement (a) arithmetic operations: DEC, INC, MUL, NEG. (b) logic operations: XOR, NOT. (c) control operations: JA, JAE, JB, JBE, JE, JG, JGE, JMP, LOOP. (d) bit operations: RCL, RCR, ROL, ROR. (e) XCHG, MOV.

Project #2. Simulation of 4-bit instruction architecture. In this project, students have to simulate a 4-bit instruction microprocessor. To do this project, the diagram shown in Figure 5 is provided to students for implementation in a PIC16F84A micro-controller.

Project #3. Simulation of 16-bit instruction architecture. In this project, students have to simulate a 16-bit instruction microprocessor. To do this project, the diagram shown in Figure

6 is provided to students for implementation in a PIC16F84A micro-controller.

Project #4. Simulating a SIMD architecture on an array of PIC16F84A micro-controllers. In this project, students have to implement the ADD, SUB, OR and AND operations using the SIMD architecture using at least three PIC16F84A micro-controllers one for the control unit and two for arithmetic and logic unit.

Project #5. Simulating a microprocessor with 8-bits pipelined operations using two PIC16F84A micro-controllers. Students are encouraged to modify the original 8-bit instruction microprocessor design to implement a three stages (fetch, compute, store) pipelined instruction execution. One micro-controller manages the stages: "fetch" and "store" while the other manages the "compute" stage.

Project #6. Simulating a microprocessor with 8-bit CISC operations. Students add vector operations to instruction set of the original 8-bit instruction microprocessor design. Vector operations are: summation of a constant value to a vector and matrix transpose operation.

Project #7. Heat and Power Consumption analyze. Students need to design hardware to measure PIC16F84A temperature and design an electronic circuit to maintain the temperatures in the range defined in the micro-controller datasheet.

IV. Preliminary results

This PBL strategy was tested with two students from Universidad Veracruzana as a didactic prototype. These students have already covered more than 90% of credits of the Electronics Engineering and Communications study program. Students are related to binary operations and micro-controllers programming but they had not worked with computer architecture before; neither, they were related to high performance computing when they started working on these projects.

As a first goal, students implemented the 8-bit operation computer architecture mapped on a PIC16F84A micro-controller as instructed in this paper. After this implementation, students selected the project #3. "Simulation of 16-bit instruction architecture". Finally, they selected the Project #2. "Simulation of 4-bit instruction architecture" as their last project. Time invested in each project is presented in Table I. During the entire process students were supervised by a teacher who oriented them and recommend didactic material to reinforce student´s knowledge.

After accomplished the three projects, students were interviewed in order to know their opinion about the PBL strategy. The transcript of this conversation is shown next.

Q: which do you think was the hardest problems you found along the three projects?

Student1: There were binary operations that I only knew in theory but not in practice. Also, I did not know how computers execute instructions and neither I knew about how to address registers in assembler codes.

Student2: the 4-bit instructions microprocessor design was the hardest problem, because, I had to use many micro-instructions to execute just one microprocessor instruction. Also, I did not know how to use binary operations in real-life problems.

Q: what do you think was the easiest part of doing the projects?

Student1: All topics related to PIC16F84a because I have already practiced micro-controller programming during the "microprocessors and micro-controllers" course. Also, I did extra projects after classes in order to better understand the topic. Furthermore, I prefer applied knowledge rather to passive learning in theoretical lectures.

Student2: I already knew about binary operations and my professional orientation is for the applied electronics.

Q: what did you learn from these three projects?

Student1: More tricks in micro-controller programming and its limitations such as: few I/O ports and small memory capacity, logic operations, building a complex circuit in a proto-board, what pseudo-code is and how to design it. Also, I learned about microprocessor architecture, instruction set design and implementation and register map design.

Student2: I learned to applied binary operations into real-world problems.

Q: what advantages do you perceive in the PBL strategy?

Student1: there is more responsibility on the student project success. I think if this strategy is applied in our School there would be many changes in classes and students would be motivated. Also, I think if students have many projects per semester then time will be an additional problem because projects take time. But on the other hand, one project can validate more than one courses and this could reduce the student´s effort.

Student2: The PBL is convenient to students who are more hardware orientation and train you better for real-world jobs.

Q: did this experience make you interested in high performance computing?

Student1: yes because now I understand how to optimize microprocessors.

Student2: yes

After the interview, students were questioned about how to implement pipeline, superscalar, vector, and multi-core microprocessors. In all cases, students have a basic idea on what modification they needed to implement in their design in order to implement each type of microprocessor. Even more, students gave detail information on the changes in microprocessor design and their consequences in performance and energy consumption. Finally, students commented on how a compiler should work in order to exploit microprocessors features.

V. Acknowledgement

This work is sponsored, in part, by Universidad Veracruzana through Facultad de Ingeniería en Electrónica y Comunicaciones and Red Iberoamericana de Supercómputo through the project #FP7-288883: "A network for supporting the coordination of Supercomputing research between Europe and Latin America" for the logistic and the financial support to accomplish this project.

VI. Conclusions

This paper proposes the idea of introducing hardware based course in learning computer organization and architecture. The given teaching tool is designed for electronics engineering and computer engineering study programs. This tool helps these students to

be familiarized practically with computer organization and architecture through development of their own instruction set, computer programming and interfacing experiments. Also, the micro-operation of the computer module and its assembler are implemented on PIC16F84A micro-controller is described.

This teaching tool was implemented in the School of Electronics Engineering and Communication at Universidad Veracruzana and students exposed to this didactic tool accomplished three of seven projects described in this paper along 353 hours distributed in 8 months period. Students successfully accomplished the tasks and after that they showed able to implement HPC hardware. Also, they felt motivated to continue working in the HPC area but with more realistic projects. Finally, students proposed that this PBL strategy should be implemented in the School of Engineering where students can practice all theoretical concepts in real-world projects.

VII. References

[1] Barrows, H. S. (1996). "Problem-based learning in medicine and beyond: A brief overview". New Directions for Teaching and Learning 1996 (68): 3–12. doi:10.1002/tl.37219966804.

[2] Bedichek, R.C. (1995). "Talisman: Fast and accurate multicomputer simulation," in Measurement and Modeling of Computer Systems, 1995, pp. 14–24

[3] Buck Institute for Education (2009). PBL Starter Kit: To-the-Point Advice, Tools and Tips for Your First Project. Introduction chapter free to download at: http://www.bie.org/tools/toolkit/starter

[4] Buck Institute for Education. (2013). Introduction to Project-based learning. Last access: January 22, 2013. Retrieved from: http://www.bie.org/images/uploads/general/20fa7d42c216e2ec171a212e-97fd4a9e.pdf

[5] Brewer, E.A., Bellarocas, C.N., Colbrook, A., Weih, W.E. (1991) Proteus: A High-Performance Parallel-Architecture Simulator. Technical Report LCSITR-5 16, MIT Laboratory for Computer Science, September 1991.

[6] Calazans, N.L.V.; Moraes, F.G. (2001). "Integrating the teaching of computer organization and architecture with digital hardware design early in undergraduate courses," Education, IEEE Transactions on , vol.44, no.2, pp.109-119, May 2001 doi: 10.1109/13.925805

[7] Fida A.Y. and Krad, H. (2011). Teaching Computer Architecture and Organization using Simulation and FPGAs. International Journal of Information and Education Technology, Vol. 1, No. 3, August 2011

[8] Djordjevic, J.; Milenkovic, A.; Grbanovic, N. (2000). "An integrated environment for teaching computer architecture," Micro, IEEE , vol.20, no.3, pp.66-74, May/Jun 2000 doi: 10.1109/40.846311

[9] Djordjevic, J.; Nikolic, B.; Milenkovic, A. (2005). "Flexible web-based educational system for teaching computer architecture and organization," Education, IEEE Transactions on , vol.48, no.2, pp. 264- 273, May 2005 doi: 10.1109/TE.2004.842918

[10] Gasser, K.W. (June 2011). "Five Ideas for 21st Century Math Classrooms". American Secondary Education 39 (3): 108-116.

[11] Goldschmidt. S. (1993) Simulation of Mul-

nprocessors. Accuracy and Performance. PhD thesis, Stanford University, June 1993.

[12] Grunbacher, H.; Khosravipour, H. (1996). "WinDLX and MIPSim pipeline simulators for teaching computer architecture," Engineering of Computer-Based Systems,1996. Proceedings., IEEE Symposium and Workshop on , vol., no., pp.412-417, 11-15 Mar 1996. doi: 10.1109/ECBS.1996.494568

[13] Henderson, W.D. (1994). "Animated models for teaching aspects of computer systems organization," Education, IEEE Transactions on , vol.37, no.3, pp.247-256, Aug 1994 doi: 10.1109/13.312133

[14] Hyde, D.C. (2000). "Teaching design in a computer architecture course," Micro, IEEE , vol.20, no.3, pp.23-28, May/Jun 2000 doi: 10.1109/40.846306

[15] Neville, AJ (2009). "Problem-based learning and medical education forty years on. A review of its effects on knowledge and clinical performance.". Medical principles and practice: international journal of the Kuwait University, Health Science Centre 18 (1): 1-9. PMID 19060483

[16] Nguyen, A.-T.; Michael, M.; Sharma, A.; Torrellas, J. (1996) "The Augmint multiprocessor simulation toolkit for Intel x86 architectures," Computer Design: VLSI in Computers and Processors, 1996. ICCD '96. Proceedings., 1996 IEEE International Conference on , vol., no., pp.486-490, 7-9 Oct 1996 doi: 10.1109/ICCD.1996.563597

[17] Purser, R.E. (2013). Problem-based learning. Last access: January 22, 2013.Retrieved from http://online.sfsu.edu/rpurser/

[18] Sharma, A. (1996). Augmint, a Multiprocessor Simulator. Master's thesis, University of Illinois at Urbana-Champaign, 1996.

[19] Theys, M.D.; Troy, P.A. (2003). "Lessons learned from teaching computer architecture to computer science students," Frontiers in Education, 2003. FIE 2003 33rd Annual , vol.2, no., pp. F1C- 7-12 Vol.2, 5-8 Nov. 2003 doi: 10.1109/FIE.2003.1264667

A Model for Synovial Articular Cartilage Under Compression

Nancy Martínez-Gutiérrez[1], Laura Alicia Ibarra-Bracamontes[1]*,
Sergio Ricardo Galván-González[1], Sixtos Antonio Arreola-Villa[1], S. García-Hernández[2].
[1]Facultad de Ingeniería Mecánica, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Michoacán, México.
[2]Instituto Tecnológico de Morelia, Morelia, Michoacán, México.
**e-mail: laibarrab@gmail.com*

Abstract

A joint is the junction of two or more bones where a relative movement between their components is produced. In the case of a synovial joint occurs a sliding between the bones; it happens through an interaction between the cartilage that covers the bones and synovial fluid. This interaction allows for lubrication on the contact region. In this paper we model the knee articular cartilage in contact with a layer of synovial fluid that is under compressive loads [1-2]. The proposed model takes into account porosity parameters in the cartilage as well as deformations in the region of the liquid by compression. The system is modeled numerically by using the dynamic mesh technique; it allows a constant remeshing in the system due to deformations. As a result of the simulation the pressure distribution in the contact area between the cartilage and synovial fluid, and the distribution of the velocity vectors in the biofluid are presented.

Keywords: synovial articular cartilage model, deforming zone, porous layer.

1. Introduction

A joint is an anatomical entity that consists of at least two bones, which have a relative mobility and an interface that allows sliding between them. In the case of synovial joints they have two important elements, which are the cartilage and synovial fluid. The cartilage is a relatively strong tissue that supports weight, but without reaching the bone strength. It is the part of the joint that covers and cushions the ends of bones. The Synovial Fluid (SF) is a highly viscous fluid, which allows lubrication and nutrition to the cartilage [3-5]. A synovial joint is the region of interest in the present investigation.

The knee is the largest joint in the human being and one of the most important of the synovial joints; it has unique characteristics and complexity in its functionality. Until this moment, there are different studies and researches in order to find solutions to various problems that affect the proper functionality of joints, trying to prevent low quality life in people due to joint disease [6-7]. Among the different diseases that may occur in the joints,

one of the main properties that it is affected is the viscosity of the synovial fluid. A decrease in its viscosity mainly affects the lubrication quality, giving rise to friction between the joint surfaces, causing wear on the cartilage. When cartilage is broken down, this may cause the bones to touch each other, producing pain and loss of mobility.

Current studies in joints modeling are limited because of the complexity of such systems and the many factors involved to simulate real functionality. In the literature, various theoretical models with analytical or some numerical solutions to the equations involved have been found [1-2, 8]. For example, some models have been proposed to take into account deformability or poroelastic properties in the articular cartilage properties, leading a better distribution of the mechanical loads [9-10]. Moreover, in Di Paolo's works [11-12] models for lubrication in hip or knee prosthesis using Finite Element techniques have been presented. Regarding to the works related with Biomechanical models and using the Dynamic Mesh Technique, there are some numerical models such as a mechanical heart valve [13] or blood flow with dynamic interfaces [14]. In those works triangulating the propagating grid points at every timestep simulates the interfacial motion.

In this paper some aspects and characteristics of the knee have been taken into account to simulate part of its functionality and some variations of its properties can be associated at joint illness. This work starts from the geometry proposed by Jurczak [1], in which the behavior of synovial fluid in human joints was studied from a biomechanical point of view. In that work a spherical bearing moves a thin fluid film in contact with a porous medium associated with cartilage was considered. Jurczak only presents analytical results for the pressure distribution in the fluid.

In this paper the results presented are based on Computational Fluid Dynamics (CFD) technique and the goal of the work is to get a better understanding of the lubrication mechanisms in human joints.

2. Methodology

2.1. Simulation Model Geometry

The present work is based on the geometry proposed by Jurczak [1], which considers a bio-spherical bearing that can displace a thin film of synovial fluid, which is in contact with a porous medium. In that work theoretical curves with circular geometry for pressure distribution were obtained in the region of the fluid. The pressure was calculated with respect to the angle φ measured from the symmetry axis (see Figure 1).

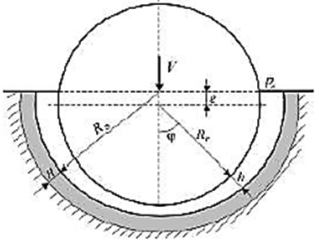


Figure 1. Geometric model proposed by Jurczak, on which this work is based [1].

According to Figure 1, the following parameters can be identified, which also were used in our numerical model: the maximum displacement of the moving wall, e, is expressed in a dimensionless quantity as follows:

$$\varepsilon = \frac{e}{C}, \qquad (1)$$

where C is the value of the initial thickness of the fluid zone, i.e. the difference between the radial distance to the fluid-cartilage interface, Rs, and the curvature radius of the moving wall, Rr, that is given by:

$$C = R_s - R_r. \qquad (2)$$

The speed of the applied load, V, is related to the deformation rate of the fluid region, $\dot{\varepsilon}$, as follows:

$$\dot{\varepsilon} = \frac{d\varepsilon}{dt} = \frac{1}{C}\frac{de}{dt} = \frac{V}{C}. \qquad (3)$$

The Jurczak's results [1] are based on the calculation of the pressure distribution in the SF region within a spherical bearing geometry. The equations for an unstable fluid motion with circular geometry are expressed in equations (4) and (5):

$$\frac{1}{R}\frac{\partial(Rv_x)}{\partial x} + \frac{\partial v_y}{\partial y} = 0, \qquad (4)$$

$$\rho\left[\frac{\partial v_x}{\partial t} + \left(\frac{R'}{R} + \frac{\partial}{\partial x}\right)v_x{}^2 + \frac{\partial}{\partial y}\left(v_x\, v_y\right)\right] =$$

$$-\frac{dp}{dx} + \mu\frac{\partial^2 v_x}{\partial y^2} - \eta\frac{\partial^4 v_x}{\partial y^4}, \qquad (5)$$

where ρ is the density of the fluid, μ is the coefficient of plastic viscosity, η is couple-stress viscosity, $\boldsymbol{R}$ is the equivalent radius, ρ is the pressure, and $\boldsymbol{v}$ is the velocity field.

Taking into account the following boundary conditions:

$$\begin{aligned} &v_x{=}0,\ v_y{=}V_H,\ \frac{\partial^2 v_x}{\partial y^2} = 0 && \text{for} \quad y = 0 \\ &v_x{=}0,\ v_y{=}\frac{\partial h}{\partial t} = \dot{h},\ \frac{\partial^2 v_x}{\partial y^2} = 0 && \text{for} \quad y = h \end{aligned} \qquad (6)$$

$$\left.\frac{\partial p}{\partial x}\right|_{x=0} = 0,$$

$$p(x_0) = p_0.$$

Figure 2 shows the geometry dimensions used in the numerical model of the present study that mainly consists in two walls and one interface.

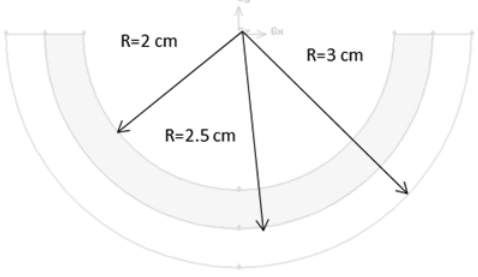


Figure 2. Dimensions of the system geometry for the simulation.

2.2. Governing equations for the porous media

Articular cartilage was represented as a porous medium and its porosity was the main property. Porous media can be modeled by the addition of a momentum source term to the standard Fluid flow equations. The source term is composed of two parts: a viscous loss term (determined by Darcy equation) and an inertial loss term; they can be expressed as:

$$S_i = -\left(\sum_{j=1}^{3} D_{ij} \mu v_j + \sum_{j=1}^{3} C_{ij} \frac{1}{2} \rho |v| v_j \right) , \quad , \quad (7)$$

where S_i is the source term for the momentum equation in the different directions (x, y, or z), μ is the viscosity, ρ is the density, |v| is the magnitude of the velocity, and *D* and *C* are prescribed matrices.

This equation for the momentum sink contributes to the pressure gradient in the porous medium, creating a pressure drop that is proportional to the fluid velocity (or velocity squared) in the SF region.

In the case of homogeneous porous media models, as the model used here, Equation 7 reduces as Equation 8:

$$S_i = -\left(\frac{\mu}{\alpha} v_i + C_2 \frac{1}{2} \rho |v| v_i \right) , \quad (8)$$

where α is the permeability coefficient and C_2 is the inertial resistance factor. *D* and *C* were specified as diagonal matrices with 1/α and C_2 values respectively on diagonals and zero for the other elements.

To calculate viscous resistance and inertial resistance Equations 9 and 10 were employed. The permeability, K, can be estimated by the Carman-Kozeny relationship, derived for a packed bed of uniform particles, expressed as follows:

$$\alpha = \frac{D_p^{\,2}}{150} \frac{\epsilon^3}{(1-\epsilon)^2} , \quad (9)$$

$$C_2 = \frac{3.5}{D_P} \frac{(1-\epsilon)}{\epsilon^3} , \quad (10)$$

where D_p is the mean particle diameter, and $\epsilon = K$ is the void fraction or porosity.

In this case, the porosity, *K*, of the system can be related to the permeability, α, by the following expression [1]:

$$K = \left(\frac{12 \alpha H}{C^3} \right)^{1/3} , \quad (11)$$

where *H* is the cartilage thickness or porous zone, and *C* is the initial thickness of the synovial fluid region, already defined in Equation 2.

Figure 3 shows the geometry used in the numerical model. It was used to study the behavior of the fluid in the different regions indicated, the synovial fluid and articular cartilage zones.

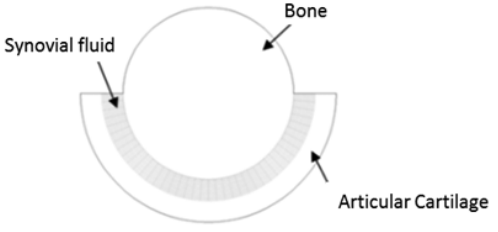


Figure 3. Schematic representation of the synovial joint with the Synovial fluid and the cartilage regions.

2.3. Schemes of discretization of the governing equations

To achieve sufficient accuracy in numerical predictions an appropriate differential scheme for the Navier-Stokes equations for incompressible and unsteady flow simulations is required. The numerical schemes for solving the flow governing equations have been a subject of concern at last decades. Most of these schemes have been used to solve the problem of wiggles appearing in solutions computed for high Reynolds numbers. In this study, the Reynolds number is very low and it has been possible to choose a simple approximation scheme known as *First-Order Upwind* (FOU) discretization scheme. This scheme was proposed by Courant et al. [14]. Since the mesh will be deformed in the fluid zone, the cell center and the face values of the cells they should be practically the same. This is the main feature of this discretization scheme because it should achieve computational savings since it is a first order exactitude scheme. In order to introduce the pressure calculated from the Momentum Equation into the Continuity Equation, the SIMPLEC (Semi-Implicit Method for Pressure-Linked Equations) algorithm was used in the *Fluent*® software employed for the numerical calculus.

2.4. Applying Dynamic Mesh

The computational tool used for the moving wall in the system was *Dynamic Mesh.* By this technique one boundary can be displaced and a consequent deformation can be obtained. To produce that deformation the remeshing of the deformed area is required to avoid the generation of negative volumes during simulations.

The Dynamic Mesh method can be used to model flows where the shape of the domain is changing with time due to motion on the domain boundaries. The mesh is updated at each time step based on the new positions of the boundaries. To use the Dynamic Mesh method, it is necessary to provide a starting volume mesh and the description of the motion of any moving zones in the model through user-defined functions (UDFs). By implementing Dynamic Mesh during simulation of the system locally remeshes in deformable regions are obtained, thereby avoiding to obtain negative volumes or cells that violate the skewness or size criteria that may be generated during the displacement and deformation in the system [15].

2.5. Deformation and remeshing

When the boundary displacement is large compared to the local cell sizes, the cell quality can deteriorate or the cells can become degenerate. This will invalidate the mesh consequently; will lead to convergence problems when the solution is updated to the next time step. The remeshing consists if the new cells or faces satisfy the skewness criterion; the mesh is locally updated with new cells to interpolate the solution from old cells. Otherwise, the new cells are discarded [15].

The Remeshing tool in Dynamic Mesh allows modifying and updating the grid over time. This means that the mesh is modified with the time and is stopped to the limit set for their displacement. In this work the maximum displacement considered was of 0.0025m in the negative Y direction.

Using Dynamic Mesh the grid is renewed or updated in the regions deformed according to

the motion defined on the border. In this case a smoothing method based on springs was used. In the spring-based smoothing method, the edges between any two nodes of the mesh are idealized as a network of interconnected springs. The initial spacing of the edges between nodes before any boundary motion constitutes the equilibrium state of the mesh. A displacement at a given boundary node will generate a force proportional to the displacement along all the springs connected to the node. The method of spring-based smoothing can be used to update any cell or face zone whose boundary is in motion or deformation features [15].

In order to apply the Remeshing and the spring-based smoothing method a tetrahedral mesh of the zone is required. Figure 4 shows the tetrahedral mesh used.

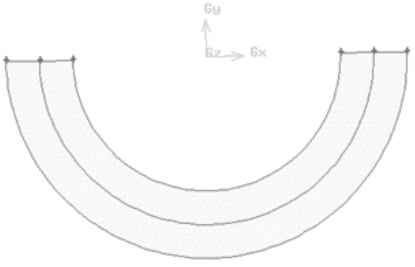


Figure 4. Tetrahedral mesh of the geometry used in this model.

Once the tetrahedral mesh of the geometry is generated, in this case the remeshing was associated to the Synovial Fluid region.

Figures 5a and 5b show the comparison between two meshes, which correspond to the states before and after to reach the maximum deformation in the synovial fluid zone respectively.

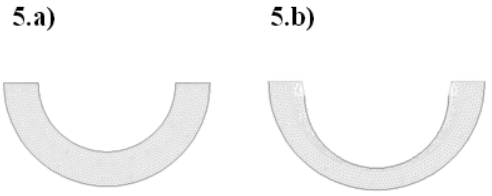


Figure 5. System mesh when maximum deformation is e = 0.0025m, a) before deformation, b) after reaching the maximum deformation.

2.6. Equation of motion of the moving wall

The numerical tool *"In-Cylinder"* available in the *Fluent*® software for Dynamic Mesh was used for the simulation. "In-Cylinder" tool is based on the rod-crank mechanism to move one piston and it may be used for transient or unsteady problems. This numerical tool was applied to deform the fluid region.

The movement of a moving wall represents the displacement caused by a load on the joint. The equations for the rod-crank mechanism were applied for the movement of a piston. Figure 6 outlines that mechanism.

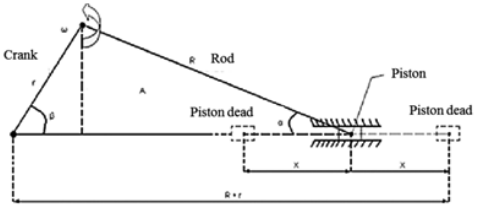


Figure 6. The rod-crank mechanism used to move a piston [16].

The equation for the moving wall displacement can be expressed in terms of the

position of the piston according to the diagram of Figure 4 as follows:

$$x = r - r\,Cos\,\beta + \frac{r^2}{2R}\,Sin^2\,\beta \quad (12)$$

For the speed of the moving wall displacement or strain rate, is associated with the respective piston speed equation:

$$V = r\,\omega\,Sin\,\beta + \frac{r^2}{2R}\,\omega\,Sin\,2\beta \quad (13)$$

Figure 7 shows the geometry of the numerical model where the moving wall and the deformation area are indicated. The interface boundary separates the fluid region from the cartilage zone.

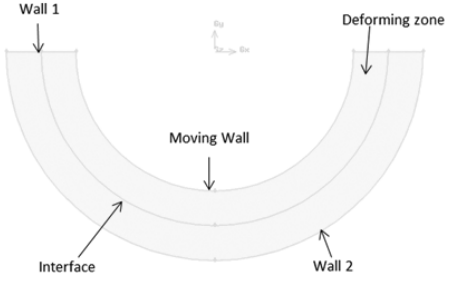


Figure 7. Schematic representation of the deformable area.

2.7. Boundary Conditions

The boundary conditions for the simulations were an incompressible unstable fluid with constant viscosity in a laminar flow and isothermal conditions as a Synovial Fluid. Velocities and pressures fields described the fluid. The viscosity of the biofluid, i.e. Synovial Fluid, was varied with respect to the water viscosity, μa.

A moving rigid wall with displacement in negative Y direction was implemented. The movement of this wall produces a deformation in the synovial fluid zone. The position and the speed of the moving wall are established by using the rod-crank mechanism. The deformation was obtained by means of Dynamic Mesh technique.

The articular cartilage area was defined as a porous medium. The SF may enter to the cartilage area by pressure gradient. This model represents the effect of self-lubrication and nutrition of cartilage by SF.

An inlet pressure was set in the upper walls (wall 1 in Figure 5) with a value of $\mathbf{P}_0 = 0$ Pa , this boundary is one of the limit regions of both SF and the cartilage. Figure 8 shows the application direction of the input pressure in the system, which is consistent with the model proposed by Jurczak [1].

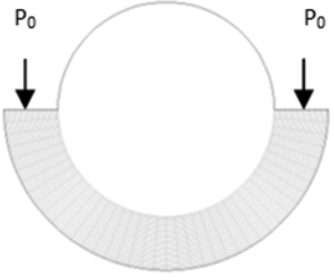


Figure 8. Direction of the inlet pressure P0 applied in the model.

The Table I shows the values of the parameters associated with the system considered in the simulations in this work.

Table I. Numerical values of the parameters used in the simulations.

PARAMETERS		NUMERICAL VALUES
Viscosity	μ	μa = 1.003x10^{-3} kg/m.s (water viscosity)
		10 μa
		100 μa
Maximum displacement	*e*	0.0025 m
Moving wall velocity	*V*	0.00125 m/s
		0.00125 m/s
		0.00416 m/s
Porosity	*K*	0.2
		0.8
Inlet pressure	P_0	0 Pa

3. Results

Once the boundary conditions were established and the simulations were developed, in Figure 9 was plotted the displacement of the moving wall. This figure shows the starting and the ending positions of the nodes associated with the moving wall.

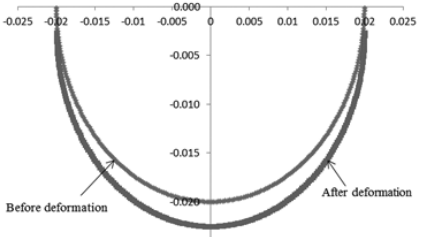


Figure 9. Graph of the moving wall when the maximum displacement is e = 0.0025 m.

Initially, Figures 10 and 11 show the results for the case when the maximum displacement of the moving wall is e = 0.0025 m, with a deformation rate of $\dot{\varepsilon} = 0.624\ s^{-1}$, the fluid viscosity is μ = 10 μa, the inlet pressure is P_0 = 0 Pa, and the porosity is *K* = 0.2.

In particular, Figure 10 shows the total pressure distribution in the system with its maximum pressure values in the area where the greatest deformation is performed, that corresponds to the load-application direction.

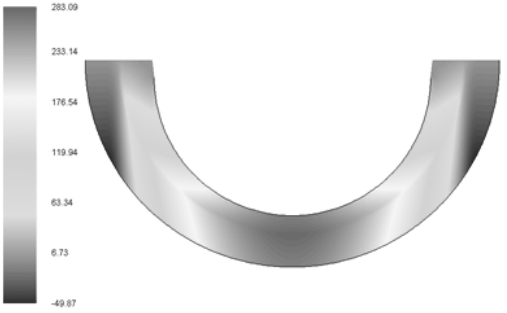


Figure 10. Graph of the total pressure in the system for the parameters e = 0.0025 m, $\dot{\varepsilon}$ = 0.624 s-1, μ = 10μa, P0= 0 Pa, and K = 0.2.

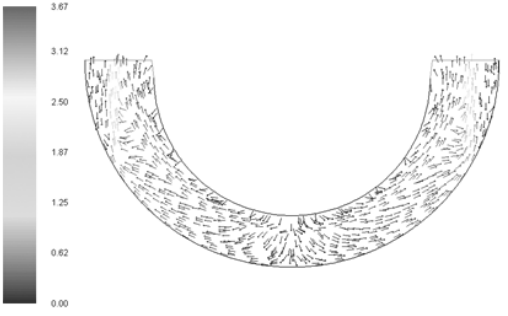


Figure 11. Graph of the velocity vectors distribution in the system for the parameters e = 0.0025 m, $\dot{\varepsilon}$ = 0.624 s-1, μ = 10 μa, P0 = 0 Pa, and K = 0.2.

Figure 12 shows the results for the dynamic pressure distribution of the system. This figure shows the dynamic pressure values for the final state when the full compression in the fluid region is reached. Figure 12.a corresponds to

the case of a deformation rate of $\dot{\varepsilon} = 0.624$ s-1. This figure shows the maximum dynamic pressure recorded on both ends of the SF and cartilage regions when the deformation rate is low. Figure 12.b corresponds to the case of a deformation rate of $\dot{\varepsilon} = 3.333$ s-1. In that case the maximum dynamic pressure is close to the maximum compression in the SF zone and the scale shows higher dynamic pressure values for a high deformation rate.

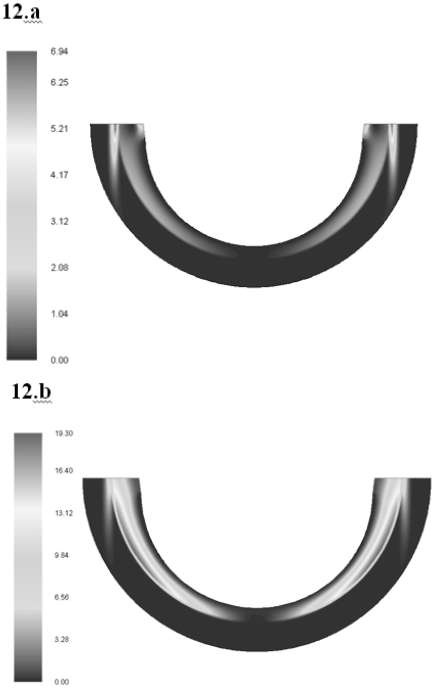


Figure 12. Graph of the dynamic pressure in the system for the parameters e = 0.0025 m, μ = 10 μa, and P0 = 0 Pa, when the deformation rate is a) $\dot{\varepsilon} = 0.624\ s^{-1}$, b) $\dot{\varepsilon} = 3.333\ s^{-1}$.

In the Table II the maximum values for the dynamic and total pressure in the entire system and at the interface have been recorded. The maximum values for the velocity vectors magnitude in different areas of the system as the SF and cartilage zones are also shown. These results correspond to the case where the maximum displacement of the moving wall is e = 0.0025 m, with a deformation rate of $\dot{\varepsilon} = 3.333\ s^{-1}$, the fluid viscosity is μ = 10 μa, the inlet pressure is P0 = 0 Pa, for two different values of porosity K= 0.2 and 0.8. Comparing the values obtained in Table II for the cartilage zone, it can be established that the maximum values of both the dynamic pressure and the fluid velocity magnitude increase when the degree of cartilage porosity decreases. If a system has low porosity the fluid is less likely to move to the porous medium and then the fluid increases the dynamic pressure and its velocities as the deformation is developed.

Table II. Maximum values obtained for the pressure and velocity when the porosity is varied

Variable	Total pressure (Pa)	Dynamic pressure (Pa)		Velocity magnitude (m/s)	
K	*Interface*	*SF zone*	*Cartilage zone*	*SF zone*	*Cartilage zone*
0.2	192.83	13.078	6.835	0.162	3.341
0.8	192.60	13.267	1.509	0.163	1.570

Figure 13 shows the case when the fluid viscosity is varied. The graph in Figure 13 shows the behavior of the calculated total pressure in fluid-cartilage interface. Pressure values are presented as a function of polar angle ϕ; the angle is measured from the negative Y-axis to the positive X-axis until reach the value of 90 degrees that correspond to the half of the system. By symmetry the results are

only displayed from 0 to 90 deg. In this figure it can be seen that higher values in viscosity results in a higher resistance of the system to be deformed.

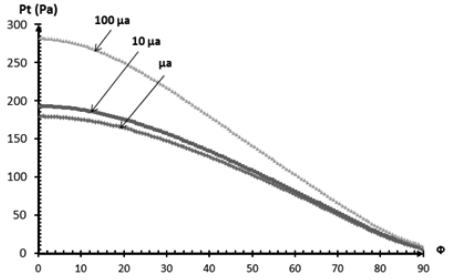


Figure 13. Plot of the total pressure distribution at the interface for the parameters e = 0.0025 m, $\dot{\varepsilon}$ = 3.333 s^{-1} , K= 0.2, P0 = 0 Pa, when the viscosity take the values of: μ = μa, 10μa, y 100μa.

In the Table III the maximum values for the dynamic and total pressure in the entire system and at the interface have been recorded. The maximum values for the velocity vectors magnitude in different areas of the system as the SF and cartilage zones are also shown. In this Table III it can be observed the values obtained when varying the viscosity of the fluid when the rest of the parameters are remained as constants. These results correspond to the case where the maximum displacement of the moving wall is e = 0.0025 m, with a deformation rate of $\dot{\varepsilon}$ = 3.333 s^{-1}, a porosity of K = 0.2 and an inlet pressure of P0 = 0 Pa. The different values for the fluid viscosity were based on the water viscosity, that is μ = μa, 10 μa, and 100 μa.

Table III. Maximum values obtained for the pressure and velocity when the fluid viscosity is varied

Variable	Total pressure (Pa)		Dynamic pressure (Pa)	Velocity magnitude (m/s)	
μ	*Inter-face*	*System*	*SF zone*	*SF zone*	*Cartilage zone*
μa	179.47	179.35	10.139	0.143	3.342
10 μa	192.83	192.84	13.078	0.162	3.341
100 μa	281.88	283.01	19.625	0.198	3.634

The maximum values for the total pressure were obtained when the fluid viscosity is higher and increase the resistance of the system for to be deformed. Moreover, the greater values for the velocity vector magnitudes were obtained when the fluid had less resistance to move that is for low viscosities, for the same reason the dynamic pressure peak values obtained at lower viscosity.

Finally, Figure 14 shows the results for the total pressure distribution at the interface when the deformation rate values are $\dot{\varepsilon}$ = 0.624 s^{-1} , 1.248 s^{-1}, and 3.333 s^{-1}; corresponding to the moving wall velocities of 0.00125 m/s, 0.001666 m/s, and 0.0025 m/s respectively. The results were obtained when the maximum displacement of the moving wall is e = 0.0025 m, the viscosity of the fluid is μ = 10 μa, a porosity of K = 0.2, and the inlet pressure is P_0 = 0 Pa. Figure 14 shows that as larger is the deformation rate greater is the system resistance for deformation.

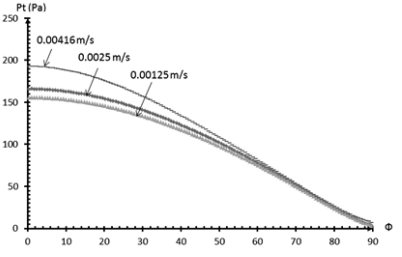


Figure 14. Plot of the total pressure distribution at the interface for the parameters e = 0.0025 m, μ = 10μa, K = 0.2, P_0 = 0 Pa, when the moving wall velocities take the values of V = 0.00125 m/s, 0.0025m/s, and 0.00416 m/s.

The Table IV shows the maximum values for the dynamic and total pressure in the entire system and at the interface. The maximum values for the velocity vectors magnitude in the areas of SF and the porous medium are also shown. These results correspond to the case when the deformation rate in the region of the fluid is varied.

Table IV. Maximum values obtained for the pressure and velocity when the deformation rate is varied.

Variable		Total pressure (Pa)		Dynamic pressure (Pa)	Velocity magnitude (m/s)	
$\dot{\varepsilon}(s^{-1})$	V (m/s)	*Interface*	*System*	*SF zone*	*SF zone*	*Cartilage zone*
0.624	0.00125	156.052	156.049	1.513	0.055	3.348
1.248	0.00250	166.080	166.018	2.557	0.072	3.307
3.333	0.00416	192.828	192.836	13.078	0.162	3.341

Remaining as constants the maximum displacement of the moving wall as e = 0.0025 m, the viscosity of the fluid is μ = 10 μa, the porosity is K = 0.2, and the inlet pressure is $\mathbf{P}_0$ = 0 Pa.

The results obtained show that the maximum values for the total pressure increase as the deformation rate increases. Similarly higher values are obtained in both the dynamic pressure and the fluid velocity at the SF zone as the deformation rate increases.

Conclusions

The simulations presented in this work have allowed modeling the behavior of the synovial cavity where cartilage and synovial fluid are in contact and they are subject to different compressive forces. Cartilage has been modeled as a porous medium, which allows the entry of synovial fluid. The implementation of Dynamic Mesh technique, through a deformable zone and a moving wall, it has allowed to control the deformation in the fluid area and then the displacement of fluid to the porous medium. The use of Computational Fluid Dynamics technique allows generating and solving models with more complex geometries and close to the synovial joint. In the case of the geometry proposed by Jurczak, the mean computational time was of 45 minutes when the mesh density was 9757 nodes.

The main results obtained from the simulations show that at higher viscosities, the system can support higher pressures. That result would help to estimate the maximum loads in a healthy high viscosity synovial fluid and compare with those who have a low viscosity because of some diseases such as osteoarthritis. From the results can also be identified the regions with higher total pressure in the system, which corresponds to the region with greater wear more frequently.

One of the limitations found in the theoretical models in the literature consulted, in such models they are limited to low porosity ($K << 1$) and the pressure distributions reported are only in the fluid region. This simulation model can provide results for a wide range in the porosity parameter and also the calculus can be carried on the porous medium and the interfacial boundary.

For future work it will be reported a new model in which the deformation region will be the cartilage zone, then the reverse process of exudation let to leave the Synovial Fluid from cartilage by compression.

Nomenclature

φ	angular coordinate
$\dot{\varepsilon}$	deformation rate
ε	dimensionless displacement
ρ	fluid density
$\mathbf{P_0}$	inlet pressure
e	maximum moving wall displacement
K	porosity
p	pressure
C	thickness of the initial fluid zone
V	velocity of the moving wall
$\boldsymbol{\mu}$	viscosity

References

[1] P. Jurczak, "Pressure distribution in a squeeze film biobearing lubricated by a synovial fluid", International of Applied Mechanics and Engineering, 11(4), 2006, pp. 857-864.

[2] A. Ruggiero, E. Gómez, & R. D'Amato, "Approximate closed-form solution of the synovial fluid film force in the human ankle joint with non-Newtonian lubricant", Tribology International, vol. 57, 2013, pp. 156-161.

[3] M.R. Brinker, "Joints. Section 2 Review of Orthopedics", 3rd. edition, M. D. Miller & M. R. Brinker (editors), W. B. Sauders, 2000, pp. 40-48.

[4] Dufour, & M. Pillu, "Biomecánica Funcional", Editorial Masson, S.A., Barcelona, España, 2006, pp. 450-465.

[5] G.A. Ateshian, & C.T. Hung, "The natural synovial joint: Properties of cartilage", Proc. Imeche, Part J: Journal of Engineering Tribology, vol. 220 (8), 2006, pp. 657-670.

[6] A. Sarma, & T.S Ravigururajan, "Analysis of Fluid Therapy during Arthritis", Proceedings 1st. Annual Symposium: Graduate Research and Scholarly Projects (GRASP), Wichita State University, 2005, pp. 27-28.

[7] J. Katta, Z. Jin, E. Ingham, & J. Fisher, "Biotribology of articular cartilage - A review of the recent advances", Medical Engineering and Physics, vol. 30, 2008, pp. 1349-1363.

[8] P. Jurczak, A. Walicka, E. Walicki, & D. Michalski, "Influence of rheological parameters on the mechanical parameters of curvilinear thrust bearing with one porous wall lubricated by a couple stress fluids", Int. J. of Applied Mechanics and Engineering, vol. 11(2), 2006, pp. 221-233.

[9] C.M. Corvalan, & J. Di Paolo, "Un simple modelo unidimensional de junta sinovial con exudación", Mecánica Computacional, vol. XVI, 1996, pp. 74-82.

[10] N. S. Landinez, J.C. Vanegas, & D.A. Garzón, "Modelado matemático del comportamiento mecánico de un fragmento de cartílago articular", Dyna, vol. 76 (157), 2009, pp. 133-144.

[11] J. Di Paolo, & E. Berli, "Análisis por elementos finitos de un modelado de lubricación 2D para una prótesis de cadera", Mecánica Computacional, vol. XXI, 2002, pp. 2440-2455.

[12] M.E. Berli, D.M. Campana, S. Ubal, & J. Di

Paolo, “Predicciones numéricas de una prótesis total de rodilla con elemento tibial deformable y microporoso”, Mecánica Computacional, vol. XXIX, 2010, pp. 6415-6431.

[13] K. Dumont, J.M.A. Stijnen, J. Vierendeels, F.N. van de Vosse & P.R. Verdonck, “Validation of a Fluid-Structure Interaction Model of a Heart Valve using the Dynamic Mesh Method in Fluent”, Computer Methods in Biomechanics and Biomedical Engineering, vol. 7 (3), 2004, pp. 139-146.

[14] N.J. Walkington, J.F. Antaki, G.E. Blelloch, O. Ghattas, I. Melcevic & G.L. Miller, “A parallel dynamic-mesh Lagrangian method for simulation of flows with dynamic interfaces”, Supercomputing ’00 Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Article 26, 2000.

[15] Guía de Usuario: Ansys Fluent.

[16] F.J. Gálvez, R. López, A. Llopis & C. Rubio, “Física curso teórico práctico de fundamentos físicos de la ingeniería”, ed. Tebar Flores, 2000, pp. 124.

A Neighbour Search Code for Galaxies-Preliminary Results

René A. Ortega-Minakata[1], Juan P. Torres-Papaqui1, Heinz Andernach[1]
[1] Departamento de Astronomía, Universidad de Guanajuato
{rene, papaqui, heinz}@astro.ugto.mx

Abstract

We present preliminary results of applying a neighbour search code to a large sample of galaxies drawn from the Sloan Digitial Sky Survey (SDSS). We draw our sample of target galaxies from the spectroscopic catalogue of the SDSS, which has redshift measurements for all its ~ 8 x 105 galaxies. We count the neighbours from the photometric catalogue of the SDSS, which is a flux-limited catalogue that has a photometric redshift estimation for all its ~ 2 x 108 galaxies.

The code counts the number (N) of galaxies within a given 3D-distance from the target galaxy, using a database manager and a simple decision tree. The 3D-distance is composed of a 2D projected distance on the sky and a radial component, which is the difference between the spectroscopic target redshift and the photometric redshift of the "neighbour" galaxies. As we keep the search volume constant throughout the sample, N is a proxy for the number density of galaxies around the target galaxies.

We also present an analysis of the preliminary results, showing the relation between N, the inferred morphology and the dominant activity type of the target galaxies, as well as the relation between N and the star formation history of the target galaxies.

Keywords: High Performance Computing: applications.

1. Introduction

In our project, we wish to quantify the number density of galaxies in three-dimensional space to investigate whether some statistical properties of galaxies, specifically their type of dominant activity, morphology and star formation history, are related to this density. This requires a neighbour search algorithm applied to a large sample of galaxies, which we call the target sample, and a very large sample of galaxies to search the *neighbours* from, referred to as the *neighbours* sample.

The density of the environment of a galaxy is difficult to define, as the best measurement of environment depends on the particular goals of each study (for a detailed discussion, see [1]). It is usually defined as the number of galaxies around a particular galaxy or in a given area of the sky. Different particular definitions yield results with small variations (for examples see section 5 in [2]), so an adequate definition of environment is crucial to achieve our goal. We have chosen a parameter represented as N_R^{-19}, based on a similar measurement by [3], which

is the number of galaxies with absolute r-band magnitude M(r) less than or equal to -19 within a physical projected search radius R of a target galaxy.

2. The Algorithm

Basically, our algorithm counts the number (N) of galaxies within a certain search radius in three-dimensional space of a target galaxy. The relevant information of the galaxies in both the target and the neighbours samples is their position in three-dimensional space, which is given in spherical coordinates by the angular coordinates in the plane of the sky (RA and Dec), and a coordinate along the radial direction, the redshift z.

The search radius in three-dimensional space can be decomposed into a component in the plane of the sky, equivalent to a physical projected search radius, R, and a radial component, equivalent to a difference in z. This decomposition yields a truncated conical search region rather than a spherical one which is adequate for the purpose of this code. It is also worth noting that we choose the physical size R as constant for all the target galaxies, so when expressed as an angular distance θR in the plane of the sky, θ_R decreases with the redshift of the target galaxy. This also implies that the search volume remains constant, so the number of galaxies within any such *volume* is directly proportional to the density of galaxies in that region.

We chose to draw our input samples from the Sloan Digital Sky Survey Data Release 7 (SDSS-DR7) [4]:

- The target sample consists of ~ 8 x 10^5 galaxies from the spectroscopic catalogue. All galaxies in this sample have measured redshifts.
- The neighbours sample comprises ~ 2 x 10^8 galaxies from the photometric catalogue. This is reduced after applying relevant selection criteria, mainly the aforementioned limit in the absolute magnitude in the r band M(r) less than or equal to -19, the sole application of which yields ~ 1.14 x 10^8 galaxies. It is important to note that we only have estimated photometric redshifts for all these galaxies. For a reference set of galaxies with spectroscopic redshifts, the rms error of the redshift estimation is 0.025 [4].

The basic algorithm consists of two nested loops. The outer loop sweeps through all the galaxies in the target sample, reads the relevant data for each of them and calculates the range in redshift and projected angular search radius θR in which neighbours need to be searched. The inner loop sweeps through the galaxies in the neighbours sample, reading the relevant data for each of them; for those within the current target's range in redshift, it calculates the projected angular distance from the target; it then counts all galaxies with projected angular distance within θ_R. Finally, the outer loop writes the final count for each target galaxy.

We coded a working version of this basic algorithm in the Perl Data *Language, PDL* (see pdl.perl.org). Since this basic algorithm would take a very long time to run using the full samples described above, we needed to find optimization strategies to reduce the run time of the code.

Some of these strategies are:

- Parallelization: both loop structures

are suitable for parallelization, since results of each comparison and also of a previous target are independent of any previous comparison or target.

• Relevant region search: we do not need to test all objects in the neighbours sample but only those within a relevant region near the target; we need to skip to the next target after searching this region.

So far we have only pursued a relevant-region search strategy, taking advantage of Perl modules that connect with database managers; such as PostgreSQL (see www.postgresql.org). We coded a version of the algorithm that queries the neighbours sample, pre-loaded into a PostgreSQL database, for galaxies within the relevant redshift range and inside a "square" projected into the plane of the sky with dimension just large enough to contain a circle of radius R around the target galaxy.

This last version of the code was tested against the basic algorithm with no optimization strategy, finding that the optimized code was ~ 65 times faster. A parallelization strategy has yet to be implemented, but a preliminary analysis that makes use of the results of the test described in the next section suggests that by using a process manager to run different regions of the sky simultaneously would be enough to run the code for the full samples mentioned above in a reasonable time (see also the last paragraph of the next section).

3. Testing the Algorithm.

Using the optimized version, we tested the algorithm for a region of sky between +20° and +30° of Declination and 170° and 180° of Right Ascension. This region of the sky comprises 10,860 galaxies from the SDSS-DR7 spectroscopic catalogue, making the target sample, and 814,704 galaxies from the SDSS-DR7 photometric catalogue with M(r) less or equal than -19, making the neighbours sample. We present here basic statistics regarding this test.

We ran the test using a physical projected search radius R = 1.5 Mpc, a radial velocity ($c \cdot z$) range of plus or minus 2500 km/s from the target redshift, and a value of the Hubble constant $H_0 = 75\,km \cdot s\text{-}1 \cdot Mpc\text{-}1$, which measures the rate of the expansion of the Universe. The median value of the number of neighbouring galaxies was $N_R\text{-}^{19} = 9$. In Figures 1 and 2 we present, respectively, the histogram of $N_R\text{-}^{19}$ and the distribution of $N_R\text{-}^{19}$ with the redshift z of the target galaxies. For comparison, we present the histograms of the redshift z of both the neighbours and target galaxies in Figure 3. Figure 4 displays the distribution on the plane of the sky of all target galaxies, coloured according to their value of $N_R\text{-}^{19}$.

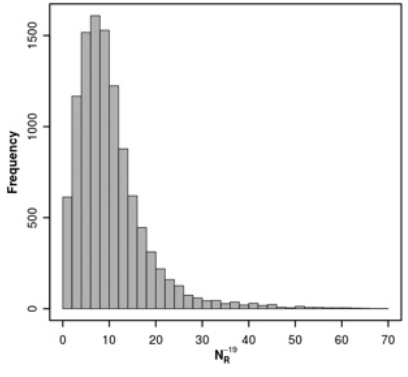


Figure 1. Histogram of $N_R\text{-}^{19}$.

While making this test, we checked for possible inconsistencies that would indicate

a failure of the algorithm, such as the mean photometric redshift of the galaxies returned by the SQL query being very different from the target galaxy redshift, or the number of neighbours being higher than the number of galaxies returned by the SQL query. We found none of these inconsistencies, suggesting the code works appropriately.

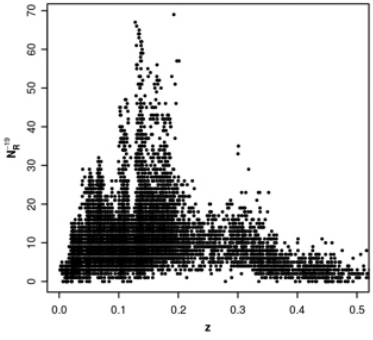


Figure 2. Distribution of NR-19 with the redshift z of the target galaxies, out to z = 0.5 .

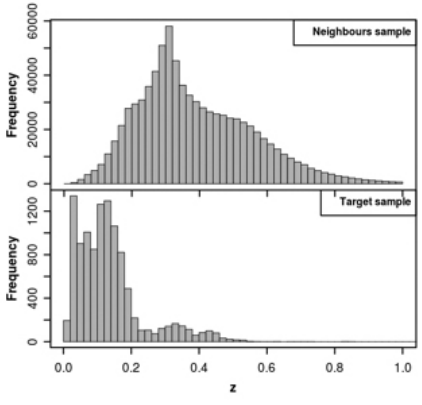


Figure 3. Histogram of redshift z of the neighbours sample (upper panel) and of the target sample (lower panel).

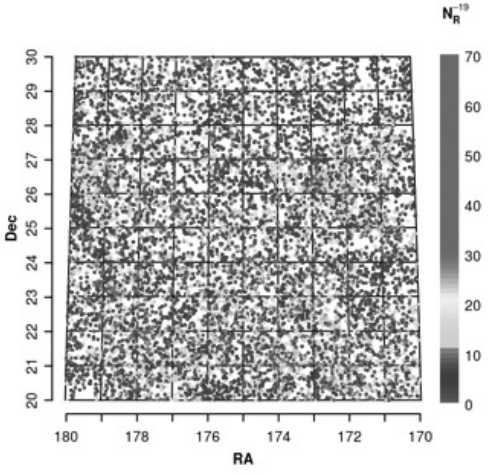


Figure 4. Aitoff projection of the distribution on the plane of the sky of all target galaxies, coloured using rainbow colours according to their value of N_R^{-19}: N_R^{-19} = 0 is represented with magenta, running smoothly to N_R^{-19} greater or equal than 30, represented with red.

Proper comparison with similar measurements of galaxy environment in the literature is work in progress. Still, we can already conclude that the code as it is fulfills its purpose of calculating the number of galaxies within a given search radius. Parallelization of the code, which is feasible given its structure, can be achieved either by direct parallelization or by running the code for different regions of the sky simultaneously, and will allow us to run it on the above mentioned samples of the order of 800,000 target galaxies and 114 million galaxies in the neighbours sample, completing our original goal. Such a run would take approximately 24 hours using 205 simultaneous processors.

4. Testing our relations between activity type, morphology, star formation history and environment using our measurement of environment

Using the results of the test described in the previous section, we explored the relations between dominant activity type, morphology, star formation history (SFH) and environment of galaxies that we had found previously (see [5]), but now using our measurement N_R^{-19}. In general, galaxies may show emission-line activity that can be dominated by either an active galactic nucleus (AGN) or star formation (SFG), or the galaxy may be a composite "transition object" (TO), while the morphology of a galaxy can generally be defined as "early type" (elliptical and lenticular galaxies) or "late type" (spiral and irregular galaxies). Since our target sample is too large for a visual inspection of the morphology, we use a morphological inference derived from photometric parameters of the galaxies. For detailed descriptions on how these characteristics of galaxies are defined and used here, see [5], [6], [7], [8] and references therein.

For this test, we have restricted ourselves to galaxies within the range 0.025 to 0.250 in z, in order to minimize aperture biases (lower constraint) and to avoid the need of a correction to the luminosity function due to the Malmquist bias (upper constraint). We considered seven bins of NR-19, as defined in Table 1. Note that we do not use NR-19 = 0, since this indicates a failure in the algorithm for that particular galaxy, because NR-19 does include the target galaxy. These failures are mostly due to larger uncertainties in the photometric redshifts of these galaxies. There are 8831 galaxies within the aforementioned redshift range, of which very few (13) have NR-19 = 0. This is also a good indicator that the code is working properly. The total number of galaxies used for our test is therefore 8818.

Table 1. Activity type distribution for different neighbourhood densities.

		Number of Galaxies				
Bin no.	N_R^{-19}	Total	SFG (%)	AGN (%)	TO (%)	Passive (%)
1	1 to 3	461	183 (39.7)	69 (15.0)	70 (15.2)	6 (1.3)
2	4 to 6	1536	536 (34.9)	275 (17.9)	225 (14.6)	29 (1.9)
3	7 to 9	2045	653 (31.9)	407 (19.9)	303 (14.8)	48 (2.3)
4	10 to 14	2537	692 (27.3)	551 (21.7)	334 (13.2)	80 (3.6)
5	15 to 19	1171	260 (22.2)	253 (21.6)	164 (14.0)	43 (3.7)
6	20 to 29	745	135 (18.1)	163 (21.9)	80 (19,7)	38 (5.1)
7	30 or more	323	36 (11.1)	72 (22.3)	30 (9.3)	15 (4.6)

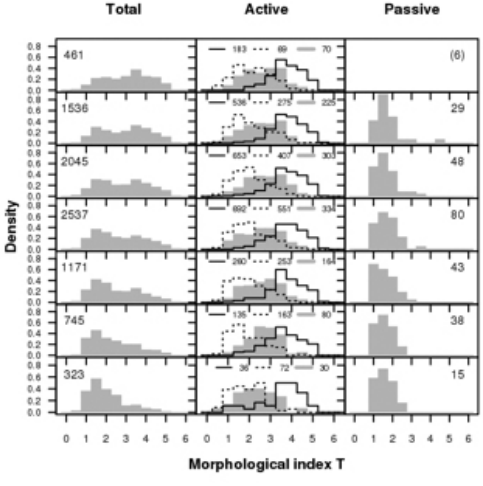


Figure 5. Morphological distribution of galaxies within the different bins of NR-19. Bin 1 is the top row, while Bin 7 is the bottom row; for the total number of galaxies in each bin (left column), separated by activity type (central column), and for passive galaxies (right column). In the central column, the solid line histograms corresponds to SFG, the dotted line histograms to AGN and the gray-shaded histograms to TO. The sizes of the samples are indicated. Areas below all histograms are equal to unity. The top right panel is omitted because of the very small amount of passive galaxies in Bin 1 (only 6).

The results of this test can be seen in Figures 5 and 6. In Figure 5 we show the morphological distribution of galaxies within the different bins of N_R^{-19}, in general and separated by dominant activity type. The left column clearly shows the morphology-local density relation [9], as the fraction of early-type galaxies clearly increases from lower density (top) to higher density environments (bottom). The central column shows the active galaxies by type: AGN-dominated (AGN), Star Formation-dominated galaxies (SFG) and Transition Objects (TO). The morphological separation between these types of dominant activity, seen previously using ACO clusters [10] and isolated galaxies [11], holds independently of N_R^{-19}, which means that there is a clear relation between the inferred morphology and the dominant type of activity of the galaxies.

In combination with Table 1, we can see how the fraction of SFGs decreases sharply with increasing
N_R^{-19}, which can be a consequence of the morphology-density relation combined with the morphology-activity relation we have confirmed. In the right column of Figure 5 we can see that passive galaxies are always of early morphological type, also independently of N_R^{-19}, and that their fraction also increases with increasing environmental density.

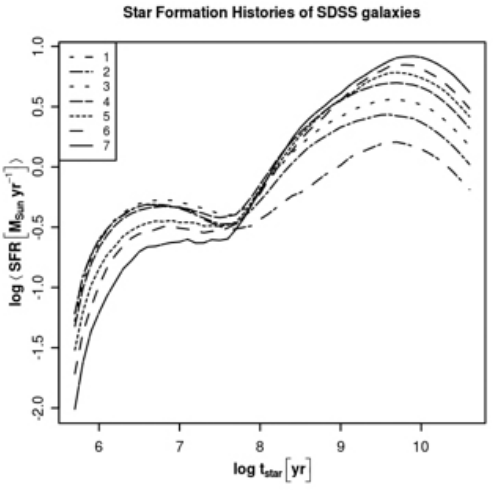


Figure 6. Median star formation histories (SFHs) of galaxies within the different bins of N_R^{-19}. The curves of SFHs are formed by relating the star formation rate (SFR) needed at a given lookback time t_{star} to form the current stellar populations of each galaxy, so a higher value of SFR at a given t_{star} value means that a higher fraction of stars with age tstar is present in that galaxy.

In Figure 6 we show the median SFHs of galaxies within the different bins of N_R^{-19}. A very clear sequence of SFHs can be distinguished, ranging from the denser to the

less dense environments: denser environments have higher first peaks of star formation (higher fraction of older stellar populations) and lower recent peaks (lower fraction of younger stellar populations), while less dense environments have lower first peaks and higher recent peaks of star formation. This is also in agreement with previous results using extreme environments (rich ACO clusters and isolated galaxies), confirming that there is an actual sequence of SFHs from low-density environments to high-density environments.

4. Conclusions

We have presented preliminary results of applying a neighbour search code to a large sample of galaxies drawn from the SDSS.

The code counts the number (N) of galaxies at a given 3D-distance from the target galaxy, using a database manager and a simple decision tree. The 3D-distance is composed of a 2D projected distance on the sky and a radial component, which is the difference between the spectroscopic target redshift and the photometric redshift of the neighbour galaxies.

We have described how this code works and tested it using a sample of 10,860 target galaxies and 814,704 galaxies in the neighbours sample, drawn from the SDSS-DR7, showing that the code fulfills its goal and can be run on the full sample in reasonable execution time.

We also presented an analysis of the preliminary results, showing the relation between N, the inferred morphology and the dominant activity type of the target galaxies, as well as the relation between N and the star formation history of the target galaxies. Our environment measurement is able to reproduce the morphology-local density relation, and is independent of the morphology-activity relation. We also show a clear sequence of SFH with N, showing that galaxies in denser environments tend to be older than galaxies in less dense environments.

5. References

[1] S.I. Muldrew, D.J. Croton, R.A. Skibba et al., "Measures of galaxy environment – I. What is 'environment'?, 2012, *Monthly Notices of the Royal Astronomical Society*, 419, 2670.

[2] B.V. Komberg, and I.N. Plashchenko, "Giant radio galaxies: Old long-lived quasars?", 2009, *Astronomy Reports*, 53, 12, 1086.

[3] J.D. Wing, and E.L. Blanton, "Galaxy cluster environments of radio sources", 2011, *The Astronomical Journal*, 141, 88.

[4] K.N. Abazajian, J.K. Adelman-McCarthy, M.A. Agüeros, et al., "The seventh data release of the Sloan Digital Sky Survey", 2009, *The Astrophysical Journal Supplement Series*, 182, 543.

[5] R.A. Ortega-Minakata, J.P. Torres-Papaqui, H. Andernach, J.M. Islas-Islas, and D.M. Neri-Larios, "Relation between activity, morphology and environment for a large sample of SDSS galaxies", 2011, *Revista Mexicana de Astronomía y Astrofísica (Serie de Conferencias*), 40, 127.

[6] R. Coziol, J.P. Torres-Papaqui, I. Plauchu-Frayn, J.M. Islas-Islas, R.A. Ortega-Minakata, D.M. Neri-Larios and H. Andernach, "The nature and origin of narrow line AGN activity in a sample of isolated SDSS galaxies", 2011, *Revista Mexicana de Astronomía y Astrofísica*, 47, 361.

[7] J.P. Torres-Papaqui, R. Coziol, R.A. Ortega-Minakata and D.M. Neri-Larios, "The star formation history and chemical evolution of star-forming galaxies in the nearby Universe", 2012, *The Astrophysical Journal*, 754, 144.

[8] R. Cid Fernandes, A. Mateus, L. Sodré, Jr., G. Stasińska, and J.M. Gomes, "Semi-empirical analysis of Sloan Digital Sky Survey galaxies - I. Spectral synthesis method", 2005, *Monthly Notices of the Royal Astronomical Society*, 358, 363.

[9] A. Dressler, "Galaxy morphology in rich clusters: Implications for the formation and evolution of galaxies", 1980, *The Astrophysical Journal*, 236, 351.

[10] G.O. Abell, H.G Corwin Jr., and R.P. Olowin, "A catalog of rich clusters of galaxies", 1989, *The Astrophysical Journal Supplement Series*, 70, 1.

[11] V.E. Karachentseva, S.N. Mitronova, O.V. Melnyk, and I.D. Karachentsev, "Catalog of isolated galaxies selected from the 2MASS survey", 2010, *Astrophysical Bulletin*, 65, 1.

Assessing the Portability of JavaFX-Based Rich Internet Applications

Ing. Juan Carlos González Córdoba
Instituto Tecnológico Superior de Irapuato
Carretera. Irapuato-Silao Km. 12.5
e-mail: jcgc88@gmail.com

Abstract

It has been claimed that JavaFX provides a powerful Java-based UI platform capable of handling large-scale data-driven business applications. JavaFX applications are completely developed in Java, one of the most widely deployed technologies with one of the largest developer communities in the world, while leveraging the power of standards-based programming practices and design patterns. JavaFX provides a rich set of UI controls, graphics and media API with high-performance hardware-accelerated graphics and media engines to simplify development of immersive visual applications. In the present work we try to answer the question: How does applications developed with JavaFX perform on mobile devices such as tablets and phones running on the Android platform, in addition to computers so called compatible with JavaFX?. As a feature of this new technology, applications have the ability to run on mobile devices, unlike its main competitors in the market for "rich applications". In the later case, specialized requirements are needed in the form of different packages consistent with the framework used for development in each platform. One example is the Adobe Flash (AF) technology, where the user must install the AF Player compatible with the Web browser used on the client and the operating system used with that account [1]. Another example is a rich application developed with Microsoft technology, where the client must have the Silverlight package which in turn requires .NET and DirectX, in addition to having problems while trying to run on different devices.

Keywords: JavaFX, Platform, Web Application.

1. Introduction

A rich application is such that incorporates most of the desk. This involves more direct interaction with the operating system, and consequently better utilization of hardware resources [2].

The JavaFX rich applications need the JDK framework to be installed on the client machine. In recent years, web technology has seen a considerable progress since its inception, when one could only use plain text. Currently, applications include many more features such

as image deployment, colors and audio-visual design [3]. Moreover, the current trend of using social networks such as Messenger, Facebook, My Space among others, causes new problems and demands of social life. In general, Internet users are unhappy with applications displaying only text and images [4].

The JavaFX Runtime is installed with the Java Runtime Environment, ensuring its availability on more than 97% of enterprise desktops worldwide [5]. With JavaFX, applications can be developed for Web browsers running on the desktop, with the advantage of being able to make use of thousands of existing Java libraries [6], seamlessly mix and match native Java capabilities and the dynamic capabilities of web technologies in the applications, develop and maintain complex user interfaces easily, playback video and audio content in popular formats within the application, leverage modern graphics cards for optimal performance for applications featuring data visualizations and complex user interfaces, and use popular JVM-based scripting languages, such as Groovy, JRuby and Scala.

2. Development

The programs developed in JavaFX generate three different oriented applications to run on:

- Desktop computer equipment.
- Mobile devices.
- Internet.

An Integrated Development Environment (IDE) can be used to develop JavaFX applications. One option is the programming tool NetBeans IDE 7.2. NetBeans IDE provides comprehensive support for all Java technologies and latest enhancements. It provides support for JDK 7, Java EE 6, and JavaFX 2.0 at the time of this writing. After creating a JavaFX project in NetBeans and writing the code necessary to build applications, one can generate a folder with three applications through the Run menu by selecting the option to generate the project : a web page, the Java application that makes use of the web page and a java application that runs directly on a computer. Figure 1 shows the applications generated by NetBeans.

Figure 1 Applications generated for different platforms.

To test the performance of the programs in these three different platforms, we used the benchmark technique adapting the project stages. This benchmark methodology is aimed at testing hardware and software, divided into four phases described below:

Phase 1: Define the technology to be evaluated, in our case JavaFX.

Phase 2: Variables evaluated. For the present work, the variables considered are:

Memory. - Determines how much memory is used by the application in the device and platform (only applies to tablets and phones given their intrinsic limitations).

Portability. - Applications have to run on different platforms and they must operate without the need to install additional packages or make upgrades to the frameworks on

different devices.

Functionality of the platform or device. - Devices or platforms must function properly when installing the application, ie, avoid freezing the screen, warrant that all processes and platform services or devices continue running correctly, etc.

Program operation. - The application should do the activities for which they were scheduled.

Phase 3: Quantification of the results. After obtaining the data set by the variables assessed, they should be represented using a defined metric specific to the problem at hand.

Phase 4: Analysis of the variables dataset. The evaluation is made based on the results obtained in the previous stage in order to determine how JavaFX technology performs in each of the devices and platforms.

In the current stage of the project, the following devices have been tested:

Ten desktop computers with different Operting Systems: three using Windows 7, one with Windows Vista, one with Windows XP, three with Linux Ubuntu 10.10, one with Linux Ubuntu 12.04 and one with Linux Ubuntu 10.10 server.

One tablet Galaxy Tab 2 with Android 4.0.3 operating system.

Four phones: Nokia Lumina 800 with Windows Mibile, Nokia 5130 XpressMusic Nokia OS operating system with Java supports, Samsung GT-S5330 with Samsung's proprietary operating system and Samsung Galaxy Mini 2 with Android OS 2.3.6.

The applications tested on each platform or devices are: a video player, an animated graphics file and an image viewer. These three applications were developed by us for testing. Figure 2 shows the video player interface, Figure 3 shows the interface of the animated graphic file and Figure 4 shows the image viewer.

Figure 2 Video player.

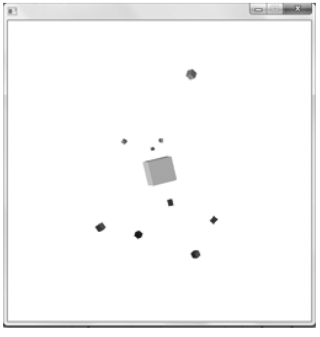


Figure 3 Program with animated graphics.

Figure 4 Images visualizer.

Assessing website performance

To test the performance of the generated web pages, the Apache 2 Web server was mounted on a local server with Ubuntu Server 10.10 operating system. Figure 5 shows the topology of the local network applied for testing.

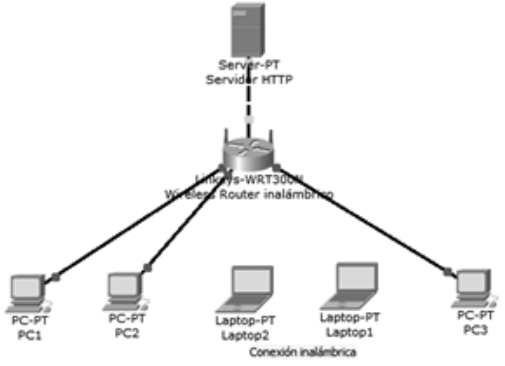


Figure 5 Local network test topology.

The applications performance on different devices is shown in Table 1. It can be seen that the tested applications show no apparent problems when ran on the Windows platform, however, they show some problems when ran in Ubuntu Linux platform and Android. Nonetheless, Windows XP had a failure in its platform when the web browser is closed the display freezes and after a while it closes and the desktop icons disappear momentarily. XpresMusic Nokia phones and Samsung GT-S5330 that support Java did allow the execution of the applications, but nothing is displayed, meaning that even though the programs run they do not work properly.

Table 1 Operating data from Web pages with applications made in JavaFX.

Plataforma	RAM (MB)	Se ejecutó correctamente	La plataforma tubo fallas	El programa funciono correctamente
Windows 7	3578	Si	No	Si
Windows Vista	2048	Si	No	Si
Windows XP	512	Si	Si	Si
Ubuntu 10.10	2048	No	No	No
Ubuntu 12.04	1024	No	No	No
Ubuntu Server 10.10	2998	No	No	No
Android	687	No	No	No
Nokia Lumina 800 with windows Movile	520	Si	No	Si
Nikia XpressMusic with Nikia OS	16	Si	No	No
Samsung GT-S5330 with Samsung OS	512	Si	No	No

Figure 6 shows the error reported by the Linux Ubuntu system.

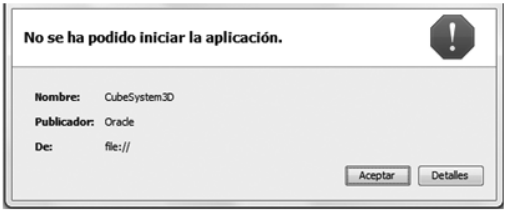


Figure 6 Error reported by the Linux platform.

Android platforms do not report any error but they do not execute any of the applications, and Web browser never finishes loading the Web page containing the application. Figure 7 shows the percentage of successful running applications in each platform.

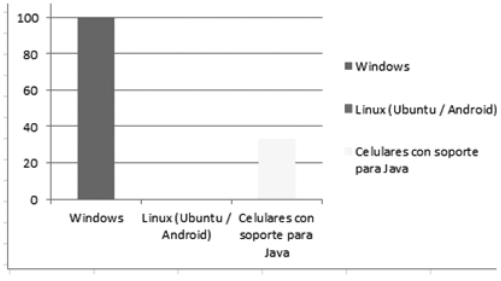


Figure 7 Percentage of Web pages successfully running on each platform.

Test applications built for mobile devices

To test mobile devices 4 phones and a tablet operating system whose characteristics were described previously were used. In order to install the applications on the Nokia phones, the Ovi software has to be used, while Android devices simply copy the application file to the device. Table 2 shows the applications performance on different mobile platforms on which they were tested. It can be seen that the Android platform does not support JavaFX technology, while the Nokia devices seem to be compatible with one exception; the XpressMusic model for which the applications do not work. The Samsung GT-S5330 model presents the same behavior as the Nokia Lumina model, the applications run but this not work.

Table 2 Operating data applications for mobile devices.

Plataforma	RAM (MB)	Se ejecutó correctamente	La plataforma tubo fallas	El programa funciono correctamente
Android 4.0.3	687	No	No	No
Android 2.3.6	687	No	No	No
Nokia Lumina 800 con windows Movile	520	Sí	No	Sí
Nikia XpressMusic con Nikia OS	16	Sí	No	No
Samsung GT-S5330 con Samsung OS	512	Sí	No	Sí

Android devices such as tablets and mobile phones cannot run JavaFX applications since they do not have Java support. Figure 8 shows the message reported by the Android platform when trying to run a JavaFX application.

Figure 8 Message reported by Android devices.

At the time of this writing, it was found that there are efforts to develop a Java emulator

for Android, which may allow execution of JavaFX applications on this platform [5]. The testing of these simulators is work in progress. Figure 9 shows the percentage of successful running applications for each platform.

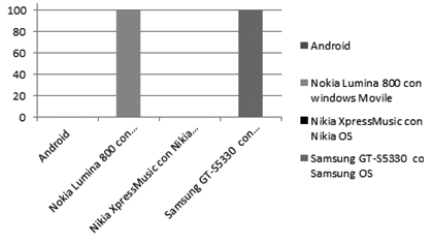


Figure 9 Percentage of operation for mobile applications.

Application testing on desktop computer equipment

These tests consisted on running the .jar file directly on the computer equipment for each of the applications. The set of ten systems previously described were used to run this test. Results are shown in Table 3. It can be seen again that the applications run on the Microsoft platform due to its compatibility with Java. In contrast, the applications did not run in the Ubuntu Linux platform. To run a Java application on Ubuntu we need to open a terminal and run the command java –jar "AplicationName.jar". This action generates an error wich is show in the figure 10, it can be seen that no displays information about the error causa.

Table 3 Performance of applications in a desktop computer.

Plataforma	RAM (MB)	Se ejecutó correctamente	La plataforma tubo fallas	El programa funciono correctamente
Windows 7	3578	Sí	No	Sí
Windows Vista	2048	Sí	No	Sí
Windows XP	512	Sí	No	Sí
Ubuntu 10.10	2048	No	No	No
Ubuntu 12.04	1024	No	No	No
Ubuntu Server 10.10	2998	No	No	No

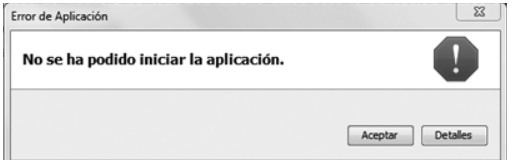


Figure 10 Linux notice given to the user when trying to run a JavaFX application.

Figure 11 shows the percentage of successful running applications for each platform.

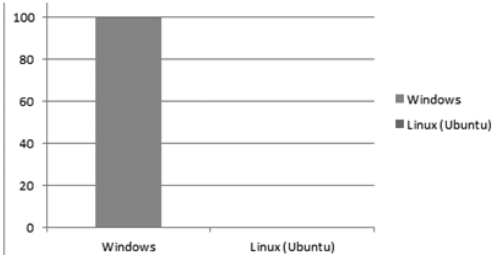


Figure 11 Percentage of application performance for computer equipment.

3. Conclusions

JavaFX is a young technology for rich application development aimed at different platforms. However, work needs to be done to extend its compatibility with mobile devices since it currently does not support the Android

platform. This is one of the platforms that dominate the market in the branch of mobile devices. With regard to the operation of the Web applications built in this work, here we show how they depend on the platform from which you access those applications. From all the platforms evaluated in this work, Windows is the only one with full support for JavaFX as either an application running on the computer equipment (.jar applications) or an application hosted on a web server. Data from this study show that although a program developed with JavaFX on NetBeans automatically generates three different executable files for the applications (the .jar for direct execution in a desktop computer system, a mobile application and a website) does not guarantee the successful operation of the program in all platforms. Therefore, the decision to use JavaFX for the development of cross-platform applications should be taken with care by the software developer.

4. References

[1] http://www.adobe.com/es/products/flashplayer/

[2]http://computomovil.files.wordpress.com/2009/10/elia-vite-y-j-armando-jaraleno.pdf

[3] http://www.oracle.com/technetwork/java/javafx/overview/index.html

[4] http://www.creaciondepaginaweb.net/informacion-pagina-web.htm

[5] Forrester, November 2009: Enterprise Platform Trends, H1 2009

[6] http://www.java.com/es/about/

[7] http://free-mobile-messenger.com/2011/02/24/jbed-java-emulator-for-android-1-5-2-2

Electroencephalographic High-Performance cloud data Processing System

M.Sc. Ricardo Ortega Magaña
Department of Information Systems
Universidad de Guadalajara, CUCEA
Guadalajara, México
ricardo.ortega@cucea.udg.mx
M. Sc. Javier M. Antelis
Instituto Tecnológico y de
Estudios Superiores de Monterrey
Guadalajara, México
antelis@unizar.es

Dr. Claudia Moreno González
Department of Mathematics
Universidad de Guadalajara, CUCEI
Guadalajara, México
claudia.moreno@cucei.udg.mx
Dr. Arturo Chavoya
Department of Information Systems
Universidad de Guadalajara, CUCEA
Guadalajara, Mexico
achavoya@cucea.udg.mx

Abstract

Analysis of electroencephalographic data involves processing a high number of signals acquired by many sensors (multivariable data). Usually, the processing of these data is performed in a low-cost computer where the analysis is sequentially performed for each dataset and sensor, thereby consuming high amount of time and machine resources to obtain those results. This work presents an on-line processing platform that performs those calculi on a computer cluster. This tool allows the user to upload to the cluster datasets acquired from EEG (Electroencephalographic) experiments and characterize them with their own attributes in the main database, so the user can share that information with other users, request common scientific data from other users, or process the datasets with the modules included in the platform. A comparison is made between the execution times of the same process on a standalone local computer and on two nodes of the cluster. The results are obtained and displayed in a reduced time when executed on the cluster without using the resources of the local machine to perform the processing.

Keywords—EEG; Databases; Cloud; HPC; Data processing

1. Introduction

Many problems in science and engineering involve processing and analysis of multidimensional time-varying signals. This characteristic is inherent to most of the neuroimaging techniques. For instance, functional magnetic resonance imaging (fMRI) and positron emission tomography (PET) provide multi-dimensional images of metabolic changes in thousands of voxels inside the brain, whereas electroencephalography (EEG) and magnetoencephalography (MEG)

techniques provide tens of measures with temporal resolutions of a few milliseconds. Thus, neuroimaging data have a spatiotemporal structure, i.e. the brain activity is sampled at multiple sensors at multiple time samples. In consequence, processing, analysis and even visualization usually require high computational capabilities.

Of all the above technologies, EEG has special interest in several research areas, as it offers a unique and direct access to the neural activation with the highest temporal resolution. Moreover, EEG is noninvasive, cheap and easy-to-use. This work focuses on the development of a high-performance computational tool for the execution of algorithms for analysis of EEG data.

There are several specialized tools focused on the processing and analysis of EEG data. The most used are EEGlab [1] and Fieldtrip[2], which are open source (GNU public licensed software for noncommercial use) MATLAB toolboxes for processing EEG traces. A list with other related open source software can be found in OpenEEG [3], a Sourceforge project with the aim of creating tools for the analysis of EEG Signals. Other commercial software applications are Net Station [4] from the EGI company, NeuroGuide [5], and the customized products of BrainProducts [6].

These packages all share the common characteristic of running over a local computer platform. However, the multidimensional nature of the EEG signals and the amount of samples and repetitions needed to deploy a trustworthy analysis lead to computing tasks that become prohibitively intensive for normal centralized computers [7]. Thus, migrating into a distributed environment and deploying new tools becomes a necessity [8]. As the problem itself keeps growing in complexity, the number of tools grows as well [2, 9].

Recent research of software development points to migrating into the cloud [10], which would allow a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources" [11]. In addition, this would resolve the issue of dealing with the need of learning configuration, usage and hardware requirements of all the tools, keeping a centralized source of computing power and a simple, yet complete set of tools to manage the information as needed.

By means of virtualization technologies, cloud computing offers end users a variety of services covering the entire computing spectrum, from the hardware to the application level, by charging a pay-per-use fee. Another important feature, from which scientists can benefit, is the ability to scale up and down the computing infrastructure according to the application requirements and the user's budget. By using cloud-based technologies, scientists can have easy access to large distributed infrastructures and completely customize their execution environment, thus providing the perfect setup for their experiments [12].

High performance cloud computing provides an ideal environment to run parallel applications, mostly due to the fact that they interconnect large amounts of computing resources and to their ability to transparently schedule and allocate the resources that are the most appropriate to run the application's processes. The cloud, utilizing virtualization techniques, allows resources to be harvested from physically distributed non-dedicated machines, contributing to the cost/benefit ratio [13].

Tools similar to the one proposed in the

present work are available in the web, such as "Code Analysis, Repository and Modeling for E-Neuroscience" (CARMEN), in which a virtual laboratory for neurophysiology (neural activity) recordings should be capable of processing and sharing data [14]. With this online platform it is possible to compile and process neural data, as well as sharing it to other researchers; at the time of writing this paper, only sharing data with other users was available. Another cloud computing system for EEG is described in [15] where they focus on a streaming calculation of EEG signals, instead of an a posteriori analysis.

Those projects were the main idea for the unification of the data source and data analysis in one place, not relaying on third-party software platforms, nor requiring a complex programming, signal processing background; instead, we propose using a common browser to manage the information and allow the data to be processed in the cloud. Working in the cloud has the option of uploading plugins for custom data management. When proprietary licenses are required, users can upload their own licensed software to work in that environment with the bought tools. From the user's perspective, this system gives access to the tools already familiar to them from any place with internet access, with the advantage of having a high performance server that processes the data remotely, and provides a share point to EEG scientific research groups.

2. EEG High Performance System

The high performance availability gives us the chance to build a user-friendly tool for the interested researcher in the field, with analysis and processing needs. The system has the following characteristics:

1) Online availability. We propose an online tool that allows different users to upload their EEG signals, characterize them with metadata, and share them to other users. We provide custom made tools to test the platform in order to analyze the data provided by the users in the cloud.

2) Database managed experiments. Data uploaded to the server needs to be characterized with metadata for sharing among users, and make custom filters for the data selection.

3) Remote processing. The computer intensive tasks are performed in the cloud, releasing the local computer from the computing load.

4) Data sharing. The data can be shared amopng users on demand, and can be accessed for processing, and catalogued by custom search filters.

5) Custom processing. The user has the option of uploading custom-made daemons to perform analysis for his or her data, or recover different data formats from the hardware equipment.

6) Results. Final results can be downloaded using daemons as different options, such as raw data, custom file formats, or images.

2.1. Server configuration

Hardware: The complete hardware set consists of three nodes, one Mac Pro Server with 6GB of DDR3 RAM and two quad core 2.80GHz Intel Xeon CPUs as head node, and two Hewlett Packard ProLiant Servers with a single quad core 2.33GHz Intel Xeon E5410 processor and 2 GB of RAM.

Software: The running operating system is Microsoft Windows Server 2012 and the different plugins that process the data were compiled with Microsoft Visual Studio 2012. To extend the capabilities of data processing, openMPI is available for the processes, and Cywin and virtual machines are optional for further data processing.

Interfaces: The database engine used to control de datasets of the different experiments of EEG is MySQL, and PHP is the main interpreter between the database engine, data processing, and the graphical user interface.

2.2. Cloud system conformation

The application described in this article uses a model view controller (MVC) paradigm in which the user performs multiple actions into data provided by himself or others users sharing their data. At the same time, it presents a platform (Figure 1) to organize and make custom filtering between experiments stored in the main database.

This platform is divided into three main parts: A database in which all the data is stored, data processing daemons that run all the tests, and a graphical user interface that interacts with the user.

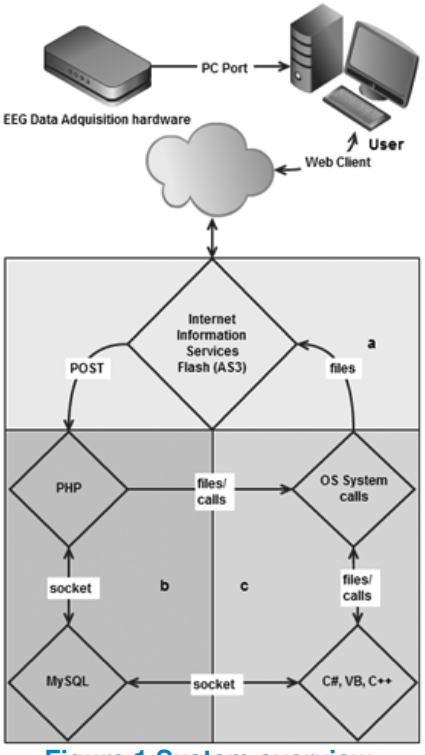


Figure 1 System overview.

Panel (a) in Figure 1 shows the graphical user interface that allows the user to interact with the tool. Panel (b) depicts the database, which allows the data to be managed in the cloud. Finally, panel (c) illustrates the daemons' block that is responsible for the processing tools.

A PHP script that gives permission to access the whole system processes every interaction between the user and the server. A more extensive description is presented next.

2.3. Graphical User Interface

An Adobe Flash interface for interacting with the server was developed with the main characteristic that the user needs three clicks at most to reach the task he or she wants to perform, without the need of reloading web pages, as Actionscript 3 allows the dynamic loading of the information from the server.

As for the main menu, seven sections allow the user complete interaction with the system. These sections are ordered sequentially by usage to facilitate the learning curve of the system by intuition. The sections are described next.

2.3.1. New Subject. Each data acquisition test has to be performed on a subject. Within this section, a new subject is inserted into the database for his/her future assignation to a test.

2.3.2. New experiment. For each test performed, there needs to be an experiment that holds the information of why those tests are related. This information is collected in this section. After a new experiment is done, new tests can be uploaded to the system.

2.3.3. New test. Each dataset that the user wants to upload needs to be associated with a test. This test should have a description and subject characteristics that make it distinct from the different tests assigned to the same experiment. Even the same tests performed on different subjects have to be described in this section.

2.3.4. File management. One of the basic functions of the web interface is the uploading of the RAW (unprocessed) data. This section allows the addition of these files to the database to be read and analyzed by the daemons tools described in Section 2.5.

2.3.5. Job status. Since all the workload will be processed within the cluster, a job scheduler was designed to check the status of each job. In this section the user monitors the progress of the processing assigned to those jobs.

2.3.6. Data selection. One of the most important goals in this work is the ability of selecting different datasets, not only those belonging to the user, but also the shared files from other users. In this section, the data selection can be narrowed using different filters in the queries to the database.

2.3.7. Job management. In this section the job scheduler makes the assignment of tasks to each node. The tasks to be performed are read directly from the database. This behavior allows the addition of new daemons by modifying the database tables, and by running a new daemon in any node in the network, instead of recompiling the main daemon.

After the graphical user interface sends the data to the web server and is recorded in the database, the job becomes visible to the daemons running in each node and allowing them to perform the task scheduled.
Each job has individual parameters, which are needed for each task to be performed and are added to the job parameter table.

2.4. Database

The database contains all the information on the data uploaded by the user. This allows the users to request tasks not only using their own

data, but also from data with specific attributes. The database is controlled by MySQL database manager, using PHP and MySQL Connector for .NET as clients, and it consists of 23 tables, described in the following six general sections.

2.4.1. Access control. The tables in this section control de access to the system, allowing to log in and secure the site by blocking hacking attempts. In general, these tables are only used for user data.

2.4.2. Jobs. All the job schedules, results, cluster task propagation, and prioritizing are managed by these tables. Every time a user sends a job to the platform, these tables store the data parameters and status for later analysis.

2.4.3. Experiments. Before parameterizing the jobs and submitting to the cluster, experiments need to be characterized with metadata and then indexed into the experiment catalog of each user, or user group. After that, each experiment might contain one or several tests to be performed in the platform; each test at the same time might contain several files and job assignments. All this data is stored in five experiment tables.

2.4.4. Characterization. These seven tables save most of the data filters for later search and experiment categorization.

2.4.5. Files. Each test might have none or several files on which to work; results from the job submission are saved as files as well. All this information is stored in two tables.

2.4.6. Groups. Two tables control the user grouping and visibility among them. These tables allow sharing the data among custom designed user groups, and they can also grant job permission for multiple submissions.

2.5. Daemons

Instead of an agent-based approach, daemons allow multiple site access and hardware requests without the need of additional system permissions. In the present work, all jobs are controlled by a main daemon with full access, which launches all the required processes as the user requests them, and has access to the entire system.

For the scheduling and core assignment of each task to be performed, the table “jobs” contains a node field that allows the daemon to dispatch the command in the homonym field. This allows creating the same job, with different parameters for each node, and optimizing the task within a distributed environment.

The modularity in the system allows new functions to be attached to the main daemon, no matter how many parameters are stored; as Visual Studio allows the communication among all its programming languages, they can be programmed in Visual Basic, C++, C, C#, J++ or J#.

As a second option to save the data from the tests in the database, the configuration of shared folders allows quick files to be stored, as needed, in case that the database manager keeps retrieving the same information. This method allows keeping a faster communication with the module asking for information.

After being processed, data gets stored in the database for further processing, or for downloading by the user.

The launching daemon that keeps pooling the database for jobs, launches not only binaries, but scripts as well, so full workflows can be performed by a single job submission. These scripts can be MATLAB files as well. This approach opens the door to other scientific

or mathematical implementations using the same base platform.

3. Results

As benchmark for the platform, a multivariate analysis of electrophysiological data was run in different architectures, using the Fieldtrip and Donders Machine Learning Toolbox [16]. This test allows statements to be made about the information content available in single trials experiments. As a result of this test, an image of the activation of the brain signals is stored into the database for further downloading. All the tests performed returned the image shown in Figure 2.

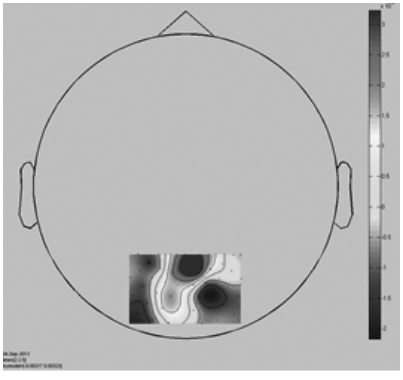


Figure 2. Example of a multivariate analysis benchmark result.

Figure 3 shows the data frequency of execution time for the test after 100 trials in a Toshiba Satellite A215, with an AMD Turion 64 X2 Mobile Technology TL-58 1.90GHz processor, 4GB DDR2 RAM and running 64-bit Windows 8.

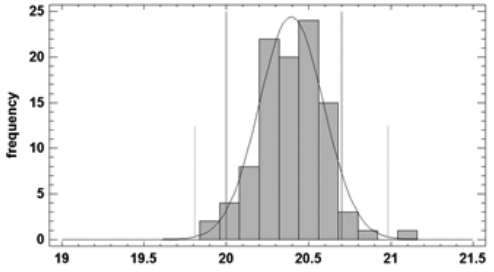


Figure 3.Toshiba Satellite execution time (in seconds).

This data was tested for normality with Chi-Square, Shapiro-Wilk W, Skewness Z-Score and Kurtosis Z-Score tests.

Table 1. Toshiba Satellite normality tests.

Test	Statistic	P-Value
Chi-Square	25.76	0.215748
Shapiro-Wilk W	0.980209	0.538121
Skewness Z-score	0.495646	0.620141
Kurtosis Z-score	1.83613	0.0663383

The same test was performed using the head node of the cluster; execution time results are shown in Figure 4.

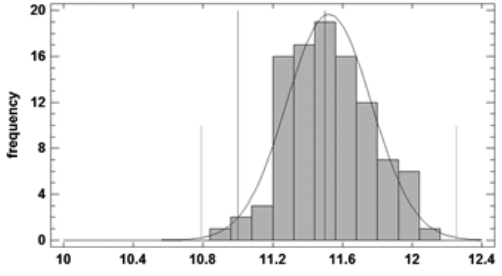


Figure 4. Mac Pro Server execution time (in seconds).

As before, a test was performed to test for normality; results are presented in Table 2.

Table 2. Mac Pro Server normality tests.

Test	Statistic	P-Value
Chi-Square	29.12	0.11115
Shapiro-Wilk W	0.982719	0.671507
Skewness Z-score	0.510966	0.609372
Kurtosis Z-score	-0.361546	0.717688

Since the cluster is not homogeneous, we tested the benchmark in the nominal computing nodes that have the same characteristics between them; execution time results are shown in Figure 5.

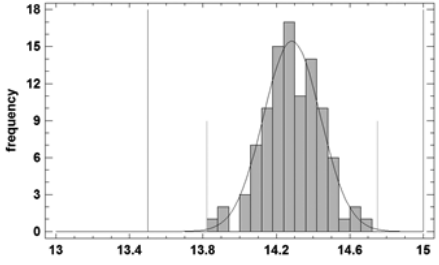


Figure 5.HP ProLiant execution time (in seconds).

Normality test was performed obtaining the scores presented in Table 3.

Table 3. HP ProLiant Server normality testsTest.

Test	Statistic	P-Value
Chi-Square	22.88	0.35041
Shapiro-Wilk W	0.98839	0.915279
Skewness Z-score	0.168071	0.866522
Kurtosis Z-score	0.548519	0.583332

Since the smallest P-value amongst the tests performed is greater than or equal to 0.05, we cannot reject the hypothesis that they came from a normal distribution at the 95% confidence level.

Table 4 shows a statistical comparison using ANOVA of the results of the normal distributions modeled by the execution times of the different architectures

Table 4. ANOVA test.

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
Between groups	4122.99	2	2061.49	50856.22	0.0000
Within groups	12.0391	297	0.0405357		
Total (Corr.)	4135.03	299			

Since the P-value of the F-test is less than 0.05, there is a statistically significant difference between the means of the three variables at the 95.0% confidence level. This is graphically shown in Figure 6.

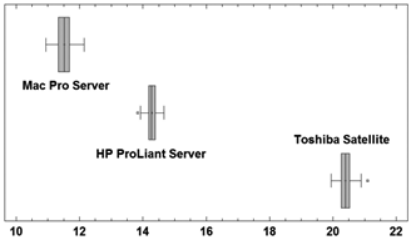


Figure 6.Box-and-Whisker Plot

4. Conclusions

In this paper, we present a web-based platform for EEG data processing in a cluster environment. With this approach, the application can be accessed from anywhere in the world by means of internet access, using the server to distribute the analysis and simulation into the cluster resources, and allowing the use of any centralized tool into a distributed environment. The system also allows the management of information workflow and its storage in a remote server, without the need of

local copies.

The performance gain by moving the computing to the cloud is also presented. Execution time decreased 30% in a nominal node (HP ProLiant), and 43% using the head computing node (Mac Pro Server), as compared to the local computer execution (Toshiba Satellite).

We provide a method to schedule jobs in a cluster with a database manager server installed, increasing the job description and status reports for further analysis. Using this database manager, multiple clients can upload their process using not only a web server, but also local software that includes MySQL connectors, generating a faster development deployment and allowing direct access to the job scheduler.

As future work, the same platform will be used in other scientific computing fields, making modules independent from the database, and increasing the plugin management extensibility.

5. Acknowledgements

We thank the DreamSpark program from Microsoft for the contribution of software infrastructure, which provided us with licenses for Windows Server and Visual Studio 2012.

6. References

[1] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," ed: ELSEVIER SCIENCE BV, 2004.

[2] R. Oostenveld, P. Fries, E. Maris, and J. M. Schoffelen, "FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data," *Computational intelligence and neuroscience*, vol. 2011, p. 1, 2011.

[3] (2012). *OpenEEG*. Available: http://openeeg.sourceforge.net

[4] EGI. (2012, 07-12-2012). Net Station. Available: http://www.egi.com/research-division-research-products/eeg-software

[5] I. D. E. Applied Neuroscience. (2012). NeuroGuide. Available: http://www.appliedneuroscience.com/

[6] B. P. GmbH. (2012). *BrainProducts home page. Available*: http://www.brainproducts.com

[7] Y. Yufeng, C. Jinyi, and X. Kaijian, "A Case of Parallel EEG Data Processing upon a Beowulf Cluster," in *Parallel and Distributed Systems (ICPADS)*, 2009 15th International Conference on, 2009, pp. 799-803.

[8] L. Wang, D. Chen, R. Ranjan, S. U. Khan, J. KolOdziej, and J. Wang, "Parallel Processing of Massive EEG Data with MapReduce," in *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on,* 2012, pp. 164-171.

[9] A. Delorme, T. Mullen, C. Kothe, Z. A. Acar, N. Bigdely-Shamlo, A. Vankov, et al., "EEGLAB, SIFT, NFT, BCILAB, and ERICA: new tools for advanced EEG processing," *Computational intelligence and neuroscience*, vol. 2011, p. 10, 2011.

[10] H. Kienle, G. Di Lucca, and S. Tilley, "Research directions in Web systems evolution IV: Migrating to the cloud," in *Web Systems Evolution (WSE), 2010 12th IEEE International Symposium on*, 2010, pp. 121-122.

[11] P. Mell and T. Grance, "The NIST definition of cloud computing (draft)," *NIST special publication*, vol. 800, p.

145, 2011.

[12] C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, 2009, pp. 4-16.

[13] E. Okorafor, "A Fault-Tolerant High Performance Cloud Strategy for Scientific Computing," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, 2011, pp. 1525-1532.

[14] C. Ingram and A. Knowles. (2013). *Code analysis, repository and modelling for E-Neuroscience*. Available: http://www.carmen.org.uk/

[15] K. Ericson, S. Pallickara, and C. W. Anderson, "Analyzing Electroencephalograms Using Cloud Computing Techniques," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 185-192.

[16] M. Gerven, A. Bahramisharif, J. Farquhar, and T. Heskes. (2013, 3-09). *Donders machine learning toolbox.* Available: https://github.com/distrep/DMLT

Modulation of the Enzymatic Activity in the dTDP-L-rhamnose Biosynthesis Pathway

Eduardo Jardón-Valadez[1] and Humberto García Arellano[2]

[1]*Departamento de Recursos de la Tierra, Universidad Autónoma Metropolitana, Unidad Lerma. Av. Hidalgo Poniente 46. Col. La Estación. Lerma, Estado de México, México. C. P. 52006*

[2]*Departamento de Ciencias Ambientales, Universidad Autónoma Metropolitana, Unidad Lerma. Av. Hidalgo Poniente 46. Col. La Estación. Lerma, Estado de México, México. C. P. 52006*

{h.jardon h.garcia}@correo.ler.uam.mx

Abstract

L-rhamnose is a 6-deoxyhexose that forms a diverse kind of glycoconjugates in the cell wall of pathogenic bacteria. dTDP-L-rhamnose is a precursor of the L-rhamnose, and it is synthetized from the nucleotide deoxy-thymidine-triphosphate (dTTP) and glucose-1-phosphate by a four-step reaction sequence involving enzymes highly conserved in bacteria and absent in humans. The Glucose-1-phosphate thymidylyltrans-ferase (RmlA) enzyme catalyzes the first reaction of the dTDP-L-rhamnose biosynthesis, the condensation of thymidine triphosphate and glucose-1-phosphate to produce dTDP-D-glucose, and its crystallographic structure has been elucidated at high resolution (1.66 Å). In this work we generated molecular dynamics trajectories, using high performance computing resources, to analyze the conformational dynamics of the RmlA enzyme in solution. The RMSD reached a stable average within the first 5 ns, and the protein secondary structure indicated stable fluctuations of the relevant domains for preserving the protein fold. Understanding the regulation mechanism is important for developing new inhibitors of the enzyme activity.

Keywords: rhamnolipids, enzyme activity, molecular dynamics, force fields, high performance computing.

1. Introduction

Pseudomonas aeruginosa is an opportunistic human pathogen, prevalent in nosocomial infections, that affects immunosuppressed patients. The pathogenicity of this bacterium is mainly due to the secretion of several toxic compounds as well as hydrolytic enzymes such as rhamnolipids, exotoxins, redox-active phenazines (pyocyanin), proteases, and phospholipase C [1-3]. The rhamnolipids secreted by P. aeruginosa are tensoactive molecules derived from hydroxylated fatty acid dimers and dTDP-L-rhamnose [4]. During infection, rhamnolipids solubilize the phospholipid surfactants coating the lungs,

making them even more susceptible for the lysis produced by phospholipase C [5].

An important virulence factor in P. aeruginosa is the shell of lipopolysaccharides (LPS) that covers its outer membrane. The LPS shell displays a complex structure, as shown schematically in figure 1A.

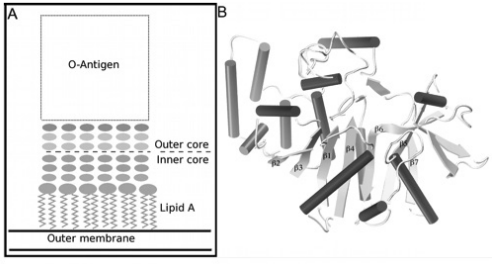


Figure 1. (A) Three domains of the lipopolysaccharide (LPS) structure covering the outer membrane in P. aeruginosa: the lipid A, the oligosaccharide core, and the O-antigen. (B) Crystal structure of the RmlA enzyme in cartoon representation, with the secondary structure domains colored in purple and yellow, respectively, for α-helices and β- sheets. The central core of β-sheets (β1-7) is surrounded by α-helices, so the structure has been classified as a αβα sandwich [13].

The lipid A domain is formed by tails of fatty acids which are esterified by disaccharide head groups, and bears binding sites to anchor the LPS [6]. The central region is the core domain, consisting of branched oligosaccharides with ~10 monomers as building blocks. The outer LPS domain, the O-antigen, contains a repetitive carbohydrate chains covalently bonded to the core domain. In P. aeruginosa, the two O-antigen characterized so far are the A-band, made of bricks of D-rhamnose homopolymer usually 70 residues long, and the B-band, formed by repetitive units with different sizes and carbohydrate types depending on the strain. The strong immunogenic response of the LPS is attributed to the B-band, since O-specific antibodies recognize the structure as well as the polysaccharide type [6-8]. The central domain of the LPS is usually divided, in more detail, in two halves known as the inner and outer core (Figure 1A) [9]. The inner core contains two residues of 3-Deoxy-D-manno-oct-2-ulosonic acid and two residues of L-glycero-D-manno-heptose, and bears phosphorylation sites. The outer core includes D-glucose, D-galactosamine and L-rhamnose. In P. aeruginosa two related outer core structures have been identified, stacked in similar amounts, namely the capped and uncapped outer core, that differ each other in the L-rhamnose termini, respectively, glucose or O-antigen [6, 9].

The precursor metabolite implicated in the LPS biosynthesis is the dTDP-L-rhamnose, which is also found in the mycobacterial cell wall of Mycobacterium tuberculosis. dTDP-L-rhamnose binds to carbohydrate polymers (D-galactofuranosyl and D-arabinofuranosyl residues) that supports the mycolic acids in the outer layer [10]. Thus, dTPD-L-rhamnose is recognized as an important molecule for the cell wall building-up in M. tuberculosis [11], and therefore its biosynthesis pathway is an obvious target for designing new antibiotics. In bacteria, the biosynthesis consists of the sequential conversion of glucose-1-phosphate (G1P) into dTDP-D-glucose → dTDP-6-deoxy-D-xylo-4-hexulose → dTDP-6-deoxy-L-lyxo-4-hexulose

→ dTDP-L-rhamnose [6]. In P. aeruginosa, these reactions are catalyzed by specific enzymes in the sequence: RmlA, RmlB, RmlC and RmlD. These enzymes are codified by the rmlBDAC operon, which is widely conserved among bacteria while absent in humans [12]. In particular, the structure of RmlA (a glucose-1-phosphate thymidylyltransferase), the first enzyme of the pathway, has been elucidated at high resolution (Figure 1B)

[13]. RmlA catalyzes the condensation of thymidine triphosphate and glucose-1-phosphate yielding dTDP-D-glucose and it is inhibited by dTDP-L-rhamnose, the final product of the reaction sequence [14]. This feedback regulation hints for the search of inhibitors resembling the dTDP-L-rhamnose, as a strategy to find new drugs against gram-negative bacteria such as P. aeruginosa and M. tuberculosis.

In this work, we performed MD simulations to explore the conformational dynamics of the RmlA enzyme, in solution, at constant pressure and temperature, respectively, of 1 bar and 300 K. Our set up included explicit water molecules, 0.15 M of NaCl as electrolytes, and the RmlA enzyme. Two simulation trajectories were generated under similar conditions: the RMLA system including the enzyme, and the RMLA-G1P system with the substrate glucose-1-phosphate in the enzyme active site. Our initial benchmarks here presented for the enzyme, and in complex with G1P, are important to identify the conformational changes involved in the dTDP-L-rhamnose biosynthesis regulation.

This work is organized as follows: in Methods we describe the simulation set up and the preliminary equilibration protocol, the algorithms used to generate the simulations trajectories along with the analysis tools; in Results we describe our observations regarding the equilibration protocol, the computational performance, and a discussion of the structure and the protein conformational dynamics; finally, we conclude with final remarks regarding the upcoming simulation runs in order to plan a systematic study of computational experiments.

2. Methods

Exploratory molecular dynamics simulations were performed for the enzyme glucose-1-phosphate thymidylyltransferase (RmlA), and for the complex RmlA-G1P. Starting from the protein structures 2 x ~50 ns long trajectories were generated according to the following protocol. 1) For the initial coordinates of protein atoms, chain C of structure PDB:1FZW [13] and chain A of structure PDB:1G23 [13], were selected, respectively, for the RMLA and RMLA-G1P systems. No missing residues were detected in the RMLA protein structure, whereas the first methionine was missing in the RMLA-G1P protein structure. To complete the enzyme coordinates, the RMLA protein fragment was fitted on top of the RMLA-G1P protein fragment, so the methionine coordinates were defined by means of the fitting procedure and included in RMLA-G1P protein structure. 2) The crystallographic water molecules interacting with RmlA, in chain C, were preserved in the initial configuration; then the protein and crystallographic water molecules were solvated, altogether in a cubic simulation box (86 Å side length). The system contained a total of 18243 water molecules. 3) Ions were

added to set up a 0.15 M salt concentration, 57 Na+ and 50 Cl- ions, which took into account the protein net charge for setting system neutrality. 4) Energy minimization was performed to prevent repulsive contacts of the initial configuration. 5) A relaxation ladder using positional restraints on the protein heavy atoms, and the oxygen atoms of the crystallographic water molecules, was completed with steps consisting of 200 ps each, with a force constant of 50, 25, 15, 10, 5, 2, and 1 kcal/mol Å2. 6) Production runs were performed without any positional restraint, at 1 bar and 300K, during ~50 ns. A similar protocol was used for the RMLA-G1P system, which includes the G1P as substrate. The G1P molecule was generated defining a bond between carbon 1 of glucose and the phosphate group. All bond interactions were specified according the atom types and atom names consistent with the CHARMM force field (see below). In the RMLA-G1P system, the single protonated state of the G1P was set up, which was expected from the slightly acidic conditions (pH ≈ 6) used to determine the enzyme activity in vitro. Finally, 57 Na+ and 49- chloride ions were added to satisfy system neutrality in the simulations box.

The CHARMM22 [15, 16] force-field parameters were used for the protein atoms, and a recent extension of the CHARMM force field parameters optimized for phosphate linked to carbohydrates was used for the G1P molecule [17, 18]. Water molecules were described with the TIP3P model [19]. The particle mesh Ewald method [20] was used to calculate the electrostatic interactions with a tolerance of 106 for the direct part of the Ewald sum, a fourth order interpolation scheme and a grid of 90X90X90. A multiple time-step scheme was used to integrate the equations of motion with time steps of 4 fs for full electrostatic forces, 2 fs for short-range non-bond interactions, and 1 fs for bond interactions. A cutoff distance of 11 Å was set for non-bond interactions, and switching functions starting at 10 Å were used to smooth the interactions at the cutoff distance. Interactions were calculated by means of a neighbor list including pairs within 12.5 Å, and the list was updated every 8 time steps. Bonded atoms forming pairs 1 to 4 were excluded form the non-bond force calculations. All bond lengths involving hydrogen atoms were constrained by using the SHAKE algorithm [21]. A Langevin thermostat [22] was used to control the temperature at 300 K, and a Nose´- Hoover-Langevin piston [23] was used to control the pressure at 1 bar. The NAMD 2.7 code was used to generate the molecular dynamics trajectories [22].

Trajectory analysis and visualization were preformed using the VMD 1.9.1 platform [24]. TCL plugins and in house scripts were used to calculate root mean square deviations (RMSD), root mean square fluctuations (RMSF), contact maps, accessible solvent areas (ASA) and the visual inspections of trajectories. The protein crystal structure was used as reference in all the calculations. Hydrogen bond interactions were identified using a geometrical criterion with a cutoff of 3.5 Å and a donor-acceptor angle of 50o or less.

3. Results

3.1 Benchmarking

As part of our simulation protocol, we tested

the scaling efficiency by running on different multicore platforms, in order to compare the CPU performance of the available HPC resources. As a measure of the CPU time we tracked the wallclock reported at the end of each MD run, in nanoseconds (ns) per day (Table 1). The processor i7-2600K was used as reference for a 100% efficiency scaling, i.e., the more CPU time to complete one nanosecond per day, the lower efficiency in the actual cluster (Table 1). Our benchmark analysis suggested that even though the AMD-Opteron architecture supports 64-cores in a single processor, the performance was rather poor in comparison with 20 interconnected Xeon-Intel servers. Hence, we completed 3.878 ns/day out of the ideal 5.256 ns with 32 cores, and 3.698 ns/day out of the 10.512 expected with 64 cores; the Xeon-Intel platform completed 6.892 ns per day using 20 nodes for a 52% efficiency scaling. Due to a better performance, trajectories were often generated in the Xeon-Intel cluster; nonetheless, the job queuing was a factor to consider in the job submission. No further performance analysis was done in this work since our main goal was to understand enzymatic processes, and the HPC was indeed a valuable tool for generating a set of protein conformations potentially important for the protein physiological functions.

3.2 Set-up protocol.

In order to evaluate the initial set up, and the choice of simulation parameters (see Methods), we calculated some indicators of the protein stability and conformational dynamics, in both the RMLA and the RMLA-G1P systems. Figure 2 shows the root mean square displacement (RMSD) for the backbone atoms of the enzyme, including α-carbons, carbonyl atoms (C and O), and the nitrogen of the amino groups. These RMSD measure indicates positional deviations (at time t) from a reference structure, which was the crystal structure. The RMSD value reached a stable average of below 2 Å, suggesting that the protein structure fluctuates around a stable average conformation. Our simulation protocol kept the protein structure intact, and the enzyme relaxed from the rigidity of the crystal environment to the aqueous solution.

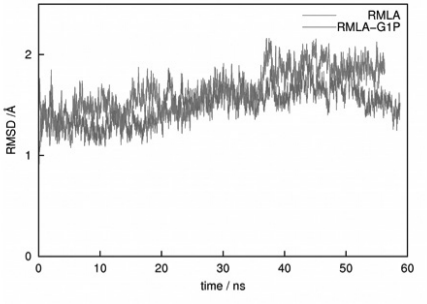


Figure 2. Root mean square deviations (RMSD) for backbone atoms of the enzyme (RmlA) and in complex with glucose-1-phosphate (RmlA-1GP). The protein structure fluctuates around a stable set of configurations with deviations below 2 Å. The protein structure equilibrates relatively fast within the first 5 ns of trajectory, which suggest that the system set-up and simulation protocol here proposed preserved the main protein interactions stabilizing its structure in solution.

3.3 Protein conformational dynamics.

To capture the mobility of the amino acids

in the protein domains, the root mean square fluctuations (RMSF) were calculated for the α-carbons. A relatively large RMSF value means that the selected atom samples positions at larger distance from its average position. In figure 3, we compared the RMSF profile along the enzyme sequence. We observe similar features in both systems, except for the domains interacting with the G1P substrate, from Ala135 to Val172. As expected, the enzyme responded to the presence of its substrate G1P by showing larger fluctuations of the residues interacting the G1P molecule. Figure 4 shows a snapshot of the simulation box after ~50 ns of trajectory. The protein conformation was superimposed on top of the initial crystal structure, in order to identify some of the conformational changes at the end of the simulation trajectories. The overall enzyme structure is well preserved, in particular the β-sheets at the core of the structure. However, displacements of α-helices and loops can be related to the fluctuations induced by the G1P, although larger simulations trajectories are required to confirm correlations between the substrate and rigid body motions of the enzyme domains.

Finally, a secondary structure analysis was performed for the enzyme in both RMLA and RMLA-G1P systems. As the previous RMSF analysis suggested, in figure 5 larger differences of the secondary structure fluctuations were observed; for instance, disruption of the β-strand structure around position Asp225, the α-helix from Gln181 to Lys190, the β-bridge formed at Lys155, among other features. By prolonging the simulation trajectories, and implementing additional analysis such as normal modes analysis, we expect to identify and confirm the correlations between the enzyme and its substrates.

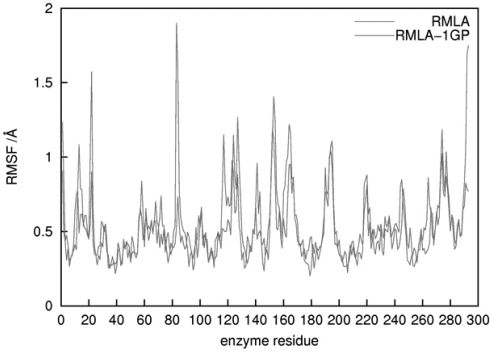


Figure 3. Comparison of the root mean square fluctuations (RMSF) in the enzyme RmlA and in complex with glucose-1-phospate (RmlA-1GP). The mobility of specific amino acids Pro83 and Ser84 seem to increase in the RMLA-1GP system. In the region important for the sugar recognition, including Ala135 to Val172, significant differences were also detected (see text).

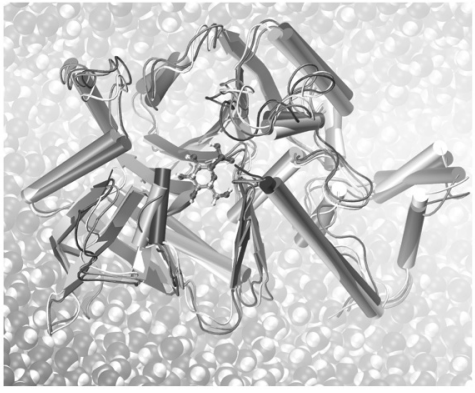


Figure 4. Simulation box showing the enzyme at the end of the MD trajectories, system RMLA in gray cartoons and system RMLA-G1P in orange cartoons. Final configurations were overlaid on top of the initial crystal structure, represented in white cartoons. The substrate glucose-1-phosphate (G1P) is drawn in sphere-rod representation, and colored by atom names according to the following color code: C-cyan, O-red, H-white, and P-gold. Secondary structure domains are drawn as rods (α-helices) and arrows (β strands). The central a β sheet is surrounded by a first layer of five α-helices, a second layer of two antiparallel β sheets, and four α-helices at the carboxyl terminus. All domains are tightly packed and the G1P substrate remained in the binding site. Solvent water molecules (red-white spheres) are shown in the background for simplicity.

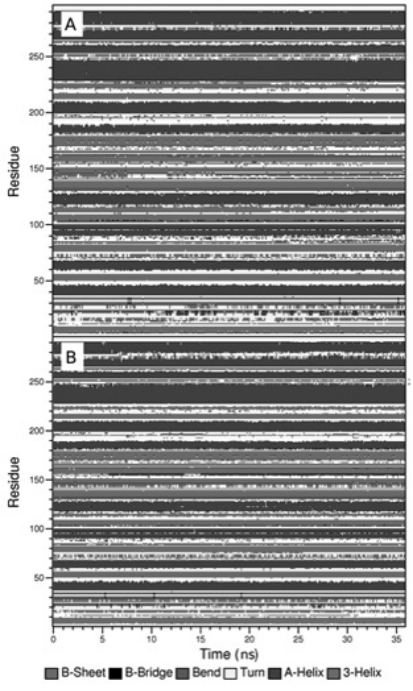


Figure 5. Secondary structure analysis for RmlA (panel A) and RmlA-G1P (panel B). Stable structural domains are mainly formed by for α-helices and β-sheets; however, specific regions showed larger fluctuations in the RMLA-G1P system likely due to the perturbation of the glucose-1-phosphate interactions in the binding site.

4. Final Remarks

We performed exploratory MD simulations to test the stability and the conformational dynamics of the RmlA enzyme in aqueous

solutions. Thanks to the access to supercomputing resources, we were able to run benchmarks on a relatively large system including ≈ 60 thousand atoms, and using a classical force field. Our suggested protocol allowed relaxing the enzyme structure from the rigid crystal structure to an aqueous solution without affecting its tridimensional folding. We detected some interesting features of the enzyme dynamics and conformational changes that could be related to the changes induced by the substrate G1P; however, larger trajectories are required to determine correlations between the substrate and the enzyme. Regarding the relevance of this exploratory MD runs upon the dTDP-L-rhamnose biosynthesis, we aim to study the full pathway by replacing each of the substrates involved, and determine the enzyme conformational changes along the intermediate formation. This conformational analysis may contribute to the understanding of the enzyme activation-inhibition cycle.

Aknowledgements

We thank the supercomputing facilities at Universidad Autónoma Metropolitana, cluster AITZALOA, and the High Performance Computing center at University of California-Irvine. This project was founded in part by the CONACyT research grant 183745.

References

[1] D.R Galloway, “Pseudomonas aeruginosa elastase and elastolysis revisited: recent developments” Mol. Microbiol. John Wiley & Sons Ltd, USA, 1991, pp. 2315-2321.

[2] M.J. Wick, D.W. Frank, D.G. Storey, B.H. Iglewski, “Structure, function, and regulation of Pseudomonas aeruginosa exotoxin-A”. Microbiology, Annual Reviews, USA, 1990, pp. 335-363.

[3] D.E. Woods, and B.H. Iglewski “Toxins of Pseudomonas aeruginosa: new perspectives” Reviews of Infectious. Diseasees. Infectious Diseases Society of America, 1983, pp S715-S722.

[4] E. Deziel, F. Lepine, D. Dennie, D. Boismenu, O.A. Mamer, R. Villemur, “Liquid chromatography/mass spectrometry analysis of mixtures of rhamnolipids produced by Pseudomonas aeruginosa strain 57RP grown on mannitol or naphthalene” BBA-Molecular and Cell Biology of Lipids, Elsevier Inc., USA, 1999, pp. 244-252.

[5] P.V. Liu, “Extracellular toxins of Pseudomonas aeruginosa” Journal of Infectious. Diseases, Oxford, UK, 1974, pp. 244-252.

[6] J.D. King, D. Kocíncová, E.L. Westman, J.S. Lam, “Lipopolysaccharide biosynthesis in Pseudomonas aeruginosa” Innate Immunity, Sage, USA, 2009, pp. 261-312.

[7] H.L Rocchetta, L.L. Burrows, and J.S. Lam, “Genetics of O-antigen biosynthesis in Pseudomonas aeruginosa”, Microbiology and Molecular Biology Reviews, ASM, USA, 1999, pp. 523-553.

[8] E. Kintz, J.M. Scarff, A. DiGiandomenico, J.B. Goldberg, “Lipopolysaccharide O-Antigen Chain Length Regulation in Pseudomonas aeruginosa Serogroup O11 Strain PA103”, Journal of Bacteriology, ASM, USA, 2008, pp. 2709-2716.

[9] D. Kocincova, and J.S. Lam, “Structural Diversity of the Core Oligosaccharide Domain of Pseudo-

monas aeruginosa Lipopolysaccharide", Biochemistry (Moscow), Springer, USA, 2011, pp. 755-760.

[10] Y.F Ma, R.J. Stern, M.S. Scherman, V.D., Vissa, W.X. Yan, V.C. Jones, F.Q. Zhang,S.G. Franzblau, W.H. Lewis, M.R. McNeil, "Drug targeting Mycobacterium tuberculosis cell wall synthesis: Genetics of dTDP-rhamnose synthetic enzymes and development of a microtiter plate-based screen for inhibitors of conversion of dTDP-glucose to dTDP-rhamnose" Antimicrobial Agents and Chemotherapy, ASM, USA, 2001, pp. 1407-1416.

[11] Y.F. Ma, F. Pan, and M.R. McNeil, "Formation of dTDP-rhamnose is essential for growth of mycobacteria" Journal of Bacteriology, ASM, USA, 2002, pp. 3392-3395.

[12] R. Rahim, L.L. Burrows, M. A.Monteiro, M.B. Perry, J.S. Lam, "Involvement of the rml locus in core oligosaccharide and O polysaccharide assembly in Pseudomonas aeruginos", Microbiology, Society for General Microbiology, UK, 2000, pp. 2803-2814.

[13] W. Blankenfeldt, M. Asuncion, S. Lam, J.H. Naismith, "The structural basis of the catalytic mechanism and regulation of glucose-1-phosphate thymidylyltransferase (RmlA)" The EMBO Journal, EMBO, 2000, pp. 6652-6663.

[14] A. Melo, A. and L. Glaser, "The Nucleotide Specificity And Feedback Control Of Thymidine Diphosphate D-Glucose Pyrophosphorylase" Journal of Biological Chemistry, American Society, 1965, pp. 398-405.

[15] A.D. MacKerell Jr., D. Bashford, M. Bellott, R.L. Dunbrack Jr., J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau,F.T.K, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher III, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, M. Karplus, "All-atom empirical potential for molecular modeling and dynamics studies of proteins" Journal Physical Chemistry B, ACS, USA, 1998, pp. 3586-3616.

[16] A.D MacKerell Jr., M. Feig, and C.L. Brooks II, Extending the treatment of backbone energetics in protein force fields: Limitations of gas-phase quantum mechanics in reproducing conformational distributions in molecular dynamics simulations. Journal of Computational Chemistry, John Wiley & Sons Ltd, USA, 2004, pp. 1400-1415.

[17] O. Guvench, S.S. Mallajosyula, E.P. Raman, E. Hatcher, K. Vanommeslaeghe, T.J. Foster, F.W. Jamison II, A.D. MacKerell Jr. "CHARMM Additive All-Atom Force Field for Carbohydrate Derivatives and Its Utility in Polysaccharide and Carbohydrate-Protein Modeling" Journal Chemical Theory & Computation, ACS, USA, 2011 pp. 3162–3180.

[18] S.S. Mallajosyula, O. Guvench, E. Hatcher, A.D. MacKerell Jr. "CHARMM Additive All-Atom Force Field for Phosphate and Sulfate Linked to Carbohydrates" Journal Chemical Theory & Computation, ACS, USA, pp. 759-776.

[19] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, "Comparison of simple potential functions for simulating liquid water", Journal of Chemical Physics, AIP, USA, 1983, pp. 926-935.

[20] T. Darden, D. York, and L. Pedersen, "Particle mesh Ewald: An N•log(N) method for Ewald sums in large systems", Journal of Chemical Physics, AIP, USA, 1993 pp. 10089-10092.

[21] J.-P. Ryckaert, G. Ciccotti, and H.J.C. Be-

rendsen, “Numerical integration of the Cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes”, Journal of Computational Chemistry, John Wiley & Sons Ltd, USA, 1977. 23: p. 327-341.

[22] J.C. Phillips, B. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, K. Schulten,”Scalable molecular dynamics with NAMD Journal of Computational Chemistry, John Wiley & Sons Ltd, USA 2005. pp. 1781-1802.

[23] S. E. Feller, Y. Zhang, R.W. Pastor, B.R. Brooks, “Constant pressure molecular dynamics simulation: The Langevin piston method Journal of Chemical Physics, AIP, USA, 1995. pp. 4613-4621.

[24] W. Humphrey, W. Dalke, and K. Schulten, “VMD: Visual molecular dynamics”, Journal of Molecular Graphics, Elsevier, USA, 1996, pp. 33-38.

Optimizations to a Distributed Repository of Multimedia Files using High-Performance Hardware and Software

Alfredo Cristóbal-Salas, Efrén Morales-Mendoza, Hermilo Israel Gayosso-Murillo, Luis Constantino Córdova-Solís

Facultad de Ingeniería en Electrónica y Comunicaciones, Universidad Veracruzana
Poza Rica de Hidalgo, Veracruz, México 93390
{acristobal, efmorales}@uv.mx

Abstract

Nowadays, it is easy to view the difference in speed between processing a remote request and transfer remote files, specially, when they are big in size. Distributed storage systems usually require a module to manage temporal information in order to reduce retrieval time for remote information and thus improve its entire performance; specially, when there is constant access to remote information. This paper presents the optimizations made to a web-based system that manages a repository of multimedia files. Preliminary results indicate that the new system has better response time to retrieve remote documents when taking into account the frequency of use and the similarity of the documents.

Keywords: distributed systems, multithreading, data locality exploitation, concept maps, distributed repositories

I. Introduction

This paper presents the results of optimizing a web-based computer system that we call CALDO (Computer-Assisted Learning DOcumentation). This system lets students create multimedia documents which are the result of their learning process; once the documents are finished by students and reviewed by teacher, they are stored in a repository.

CALDO system allows teachers to provide multimedia information to students that would be used as didactic material which is stored in the same system repository. These files are mostly text with images, pictures, sound or video whose sizes could range from few megabytes to almost gigabytes. For example: full HD video of a class session or a book with high defined pictures as in art sciences.

CALDO system is not intended to be a broadcasting service but a file repository; so, once files are already in the repository then these files should be retrieved by teachers or students in order to be used during class sessions. This repository should allow request for multimedia files which has to be associate to a course´s topic and if several file types are available for one topic then the system should retrieve the file type which is more preferred by user.

Another important feature of this system is that user has to have a partial answer of his/hers request, i. e., if there are more than one answer for a topic then system should deliver answer to user by updating the initial partial answer with

the answer already known up to that point.

All requests for multimedia files should have an ID number to identify the request when updating the initial partial answer with new results. A request has an ID number equals zero when it is sent to the repository for the first time, after storing the request in the database, a new unique ID number is assigned to it.

After analyzing the functionality of CALDO system it is easy to discover that the number of files in the repository will grow very fast and if we consider a highly used system then the system performance will decrease rapidly. So, it seems quite reasonable to introduce distributed computing techniques that guaranty that system will keep its performance even if number of users request or files increases.

II. Distributed Caldo Design

Computer web-based systems working on distributed environment often require to have access and to process data which is stored in different repositories that may be placed (physically) in different locations. Programs, for distributed storage systems, access data in these distributed repositories by means of request-and-response messages which may consume system resources and have much higher latency than local memory access. According to [1, 8, 20] there are some techniques to improve system performance such as:

- Overlapping communication and computation time, keeping the local web service busy while a remote memory request is fulfilled,
- retrieving more than one datum per request bringing the most likely to be accessed, moving remote processes to where data is located [11, 12]
- Moving data where they are required [13, 14].

Adding caching to remote storage requests reduces both, the number of requests and their latency. Our research goes in this direction in order to introduce a distributed storage of multimedia files. So, before giving more details about the distributed version of CALDO system, it is convenient to review some of the techniques to improve performance of distributed and multi-processor systems.

Multithreading [12]. This technique is based on threads which are sequences of instructions that, once started, run to completion without blocking. Multithreading helps to tolerate latency of remote information by allowing the processing elements to perform other useful operations while a fetch instruction occurs. This technique works quite well as long as there are ready threads available and the thread switch overhead is low and it requires the presence of sufficient available parallelism to be able to tolerate long-latency operations.

Prefetching [16]. Here the remote memory operations are initiated before the results are needed either automatically or under program control. Prefetching needs advance knowledge of upcoming remote data access patterns to make requests far enough ahead of their actual utilization.

Nomadic threads [12]. In this technique, a thread of control is migrated from the processing element that initially executes a remote memory access to a processing element that holds data needed. Nomadic thread is an approach to reducing the number of remote memory access messages.

A number of research projects have showed migration to be quite effective at exploiting locality for data structures, such as lists and trees, where caching is not very effective [4,

10, 18].

Split-phase fetch operation [13]. All fetch operations requesting for data in a distributed memory systems are re-designed in two or more phases: one to initiate the fetch operation and other to be activated upon the requested data arrive. Using this technique the processing element is not idle waiting for a request to be answered.

Data reuse exploitation [15]. There are two major sources of data reuse: multiple accesses to the same memory location and accesses to consecutive memory locations. Multiple accesses to the same memory location may arise from either a single array reference or multiple array references. This type of reuse is called temporal reuse. A second source of data reuse is caused by multiple accesses to consecutive memory locations. This reuse is called spatial reuse. In [2, 13, 15, 17, 19] some optimizations made to concurrent systems by temporal and spatial data exploitation are presented.

Caching data [13]. Cache memory is a common optimization technique in computer architecture that refers to a small in size and fast access storage that contains remote data that is constantly used by local processing element. Current microprocessors contain three levels of cache memory which the closest to microprocessor has few registers to store data while the following cache memories have more registers to store data. With this strategy, if data is not in microprocessor registers then it is searched in the cache memory level1 if it is not in leve1 then it is searched in cache memory level2, if it is still not found in this level then it is searched in level3. If the requested data is not present in all three levels of cache memory then a request is sent to main memory to retrieve such data. Some research on the impact of caching data on system performance is shown in [3, 9, 21].

After reviewing general techniques to design efficient distributed repositories it is time to describe the distributed version of CALDO system. In Figure 1 we can see that the new system is organized in two modules:

(a) main module (MM). This module is in charge of all interactions with users. This module has two software agents: a request manager and a request scheduler. The first agent receives user requests and stores them in the network of databases. This agent is also in charge of receiving requests for updating coming from user interface which makes reference to a specific request via ID number. The second agent is in charge of distributing the request among the Institutional Modules (IMs) using round-robin scheduling or using user-specified IMs option which is available in the interface. This latter option sends the request to a specific IM where user wants his/hers request to be completed. It is important to mention that the distributed system administration can activate or deactivate IMs via web. IMs have three states: inactive, active, saturated. The scheduler can change the status of an IM from “available” to “saturated” when the IM has more requests than a threshold or when it exceeds the allowed response time, so no more work will be assigned to that particular IM until it has less requests than the threshold.

The requests have four states: ready to be processed, sent to an IM, updated by an IM and solved.

(b) Institutional module (IM). This module has several agents called “worker” which are in charge of solving the user requests where each worker solves one by one request. All workers check the “requests” database to check

for unsolved requests; they select the first request in the list and the database "concepts", which contains the URL of multimedia files, is consulted in order to solve the request. The answer to a specific request is stored in the main request waiting list allocated in the MM to allow the request manager to update the result in the user interface.

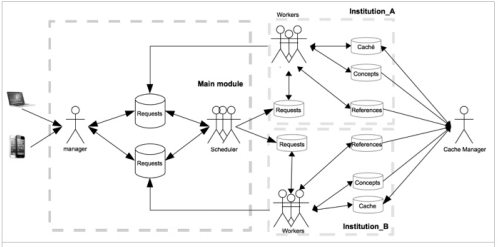


Figure 1. Distributed version of CALDO system

Figure 1. Distributed version of CALDO system

III. Caching Remote Files

In the first stages of this project, only local accesses where taken into account; however, in a distributed storage system the repositories could be physically separated or located in places with high latency interconnection networks. Also, the system performance could be reduced when transferring large files, that's why, an additional service is introduced in the distributed system. This service copies remote files into local temporary space located in each IM (see Figure 1). With this new service, when a worker searches for a remote file, it first checks the local "cache" database, if the remote requested file is found in local cache then the transfer is avoided and the local file is used instead. Also, each time a worker finds a file in local repositories then the references count for that particular file in that remote location is increased.

This new service is inspired in the work presented in [5,6].

As seen in Figure 1, this new service has an agent called "cache manager" which checks references counters of all files and selects the first ten files with high references and copy the file to the remote location, i. e., if location Institution A frequently request File1 and if the references count exceeds a threshold, then, File1 is copied to "cache" database and the references count is set to zero.

The idea of introducing temporary storage for remote files can reduce accessing time, however, when transferring large files accessing time could take more time than expected. This is why, in Figure 2, an algorithm to reduce time for file transfer is described. This algorithm takes into account the size of file and the peak network traffic rates in the interconnection network between IMs.

This algorithm is currently implemented in GNU/Linux commands and it is invoked via PHP. The algorithm uses commands: "zip" to compress files, "split" to divide files into pieces, "scp" for remote copying, "ssh" to run commands in a remote system, "cat" to join a split file, and finally "rm" to remove temporary files used during the transferring.

```
Select one IM
Retrieve peak network traffic rates
Retrieve from its database "requests" ordered by number of accesses
Select the 10 most consulted files requested by remote users
For all files selected
  if file size more than 300MB
     if current time is between 09:00hrs and 13:00 and between 16:00hrs and 19:00hrs
          source node: compress the file to transfer (zip command)
          source node: divide the file to transfer into X parts. X=2,4,6,10 (split command)
          source node: send each part of the file (scp command)
          source node: wait until send operations are already finished
          destination node: join the parts of file (ssh+cat command)
          destination node: delete the parts of file transferred (ssh+rm command)
          destination node: rename the file transferred to be named as the original file (ssh)
          destination node: update "concepts" database
  else
     Send file to destination node (scp command)
```

Figure 2. Algorithm used to transfer files between IMs

Figure 2. Algorithm used to transfer files between IMs

IV. Locality Exploitation

In [7] a search engine to retrieve files based on concept maps is presented. That system can retrieve files associated to a concept and because that system is represented as a directed-tagged graph it is possible to know all files related to that specific concept. Adapting that search engine with this distributed system, it is possible to transfer not only one file but transfer all files related to a specific file. Figure 3 shows an example of this feature, user searches for concept "physics" and the graph shows a group of concepts that are related to the concept "physics", once the graph is retrieved then all files can be compressed in order to transfer all files at once. If user is searching for "physics" there is high probability that surrounding concepts are consulted too. By doing this, file transfer latency is hidden and spatial data locality is exploited.

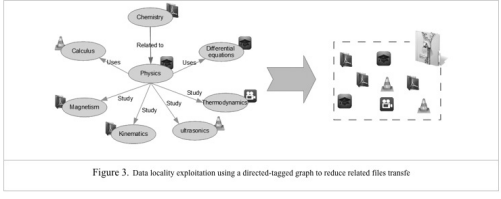


Figure 3. Data locality exploitation using a directed-tagged graph to reduce related files transfe

Figure 3. Data locality exploitation using a directed-tagged graph to reduce related files transfe.

V. Preliminary Results

This distributed system was tested at School of Engineering in Electronics and Communications, Universidad Veracruzana. The MM is running in a MACPRO 2.4Ghz, 12MB cache L3, 8GB SDRAM ECC DDR2 1024Mhz, GDDR5 1Gb ATI Radeon HD 5870MB running OSX server 10.7 while the IM is running in a iMAC 2.5Ghz 64-bits intel core i5 4GB SDRAM ECC DDR3 1333Mhz GDDR5 1Gb ATI Radeon HD 6750MB Mac OS X Lion 10.7.3. The computers are interconnected by a 100Mbps which has regular network traffic. To test the system performance we select three different situations where we consider network could be saturated: (a) FIEC-FIEC, user requests comes from the same network segment where the server is. In this segment, there is potential network traffic of 200 users, (b) USBI-FIEC, the server is in one location and the request comes from another network segment. In this segment there is potential network traffic of 800 users, (c) FPED-FIEC, MM and IM are in different network segments and there is potential network traffic of 1000 users.

In Figure 4, the time for transferring files with different sizes, such as: 1MB, 10MB, 50MB, 100MB, 300MB and 600MB are shown. From this figure is possible to see that only file sizes 300MB and 600MB are over 5 seconds which is our threshold for waiting a transfer to complete.

For files over 100MB, four optimization techniques depending on how many parts the original file was divided before the transfer and following the algorithm shown in Fig. 2: (a) Op1, original file is divided in two parts, (b) Op2, original file is divided in four parts, (c) Op3, original file is divided in six parts, (d) Op4, original file is divided in ten parts.

In Figure 5, Time spent for transferring files in three different network segments and applying different optimization strategies is shown. From this figure we see that only in Op4 the time spent during the transferring is less than 5 seconds. So by default our system divides compressed files bigger than 300MB in ten parts before the transferring.

In Figure 6, the speedup achieved by the system after optimizations is presented. As can be seen the bigger is the file to transfer, the highest is the speedup. This new system´s feature allows time and spatial information locality improving the system performance.

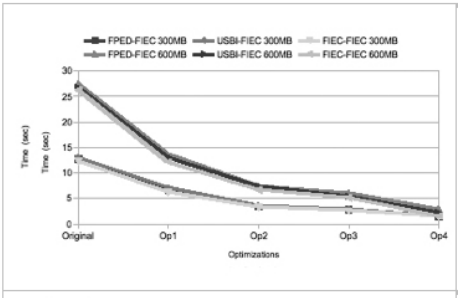


Figure 4. Time spent to transfer files in three different network segments.

Figure 4. Time spent to transfer files in three different network segments.

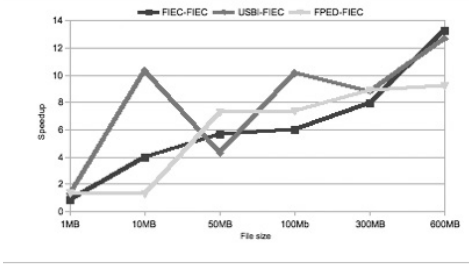


Figure 6. Speedup achieved after optimizations.

VI. Acknowledgement

This work is sponsored, in part, by Universidad Veracruzana through Facultad de Ingeniería en Electrónica y Comunicaciones and by Secretaría de Educación Pública through the 2010 call for projects in support to new full-time professors with the title (PROMEP/103.5/11/838): "Documentación digital y colaborativa del aprendizaje para el fortalecimiento de las competencias académicas".

Also, authors want to thank to Red Iberoamericana de Supercómputo through the project #FP7-288883: "A network for supporting the coordination of Supercomputing research between Europe and Latin America" for the logistic and the financial support to accomplish this project.

VII. Conclusions

This paper reports the details of optimizing a computer system for retrieving multimedia teaching materials by introducing multi-thread and distributed computing techniques to separate the system's layers to be processed by different software agents and hardware servers. By doing this, the entire system can process more request keeping its performance. All agents introduced in the system can be multiplied as much as the server's main memory can handle. The system was divided in two modules: main and institutional. Both modules can be replicated depending on how complex organizations are willing to work together.

Paper also presents the advances in designing a strategy to exploit time and spatial information locality by cache memory techniques and by using concept maps as data structure. These two optimization techniques allow the system to detect files related to a user request, all these related files are then compressed, divided and transferred in a concurrent way and stored in a temporary storage waiting for future references.

Preliminary results show that new system has improved its performance by reducing the most time consuming operations which are to find

relevant files which are associated or related to the original user request for information. Our new system design allows institutions to share information about different topics which may enrich their teachers and students.

VIII. References

[1] Bakshi, A., Lin, W-Y., Gaudiot, J-L., Makhija, M., Prasanna, V-K., Ro, W., Shin, C. (2000). Memory Latency: to Tolerate or to Reduce?, Invited Paper. The 12th Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2000 Oct 24-27, 2000

[2] Bandera, G., Zapata, E-L. (2001). "Data locality exploitation in algorithms including sparse communications," Parallel and Distributed Processing Symposium., Proceedings 15th International , vol., no., pp.6 pp., Apr 2001 doi: 10.1109/IPDPS.2001.924961

[3] Bellas, N-E., Hajj, I-N., Polychronopoulos, C-D. (2000). Using Dynamic Cache Management Techniques to Reduce Energy in General Purpose Processors. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8(6), December 2000.

[4] Blumofe, R-D., Joerg, C-F., Kuszmaul, B-C., Leiserson, C-E., Randall, K-H., and Zhou, Y. (1995). "Cilk: An Efficient Multithreaded Runtime System," in Proceedings of Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Santa Barbara, CA, 1995.

[5] Cristóbal-Salas, A., Tchernykh, A. (2002). Design of and I-Structure Software Cache for Distributed memory Systems. In proceeding of: the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '02, June 24 - 27, 2002, Las Vegas, Nevada, USA, Volume 4

[6] Cristóbal Salas, A., Tchernykh, A. (2004). I-Structure software cache for distributed applications. Dyna, marzo, 67-74. ISSN 0012-7353. Available in: <http://redalyc.uaemex.mx/redalyc/src/inicio/ArtPdfRed.jsp?iCve=49614108>

[7] Cristóbal-Salas, A., Morales-Mendoza, E., Rodríguez-Jiménez, A. (2011). Interacción con Repositorios de Documentos Multimedia usando Mapas Conceptuales. In proceeding of: XXIV Congreso Nacional y X Congreso Internacional de Informática y Computación. Colima, Colima, México 26, 27 y 28 de Octubre de 2011.

[8] Dennis, J. B. and Gao, G. R. (1994) "Multithreaded Architectures: Principles, Projects, and Issues," in Multithreaded Computer Architecture: A Summary of a State of the Art, R. Iannucci, G. Gao, J. R. Halstead, and B. Smith, Eds. Boston: Kluwer Academic Publishers, 1994, pp. 1-72.

[9] Dennis, J. B.; Gao, G. R. (1995). On Memory Models and Cache Management for Shared-MemoryMultiprocessors.CSGMEMO 363,Laboratory for Computer Science, MIT., March1995

[10] Hsieh, W-C., Wang, P., and Weihl, W-E. (1993). "Computation Migration: Enhancing Locality for Distributed-Memory Parallel Systems," in Proceedings of 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, San Diego, CA, 1993.

[11] Jenks, S. and Gaudiot, J-L. (2013). A Multithreaded Runtime System With Thread Migration for Distributed Memory Parallel Computing. January 10, 2013, available from <http://www.stephenjenks.com/>

[12] Jenks, S. and Gaudiot, J-L. (1997) "Exploiting Locality and Tolerating Remote Memory Access Latency Using Thread Migration," International Journal of Parallel Programming, vol. 25, no. 4, pp. 281-304, August 1997.

[13] Lin, W-Y., Gaudiot, J-L. (1998). The Design of I-Structure Software Cache System. Workshop on Multithreaded Execution, Architecture and Compilation (MTEAC'98). Las Vegas, Neveda, Jan. 31 - Feb. 1, 1998

[14] Lin, W-Y., Gaudiot, J-L., Amaral, J-N., and Gao, G-R. (2000). Performance Analysis of the I-Structure Software Cache on Multi-Threading Systems. 19th IEEE Internation Performance, Computing and Communication Conference, IPCCC2000, Phoenis, Arizona, Feb. 20-22, 2000

[15] Kennedy, K., McKinley, K-S. (1992). Optimizing for Parallelism and Data Locality. Proceedings of the 1992 ACM International Conference on Supercomputing, Washington, D.C, July, 1992.

[16] Mowry, T.C. (1994). "Tolerating Latency through SoftwareControlled Data Prefetching," Ph.D. Thesis, Computer Systems Laboratory. Stanford, CA: Stanford University, 1994.

[17] Prieto, M., Llorente, I-M. Tirado, F. (2000)"Data locality exploitation in the decomposition of regular domain problems," Parallel and Distributed Systems, IEEE Transactions on , vol.11, no.11, pp. 1141- 1150, Nov 2000 doi: 10.1109/71.888635

[18] Rogers, A., Carlisle, M-C., Reppy, J-H., and Hendren, L-J. (1995). "Supporting Dynamic Data Structures on Distributed Memory Machines" ACM Transactions on Programming Languages and Systems, vol. 17, no. 2, pp. 233-263. 1995

[19] Talaga, P. (2012). "Exploiting Data Locality in Dynamic Web Applications" (2012). Electrical Engineering and Computer Science - Dissertations. Paper 323.

[20] SSUIF. (2013). "The Stanford SUIF Compiler Group," January 10, 2013, available from <http://suif.stanford.edu/>

[21] Su, C-L., Despain, A. (1995). Cache design tradeoffs for power and performance optimization: A case study. In Proceedings of International Symposium on Low Power Design, April 1995.

[22] Yang, C-L., Lee, C-H. (2004). HotSpot Cache: Joint Temporal and Spatial Locality Exploitation for I-Cache Energy Reduction. ISLPED'04, August 9–11, 2004, Newport Beach, California, USA pp. 114-119.

Solving Efficiently Multiple Approximate Similarity Queries

Mariela Lopresti, Natalia Miranda, Fabiana Piccoli, Nora Reyes
LIDIC. Universidad Nacional de San Luis,
Ejército de los Andes 950 - 5700 - San Luis - Argentina
e-mail: {omlopres, ncmiran, mpiccoli, nreyes}@unsl.edu.ar

Abstract

Content-based retrieval of similar objects in multimedia databases can be very useful for web search engines, industrial applications, medicine, and molecular biology, among others. For this kind of retrieval it is meaningless to look for elements exactly equal to a given one as query. Instead, we need to measure the dissimilarity between the query object and each database object.

The metric space model is a paradigm that allows to modelized all the similarity search problems. Metric databases permit storing objects from a metric space and efficiently performing similarity queries over them; that is, in general, by reducing the Lumber of distance evaluations needed. Therefore, the goal is to preprocess the dataset such that queries can be answered with as few distance computations as possible. Moreover, for very large metric database is not enough to preprocess the dataset by building an index, it is also necessary to speed up the queries by using high performance computing using GPU. Besides, in some cases, to speed up similarity queries we can accept approximate answers where accuracy or determinism is traded for faster searches.

A good representative of indexes for approximate similarity searches is the Permutation Index. In this work we present, and empirically evaluate using databases of different data nature, an implementation of the Permutation Index for a pure GPU architecture. This implementation is able to solve many approximate similarity queries at the same time.

Keywords: Approximate similarity search, Metric spaces, High performance computing, GPGPU.

1. Introduction

Due to an increasing interest in manipulating and retrieving multimedia data, nowadays the problem of similarity searching receives much attention. The metric space model is a paradigm that allows modelizing all the similarity search problems. A metric space *(X, d)* is composed of a universe of valid objects X and a distance function $d : X * X \rightarrow R^+$ defined among them. The distance function determines the similarity (or dissimilarity) between two given objects and satisfies several properties which make it a metric. Given a dataset of $|U| = n$ objects, a query can be trivially answered by performing n distance evaluations, but sequential scan does not scale for large problems. The reduction of Lumber of distance evaluations is important to achieve better results. Therefore, in many cases

preprocessing the dataset is a good option to solve queries with as few distance computations as is possible. An index helps to retrieve the objects from U that are relevant to the query by making much less than n distance evaluations during searches [1]. One of these indexes is the Permutation Index [2].

Moreover, for very large metric database is not enough to preprocess the dataset by building an index, it is also necessary to speed up the queries by using high performance computing (HPC). In order to employ HPC to speed up the preprocess of the dataset to obtain an index, and to answer posed queries, the Graphics Processing Unit (GPU) represents a good alternative. The GPU is attractive in many application areas for its characteristics, especially because of its parallel execution capabilities and fast memory access. They promise more than an order of magnitude speedup over conventional processors for some non-graphics computations.

A GPU computing system consists of two Basic components, the traditional CPU and one or more GPUs (Streaming Processor Array). The GPU can be considered as a many-cores coprocessor allows to support fine grain parallelism (a lot of threads run in parallel, all of them collaborate in the solution of the same problem) [3], [4]. GPU is different than other parallel architectures because it shows flexibility in the local resources allocation to the threads. In general, a GPU multiprocessor consists of several streams multiprocessors; each of them has multiple processing units, records and on-chip memory. Each stream multiprocessor can run a variable number of threads. There are many tools to program the GPU, CUDA is one of them.

CUDA is a standard C/C++ extended by several keywords and constructs. Its programming model is SPMD (Single Process-Multiple Data) with two main characteristics: the parallel work through concurrent threads and the memory hierarchy. A CUDA program consists of multiple phases executed on either CPU or GPU.

In metric spaces, the indexing and query resolution are the most common operations. They have several aspects that accept optimizations through the application of high performance computing techniques. There are many parallel solutions for some metric space operations implemented to GPU. Querying by k-NN has concentrated the greatest attention of researchers in the area, so there are many Solutions that consider GPU. In [5], [6], [7], [8], [9] different proposal are made, all of them improvement to brute force algorithm (sequential scan) to find the *k*-NN of a query object. They differ in which process part is parallelized or which methodology is applied. Besides, different parallel implementations of scan sequential, there are others proposals, [5], [10], [11], that implement solutions for metric indexes: List of Clusters, SSS-Index and Spaguettis index. In every previous research work, the authors report benefits, which are strictly linked to the characteristics and architecture of the GPU.

The paper is organized as follows: Sections II and III describe all the necessary concepts to understand our work and the state of art in the use of GPU to accelerate metric indexes, Section IV introduces the sequential version of Permutation Index, Sections V, VI, and VII sketch the characteristics of our proposal and its empirical performance. Finally, the conclusions and future works are exposed.

2. Metric Spaces, Queries and Indexes

A metric space *(X, d)* is composed of a universe of valid objects X and a distance function $d : X * X \rightarrow R^+$ defined among them. The distance function determines the similarity (or dissimilarity) between two given objects and satisfies several properties such as strict positiveness (except $d(x, x) = 0$, which must always hold), symmetry $(d(x, y) = d(y, x))$, and the triangle inequality $(d(x, z) \leq d(x, y) + d(y, z))$. The finite subset $U \subseteq X$ with size $n = |U|$, is called the database and represents the set of objects of the search space. The distance is assumed to be expensive to compute; hence it is customary to define the search complexity as the number of distance evaluations performed, disregarding other components.

There are two main queries of interest [1], [12], [13]: Range Searching and the *k* Nearest Neighbors (*k*-NN). The goal of a range search $(q, r)_d$ is to retrieve all the objects $x \in U$ within the radius *r* of the query *q* (i.e. $(q, r)_d = \{x \; U / d(q, x) \leq r\}$). In *k*-NN queries, the objective is to retrieve the set $(q) \subseteq U$ such that $|k\text{-NN}(q)| = k$ and $\forall x \in k\text{-NN}(q), v \in U \wedge v \notin k\text{-NN}(q), d(q, x) \leq d(q, v)$.

When an index is defined, it helps to retrieve the objects from *U* that are relevant to the query by making much less than *n* distance evaluations during searches. The saved information in the index can vary, some indexes store a subset of distances between objects, and others maintain just a range of distance values. In general, there is a tradeoff between the quantity of information maintained in the index and the query cost it achieves. As more information an index stores (more memory it uses), lower query cost it obtains. However, there are some indexes that use memory better than others. Therefore in a database of n objects, the most information an index could store is the $n(n-1)/2$ distances among all element pairs from the database. This is usually avoided because $O(n^2)$ space is unacceptable for realistic applications [14].

Proximity searching in metric spaces usually are solved in two stages: preprocessing and query time. During the preprocessing stage an index is built and it is used during query time to avoid some distance computations. Basically the state of the art in this area can be divided in two families [1]: *pivot based algorithms and compact partition based algorithms*. In the first case, the index consists in a set of pivots $\{p1, \ldots, pm\} \subseteq U$, which computes and keeps (in a data structure, usually like a tree) some (or all) distances $\{d(p1, x), d(p2, x), \ldots, d(p_m, x)\}$, $x \in U$. The queries are solved considering all pivots.

In the second case, the space is divided into small and compact zones. A set of objects, called centers, $\{c_1, \ldots, c_s\} \subseteq U$ are chosen and the rest of the elements are distributed into the s zones defined in different ways by the *centers* c_i. The index is composed by the centers, the elements of each zone, and often some additional distances.

There is an alternative to "exact" similarity searching called approximate similarity searching [15], where accuracy or determinism is traded for faster searches [1][12][13][16], and encompasses *approximate and probabilistic algorithms*. The goal of approximate similarity search is to reduce significantly search times by allowing some errors in the query outcome. In approximate algorithms one usually has a threshold ε as parameter, so that the retrieved elements are guaranteed to have a distance to

the query q at most (1 + ε) times of what was asked for [17]. Probabilistic algorithms on the other hand state that the answer is correct with high probability. Some examples are [18], [19]. In the next section we detail a probabilistic method: *Permutation Index* [2].

3. GPGPU

Mapping general-purpose computation onto GPU implies to use the graphics hardware to solve any applications, not necessarily of graphic nature. This is called GPGPU (General-Purpose GPU), GPU computational power is used to solve general-purpose problems [3], [4]. The parallel programming over GPUs has many differences from parallel programming in typical parallel computer, the most relevant are: *The number of processing units, CPU-GPU memory structure and Number of parallel threads.*

Every GPGPU program has many basic steps, first the input data transfers to the graphics card. Once the data are in place on the card, many threads can be started (with little overhead). Each thread works over its data and, at the end of the computation, the results should be copied back to the host main memory.

Not all kind of problem can be solved in the GPU architecture, the most suitable problems are those that can be implemented with stream processing and using limited memory, i.e. applications with abundant parallelism.

The Compute Unified Device Architecture (CUDA), supported from the NVIDIA Geforce 8 Series, enables to use GPU as a highly parallel computer for non-graphics applications [3], [20]. CUDA provides an essential high-level development environment with standard C/C++ language. It defines the GPU architecture as a programmable graphic unit which acts as a coprocessor for CPU. It has multiple streaming multiprocessors (SMs), each of them contains several (eight, thirty-two or forty-eight, depending GPU architecture) scalar processors (SPs).

The CUDA programming model has two main characteristics: the parallel work through concurrent threads and the memory hierarchy. The user supplies a single source program encompassing both host (CPU) and kernel (GPU) code. Each CUDA program consists of multiple phases that are executed on either CPU or GPU. All phases that exhibit little or no data parallelism are implemented in CPU. Contrary, if the phases present much data parallelism, they are coded as kernel functions in GPU. A *kernel* function defines the code to be executed by each thread launched in a parallel phase.

GPU computation considers a hierarchy of abstraction layers: *grid, blocks and threads.* The threads, basic execution unit that executes *kernel* function, in the CUDA model are grouped into *blocks*. All threads in a block execute on one SM and communicate among them through the shared memory. Threads in different blocks can communicate through global memory. Besides shared and global memory, the threads have their local variables. All *Thread−blocks* form a grid. The number of grids, blocks per grid and threads per block are parameters fixed by the programmer, and adjustable to improve performance.

Respect of memory hierarchy, CUDA threads may access data from multiple memory spaces during their execution. Each thread has private local memory and each block has shared memory visible to all its threads. These memories have the same lifetime that the kernel. All threads have access to the

same global memory and two additional read-only memory spaces: the constant and texture memory spaces, which are optimized for different memory usages. The global, constant and texture memory spaces are persistent across launched *kernel* by the same application. Each kind of memory has its own access cost, and the global memory accesses are the most expensive.

4. Sequential Permutation Index

Let P be a subset of the database U, P = $\{p_1,p_2,...,pm\} \subseteq U$, that is called the permutants set. Every element x of the database sorts all the permutants according to the distances to them, thus forming a permutation of *P*: $\Pi_x = <pi_1, pi_2, ...p^i_m>$. More formally, for an element $x \in U$, its permutation Π_x of *P* satisfies $d(x,\Pi_x(i)) \leq d(x,\Pi_x(i+1))$, where the elements at the same distance are taken in arbitrary, but consistent, order. We use $\Pi{-}x^1(pi_j)$ for the rank of an element pi_j in the permutation Π_x. If two elements are similar, they will have a similar permutation [2].

Basically, the permutation-based algorithm is an example of probabilistic algorithm, it is used to predict proximity between elements, by using their permutations. The algorithm is very simple, in the offline preprocessing stage, it computes the permutation for each element in the database. All these permutations are stored and they form the index. When a query *q* arrives, its permutation *Πq* is computed. Then, the elements in the database are sorted in increasing order of a similarity measurement between permutations, and next they are compared against the query *q* following this order, until some stop¬ping criterion is achieved. The similarity between two permutations can be measured, for example, by *Kendall Tau, Spearman Rho,* or S*pearman Footrule metrics* [21]. All of them are metrics, because they satisfy the aforementioned properties. We use the Spearman Footrule metric because it is not expen¬sive to compute and according to the authors in [2] it has a good performance to predict proximity between elements. The Spearman Footrule distance is the Manhattan distance L_1, that belongs to the Minkowsky's distances family, between two permu-tations. Formally, Spearman Footrule metric *F* is defined as: $F(\Pi_x,\Pi_q)=\Sigma_{i=1..m} |\Pi^{-1}(p_i)-\Pi^{-1}(p_i)|$.

This distance $F(\Pi_x,\Pi_q)$ can be computed in O *(m)* time [21]. Therefore, during preprocessing phase we first compute the *m*n* distances $d(x,p_i)$, and then compute and sort all the permutations for each element x in the database. This stage costs *O(nm logm)* additional time, and requires *O(n*m log m)*bits to store the whole index.

At query time we first compute the real distances $d(q,p_i)$ for every p_i P, then we obtain the permutation Πq , and next we sort the elements *x U* into increasing order according to $F(\Pi_x,\Pi_q)$ (the sorting can be done incrementally, because only some of the first elements are actually needed). Then *U* is traversed in that sorted order, evaluating the distance *d(q,x)* for each $x \in U$. For range queries, with radius *r*, each *x* that satisfies $d(q,x) \leq r$ is reported, and for *k*-NN queries the set of the *k* smallest distances so far, and the corresponding elements, are maintained.

```
RangeQuery(element q, radius r, DB fraction f)
1. Let A[1,n] be an array of tuples
2. Let U = {x1,...,xn}
3. Compute Π~q
4. For i ← 1 to n do
5.      A[i]←<xi ,F (Π~q , Π~i )>
6. /*Sorting by second component of tuples*/
7. SortIncreasing(A)
8. For i ← 1 to fn do
9.      <x, s>← A[i]
10.     If d(q, x)≤ r Then Report x
```

Algorithm 1: Range query of q with radius r in a permutation index.

Algorithm 1 shows the process for a range query. The efficiency and the quality of the answer ob¬viously depend on f: as f grows the efficiency degrades, but the answer quality improves. A way to obtain good values for the database fraction f is discussed in [2].

5. GPU-CUDA Permutation Index

The Figure 1 shows the GPU-CUDA system to work with a permutation index: the processes of indexing and querying. The Indexing process has two stages and the Querying process four steps. In this last process, we pay special attention to one step: the sorting. The next sections detail the characteristics of each process, their steps and peculiarities.

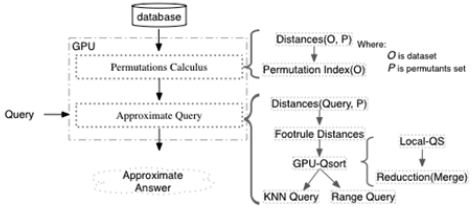


Figure 1. Indexing and Querying in GPU-CUDA permutation index.

5.1. Building the Permutation Index

Building a permutation index in GPU involves at least two steps. The first step calculates the distance among every object in database and the permutants. The second one sets up the signatures of all objects in database, i.e. all object permutations. The process input is the database and the permutants. At process end, the index is ready to be queried. The idea is to divide the work in threads blocks, each thread calculates the permutation object according to a global permutants set.

In the first task *(Distances(O,P))*, the number of blocks will be defined according of the size of the database and the number of threads per block which depends of the quantity of resources required by each block. At the step end, each threads block save in device memory its calculated distances. This stage requires a structure of size $m * n$ (*m:* permutants number and *n*: database size) and an auxiliar structure of fixed size defined in the shared memory of block (It stores the permutants, if the permutants size is greater than auxiliar structure size, the process is repeated until all distances to permutants are calculated).

The second step *(Permutation Index(O))* takes all calculated distances in the previous step and deter-mines the permutations of each object in database: its signature. To stablish the object permutation, each thread considers an object in database and sorts the permutants according to their distance. The output of second step is the *Permutation Index*, which is saved in device memory. Its size is $n * m$.

5.2. Solving Approximate Queries

The permutation index allows to answer to

all kinds of queries in approximated manner. Queries can be *"by range"* or *"k-NN"*. This process implies four steps. In the first, the permutation of query object is computed. This task is carried out by so many threads as permutants exist. The next step is to contrast all permutations in the index with query permutation. Comparison is done through the *Footrule* distance, one thread by object in database. In the third step, it sorts the calculated *Footrule* distances. As sorting methodology, we implement the Quick-sort in the GPU, its characteristics are explained bellow. Finally, depending of query kind, the selected objects have to be evaluated. In this evaluation, the *Euclidean distance* between query object and each candidate element is calculated again. Only a database percentage is considered for this step, for example the 10% (it can be a parameter). If the query is by range, the elements in the answer will be those that their distances are less than reference range. If it is *k*-NN query, once that each thread computes the *Euclidean distance*, all distances are sorted (using GPU-Qsort) and the results are the first k elements of sorted list.

GPU-Qsort carries out the task into two stages: *Local-Qsort* and *Merge-Reduction*. The first stage, **Local-Qsort**, has a data sequence as input and its output are *N* sorted subsequences. Each subsequence is ordered by a threads block according to iterative quicksort. Therefore, there are N threads blocks, where the number of threads by block is fix and is determined in relation to the required resources by block. Each block chooses a local pivot (it has to belong to input data list of block) and divides the data sequence in two subsequences: one has the elements smaller than pivot and another has the elements greater or equal than pivot. The pivot is the median among three elements of data subsequence: the first, middle and last element [22]. Each block works independently of other blocks eliminating the need of synchronization among threads of different blocks. In base to the selected pivot, all elements lower than the pivot are moved to a position to the pivot's left, and the greater or equal are shifted to the pivot's right. The task is made by using shared memory and each thread can determine itself the position for its element in shared structure (using CUDA atomic functions).

The process is applied iteratively over two subsequences. It is possible if it uses a stack. The stack saves all subsequences that still remain to be sorted. When there are two ready subsequences to work, one is selected and the other is pushed in the stack. If one subsequence is sorted, the subsequence in the top of stack is selected to work. The iterative process ends when the stack is empty and the list is sorted. When the number of elements in the sequence is lower than eight, it is sorted in sequential manner, because the process overhead is too large compared to sequence size.

At the end of stage, each one of *N* blocks copies its sorted subsequence to device memory. The output is *N* sorted subsequence.

For second stage of GPU-Qsort, **Merge-Reduction**, its input is N sorted list and the output is whole sorted sequence. This phase makes a reduction; the reduction operation is a merge of sorted lists. A block merges two lists at a time. In consequence, $log_2 N$ iterations are necessary to find the final result. This stage requires [*N/2*] blocks with thirty-two threads per each and an auxiliary structure in device memory.

In both stages, different techniques are design to optimize the performance; they are

use for shared memory, anticipatory copies and coalesced access to global memory.

6. Solving Multiple Queries

In large-scale systems such as Web Search Engines indexing multimedia content, it is critical to deal efficiently with streams of queries rather than with single queries. Therefore, it is not enough to speed up the time to answer only one query, but it is necessary to leverage the capabilities of the GPU to parallel answer several queries. So we have to show how to achieve efficient and scalable performance in this context. We need to devise algorithms and optimizations specially tailored to support high-performance parallel query processing in GPU. GPU has characteristics of software and hardware which allow us to think in to solve many approximated queries in parallel. For this, the represented system in Figure 1 is modified and it is shown in Figure 2. In this Figure, it can be observed that the permutation index is built once and then is used to answer many queries.

In order to answer parallely many approximate queries, GPU receives the queries set and it has to solve all of them. Each query, in parallel, applies the process explained in 5.2, therefore the number of needed resources for this is equal to the resources amount to compute one query multiplied the number of queries solved in parallel. The number of queries to solve in parallel is determined according to the GPU resources mainly its memory. If Q are parallel queries, M the needed memory quantity per query and I the needed memory by permutation index, $Q^* M + I$ is the total required memory to solve Q queries in parallel. Once the q parallel queries are solved, the results are sent from the GPU to the CPU through a single transfer via PCI-Express.

Solving many queries in parallel involves carefully manage the blocks and their threads. At the same time, blocks of different queries are accessed in parallel. Hence it is important a good administration of threads: which query it is solved and which database element it is responsible. The task is possible by establishing

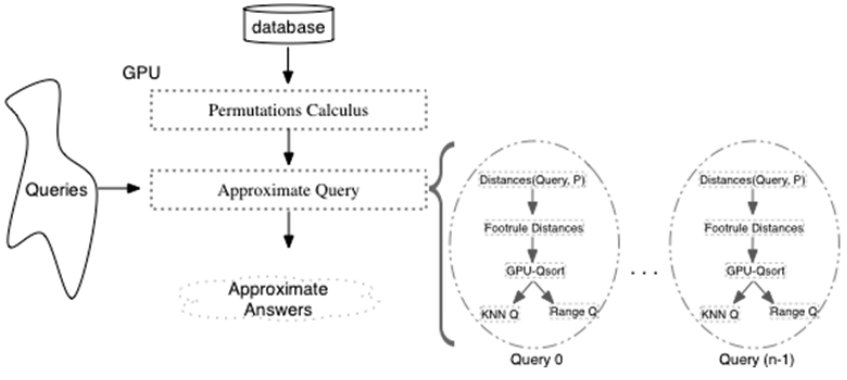


Figure. 2. Solving q queries in GPU-CUDA permutation index

a relationship among *Thread Id, Block Id, Query Id, and Database Element.*

7. Experimental Results

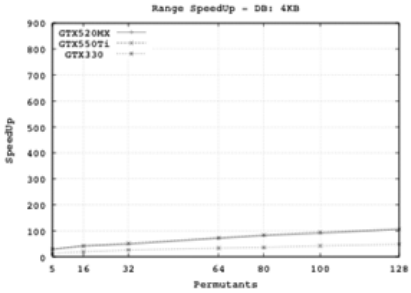
Our experiments considered different database sizes: 4KB, 29KB, 68KB, and 84KB, on a metric database consisting of English words and using the Levenshtein distance, also called edit distance edit (that is, the minimum number of character insertions, deletions, and substitutions needed to make two strings equal). The analysis was made for three GeForce GPU whose characteristics (Global Memory, SM, SP, Clock rate, Compute Capability) are GTX330: (512MB, 6, 48, 1.04GHz, 1.2), GTX520MX: (1024MB, 1, 48, 1.8GHz, 2.1) and GTX550Ti: (1024MB, 4, 192, 1.96GHz, 2.1). The CPU is an Intel core i3, 2.13 GHz and 3 GB of memory. The results are expressed in Speed up $(Sp=Time_{Sec}/Time_{Par})$.
In this paper, we do not display the speed up of construction of *Permutation Index*. These results are illustrated in [23].

Figures 3 and 4 show the obtained acceleration in range queries (3) and *k*-NN (4) queries for three database sizes and different number of permutants. In these results, 80 queries are solved in parallel. As it can be noticed Range queries show improve¬ments respect to *k*-NN queries, but in both cases the achieved speed up is very good. In all cases, it is clear the influence of database size, but evenly we accomplish good performance. The best case is for largest database and maximum number of permutants.

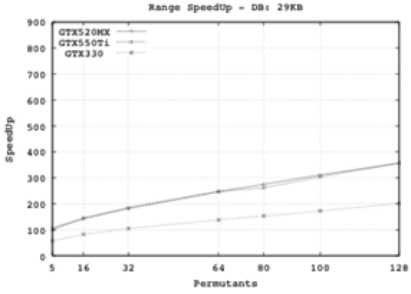
Tables 1 and 2 show the obtained throughput (number of queries by second) by our implementa¬tion. The results clearly show the benefits for all used architectures of GPU. In every case and query kind, the number of queries by second is high.

Also we can observe that the number of permutants is not so important, for each GPU architecture and all number of permutants, the throughput is similar, quasi constant.

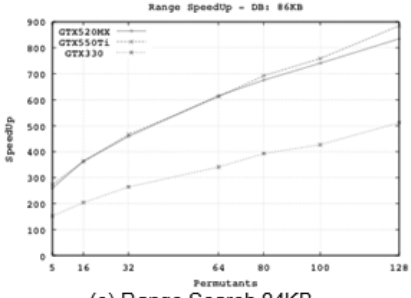
Figures 5 and 6 resume the behavior of two operations: Range Query (Figure 5) and *k*-NN Query (Figure 6), for biggest database and three numbers of permutants (5, 64, 128) when we vary the number of parallel queries in three GPUs. It can be seem that the best speed up was obtained when the number of queries is equal to 80 and the number of permutants is the maximum. Also it is clear the influence of GPU architecture, when it has more resources, better speed up are achieved. Figures 5(b) and 6(b) depict these results.



(a) Range Search 4KB

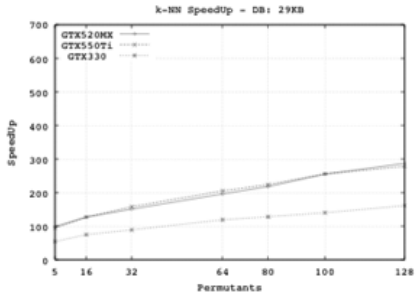


(b) Range Search 29KB

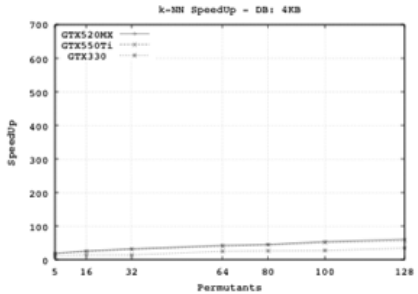


(c) Range Search 84KB

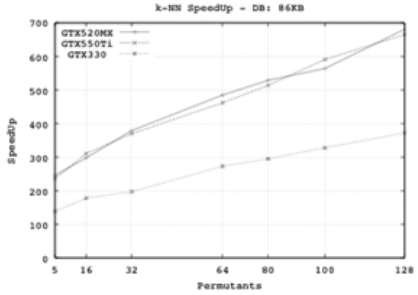
Figure. 3. Speedup of Range search Queries on three different GPUs.



(a) k-NN Search 4KB



(b) k-NN Search 29KB

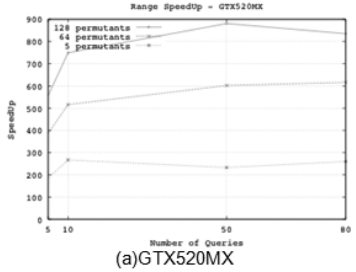


(c) k-NN Search 84KB

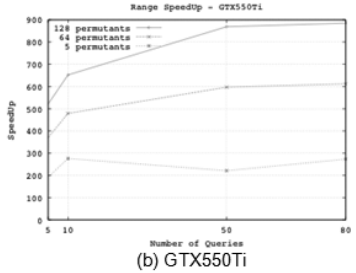
Figure. 4. Speedup of *k*-NN search Queries on three different GPUs.

Table I: range search throughput

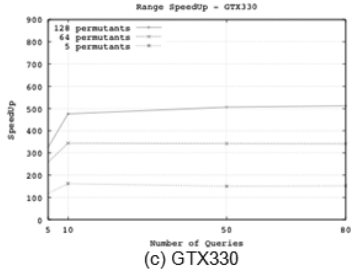
# Permut.	GTX520MX	GTX550Ti	GTX330
128	27639,72	29310,63	16973,21
100	28188,86	28907,35	16279,76
80	28921,82	29621,19	16828,75
64	29539,57	29362,77	16379,24
32	29144,71	29582,42	16774,19
16	29255,39	29248,81	16464,29
5	28197,27	29604,32	16474,46



(a)GTX520MX



(b) GTX550Ti

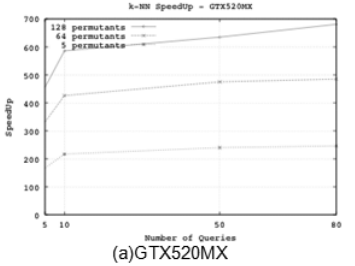


(c) GTX330

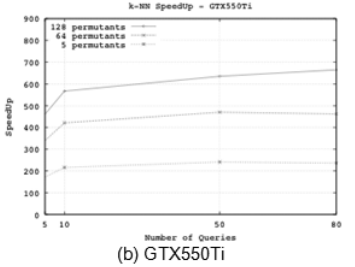
Figure 5. Speedup of Range Search Queries for different number of parallel queries.

Table 2: k-NN search througput

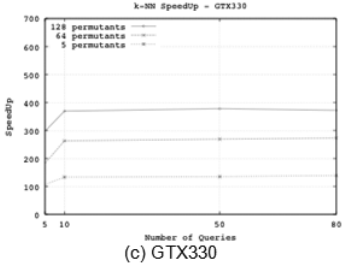
# Permut.	GTX520MX	GTX550Ti	GTX330
128	19824,25	19377,68	10850,85
100	18816,38	19696,23	10956,71
80	19771,86	19203,07	11051,72
64	19797,83	18857,32	11137,65
32	19774,12	19289,62	10263,80
16	18785,79	19645,99	11237,29
5	19906,59	19121,16	11262,48



(a)GTX520MX



(b) GTX550Ti



(c) GTX330

Figure 6. Speedup of *k*-NN Search Queries for different number of parallel queries.

7.1. GPU-Qsort with Other Solution

There are many quick sort libraries, one of them is *Thrust*. It is part of CUDA repositories. *Thrust* library provides a collection of fundamental parallel algorithms such as *scan, sort and reduction.* It solves a complementary set of problems, namely those that are (1) implemented efficiently without a detailed mapping of work onto the target architecture or those that (2) do not merit or simply will not re¬ceive significant optimization effort by the user. With this library, developers describe their computation using a collection of high-level algorithms and com¬pletely delegate the decision of how to implement the computation to the library. This abstract interface allows programmers to describe what to compute without placing any additional restrictions on how to carry out the computation [24]. A disadvantage of *Thrust* is that it can isolate the developer from the hardware and expose only a subset of the hardware capabilities. In some circumstances, the C++ interface can become too awkward or verbose [25].

We compared our implementation with a solution based in *Thrust* library. We used the Thrust as a black box. Figure 7 shows the comparison considering four DB size. The results are the average of one hundred executions.

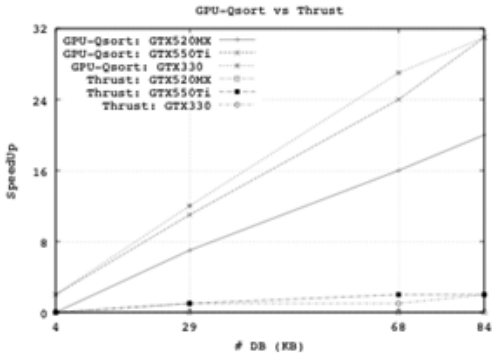


Figure 7. Speedup of GPU-Qsort and Thrust on three different GPUs

In the figure, we can observe that our implementation obtains better speed up than the solution using Thrust library. Besides it is important to notice the independence of GPU-Qsort from GPU characteristics, it works fine in all GPU.

8. Conclusions

As it is mentioned before, in large-scale Systems such as Web Search Engines indexing multimedia content, it is critical to deal efficiently with streams of queries rather than with single queries. Therefore, it is not enough to speed up the time to answer only one query, but it is necessary to solve several queries at the same time. In this work we present a solution to solve many queries in parallel taking advantage of GPU architecture: it is a massively parallel architecture, it has a high throughput because its capacity of parallel processing for thousands of threads.

In this work we show an implementation that uses a *Pemutation Index* to solve approximate similarity search on a database of words.

However, it is possible to easily extend our proposal to other metric databases of different data nature, such as vectors, documents, DNA sequences, images, music, among others.

The empirical results have shown improvements in every different GPU architecture considered. Both obtained speed up and throughput are very good, showing better performance when the load work is hard.

In the future, we plan to make an exhaustive experimental evaluation considering others kinds of database, comparing with other solutions that apply GPU in the scenario of metric space approximate searches. We need also to evaluate retrieval effectiveness of the answer of the *Permutation Index*, as the number of objects directly compared with the query grows, by using *Recal*l and Precision measures.

9. Acknowledgements

We wish to thank to the UNSL for allowing us the access to their computational resources. This research has been partially supported by Project UNSL-PROICO-30310 and Project UNSL-PROICO-330303.

10. References

[1] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín, "Searching in metric spaces," ACM Comput. Surv., vol. 33, no. 3, pp. 273–321, 2001.

[2] E. Chávez, K. Figueroa, and G. Navarro, "Proximity searching in high dimensional spaces with a proximity preserving order," in Proc. 4th Mexican International Conference on Artificial Intelligence (MICAI), ser. LNAI 3789, 2005, pp. 405–414.

[3] D. B. Kirk and W. W. Hwu, Programming Massively Parallel Processors, A Hands on Approach. Elsevier, Morgan Kaufmann, 2010.

[4] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "GPU Computing," in IEEE, vol. 96, no. 5, 2008, pp. 879 – 899.

[5] R. Barrientos, J. Gomez, C. Tenllado, and M. Prieto, "Heap based k-nearest neighbor search on gpus," 09/2010 2010, pp. 559–566.

[6] B. Bustos, O. Deussen, S. Hiller, and D. Keim, "A Graphics hardware accelerated algorithm for nearest neighbor search," in Proc. International Conference on Computational Science (ICCS'06) Part IV, ser. LNCS, vol. 3994. Springer, 2006, pp. 196–199.

[7] V. Garcia, E. Debreuve, F. Nielsen, and M. Barlaud, "knearest neighbor search: fast GPU-based implementations and application to high-dimensional feature matching," in IEEE International Conference on Image Processing, Hong Kong, Sept. 2010.

[8] K. Kato and T. Hosino, "Solving k-nearest neighbor problem on multiple graphics processors," in 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID, ACM, Ed., 2010, pp. 769–773.

[9] S. Liang, Y. Liu, C. Wang, and L. Jian, "Design and evaluation of a parallel k-nearest neighbor algorithm on CUDA-enabled GPU," in IEEE 2nd Symposium on Web Society (SWS), 2010, pp. 53 – 60.

[10] R. J. Barrientos, J. Gomez, C. Tenllado, M. Prieto, and M. Marin, "kNN Query Processing in Metric Spaces using GPUs," vol. 6852, 2011, pp. 380–392.

[11] R. Uribe-Paredes, P. Valero-Lara, E. Arias, J. L. Sánchez, and D. Cazorla, "A GPU-Based Implemen-

tation for Range Queries on Spaghettis Data Structure," in ICCSA (1), ser. Lecture Notes in Computer Science, vol. 6782. Springer,
2011, pp. 615–629.

[12] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity Search: The Metric Space Approach, ser. Advances in Database Systems, vol.32. Springer, 2006.

[13] H. Samet, Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[14] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes, "Speeding up spatial approximation search in metric spaces," ACM Journal of Experimental Algorithmics, vol. 14, p. article 3.6, 2009.

[15] P. Ciaccia and M. Patella, "Approximate and probabilistic methods," SIGSPATIAL Special, vol. 2, no. 2, pp. 16–19, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1862413.1862418

[16] M. Patella and P. Ciaccia, "Approximate similarity search: A multi-faceted problem," J. Discrete Algorithms, vol. 7, no. 1, pp. 36–48, 2009.

[17] B. Bustos and G. Navarro, "Probabilistic proximity searching algorithms based on compact partitions," Discrete Algorithms, vol. 2, no. 1, pp. 115–134, Mar. 2004. [Online]. Available: http://dx.doi.org/10.1016/S1570- 8667(03)00067-4

[18] A. Singh, H. Ferhatosmanoglu, and A. Tosun, "High dimensional reverse nearest neighbor queries," in The twelfth international conference on Information and knowledge management, ser. CIKM '03. New York, NY, USA: ACM, 2003, pp. 91–98. [Online]. Available: http://doi.acm.org/10.1145/956863.956882

[19] F. Moreno-Seco, L. Mic´o, and J. Oncina, "A modification of the laesa algorithm for approximated k-nn classification," Pattern Recognition Letters, vol. 24, no. 1-3, pp. 47 – 53, 2003.

[20] NVIDIA, "Nvidia cuda compute unified device architecture, programming guide version 4.2." in NVIDIA, 2012.

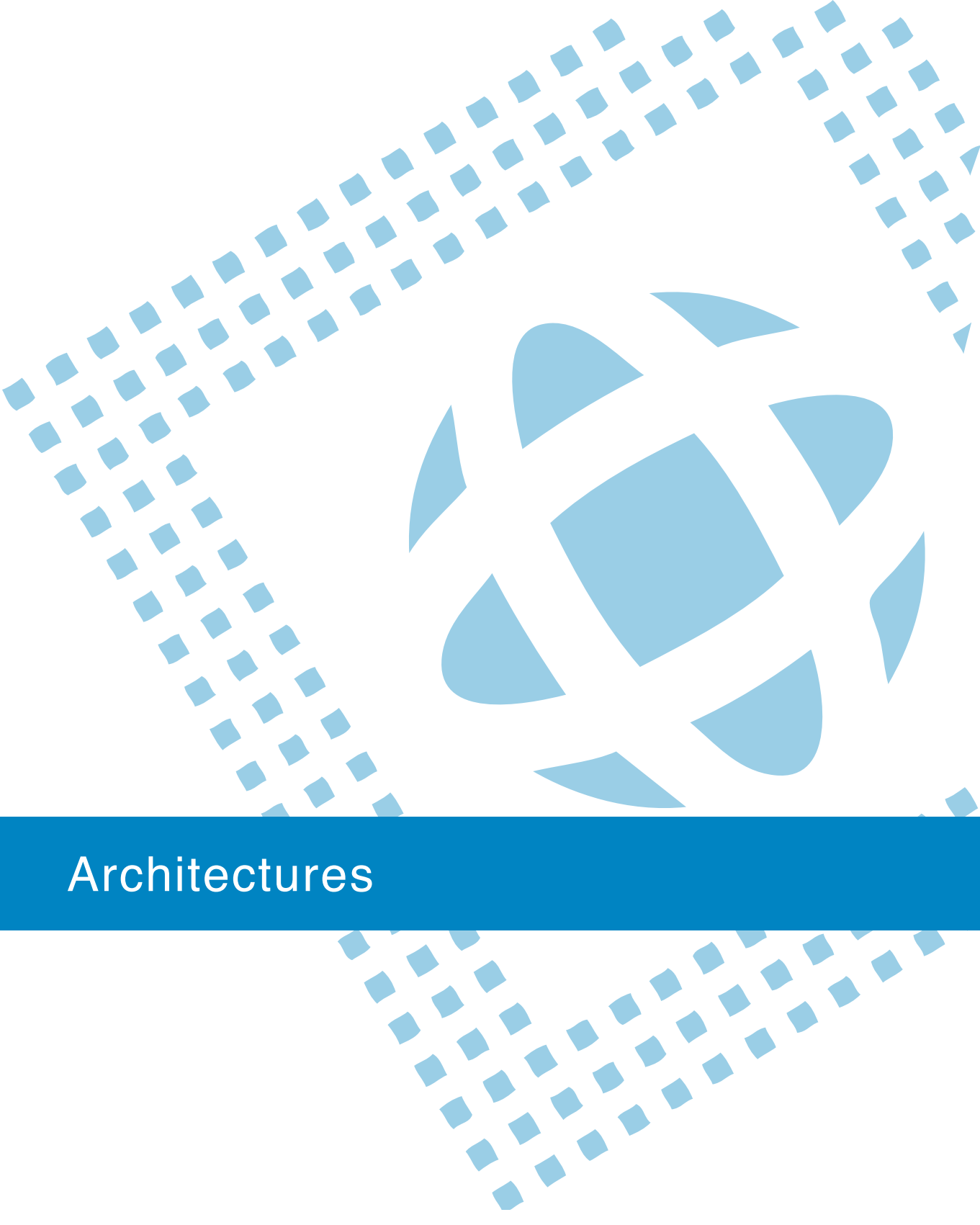
[21] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," in Proceedings of the fourteenth annual ACMSIAM symposium on Discrete algorithms, ser. SODA '03. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 28–36. [Online]. Available: http://dl.acm.org/citation.cfm?id=644108.644113

[22] R. Singleton, "Algorithm 347: an efficient algorithm for sorting with minimal storage [m1]," Commun. ACM, vol. 12, no. 3, pp. 185–186, Mar. 1969.

[23] M. Lopresti, N. Miranda, F. Piccoli, and N. Reyes, "Efficient similarity search on multimedia databases," in XVIII Congreso Argentino de Ciencias de la Computación, CACIC 2012, 2012, pp. 1079–1088.

[24] J. Hoberock and N. Bell, "Thrust: A parallel template library," 2010, version 1.3.0. [Online]. Available: http://www.meganewtons.com/

[25] R. Farber, CUDA Application Design and Development, 1st ed. San Francisco, CA, USA: Morgan Kaufmann
Publishers Inc., 2011.



Architectures

Autonomous Recovery Technology for Fault Tolerance in Distributed Service-Oriented Mission Critical Systems

Raymundo García-Gómez, Juan Sebastián Guadalupe Godínez-Borja, Pedro Josué Hernández-Torres, Carlos Pérez-Leguízamo
Banco de México
Av. 5de Mayo #2, D.F., 06059, MEXICO
E-mail: {rgarciag, jgodinez, pjhernan, cperez}@banxico.org.mx

Abstract

Mission Critical Systems (MCS) require continuous operation since a failure might cause economic or human losses. Autonomous Decentralized Service Oriented Architecture (ADSOA) is a proposal to design and develop MCS in which the system functionality is divided into service units in order to provide functional reliability and load balancing; on the other hand, it offers high availability through distributed replicas. A fault detection technology has been proposed for ADSOA. In this technology, an operational service level degradation can be detected autonomously by the service units at a point in which the continuity of the service may be compromised. However, this technology is limited because it requires human supervision for recovery. In this paper, we propose an autonomous recovering technology, which detects and instructs to service units to be gradually cloned in order to recover the operational service level. A prototype has been developed in order to verify the feasibility of this technology.

Keywords: Service continuity; fault tolerance; service-oriented architecture; autonomous decentralized systems; fault detection; fault recovery

1. Introduction and motivation

In the presence of a failure, most of the conventional systems implement reactive fault detection and recover mechanisms either automatically or manually. In both cases, the aim is to switch to a redundant or standby computer server upon the failure or abnormal termination of the previously active system. In some cases, the Mean Time to Recovery (MTTR) [15] of these technologies may represent a low risk for the service that the system offers. However, since a failure in MCS may provoke fatal consequences, it is important to reduce the MTTR to a value near zero.

In this paper, we briefly present ADSOA [4] [5][6], which has been proposed as a service-oriented architecture for designing MCS, and it has been mainly utilized in financial sector applications. This architecture provides high functional reliability since it is possible to distribute and replicate the functionality of

a system in specialized service units. One of the main technologies of ADSOA, called Loosely Coupled Delivery Transaction and Synchronization Technology [6], allows the system to detect when the provision of a service has reached a point in which the continuity of the service may be compromised and it sends a signal alarm to a monitor. This approach may represent a risk for a MCS since it depends on human intervention for taking the necessary actions to repair the system.

This has motivated this paper which presents a technology to autonomously detect and recover gradually all the unit services required for the operational service level in ADSOA. This technology is based on a cloning mechanism that is activated once the operational service level has been compromised due to some failed services units. We describe the protocol and algorithms that the healthy services units utilize in this cloning mechanism and show how they coordinate among them in order to avoid a massive creation of replicas. We developed a prototype in order to illustrate this approach.

The rest of this paper is organized as follows: In Section II, we show the related work. In Section III, we give a view of ADSOA concept and architecture. In Section IV, we present the proposed technology. In Section V, we show a prototype, and finally, in Section VI, the conclusion and future work.

2. Related work

Cloning technologies have been widely used in different technological areas for providing high reliability to the system in which it is applied. In Optical Burst Switching (OBS) Networks, burst cloning has been proposed as a proactive loss recovery mechanism that attempts to prevent burst loss by sending two copies of the same burst, if the first copy is lost, the second copy may still be able to reach the destination [9][10]. When designing cloning technologies one relevant issue that has to be considered is the resource utilization by the new clones. In this sense, in OBS Networks some technologies have been proposed for optimizing such resource utilization and maintaining a QoS [11][12]. In Multi Agents Systems (MAS), a frequently proposed solution to avoid performance bottlenecks due to insufficient resources is cloning an agent and migrate it to remote hosts [13][14].

Our approach is also comparable to the existing work on cloning technologies in terms of concept and objectives but applied to a novel service-oriented architecture for MCS. The main contribution of the proposed cloning technology are the protocol and algorithms that services units utilize to detect some failures in the service provision and the way they coordinate themselves to recover gradually the operation of that damaged part of the system.

3. Autonomous Decentralized Service Oriented Architecture

3.1. ADSOA concept

A proposal used to implement MCS in financial sector is ADSOA [4][5][6], it provides load balancing and functionality, high availability and service-oriented modeling. ADSOA is based on Autonomous Decentralized Systems (ADS) [1][2][3] and Service Oriented Architecture (SOA)[7][8].

ADS is based in the concept of analogizing living things. The living body is built of numerous cells, each growing up independently,

maintaining by body metabolism, and going on living. Depending on concept of perceiving the system that it is consisted by autonomous controllability and autonomous coordinability, ADS technology achieves fault tolerance, on-line maintenance and on-line expansion of the computer system. On the other hand, SOA offers a system modeling oriented to services and allows composition and reusability. ADSOA is the combination of SOA concept with ADS characteristics.

The ADSOA conceptual model, shown in Figure 1, is composed of autonomous entities that offers or requests services via messages. Each entity is formed by several instances fully independent. Each instance has the same functionality that its entity represents. A subsystem can be formed by a group of entities and in the same sense a business may be formed by a group of subsystems. This is similar to a living organism where an instance is like a cell, a subsystem could be an organ and the business is like a living organism.

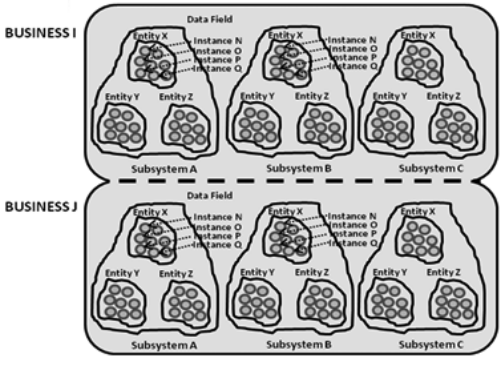


Figure 1. ADSOA conceptual model

In order to model a MCS using ADSOA, it is necessary to have a service-oriented thinking. At the beginning the system architect identifies the businesses involved in the process and then models the sub-systems in a business according to their responsibility. Finally, entities are modeled according to their atomic functionality. This modeling will allow to the system to grow, evolve, do composition and reuse the components. The next phase is to develop the services entities.

All the systems immersed in ADSOA are able to configure according to physical resources and criticality level. To offer high service availability, it is necessary to have a distributed environment and put on replicated entities. On the other hand, for load balancing it is necessary to divide the functionality in the entities, in such a way that the work be split without a coordinator. The challenge is to provide auto-coordination and auto-control to the system. In this sense, the Autonomous Processing Entity (APE) was proposed; it implements the communication protocol, manages the control instance messages and the services execution. Also, it is possible to define in each service (offered or requested) of the APE its criticality. All these elements form a technology denominated “Loosely Coupling Synchronization and Transactional Delivery Technology”.

3.1. Loosely Coupling Synchronization and Transactional Delivery Technology

In this technology, we define the concept of transaction in the scenario in which an entity requests a service to another and requires knowing if it has been received. The requesting entity must maintain this request in pending processing state until it receives an acknowledgement from receiving entity. Also,

we define sequential order in the sense that the entity requester must receive a minimum number of acknowledgments from receiving entities in order to send the next service request, for example, a X+1 request should not be sent until it receives the minimum number of acknowledgments of the X request.

The service request information structure should include the following elements: Content Code, Foliated Structure and Request Information.

The Content Code specifies the content and defines the requested service.

The foliated structure identifies the transaction. This structure is based on:

1. requester id,
2. specialized task id for that request (Pivot),
3. a sequence number,
4. a generated id based on the original request information (event number) and
5. a dynamic and unique id for the instance of the entity (instprintid).

With these elements the identification of acknowledgments received by the entity is guaranteed. We can also ensure the sequence of multiple requests, as shown in Figure 2.

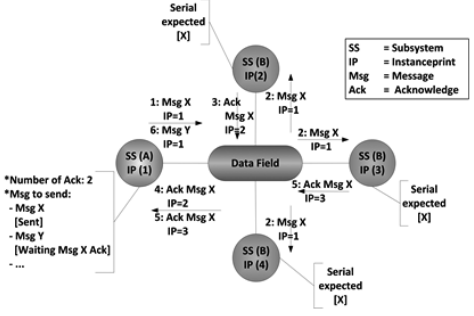


Figure 2. Sequentiality and transactionality

If an instance receives a service request with a sequence number greater than expected, then by the principle of sequential order, knows that another instance of its entity will have the missing messages. In this case, the receiver instance asks to his entity the missed messages, that is, the other instances of the same entity. This idea is represented in Figure 3.

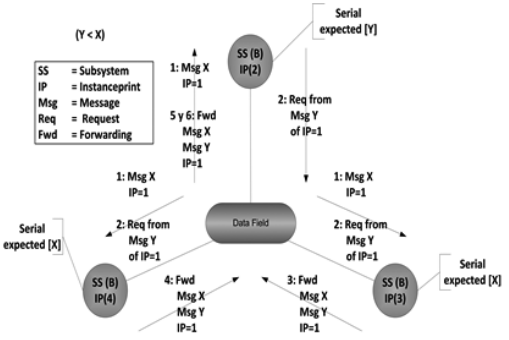


Figure 3. Synchronization with other instances.

On the other hand, if an entity receives several times the same service request, this can be distinguished by the instprintid if this request belongs to the same requester instance or from a different instance of the same entity. According to this, the receiver entity can determine whether requests received are in accordance with the minimum number of requests that the requester entity are required to send, as shown in Figure 4.

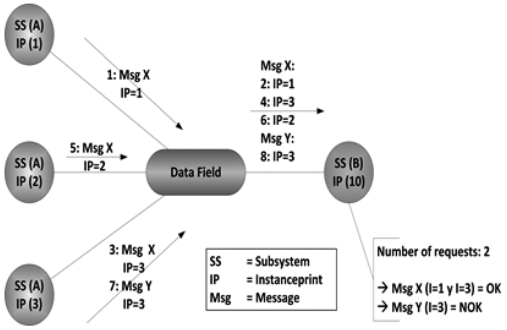


Figure 4. Receiving multiple requests from an entity.

In resume, the communication among the elements and its instances is based on an asynchronous and event-driven message protocol. This technology detects if an entity does not provide the service level required. It occurs when an instance sends a service request to an entity and each entity instance receives it and send back an acknowledge message, then sender registers how many acknowledges have received and evaluates if it covers the criticality level, if it is not proper, the sender repeats the sending process. E.g. consider a service with a criticality level equals to 3, its means that this business requires at least three distributed instances; when another instance requests a service to them, it expects at least three acknowledges to satisfy the criticality level, if it is not satisfied the entity will send the request of service again. When the sender detects that the maximum number of retries has reached, it triggers the alert process, which consists in sending an alert message that could be processed by a monitor element. This monitor alerts ADSOA infrastructure managers to perform the necessary activities and recover service continuity (creating new instances required to reach the criticality level). Unfortunately, this goes against MCS's principles since manual intervention is required thereby MTTR becomes dependent on operator's reaction.

In the next section, we present a technology that allows ADSOA subsystems to autonomously detect and recover for a failure in a replicated entity by cloning one by one an operational entity until the system reaches the criticality level required.

4. Autonomous Recovery Technology for Fault Tolerance in ADSOA

This technology is created to allow an MSC that uses an ADSOA infrastructure to self-recover automatically. This basic operation is to use the current self-monitoring scheme and instead of sending alerts to the operator when the service level is not appropriate, it instructs one entity of the degraded group to clone itself (functionality and state). An important challenge in the cloning process is to avoid the generation of multiple indiscriminate copies, which in a living organism would be a cancer. To ensure the healthy recovery, the entity selected to recover the system, generates a cloning-key with information of the times it has been cloned, its id, its instprintid and the requested entity id; this information is introduced into the algorithm to generate the cloning-key, that will be unique to only one cloning process between this entity and the requested id.

In this architecture, all the entities offer and request services, one of these services are the recovering by cloning an entity. In self-recovering technology at least two entities are involved; to explain the protocol let's imagine a group of entities ("A subsystem"), which request a service to other group of

entities ("B subsystem"). In Figure 5, "A subsystem" is requesting a service to "B subsystem", the message exchange is carried out in compliance with ADSOA Loosely Coupling Synchronization and Transaction Delivery Technology, with the number of acknowledgments needed to ensure that the level of service is appropriate for. In this example, the "A subsystem", requires 3 acknowledgments by "B subsystem", and the "B subsystem" needs 2 service requests by "A subsystem"; when the number of acknowledgments is complying, "B subsystem" attends "A subsystem".

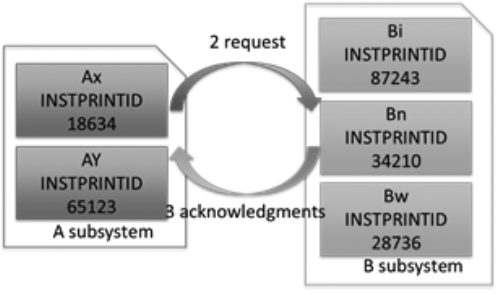


Figure 5. Normal operation cycle.

If the number of the entities of the "B subsystem" is decreased because of a failure in the process or the server, the "A entities" detects that the service level is not complying within "B subsystem", since the minimum number of acknowledgments, 3 for this example, cannot be reached within a specific period of time. Thus, the "A entities" starts the recovery mechanism instead of sending alerts.

Figure 6 shows the first steps in the recovery protocol. Firstly, the "A subsystem" receives the acknowledgments from "B subsystem". Secondly, based in the lowest "B"'s instprintid all the "A entities" select one healthy entity, which will be responsible for cloning itself. Thirdly, all the "A entities" request the "Auto-Cloning Service (reqidclon)", with the instprintid of the "B entity" selected for auto-cloning. In this example the "Bi entity" will be the responsible for cloning itself; although the "Bn entity" received the same request, only the "Bi entity" will clone. Fourthly, when the "Bi entity" receives the reqidclon request, it generates a cloning-key and sends both this cloning–key and its instprintid as a "Send the key (sendkey)" request service message. By sending its instprintid it can be ensured that the "A subsystem" will instruct to only the selected "B entity" to continue with the cloning process. Fifthly, when the "A subsystem" receives the sendkey service request, it takes the cloning-key in the message and sends it by the "Automatic recovery (autrecov)" request service message to the "B subsystem", it also attaches to this message the instprintid selected in the second step of this protocol.

Figure 7 shows the final step of the protocol. When the "B subsystem" receives the autrecov request service message, as it occurs in the third step of this protocol, only the "Bi entity" will attend it, since its instprintid is in the received service message. "Bi entity" will validate if the cloning-key in the message is still valid and if so it will make a cloning of itself.

During this process, "Bi entity" will close all the communication with outside and generate a new element in the same state like itself; once the cloning process is finished, it will open the communication again. Otherwise, if the cloning-key is not longer valid because the cloning process has already been completed, the message is ignored. It is important to notice, that the others "A entities" also send

this final request service message, but only the first message which reaches "Bi entity" will be processed.

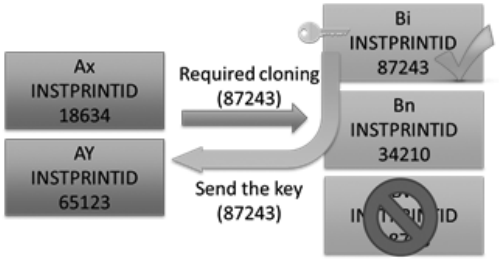


Figure 6. Start up the cloning mechanism.

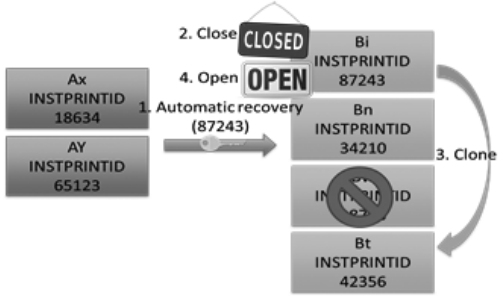


Figure 7. Cloning phase.

In this sense, this technology will autonomously maintain the operational service level without human intervention.

5. Prototype

A prototype which implements this technology has been developed, as shown in Figure 8. This prototype consists of two subsystems with one entity each one, the Requester subsystem/entity, which is shown in blue color, and the Counter subsystem/entity, which is shown in orange color. The Requester demands a service to the Counter for providing a number which later it will be displayed in its screen. When the Counter receives this service request, it will increase by one the previous sent number and send it into a service request message to the Requester. For this example, the service level operation was set to 1 to the Requester and 3 to the Counter; it means that there will be only 1 instance of the Requester and 3 instances of the Counter. On the other hand, the Requester will send its next service request only if it receives from the Counter instances 3 acknowledges for the current request. In order to simulate a failure in a Counter instance, a PAUSE button has been implemented.

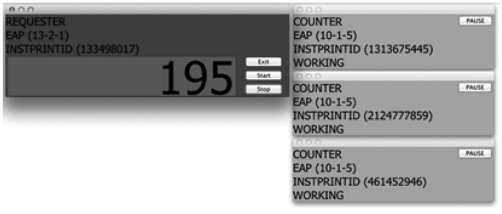


Figure 8. Normal operation cycle I (prototype).

In Figure 9, it is shown that when the entity "2124777859" is stopped, the cloning-mechanism detects such failure and selects entity "1313675445" to repair the system. The reqidclon service request message is sent from the Requester to the Counter with instprintid "1313675445". This entity generates the cloning-key and then it sends the sendkey service request message to the Requester.

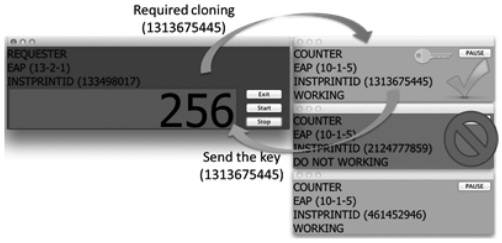


Figure 9. Start up the cloning mechanism (prototype).

In Figure 10, the final part of the mechanism is shown. The Requester processes the sendkey service request message and sends to the selected entity the autrecov service request message. The "1313675445" entity starts the cloning process; firstly it closes the communication with outside, then it clones itself, and when the entity "191232582" is started, the "1313675445" entity finally opens the communication.

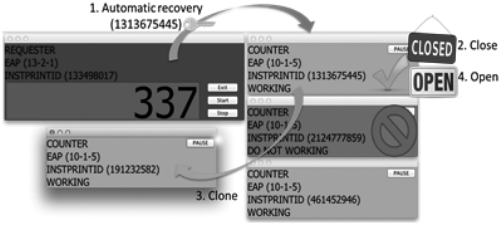


Figure 10. Normal operation cycle II (prototype).

In this sense, the system has repaired autonomously the damaged part and it can continue its normal operation.

6. Conclusion and Future Work

In this paper, we briefly presented ADSOA, which has been proposed as a service-oriented architecture for designing MCS, which has been mainly utilized in financial sector applications. We have proposed a cloning mechanism to recover quickly and efficiently the operational service level when a decrease on it is detected. We have built a prototype to verify the feasibility of this technology.

Besides the ongoing development efforts to complete the cloning prototype implementation, future work in this area focuses on getting some metrics about resource utilization, network partition and multiple clones' coexistence. We will also compare the proposed technology with others such as those mentioned in Section II.

7. References

[1] K. Mori, S. Miyamoto, and H. Ihara, "Proposition of Autonomous Decentralized Concept", Journal of IEEE Japan, vol. 104, no. 12, pp. 303-310, 1994.

[2] H. Ihara and K. Mori, "Autonomous Decentralized Computer Control Systems", IEEE Computer, vol. 17, no. 8, pp. 57-66, 1984.

[3] K. Mori, "Autonomous Decentralized Computer Control Systems", First International Symposium on Autonomous Decentralized Systems (ISADS'93), Kawasaki, Japan, pp. 28-34, 1993.

[4] L.C. Coronado-García and C. Pérez-Leguízamo, "A Mission-Critical Certification Authority Architecture for High Reliability and Response Time", IJCCBS Special Issue on Autonomous Decentralized Systems in Web Computing, vol. 2, no. 1, pp. 6-24, 2011.

[5] L.C. Coronado-García, P.J. Hernández-Torres,

and C. Pérez-Leguízamo, "An Autonomous Decentralized System Architecture using a Software-Based Secure Data Field", The 10th International Symposium on Autonomous Decentralized Systems (ISADS'11), Kobe, Japan, 2011.

[6] L.C. Coronado-García, P.J. Hernández-Torres, and C. Pérez-Leguízamo, "An Autonomous Decentralized Service Oriented Architecture for High Reliable Service Provision", The 10th International Symposium on Autonomous Decentralized Systems (ISADS'11), Kobe, Japan, 2011.

[7] Thomas Erl, 2005, "Service-Oriented Architecture (SOA): Concepts, Technology, and Design", Ed. Prentice Hall.

[8] Nicolai M. Josuttis, 2007, "SOA in Practice: The Art of Distributed System Design", Ed. O'Reilly Media.

[9] H. Xiaodong, V.M. Vokkarane, and J.P. Jue, "Burst cloning: a proactive scheme to reduce data loss in optical burst switched networks", IEEE International Conference on Communications (ICC'05), Seoul, Korea, 2005.

[10] S. Riadi and V-A. Mohammed, "A decision algorithm for efficient hybrid burst retransmission and burst cloning scheme over star OBS networks", Second International Conference on Innovating Computing Technology (INTECH'12), Casablanca, Morocco, 2012.

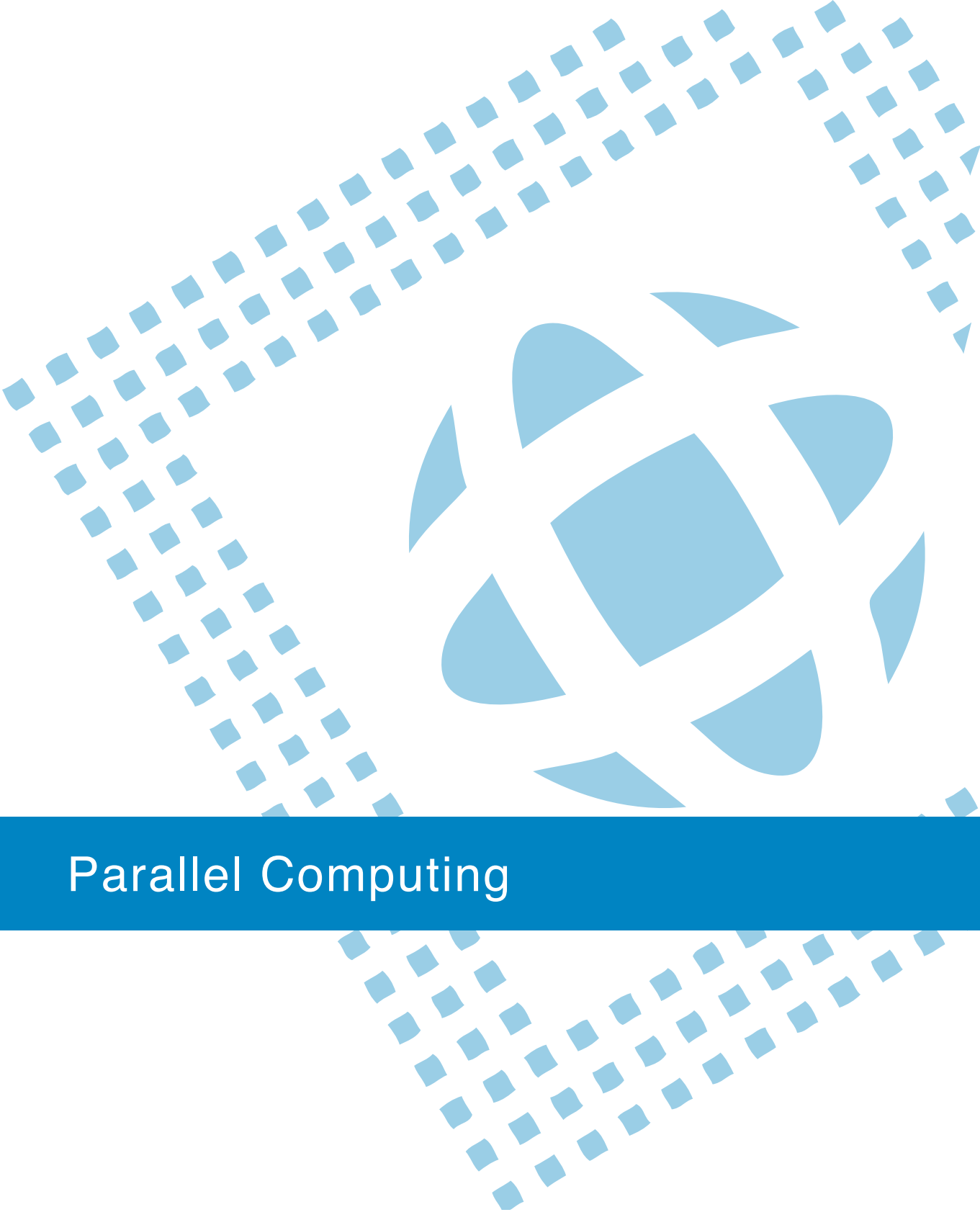
[11] L. Ji and K.L. Yeung, "Burst cloning with load balancing", Optical Fiber Communication Conference (OFC'06), Anaheim, California, 2006.

[12] S. Askar, G. Zervas, D.K. Hunter, and D. Simeonidou, "Classified cloning for QoS provisioning in OBS networks", The 36th European Conference and Exhibition on Optical Communication (ECOC'10), Turin, Italy, 2010.

[13] O. Shehory, K. Sycara, P. Chalasani, and S. Jha, "Agent cloning: an approach to agent mobility and resource allocation", IEEE Communications, vol. 36, no. 7, pp. 63-67, 1998.

[14] D. Ye, M. Zhang, and D. Sutanto, "Cloning, Resource Exchange and Relation Adaptation: An Integrative Self-Organisation Mechanism in a Distributed Agent Network", IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99, pp. 1, 2013.

[15] P.A. Laplant and S.J. Ovaska, 2011, "Real Time Systems Design and Analysis", Ed. Wiley-IEEE Press.


Parallel Computing

Calculation of Vibration Modes and Optimization of Structures Subject to Seismic Actions

Maximino Tapia[1]*, Salvador Botello[1], Jacob Salazar[1,2], Ernesto Ortega[1,2], Iván Munguía[1]

1: Centro de Investigación en Matemáticas (CIMAT)

Callejón Jalisco S/n, Mineral de Valenciana, C.P. 36240, Guanajuato, Gto., México.

e-mail: {max,botello, jacobess, ortegae,munguia} @cimat.mx, web: www.cimat.mx

2: Civil Engineering Department

Engineering Division

Universidad de Guanajuato

Av. Juárez No.77, Zona Centro, C.P. 36000, Guanajuato,Gto., México.

web: www.ugto.mx

Abstract

This paper presents a parallelized seismic analysis algorithm, applied to the optimal design of structures. The method proposes different structure configurations, varying location, quantity, material and cross sectional type of elements that compose it, and searches for the design with the least amount of material that meets current structural building regulations.

The optimization process is performed with the stochastic search method Simulated Annealing [1].

Seismic analysis is applied on every configuration; it calculates the structure vibration modes, the equivalent static forces (for each of the main vibration modes) and the maximum structure efficiency. The applied seismic spectrum depends on geographic location and is defined according to the National Electricity Commission Regulations 2008 (Comisión Federal de Electricidad) [11].

The solution requires, given its complexity, large amount of computational power since main vibration modes are calculated for each configuration, during the whole optimization process, by solving a eigenvalues and eigenvector problem. Because of this, it is necessary to use parallel computing. This algorithm was parallelized using the multithreading shared memory programming interface OpenMP where each core performs the complete analysis of a structure configuration.

1. Introduction

One of the first application's fields for computational and numerical methods was structural analysis that is calculating the mechanical actions on structural elements. The development of this field continued to the point that today there is plenty of software focused in this area. However, the programs

able to perform structural analysis and design, this is to calculate whether or not the proposed structural elements are capable of withstanding the mechanical actions to be applied on them, are just a few.

The software here presented was developed to perform analysis, design and optimization of structures; while considering accidental and, more importantly, seismic actions.

Various materials can be used: hot rolled steel, cold formed steel and concrete. Combining these materials opens up a wide-range of possibilities for building structures. Selecting the optimal structure, in terms of geometry and material, represents an opportunity niche as it generates considerable savings whilst ensuring structural safety and functionality.

2. Calculation methodology

This section describes the methodology for calculating a structure, which is one of the main objectives of this project. The program calculates the given structure using the geometric characteristics defined by the user and the load cases defined in the Federal District Building and Construction Regulations [10]. The accidental loads taken in account are the wind and earthquake forces according to the Mexican National Electricity Commission Regulations [11].

Structures optimization depends greatly on the seismic effects, which are estimated with a modal-spectral analysis. It requires the calculation of the main vibration modes.

From a dynamic structural model with n degrees of freedom:

$$\boldsymbol{M\ddot{D}(t) + C\dot{D}(t) + KD(t) = -M[J_x a_x(t) + J_y a_y(t) + J_z a_z(t)]} \quad (1)$$

Each movement direction is analyzed:

$$\boldsymbol{M\ddot{D} + C\dot{D} + KD = -MJa(t)} \quad (2)$$

Free non-damped vibrations are calculated, disregarding damping, with the system:

$$\boldsymbol{M\ddot{D} + KD = 0} \quad (3)$$

In order to calculate the natural vibration modes, the above system must satisfy solutions of the form:

$$\boldsymbol{D(t) = \varphi sen(\omega t + \psi)} \quad (4)$$

With this a homogeneous and linear system of equations is generated, which solution implies an eigenvalues problem:

$$\boldsymbol{(K - \omega^2 M)\varphi = 0} \quad (5)$$

By solving det|K-ω^2 M|=0, eigenvalues ωi, called also natural frequencies, and eigenvectors φi, called also natural vibration modes, are calculated. This procedure is particularly important because it requires high computing power.

Next, movement equations are decoupled considering that the set of natural vibration modes represents a complete basis:

$$\boldsymbol{D = \sum_{j=1}^{n} \varphi_j y_j(t)} \quad (6)$$

$\boldsymbol{D}$ is substituted in the dynamic model for one direction, premultiplying by the transpose of any natural mode φ_j and considering the orthogonality of them we get:

$$\boldsymbol{\varphi_i^T M \varphi_j \ddot{y}_j(t) + \varphi_i^T C \varphi_j \dot{y}_j(t) + \varphi_i^T K \varphi_j y_j(t) = -\varphi_i^T M J a(t)} \quad (7)$$

Changing C and K notation as well as $M{:}M_{\_j}\text{^}*=\varphi_{\_j}\text{^}T\ M\varphi_{\_j}$, the before equation corresponds to the one degree of freedom model.
Dividing by M_j^* both sides of the equation:

$$\ddot{y}_j(t) + 2\nu_j\omega_j\dot{y}(t) + \omega_j^2 y_j(t) = -\frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}a(t) \quad (8)$$

The value of a(t) is obtained from the response spectrum accordingly to the period of the j natural mode and it is called S_a. It is clear then, that maximum displacement for the j natural mode is:

$$\left|\ddot{y}_j(t)\right|_{max} = \frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}\frac{(S_a)_j}{\omega_j^2} \quad (9)$$

Maximum displacement on each node of this natural mode is obtained premultiplying by

$$D^j{}_{max} = \boldsymbol{\varphi}_j\left|\ddot{y}_j(t)\right|_{max} = \boldsymbol{\varphi}_j\frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}\frac{(S_a)_j}{\omega_j^2} \quad (10)$$

Maximum static lateral equivalent forces are obtained by premultiplying maximum displacements by K:

$$F^j{}_{max} = \boldsymbol{K}\boldsymbol{\varphi}_j\left|\ddot{y}_j(t)\right|_{max} = \boldsymbol{K}\boldsymbol{\varphi}_j\frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}\frac{(S_a)_j}{\omega_j^2} \quad (12)$$

$$F^j{}_{max} = \frac{\boldsymbol{K}}{\omega_j^2}\boldsymbol{\varphi}_j\frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}(S_a)_j$$

Considering that

$$\boldsymbol{K}\frac{1}{\omega_j^2} = \mathbf{M} \quad (13)$$

it is clear that maximum force for the j natural mode is:

$$F^j{}_{max} = \boldsymbol{M}\boldsymbol{\varphi}_j\frac{\boldsymbol{\varphi}_j^{\mathrm{T}}\boldsymbol{MJ}}{\boldsymbol{\varphi}_j^{\mathrm{T}}\mathbf{M}\boldsymbol{\varphi}_j}(S_a)_j \quad (14)$$

Maximum lateral equivalent force, considering the effects of all principal natural modes is:

$$\boldsymbol{F}_{max} = \sqrt{\sum_{j=1}^{q}(D^j{}_{max})^2} \quad (15)$$

These forces are included in the load cases and then the stiffness matrix method is used.
Solving the eigenvalues problem is fundamental, given the large amount of computational power it requires; in this project the power method was used.
The structural analysis provides the service loads that will be acting on each structural member. The structural design determines the maximum load that each member can withstand. Efficiency is then computed as the ratio between these two values.
Efficiency calculations are based on the regulations for each material: for cold formed steel AISI-ASD [8,9] is used; for concrete, ACI [12]; and for hot rolled steel, AISC [13].

3. Local optimization methodology

Stochastic search techniques have been applied to solving complex combinatory problems. The most antique of them is probably Simulated Annealing [3] that generates a sequence of solutions combining mutation operations with an acceptance probability function that restrict changes over time [4].
Other techniques like Evolutionary Strategies [5] and Genetic Algorithms (GA) [6] also involve a mutation operation but this is applied to all elements of the population that compete ones against each other (in the selection process) or exchange information (recombination in GA).
The general problem being solved is:

$$\Omega = Q_1 \times Q_2 \times Q_3 \times \dots Q_n \quad (16)$$

Where each Q_i, i=1,…,n is a set of finite size that evolves towards finding the best structure in some way; and given a cost function as in eq. 17 we want to find a solution vector (eq. 18) that minimizes U globally. Such solution vector is the set of structural elements that, while assuring structural optimal service conditions, represents the lightest structure.

$$U: \Omega \mapsto R \quad (17)$$

$$x^* = (x_1^*, x_2^*, x_3^*, \dots, x_n^*) \quad (18)$$

The basic components of the algorithm are: a population X (eq. 19) that is a set of structure configurations, where each of them represents an ordered series of vectors (eq. 20), representing the structural elements; and the mutation and acceptance operators which are applied on the population.

$X=\{x^1,x^2,x^3,\dots,x^N,\}$ (19)

$$x^i = (x_1, x_2, x_3, \dots x_n) \; i = 1,2,3,\dots,n \quad (20)$$

Mutation operator:

A family of operators is defined with a mutation parameter M_μ(eq. 21) with the following algorithm:

$$M_\mu: \Omega^N \mapsto \Omega^N \quad (21)$$

For each element xof the population, generate an element y such that eqs. 22 y 23 are met.

$$y_i = r_i \text{ with a probability } p(\mu, x, X) \quad (22)$$

$$y_i = x_i \text{ with a probability } 1 - p(\mu, x, X) \quad (23)$$

Where r_i is a random element selected from Q_i, with a uniform probability $\mu \in [0,1]$.

Make M_μ (x) as in eq. 24.

M_μ (x)=(y_1,y_2,y_3,…,y_N)

$$M_\mu(x) = (y_1, y_2, y_3, \dots, y_N) \quad (24)$$

The mutation probability p(μ,x,X) can be uniform (i.e, p(μ,x,X)=μ) or adaptive, depending on the acceptance value x. The adaptive probability is written as in eqs. 25 and 26.

$$p(\mu, x, X) = \mu \frac{f_{max} - f(x)}{f_{max} - \bar{f}} \quad if \;\; f(x) > \bar{f} \quad (25)$$

$$p(\mu, x, X) = \mu if \;\; f(x) \leq \bar{f} \quad (26)$$

Where f_{max} and $\bar{f}$ are the maximum and average value, respectively, of the acceptance function in population X.

Acceptance operator:

In order to select a candidate between population X and a candidate from the mutated population Y, a metropolis acceptance criterion is applied to each element of X and Y. This defines a family of acceptance operators shown below.

$$A_\beta: \Omega^N \times \Omega^N \mapsto \Omega^N \quad (27)$$

For each element $x^i \in X$,M_μ (x^i)=$y^i \in Y$ with i=1,2,3,…,n do:

- $\Delta U = U(y^i) - U(x^i)$ if $\Delta U \leq 0$, make $u^i = y^i$
- If $\Delta U > 0$, make $u^i = y^i$ with probabilityexp[-βΔU]
- Make $u^i = x^i$ with probability (1-exp[-$\beta \Delta U$])
- Make $A\_\beta\ (X,Y) = \{u^1, u^2, u^3, \dots, u^N\}$

The dynamic system in equation 28 defines the used algorithm.

$$X^{(t+1)} = A_\beta \left(Y^{(t)}, M_\mu \left(Y^{(t)} \right) \right) \quad (28)$$

Where in general, parameters β,μ can

vary through time and population $X^{(0)}$ can be randomly chosen.

The optimization of a structure consists in finding the cross-section type for each element selected from a Ccatalog (ec. 29) (this is, each element of the population $x=(x\_1,x\_2,x\_3,\ldots x\_n) \in X$, is formed by elements in C, $x_i \in C$, $i=1,2,3,\ldots,n$) in such manner that all stresses acting upon each bar are less than an allowable value and the total weight of the structure is the lightest possible. The cost function to be minimized is defined by eq. 30.

$$C = \{e^1, e^2, e^3, \ldots, e^M\} \quad (29)$$

$$U(x) = \sum_{i=1}^{n} [\rho_x Z_{xi} L_{xi} + \lambda_1(\partial\sigma_{xi}) + \lambda_2(\partial\vartheta_{xi})] \quad (30)$$

Where ρ,Z,L are the specific weight, the catalog cross section and the element length $x(e)$; $\partial\sigma$ is the value in excess of the bearing capacity of the structural element, being it axial loading, flexion and compression or flexion and traction (the shear effects are also considered), $\partial\vartheta$ is the displacement value of a node that exceeds the maximum. λ_1 is a penalty factor for excess strain, in our case it is $\lambda_1=10000$. λ_2 is scaled in function of λ_1, in our case $\lambda_2=1000\lambda_1$. Further details about the algorithm can be consulted on previous works developed at CIMAT [1,2].

4. Global optimization methodology

An optimization over several structural shapes was performed, as example here is shown the process for a truss type roof:

Structure type selection. A group of possible structures is selected from a pre-defined catalog of geometric shapes and organized into an array. Each element of the catalog is a different structural shape, for example roof systems can be Howe, Gable or Warren type.

1. Parameters. For each type of structure a matrix is formed with combinations of two parameters: the number of plane structures and the number of internal divisions for each one. Each element of the matrix represents a possible structure to be built.
2. Optimization. For each element of each matrix, local optimization (described in the preceding section) is performed; this is material and cross section optimization.
3. Selection of the minimum. When local optimizations are finished, the minimum of each structure type is compared and the global minimum is selected as the optimal roof.
4. Selection of the minimum. When local optimizations are finished, the minimum of each structure type is compared and the global minimum is selected as the optimal roof.

5. Algorithm parallelization

As mentioned before, in order to find the optimal a great number of evaluations is required. For each evaluation loading cases vary even if they have the same geometric configuration, since different elements (in terms of cross-section and material) have different self-weight. Various methods for implementing Open-MP parallelization were available. For example, the parallelization could be done in the algorithms for solving the systems of equations or in the routines structural seismic analysis. Since the first option require less exchange of information, that was implemented.

6. Application examples

Results for the main six vibration modes

of a six levels office building (Figure 1) are shown. In such figure, bars of the same material are shown with the same color. This has an important constructive consequence: allowing symmetry for better structural behavior. In this example, the structure has 390 bars grouped in 25 different types, obtained from 3 materials catalogs (concrete, hot rolled steel and cold formed steel). As a natural consequence, optimization process modified the structure and reduced horizontal equivalent seismic loads. With this, optimal structural design was achieved for the geographical design region, in this case Irapuato Gto. The next figures show the first six vibration modes.

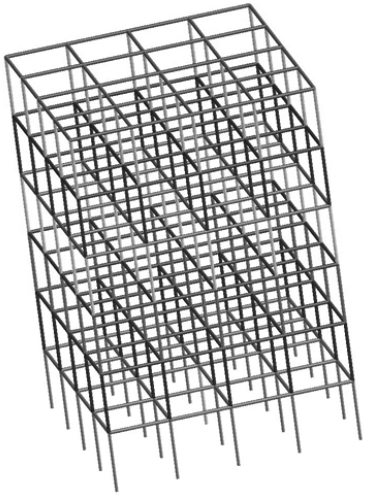


Figure 1. Geometry of the structure to be optimized. Each color represents a different group of elements.

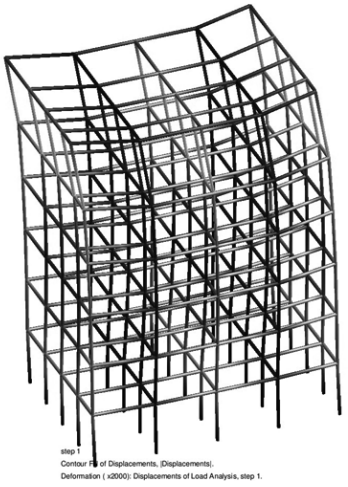


Figure 2. First vibration mode

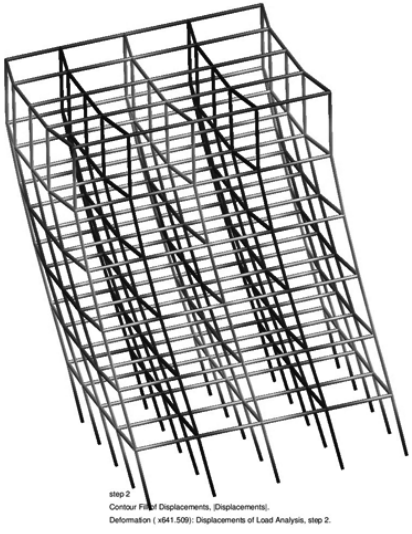


Figure 3. Second vibration mode

Figure 4. Third vibration mode

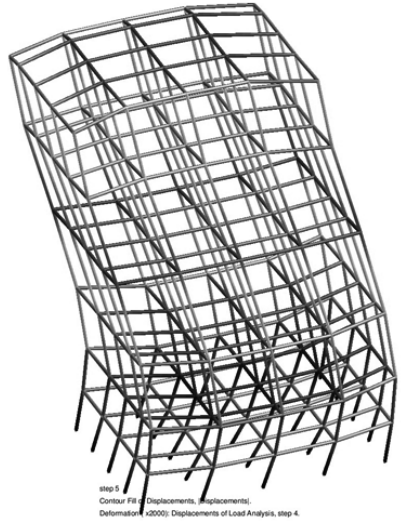


Figure 5. Fourth vibration mode

Figure 6. Fifth vibration mode

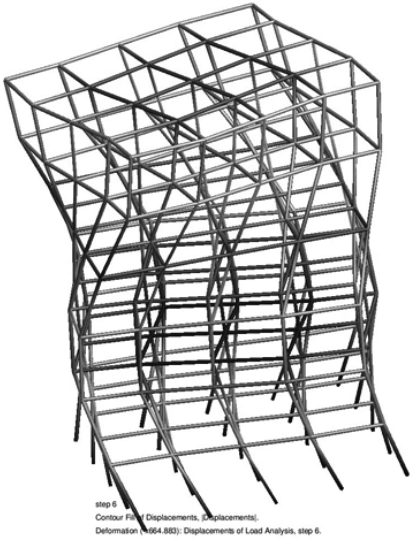


Figure 7. Sixth vibration mode

7. Conclusions

A structural analysis, design and optimization software has been developed, which is easy to use, versatile, reliable and has a friendly user interface. It quickly provides the cheapest solution for each particular scenario. Besides it can select elements from three different catalogs, (concrete, hot rolled steel and cold formed steel) which benefits population from all over the country allowing them to build structures considering local availability of materials. The program can also be used to design individually each section of a structure, for example, roof can be built with cold formed steel and columns with concrete.

It has been shown, through numerical computations, that the software provides safe and reliable solutions for a wide range of structures, spanning from residential houses trough multi-level industrial buildings. It is also shown that developing optimization technologies can promote business opportunities with social relevance.

The software shown in this article is a continuation of research and development efforts performed at CIMAT with the objective of achieving better solutions for sophisticated calculations in shorter time and taking advantage of the parallel computing infrastructure available, ensuring a correct application of numerical methods.

References

[1] S. Botello, J.L. Marroquín, E Oñate and J. Horebeek, "Solving structural optimization problems with genetic algorithms and simulated annealing", *Int. Jou. Num. Met. Eng*., vol. 45, pp. 1069-1084, 1999.

[2] S. Botello, J.L. Marroquin, A. B. Rionda, R. Ducoingx, *"MECA Programa para el Análisis Matricial de Estructuras"*, Facultad de Ingeniería Civil, Universidad de Guanajuato, diciembre de 1997.

[3] S. Kirkpatrick, C.D. Gelatt and M.P.,"Vecchi Optimization by simulated annealing", *Science* 220 (4598), pp 671—680, 1983.

[4] S. Anily and A. Federgruen, "Simulated annealing methods with general acceptance probabilities", *J. Applied Prob.* 24,pp 657-667, 1987.

[5] L.J. Fogel, A.J. Owens and M.J.,"Walsh Artificial Intelligence through Simulated Evolution", New York, Wiley Pub., 1966.

[6] D.E. Goldberg, "Genetic algorithms in search, optimization and machine learning", *Addison Wesley, Reading, MA*, 1989.

[7] M. Srnivas and L.L. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms", *IEEE Trans on Syst., Man and Cyb*., 24 (5),pp 656—667, 1994.

[8] American Iron and Steel Institute, *"North American Specification for the Design of Cold-Formed Steel Structural Members AISI (2002)"*.

[9] Wei-Wen Yu, *"Cold-formed Steel Design"*, John Wiley and Sons, Inc., 2000.

[10] Asamblea de Representantes del Distrito Federal,"Reglamento de construcciones para el Distrito Federal", *Diario Oficial de la Federación*,1993.

[11] Comisión Federal de Electricidad,"*Manual de Diseño de Obras Civiles, CFE (2008)*".

[12] Comitte ACI 318, "Building Code Requirements for Reinforced Concrete (318-02)", *American Concrete Institute*, Detroit, 2002.

[13] American Institute of Steel Construction, "*Specification for the Design, Fabrication and Erection of Structural Steel for Buildings*", 1969.

Comparative Analysis of Sorting Methods using OpenMP

Edgar A. Guerrero, Abel Palafox, Juan P. Serrano, José L. Alonzo.
Centro de Investigación en Matemáticas, CIMAT A.C., Guanajuato, Gto., México.
E-mail: {guae, abel.palafox, jpsr, pepe}@cimat.mx.

Abstract

The development of new technologies has reduced the computing power limits. This allows handling large volumes of data. There are plenty of applications involving data sorting implementations. It is necessary to have suitable methods for efficiently sorting large data sets. Several sorting methods using computers with multiple processors using OpenMP technology are implemented. A comparative analysis of the computational cost of the sorting algorithms is presented. The results provide information to choose a sorting method suitable for the type of application and available computing power. We mention applications that require efficient sorting methods for large data volumes.

Keywords: Bubble sort, Odd-Even Transposition Sort, Rank Sort, Counting Sort, Bitonic Sort, Quicksort, Radix Sort, Merge Sort, OpenMP.

1. Introduction

Computation power has increased widely in the span of about 20 years and still grows. There are many areas requiring computational speed including numerical simulation of scientific and engineering problems. Problems such as weather forecasting have a specific deadline for the computations. Modelings of large DNA structures and quantum mechanics simulations have grand challenge problems due to the high amount of data. Additionally, analysis by finite element method, Monte Carlo simulations, roughly molecular dynamics, among others, it requires computing expensive calculations on large data sets in order to give confident results. Computations must be completed within a suitable time. In manufacturing realm and engineering, calculations and other simulations must be achieved within minutes or even seconds.

Sorting procedure is an important component of many applications. Then there is a wide variety of sorting methods. On the other hand, parallel algorithms have been studied extensively in the last four decades [1-7]. Therefore, many of the sorting methods have a parallel version. However the parallelization itself is not enough to reach optimal results. We consider essential to develop a benchmark of sorting methods for choosing a suitable one based on the specific application and the available computer power. In this paper, eight sorting algorithms (Bubble Sort, Odd-

Even Transposition Sort, Rank Sort, Counting Sort, Bitonic Sort, Quicksort, Radix Sort and Mergesort) are implemented in parallel using OpenMP platform. For comparison purposes the performance of the parallel implementation is compared to the serial implementation. Their performance for various array sizes is measured with respect to sorting time.

In this context, there are developments which analyze the performance of the sorting algorithms using other platforms GPU (Graphic Processing Unit) architectures, OpenCL (Open Computing Language) platform [15, 16, 17]. GPU and OpenCL devices provide better results than traditional mechanisms. For the scope of this work, analyzing sorting methods with those platforms is lead as future work.

This paper is structured as follows. In section 2 we briefly explain eight classical sorting methods. The results of our implementations are presented on the section 3. Conclusions and future work are discussed on section 4.

2. Methods

Brief descriptions of several sorting methods on serial and parallel version are presented. A comparison of execution times with different number of processors and different data sets are presented.

In this work the following sorting methods are studied:

1. Bubble Sort
2. Odd-Even Transposition Sort
3. Rank Sort
4. Counting Sort
5. Bitonic Sort
6. Quicksort
7. Radix Sort
8. Merge Sort

We denote as X an array of integers whose size is n. The number of processor is denoted by p.

2.1 Bubble sort

The bubble sort method compares each element of the array with its neighbor. This procedure moves the highest order elements towards the last positions of the list on each iteration. All the elements are compared until the array is sorted.

This method is expensive because the whole array is analyzed on each iteration even already sorted elements. Bubble sort method is widely used due to its simplicity.

Placing an element on the sorted array takes n comparisons on average where n is the number of elements. Then, the complexity level is $O(n^2)$.

The parallel version of the Bubble Sort algorithm executes p phases. On each phase, the element xi is compared with its neighbor xi+1 and i is moved every p places. As we can see, the array is divided into sub-arrays of size p and the serial Bubble Sort is executed on each sub-array. Then, the complexity level is $O(n^2/p)$. This parallelization idea leads to the Odd-Even Transposition Sort method.

2.2 Odd-Even Transposition Sort

The Odd-Even Transposition Sort is a variant version of the Bubble Sort. This method operates in two phases: odd and even. In the even phase, each element xi of the list X, where i is even, is compared with its right neighbor xi+1. In the odd phase the right neighbor xj+1 of xj is compared where j is odd.

The serial version of the Odd-Even

Transposition Sort takes no advantage with regard to Bubble Sort. In the worst case, Odd-Even method has a complexity level of $O(n^2)$.

The parallel version is more efficient. Using p processors, the complexity level is about $O(n^2/p)$. So, using n processors, the complexity decreases to O(n).

An important aspect about Odd-Even is the fact that reduces concurrency. During the execution of each phase (Odd or Even), every process only access to the element xi and its neighbor xi+1. This is provided both phases do not run simultaneously.

The parallel version of Odd-Even method uses a flag variable. This flag indicates whether the list is sorted (i.e. whether there are swaps). The shared variable flag takes only a value 0 or 1. However, this increases the execution time due to concurrencies. The directive reduction of OpenMP lets several processors access to the flag variable. An increment on flag and the directive reduction are used instead assign the value 1.

2.3 Rank Sort

The Rank Sort is designed for arrays of unique elements. For every element xi on the input array, the algorithm calculates the total number of elements xj that are less or equal to xi. The computed number is called the rank of xi. To calculate the rank of a number, we need to compare it with every element on the array. Then, each element on the input is placed into a new array. The position of the element on the new array corresponds to its rank.

For sorting an array of size n, the algorithm uses n iterations of the principal loop. In every iteration the algorithm requires n comparisons. With this analysis, we say that the complexity of the method is $O(n^2)$. We note that the execution time does not depend on the initial configuration of the array.

The parallel version of the algorithm could be obtained by computing the rank of every element on a different processor. The algorithm uses two arrays (unsorted and sorted), which are on a shared memory location. Both arrays are available for all the processes at the same time.

In the parallel version the complexity level is about $O(n^2/p)$. The main loop is divided between p processors and then n comparisons are placed for calculating the rank for every element. Using n processors, the complexity level is O(n).

Nevertheless Rank Sort algorithm is designed for arrays of unique elements, it is possible to improve it in order to deal with arrays with non-unique elements. This modification could lead concurrencies. For instance the final array is analyzed in order to treat repeated elements.

2.4 Counting Sort

This algorithm is designed for sorting arrays of integers with values in range [1,m]. This method builds the histogram C for the array X. Histogram C is updated by computing a prefix sum along the histogram. Finally, the method computes the position of each element on a sorted array B using the C array.

Counting sort method requires an array C of size m to store the histogram. Building the histogram, as well as computing positions, requires only one access to each element of the array. On the other hand, the prefix sum is performed over the elements of C, so there are m computations. Then, the complexity order is O(n+m).

The parallel version of Counting sort methods requires a deeper work. For instance, the copy of array X to array B is done using p processors.

2.5 Bitonic Sort

Bitonic Sort is based on building bitonic sequences. A bitonic sequence has two subsequences; one has increasing order and the other one decreasing order. The bitonic method for a sequence of n=2k elements consists of three blocks. The first two blocks sorts the two halves of the sequence in ascending and, respectively, descending order. The third part merges the two sorted halves to obtain the sorted array.

The parallelization is done in both parts: on creating the blocks and merging them. The serial version of Bitonic Sort has a complexity level of $O(n \log_2 n)$ whereas parallel version has a complexity level of $O(n \log_2 n/p)$.

2.6 Quick sort.

The Quicksort method is based on the divide-and-conquer paradigm. This method first divides a large array into two smaller sub-arrays: low elements and high elements. The complexity level of this algorithm is O(n log n) on average, assuming there are just distinct elements in the array. The worst-case performance is $O(n^2)$.

There are three-step divide-and-conquer stages for sorting a typical sub-array $[x_i, ..., x_k]$ of a array X:

- Divide: Partition (rearrange) the array $[x_i, ..., x_k]$ into two (possibly empty) sub-arrays $[x_l, ..., x_{j-1}]$ and $[x_{j+1}, ..., xm]$, such that each element of $[x_l, ..., x_{j-1}]$ is less than or equal to xj. In the same way, x_j is less than or equal to each element of $[x_{j+1}, ... , x_m]$.
- Conquer: Sort the two sub-arrays [xl, ..., x_{j-1}] and $[x_{j+1}, ..., xm]$ using recursive calls to Quicksort.
- Combine: Since the sub-arrays are sorted in place, no work is needed to combine them. The entire array X is now sorted.

An important aspect is how the partition is done. Choosing a random pivot index j does not require additional computation. Nevertheless this could increase the complexity level up to $O(n^2)$ for certain cases. The qsort function of the C language performs a partition which sorts the subarray in place.

For the parallel procedure, we divide the array of size n into p sub-arrays. Then we use the qsort procedure to sort every sub-array on a different processor at the same time. Finally, all sub-arrays are merged. The complexity level for the parallel version is O(n log n/p).

2.7 Radix Sort.

Radix Sort can be used to sort items that are identified by unique keys. This method works as follows:

- Take the least significant digit (or group of bits) of each key.
- Sort the array of elements based on that digit, but keep the order of elements with the same digit (this is the definition of a stable sort).
- Repeat the sort with more significant digits each time.

Its efficiency is O(kn) for n numbers with k or fewer digits.

2.8 Merge sort

The Merge sort method splits the array into many sub-arrays until just two elements are contained. Then the algorithm starts merging such sub-arrays while sorts them. The algorithm can be described in two parts: splitting the initial array and merging it on the sorted array. This algorithm has a complexity level O(n log n).

For the parallel version, the initial array is divided into p sub-arrays. Each processor sorts a sub-array using the serial Merge sort version. Once every sub-array is sorted, those are merging on a single array. So, the parallel version of this method has a complexity level of O(n log n/p).

Results and Discussion

The test cases are arrays of integers randomly chosen on the interval [1, 1000]. The size of the test arrays are powers of 2 on the interval [26, 217]. For every size, we execute all the methods presented in this work using 2, 4, 8, 16 processors and executions of the serial version. Our implementations were made on C and we use OpenMP for executing the parallel versions. All the test cases were run on the host described below.

In order to obtain statistical information about the running time for each method we run 30 independent tests for each array size and for each number of processors. On each test, the same unsorted array was used to measure the execution times. With this information, we report mean value, standard deviation, maximum value and minimum value for each method.

Time and String Metrics		Constant Metrics	
Machine Type	x86_64	CPU Count	16 CPUs
Operating System	Linux	CPU Speed	2395 MHz
Operating System Release	3.2.0-24-generic	Memory Total	32938000 KB
		Swap Space Total	1020 KB

Figure 1 presents the mean times obtained varying the array size in serial version. Figure 2 shows the elapsed times for each method on parallel version using two processors. For visualization purposes the results obtained from 4^{th} to 8^{th} methods are presented in Figure 3. Similarly, the results obtained for 4, 8, and 16 processors are presented in Figures 4 to 9 respectively.

The first three methods has the expected behavior since these algorithms are the order $O(n^2)$. However, Odd-Even and Rank Sort takes advantage of the parallelization and improves the execution time when increasing the number of processors. Indeed, execution time seems to be linear when increasing the number of processors. However this improvement is not enough.

The best results are obtained by Counting Sort, Bitonic Sort, Quicksort, Radix Sort and Merge Sort. The lowest execution time is reported by Counting Sort method. Low execution time of the Counting Sort method is due to the range of the numbers which are between 1 and 1000. Thus, the cost for the histogram computation is negligible. In future we suggest using a range of values comparable with the size of the array.

Figures 10 and 11 present the mean elapsed times for a test case of size 217 varying the number of processors. For visualization purpose

the results are presented Figure 10 shows all methods and Figure 11 shows from 4th to 8th methods. The information on Figures 10 and 11 is important due to the following: it is expected that the parallelization of a procedure reduces the computational cost. However, as can be seen on Figure 10 and 11 the parallelization could increase the execution time. For instance the Merge sort seems to have a not surprising performance. However, this method still takes advantage on the parallelization. This is, increasing the number of processors it is possible to reduce the execution time. On the other hand, the Quicksort algorithm has a good performance but it does not improve from 8 to 16 processors.

The Radix Sort has a execution time between the Counting and Bitonic Sort. This method is a good option for sorting based on the obtained results. In contrast Figure 11 shows that the mean time gets worse even using just 2 processors.

Statistic information for the execution times on the case 217 is computed. With this information robustness of every method is compared. Methods with lowest standard deviation (SD) are less sensitive to the sort on input data. By comparing the SD and mean values when varying the number of processors, robustness of every method according to the parallelization is obtained.

Table 1 shows the statistical results for the serial version. Table 2 presents the results for 8 processors and for 16 processors are presented in Table 3. Counting Sort, Bitonic Sort, Quicksort, Radix Sort and Mergesort have low values for the SD. Also, those methods are executed in short times. Counting Sort, Bitonic Sort, Quicksort and Mergesort algorithms keep their values for SD in all cases. The Radix Sort increments its initial SD value. On the other hand, the Bubble Sort, Odd-Even and Rank Sort algorithms have big values in all the statistical values.

4. Conclusions and Future Work

We implemented eight classical sorting algorithms. We focused on studying their performance on large data sets using parallel computing. Also, we analyzed the sensitivity of those algorithms when increasing the number of processors. The results obtained include statistical information on a specific case.

The sensitivity analysis and the statistic information provide additional elements on the comparisons. For instance, the Mergesort seems loose performance. However we can deduce that this method could offer better results using a higher number of processors. On the other hand, the Quicksort algorithm presents, in general, the best performance. Nevertheless, increasing the number of processors will not improve the results.

As future work, would be implemented the algorithms using MPI and CUDA technology which allow us to increase the size of the test arrays. Additionally, there are some areas which present particular structures along large data sets. The performance of sorting methods could be improved by taking advantage of such structures.

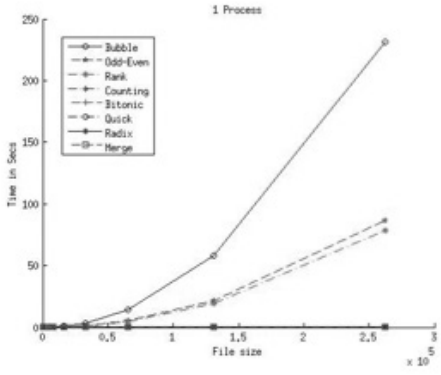


Figure 1. Elapsed times for serial versions.

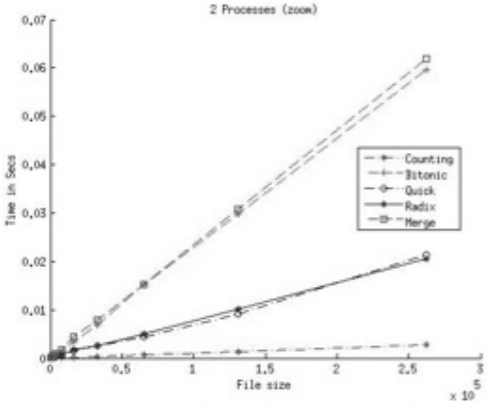


Figure 3. Elapsed time for two processors from 4th to 8th methods.

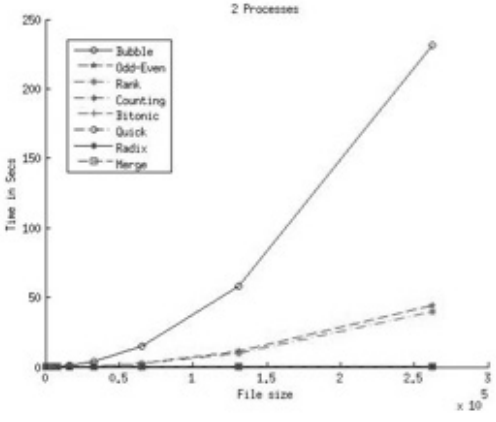


Figure 2. Elapsed times for two processors.

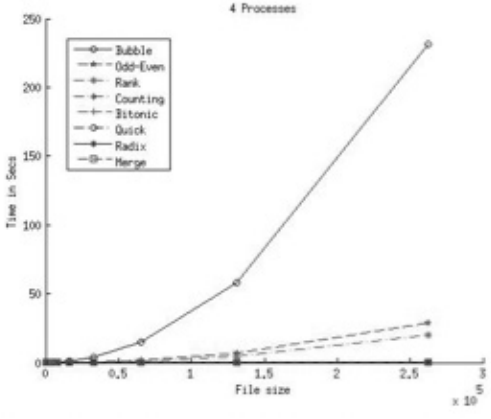


Figure 4. Elapsed times for four processors.

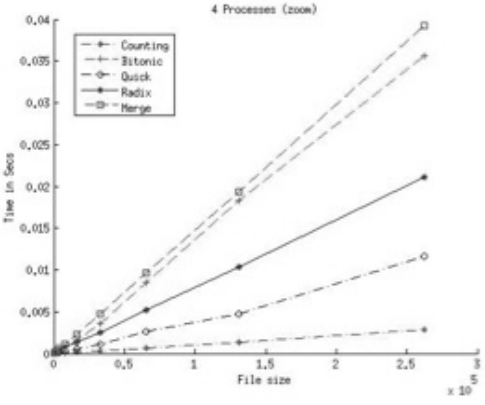


Figure 5. Elapsed time for four processors from 4th to 8th methods.

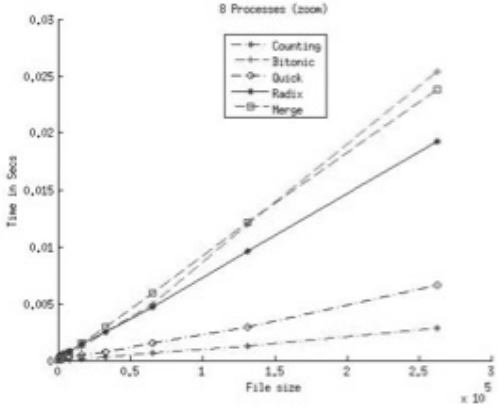


Figure 7. Elapsed times for eight processors from 4th to 8th methods.

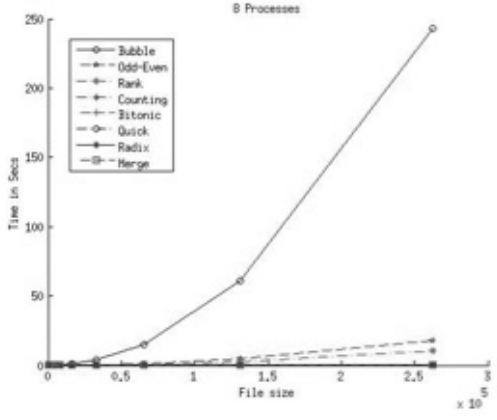


Figure 6. Elapsed times for eight processors.

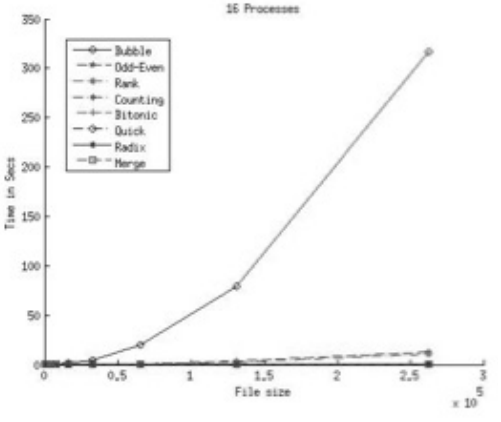


Figure 8. Elapsed times for sixteen processors.

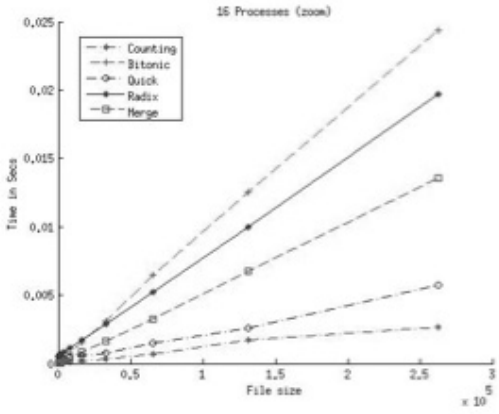


Figure 9. Elapsed times for sixteen processors from 4th to 8th methods.

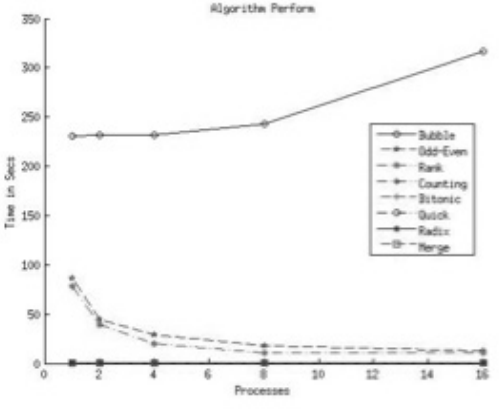


Figure 10. Algorithm performance varying processors number.

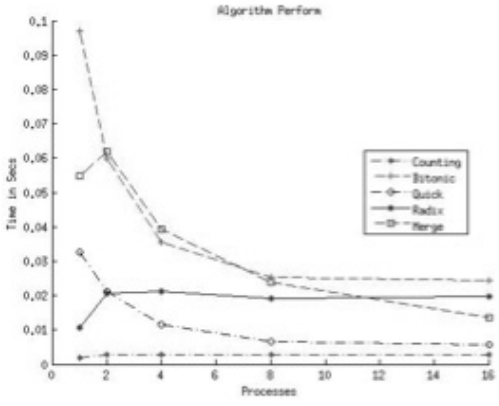


Figure 11. Algorithm performance varying processors number from 4th to 8th methods.

Table 1 Stats for serial version on case 217.

Method	Min	Max	Mean	SD
Bubble	230.45	232.53	231.1843	0.60486
Odd-Even	86.035	86.872	86.3991	0.22909
Rank	78.497	79.189	78.6325	0.2239
Counting	0.001861	0.002044	0.0018792	3.5795e-5
Bitonic	0.095856	0.1116	0.096794	0.0028029
Quicksort	0.032015	0.04822	0.032622	0.0029462
Radix	0.010525	0.011296	0.010682	0.0001523
Merge	0.054602	0.055496	0.054921	0.0002291

Table 2 Stats for parallel version on case 217 with 8 processors.

Method	Min	Max	Mean	SD
Bubble	242.92	243.71	243.2953	0.2165
Odd-Even	18.16	18.983	18.2453	0.1416
Rank	10.332	10.465	10.3478	0.025671
Counting	0.002506	0.0035	0.0028552	0.0002880
Bitonic	0.022366	0.030199	0.025362	0.0021564
Quicksort	0.005486	0.011502	0.0066391	0.0011887
Radix	0.018316	0.021488	0.019289	0.0005774
Merge	0.023099	0.024108	0.023795	0.0003563

Table 3 Stats for parallel version on case 217 with 16 processors.

Method	Min	Max	Mean	SD
Bubble	316.17	318.39	317.2497	0.49886
Odd-Even	12.381	12.798	12.5655	0.10452
Rank	10.268	10.35	10.2772	0.020761
Counting	0.002608	0.003006	0.0026888	9.5532e-5
Bitonic	0.023041	0.029239	0.024378	0.0014214
Quicksort	0.004882	0.011883	0.0057337	0.0012452
Radix	0.018195	0.02245	0.019681	0.0010663
Merge	0.013186	0.013898	0.013541	0.0002427

References

[1] K. W. Batcher, “Sorting networks and their applications”, AFIPS Conf. 32, (1968), pp. 307-314.

[2] H. Barlow, D. J. Evans and J. Shanehchi, “A parallel merging algorithm”, Information Processing Letters 13, No. 3, Dec. 1981, pp. 103-106.

[3] G. Baudet and D. Stevenson, “Optimal sorting algorithms for parallel computers”, IEEE Trans. Comput. C-27, No. I, Jan. 1978, pp. 84-87.

[4] F. Gavril, “Merging with parallel processors”, Comm. ACM 18; 10, Oct. 1975, pp. 588-591.

[5] D. E. Muller and F. P. Preparata, “Bounds to complexities of networks for sorting and for switching”, J. Assoc. Comput. Math. 22; 2, April 1975, pp. 195-201.

[6] L. G. Valiant, “Parallelism in comparison problems”, SIAM J. Comput. 4; 3, Sep. 1975, pp. 348-355.

[7] Yasuura, N. Takagi and S. Yajima, “The parallel enumeration sorting scheme for VLS1”, IEEE Trans. Comput. C-3, No. 12, Dec. 1982, pp. 1192-1201.

[8] B. Chapman, G. Jost, R. van der Pas, Using OpenMP:
Portable Shared Memory Parallel Programming, The MIT Press, (2008).

[9] B. Wilkinson, M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, 2nd edition. Pearson Education Inc. (2005).

[10] B. Barney. OpenMP. (2012) https://computing.llnl.gov/tutorials/openMP.

[11] C. Breshears. The Art of Concurrency: A Thread Monkey’s Guide to Writing Parallel Applications, O’Reilly Media, Inc., (2009).

[12] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson, Introduction to Algorithms (2nd ed.), McGraw-Hill Higher Education. (2001)

[13] Shi-Kuo Chang, Data Structures and Algorithms, World Scientific (2003).

[14] Karl Fürlinger, “OpenMP Application Profiling - State of the Art and Directions for the Future”, In Proceedings of the 2010 International Conference on

Computational Science (ICCS 2010), Amsterdam, NL, May (2010).

[15] Khan F.G., O.U. Khan, B. Montrucchio, P. Giaccone, "Analysis of Fast Parallel Sorting Algorithms for GPU Architectures", Frontiers of Information Technology (FIT), pp. 173 – 178. (2011).

[16] Y. Quan, D. Zhihui, S. Zhang, "Optimized GPU Sorting Algorithms on Special Distributions", In 11th International Symposium on Distributed Computing and Applications to Bussines, Enginering & Science, pp. 57-61, (2012)

[17] K. Zhang, J. Li, G. Chen, B. Wu, "GPU accelerate parallel Odd-Even merge sort: An OpenCL method", Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference, pp. 78-83, (2011).

[18] C. Cerín, J.L. Gaudiot., "Parallel Sorting Algorithms with Sampling Techniques on cluster with Processors Runing at Different Speeds", 7th International Conference Bangalore, India. pp. 301-309. (2000)

[19] V. Singh, V. Kumar, G. Agha, C. Tomlinson. "Efficient Algorithms for Parallel Sorting on Mesh Multicomputers", International Journal of Parallel Programming, Vol. 20 (2), (1991).

[20] S.S. Tseng, R.C. T. Lee, "A new parallel sorting algorithm based upon min-mid-max operations", BIT Numerical Mathematics. Vol. 24 (2), pp. 187-195. (1984).

[21] J. Wassenberg, P. Sanders, "Engineering a Multi-core Radix Sort" Euro-Par 2011. Parallel Processing. Lecture Notes in Computer Science. Vol. 6853, pp. 160-169. (2011).

Modeling Rock Failures Using Particles and Potential Energy Functions Under a GPU Parallelized Scheme

Victor Eduardo Cardoso-Nungaray, Salvador Botello
Industrial Mathematics and Computational Department,
Centro de Investigación en Matemáticas A.C.,
Jalisco S/N, Col. Valenciana, Zip 36240, Guanajuato, Gto. México,
email: victorc, botello@cimat.mx, web: www.cimat.mx

Abstract

The continuous modeling without failures is well analyzed by conservative methods, such as Finite Element Method (FEM) and the recent Isogeometric Analysis, but there is not an equivalent successful method to model discontinuous media, or continuous at the fracture moment. Since 1971, when Peter Cundall [1] proposed a method based on discrete elements bonded with springs, the Discrete Element Method (DEM) has been used to model this phenomena, unfortunately the parameters are difficult to calibrate in order to obtain physically realistic results. An industrial problem could require large scale simulations, but today's computational resources are not enough to satisfy this demand using a serial implementation.

This approach uses particles with associated energy potentials to allow the interaction between linked particles (modeling continuous media) and unlinked particles (modeling discontinuous media) to get rock failure simulations. The potential functions defines physical properties of the material (rock, concrete, masonry, etc) and the particles are ruled by the Newtonian movement equations.The mass is equally distributed among the particles, and links between them must be defined previously. The discontinuities in the media, such as failures, arise when those links are broken. A GPU is the ideal device to perform this computation, because there is the need to process hundred of thousands or even millions of particles independently at each time step, so that CUDA was selected to implement the program.

Keywords: GPU, parallel computing, particles, potentials, rock failure, simulation.

1. Introduction

The main goal of this work is to study a proposed model capable to emulate the rock nature under high pressures and other external forces, where fractures arises as a consequence of body failures caused by the internal stress, the state of the art models are based on Discrete Element Method (DEM), as those explained on [2], and some of them are based on a combined Finite-Discrete Element Method, as is shown by [3], there are also works to simulate terrain erosion (see [4]).

To accomplish the goal, we discretize the body using small enough particles and linking them in contiguous pairs to model the continua, we name this system: Netlink. Each particle's movement is going to be calculated using the Newton equations of motion, also known as Newton laws. Due to the potentials used, the linked particles must be equally spaced, resulting equilateral triangular discretization for 2D models and equilateral tetrahedral discretization for 3D models; the figure 1 shows a rough 2D body discretization to give a general idea about how the particles distribution should be.

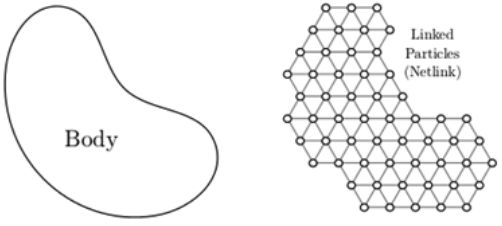


Figure 1: Free body and its rough 2D discretization using particles, note that

When the external forces acts over the body, some links could be broken forming discontinuities in the material, and giving place to the body fractures.
The model geometry dimensions are measured in meters (thousands of particles or even millions) with the intention to simulate a few seconds (hundreds of thousands of time steps), for this reason the implementation should be parallelized. Considering that each particle could be processed independently, we choose the GPU to compute the job, in order to maximize the processing performance.

2. Newton Laws

The simulation dynamics is ruled by the three universal laws of motion defined by Sir Isaac Newton in 1687, while the interaction of the i^{th} particle with respect to the jth particle is modeled with a central force field $G_i \in R^d$ (d is the dimension) derived from a potential energy function $U(r_{ij}):R \rightarrow R$ (which is going to be discussed later):

$$G_{ij} = -\frac{d}{dr_{ij}} U(r_{ij})\mathbf{n} \quad (1)$$

where r_{ij} is the euclidean distance between the ith and j^{th} particles, and $\mathbf{n} \in R^d$ is the normal vector pointing from the j^{th} particle position to the i^{th} particle position, as can be seen in the figure 2.

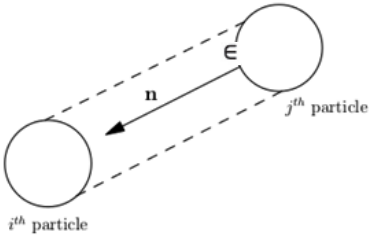


Figure 2: The normal vector n to compute the i^{th} particle force with respect to the j^{th} particle.

Afterwards, the kinetic energy could be monitored to understand the system dynamics. Such energy is definer by

$$K_i = \frac{1}{2} m_i \dot{x}_i^2 \quad (2)$$

for the ith particle, where mi is the particle mass and $\dot{x}_i$ is the particle velocity. In this work, we propose to distribute equally the body mass between the particles, but a different mass distribution could be assigned.

2.1 First Law

The Newton's first law, also known as inertia law, says that a body which is in a resting state will remain in such state, but if the body is moving, it will be in perpetual motion, with the same velocity, until an external force acts over it.

2.2 Second Law

The second Newton's law assures that the particle acceleration $\ddot{x}_i$ is directly proportional to the force applied to it, where the particle mass mi is the proportional constant, so we have

$$F_i = m_i \ddot{x}_i, \tag{3}$$

relating the force with the acceleration, connecting them in an elegant manner, which allows to integrate the particles velocity $\dot{x}_i$ and position xi over time.

2.3 Third Law

The third Newton's law is about the force reciprocity, for every force exist another force with the same magnitude but opposite direction, which could be nicely resumed by

$$G_{ij} = -G_{ji}, \tag{4}$$

meaning that the force from the ith to the jth particle plus the force from jth to the ith particle is equal to zero, balancing the system dynamics.
A complete review about the universal motion equation can be found in [5].

3. Time integration

After computing each particle force as a function of its position, we obtain the particles acceleration, and the remaining work is the time integration of $\ddot{x}_i$ to obtain the particle displacement in Δt time units, since Δt is the step size across the time t. Due to the discontinuities in the material, the numerical integration must be an explicit scheme, such as the Euler, leapfrog or Runge-Kuta methods, for a deeper analysis see [6].

The Velocity Verlet algorithm was selected because have a second order error with no additional cost of function evaluations.

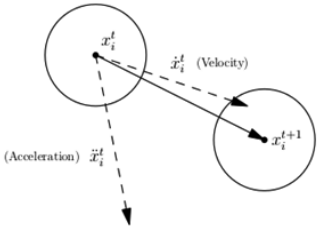


Figure 3: Particle time step using the Velocity Verlet Integration Algorithm.

The first step is to compute the position at x^{t+1} = xi(t+Δt) from the velocity $\dot{x}^t = \dot{x}_i(t)$ and acceleration $\ddot{x}^t = \ddot{x}_i(t)$:

$$x_i^{t+1} = x_i^t + \dot{x}_i^t + \frac{1}{2}\ddot{x}_i^t \Delta t, \quad (5)$$

the figure 3 illustrates the equation 5 calculation. Then, the force applied to each particle is obtained from

$$F_i = \sum_{i \neq j} G_{ij}, \quad (6)$$

where G_{ij} is defined in equation 1. Moreover, the acceleration at $\ddot{x}^{t+1} = \ddot{x}(t+\Delta t)$ should be calculated from the third Newton's law as

$$\ddot{x}_i^{t+1} = \frac{1}{m_i} F_i, \quad (7)$$

and finally, the velocity at $\dot{x}^{t+1} = \dot{x}_i (t + \Delta t)$ is computed from

$$\dot{x}_i^{t+1} = \dot{x}_i^t + \frac{1}{2}(\ddot{x}_i^t + \ddot{x}_i^{t+1})\Delta t, \quad (8)$$

completing the Velocity Verlet algorithm to integrate the time. Each iteration over the equations 5, 7 and 8 is going to simulate a Δt time step.

4. Particle Links

The links serves as a glue to stick the particles which conforms the body, creating the continua, and must be assigned before the computation starts, there is no manner to create links during the simulation.

Then, a link can be broken by two ways, the first is because an applied force produces a stress σ_{ij} over the link superior than some predefined maximum stress σ_{max} (over-stressed), and the second is because a maximum deformation ε_{max} has been exceeded by the link strain ε_{ij} (over-strained).

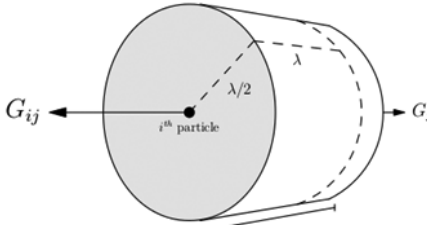


Figure 4: Link visualized as a three dimensional cylinder.

The parameters σ_{max} and ε_{max} are material properties, while the link stress is computed as

$$\sigma_{ij} = \frac{4||G_{ij}||}{\pi \lambda^2}, \quad (9)$$

because the links are treated as a cylinders with diameter and length equal to λ (λ is going to be discussed later), where the area of each cap surface is $\pi(\lambda/2)^2$, as can be seen in the figure 4. In the same way the link strain is obtained from

$$\varepsilon_{ij} = \frac{|\lambda - r_{ij}|}{\lambda}, \quad (10)$$

in a few words, a link will be broken when is over-stressed if

$$\sigma_{ij} > \sigma_{max}, \quad (11)$$

or over-strained if

$$\varepsilon_{ij} > \varepsilon_{max} \quad (12)$$

5. Energy potentials

The interaction among the body particles are modeled as pair wise potentials. To make the system numerically stable, the basic potential must be symmetric.

$$U(\lambda - r) = U(\lambda + r), \tag{13}$$

where $r = r_{ij} = r_{ji}$ and λ is the equilibrium distance between a pair of particles.

The potentials behavior must be problem geometry independent. The discretization size, defined by λ, and the shape distribution of the material doesn't have to affect the associated properties, in this way, the normalization of the potentials is primordial. A potential is considered normalized when satisfies the following conditions:

$$\begin{aligned} U(\lambda) &= 0, \\ \tfrac{d}{dr}U(0) &= -s, \\ \tfrac{d}{dr}U(\lambda) &= 0, \\ \tfrac{d}{dr}U(r) &< 0, \quad \forall r < \lambda, \\ \tfrac{d}{dr}U(r) &> 0, \quad \forall r > \lambda, \end{aligned} \tag{14}$$

where s is the force magnitude acting over a particles pair when they are at the same geometric point or when the space between them doubles the equilibrium distance.

The complete potential is evaluated between linked particle pairs, and for the non-linked particle pairs the potential is cut off at $r = \lambda$, leaving just the potential repulsion part, where $r \leq \lambda$, and zero for $r > \lambda$, as can be seen in the figure 5.

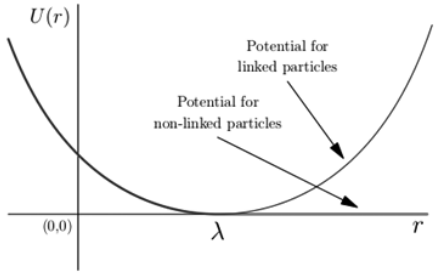


Figure 5: Complete potential for linked particle pairs (blue) and cutted potentials for non-linked particle pairs (red).

6. Proposed potentials

To simulate the non-linear natural behavior of materials, several basic potentials are proposed and described in this section. Each candidate utilizes a subindex to identify itself from the rest.

6.1 Perfect elastic potential

The perfect elastic potential is frequently used in particle systems to model all variety of phenomena, also known as spring-mass systems, where the spring potential energy is described by

$$U_1(r) = s\left(\frac{1}{2\lambda}r^2 - r + \frac{1}{2}\lambda\right), \tag{15}$$

and the spring force is given by the potential derivative:

$$\frac{d}{dr}U_1(r) = s\left(\frac{r}{\lambda} - 1\right), \tag{16}$$

since the potential is normalized, the spring

stiffness is equal to s/λ, while the force (potential derivative) is equal to minus s when then particles pair distance is zero, no matters which value λ takes.

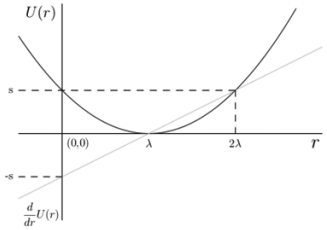


Figure 6: Elastic potential (blue) and its derivative (green).

tThe figure 6 shows the elastic potential plot and its derivative, note the linear (elastic) behavior and how the normalization ensures the values at r equal to zero and 2λ.

6.2 Exponential potential

The exponential potential is formulated as an exponential functions sum and is defined by

$$\frac{d}{dr}U_2(r) = \frac{b_2}{\lambda}\left(e^{(\frac{r}{\lambda}-1)} - e^{(1-\frac{r}{\lambda})}\right) \quad (17)$$

where

$$b_2 = \frac{s\lambda}{e - \frac{1}{e}} \quad (18)$$

is a normalization constant computed as a function of s and λ. Then, the exponential potential derivative is given by

$$\frac{d}{dr}U_2(r) = \frac{b_2}{\lambda}\left(e^{(\frac{r}{\lambda}-1)} - e^{(1-\frac{r}{\lambda})}\right) \quad (19)$$

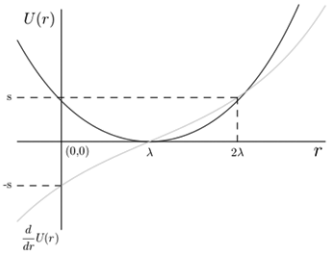


Figure 7: Exponential potential (blue) and its derivative (green).

The figure 7 illustrates the non-linear curve generated with the potential derivative. Contrasting with the elastic potential, where U1 (2λ) (equation 15) is equal to (d/dr)U_1(2λ) (equation 16), in the exponential potential U_2(2λ) (equation 17) is lower than (d/dr)U_2(2λ) (equation 19).

6.3 Trigonometric potential

The trigonometric potential behaves as a sinusoidal wave, and it could be used to model brittle materials. It is defined by

$$U_3(r) = b_3 \sin^2(\alpha(r - \lambda)), \quad (20)$$

where α ∈ (0, 1] controls the frequency of the wave (the potential slope), and

$$b_3 = \frac{s}{2\alpha \sin(\alpha\lambda)\cos(\alpha\lambda)} \quad (21)$$

is a normalization constant. The α parameter gives the following feature:

$$\lim_{\alpha \to 0} U_3(r) \to U_1(r), \quad (22)$$

while α goes to zero, the exponential potential $U_3(r)$ approximates the elastic potential $U_1(r)$.

Then, the trigonometric potential derivative is given by

$$\frac{d}{dr}U_3(r) = 2\alpha b_3 \sin(\alpha(r-\lambda))\cos(\alpha(r-\lambda)) \quad (23)$$

The figure 8 plots the trigonometric potential and its derivative, making clear the non-linear curve behavior, in this case the potential evaluated at $U_3(2\lambda)$ (equation 20) is greater than its derivative in the same point (d/dr) $U_3(2\lambda)$ (equation 23).

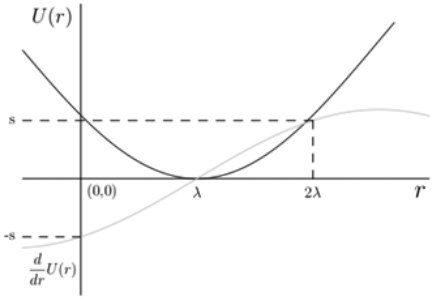


Figure 8: Trigonometric potential (blue) and its derivative (green) using α = ¼.

6.4 Logarithmic potential

The logarithmic potential is based on the natural logarithm function, and is defined by

$$U_4(r) = b_4 \ln\left(\frac{(\beta+1)^2}{\beta^2\left(2\frac{r}{\lambda} - \left(\frac{r}{\lambda}\right)^2\right) + 2\beta + 1}\right) \quad (24)$$

where β > 0 controls the slope rise of the potential, and

$$b_4 = \frac{s\lambda(2\beta+1)}{2\beta^2} \quad (25)$$

is a normalization constant.

The parameter β could be utilized to manipulate the potential concavity with the following property

$$\lim_{\beta\to 0} U_4(r) \to U_1(r), \quad (26)$$

when β goes to zero, the logarithmic potential $U_4(r)$ (equation 24) approximates the elastic potential $U_1(r)$ (equation 15).

To compute the force, the logarithmic potential derivative is described by

$$\frac{d}{dr}U_4(x) = \frac{\lambda}{\beta}b_4\left(\frac{1}{\left(2-\frac{r}{\lambda}\right)\beta+1} - \frac{1}{\frac{r}{\lambda}\beta+1}\right), \quad (27)$$

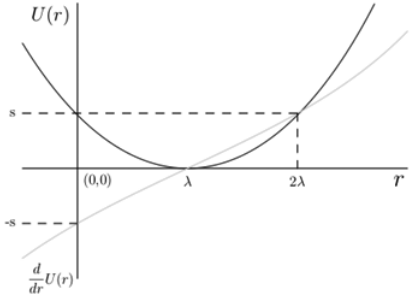


Figure 9: Logarithmic potential (blue) and its derivative (green) using β = 1/3.

The figure 9 shows the logarithmic potential and its derivative graphs. This potential has a quasi-elastic behavior using β = 1/3.

6.5 Power potential

The power potential could be used to model ductile materials. It is defined in terms of r power expressions:

$$U_5(r) = \frac{(r-\lambda)^{\gamma}}{\gamma\lambda^{(\gamma-1)}} \tag{28}$$

where $\gamma > 0$ is a positive pair integer (e.g. 2,4,6,8,...). If $\gamma = 2$, then the power potential $U_5(r)$ (equation 28) is equal to the elastic potential $U_1(x)$ (equation 15).

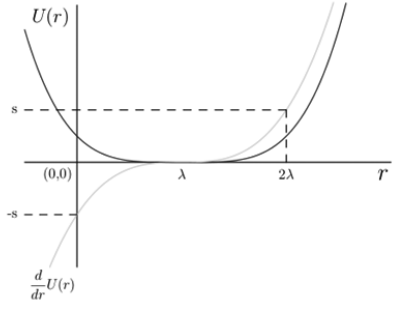


Figure 10: Power potential (blue) and its derivative (green) using $\gamma = 4$.

Then, the potential derivative is given by

$$\frac{d}{dr}U_5(r) = \left(\frac{r-\lambda}{\lambda}\right)^{(\gamma-1)}, \tag{29}$$

The figure 10 illustrates the power potential curve demeanor and its derivative graph using $\gamma = 4$.

6.6 Rational potential

The rational potentials are commonly used in Molecular Dynamics (such as the Lennard-Jones potential, which doesn't satisfies the equation 13), in First Principles Methods (such as the Coulomb Law, to model the electronic interaction), and in Gravitational simulations. Normally, the rational potentials are utilized because they are asymptotic to r = 0 in the following manner:

$$\lim_{r\to 0} U(r) \to \infty \tag{30}$$

but in this case, the equation 30 is not true because the rational potential proposed is normalized, this means that the potential satisfies the equations 14, and is defined by

$$U_6(r) = b_6 b_7 \left(b_8 - \frac{1}{\left(3-\frac{r}{\lambda}\right)^{(z-1)}} - \frac{1}{\left(1+\frac{r}{\lambda}\right)^{(z-1)}}\right) \tag{31}$$

where $z > 1$ controls the potential decay before λ (assuming the equation 13 is true) to suits the material conduct, and

$$b_6 = \frac{s3^z}{3^z-1}, b_7 = \frac{\lambda}{1-z} \text{ and } b_8 = 2^{(2-z)} \tag{32}$$

are normalization constants calculated as a function of s, λ, and z.

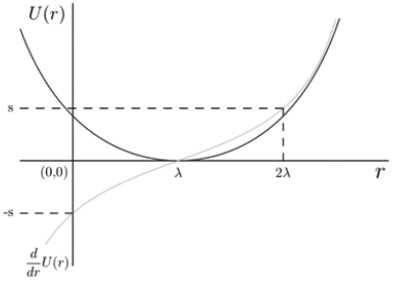


Figure 11: Rational potential (blue) and its derivative (green) using r = 1.1.

Then, the rational potential derivative is

$$\frac{d}{dr}U_6(r) = b_6\left(\frac{1}{\left(3-\frac{r}{\lambda}\right)^z} - \frac{1}{\left(1+\frac{r}{\lambda}\right)^z}\right), \qquad (33)$$

The figure 11 shows the rational potential and its derivative plots utilizing z = 1.1.

7. GPU Implementation

Since the body is formed by a huge number of particles, we need a robust algorithm to compute the N-Body M-Links problem, where N is the number of particles and M is the number of links.
We assumes that a particle can only interact with 6 neighbors at the same time in two dimensional problems (as can be seen in figure 12), and 12 in three dimensional problems. For understanding purposes, the two dimensional case is explained in this section.

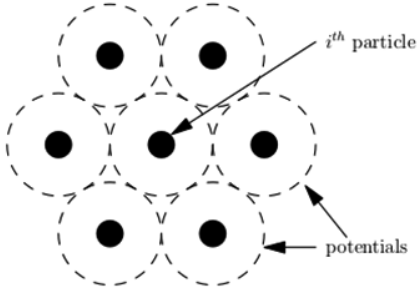


Figure 12: Maximum number of particles interacting with the ith particle at the same time. The distance between all the particle pairs is λ.

At first, we are going to allocate the 5 following arrays to store the particles information:

float x[2*N]← Contains the particles positions in a simple array sorted by particle id.
float v[2*N]← Contains the particles velocities in a simple array sorted by particle id.
float a[2*N]← Contains the particles acceleration in a simple array sorted by particle id.
float F[2*N]← Stores the particles forces summation in a simple array sorted by particle id.
float U[2*N]← Stores the particles potential energy sorted by particle id.

then, we need to allocate the links information in the next 5 arrays:

unsigned int `links[2*M]`← Stores the linked particle id pairs in a simple array sorted by link id.
float `stress[M]`← Stores the stress sorted by link id.
float `strain[M]`← Stores the strain sorted by link id.
float `Ur[2*M]`← Stores the potential energy computed for the first particle with respect to the second particle, sorted by link id.
float `G[2*M]`← Stores the force computed with the potential derivative of the first particle with respect to the second particle, sorted by link id.

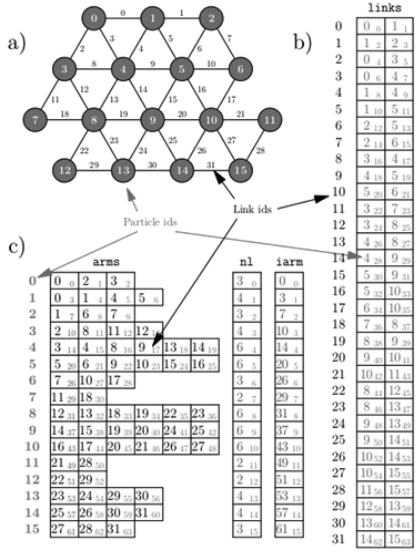


Figure 13: a) The particles (blue color) and the links connecting them (black color), both elements have its respective ids (C-style numbering). b) The links array illustration using blue color to remark the particle ids and black to mark the link ids serving as a guide to the eye. c) The arms array using black color to designate the link ids and blue to show the particle ids helping to locate its position in the array, then, the nl array contains the number of links (colored with green) for each particle, and the iarm array contains the index where start the particle connections in the arm array. The gray color are used to identify the arrays index (C-style numbering).

After, we need meta data structures to process whole the particles fast. Taking advantage of the maximum possible linked particles, we allocate the following arrays to relate the particles and links:

unsigned int `arms[2*M]`← Contains the link ids for each particle in a simple array sorted by particle id.
char `nl[N]`← Stores the number of particle links sorted by particle id.
unsigned int `iarm[N]`← Contains the arms array index where starts the links for every particle, sorted by particle id.

The previous arrays are used to exploit the netlink information and reduce the computng time, but they must be generated carefully, as can be seen in the figure 13.

Afterwards, we must have structures to process the partially linked and disconnected particles, so we also allocate:

char `np[C]`← Contains the cell particles quantity sorted by cell id.
unsigned int `cells[n]`← Contains the particle ids sorted by the cell id where they are located, in order to access speedily to the particles of an specified cell id.
unsigned int `icell[C]`← Contains the cells array index where starts the particle ids located inside each
cell, sorted by cell id.

where C is the grid cells quantity, and n is the partially linked and disconnected particles. The figure 14 illustrates how are related the previous arrays.

In our implementation, the CPU controls the program flow, while the GPU computes the particle system processes. The main pipeline is

going to execute the following actions:

1. Generate the netlink.
2. Compute links data.
3. Sum particle forces from links.
4. Break over-stressed and over-strained links.
5. Process partially linked and disconnected particles.
6. Integrate over time.
7. Repeat from step 2 until the time simulated is enough

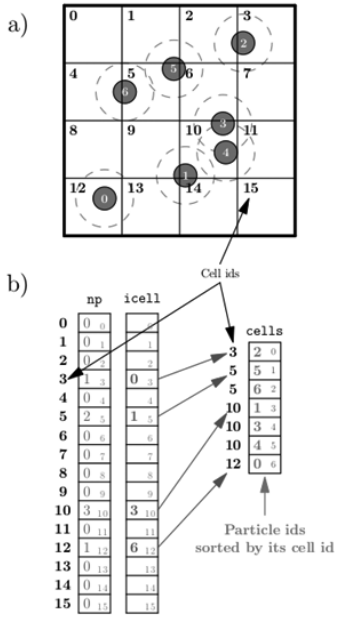


Figure 14: a) Partially linked particles spatially sorted using a grid, the cell ids are marked with black color. b) The np array contains the number of particles occupying each cell, while the cells array stores the particle ids sorted by the cell id where they are located, and the icell array contains the cells array index where start the particles of every cell, the empty spaces means that such cell doesn't have any particles. The gray color are used to identify the arrays index (C-style numbering).

The second and fifth steps are going to use a kernel to compute the potential energy and its derivative, however, the parameters depends on the potential energy function selected, the figure 15 shows how to implement the elastic potential functions.

```
/* Elastic potential sample parameters */
float pl_params[2] = {9e6, 2.5};
/* Copy the data to the device array dev_pl_params */

/* Elastic potential */
__device__ void pl(float r, float *params, float *out){
    float s = params[0];
    float lambda = params[1];
    *out = s*( (r*r)/(2*lambda) - r - lambda/2 );
}
/* Potential derivative */
__device__ void fl(float r, float *params, float *out){
    float s = params[0];
    float lambda = params[1];
    *out = s*( r/lambda - 1 );
}
```

Figure 15: GPU Implementation of the elastic potential functions. The parameters are defined inside the functions for clarity, but this is not necessary.

7.1 Netlink generation

This process is responsible to create the body discretization into small enough particles linked where the body is continuous. At implementation level, it should allocate and fill the following arrays:

`x[], links[], arms[], nl[],` and `iarm[].`

The job could be simplified using CAD software to generate the geometry.

7.2 Compute links data

Now we need to calculate the links information, such as the stress σ_{ij} (equation 9), the strain ε_{ij} (equation 10), the potential energy $U(r_{ij})$ from the i^{th} to the j^{th} particle, and the force G_{ij} from the ith to the jth particle (equation 1).

To accomplish the goal we use the kernel `process_links`, which have to compute the values of the `stress[], strain[], Ur[]` and `G[]` arrays, as can be seen in the figure 16.

```
__global__ void process_links(unsigned int *links,
             float *x, float *strain, float *stress,
             float *Ur, float *G,
             float *potential_params,
             void (*potential)(float, float*, float*),
             void (*force)(float, float*, float*),
             unsigned int *M){
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   while(i < *M){
      unsigned int pid1 = links[i*2];
      unsigned int pid2 = links[i*2+1];
      float dx1 = x[pid1*2]-x[pid2*2];
      float dx2 = x[pid1*2+1]-x[pid2*2+1];
      float r = sqrt(dx1*dx1 + dx2*dx2);
      float n[2] = {dx1/r, dx2/r };
      float p, g;
      potential(r, potential_params, &p);
      Ur[i*2] = p*n[0];
      Ur[i*2+1] = p*n[1];
      force(r, potential_params, &g);
      G[i*2] = g*n[0];
      G[i*2+1] = g*n[1];
      float lambda = potential_params[0];
      stress[i] = 4*fabs(g)/(PI*lambda*lambda);
      strain[i] = fabs(lambda-r)/lambda;
      i += blockDim.x * gridDim.x;
   }
}
```

Figure 16: GPU kernel to compute the links information. Some memory could be saved, such as the pid1, pid2, n[], Gnorm and lambda declarations, but for clarity we broke the equations into several lines of code. On the other hand, (*potential) and (*force) are pointers to the kernels which compute the potential energy and its derivative respectively, while the potential_params array contains the parameters needed.

7.3 Sum particles forces from link

At this stage we should sum the links forces and potentials at every linked particle, in other words, we should accumulate on the `F` and `U` arrays the previously computed data stored on the `G` and `Ur` arrays respectively, considering that for links between partially linked particles only the attraction part of the potential will be taken in account, because the repulsion part will be computed and summed again when we process the geometric grid. In order to complete the task we utilize the kernel `process_linked_particles` coded in the figure 17.

```
__global__ void process_linked_particles(
                        unsigned int *links,
                        unsigned int *arms,
                        char *nl,
                        unsigned int *iarm,
                        float *Ur, float *G,
                        float *U, float *F,
                        unsigned int *N){
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   char j;
   while(i < *N){
      U[i*2] = 0;
      U[i*2+1] = 0;
      F[i*2] = 0;
      F[i*2+1] = 0;
      for(j = 0; j < nl[i]; j++){
         unsigned int lid = arms[iarm[i]+j];
         /* Evaluate if both prt are partially linked */
         char arePL = (nl[links[lid*2]]<6) &&
                      (nl[links[lid*2+1]]<6);
         /* Take just attraction if arePL is true */
         char isAttr = G[lid]>0;
         float attr1 = Ur[lid*2] * (!arePL) +
               Ur[lid*2] * isAttr * arePL;
         float attr2 = Ur[lid*2+1] * (!arePL) +
               Ur[lid*2+1] * isAttr * arePL;
         /* Consider the particle link side */
         U[i*2] += (links[lid*2]==i)*attr1 -
                   (links[lid*2+1]==i)*attr1;
         U[i*2+1] += (links[lid*2]==i)*attr2 -
                     (links[lid*2+1]==i)*attr2;
         /* Do the same with the force... */
         attr1 = G[lid*2]*(!arePL) +
                 G[lid*2]*isAttr*arePL;
         attr2 = G[lid*2+1]*(!arePL) +
                 G[lid*2+1]*isAttr*arePL;
         F[i*2] += (links[lid*2]==i)*attr1 -
                   (links[lid*2+1]==i)*attr1;
         F[i*2+1] += (links[lid*2]==i)*attr2 -
                     (links[lid*2+1]==i)*attr2;
      }
      i += blockDim.x * gridDim.x;
   }
}
```

Figure 17: GPU kernel to accumulate the potential energy and force of linked particles. Note that the link id lid declaration is not necessary.

7.4 Break over-stressed and over-strained links

In this task we will break the over-stressed and over-strained links, increasing n (partially linked and disconnected particles quantity) and decreasing M (links quantity) in the fly. For the program concerns, we are going to update the `links[]`, `arms[]`, `nl[]`, `iarm[]`, `np[]`, `cells[]` and `icell[]` arrays according with the new values of M and n.

Due to the memory freeing and arrays reallocation, we do the job in the CPU. It don't have to be done every step, depending on the material dynamics, you could ignore this stage for tens of time steps.

7.5 Process partially linked and disconnected particles

Then, the interaction between partially linked and disconnected particles will be performed using a grid sorting structure, so we need the `cell_id` kernel to map the cell where each particle is located, as is shown in the figure 18 for the two dimensional case.

The geometrical grid is represented by the `np[]`, `cells[]` and `icell[]` arrays, built by the `count_cells_particles` and `create_grid` kernels (nicely defined in the figure 19), and `create_grid` kernels (nicely defined in the figure 19), in figure 20. This strategy reduces the neighbor search cost from $O(n^2)$ to O(n) (deeply explained at [7]), where $n < N$ because the completely linked particles doesn't are processed.

```
__device__ void cell_id(float *position,
                        unsigned int *cid,
                        void *grid_params){
   /* (GX1, GX2) <- Grid lower corner position.
    * CW <- Grid Cells Width.
    * CH <- Grid Cells Height.
    * NC <- Grid columns quantity.
    */
   float GX1 = (float) grid_params[0];
   float GX2 = (float) grid_params[1];
   float CW = (float) grid_params[2];
   float CH = (float) grid_params[3];
   unsigned int NC = (unsigned int) grid_params[4];
   *cid = (int)((position[0]-GX1)/CW)*NC +
          (int)((position[1]-GX2)/CH);
}
```

Figure 18: GPU kernel to get the cell id as a function of the particle position.

```
__global__ void count_cells_particles(float *x,
                        char *nl, char *np,
                        void *grid_params,
                        unsigned int *N){
   /* The np[] array must be in zeros */
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   while(i < *N){
      unsigned int cid;
      cell_id(&x[i*2], &cid, grid_params);
      atomicAdd(&np[cid], nl[i]<6);
      i += blockDim.x * gridDim.x;
   }
}
```

```
void CPU_fills_icell(char *np,
                     unsigned int *icell,
                     unsigned int C){
   unsigned int i;
   unsigned int acum = 0;
   for(i = 0; i < C; i++){
      icell[i] = acum;
      acum += np[i];
   }
}
```

```
__global__ void create_grid(float *x, char *nl,
                        unsigned int *cells,
                        char *np,
                        unsigned int *icell,
                        void *grid_params,
                        unsigned int *N){
   /* The cells[] array must have -1 in every entry */
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   while(i < *N){
      unsigned int cid;
      cell_id(&x[i*2], &cid, grid_params);
      /* Eval if the particle is partially linked */
      char isPL = (nl[i]<6);
      unsigned int j;
      for(j = 0; j < np[cid]; j++){
         char index = icell[cid]+j;
         /* Eval if cells array position is empty */
         char isEmpty = (cells[index]==-1);
         atomicAdd(&cells[index], (i+1)*isEmpty*isPL);
      }
      i += blockDim.x * gridDim.x;
   }
}
```

Figure 19: GPU kernels to create from scratch the geometrical grid structure.

7.6 Integrate over time

Now we are going to integrate the velocity and the dis- placement using the Velocity Verlet Algorithm. Technically speaking, we will compute the values for the x, v and a arrays using the integrate kernel detailed in the figure 21.

7.7 Main routine

Finally, we need put them all together in the correct order, as is shown in the figure 22.

```
__global__ void process_grid(float *x, float *F,
             unsigned int *cells, char *np,
             unsigned int *icell,
             float *potential_params,
             void (*potential)(float, float*, float*),
             void (*force)(float, float*, float*),
             void *grid_params,
             unsigned int *n, unsigned int *C){
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   while(i < *n){
      unsigned int cid1, cid2;
      cell_id(&x[cells[i]*2], &cid1. grid_params);
      char j, k;
      /* Iterate over surrounding cells */
      for(j = 0; j < 9; j++){
         unsigned int NC = unsigned int grid_params[4];
         cid2 = ((cid1-NC-1+j)*(j<3) +
                 (cid1-1+j)*(j>=3)*(j<6) +
                 (cid1+NC-1+j)*(j>=6)) % (*C);
         /* Iterate over cell particles */
         for(k = 0; k < np[cid2]; k++){
            unsigned int index = icell[cid2] + k;
            /* Evaluate if it's the same particle */
            char notMe = (index != cells[i]);
            float dx1 = x[index*2]-x[cells[id]*2];
            float dx2 = x[index*2+1]-x[cells[id]*2+1];
            float r = sqrt(dx1*dx1 + dx2*dx2);
            float normal[2] = {dx1/r, dx2/r};
            float p, g;
            potential(r, potential_params, &p);
            U[cells[i]*2] += notMe * p * normal[0];
            U[cells[i]*2+1] += notMe * p * normal[1];
            force(r, potential_params, &g);
            G[cells[i]*2] += notMe * g*normal[0];
            G[cells[i]*2+1] += notMe * g*normal[1];
;
         }
      }
      i += blockDim.x * gridDim.x;
   }
}
```

Figure 20: GPU kernel to process the partially linked and disconnected particles using a geometrical grid structure.

```
__global__ void integrate(float *x, float *v,
                          float *a, float *F,
                          float *mass, float *Delta_t,
                          unsigned int *N){
   /* Using Velocity Verlet Algorithm */
   unsigned int i = threadIdx.x +
                    blockIdx.x * blockDim.x;
   while(i < *N){
      /* Integrate displacement */
      x[i*2] += v[i*2]+0.5*a[i*2]*(*Delta_t);
      x[i*2+1] += v[i*2+1]+0.5*a[i*2+1]*(*Delta_t);
      /* Get acceleration from force */
      float at[2] = {a[i*2], a[i*2+1]};
      a[i*2] += a[i*2]/(*mass);
      a[i*2+1] += a[i*2+1]/(*mass);
      /* Integrate velocity */
      v[i*2] += 0.5*(at[i*2]+a[i*2])*(*Delta_t);
      v[i*2+1] += 0.5*(at[i*2+1]+a[i*2+1])*(*Delta_t);
      i += blockDim.x * gridDim.x;
   }
}
```

Figure 21: GPU kernel to process the partially linked and disconnected particles using a geometrical grid structure.

```
int main(){
   /* Define n, N, M, C and pl_params[2] in some way */
   /* Allocate arrays into the CPU */
   /* Generate the netlink */
   /* Allocate arrays into the GPU device */
   /* Copy arrays from CPU to GPU device */
   while(!finish){
      process_links<<<128,128>>>(dev_links, dev_x,
                                 dev_strain,
                                 dev_stress,
                                 dev_Ur, dev_G,
                                 dev_pl_params,
                                 pl, fl, dev_M);
      process_linked_particles<<<128,128>>>(dev_links,
                                 dev_arms, dev_nl,
                                 dev_iarm,
                                 dev_Ur, dev_G,
                                 dev_U, dev_F,
                                 dev_N);
      /* Here you will paste the break links code to:
       *    > Update M and n.
       *    > Free and reallocate links[] and arms[].
       *    > Update nl[] and iarm[] values.
       */
      count_cells_particles<<<128,128>>>(dev_x,
                                 dev_nl,
                                 dev_np,
                                 dev_grid_params,
                                 dev_N);
      /* Copy results from GPU to CPU */
      CPU_fills_icell(np, icell, C);
      /* Copy results from CPU to GPU */
      create_grid<<<128,128>>>(dev_x, dev_nl,
                               dev_cells, dev_np,
                               dev_icell,
                               dev_grid_params,
                               dev_N);
      process_grid<<<128,128>>>(dev_x,
                               dev_F, dev_cells,
                               dev_np, dev_icell,
                               dev_pl_params,
                               pl, fl,
                               dev_grid_params,
                               dev_n, dev_C);
      integrate<<<128,128>>>(dev_x, dev_v, dev_a,
                             dev_F, dev_mass,
                             dev_Delta_t, dev_N);
   }
   /* Free device memory */
   /* Free CPU memory */
   return 0;
}
```

Figure 22: CPU main function.

8. Results

The figure 23 shows the particles velocities when the press starts to compress the body, simulating a Brazilian compressive test. The particle interaction using potentials generate waves to transfer the force peer to peer, as in a giant pool game.

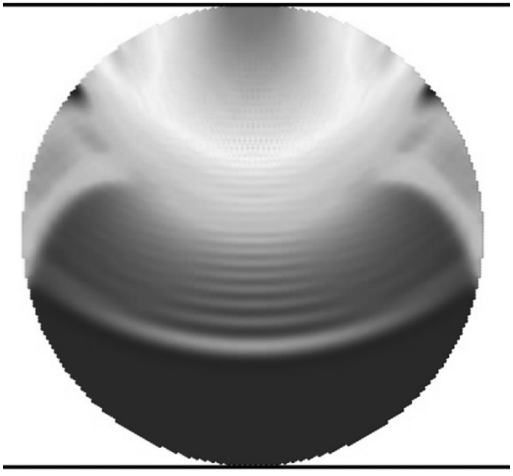


Figure 23: Particle velocities when the press start to go down.

When the elastic potential is used, we obtain failures such as the showed in the figure 24, which can be compared with the 25, 26 and 27, that are using different potentials.

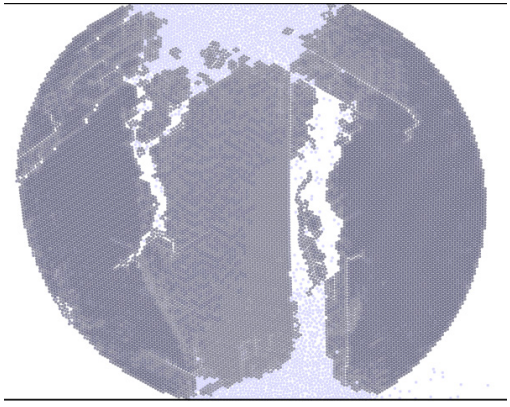


Figure 24: Failure produced using the elastic potential.

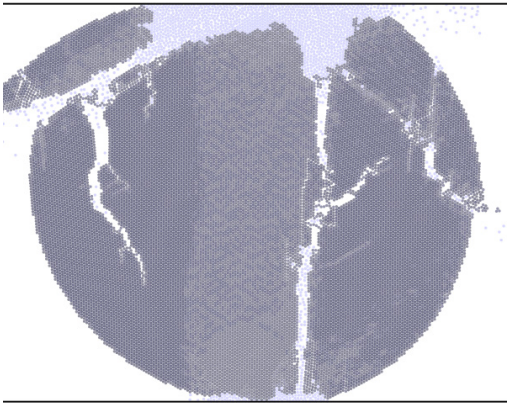


Figure 25: Failure produced using the elastic potential.

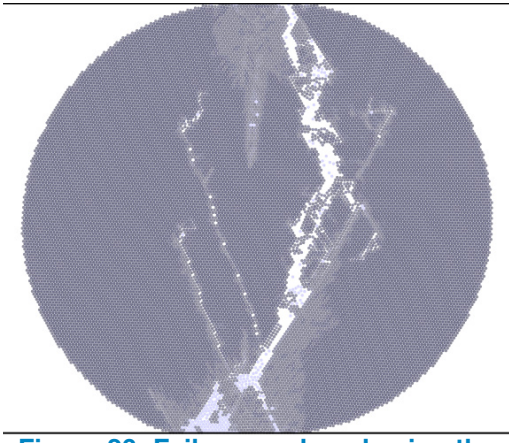


Figure 26: Failure produced using the logarithmic potential.

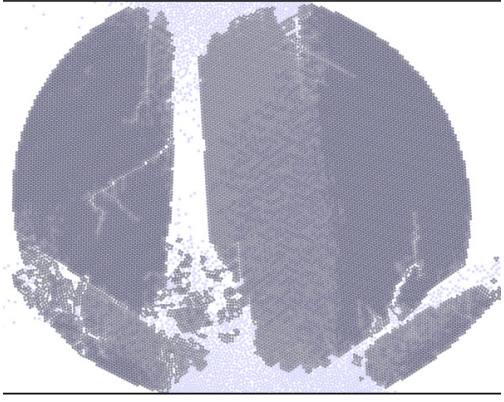


Figure 27: Failure produced using the rational potential.

9. Conclusions

The netlink generation will be faster than any kind of mesh because its geometrical pattern, and the neighbor particles search implementation is simpler than those methods using different particles sizes, this facility make the computation very efficient in any architecture, especially in the GPU due to its ability to process thousands of threads each clock cycle, making the rock fracture simulation possible to greater scales, but reminding the George E. P. Box cite: "essentially, all models are wrong, but some are useful".

10. References

[1] Cundall, Peter A. "A computer model for simulating progressive large scale movements in blocky rock systems". Proceedings of the international symposium on rock fracture, October 1971. International Society for Rock Mechanics (ISRM), vol 1, paper no II-8, pp 129-136.

[2] Jing Lanru. & Stephansson, O. "Fundamentals of Discrete Element Methods for Rock Engineering Theory and Applications". Elsevier, Volume 85, Pages 1-545, 2007

[3] Munjiza, Antonio. "The combined finite-discrete Element Method". John Wiley & Sons, Ltd, 2004.

[4] Matej Hudak. "Physical Animation of wetting terrain and erosion". Proceedings od CESCG, 2011.

[5] Oden, J. Tinsley. "An introduction to Mathematical Modeling: A course in mechanics", Wiley 2011.

[6] Buehler, Markus J. "Atomistic Modeling of Materials Failure". Massachusetts Institute of Technology, Springer 2008.

[7] Nguyen, Hurbert. "GPU Gems 3". Chapter 31, nvidia 2011.

Optimization and Parallelization Experiences Using Hardware Performance Counters

Fernando G. Tinetti[1],[2], Sergio M. Martin[3], Fernando E. Frati1, Mariano Méndez[1]
[1]*III-LIDI, Fac. de Informática, UNLP*
Calle 50 y 120, 1900, La Plata, Argentina
[2]*Comisión de Inv. Científicas de la Prov. de Bs. As.*
3Universidad Nacional de La Matanza
Florencio Varela 1903 - San Justo, Argentina
email: {fernando, fefrati}@info.unlp.edu.ar,
smartin@ing.unlam.edu.ar, marianomendez@gmail.com

Abstract

Current hardware for compute intensive tasks includes a large amount of processing facilities, which are sometimes difficult to use in an optimized way. High performance computing (HPC) is always focused in solving challenging (or, at least, compute intensive) problems for which the response time is the priority. We have been working from two different but usually complementary research problems: a) updating and parallelizing legacy (HPC/numerical) software, and b) analyzing different problems and approaches to optimization and parallel processing in clusters. We have found that raw hardware event counters do not always directly provide useful information. We also found some guidelines for evaluating performance using those counters in the context of optimization and parallelization. In this article, we present those guidelines along with the performance evaluation tools that we used to determine objectively what parts of the algorithm offered better chances of improvement.

Keywords: High Performance Computing, Source code optimization, Performance Evaluation, Parallel Computing, Cluster Computing.

1. Introduction

High performance computing (HPC) in clusters is one of the most popular approaches for solving compute, or more specifically, numerical intensive workloads/tasks. Computers currently used as cluster's nodes usually range from low end desktop/PC computers (which are economically cheap) to high end servers/PCs. Terms such as *low* end *desktop computer* and *server computer* have many definitions depending on hardware, hardware corporations marketing, and specific market status. We will refer to them just as PCs or cluster nodes, emphasizing on their proven advantageous features, which at least include low price and high availability [10] [13]. Furthermore, we will refer to clusters as shown

in Fig. 1: a local area network of computers made up of commodity hardware components. Cluster nodes, as shown in Fig. 1 are expected to be multi-core computers sharing memory, which is the current conceptual configuration even in NUMA (Non-Uniform Memory Access) hardware [4]. The most commonly used interconnection network is 1Gb/s Ethernet (availability/commodity hardware, cost), but almost any other interconnection network could be used, such as those specifically designed for HPC in clusters, such as Infiniband.

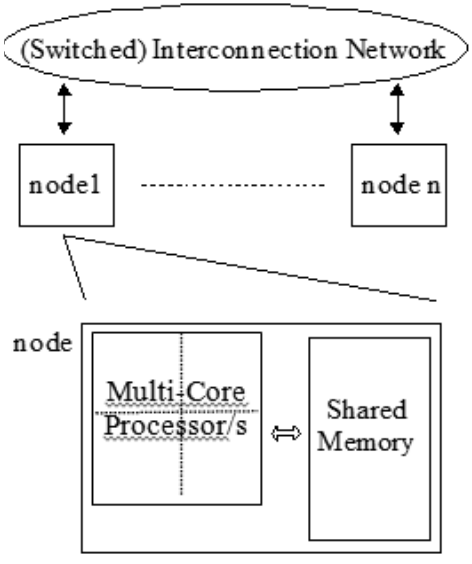


Figure 1. Cluster used for HPC.

There are two levels or types of parallel processing in current clusters: intra node and cluster-wide (inter-nodes). The most common programming models for parallel processing in clusters are the message passing and shared memory (threaded) ones. Message passing is usually implemented by using some (message passing library) implementation of MPI (Message Passing Interface) [9], such as OpenMPI. Shared memory parallel processing can be implemented in terms of OpenMP [10], which is currently implemented by most C and Fortran compilers. Also, it is possible to combine both programming models (message passing and shared memory) in the so called hybrid approach as in [12].

From the point of view of HPC applications, legacy software has a strong need to be updated to the new multi-core environment/s. By definition, every programmer has to deal with strong problems when facing (updating, etc.) legacy software, but HPC applications in particular have to be parallelized since the processors' clock rate is not going to increase beyond 3.8-4.2 GHz in the near future [14] [5] [16]. However, legacy software is not the only software which needs to be parallelized or even updated. New algorithms and new parallel platforms are always analyzed in order to model and optimize performance. Also, new applications and new applications sizes are taken into account as more cores are included in a multicore chip and more computers are available (or interconnected) for parallel computing.

New microprocessors also include new microarchitectures which not always are fully exploited for maximum performance. Moreover, new microprocessors often provide access to (internal) performance counters in order to analyze and optimize runtime performance [1] [6]. Interestingly, performance is associated to debug in [1], which includes a chapter named "Software Debug and Performance Resources" (Chapter 13). This *association* would also provide an idea of the related complexity.

This possibility of measuring hardware

counters as a way to determine the performance of the algorithms being executed motivated us to use profiling tools, such as Perf [19], and Perfsuite [8], to determine how our algorithms performed in such low-level. Before we did this research, we had to use a "guessing" and code analysis approach to determine which could be the causes of performance degradation, especially on multi-core architectures. Even though we could achieve some improvement, the exact causes of the initial problems remained undiscovered.

However, using tools to access and analyze hardware counters, allowed us to get a more objective idea of how to improve our algorithms. In this article, we will show how we used Perf and PerfSuite to determine which were the specific hardware counters that provided us with the clues as to how to increase their performance. As a result of using these tools, we could objectively determine which were the causes of several of their performance-degrading problems. We would expect other scientist programmers to apply these same tools on their algorithms, and analyze these same counters in search of analog opportunities to improve their performance.

The rest of this article is organized as follows. Section 2 describes the classical performance metrics and hardware performance event counters (which are also referred to as hardware performance monitoring event counters). The main concepts for performance evaluation and our initial work on a *real world* legacy program are explained in Section 3, with specific results we deduced hardware counters and experimentation. Section 4 introduces a complete example based on a well known problem and focused on parallelization, and the effect of some classical optimizations on parallel performance and the so called *memory* wall related to parallel computing on multicore multiprocessors. Finally, includes several conclusions and further work taking into account the work done and explained in this paper, mainly on legacy as well as parallel code.

2. Performance Metrics and Counters

Performance has been always the ultimate goal in the HPC field. Usually, sequential performance has been measured directly in terms of runtime or rates of instructions or floating point operations per second, such as MFlop/s (millions of floating point operations per second). Parallel performance has been measured also in terms of (plain) runtime or with more specific metrics such as Speedup and Efficiency (usually as functions of the number of processors), defined as in Eq. (1) and Eq. (2) respectively [16], where p is the number of processors, op_st is the runtime of the optimum sequential algorithm, and $pt(p)$ is the parallel elapsed runtime using p processors. It is expected (but not always possible) to obtain a $Speedup(p)$ value near p, since it means that every processor has been used for a 1/p fraction of the total processing to be done.

$$Speedup(p) = \frac{op\_st}{pt(p)} \quad (1)$$

It is worth noting that $0 < Efficiency(p) \leq 1, \forall p$ (at least in non pathological cases), and values of $Efficiency(p)$ near 1 indicates that about 100% of computing resources are used at runtime.

$$Efficiency(p) = \frac{Speedup(p)}{p} \quad (2)$$

We could argue that in the end, every performance metric is computed using elapsed runtime. On one hand, it is fair enough, since runtime is *independent* of the underlying processing hardware. But on the other hand, given a performance value, it is hard or unlikely to guess specific performance problems and/or penalties. There are some *classical* guidelines to look for performance penalties in the parallel processing area. Most of the parallel performance optimization ideas and specific algorithms try to solve several communication, synchronization, and/or computing (un)balance problems. At this point, hardware performance counters provide the most specific and accurate information of the hardware performance.

Having access to hardware performance counters tends to reduce the number of unknowns/guesses at least on specific parts of the available hardware. Thus, we can use performance (monitoring) counters for identifying (some) performance penalties and evaluate algorithms changes which are made for optimization/s [9] [2] [3] [7]. Unfortunately, hardware counters report very specific and low level information, which is not always directly/easily related to the algorithms. We will show that performance counters have to be carefully analyzed and, sometimes, specific experiments have to be carried out to collect relevant data. We will avoid using proprietary manufacturers' tools and hardware/model low level information whenever possible (e.g. Intel definition and usage of incore-uncore events). From this point of view, tools such as perf and API (Application Programming Interfaces) such as PAPI (Performance API) [18] provide more or less general and vendor independent information.

3. Legacy Code Example

We have selected a global climate model as an example of legacy code example: GISS-AOM(C4x3) from GISS, the NASA Goddard Institute for Space Studies [11]. We have experimented with this legacy Fortran code looking for several interesting information regarding compilers, performance counters, legacy code optimization, and prospective issues for parallelization:

- Legacy code approach/methodology: starting with profiling, and advancing to source code behavior and optimization. We consider the starting points as a completely unknown legacy code, so that experiments, profiling, and monitoring events provide a basis for a methodology on enhancing such legacy source code.
- Similarities/differences among compilers, mostly from the point of view of performance and optimization levels. We have used gfortran and ifort (Intel Fortran compiler). We are not interested in compiler-specific optimizations and language (Fortran) extensions, and this is why we have used "-O[1/2/3]" optimization levels.
- Optimization level impact on performance, based on monitoring events reported by the processor/s.

We have used PerfSuite as a high level approach for gathering hardware performance events information, i.e. to avoid doing a direct analysis of lower level tool results such as perf

[19] and instrumentation libraries such as PAPI [18]. A general syntax of usage for PerfSuite as used in our experiments is shown as follows:

```
$ psrun -o perfOutput  -a -p ./program
< I > output.prt
```

Where *perfOutput* is the output previously generated by perf, program is the binary to profile, and *output.prt* is the result file to be generated by PerfSuite. *Output.prt* will then contain an easy to interpret set of profiling data.

The experiments were carried out using the Intel Core i5-2400 (3.1 GHz) and the Intel Xeon x5550 (2.66 GHz), with Linux (kernel 2.6.38). Most of the results are similar in both compilers and platforms, so we present averages and/or main characteristics which are independent of the specific hardware and compiler. Profiling has shown that more than 80% of the total runtime is spent in 21 subprograms (Fortran functions and subroutines). This means that most of the optimization and parallelization effort should be employed in those 21 subprograms. For large legacy applications, this could be a huge reduction in the amount of source code to work on.

As a first step in the process of enhancing performance of the legacy software, we used several optimization options, shown in Table I. Most of the improvement is obtained in the first optimization level, -O1, which implied to reduce the runtime to the 62% of the non-optimized binary code. In this case, the second optimization level added some gains (which is not always obtained, i.e. in all hardware and compiler variants).

Table I. Optimization gains (time reduction)

-O1	-O2	-O3
0.62	0.52	0.51

We have collected available information from event counters and one of the first problems we found can be exemplified with the data in Table II, for a reduced number of counter data. Clearly, raw numbers collected from the hardware do not provide any useful information. However, with the event counters data it is possible to obtain information not only about optimizations, but also about optimization focus.

Table III shows the improvement in cache misses, i.e. the reduction in (instructions and data) caches and TLB (Translation Lookaside Buffer) misses when using O1 relative to those obtained with –O0 (no compiler optimization). Most of the performance improvement provided by the -O1 optimization level is due to the very good work of the compiler on the instructions. Compilers are able to optimize the available resources in hardware pipelines, superscalar units, branch prediction, and almost every hardware facility for ILP (Instruction Level Parallelism). Branch instruction event counters can be also used to support this behavior:

- Conditional branch instructions have been lowered by 11.95%.
- Mispredicted conditional branch instructions have been decreased by 36.49 %.
- Not taken conditional branch instructions have been decreased by 37.97 %.

Table II. Raw event counters numbers

Event	Result
Conditional branch instructions	111948989803
Branch instructions	130698632623
Conditional branch instructions mispredicted	1849659750
Conditional branch instructions not taken	24036555287
Floating point divide instructions	23168250057
Floating point operations	56515006398
Level 1 data cache	5609489524
Level 1 instruction cache misses	172229550
Level 2 data cache accesses	29185949460
Level 2 instruction cache accesses	222243552
Level 2 instruction cache misses	74014152
Level 2 cache misses	11534648622

Table III shows that almost no performance enhancement is obtained by taking advantage of data cache/s.

Table III. Misses (caches and TLB) improvement with -O1

Event	Reduction
Level 1, 2, and 3 data cache misses	< 5%
Level 1 instruction cache misses	> 98%
Level 2 instruction cache misses	> 85%
Level 2 instruction cache accesses	> 85%
Instruction TLB misses	53%

It is worth noting that legacy code parallelization has a strong relationship with data usage (accesses to cache/s and main memory, i.e. the memory hierarchy), since it is nearly impossible to recode or change the underlying algorithms on almost unknown code. At least in the initial parallelization stages/tasks, the basic algorithm is kept unchanged and the way in which data and threads and/or processes is defined so that every processing facility is used. It is clear from Table III that specific data cache improvement has to be taken into account (preferably at early stages) in the parallelization work.

4. Parallelization Example

In [15] is reported the work on a very simple but time consuming algorithm used for N-body/particle simulation. It has also been shown how tiling (a very common optimization technique for memory accesses) has made possible a huge performance gain for two, four, and eight cores running OpenMP threads. More specifically:

- Tiling does not provide a huge improvement in sequential computing: less than 10% [15].
- When each OpenMP thread runs the tiled code, efficiency, calculated as in Eq. (2), becomes greater than 95% for 2, 4, and 8 cores (using two quad-core processors). Conversely, when tiling is not used, efficiency is just 0.69 for two threads and drops to 0.51 for eight threads (running on eight cores sharing main memory) [15].

We had not to determine if it is possible to identify the memory contention using performance monitoring counters. To answer this question, we used perf [19] in order to experiment and gather information about hardware. We have found that perf is a simple yet powerful tool, and easier to install than PerfSuite, which also depends on other software/libraries, such as PAPI. We run specific experiments with the program (coded in C language) for identifying cache events, i.e.:

- A small number of bodies: 100000, this is useful to avoid long experiments runtime as well as issues due to memory

accesses other than memory contention.
• Several different numbers of threads: 2, 4, 6, and 8.

The syntax used for our experiments using perf is shown as follows:

```
$ perf stat –e L1-dcache-load-misses
        -e LLC-load-misses
    ./sim4OMP 10000
```

Where all the profiled counters are defined after a *–e* flag. In our case, we only need L1 cache load misses, and LLC load misses to be profiled. Finally, the binary name and its arguments are specified. In our case, sim4OMP represents the OMP version of our N-body algorithm that is being run for 10000 and one step.

The results of this experiment are shown in Table IV, in terms of Mflop/s and Efficiency (Eff.), and the percentage of Last Level Cache (LLC) and L1 data cache misses relative to the "previous" number of threads experiment. Initially, we ran experiments with 2, 4, and 8 threads, and we found a huge performance loss specifically for 8 threads: efficiency dropped to about 0.47 for 8 threads, when it was about 0.98 when 4 threads process data. Thus, we added several experiments with one more number of threads in between 4 and 8, i.e. 6, so that we are able to analyze the performance problem in detail.

Table IV. Cache load misses (relative %) and Mflop/s

#cores	1	2	4	6	8
L1-dclm(1)	100	100	92	326	220
LLC-lm(2)	100	88	43	599	358
Mflop/s	1858	3669	7297	8275	6993
Eff.	1	0.99	0.98	0.74	0.47

(1) L1-dclm: Level 1 data cache load misses
(2) LLC-lm: Last Level Cache load misses

We use "previous" in the sense of number of "previous number of threads" because it highlights the main differences related to algorithm scalability. For example, Table IV shows that the number of L1 data cache misses is almost constant for 1, 2, and 4 threads, since for 2 cores, it is about 100% of the misses for reported for 1 core, and, rather surprisingly, the number of L1 data cache misses for 4 threads is 92% of the L1 data cache misses for 2 threads. Minor differences could appear given to the statistical nature of event counting by multiplexing hardware counters.

The parallel performance is very good For 2 and 4 threads, i.e. both have more than 95% of efficiency. There is a clear problem in performance and scalability for 6 and 8 threads. One could be confused by the increase of 326% of L1 data cache misses for 6 threads regarding the number of L1 data cache misses for 4 threads. Actually, the real problem is the huge increase of 599% LLC-lm for 6 threads regarding those for 4 threads. Part of that 599% LLC-lm is "hidden" by the other level/s of cache/s, but it is clear that data from main memory is not arriving at the rate the processor needs for computing. More specifically, there are 6 threads requiring data to process from the same memory, and from those 6 threads, 4 threads share the LLC in a quad-core processor while the other two needs almost the same data to process in the other processor. The situation is even worse for 8 threads, as expected: there are 358% more LLC load misses than for those happening for 6 threads.

These results show clearly that using more cores imply more shared memory contention

when the threads are not running in the cores of a unique processor (thus having access to the shared LLC). Even though processing-time is reduced by adding additional cores, the benefits of adding them are severely reduced. This lead us to think that there may be a shared LLC cache throughput limit that, at a certain demand, collapses and reduces the overall performance. Once this limit is reached, penalization for L1 cache misses is much more time-consuming. Therefore, the overall per-core speedup is reduced. In our case, this limit is reached when using more than 4 cores (on the 6 and 8 core cases).

Previous research [17] indicate us that this so-called *memory wall* is quickly found not only when a processor runs faster but also when more cores run in the same processor, which is our case for the 6-8 core run. The number of cores per processor and the number of processors sharing main memory can be increased, but performance would be unacceptable (or even disappointing) if memory and/or the algorithms are not improved/optimized/adapted. There is clear evidence now about what we explained by the end of the previous section: data cache improvement has to be taken into account (preferably at early stages) for parallel computing. We have collected and shown in Table IV specific performance monitoring event counters evidence.

5. Conclusions and Further Work

Until we performed this research, we had to rely on guessing techniques, and high-level analysis to determine which were the causes of performance degradation on our parallel and legacy codes. Although some of our results were positive [15] [16], we were still lacking a formal method to know the exact causes (or their amount) that caused this degradations. Now, most of this classical guesswork used in optimization and parallelization performance analysis is now supported by performance counters.

We have shown valuable specific runtime information is now possible to be gathered for almost unknown (legacy) and self developed source code. Furthermore, we can access to and collect information from the performance event monitoring counters from different programming languages, such as Fortran and C (which, besides, are among the most popular languages in HPC).

Even when the optimization and parallelization problems remain the same in the long term, performance counters provide very useful information for both tasks. Moreover, new possibilities could be explored based on specific combinations of values. We have also shown that single events are not necessarily good enough for performance analysis, but having a minimum knowledge of hardware architecture/s will lead to combine events so that the searching space for optimizations and parallelization would be narrowed down without missing important details.

Performance event counters do not solve by themselves optimization- and parallelization-related problems, they help taking informed/supported decisions. Moreover, low level hardware details (those accounted for by event counters) could produce an unmanageable amount of information and should be used carefully. So far, there is no general methodology for approaching a program using performance event counters, and we are working on producing one. The final objective would be a

tool o set of tools for aiding HPC programmers for optimization and parallelization.

Similar experiences have also been shared by scientific programmers, for both high-performance computing and parallel algorithms in these last few years [20] [21]. These experiences along with the ones presented in this article can be used for further research trying to establish a knowledge base for algorithm optimization using profiling tools.

References

[1] Advanced Micro Devices, AMD64 Technology, AMD64 Architecture Programmer's Manual, Volume 2: System Programming, March 2012.

[2] Advanced Micro Devices, Software Optimization Guide for AMD Family 10h and 12h Processors, February 2011. Available at http://support.amd.com/us/Processor_ TechDocs/40546.pdf

[3] Advanced Micro Devices, Software Optimization Guide for AMD Family 15h Processors, Available at http://support.amd.com/us/Processor_TechDocs/47414_15h_sw_opt_guide.pdf

[4] Advanced Micro Devices, Surviving and Thriving in a Multi-Core World, Nov. 2006.

[5] Shekhar Y. Borkar, Pradeep Dubey, Kevin C. Kahn, David J. Kuck, Hans Mulder, Stephen S. Pawlowski, Justin R. Rattner, Platform 2015: Intel Processor and Platform Evolution for the Next Decade, Intel White Paper, 2005. Available at ftp://download.intel.com/tech nology/computing/archinnov/platform2015/download/Platform_2015.pdf

[6] Intel Corp., Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide, Part 2, Aug. 2012.

[7] Intel Corp., Intel Microarchitecture Codename Nehalem Performance Monitoring Unit Programming Guide (Nehalem Core PMU), 2010. Available at http:// software.intel.com/sites/default/files/m/5/2/c/f/1/30320-Nehalem-PMU-Programming-Guide-Core.pdf

[8] Rick Kufrin, "PerfSuite: An Accessible, Open Source Performance Analysis Environment for Linux", 6th International Conference on Linux Clusters: The HPC Revolution, Chapel Hill, NC, April 2005.

[9] David Levinthal, "Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 processors", Intel Corp., 2009. Available at http://software.intel.com/ sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf

[10] Gregory Pfister, In Search of Clusters, 2nd Edition, Prentice Hall, Dec. 1997, ISBN 0138997098.

[11] Gary L. Russell, James R. Miller, David Rind, "A Coupled Atmosphere-Ocean Model for Transient Climate Change Studies", Atmosphere-ocean, 33(4), 1995.

[12] Gerald Schubert, Holger Fehske, Georg Hager, Gerhard Wellein, "Hybrid-parallel sparse matrix-vector multiplication with explicit communication overlap on current multicore-based systems", Parallel Processing Letters 21(3), 2011.

[13] Thomas L. Sterling, Beowulf Cluster Computing With Linux, MIT Press, 2001, ISBN 0262692740.

[14] Herb Sutter "The Free Lunch is Over: a Funda-

mental Turn Toward Concurrency in Software", Dr. Dobb's Journal, Vol. 30, No. 3, 2005, http://www.gotw.ca/publica tions/concurrency-ddj.htm.

[15] Fernando G. Tinetti, Sergio M. Martin, "Sequential and Shared and Distributed Memory Parallelization in Clusters: N-Body/Particle Simulation", 24th IASTED International Conference on Parallel and Distributed Computing and Systems, PDCS 2012, November 12 – 14, 2012, Las Vegas, USA.

[16] Fernando G. Tinetti, Mariano Méndez, Fortran Legacy Software: Source Code Update and Possible Parallelization Issues, ACM Fortran Forum, April 2012, Vol. 31, No. 1.

[17] Wlliam A. Wulf, Sally A. Mckee, "Hitting the Memory Wall: Implications of the Obvious", ACM SIGARCH Computer Architecture News, Vol. 23, 1995.

[18] PAPI, http://icl.cs.utk.edu/papi/

[19] Tutorial - Perf Wiki, https://perf.wiki.kernel.org/in dex.php/Tutorial

[20] Wucherl Yoo, "Automated Performance Characterization of Applications Using Hardware Monitoring Events", Doctoral Dissertation. University of Illinois at Urbana-Champaign. 2012, Illinois, USA.

[21] Xingfu Wu, Valerie Taylor, "Performance Modeling of Hybrid MPI/OpenMP Scientic Applications on Large-scale Multicore Supercomputers", Elsevier, Journal of Computer and System Sciences. May 2012, Vol. 79, No. 8.

Parallel Encoding of Audio in Architectures Multi-Core

Angel González Méndez*[1], Oscar Alvarado Nava[2]
[1]*Department of Electrical Engineering, UAM - Iztapalapa, D.F., México*
[2]*Department of Electronic, UAM - Azcapotzalco, D.F., México*
E-mail: agonzale@xanum.uam.mx, oan@correo.azc.uam.mx

Abstract

The technology is growing rapidly and this is reflected with the rise of multi-core processors and thanks to this increased the dissemination of parallel processing in audio and video encoders have the commitment that higher quality processing the time will be longer; to speed up the response time to of these encoders can make use of multi-core architectures. This paper presents the adaptation of the encoder LAME for the encoding process WAV to MP3 can occur in parallel so the obtained acceleration is proportional to the number of cores in the architecture without losing the audio quality. LAME was chosen since is categorized as one of the best MP3 encoders besides being free software is used in research and in commercial and free software.

Keywords: multi-core, parallel encoding, lame, audio encoding, parallelism, parallel processing

1. Introduction

Nowadays the audio capture process is made so that losses are minimized possible. The audio present in the most digital media is encoded in some compressed format, which implies loss of data; even so, the audio encoding/compression algorithms are based on the human ear perceptible aspects which makes this data loss is not perceived. The operations required by the audio encoding/compression algorithms involve high costs both time and processing, for example encoding to MP3[1] uses the Modified Discrete Cosine Transform(MDCT), Fast Fourier Transform(FFT) and Huffman Coding.

With the passage of time shared-memory systems are integrating more cores in each processor, which increases the computing capability of these systems. Having multiple processing units computing needs of several applications require us to develop software that takes advantage of the multi-core architectures. Nonetheless, to use this type of architectures in an efficient way should switch from the sequential processing to parallel processing and one option is to do it by programming multi-threaded.

Parallel computing uses multiple processing elements simultaneously to solve a problem, this accomplished by dividing the problem into independent parts so that each processing element can perform his part of the algorithm at the same time than others. For the design of parallel programs through the multi-threaded programming should consider issues such as: the complexity of the problem, synchronization,

communication and competitive conditions.

The format MPEG-1 Audio Layer III better known as MP3 is the format with more extension on the Internet, since provides a CD quality with an approximate relationship in size 11 to 1.

In this paper we propose an adaptation of the LAME encoder[2] for audio encoding WAV to MP3 advantage of new multi-core architectures, in the rest of the document is presented in Section II a description of sequential LAME encoding algorithm, Section III shows the LAME parallel proposal, in Section IV shows the results obtained, finally in Section V presents the conclusions.

2. Related Work

Multimedia applications are characterized by their high computational cost, some of the most demanding applications are encoding / decoding images, audio and video. This paper emphasizes the audio encoding, there are different methods to optimize the audio encoding and get acceleration in time, and these methods can be grouped into three major categories: parallelization, optimization and dedicated hardware.

In [3], [4] and [5] shows various methods for parallel encoding. In [3] is parallelized MPEG-2 video encoder using MPI, obtaining an acceleration of 3.5 times using four processors. In [4] is parallelized JPEG image encoder using OpenMP, obtaining acceleration 20 times using twenty-two processors. In [5] is parallelized H.264 video encoder using MPI, the results show that the acceleration is affected by the speed of the network, the source of the data, the complexity of the algorithm, among others.

Another method for the acceleration time is the optimization of the algorithms. In [7] propose a modification to the MDCT, reducing time MP3 encoding between 5% and 10%. In [8] propose the use of vector instructions (SIMD) in the encoding of MPEG-2/4 Advanced Audio Coding to optimize operations of vector algebra and mathematics, obtaining an acceleration close to 15 times.

Finally in [5] shows the hardware implementation of the MDCT, to accelerate the MP3 audio encoding.

3. LAME Sequential Encoding

MPEG-1 Audio Layer III, is a compression standard that differs from other encoders due to its compression performed without making assumptions about the nature of the audio source. This encoder takes advantage of the limitations of the human ear as it removes those parts that are imperceptible, hence the resulting audio is not identical to the original it has lost, and even so the human is not capable of perceiving these losses. In Figure 1 shows the basic diagram of a MPEG-1 encoder.

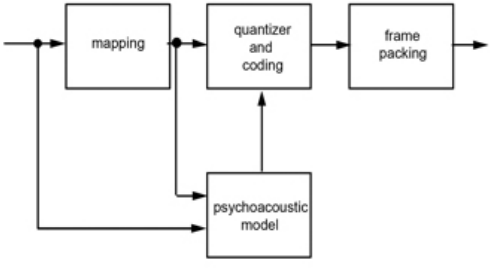


Figure 1. Basic diagram of an encoder MPEG-1.

On the structure of the audio MP3 noteworthy that the encoder generates independent MP3 frames of variable-length, the length varies according to the audio input; frames make up a header that contains a synchronization word, the audio features and the audio encoded. For the generation of a required MP3 frame is required 1152 audio samples, which apply the encoding algorithm.

The software LAME is an MP3 encoder licensed under the LGPL[9]. LAME it is highly used both in industry and in research due to its quality and to their constant improvement, industry has been integrated into various software's commercial and free, in the area of research have been made different works for the improvement or optimization of the encoder shown in Figure 1 as you can see from [6,7].

The sequential algorithm of LAME consists of the following stages: get properties from the input file, initialize the properties of coding, reading samples, coding of samples and writing MP3 frames; in Algorithm 1 shows the stages and their order.

Algorithm 1: Sequential Algorithm of LAME

```
Require: I input file
Require: O output file
Local: A input file properties
Local: B encoding properties
Local: P samples read
Local: Q encoding samples

  A=getProperties(I);
  B= initialize(A);
  do{
    P = read(A,I);
    Q = encoding_samples(B,P);
    write(O,Q);
  }while(I != EOF);
```

4. LAME Parallel Proposal

Parallel computing has been seen as an evolution, which is trying to solve the most complex problems and achieve results in the shortest time possible. To the above there are several models[10] for the design of some parallel programs from them are: Shared-Memory with Processes, Shared-Memory with Threads, Distributed-Memory with Message Passing, Parallelism in Data, Hybrid, SIMD & MIMD.

There are various proposals to improve of the MPEG audio or video encoder, either in performance or quality, some modifications are: use SIMD vector instructions[8], hardware acceleration[6] and amendment to MDCT[7], among others.

Analyzing Algorithm 1, we saw that it is highly parallelizable, since all samples are subject to the same encoding process. The model of parallelism that was used was that of data parallelism (with threads), which is based on the application of the same operations to a data series, the tasks in the model used are obtaining data, processing of data, and writing the data.

The basic structure of the proposed algorithm is shown in Figure 2 in which can be observed that each read sample has its corresponding MP3 frame on the output, in the figure will have *2* threads, at time t1 are encoded samples *1* and *2* and at time tm are encoded samples n and *n-1*.

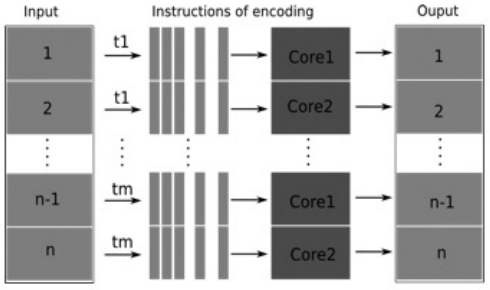


Figure. 2. Parallel Encoding.

4.1 Reading

The variables thread_turn and trun1 are used for reading, turn1 is global and thread_turn is local, turn1 always starts at zero. The value of turn1 this bounded between 0 and p where p is a number of threads to use, at the beginning of the program p is greater than or equal to 2. For each read turn1 is incremented by one and thread_turn is equal to turn1-1; if a thread reads zero samples EOF is equal to TRUE and thread_turn equals to -1; read and the increase of turn1 are made on an exclusive basis.

4.2 Encoding

Once read audio samples by each thread applies the encoding process to their respective samples. If any thread has no samples to encode it remains in waiting that the others finish.

4.3 Writing

The variables thread_turn and trun2 are used for write turn2 is global and thread_turn is local, turn2 always starts at zero. The value of turn2 this bounded between 0 and turn1. To write turn2 must equal thread_turn, for each write turn2 is incremented by one. The write ordering will be 0,1,2 ... turn1-1; write and the increase of turno2 are made on an exclusive basis.

4.4 Synchronization

For synchronization will occupy a barrier and a lock, have three global variables turn1, turn2 and EOF, in addition each thread has a unique identifier tid and its respective thread_turn.

4.5 Algorithms

The Algorithm 2 is initialization of global variables, obtaining the characteristics of the input file and the creation of the threads. In Algorithm 3 is shown in pseudocode that described in Sections 4.1, 4.2 and 4.3.

Algorithm 2: Initialization and creating threads.

```
Require: I input file
Require: O output file
Require: A input file properties
Require: N number of threads
Local: Tids Array of thread id's

   barrier_init(mybarrier, N);
   lock_init(mylock);
   EOF=FALSE;
   A=getProperties(I);

   For i in 0 to N do
     Tids[i] = thread_create(i, &A, &I, &O,
   encode);
   done

   For i in 0 to N do
     thread_join(Tids[i]);
   done
```

5. Results

For testing we used three audio WAV files, which have the following sizes: File1 has a size of 42MB, File2 has a size of 133MB and finally the File3 has a size of 531MB. The tests were conducted in two computers which have the following characteristics: Computer1 has the OS Mountain Lion OSX, consists of one quad-core processor Intel(R) Core(TM) i5 at a speed of 2.5GHz and 4GB of RAM, Computer2 has the OS Debian Linux 6.0, consists of four quad-core processors Intel(R) Xeon(R) at a speed of 2.53 GHz and 31GB of RAM.

5.1 Encoding Times

Table 1 shows the encoding times using LAME software, in Tables 2 and 3 show the encoding times using proposed software.

Algorithm 3: Function *encode*

```
Require: I reference to input file
Require: O reference to output file
Require: A reference to input file properties
Require: tid id of thread
Local: B encoding properties
Local: P samples read
Local: Q encoding samples
Local: trhread_turn turn

B=initialize(A);

while !EOF do
   thread_turn=-1;
   if tid ==0 then
      turn1=0;
      turn2=0;
   end if

   barrier_wait(mybarrier);

   lock(mylock);
   P = read(A,I);
   if lenght(P)>0 then
      thread_turn=turn1;
      turn++;
   else
      EOF=TRUE;
   end if
   unlock(mylock);

   barrier_wait(mybarrier);

   Q = encoding_samples(B,P);

   wr = TRUE;
   while wr && thread_turn>0 done
      lock(mylock);
      if turn2 == thread_turn then
         write(O,Q);
         turn2++;
         wr = FALSE;
      end if
      unlock(mylock);
   done

   barrier_wait(mybarrier);

done
```

Table 1. Encoding Times using LAME

Computer	File1 [s]	File2 [s]	File3 [s]
1	7.64	24.17	93.73
2	10.70	34.56	135.06

Table 2. Encoding Times using Proposed Software (Computer1)

Threads	File1 [s]	File2 [s]	File3 [s]
2	3.83	12.05	46.44
3	2.68	8.44	33.55
4	2.17	6.84	26.60

Table 3. Encoding Times using Proposed Software (Computer2)

Threads	File1 [s]	File2 [s]	File3 [s]
2	5.15	17.82	70.68
4	2.83	9.03	35.55
6	1.90	6.13	23.98
8	1.45	4.60	17.86
10	1.50	4.88	18.74
12	1.30	4.22	16.40
14	1.22	3.89	14.94
16	1.11	3.55	13.84

5.2 Acceleration and time gain

The acceleration obtained in a file i using j threads is obtained from Equation 1, the average acceleration using j threads is obtained from Equation 2 and the average gain in time using j threads is obtained from Equation 3.

$$A[i][j] = \frac{t_{sec}}{t_{par}[j]} \quad (1)$$

$$A_{avg}[j] = \frac{A[1][j] + A[2][j] + A[3][j]}{3} \quad (2)$$

$$G_{avg}[j] = (1 - \frac{1}{A_{avg}[j]}) \times 100 \quad (3)$$

Based on the results in Tables 1, 2 and 3 we see that we have obtained with 2 threads a average acceleration of 2.0 times in the *Computer1* and 1.91 in the *Computer2*, this means a saving in time of 50.09% in the *Computer1* and 47.87% in the *Computer2*, with 4 threads is obtained acceleration average of 3.51 times in *Computer1* and 3.79 times in *Computer2*, this means a saving of 71.58% time in the *Computer1* and 73.62% in the *Computer2* , with 16 threads is obtained acceleration average of 3.54 times in *Computer1* and 9.63 times in *Computer2*, this means a saving of 71.82% time in the *Computer1* and 89.62% in the *Computer2*. Overall we can say that our adaptation had satisfactory results with respect to the sequential version and the works presented in Section 2.

In Figures 3 and 4 show graphs of the encoding times of *File3* in PC1 and PC2 respectively, in the firs graph we can see that at the time of use a number of threads greater than the number of cores generate unpredictable behavior it depends on the Operating System.

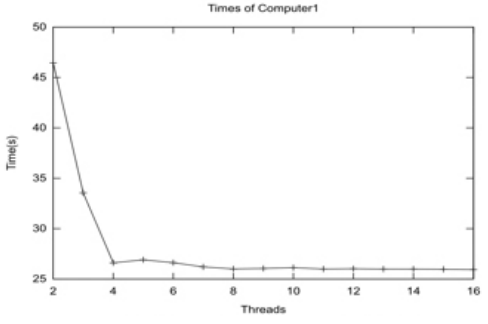


Figure 3. Encoding times of *File3* in *Computer1*

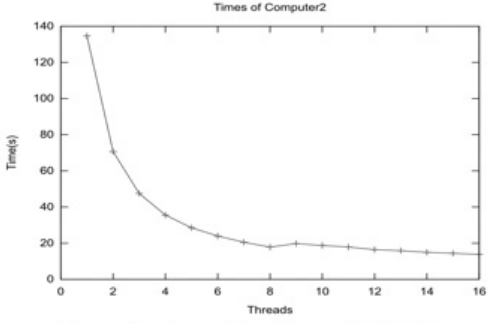


Figure 4. Encoding times of *File3* in *Computer2*

5.3 Integrity and Spectrum Analysis

To verify the integrity of the audio encoded with the proposed software was used mp3check, mp3check not found errors in the encoded files. For spectral analysis was used the software audacity to verify that files encoded with proposed software have the same quality as those encoded with LAME, the results are presented in Figures 5 and 6 in these it can be seen that both files have the same quality.

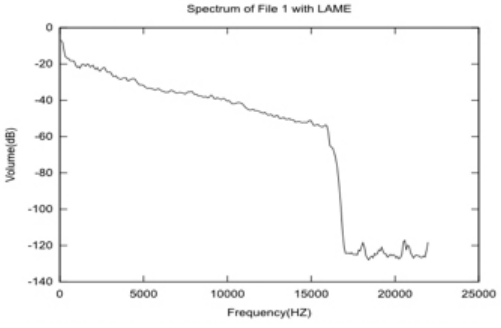


Figure 5. Spectrum of *File1* encoded with LAME

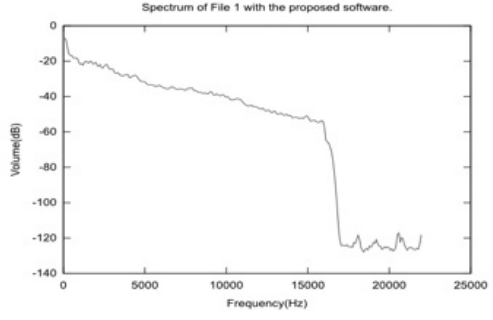


Figure 6. Spectrum of *File1* encoded with the Proposed Software

6. Conclusions

It has been shown that the parallel encoding, significantly reduces the execution time, the solution described in this paper is open for optimizations, such as: optimize or modify the model parallelism, use SIMD instructions with threads, among others. Currently multimedia applications require increased processing power and increasingly include more cores per processor forcing us to take advantage of current and future architectures.

7. References

[1] ISO/IEC IS 11172-3, "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s, Part 3: Audio," 1993

[2] "LAME - Lame Ain't an MP3 Encoder," http://lame.sourceforge.net/, January 2013

[3] Ravin Sachdeva; Kaushik Saha, "Parallel MPEG-2 Video Encoder," IEEE International Conference on Consumer Electronics (ICCE), pp.159-160, 9-12 January 2011

[4] Castells-Rufas, D.; Joven, J.; Carrabina, J., "Scalability of a Parallel JPEG Encoder on Shared Memory Architectures," International Conference on Parallel Processing (ICPP) 39th, pp.502-507, 13-16 September 2010

[5] Fan Niu; Dongmei Li; Shuai Peng, "Parallel Coding Efficiency Analysis of H.264 on PC Cluster," Symposium on Photonics and Optoelectronic (SOPO), pp.1,4, 19-21 June 2010

[6] Xingdong Dai; Wagh, M.D., "An MDCT Hard-

ware Accelerator for MP3 Audio," *Application Specific Processors, 2008. SASP 2008. Symposium*, pp.121-125, 8-9 June 2008

[7] Mathew, M.; Bhat, V.; Thomas, S.M.; Changhoon Yim;, "Modified MP3 encoder using complex modified cosine transform," *Multimedia and Expo, 2003. ICME '03.* , pp. II- 709-12 vol.2, July 2003

[8] Dimkoviae, I.; Milovanoviae, D.; Bojkoviae, Z., "Fast software implementation of MPEG advanced audio encoder," *Digital Signal Processing. 14th International Conference,* pp. 839- 843 vol. 2, 2002

[9] "GNU Lesser General Public License," http://www.gnu.org/licenses/lgpl.html, January 2013

[10] "Introduction to Parallel Computing," https://computing.llnl.gov/tutorials/parallel_comp/, January 2013

Suffix Array Performance Analysis for Multi-core Platforms

Cesar Ochoa[1], Veronica Gil-Costa[1,2] and A. Marcela Printista[1,2]
[1]*LIDIC, University of San Luis, Argentina*
[2]*CONICET, Argentina*
e-mail: {cochoa, gvcosta, mprinti}@unsl.edu.ar

Abstract

Performance analysis helps to understand how a particular invocation of an algorithm executes. Using the information provided by specific tools like the profiler tool Perf or the Performance Application Programming Interface (PAPI), the performance analysis process provides a bridging relationship between the algorithm execution and processors events according to the metrics defined by the developer. It is also useful to find performance limitations which exclusively depend on the code. Furthermore, to change an algorithm in order to optimize the code requires more than understanding the obtained performance. It requires understanding the problem being solved. In this work we evaluate the performance achieved by the suffix array using a 32-core platform. Suffix arrays are efficient data structures devised to solve complex queries in a number of applications related to text databases, as for instance biological databases. We run experiments to evaluate hardware features directly aimed to parallelize computation. Moreover, according to the results obtained by the performance evaluation tools, we propose an optimization technique to improve the use of the cache memory. In particular, we aim to reduce the number of cache memory replacements performed every time a new query is processed.

Keywords: suffix Array, multi-core platforms.

1. Introduction

Several microprocessor design techniques have been used to exploit the parallelism inside a single processor. Among the most relevant techniques we can mention bit-level parallelism, pipelining techniques and multiple functional units [10,17]. These three techniques assume a single sequential flow of control which is provided by the compiler and which determines the order of execution in case there are dependencies between instructions. From the point of view of a programmer, this kind of internal techniques have the advantage of allowing parallel execution of instructions by a sequential programming language. However, the degree of parallelism and pipelining obtained by multiple functional units is limited.

Recently, the work in [8, 9] showed that increasing the number of transistors on a chip leads to improvements in the architecture of a processor to reduce the average time of execution of an instruction. Therefore, an alternative approach to take advantage of the increasing number of transistors on a chip is to place multiple cores on a single processor chip. This approach has been used in desktop

processors since 2005 and is known as multi-core processors. These multi-core chips add new levels in the memory (cache) hierarchy. There are significant differences in how the cores on a chip or socket may be arranged. These multi-core chips add new hierarchical levels inside the memory (cache) [10]. Memory is divided into pages and pages can be variable size (4K, 64K, 4M). Each core has an L1 cache capable of storing a few Kb. A second level called L2 cache is shared by all cores grouped on the same chip and its storage capacity has few Mb. An example is the Intel machine Sandy Bridge-E which has up to 8 cores on a single chip, 32K L1 cache, 256 K L2 cache, and up to 20M L3 cache that is shared by all cores in the same chip.

To take advantage of current commodity architectures, it is necessary to devise algorithms that exploit (or at least do not get hindered by) these architectures. There are several libraries for multi-core systems like OpenMP[12] which has a high level of abstraction, TBB [13] which uses cycles to describe the data parallelism, and others like IBM X10 [4] and Fortess [1] which focuses on data parallelism but also provides task parallelism. A multi-core system offers all cores quick access to a single shared memory, but the amount of memory available is limited. The challenge in this type of architecture is to reduce the execution time of programs.

In this work, we propose to analyze the performance of a string matching algorithm in a multi-core environment, where communication is made only through cache and main memory, there are no dedicated instructions to do neither synchronization nor communication, and dispatch must be done by (slow) software. In particular, we propose to use the Performance Application Programming Interface (PAPI) tool and the Perf tool for performance analysis (which helps to improve applications by revealing its drawbacks) over the suffix array index [11] and then, guided by the results obtained, we propose an optimization technique to increase the number of cache hits. The suffix array index is used for string matching, which is perhaps one of those tasks on which computers and servers spend quite a bit of time. Research in this area spans from genetics (finding DNA sequences), to keyword search in billions of web documents, to cyber-espionage. In fact, several problems are also recast as a string matching problem to be even tractable.

Recently, the work in [18] presents a modified version of a PSISA tool for multi-cores. This version is efficiently used to find the structural similarities between different proteins and maintains the load balance between cores. Also the authors in [19] present a simple shared memory parallel algorithm which does not take advantage of the multi-core cache memory neither the fact that suffix arrays always access to the same elements at the beginning of the search algorithm.

The remainder of this article is organized as follows. In Section 2, we describe the suffix array index and the search algorithm. In Section 3, we present the parallel algorithm and the proposal optimization. In Section 4 we present experimental results. Conclusions are presented in Section 5.

2. Suffix array

String matching is perhaps one of the most studied areas of computer science [5]. This is not surprising, given that there are multiple areas of application for string processing, being information retrieval and computational

biology among the most notable nowadays. The string matching problem consists of finding some string (called the pattern) in some usually much larger string (called the text).

Many index structures have been designed to optimize text indexing [14,16]. In particular, suffix arrays [11] has already 20 years old and it is tending to be replaced by indices based on compressed suffix arrays or the Burrows-Wheeler transform [2,3], which requires less memory space. However, these newer indexing structures are slower to operate. A suffix array is a data structure that is used for quickly searching for a keyword in a text database.

Essentially, the suffix array is a particular permutation on all the suffixes of a word. Given a text T[1..n] over an alphabet $\sum$, the corresponding suffix array SA[1..n] stores pointers to the initial positions of the text suffixes. The array is sorted in lexicographical order of the suffixes. As shown in Figure 1 for the example text "Performance$"—the symbol $\$ \notin \sum$ is a special text terminator, that acts as a sentinel. Given an interval of the suffix array, notice that all the corresponding suffixes in that interval form a lexicographical subinterval of the text suffixes.

	1	2	3	4	5	6	7	8	9	10	11	12
T:	P	e	r	f	o	r	m	a	n	c	e	$

i	text suffix	SA[i]
1	$	12
2	ance$	8
3	ce$	10
4	e$	11
5	erformance$	2
6	formance$	4
7	mance$	7
8	nce$	9
9	ormance$	5
10	Performance$	1
11	rmance$	6
12	rformance$	3

Figure 1: Suffix array for the example text "Performance$".

```
Search(SA, pattern X, pattern size L)
left =1
right=n
while (left<right)
    center = (left+right)/2
    strncpy(buff, T+SA[center-1], L)
    if (cmp=strcmp(X,buff) == 0)
       return (center-1);
    else if (cmp<0)
          right =center-1
       else
          left  = center+1
```

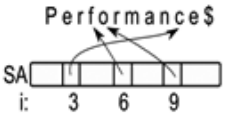
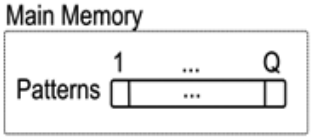


Figure2: Search algorithm for a pattern X of size L.

The suffix array stores information about the lexicographic order of the suffixes of the text T, but there is no information about the text itself. Therefore, to search for a pattern X[1..m], we have to access both the suffix array and the text T, with length |T|=n. Therefore, if we want to find all the text suffixes that have X as a prefix—i.e., the suffixes starting with X, and since the array is lexicographically sorted, the search for a pattern proceeds by performing two binary searches over the suffix array: one with the immediate predecessor of X, and other with the immediate successor. We obtain in this way an interval in the suffix array that contains the pattern occurrences. Figure 2 illustrates how a pattern search is performed. Finally, Figure 3 illustrates the code of the sequential search algorithm for NQ patterns. To this end, all patterns of length L are loaded into main memory (to avoid the interference of disk access overheads). The algorithm scans sequentially the patterns array using the next() function, and for each pattern X the SA search

algorithm is executed.




```
for(int q=0; q<NQ; q++)
    X =Patterns.next()
    results = SA.Search(SA,X,L)
```

Figure3: Sequential search algorithm.

3. Shared memory parallelization

Parallelization for shared memory parallel hardware is expected to be both, the most simple and less scalable parallel approach to a given code [15]. It is simple, due to the shared memory paradigm requires few changes in the sequential code and it is almost straightforward to understand for programmers and algorithm designers.

In this work we use the OpenMP [12] library, which allows a very simple implementation of intra-node shared memory parallelism by only adding a few compiler directives. The OpenMP parallelization is made with some simple changes to the sequential code. To avoid read/write conflict, every thread should have a local buff variable which stores a suffix of length L, also local variables *left, right* and *cmp* are used to decide the subsection of the SA where to continue the search (see Figure 2 above). To avoid using a critical section which incurs into computational overhead (delay in execution time), we replace the result variable of Figure 3 by an array *results*[1..NQ]. Therefore, each thread stores the results for the pattern Xi into results[i]. The "for" instruction is divided among all threads by means of the "#pragma omp for" directive. Figure 4 shows the threaded execution using the OpenMP terminology. Also, the sched_setaffinity function is used in this implementation to obtain performance benefits and to ensure that threads are allocated into cores belonging to the same sockets.

```
omp_set_num_threads(NT)  //Number of threads
#pragma omp parallel private(tid,X) shared(L,results)
  tid = omp_get_thread_num()
  #pragma omp for
    for (int q=0; q<NQ; q++)
      X = Patterns[q]
      results[q]  = SA.Search(SA,X,L)
```

Figure 4: Parallel search algorithm.

We further investigate the effect of using a local SA index per thread or a global shared SA index. To this end, we performed some experiments to measure the speedup (measured as the sequential execution time divided by the parallel execution time) achieved with both approaches. Figure 5 shows the results obtained with up to 32 threads, an index size of 801M and pattern length of L=10 and L=20. In both cases, efficiency is improved by using a local data structure (17% with L=10 and 41% with L=20). Hence, despite performing read-only access to the SA index there is an overhead associated with the search operation over the shared data structure when many threads process the search patterns. This overhead includes administration (creation/deletion) of the local variables such as buff, cmp, *left, right,* every time we perform a call to the Search() function. But as the local approach requires more memory, it can be used only when NT*sizeof(SA_Index) < Memory size, where NT is the number of threads.

3.1. Optimization

The suffix array search algorithm presents features which make it suitable for improving its efficiency in multi-core processors. In particular, we propose to take advantage of the binary search algorithm which makes some elements of the suffix array more feasible to be accessed than others. Namely, the center element of the suffix array and the left and right centers as shown in Figure 6 for a pattern length L=3.This effect is known as temporal locality.

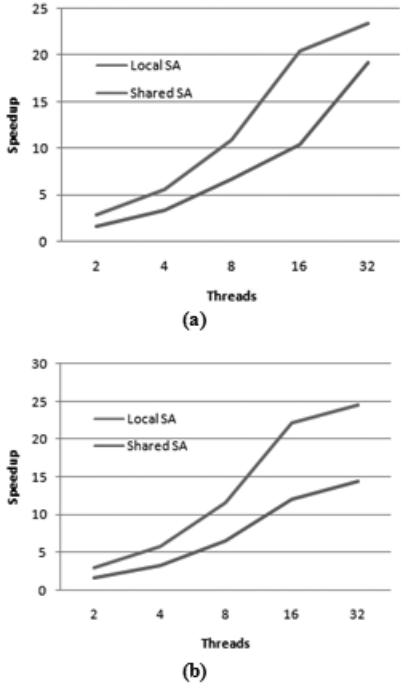
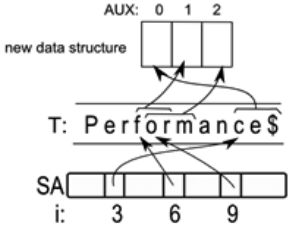


Figure 5: Speedup achieved when each thread uses a local copy of the index (Local SA) and when all threads share the same SA index. (a) Results obtained with L=10 and (b) L=20.

Therefore, we propose to introduce an extra data structure storing the most frequently referenced elements of the suffix array. This new data structure is small enough to fit into a single block of the low level cache memory and it is used to prune the search of patterns. The proposed algorithm is detailed in Figure 6. At the beginning of a new pattern search (iter=1), we access to the second position of the AUX data structure. If we have to continue the search at the left side of the text stored into AUX[1], the next iteration (iter=2) compares the first element of AUX with the search pattern X, otherwise it compares the third element of AUX with the pattern X.




```
Search(SA, pattern X, pattern size L)
left =1;  right=n;  step = -1;  it =0;
while (left<right)
  center =(left+right)/2
  if (step==-1)
   strncpy(buff, T+AUX[1], L)
   else if ( step == 1 && it==1 )
    strncpy( buff, AUX[0], L ); //left
    else if ( step == 2  && it==1 )
      strncpy( buff, AUX[2], L ); //right
      else
       strncpy( buff, T+SA[center-1], L );
  if (cmp=strcmp(X,buff) == 0)
       return (center-1);
  it++;
  if (cmp<0)
     right =center-1
     step = 1
  else
     left  = center+1
     step = 2
```

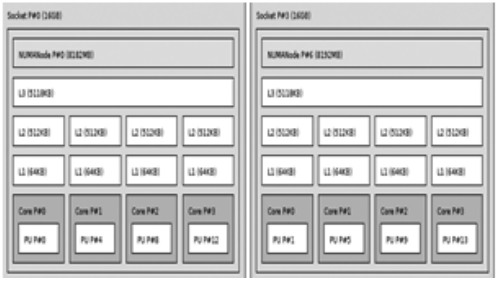
Figure 6: Cache optimization.

6. Evaluation

6.1. Computing Hardware and Data Preparation

Experiments were performed on a 32-core platform with 64GB Memory (16x4GB), 1333MHz and a disk of 500GB. 2x AMD Opteron 6128, 2.0GHz, 8C, 4M L2/12M L3, 1333 Mhz Maximum Memory. Power Cord, 250 volt, IRSM 2073to C13. The operating system is Linux Centos 6 supporting 64 bits. As shown in Figure 7, we used the hwlock (Hardware Locality) tool [7] which collects all information from the operating system and builds an abstract and portable object tree: memory nodes (nodes and groups), caches, processor sockets, processor cores, hardware threads, etc.

To evaluate the performance of the SA index we use a 3-megabyte and 100-megabyte DNA text from the Pizza&Chili Corpus[6]. The resulting suffix array requires 25M and 801M respectively. The text length in each case is n=3145728 and n=104857600. For the queries, we used 1000000 random search patterns of length 5, 10, 15 and 20.



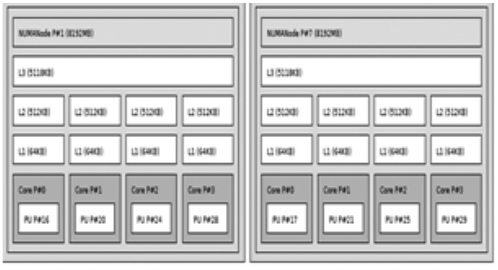
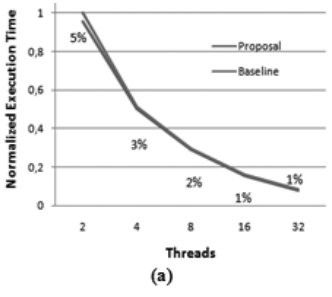


Figure 7: Four sockets with two nodes. Each node has four cores.

6.2. Performance Evaluation

In this section we evaluate the performance of the SA index using the PAPI tool version 5.0.1. In particular we measure the cache hit ratio, as well as the execution time. We use the PAPI_thread_init function which initializes thread support in the PAPI library. When all threads have finished their work, we executed the PAPI_unregister_thread function. We also used the Perf performance tool version 3.2.30. We counted the L1 data cache hits and the Last level cache (LLC) hits.

In Figure 8 and Figure 9 we show results obtained for a small index of 25M and different patterns lengths. Figure 8.(a) shows that the normalized running time obtained with both the baseline and our proposed optimization algorithm for a pattern length of L=5 are almost the same.



(a)

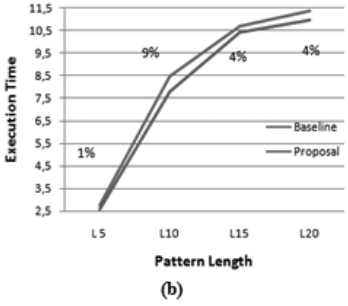
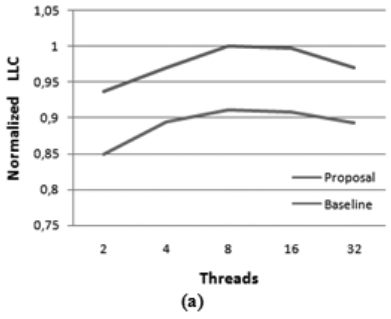


Figure 8: Results obtained with a small index size. (a) Execution time reported by the baseline and the proposal approach for a pattern length L=5; (b) Execution time for different pattern length and 32 threads.

We show results normalized to 1 in order to better illustrate the comparative performance. To this end, we divide all quantities by the observed maximum in each case. Figure 9.(a) shows the LLC hits reported for the same experiment. Our proposed cache-based algorithm repots an improvement of almost 20% in average which remains constant with different number of threads. However this benefit is not reflected on the running time. Figure 8.(b) shows that the best performance is achieved by our proposal for a pattern length of L=10. This results are confirmed in Figure 9.(b) where the best ratio (running time reported by the baseline divided by the running time reported by the proposal) is obtained with a pattern length L=10.



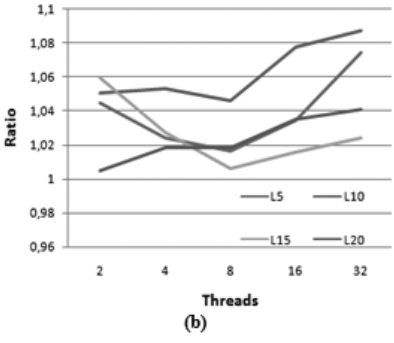
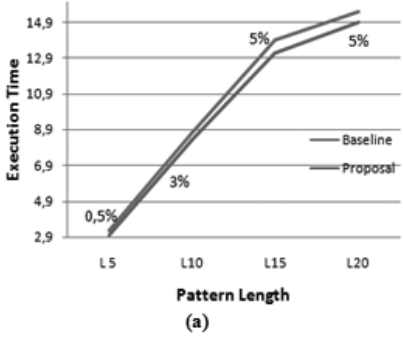


Figure 9: Results obtained with a small index size. (a) LLC hits for L=5 and (b) Ratio between the baseline execution time and the proposal approach execution time with 32 threads.



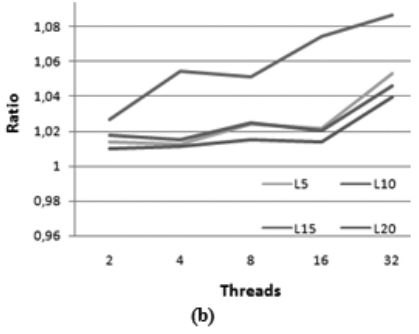


Figure 10: Results obtained with a larger index size. (a) Execution time for different pattern length and 32 threads; (b) Ratio between the baseline execution time and the proposal approach execution time.

Figure 10 and Figure 11 show results obtained with a large index of size 801M. In this case, the best performance of our optimization is reported with a pattern length of L=15 and L=20.

These results are confirmed in Figure 10.(b) where the best ratio between the running time reported by the baseline and the running time reported by the proposal is given for a search pattern L=15. Figure 11.(a) and (c) show the LLC hit for L=5 and L=20. In both figures, the LLC hits tend to go down as we increase the number of threads. In the former, our proposal obtains a gain of 15% in average but in the last figure the gain is reduced to 4%.

5. Conclusions

In this work we analyzed the performance of the suffix array index by means of the Perf and PAPI tool on a multi-core environment. Parallel codes were implemented with the OpenMP library. Experiments were performed with different index size and different patterns length over a 32-core platform. We ran experiments to evaluate hardware features directly aimed to parallelize computation. Results show that read-only operations performed over shared data structures affect performance, specially running time. We also proposed an optimization scheme which allows increasing the number of hit cache and tends to improve running time.

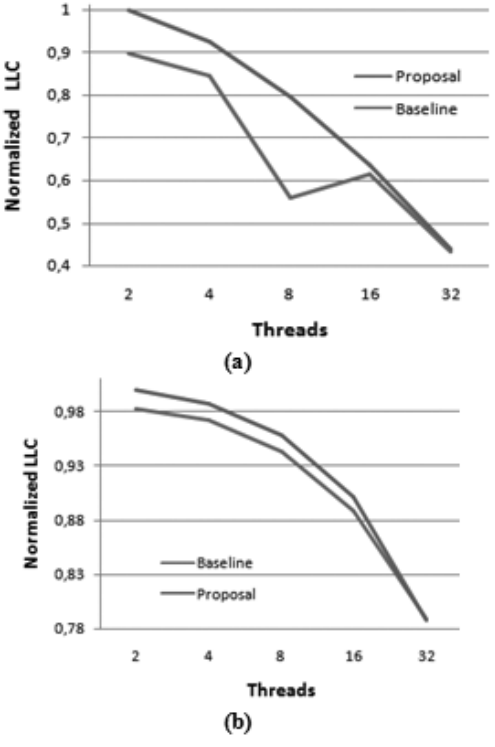


Figure 11: Results obtained with a larger index size. (a) LLC hits for L=5 and (b) LLC hits for L=20.

6. References

[1] E. Allen and D. Chase and J. Hallett and V. Luchangco and W. Maessen and S. Ryu and S. Tobin-Hochstadt, S. The Fortress Language Specification, version 1.0beta. 2007.

[2] Donald Adjeroh, Tim Bell, and Amar Mukherjee. The Burrows-Wheeler Transform: Data Compression, Suffx Arrays, and Pattern Matching. Springer, 2008.

[3] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. Research Report 124, Digital Systems Research Center, Palo Alto California, May 1994.

[4] P. Charles and C. Grothoff and V.A. Saraswat and C. Donawa and A. Kielstra and K. Ebcioglu and von Praun and V. Sarkar. X10: an object oriented approach to non-uniform cluster computing. In Internacional Conference on Object Oriented Programming, Systems, Languages and Applications. pages 519-538. 2005.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms (3rd ed.). MIT Press, 2009.

[6] Ferragina, P. and Navarro, G. 2005. The Pizza&Chili corpus — compressed indexes and their testbeds.

[7] The Portable Hardware Locality (hwloc): http:// www. open-mpi. org/ projects/ hwloc/

[8] J. L. Hennesy and D. A. Patterson. Computer Architecture - A Quantitative Approach. Morgan Kaufmann, 4th edition, 2007

[9] J. L. Hennesy and D. A. Patterson. Computer Organization & Design - The Hardware/Software Interface. Morgan Kaufmann, 4th edition, 2008.

[10] Georg Hager, Gerhard Wellein. Introduction to High Performance Computing for Scientists and Engineers. Chapman & Hall/CRC Computational Science. 2010.

[11] Manber, U. and Myers, G., “Suffix arrays: A new method for on-line string searches”, SIAM J. Computation, vol.22, no.5, pp.935-948, 1993.

[12] OpenMP Architecture Review Board, OpenMP Application Program Interface - Version 3.1, July 2011.Available athttp://openmp.org/wp/

[13] J. Reinders. Intel Threading Building Blocks: Outfitting C++ for Multicore Processor Parallelism. OReilly (2007).

[14] Jens Stoye. Suffix tree construction in ram. In Encyclopedia of Algorithms. Springer, 2008.

[15] Fernando G. Tinetti, Sergio M. Martin. Sequential Optimization and Shared and Distributed Memory Optimization in Clusters: N-BODY/Particle Simulation. Parallel and Distributed Computing and Systems -PDCS. 2012

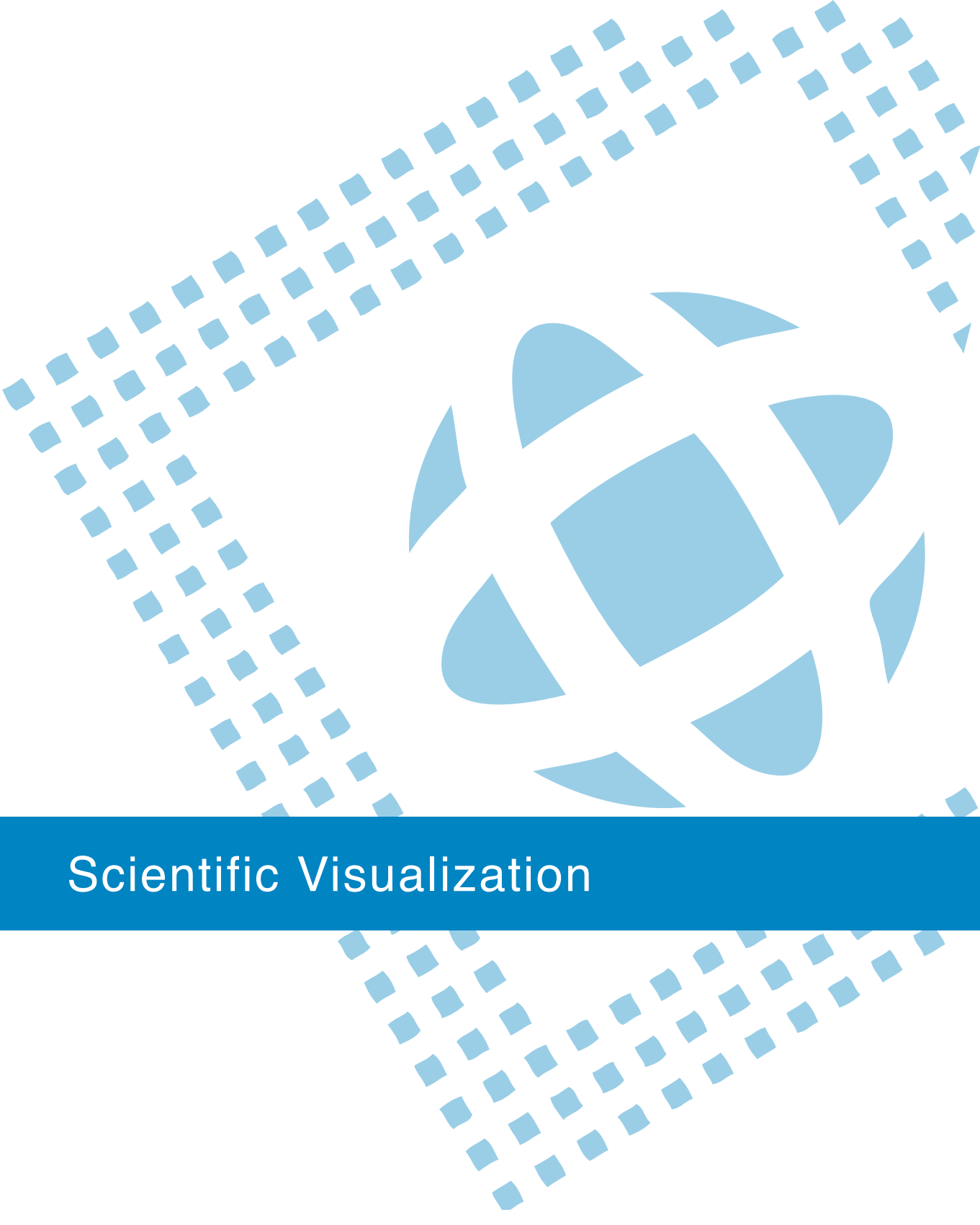
[16] Peter Weiner. Linear pattern matching algorithms. In Proceedings of the 14th Annual Symposium on Switching and Automata Theory, pages 1-11, 1973.

[17] R. J. Bailey, D. C. Defoe, R. H. Halverson, N. L. Passos, and R. P. Simpson. A Study of Software Pipelining for Multi-Dimensional Problems. In the Proceedings of the 13th International Conference on

Parallel and Distributed Computing Systems, Las Vegas, NV, August, 2000, pp. 426-431.

[18] Ahmed Salah and Kenlili and Tarek F. Gharib and Abdul Fattah Mashat. A Multi-core Tool for Searching Protein Structural Similarities. International Journal of Computer Applications (50)12:34-36. 2012.

[19] Mohamed, H. Abouelhoda, M. Parallel suffix sorting based on bucket pointer refinement. Biomedical Engineering Conference (CIBEC), pg. 98 - 102 2010 5th Cairo International


Scientific Visualization

DVO Model Using Mask for Data Distribution on Tiled Display

Laura P. Ramírez Rivera, Sergio V. Chapa Vergara, Amílcar Meneses Viveros
Department of Computer Science
Cinvestav-IPN
lramirez@computacion.cs.cinvestav.mx

Abstract

Cluster-based tiled display walls CBTDs provide advantages in the scientific visualization area. CBTDs became in an important solution to understand big amount of data. In spite of this importance CBTDs drivers don´t have a standard to work with the different necessities that the CBTDs drivers involve. In this paper we propose a simple data distributions strategy based in the DVO model.

Keywords: Tiled display wall, scientific visualization, data distribution.

1. Introduction

This paper is a contribution to the growing literature on design in visualization data on CBTDs. It proposes a way in which general functions to make a handler for CBTDs can be found, and how these functions can be used to guide visualization design on CBTDs environment. The paper elaborates a theme that we began in [1].

Because CBTDs provides high resolution and large scale, tiled display became in a popular solution for scientific visualization. The advantages of the use of the scientific visualization of CBTDs, such as significant productivity benefits as well as to engage in more complex multitasking behavior to name a few was shown in [2][3][4].

The CBTDs demonstrated benefit to scale the size of the tiled display with low cost. CBTDs approaches work in two main ways: (1) Parallel renders (PR) and (2) Shared memory (SM). PR was explained in [5], and was implemented using the OpenGL library. That was extended to WireGL library adding MPI in some of the original functions [6]. The project grows up to Chromium framework [7] adding data distribution. When the approach adds network protocol as UDP and other functionalities, CGLX [8] and Equalizer [9] were created. This time line was implemented thinking in 3D models made with big amount of data. The main idea is the basic primitive distribution model. With this distribution each node can works at the same time making the render of part of the 3D model. In this case the display is based on context. When a context starts cannot be changed at runtime. This means that a display area is assigned only for one application and cannot be changed at runtime.

SM lets the user move the display

application over the display area at runtime. RFB [10] [11] is a protocol used to share the graphic card memory. With the adding of others protocol VNC [12] was created. VNC is used in the actual remote desktops. SAGE [13] was created to the tiled display handler using optical networks as OptIPuter [14] and big memory pools as LambdaRAM [15].

The main idea with the shared memory is getting the information on the memory graphic card to display it on the tiled display (whatever display in the general case). When we use this approach with big scale the package to send is big also.

Both approaches PR and SM has advantages and disadvantages to solve. The advantages using parallel render are mainly the use of data distribution in the visualization generation. The disadvantage in PR is that work only one visualization each time in a definite display. In the SM approaches the network receives all the work passing the image previously generated. The advantage is mainly showing the visualization of many applications at the same time.

The advantage of the PR is the disadvantage of the SM. The join of the advantages of both approaches not is possible, but we can get an abstraction of the ideas to define general functions necessaries to get a good approach.

In this paper, we extended a proposal for the data distribution of the DVO model CBTDs driver. The data distribution is one of the general functions defined in the DVO model; which are data distribution, display distribution, event handler and synchronization. The proposed strategy is an algorithm to crop images using a mask.

2. Tiled Display features

A tiled display is a group of displays that deploy a common visual output. CBTDs can provide cost-effective and scalable displays with high resolution and large display area software to drive them needs to scale too if arbitrarily large displays are to be built [21].

We defined our architecture as follows. (Figure 1).

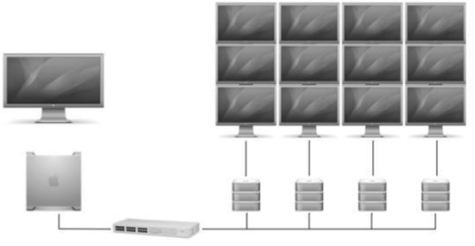


Figure 1. CinvesWall architecture

The logic model that we propose using is the client-server, because using a client for the data distribution can avoid a lot of work for a node, like in master-slave. On the client side is the container, the Input from the user and the data distribution. On the server side is the copy of the container, the data reception and the copy control of the visual objects. The server is the responsibility of the displays deploy.

3. GUI on CBTDs environment

The applications worked on tiled display applications mainly are defined using the client-server model. The architecture used is a cluster-based and is divided in the next components: (1) visualization server, which works providing the control to the user of this machine the

tiled display appearance, is controlled. (2) Visualization cluster, which are the machines that display into the screens, works together to provide the impression to work on a single display. (3) Tiled display, which are the screen used to display in high resolution and big scale. (4) Server display, which shows the tiled display GUI, it is the visualization server display.

The figure 2 shown how each component works together, but the important result is observed only in the tiled display component. The others could be hiding from the final user. The dimension of the tool provides a wide perspective of some visualization and many users could be in front of the tiled display. By instance when a big image is shown onto the tiled display, the discussion over the image features can be more important than the manipulation over the image.

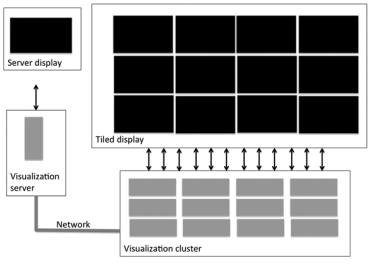


Figure 2. CBTDs architecture

It is possible to require more than one visualization each time to make comparatives and intersections among the images. When the user open and position the images in the position required the group of persons can accept to review the images or ask about changes necessaries in those images. But again when the discussion begins and some knowledge has been founded, changes in the images cannot be required.

Visualizations that require a record time as video only need the initial configuration as initial position or change of size. There could be requiring pause, stop and play the video. To start a discussion about the results offer it. In this case it possibly suggests a remote control (or something like this) obtained from many kinds of machines to use direct control from the visualization server.

The 3D models require a closer interaction with the user, when a model renders some transformations is required at runtime. Rotation, translation and scale are the principal challengers to front. Also the performance to build the model is a challenge. Strategies as parallel render are used to increase the performance.

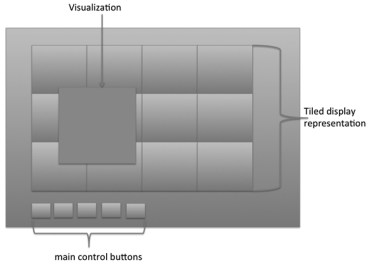


Figure 3. Basic control GUI

The figure 3 shown the general aspects found in a CBTDs GUI on the visualization server. The parts that compose the GUI are: (1) tiled display representation; this representation shows the visualizations that are displayed in large scale of the CBTDs. (2) main control buttons; those buttons let the control over the visualizations, let change size, position or apply some filter depending of the kind of visualization, by instance a video needs stop,

pause or play, also can require some change of position or size. (3) Visualization; shows the scaled representation if the final visualization, some occasion lets the direct manipulation using mouse pointer to change the position of size on the tiled display.

One of the buttons that cannot be changed is the open button, it is used independently the final application or visualization worked in the CBTDs. The next sections shown how then GUI for each main application is constructed depending of the final visualization, some examples are mentioned.

4. Image Viewer on CBTDs

An Image Viewer works displaying images with different formats and in high resolution of the tiled display screens. This kind of application has a flow of tasks to work on the tiled display, there are the next: (1) open image, (2) display image, and (3) make some change in the image (mainly change the size).

The process that occurs inside the application are the data distribution and communication among the machines that control the screens. For instance when an image is open in the main node the user sees the image on the tiled display, without knowing that this has a process such this: (1) the image was open, (2) the data about this image were distributed in all displays nodes, (3) the display node needs display the data, and so on, for the other functionalities.

The goal in an image viewer working on a tiled display is show whatever image obtained from some device, with high resolution and large scale. This display has to be done with good performance to display the full image in all screens at the same time. It is notable also that the change of the original format is made to achieve a good manipulation of the image.

The visualization server has to handler the image as change the position in the display, and opens a new image if the user wants. It is possible add features to this application as a filter or other transformations adding these functionalities in the control buttons. An additional interface used in some viewer is a chat window with people that can send their information to be displayed on the tiled display. By instance CinvesView is an image viewer working on the CinvesWall. The CinvesView GUI is shown in figure 4.

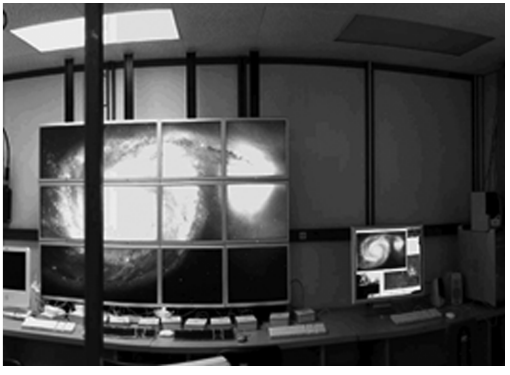


Figure 4. CinvesView GUI

5. DVO model

The DVO model is a contribution to the growing literature on design in visualization data on tiled display. It proposes a way in which general functions to make a handler for tiled display can be found, and how these functions can be used to guide visualization design on

tiled display environment. The paper elaborates a theme that we began in [1].

The CBTDs demonstrated benefit to scale the size of the tiled display with low cost. CBTDs approaches work in two main ways: (1) Parallel renders (PR) and (2) Shared memory (SM).

Both PR and SM approaches have been used to develop libraries, toolkits, frame- works and middleware to build CBTDs driver. The DVO model takes an abstraction of the principal components of this driver. In the figure 5 those components were defined as layers. Top-level goals shown the skills required for the use of the tiled display, by instance when we are working with images, we need to provide whole interaction with them (display, applies filters, recognize different input formats, etc.).

Restrictions and priorities are defined by the logic model used and features defined in the preliminary design of the CBTDs build. General functions are the most important layer to considerate. These functions let to get a better performance independently of the other layers.

The strategies necessaries to work with each one are primordial. Data distribution and display distribution let the use of the parallel power provides for the visualization cluster. The strategy used in the event handler can avoid the network resources waste.

The strategy to synchronization can help to increase the performance, if it is enough good. The physical functions are basically the CBTDs hardware; in this case we cannot increase the performance in this layer. We need to maintain independence among the other layers.

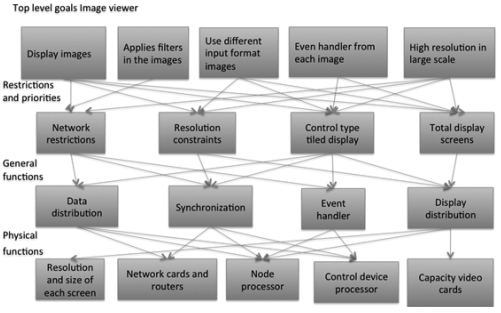


Figure 5. Hierarchy abstraction

DVO model is an abstraction of ideas to make a CBTDs driver. In the following, we will focus on the basic model aspects of the DVO. Starting with the logic model and a state diagram followed by the client-server model employed. One of the main DVO contributions, the hierarchy abstraction, which shows the general functions, which describes the main aspects to solve in this kind of handler, is then introduced in detail.

DVO (Distributed Visual Object) provides a model to facilitate the development of distributed applications on CBTDs. Indeed, this approach uses a concept called distributed visual object. This concept involves the basic components in a CBTDs. Visualization server is which has the original data to be displayed on the CBTDs. The Visualization server can be taken as the concept to the source visualization (databases, image source, etc.). Visualization cluster in some cases is taken for many clusters, which works each one as cluster-render, cluster-distribution, etc.

A distributed visual object (DVO) works as a composition of a visual object (VO) and a distributed object (DO). VO is the set of objects that can display whatever visualization on a rectangular area and handler events in it as well. DO is the set of objects with the capacity to make remote method invocations from different address space.

The previous concepts are the core of the model DVO. They need some resources to work as communication, data distribution, display distribution, event handler, and synchronization strategies. Those are the general functions required to build the handler of the tiled display. In the next section, we show a perspective using a state diagram to identify the behavior of the tiled display handler.

6. State diagram

We can define a state diagram of the behavior of a tiled display handler as in figure 5. There are 6 states, they are (1) initial state - sa (2) expected state - sb (3) consumer state -sc (4) display object state - sd (5) action state - se (6) search state -sf. The events can change the state of the system. The system recognizes 7 events; they are (1) starting event - e0 (2) opening event - e1 (3) deploying event - e2 (4) receiving event - e3 (5) handling event - e4 (6) recognizing event - e5 (7) messaging event - e6.

The initial state is sa. In this state the system is off and need to start event e0 to begin the process. The expected state sb begin when the system is ready to open the first image. At this state the user interface was deployed into the visualization server and the tiled display. When the open event e1 occurs, the system will pass for the consumer state sc. Then the deploy event e2 is generated automatically to get the image and send it to the visualization cluster and pass the image to the display object state sd. The receive event e3 is generated to get the position into the screen and pass to the expected state sb. When the system is in the expected state sb and if some image was displayed before, the system can receive a user event e4. This event changes the system and put it in the action state se. In this state the event has to be recognized and send messages with the information to the nodes. Next occurs e5, and then the system pass to the search state sf to locate the object that receives the event and send the information to the nodes, and then pass to the display state (to refresh the image) sd, and the event e3 is generated and finish in the expected state sb.

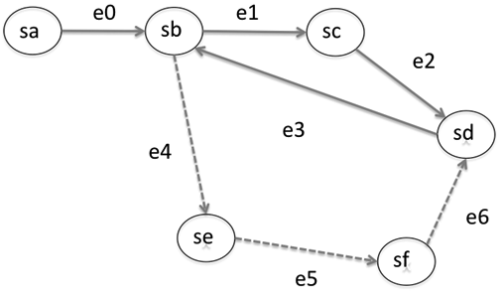


Figure 6. States diagram Cinveswall

When a specific application was select by instance an image viewer, it can be define the next behavior. It start with two necessary tasks in the image viewer case; both open images and receive user events. Each one can explain with the next sequence.

The system is in the initial state, sa if the hardware is ready to begin. The expected state sb requires that the visualization server has to be connected with the visualization cluster using necessary protocols. It is also needed to make the principal interface; the scaled interface and the arrays include the new objects. The consumer state sc requires sending control and data messages to the visualization cluster, to store the new object and to generate a new event to do the display. This sequence can increase the performance using less bandwidth with the data distribution (display data in the image case) general function. The display state sd needs to know the position, size and the kind of image in order to take the data to generate the image onto the screen. In addition, this information has been sent to the visualization cluster.

The action state se needs to make a search of the object in the storage to know which object receives the event and locate it into the visualization cluster of the same object and to replicate the action there. It is also needed to know the kind of event that occurs using the operating system event queue, and to send the information to the visualization cluster. There is where the handler event general function has to be used. The strategy to control the event can be reduced when the object schedule is well constructed. The search state sf waits until the visualization server locates the object in the store of visualization cluster and uses the received information to replicate the action of the visualization server, and then passes to the display state at the same time.

There is a cycle in the system which starts in the state sb. This cycle lets us put many internal tasks automatically. These states have to be changed with the less delay as be possible. The general functions can be used in some states indeed can increase the performance.

The general functions strategies have to be evaluated for each case (hardware architecture). When the strategies proved their benefit they can be used in all applications implemented on the CBTDs. In the next section we will show a strategy used in the data distribution on CBTDS Image Viewer application.

7. Algorithm using mask for data distribution on CBTDs

The DVO model structures previously mention requires strategies for each layer that solves the general functions for the CBTDS drivers. One of them is the data distribution general function that works in the control layer. Now we propose a strategy of data distribution for high-resolution images used in image viewer application on the CBTDs environments.

When you need to divide a high-resolution and large image, for distribution in the deployment consists of n number of screens, each screen has a resolution, taking the sum of the screen resolutions such as resolution total the CBTDS.

We propose an algorithm which allows the distribution of the image matrix screen suitable for each intersection by a mask. Each mask is assigned by screen position and the intersection with the array of colors of the original image gets the drop on each display. Each screen intersecting with a mask. Each mask is assigned by screen position and the intersection with the

array of colors of the original image gets the drop on each display.

Initially it must perform a matching mask array size and the size of the original image. Positioning masks are assigned from the configuration on each screen. The trimming process is performed on each node to distribute the work.

Algorithm 1 Image distribution

Require: Matrix mask with the full resolution of the tiled display

Require: Original image with the size of matrix mask
Ensure: x= high of the matrix mask
Ensure: y= width of the matrix mask
Ensure: MR=result matrix
Ensure: MM=mask matrix
Ensure: MO=original matrix
Ensure: MF=final matrix

```
for i=1 to x do
        for j=1 to y do
                MR[x][y]= MM[x][y] and MO[x][y]
        end for
end for
MF=BorraZeros(MR)
```

The MF is the final matrix, which will be displayed for each node. The size of the matrix MM is defined by the coordinates (x, y) obtained from the full resolution video wall RTP. After pairing the size of the original image matrix MO MM becomes equal to (see Figure 6).

When needed the use of the complete video wall is necessary to scale the size of the original image. The fraction process can be done on the ~~client or on each server. In this case we propose~~ to divide within each server to distribute the work.

Depending on the format of the image values can be divided by color, regardless of that in this case what matters is the position, and the mask is to be used for each color.

When performing the process of BorraZeros can find zeros within the image, so that an aspect percentage added to remove zeros, wherein at least one pixel must be different from zero, otherwise a warning should be sent.

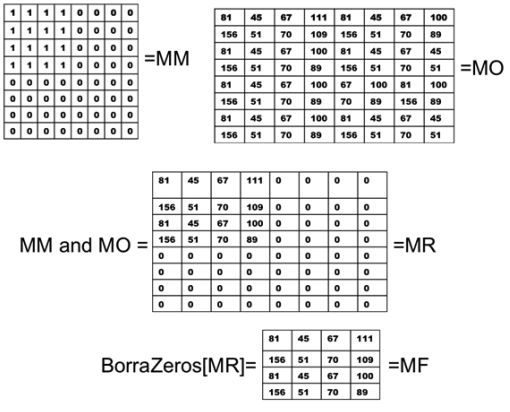


Figure 7. Matrix diagram cropping an image.

8. Data distribution performance obtained in Image Viewer application on the CinvesWall.

This approach shows the performance of a basic algorithm to crop an image. This

algorithm is used to test the image viewer performance compared with the strategy of sending the whole image. First the algorithm 1 has to find the crop points depending of both the tiles resolution and matrix position of the tiles. In the figure 8 the input requirement algorithm is shown.

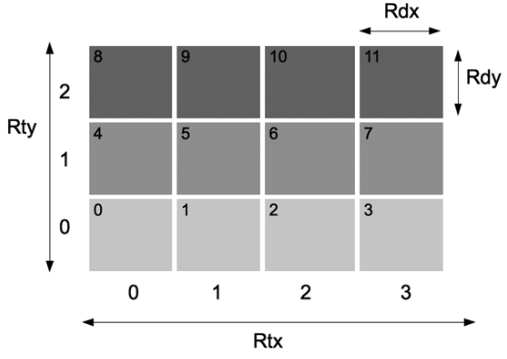


Fig. 8 Input requirments.

The obtained array has the initial position of each tile in the global coordinates of the whole tiled display resolution. When the crop initial points were obtained the second algorithm pass a mask with the values necessaries for each tile. The result of the second algorithm is part of the original image necessary to be displayed in each tile. When the initial points were founded the mask to crop the original image has to be made. The original image is scaled in the tiled display resolution to crop each part of each tile. The image result has the size of the tile. The mask is used because a pixel matrix represents the image.

The figure 9 shows the comparative chart obtained in the image viewer execution. The red line indicates the performance obtained when the image partition. This time includes the crop time and the transfer time. The blue line shows the behavior obtain when the original image is sent. This is only the transfer time. With the same implementation and the same image the constant behavior is obtained with the partition (Data distribution strategy).

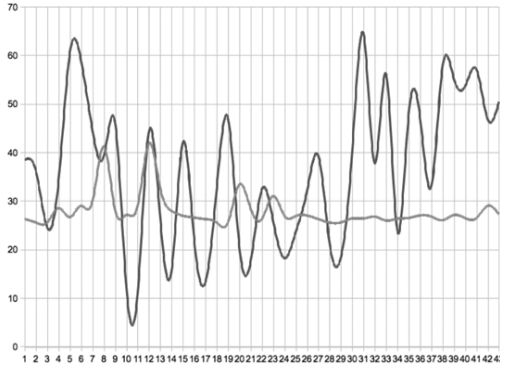


Fig. 9. Image Resolution18001x11438 (seconds x test number)

With the red line representing the data distribution strategy and the blue line the send of whole image. This chart shows that the use of data distribution can help in the performance of this kind of applications. Also we can see that select the correct algorithm for data distribution makes a visible difference. The architecture used in the CBTDS represents the main restriction to select the data distribution strategy. In the specific case of CinvesWall the data distribution works better that the whole data.

9. Conclusions

This document shows some aspects of the model specification DVO. This model allows the creation of a distributed window manager. One of the most important characteristics in the model definition is the data distribution. It shows an algorithm based on mask position to split an image, the division is performed on each node to distribute the work.

10. References

[1] L. Ramirez. , A. Meneses, S. Chapa ,"DVO: Model for Make a Handler for a Tiled Display ", Proceedings of the World Congress on Engineering 2012, WCE 2012, July 4-6, 2012, London, U.K.

[2] G. Kurtenbach and G. Fitzmaurice, "Applications of large displays", Guest Editors Introduction, Published by the IEEE Computer Society, July/August 2005.

[3] Leigh, J., et al., The global Lambda Visualization Facility: An International Ultra-High-Definition Wide Area, Visualization Collaboratory Future Generation Computer Systems, 2006.

[4] Funkhouse, T.; "Large-format displays", Kai Li, Computer Graphics and Applications, IEEE, pp 20 - 21, 2000.

[5] S. Molnar and M. Cox and D. Ellsworth et al., A sorting classification of parallel rendering, IEEE CGA, 1994.

[6] Wylie B. and Pavlakos C. and Lewis V. and Moreland K., Scalable rendering on PC clusters, Computer Graphics and Applications, IEEE , vol.21, no.4, pp.62-69, Jul/Aug, 2001.

[7] Greg Humphreys and Mike Houston and Ren Ng and Randall Frank et al., Chromium: a stream-processing framework for interactive rendering on clusters, 2002 Proceedings of the 29th annual conference on Computer graphics and interactive techniques, 2002.

[8] Kai-Uwe Doerr, Falko Kuester, CGLX: A Scalable, High-performance Visualization Framework for Networked Display Environments, IEEE Transactions on Visualization and Computer Graphics, vol. 99, no. PrePrints, 2010.

[9] StefanEilemann,MaximMakhinya,andRenatoPajarola,Member, IEEE Computer Society, Equalizer: A Scalable Parallel Rendering Framework,IEEE Transactions on visualization and computer graphics, Vol. 15, No. 3, may/jun 2009.

[10] Tristan Richardson, RFB protocol, Olivetti Research Ltd AT and T Labs Cambridge 2010.

[11] Joel McCormack and Bob McNamara, A smart frame buffer, WRL Research Report 93/1.

[12] Richardson T. and Stafford-Fraser Q. and Wood K.R. and Hopper A, Virtual Network Computing, Internet Computing, IEEE , vol.2, no.1, pp.33-38, Jan/Feb 1998.

[13] Jeong, B., Leigh, J., Johnson, A., Renambot, L., Brown, M., Jagodic, R., Nam, S., Hur,H. Ultrascale Collaborative Visualization Using a Display-Rich Global Cyberinfrastructure IEEE Computer Graphics and Applications, Vol. 30, No. 3, pp. 71-83, May/June 2010.

[14] Larry L. Smarr and Andrew A. Chien and Tom DeFanti and Jason Leigh and Philip M. Papadopoulos, The OptIPuter, Communications of the ACM Volume 46 Issue 11, November 2003.

[15] Venkatram Vishwanath, LambdaRAM a high-performance, multi-dimensional, distributed cache over ultra-high speed networks, PHD thesis B.E. University of Illinois at Chicago, 2009.

Generating Large Varied Animated Crowds in the GPU

Isaac Rudomin[1], Benjamín Hernández[1],[2], Oriam deGyves[3],
Leonel Toledo[3], Ivan Rivalcoba[3], Sergio Ruiz[2]
[1]*Barcelona Supercomputing Center, Spain*
[2]*Tecnológico de Monterrey, Campus Ciudad de México, México*
[3]*Tecnológico de Monterrey, Campus Estado de México, México*
isaac.rudomin@bsc.es, {benjamin.hernandez@bsc.es, hbenjamin@itesm.mx}

Abstract

We discuss several steps in the process for simulating and visualizing large and varied crowds, in real time, for consumer-level computers and graphic cards (GPUs). Animating varied crowds using a diversity of models and animations (assets) is complex and costly. One needs models that are expensive if bought, take a long time to model, and consume too much memory and computing resources. We discuss methods for simulating, generating, animating and rendering crowds of varied aspect and a diversity of behaviors. Efficient simulations run in low cost systems because we use the power of modern programmable GPUs. One can apply similar technology using GPU clusters and HPC for large scale problems. Such systems scale up almost linearly by using multiple nodes. One must combine parallel simulation and parallel rendering in the cluster with interaction and final render in lighter clients. However as the latest developments such as the new family of mobile multicore chipsets and GPU based cloud gaming platforms the pieces are almost there for this kind of architecture to work.

Keywords: Simulation; Real-Time Crowds, Rendering & Animation.

1. Introduction

The development of graphical models and the incorporation of intelligent behavior has made it possible to create virtual environments with many intelligent characters interacting. The use of these types of models can be seen in many applications. One example is videogames – where one can design more challenging games with many characters of different physical appearance and each with its own behavior. Another area where these models can be used are in the design of strategies for people entering or leaving an environment or structure – a stadium, an auditorium, an entertainment center, even in the case of emergency. Also, for the simulation of vehicular traffic –for building new roads, taking into account the behavior of drivers. Similar methods can be applied to other social or health applications involving

agents (epidemics, infection) and for financial agent based applications.
The problem is that crowd simulations need a scalable multi agent system architecture that simultaneously supports the simulation of hundreds of thousands (or millions) of complex autonomous agents and the rendering of all those agents as diverse, animated characters, even in crowded scenes, yet achieving good frame rates.

Today there is no simple process for simulating and visualizing large and varied crowds in real time for consumer-level computers. Animating varied crowds using a diversity of models and animations (assets) is complex and costly. One would require models that are expensive if bought, take a long time to model, and anyway consume way too much in memory and computing resources. As solutions to these problems, we have developed methods for simulating and animating crowds of varied aspect and a diversity of behaviors. They can be used for simulations for PCs with consumer level graphic cards (GPUs). Efficient simulations run in low cost systems because we use the power of modern programmable GPUs.

One can apply similar technology using GPU clusters and HPC for large scale problems. Such systems scale up almost linearly by using multiple GPUs. In the following we will review our methods for crowd simulation, level of detail rendering, generation and animation of varied character families, and what we are doing in the way of parallelizing for larger systems.

2. Simulation and collision avoidance

A key component in every crowd simulation is the synthesis of behavior; for a crowd simulation will be accurate in the measure it is able to faithfully emulate both individual and group human behavior. Collision avoidance refers to the anticipated movements with which an agent avoids colliding with other agents present in their environment.
There is extensive research focusing on collision avoidance behaviors for virtual agents. The main methods are:

a. "Flocking", by Craig Reynods, [1], uses three rules: separation alignment and cohesion to steer the agents. This can be done using forces for each rule and summing the effects.
b. "Social forces", by Dirk Helbing, [2], [3], which is very similar in that it uses forces, including forces for avoiding other pedestrians, others for avoiding walls, etc.
c. Reciprocal Velocity Obstacles [4], applies ideas from robotics to crowd simulations: it finds velocities that will cause collisions and avoids them. Planning happens in velocity space.

All these methods require data from the nearest neighbors of each agent. Exhaustive proximity queries can be prohibitive as the number of agents increases, since the naïve methods are O(n2). Researchers [5], [6] have explored several techniques and data structures to lower the complexity of proximity queries, that effectively reduce this to O(n log n).

2.1. Encoding neighbor information on world maps

Another way to do this, however, is to use a world-space map to encode the information in the environment rather than the agents and thereby reduce the problem of determining neighbor contributions to O(n). Some methods [7], [8] encode the environment with information guiding the agents, but then treat these agents as simple particles following the forces encoded in the environment.

We explored a more general method in [9]. Here the agents encode (paint) information in world space images representing the environment (as can be seen in figure 1). The agents, however, use this information using their own goals and perceptions since behaviors are specified as state machines (also stored as images). This, and world and agent space images are used to determine collision avoidance and other steering behavior. In particular, attraction and repulsion forces in figure 1 are added, to achieve behavior similar to that achieved by Reynolds and Helbing.

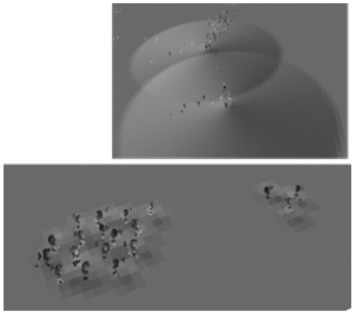


Figure 1. Using world space maps to encode forces that attract or repel agents from each other

Other images in agent space codify agent state. In a simplified version (figures 2 and 3) we use a Label map as the single world space map. We codify state and position in an agent map. Then, given state and position, and by consulting color in the appropriate position in the label map, we determine new state and positions from the FSM map.

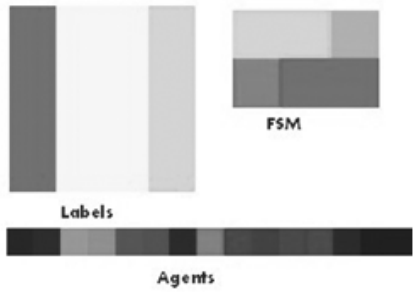


Figure 2. In a simplified version of the method, three textures are used. A FSM texture, a world texture specifying labels and an agent texture where state and position are encoded.

In general, we can use several auxiliary images in world space to codify obstacles, heights, gradients. This became rather cumbersome, so in latter work, and for easier authoring, maps and finite state machines were codified in XML and translated to GLSL shaders. This allowed us to generate more complex scenarios.

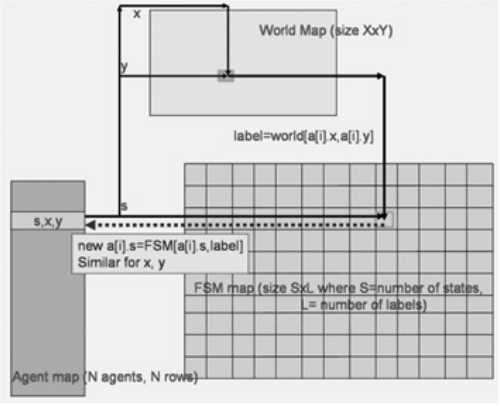


Figure 3. Simulation Process: consists of consulting the three textures from figure 1.

2.2 VL Path Planning

Realistic crowd simulations face several problems in order to achieve more accurate results; one problem is collision avoidance. We have studied a parallel technique, using the graphics hardware, to avoid all collisions between agents by efficiently finding the nearest neighbors in a crowd simulation and calculating the response.

It is basically an image based technique similar to the one used in the previous section, but here, instead of painting and adding different forces, every agent paints an area of a given radius that codifies the distance to the agent. When these areas overlap the distance to the closest agent is kept. As a result this texture is a truncated Voronoi Diagram (figure 4 top left) that other agents can read to move through the environment (figure 4 top right).

A truncated Voronoi Diagram can be described as: given a set of S seed points, divide a plane into |S| tiles such that all points 'p' inside a tile are closer to a particular seed than to any other seed and are within a predefined radius, called tile radius 'r', per seed, using the seed coordinates as a center. Like other similar techniques that use painting of influence areas, this technique can achieve simulations with thousands of agents in real-time (figure 4 bottom). But in contrast to methods that codify forces, our method in the previous section, agents do not get stuck.

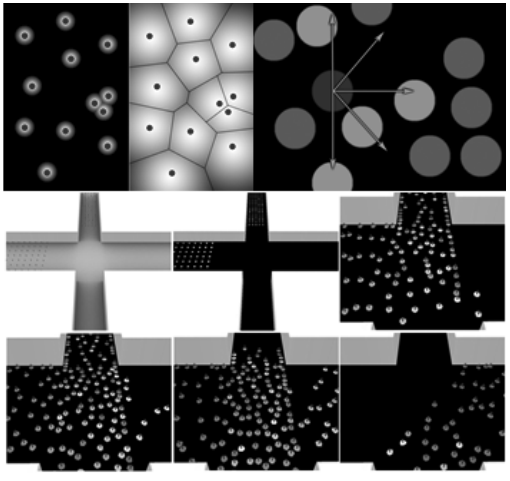


Figure 4. Voronoi-Like method for collision avoidance (top) and Voronoi + RVO example (bottom)

To detect neighbors using the truncated Voronoi diagrams, we take samples in the local vicinity of the agents. To sample its neighborhood, an agent 'a' performs a ray

marching technique over the diagram to sense the viewing area in the direction of its movement. Figure 4 (top right) shows a graphic representation of the ray marching process. The circles represent the agents in the simulation; the purple agent is the one currently looking for its nearest neighbors and the green agents are the nearest neighbors. The red agents are not considered by the purple agent because are currently out of its viewing area.

This method is capable of simulating thousands of autonomous agents at interactive frame rates, while performing accurate collision avoidance. It is important to note that the proximity queries are performed in the graphics hardware, which allows us to implement the collision avoidance also in the graphics card.

Following the previous idea, we have developed other two techniques for proximity queries that are suitable for simulating thousands of agents in real time. These techniques are gather-based and scatter-based approaches.

For the scatter technique, we follow the idea that an agent should be able to find its closest neighbors with only one texture fetch. To accomplish this, each agent paints in an environment map an area where it is visible. Each pixel in the environment map holds a list with the Ids of neighbor agents. We use a Layered frame buffer (LFB) to generate the nearest neighbor lists and render the environment map for the scatter technique (figure 5 top left). LFBs are commonly used for Order Independent Transparency (OIT) and to our knowledge, have not been used for proximity queries or behaviors.

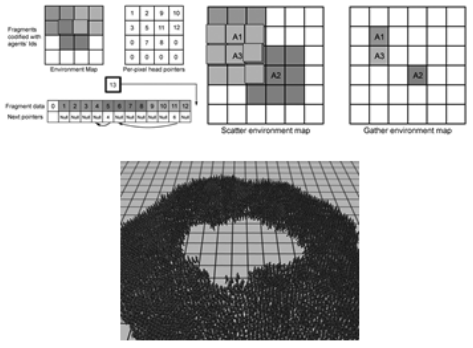


Figure 5. Top left. Layered frame buffer used to generate a list of nearest neighbors. Top right. Comparison between the environment maps generated for scatter and gather. Bottom. Circle-4096 test. Agents initially in circle and have to get to their diametrically opposite position.

Contrary to scatter, in the gather technique each agent only writes one pixel in the environment map. To find their neighbors, every agent scans over an area in the environment map. The purpose of scanning an area is for each agent to generate its own neighbor list. As in the previous technique, every agent paints in a texture its Id, but only on a single texture element. This eliminates the need to create a LFB. Then, every agent reads an area searching for its nearest neighbors. Figure 5 top right shows the main difference between the environment maps created. As the first method presented in this section, these two techniques are capable of simulating thousands of autonomous agents at interactive frame rates (figure 5 bottom).

2.3 Data Driven Simulation

A recent trend is to try to validate models with real world observations, rather than by reproducing phenomena that seem to be right. Even Helbing et al, [10] have recently stated that force models, while successful are not consistent with empirical observations and are hard to calibrate. They suggest a cognitive science approach based on behavioral heuristics. They develop a new model where guided by visual information, namely the distance of obstructions in candidate lines of sight, pedestrians apply two simple cognitive procedures to adapt their walking speeds and directions. Others have done this perceptual or synthetic vision based steering [11], however, in most of these approaches each agent needs its own memory, and so this methods fail to scale up to several hundreds of thousands of agents and do it efficiently. Our work in [9] using the world space maps performs efficient collision avoidance operations in the spirit of synthetic vision even for millions of agents, if instead of painting round areas, we paint areas with the shape of the appropriate perception cones. However, we have yet to validate this model with actual real-world behavior. Data driven simulation is fundamental for validating the behavior models of the agents.

We have already coupled our simulation module with the vision module. The image involving the nucleus is first transformed from RGB color space to grayscale, then is equalized and a multi-resolution histogram of SLBP [27] (semantic local binary patterns) is extracted to form a feature vector, then this vector is fed to a trained support vector machine (figure 6 top). The result is to label each image as having a head or not having a head (figure 6 center). Each detected head is tracked over every frame by a pyramidal version of the Lucas-Kanade tracker. Using these points and the simulation stage we combine steering models with actual behavior (figure 6 bottom).

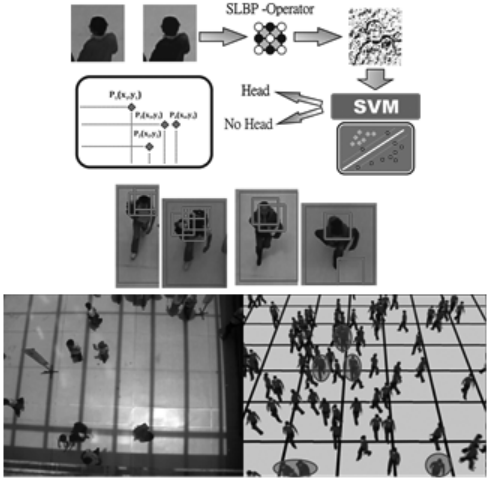


Figure 6. Vision module combined with simulation (avatars in red, simulated agents in black).

3 Navigation

Closely related to collision avoidance, navigation in a crowd simulation context refers to the ability of agents to effectively avoid collisions against objects within a scene while moving toward their goal. These obstacles may or may not move, but the criterion that classifies them as objects within a scene is the fact that obstacles do not avoid collisions: they do not

present reactive behavior.

There is a phenomenon occurring when a crowd is observed; at near distances individual behavior arises while at far distances group human behavior can be noticed. This assumption may not imply that two types of simulations are required but describes human behavior when observed from different perspectives, as an accurate simulation would. Referring to Zhou et al. [12], this phenomenon can be interpreted in two levels:

a. Micro level: When experienced from a close distance, a crowd appears as a small-sized crowd regardless of its actual size, revealing fine entity traits.
b. Macro level: When viewed as a whole, from far distance, a crowd presents coarse, fluid-like traits.

Nevertheless, in practice, there are crowd navigation techniques which perform well at either micro or macro level. For example, classic path finding algorithms such as A* or Dijkstra's [13] has been used extensively at micro scale level when the number of agents is low (in a number that a CPU can handle), and obstacles do not move. For a macro scale level simulation, where higher agent numbers and the option of moving obstacles, several techniques model and solve the agent navigation problem, in the form of grid partitioning, formation and navigable areas or using video sequences as input.

Grid partitioning methods consist on dividing the navigable space into cells where according to different rules or mathematical models, agents are able to find their path to a given goal even if objects are present in the environment. An approach to grid partitioning is the use of cellular automata [14, 15, 16]. Cellular automata solve the collision avoidance and navigation problems with one algorithm, but it presents the lack of separation and control for individuals as they must follow the majority flow and direction towards an exit. A navigation method similar to cellular automata –in the sense of grid partitioning of the navigable space- is the one based on vector fields, such method is able to produce a character motion flow for one agent, which is responsive to user input [17].

Another option for navigation consists in defining navigable areas instead of a well-structured grid. Pettré et al. [18] opt for decomposing the navigable area into overlapping cylindrical cells with the aid of a Voronoi Diagram that will generate a Navigation Graph which provides path variety and batch processing for groups of pedestrians. It does not, however, support dynamic scenarios.

Video sequences also have been used as input to generate the motion of a virtual crowd [19]. It turns out that since video sequences capture reality it is expected that the agents' navigation is shaped by the physical characteristics of the captured scenario.

3.1 MDP for planning

As mentioned earlier, navigation techniques can simulate crowds at micro or macro level, our work in progress approach will unify these levels of abstraction thus agents will be able to take its own decisions which may affect the whole crowd, while group dynamics may affect

the individual. Our first observation is that an agent, while moving through an environment performs a sequential decision problem which has to solve to find its path from an origin to a destination following a set of additive rewards; this is usually called Markovian Decision Process (MDP). A layered based method which solves a semi MDP [20] may turn impractical since it requires different layers encoding extra information of the navigable space. In cases where obstacles or other kind of objects are dynamically introduced a new layer must be added which increments complexity to the whole solution. Another problem with the MDP formalism is that the state space grows exponentially with the number of domain variables, and its inference methods grow in the number of actions. Thus, in large problems, MDPs become impractical and inefficient.

In our case we have implemented a layer independent GPU based approach based on MDPs which MDP is run as a preprocess to calculate multiple free-of-collision trajectories that agents in a crowd will follow (figure 7 top). Then on run-time the MDP is adapted within a given radius, using a spatial kernel which encodes route alternatives inside the given radius, thus characters are able to avoid collisions against other agents or moving objects (figure 7 bottom).

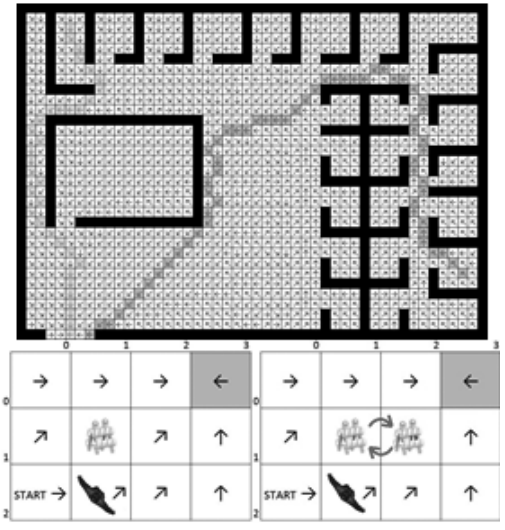


Figure 7. MDP for crowd navigation and collision avoidance

Table 1 shows performance results obtained from calculating MDP in GPU and CPU for the test scenario shown in figure 7 top. In the case of the CPU based MDP, it only used a single thread process and after 25 minutes the program calculated only 10 of 184 iterations needed to reach the optimal policy.

Table 1. GPU and CPU based MDP comparison.

System	Iterations	Optimal Policy
Nvidia Geforce GT445M	184	4.1s
Inter Core i7 1.87 GHz, single thread	10	1,500s After 10 iterations

3.2 Lattice-Boltzmann crowd flow

For extreme densities of agents, intersection computation becomes expensive or our adaptive MDP approach might have unnecessary adaptations since different agents might be in the same cell. At this macro simulation level, crowds mostly follow a general flow; only certain agents need to be simulated as such and most crowd movement can be simulated as fluids. We can use Lattice Boltzmann Method (LBM), which is a method for solving the Navier-Stokes equations using Lattice Gas Cellular Automata. It consists of discretizing the dominion into a set of connected sites (the lattice or mesh), with state variables defined at each site and update rules, based on local and neighbor information (collision and streaming). The generic LBM simulation algorithm follows the next steps:

a. Each particle travels with a discrete direction e. This is the discretization of velocity space.
b. At each time step, the particles move along their assigned directions toward the next lattice point: they are streamed.
c. If more than one of these particles arrives simultaneously at the same lattice point, a collision rule is applied, redistributing the particles such that the conservation laws (Navier-Stokes for mass and momentum) are satisfied.

We implemented LBM in CUDA as Single Component Single Phase (SCSP):

```
foreach time step do
   CalculateMacroscopicVariables()
   DetermineNewEquilibriumDF()
   Collide()
   ProcessBoundaries()
   Stream()
End
```

Readers can find a detailed explanation of the algorithm in [21]. Implementation results of the method for a Lattice of 1024x1024 cells the iteration time was 90.3 ms in a Nvidia Geforce GTX 560M GPU.

4 Level of Detail

Rendering many characters is a must for crowds. In the following we explore several methods for level of detail that allow rendering crowds with many characters.

4.1 Uniform crowd Impostor LOD

Impostor based LOD in the GPU and instancing allows real time rendering of large crowds of similar characters. In figure 8 you can see a texture with images the same character in several poses and different camera angles. This is the impostor texture. When the character is far enough from the camera, instead of displaying the character's geometry, we display the image of the character closest in pose and camera angle.

4.2 Varied crowd discrete LOD

For varied crowd visualization: using impostors is too expensive, thus we do discrete

LOD and View Frustum Culling in the GPU. In figure 9, this method is outlined. First, all necessary initializations are performed on the CPU. These include loading information stored on disk (e.g., animation frames and polygonal meshes) and information generated as a preprocess (e.g., character positions) or in runtime (e.g., camera parameter updates).

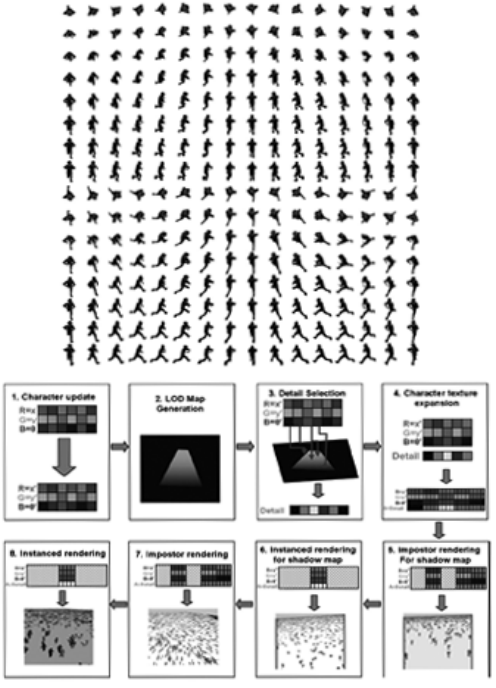


Figure 8. Impostor images and process.

This information is used on the GPU to calculate the characters' new positions, do view frustum culling, and assign a specific level of detail (LOD) for each character and for level of detail sorting and character rendering.

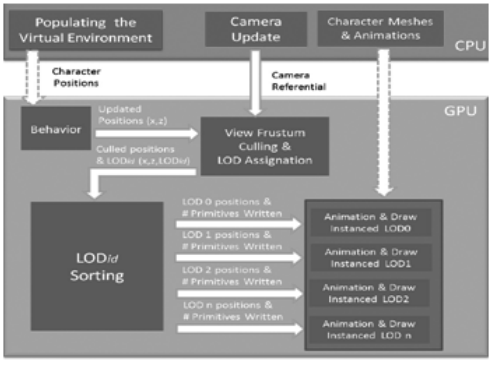


Figure 9. Discrete LOD in GPU.

A brief description of each stage is given next:

Populating the Virtual Environment and Behavior. In these stages we specify the initial positions of all the characters, how they will move through the virtual environment and how they will interact with each other. The result is a set of updated character positions.

View Frustum Culling and Level of Detail Assignation. In this stage we use the characters' positions to identify which characters will be culled. Additionally, we assign a proper LOD identifier to the characters' positions inside the view frustum according to their distance to the camera.

Level of Detail Sorting. The output of the View Frustum Culling and Level of Detail Assignation stage is a mixture of positions with different LODs. In this stage we sort each position according to its LOD identifier into appropriate buffers such that all the characters' positions in a buffer have the same level of detail.

Animation and Draw Instanced. In this stage we will use each sorted buffer to draw the appropriate LOD character mesh using instancing. Instancing allows us to translate the characters across the virtual environment and add visual and geometrical variety to the individuals that are part of the crowd.

4.3 Point based LOD

Applying the GPU for discrete LOD and view frustum culling calculation results in a very small time penalty (~ 0.39 ms) [22]. Nevertheless this approach is limited for the rendering step. Drawing the characters using polygons as their geometrical representation takes a considerable amount of resources in modern GPUs. It has been shown [23] that alternative rendering methods such as hierarchical point based methods (figure 10 top) improve rendering performance.

Our point-based LOD approach consists in using a point-based system combined with an hybrid hierarchical skeleton structure that allow us to create varied animated crowds, as well as reducing rendering costs. The system takes as input any given model vertex, normal and UV parameterization to generate a hierarchical skeleton structure based on octrees (figure 10 bottom). To create that skeleton, containing volumes are calculated for every limb in the model. This way data from individual parts of the model are stored and can be used for render and animation. For rendering, each limb's data is reduced to create a point sample with fewer points; this was done in four levels, the last two levels have a single point that represents the whole limb and a single point for the whole character, respectively.

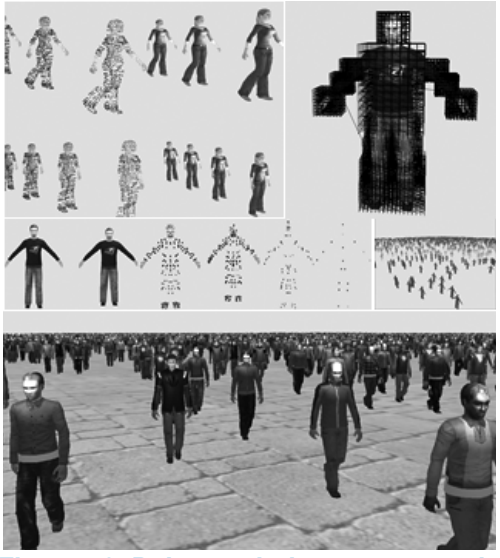


Figure 10. Point rendering vs geometric models (top left). Hierarchical skeleton structure based on octrees (top right). Different Levels of Detail and coding (middle). Scene rendered using complete system (bottom).

For animation the process is the same, as the animation of the joints is applied to vertices or points on the appropriate limbs at any level of the associated octree, making it possible to animate any given character. For lower levels of detail animations can be reduced by only the most important limbs, such as the whole legs or arms.

The system is divided into three stages. The first stage holds the highest level of detail: models are fully rendered using the complete geometry, as well as tessellation and displacement mappings for better rendering results. This is expensive and resource-consuming, so only very few models are

selected within the crowd to be rendered at the first stage.

The second stage is the result of using the octree structure: this process gives as a result different point samples for each model. In the last stage the world is divided in tiles and a quadtree is made with these tiles as leaves: agents in tiles or their quadtree parents that are far away from viewer are blended together to reduce the required rendering resources. Animation and collision avoidance computations can be skipped in these tiles.

5 Texture Based Characters

Current character modeling and animation techniques require a great amount of work and time. If we want to model and animate each character of a crowd made of hundreds of characters, the artist must design or scan each model, rig each character, specify the source of the animation for each character (motion capture, forward kinematics, inverse kinematics, video sequences, among others) and render each character. We can do better. We have developed texture based methods for modeling, rigging, skinning and animating varied crowds. These methods require the generation of a family of characters which has the same UV parameterization then a small set of textures is needed for rigging, skinning and animating a crowd made of thousands of characters.

5.1 Generation of Diversity

As crowds increase in complexity and size, more assets such as meshes and textures need to be designed and created, body templates, alleviates the need to create more assets. Our approach consist on generate crowds of characters visually and geometrically different using body-part templates which are combined together. These body-part templates or basis meshes represent a specific part of a body such as the head, torso, arms, or legs, these basis meshes and their textures were extracted from a character data set. Then by combining the basis meshes and their textures we generate families of novel characters according to specific features (sex, ethnicity, complexion or clothing) given by the user (figure 11 top).

Following Blanz' approach [24] where all facial models share the same parameterization space and are thus form a subspace of morphable models, our character dataset and the character family generated from them also have this feature. The advantages of having this feature are explained next and following subsections.

To improve the visual variety of our characters we use a manual generated texture called anatomy image which is a gray scale image were brighter areas represent high levels of fat and darker areas represent low levels, we are able to generate different body complexities from skinny to obese characters (figure 11 bottom). The first advantage of having the family in full correspondence is that we can apply this anatomy image in different characters indistinctly thus eliminating the possibility of having one for each character of the crowd.

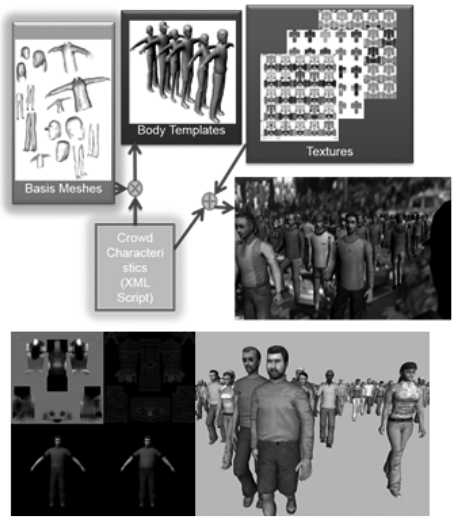


Figure 11. Generation of Diversity (GOD): templates, fat/muscle displacement maps, results.

5.2 Texture space Rigging

The animation of characters is usually done through the use of a skeleton, which is an articulated structure of segments and joints combined with information detailing how the surface geometry of the figure is bound to that structure. This is a time consuming process when animating a character since an experienced artist can spend several hours in this process thus conventional techniques can't be used to animate a crowd made of hundreds or thousands of different characters. Our alternative is to take advantage of the UV parameterization of the characters. If all of them have the same UV coordinates and rigging and skinning information is stored in textures which correspond to that UV parameterization, we can reuse these textures in all the characters.

Figure 12 (left) shows a texture used to calculate a skeleton of a character. Each joint area (dark zones) has a unique id representing a body joint, notice that this texture must be in correspondence with the character's texture parameterization (figure 12 right). To obtain the 3D pivot points for character's limb rotation, which connected will conform the skeleton, the computation of the centroid at each joint area is performed by selecting and averaging all the vertices which belong to a given joint area.

Figure 12. Left. Joint texture. Right. Joint texture and its correspondence with the character texture parameterization.

As mentioned earlier, texture based rigging can be transported from one model to the other if texture parameterization is the same in both cases, i.e. between characters of the same family (figure 13).

Figure 13. Rigging two characters with aligned parameterizations.

5.3 Texture space Skinning

The next step in character animation consists in specifying how the character's mesh will be attached to the skeleton, this process is called skinning. The idea behind this is to specify weight values used to modify the character mesh according to the skeleton pose with a reduced amount of artifacts that appears at joint sections. Thus, we encode these weights, which can be generated using authoring tools such as Maya, into a texture (figure 14 top).

Finally the rigging is completed with animation key frames. Key frames can be generated via motion capture, forward kinematics, inverse kinematics, video sequences, among others. Our current implementation uses forward kinematics (computation of the position and orientation of character's end effector as a function of its joint angles) to generate a pose. Each pose is a set of local transformation matrices that specify rotations on each articulation, while an animation clip is a set of poses.

Using texture based methods as described allow us to transfer animations straightforwardly between characters which share the same UV mapping (figure 14 bottom), thus material and computational resources are reduced.

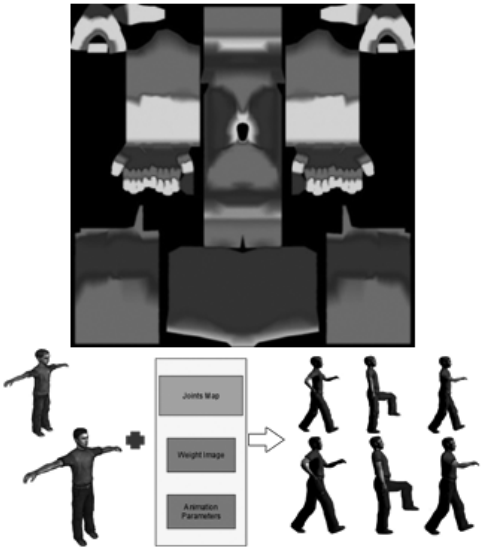


Figure 14. Blending maps for skinning. Skinning two characters with aligned parameterizations: matching poses allow animation transfer.

6 Conclusions, ongoing work

We have achieved interesting results in simulation and rendering of varied animated crowds. What we want to do is to continue development of what would be game engines specialized in simulating complex behavior such as that of crowds. What are we doing? improving graphics: LOD, GOD, animation, more complex models (clothing, hair) and interaction with physics. We want to use rendering system for other applications requiring drawing large numbers of units (particles, neurons, for example). We want to have more complex behavior of the "agents", including an increased cognitive level, messages, negotiations, while preserving

efficient execution in the GPU. Deriving behavior from data is important for real applications beyond gaming.

Then, moving this into HPC because we need to develop large systems using multiple GPUs, using parallelism for both real-time simulation and for real time rendering on clusters. Leaving the interaction and rendering of each point of view for lighter clients.

We have been doing GLSL/CUDA data parallelism one GPU/thread per agent on a single node. The idea is that one can apply similar methods on GPU clusters and HPC for larger scale problems. Such systems scale up almost linearly by using multiple nodes.

One could use MPI. Subdivide the world into 2D or 3D areas, and exchange data in the borders, between neighbors.

In [25] a better architecture is proposed: Subdivide into (2D/3D) areas, with interchange of buffers at borders, areas manage their own agents, while manager distributes areas to each worker to maintain load balance. Other approaches add separate managers for cameras [26].

For parallel rendering one would need to run OpenGL on each node and do partial renderings, to be combined to achieve final visualization results. There are several ways to do this and they each have pros and cons. However as the latest developments such as the new family of mobile multicore chipsets and GPU based cloud gaming platforms the pieces are almost there for this kind of architecture to work.

Acknowledgements

This work has been partially funded by: CONACyT-BSC postdoctoral fellowship; SNI-54067; “Agentes virtuales y Robóticos en Ambientes de Realidad Dual” Tecnológico de Monterrey research initiative. We would also like to thank Nvidia for hardware donations used in some of these algorithms

References

[1] C.W. Reynolds (1987), “Flocks, herds and schools: A distributed behavioral model”, In Proceedings SIGGRAPH ’87, pp. 25–34.

[2] D. Helbing, P. Molnar (1995), “Social force model for pedestrian dynamics”, Physical review E 51 (5), 4282

[3] D. Helbing, I. Farkas, T. Vicsek (2000), “Simulating dynamical features of escape panic”, Nature 407 (6803).

[4] J. Van Den Berg, M.C. Lin, D. Manocha (2008), “Reciprocal velocity obstacles for real-time multi-agent navigation.”, In IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, IEEE.

[5] Stephen J. Guy, Jatin Chhugani, Sean Curtis, Pradeep Dubey, Ming Lin, and Dinesh Manocha (2010), “ PLEdestrians: a least-effort approach to crowd simulation”, Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 119-128

[6] Thomas Jund, Pierre Kraemer, and David Cazier (2012), “A unified structure for crowd simulation”, Computer Animation and Virtual Worlds, 23(3-

4):311-320.

[7] F. Tecchia, C. Loscos, R. Conroy and Y. Chrysanthou (2001), "Agent Behaviour Simulator (ABS): A Platform for Urban Behaviour Development", ACM/EG Games Technology Conference.

[8] A. Treuille, S. Cooper, Z. Popović (2006), "Continuum crowds", SIGGRAPH '06 ACM SIGGRAPH 2006 Papers.

[9] E.Millan, B. Hernandez, I. Rudomin (2006), "Large crowds of autonomous animated characters using fragment shaders and level of detail", In ShaderX5: Advanced Rendering Techniques, Engel W., (Ed.). Charles River Media, 2006, pp. 501–510.

[10] M. Moussaïd, D. Helbing, and G. Theraulaz (2011), "How simple rules determine pedestrian behavior and crowd disasters", Proc Natl Acad Sci U S A. 108(17): 6884–6888.

[11] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, Stéphane Donikian (2010), "A synthetic-vision based steering approach for crowd simulation", Proceeding SIGGRAPH '10 ACM

[12] Zhou, S., Chen, D., Cai,W. Luo (2010). "Crowd modeling and simulation technologies". Transactions on Modeling and Computer Simulation (TOMACS) 20, 4.

[13] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C (2001). "Introduction to Algorithms", second ed. McGraw-Hill Book Company, 2001.

[14] Blue, V. J., Embrechts, M. J., and Adler, J. L (1997). "Cellular automata modeling of pedestrian movements". In Systems, Man, and Cybernetics, vol. 3, IEEE.

[15] Zhang, S., Li, M., Li, F., Liu, A., and Cai, D (2011). "A simulation model of pedestrian flow based on geographical cellular automata". In Geoinformatics, IEEE.

[16] Zhiquiang, K., Chong, Y., Li, T., and Jinyan, W. (2011) "Simulation of evacuation based on multi-agent and cellular automaton". In Mechatronic Science, Electric Engineering and Computer, IEEE.

[17] Bian, C., Chen, D., and Wang, S (2010). "Velocity field based modelling & simulation of crowd in confrontation operations". In International Conference on Parallel and Distributed Systems, IEEE.

[18] Pettré, J., Grillon, H., and Thalmann, D. (2008) Crowds of moving objects: Navigation planning and simulation. In SIGGRAPH, ACM.

[19] Ju, E., Choi, M. G., Park, M., Lee, J., Lee, K. H., and Takahashi, S. (2010) Morphable crowds. In Graphics, vol. 29, ACM.

[20] Banerjee, B., Abukmail, A., and Kraemer, L. (2008) "Advancing the layered approach to agent-based crowd simulation". In Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society.

[21] M. C. Sukop and D. T. Thorne (2007), Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers, 1st ed. Springer Publishing Company, Incorporated.

[22] Benjamín Hernández, Isaac Rudomín (2011), "A rendering Pipeline for Crowds", GPU Pro 2: Advanced Rendering Techniques. Engel W. (Editor),

AK Peters.

[23] Szymon Rusinkiewicz and Marc Levoy (2000). QSplat: a multiresolution point rendering system for large meshes. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00).

[24] Volker Blanz and Thomas Vetter (1999). "A morphable model for the synthesis of 3D faces". In Proceedings of the 26th annual conference on Computer graphics and interactive techniques.

[25] Vigueras, G., Lozano, M., Perez, C., Orduña, J.M. (2008) "A Scalable Architecture for Crowd Simulation: Implementing a Parallel Action Server", 37th International Conference on Parallel Processing.

[26] Vigueras, G., Lozano, M., Orduña, J.M. (2011), "Workload balancing in distributed crowd simulations: the partitioning method", The Journal of Supercomputing archive Vol 58 Issue 2, Kluwer Hingham, MA, USA

[27] Huang, T. (2008). Discriminative local binary patterns for human detection in personal album. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE.

Molecular Simulation of Hydration Layers on Hydrophilic Mineral Surfaces in Water

Qian Wan[1, 2], Zili Huang[1], Shaoxian Song[2,*]
[1] *College of Resources and Environmental Engineering, WuhanUniversity of Science and Technology, Wuhan, China*
[2] *Instituto de Metalurgia, Universidad Autónoma de San Luis Potosí, San Luis Potosí, Mexico*
e-mail: shaoxian@uaslp.mx

Abstract

Hydration Layers on mineral surfaces in water plays a very important role in mineral processing, including mineral flotation and dispersion. Molecular simulation with computations is a useful way to approach into the fundamental aspects of the hydration layers. In this paper, the molecular simulation of hydration layer on mineral surfaces in water is highlighted. It includes the structure of the hydration layers, density and viscosity in the layers, hydration layer thickness, hydrogen bonding, and adsorption sites, etc. Also, the commonly used software in the molecular simulation is summarized.

Keywords:Hydration layers; molecular simulation;

1. Introduction

When water molecules come close to a hydrophilic surface, they orientate forward the surface because of the stronger force between surface and water molecules than between water molecules themselves, forming specific water molecule layers on the surfaces. The layer is called hydration layer. The water in hydration layers is different from that in bulk water, having a tighter structure, a higher density and viscosity [1, 2]. For instance, talc particle consists of hydrophobic basal planes and hydrophilic edges. Hydration layers appear on the edge, instead of the basal plane, as shown in Figure. 1 [3].

Hydration layer plays a very important role in many fields such as biology, material science and engineering, environmental science and engineering, etc [4]. In mineral processing, it strongly affects mineral flotation and dispersion. Numerous experimental approaches have been carried out to understand the hydration layers on mineral surfaces in water. However, due to the limitation in space and time, they have not revealed all the behavior of hydration layers. With small scales in both time and space, molecular simulation might obtain the details on the microscopic information of the hydration layers.

This paper attempts to highlight the results of recent molecular simulations of hydration layers on mineral surfaces in water.

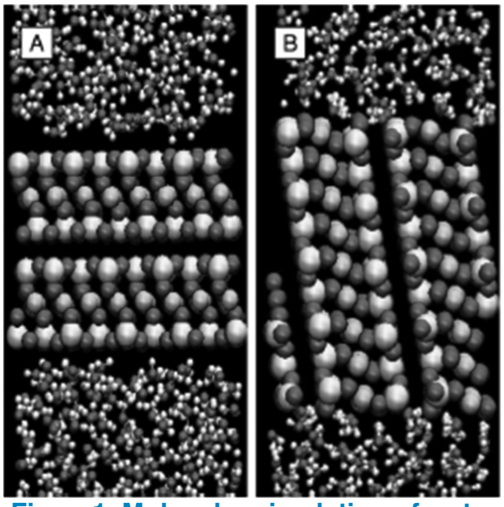


Figure1. Molecular simulation of water on the talc basal plane surface (A) and the edge surface (B) [3]

2. Hydration layers on mineral surface

The structure of hydration layer on a hydrophilic surface in water is schematically represented in Figure.

2. In the hydration layer, the water molecules pack tightly with an orientated arrangement. With the increase of distance to the surface, the attractive forces between the surface and water molecules decrease, leading the arrangement of the water molecules to be loose and weak orientation. At the distance that no attractive forces to the water molecules come from the surface, the water becomes bulk water again. The AFM and viscosity method are used to determine the thickness of hydration layer [5, 6], showing that the thicknesses of hydration layers on mica and silica in water were around is 30 and 14 nm, respectively.

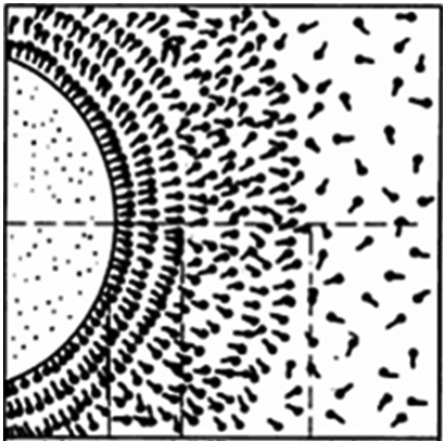


Figure2. The structure of hydration layers on hydrophilic surfaces in water.

The properties of water in hydration layers are much different from the bulk water. The studies with high-resolution x-ray crystallography and NMR spectroscopy have demonstrated the ordering and mobility of the molecules in the layers on hydrophilic surfaces[7], showing that on mica/water interfacesthere wasa strong oscillatory region of 0.25-0.27nmwith a higher density[8].

3. Molecular simulation of hydration layer

3.1. Building of System Configuration

3.1.1. Mineral Substrate. To begin a molecular simulation of hydration layer, the first thing is to build the desired system, assigning the positions and velocities to the atoms or molecules of the initial system. In our case, substrate mineral structures should be built first. The model of the substrates is often based on the experimental method like X-ray reflectivity

[9], neutron diffraction, etc. Figure. 3 shows the structure used for the surface simulations based on neutron diffraction data [10].

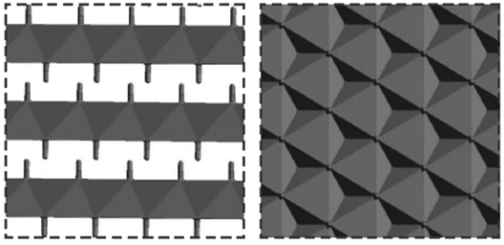


Figure3. The base crystal structures of brucite. [10]

3.1.2. Force Field Models of Water in Simulation. To begin a molecular simulation of hydration layer, the first thing is to build the desired system, assigning the positions and The water models play an important role in the research on mineral/water interface. Currently many different water models have been developed to study water, including three types [11]:

a) Simple interaction-site models. Each water molecule is maintained in a rigid geometry and the interaction between molecules is described using pairwise Coulombic and L-J expressions. Some properties cannot be determined by using a rigid model.

b) Flexible models. It permits internal changes in conformation of the molecule. In Toukan and Rahman's model, the variable O-H stretching makes a well described dynamical behavior [12].

c) Polarisbale models. It has been developed to include explicitly the effects of polarization and many-body effects.

In the simulation of water on the mineral surface, the simple point charge model and the flexible SPC water model often been used. The potential for the simple point model such as TIP3P is represented by

$$E_{ab} = \sum_{i}^{ona} \sum_{j}^{onb} \frac{k_C q_i q_j}{T_{ij}} + \frac{A}{r_{OO}^{12}} - \frac{B}{r_{OO}^{6}} \quad (1)$$

where kc the electrostatic constant; q_i and q_j are the partial charges relative to the charge of the electron; rij is the distance between two atoms or charged sites; and A and B are the Lennard-Jones parameters.

3.2. Force Field in Simulation

The description of the interatomic force is an important step of molecular simulation. There are two methods to do this, the quantum mechanical methods and force field methods (also known as molecular mechanics). Quantum mechanical methods deals with the electrons in a system, so that a large number of particles must be considered, leading to very large computations even if the semi-empirical schemes is used. Force field method ignores the electronic motions and only calculates the energy of a system as a function of the nuclear positions only, so that the computation is much faster [11].

The current force field describes the potential energy of a chemical system through summation of individual interatomic energies and the geometry of the molecular configurations as:

where E_{coul} and E_{VDW} are the non-bonded energy terms; $E_{bondstretch}$, $E_{anglebend}$ and $E_{inversions}$ are the bonded energy terms [13].

CLAYFF is a force field available in molecular simulation codes and was developed by Cygan and collaborators at the University of Illinois at Urbana-Champaign [14]. It is based on

an ionic (non-bonded) description of the metal-oxygen interactions associated with hydrated phases, and is widely used to probe the interface of water on mineral surfaces. All atoms are represented as point charges and are allowed complete translational freedom in the frame work. This force field is less accurate than the quantum method because it doesn't

$$E_{total}=E_{Coul}+E_{VDW}+E_{bondstretch}+E_{anglebend}+E_{inversion} \quad (2)$$

define all the chemical bonds, but it can study the complex interactions of solids, fluids and interfaces effectively and relatively simply.

3.3. System condition

3.3.1. Boundary conditions. The selection of boundaries and boundary effects are crucial to simulation methods, because macroscopic properties in simulation are obtained from small numbers of particles. In macroscopic systems, only a few atoms are close enough to the boundary. For a three-dimensional system with 1021 atoms, only 1014 atoms are near the boundary. However, for a typical MD with 1000 atoms, roughly 500 atoms are immediately adjacent to the boundary, leading simulation to be hard to capture the typical state of an interior. Therefore, the selection of an appropriate boundary condition is very important to the result of the simulation [15].

The boundary conditions include periodic boundary condition and non-periodic boundary conditions. Periodic boundary conditions do not consider the boundary effect, while non-periodic boundary conditions are just the opposite. Because of keeping the constant particle number in the simulation system and without considering the boundary effect, the periodic boundary condition is always used in the simulation of mineral/water interfaces.

3.3.2.Ensemble. An ensemble is an idealization consisting of a large number of mental copies of a system [16, 17]. Different macroscopic environmental constraints lead to different types of ensembles. The common ensembles include:

a) Microcanonical (NVE) ensemble: an ensemble of systems, each of which has the same energy, particles number and volume [18]. The characteristic state function is described as:

$$\Omega(U,V,N)=e^{\beta TS} \quad (3)$$

b) Canonical (NVT) ensemble: an ensemble of systems, each of which can exchange the energy with the same temperature, particle number and volume. The characteristic state function is described as:

$$Z(T,V,N)=e^{-\beta A} \quad (4)$$

c) Grand canonical ensemble: an ensemble of systems, each of which can exchange the energy and particle number with the same volume, temperature and chemical potential. The characteristic state function is described as:

$$E(T,V,\mu)=e^{\beta PV} \quad (5)$$

The choice of ensemble is based on the properties which needed to obtain.

3.4. Methods of molecular simulation

3.4.1. Molecular dynamic simulation. Molecular dynamic is a computer simulation by Newton's laws of motion to compute the positions and velocities of the particles in the

system varying with time for the trajectories of all atoms in the system. A variety of properties can be obtained from these trajectories. It is divided as three basis steps. First, the initial positions and velocities should be provided by stochastic acquired basis for Maxwell-Boltzman distribution. Secondly, the forces on the particle at each step with the current positions and velocities are computed to generate new positions and velocities. The atoms are moved to the new positions and an updated force is computed, and so on. The trajectory that describes the dynamic variables changing with time would be established by solving the differential equations embodied in Newton's second law:

$$\frac{d^2 x_i}{dt^2} = \frac{F_{x_i}}{m_i} \quad (6)$$

where x_i is the distance of particle moves in t time, m_i is the mass of the particle i; t is the time of the particle moves; F_{xi} is the force to the particle when it move. Thirdly, the trajectory is used to calculate relative physical quantities to obtain various properties about thermodynamics and statistic mechanics.

3.4.2. Monte carlo simulation. The Monte Carlo simulation is the first computer technique for the simulation of a molecular system. It is a category of computational algorithms to calculate the results by repeated random sampling. It has four basic steps for molecular simulation:

a) A random system configuration is generated by a random number generation.

b) The positions of the molecules in this molecular configuration are irregularly changed and then the potential energy of new configuration is calculated.

c) The energies of the two configurations are compared. If the energy of the new configuration is lower than the old, the new configuration would be used for iteration. If it is the opposite, the Boltzmann factor is computed while a random number is generated from the computation. If the random number is larger the Boltzmann factor, this new configuration would be abandoned; if the number is smaller than the Boltzmann factor, the new configuration would be used for further iteration.

d) The iteration is repeated until the new configuration satisfies the requirements.

In general, molecular dynamics method is widely used to calculate time-dependent quantities, such as transport coefficient, In the case of studying the systems in certain ensembles, it is better to use the Monte Carlo method.

3.5. Molecular simulation programs

3.5.1. Gaussian. Gaussian is a computer program for computational chemistry initially released in 1970 as Gaussian 70. It has been used in quantum chemistry and other fields.

3.5.2. Gaussian. GROMACS is a molecular dynamic package developed in the Biophysical Chemistry department of University of Groningen. It is primarily designed for simulations of proteins, lipids and nucleic acids [19, 20]. Now, GROMACS is one of the fastest and most popular software packages available [21].

3.5.3. Material Studio. Material Studio is a software developed by Accelrys. It offers a complete modeling and simulation environment that allows researchers in materials scien-

ce and chemistry to predict and understand the relationships of a material's atomic and molecular structure with its properties and behavior.

4. MD simulation of hydration layers on mineral surfaces

4.1. Structure of hydration layer

4.1.1. Water on interface. Structure of hydration layer has been of interest to many researchers. The MD simulation on muscovite showed the atomistic structure and dynamics of the hydration layer up to 3 nm in thickness [22]. The similar result also demonstrated that the hydration layer on hydroxyapatite was about 2~3 structured water and had ice-liked behaviors [23].

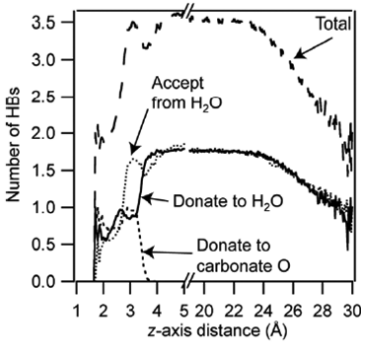
The MD simulation for brucite surfaces in water has shown the detailed structure of water on the surface, which consisted of three obvious delamination of hydration layer [24]. The first layer was near the surface with high density and the water molecule in the layer directly contacted the substrate. The distance between the center of this layer and surface was of 0.25 nm. The second layer had a relatively low density with the distance about 0.4nm apart from the surface. The third layer was a transitional layer with the thickness about 0.5~1.5nm, in which the water structure became more like bulk water.

4.1.2. Number of H-bond in hydration layer

In early 1950s, water structure in hydration layer was considered as ice-like because of the higher viscosity. MD simulation studies have demonstrated that this hypothesis was not true. Figure.4 illustrates the MD simulation on calcite/water interface on the H-bond number in the hydration layers [25]. It showed that there were 2 H-bonds in the first water molecular layer near the calcite surface (0.15 nm distance). Then, the number of H-bonds increased with the distance and reached 3.5 around 0.5 nm distance. In bulk water the H-bond number is about 3.5 at ambient temperature and pressure, while in ice, the H-bond number is about 4.0. Obviously, there is no ice structure in the hydration layer. In other words, the water in hydration layer could not be simply termed as "ice-like" water. The MD simulations for water on the surface of muscovite [22] and brucite [25] also suggested that the H-bond number of water in the interfaces were are not similar to that in ice.

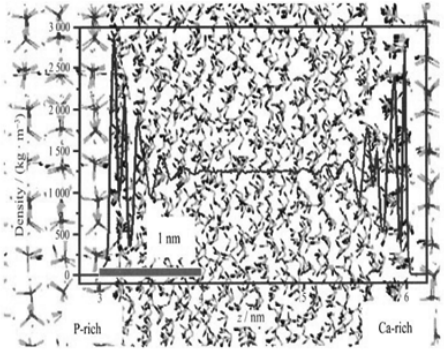
4.2. The property of hydration layer

4.2.1. Density. The water in hydration layers has a more compact structure, which is significantly different from the bulk water. The MD simulation on hydroxyapatite (001)/water showed that the density of water nearest the surface could be larger than 2.5g/cm3 [23]. The density oscillating decreased with the increase



of the distance, as shown in Figure. 5.

Figure. 4. Type and total number of



hydrogen bonds on calcite surface [25]

Figure5. Snapshot and density profile of water on the hydroxyapatite (001)/ water interface [23]

4.2.2. Diffusion coefficient. The MD simulation for magnesium-smectite/water interface at 300K showed that the water self-diffusion coefficient was of 2.48×10-10 m2s-1 [27], while it is of 2.3×10-9m2s-1 in bulk water. This result suggested a substantial hindering of diffusion at a short range of distance to the substrate. The diffusion coefficient decreased in the high density regions and increased in the low density regions. The layered structure of water at the interface in the direction normal to the surface indicated that there were energy barriers separating the different layers [28].

4.3. Factors of affecting hydration layer

4.3.1. Hydrophilicity of mineral surface.

The minerals of hydrotalcite, gibbsite, brucite and portlandite have similar rhombic arrays of OH groups on the surfaces, indicating a similar hydrophilicity, while there are very different surface charges and cation occupancies on the octahedral sheet [10, 29]. Table 1 gives the distances appeared density peaks to the surfaces of the minerals in water. For the four minerals, the density peaks appeared at very similar distances, indicating that substrate structural charges have a very slight effect on the hydration layers. As already stated, talc is featured by the strong hydrophobicity of the basal planes. The density peaks of talc appeared at different distance from the four minerals. These results suggested that surface hydrophilicity is crucial to defining the structure of hydration layers. The MD simulation of hydration layers on hematite/water interfaces also identified that the surface characteristics predominate the interfacial water structure [30].

	First density peak	Second density peak	Third density peak
Hydrotalcite	0.24	0.495	0.63
Gibbsite	0.255	0.51	0.65
Brucite	0.257	0.49	0.69
Portlandite	0.255	0.475	0.645
Talc	0.31	0.62	

Table 1. The distances appeared the density peaks to surfaces for various kinds of minerals (nm).

4.3.2. Orientation of water molecule on mineral surface.

The orientation of water molecule on mineral surfaces is different from hydrophobic surfaces to hydrophilic surfaces, as shown in Figure.1. On the basal planes of talc, the orientation of interfacial water preferentially parallel to the surface, because the surface has no hydrogen boning donor/acceptor sites. The MD simulation showed that the mean H2O orientation is characterized by dipole parallel to the surface [31]. On the edges of talc, interfacial water dipoles pointed away from the surface due to oxygen atoms afford

electron to participate in hydrogen bond [3]. The orientation also affects the structure of hydration layers on mineral surfaces in water.

5. Conclusions

Hydration layer is a structured water layer and has some behaviors different from the bulk water. However, due to the limitation in space and time, some results may not be revealed by the experimental approaches. The details of hydration layer on hydrophilic mineral surfaces in water, like density and viscosity in the layers, layer thickness, hydrogen bonding, etc., can be obtained through MD simulation. So the MD simulation plays an important role to study hydration layer and reveal the effect of hydration layer in many fields.

Acknowledgement

The financial supports for this work from the Consejo Nacional de Ciencia y Tecnología (CONACyT) of Mexico under the grant No. 155148 are gratefully acknowledged.

References

[1]C. Peng, Q. Gu, and S. Song, “Effect of Solvation Film on the Viscosity of Colloidal Dispersions”, Chinese J. Chem., vol.23, pp.603-607, 2005.

[2]S. Song, C. Peng, M.A. Gonzalez-Olivares, A. Lopez-Valdivieso, and T. Fort, “Study on hydration layers near nanoscale silica dispersed in aqueous solutions through viscosity measurement”, J. Colloid Interface Sci., vol.287, pp.114-120, 2005.

[3]H. Du, and J.D. Miller, “A molecular dynamics simulation study of water structure and adsorption states at talc surfaces”, Int. J. Miner. Process., vol.84, pp.172-184, 2007.

[4]J.J. Virtanen, L. Makowski, T.R. Sosnickand, and K.F. Freed, “Modeling the Hydration Layer around Proteins: HyPred”, Biophys. J., vol.99, pp.1611-1619, 2010.

[5]C. Peng, S. Song, and Q. Gu, “Determination of hydration film thickness using atomic force microscopy”, Chinese Sci. Bull., vol. 50, pp.299-304, 2005.

[6]S. Song, and C. Peng, “Thickness of Solvation Layers on Nano-scale Silica Dispersed in Water and Ethanol”, J. Disper. Sci. Technol., vol.26, pp.197-201, 2005.

[7]W. Zhou, S. Song, M.A. Gonzalez-Olivares, A. Lopez-Valdivieso, C. Ke, and Y. Zhang, “Experimental Study on Viscosity of Colloidal Silica in Aqueous Electrolytic Solutions”, J. Disper. Sci. Technol., vol.29, pp.842-847, 2008.

[8]L. Chen, P. Fenter, K.L. Nagy, M.L. Schlegel, and N.C. Sturchio, “Molecular-Scale Density Oscillations in Water Adjacent to a Mica Surface”, J. Phys. Rev. Lett., vol.87, pp.156103-1-4, 2001.

[9]S. Kerisit, C. Liu, and E.S. Ilton, “Molecular dynamics simulations of the orthoclase (001) - and (010)-water interfaces”, Geochim. Cosmochim. Acta, vol.72, pp.1481-1497, 2008.

[10]J. Wang, A.G. Kalinichev, and R.J. Kripatrick, “Effects of substrate structure and composition on the structure, dynamics, and energetics of water at mineral surfaces: A molecular dynamics modeling study”, Geochim. Cosmochim. Acta, vol.70, pp.562-582, 2006.

[11]A.R. Leach, Molecular modeling principles and

applications. 2nd ed., Academic Press: Waltham, 2001, pp.216.

[12]K. Toukan, and A. Rahman, "Molecular-dynamics study of atomic motions in water", J. Phys. Rev. B, vol.31, pp.2643-2648, 1985.

[13]R.T. Cygan, "Molecular Modeling in Mineralogy and Geochemistry", Rev. Mineral Grochem., vol.42, pp.1-35, 2011.

[14]R.T. Cygan, J, Liang, and A.G.Kalinichev, "Molecular Models of Hydroxide, Oxyhydroxide, and Clay Phases and the Development of a General Force Field", J. Phys. Chem. B, vol.108, pp.1255-1266, 2004.

[15]D.C. Rapaport, The art of molecular dynamics simulation. 2nd ed., Cambridge University Press: Cambridge, 2004, pp.15-16.

[16]K. Charles, and H. Kroemer, Thermal Physics, 2nd ed.,San Francisco: Freeman and Company, pp.31, 1980.

[17]L.D. Landau, and E.M. Lifshitz, Statistical Physics,New York: Pergamon Press, pp.9.

[18]D.A. Beard, and H. Qian, Chemical Biophysics: Quantitative Analysis of Cellular Systems, Cambridge: Cambridge University Press, pp.282.

[19]D.V.D. Spoel, E.Lindahl, B. Hess, G. Groenhof, A.E. Mark, and H.J.C. Berendsen, "GROMACS: Fast, Flexible, and Free", J. Comput. Chem., vol. 26, pp.1701-1718, 2005.

[20]B. Hess, C. Kutzner, D.V.D. Spoel, and E. Lindahl, "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation", J. Chem. Theory Comput., vol. 4, pp.435, 2008.

[21]C. Kutzner, D.V.D. Spoel, M. Fechner, E. Lindahl, U.W. Schmitt, B.L.D. Groot, and H. Grubmüller, "Speeding up parallel GROMACS on high-latency networks", J. Comput. Chem., vol.28, pp.2075-2084, 2007.

[22]J. Wang, A.G. Kalinichev, R.J. Kirkpatrick, and R.T. Cygan, "Structure, Energetics, and Dynamics of Water Adsorbed on the Muscovite (001) Surface: A Molecular Dynamics Simulation", J. Phys. Chem. B, vol.109, pp.15893-15905, 2005.

[23]H. Pan, J. Tao, T.Wu, R. Tang, "Molecular simulation of water Behaviors on Hydroxyapatite Crystal Faces", Chinese J. Inorg. Chem., Vol.22, pp.1392-1400, 2006.

[24]R.J. Kirkpatrick, A.G. Kalinichev, and J. Wang, "Molecular dynamics modeling of hydrated mineral interlayers and surfaces: structure and dynamics", Mineralogical Mag., Vol. 69, pp.287-306, June 2005.

[25]T.D. Perry IV, R.T. Cygan, and R.Mitchell, "Molecular models of a hydrated calcite mineral surface",Geochim. Cosmochim. Acta, vol.71, pp.5876-5887, 2007.

[26]F.R.C. Chang, N.T. Skipper, and G. Sposito, "Computer simulation of interlayer molecular structure in sodium montmorillonite hydrates", Langmuir, vol.13, pp.2734-2741, 1997.

[27]J.A. Greathouse, K. Refson, and G. Sposito, "Molecular Dynamics Simulation of Water Mobility in Magnesium-Smectite Hydrates", J Am. Chem. Soc., vol.122, pp.11459-11464, 2000.

[28]S. Kerisit, and C. Liu, “Molecular Simulations of Water and Ion Diffusion in Nanosized Mineral Fractures”. Environ. Sci. Technol., vol.43, pp.777-782, 2007.

29]A.G. Kalinichev, and R.J. Kirkpatrick, “Molecular dynamics modeling of chloride binding to the surfaces of Ca hydroxide, hydrated Ca-aluminate and Ca-silicate phases”, Chem. Mater. vol.14, pp.3539-3549, 2002.
[30]S. Kerisit “Water structure at hematite–water interfaces”, Geochim. Cosmochim. Acta, vol.75, pp.2043-2061, 2011.

[31]J.Wang, A.G. Kalinichev, and R.J. Kirkpatrick, “Asymmetric Hydrogen Bonding and Orientational Ordering of Water at Hydrophobic and Hydrophilic Surfaces: A Comparison of Water/Vapor, Water/Talc, and Water/Mica Interfaces”, J. Phys. Chem. C, vol.113, pp.11077-11085, 2009.

Speeding-up the Rendering Process of High Definition Animations using Four Quad-Core Intel Xeon Microprocessors

Alfredo Cristóbal-Salas, Silverio Pérez-Cáceres, Raul Varguez-Ferndández, Efrén Morales-Mendoza, Salvador Santos-Reyes, Emmanuel García-Gómez, Jesús Valdemar Pérez-San Martín, Angel-Allan Hernández Quezada, Miguel Jimenez-Zárate, Delfino Lazcano-Vargas
Facultad de Ingeniería en Electrónica y Comunicaciones, Universidad Veracruzana
Poza Rica de Hidalgo, Veracruz, México 93390
{acristobal, sperez, rvarguez, efmorales}@uv.mx

Abstract

This paper presents the experience of improving the rendering process of high definition animations using Blender 2.64a and the native option for rendering called Blender-Render. A 3D simulation of a mechanical equipment was used to test the improvements in a 2x2.4GHz Quad-Core Intel Xeon MACPRO workstation. Also, we analyze the impact in energy consumption in each optimization. Results show that energy issues could be a key point to consider when rendering large 3D animations.

Keywords: Rendering optimization, mechanical equipment, mechanical vibrations, Blender, green computing.

I. Introduction

This paper presents preliminary results about the design of a course for mechanical vibrations measurements in which it is necessary to consider interaction with virtual mechanical equipment that suffer from imbalance, misaligned or metal-metal friction. This project requires of 3D animations in 1280x1080 resolution and 24 fps where employees can visualize all possible problems that could be found in the area of mechanical vibrations. Training is an important area in the today´s industry due to the need to have qualified and productive personnel. Also, training allow employees to understand the high-risk situations and conditions that helps to reduce corrective maintenance.

The article 3 in Mexican Federal Labor Law (ley federal del trabajo) establishes the social interest of promoting and monitoring the education, staff training, and skill development [15]. For companies or organizations, human resource training should be an activity of vital importance because it contributes to personal and professional development of individuals within the organization, while resulting in benefits for the company.

According to Santiago García Garrido in his work: "organization and all-inclusive maintenance management", maintenance can be defined as "the set of techniques designed to keep equipment and facilities in service for as long as possible (looking for high availability) and at peak performance" [10]. Maintenance is the way industries take care of the plat needs while seeking to obtain better performance. According to Dr. Kunitoshi Sakurai [16] in his work called "equipment maintenance", some basic objectives for maintenance are: (1) To reduce equipment breakdowns. (2) Reduce

injuries caused by malfunctioning equipment. (3) Maintain punctuality in service, (4) Prevent an increment in operating costs, such as bonuses, excessive repairs, very short shelf life of the equipment, etc. (5) Protect the high investment in equipment. On the other hand, industrial maintenance can be divided in two groups: preventive and corrective. Through preventive maintenance it is possible to detect defects in equipment performance and to apply corrective decisions before any emergency or failures may occur. This corrective maintenance takes care of equipment repair after their malfunctioning.

Today, most of the industries, as part of their mechanical predictive maintenance programs and monitoring, use vibration analysis to establish what is the state of health of mechanical equipment and, in particular, how are their most critical parts and thus to prevent catastrophic failures [8].

In the area of maintenance, the intention is to have a well trained personnel staff able to prevent equipment failures and to diagnose different machinery. Also, another point of great importance in the area of prevention is to provide training to employees in order to improve their knowledge and skills. With this kind of staff, companies can improve confidence and productivity while saving resources not caring if they are new employees or have years of experience [6].

One of the most important demands of today's organizations is to have multi-talented employees who can develop several activities and they can also adapt themselves to the needs of the workplace and organizational demands, including all processes related to learning that allow them to be better and constantly prepared.

Various factors can limit the understanding when taking a course such as: the lack of imagination, apathy, course length, among others, in spite of also external factors such as work pressure, emotional intelligence of employees, weather conditions, etc. We can say that the integration of ICT and the teaching/learning process are an important and transcendent point of discussion. Like any other technology, multimedia technology can be used and abused, but when it is used properly, multimedia programs have many benefits (Adams, 1992).

One of the indicators of quality of education in technologically developed countries is how the digital gap (social division between those who know and do not know how to use new technologies to improve their social and labor relations) is faced and reduced [2].

II. Mechanical Maintenance Teaching Difficulties

This project was develop together with the mechanical maintenance department at PEMEX´s "Escolín" Petrochemical complex located at Poza Rica, Veracruz. As a first step in the development process, the course that the company uses to train their employees was analyzed and their instructors were interviewed. From this first step, the main difficulties related to the teaching/learning process for mechanical maintenance were detected. Some of the difficulties are: (a) operator staff has only basic knowledge in engineering concepts. (b) Courses are designed as a lecture with almost no interaction between participants and lecturers. (c) Didactic material is mostly slide presentations with few 2D animations. (d) Courses are designed to cover sessions of 8 hours long. (e) Practice sessions can only occur only when there is some mechanical equipment with any kind of problem or malfunction. (f)

any practice sessions involves funding, thus in order to keep the cost of course under budget it is necessary to plan few of these sessions.

After the analysis of these difficulties, some ideas on how to improve the original course emerged, such as: (a) it is necessary to review and improve all theoretical sections in the slide presentations in order to explain concepts in a more efficient way. (b) Design a new version of this course that can be more interactive and customized to participants and their level of knowledge in engineering and their skills in mechanics.

III. System Requirements Analysis

The entire course was re-designed into a web based online course that considers the introduction of 3D animations that help employees (operator's staff) to understand how to perform mechanical maintenance. With these modifications participants can review didactic material as many times as they needed in order to understand theoretical concepts and view all study cases related to mechanical maintenance.

In [4, 12], it is mentioned that introduction of multimedia in a course could be positive for instructors and participants. Considering the theory developed by David Perkins (1995), co-director of Project Zero Research Center for Cognitive Development, this theory tells us that you learn more when you have a reasonable opportunity and motivation to do it; so, as starting point one can state that: considering a task to be taught, if clear information is given by examples and descriptions, if students have enough time to practice the activity and to think how to approach it, if informative feedback is provide by means of clear and accurate advices to improve student performance and if instructor works from a platform of intrinsic and extrinsic motivation, is likely to produce significant achievements in teaching [13].

According to the learning pyramid proposed by Cody Blair about how people learn and remember more effectively, it is observed that the use of multimedia content such as visual and auditory elements have a relationship between the retention rate after 24 hours. Relationships for "listening" has 5% retention while "reading" represents 10% and the use of "visual elements" by 20% [14].

Also, it is important that visual elements are linked together through Hypermedia were it is configured as a medium in which the information, in the form of interconnected networks, allows the user to navigate freely, while the various sensory pathways are activated. These materials offer several advantages [3, 7]: (1) Facilitate access to information and enhance learning of concepts. (2) Allow better adaptation to users' characteristics, attitudes and skills as well as the characteristics of the content, showing a phenomenon, concept or object from different symbolic systems. (3) Increase motivation and arouse positive attitudes in course participants. (4) Develop intellectual skills, and the application of new learning strategies, not based on rote learning.

Our system is based on the idea of using 3D animations that simulate different situations described in text that work as visual representations that support theoretical topics. Simulations can be used to create virtual constructivists learning environments during the educational learning process. This environment involves the processing of projects, issues or problems of interest to course participants who generate research process and develop of skills [7, 9]. Moreover, course participants gain a better understanding of systems, processes or

concepts exploring real phenomena, testing hypotheses or explanations discovering when interacting with simulations [5]. This interactivity allows restructure their mental models in order to compare the performance of models with the previously acquired [11]. The simulations consider the case studies which analyze the different disturbances that may be present in the machinery and expose theoretical information in a more innovative way.

IV. 3D Animations Design And Implementation

This project considers 3D animations of: Rolling and Arrows, pulleys, bearings, belts, motors, and gears. Animations should be in high quality (24 fps) in order to be projected in HDTV. So, this project considers two animations resolutions: DVCPRO HD 1080p 1280x720 and HDTV 720 1920x1080. Despite of the rest of animations, in this paper we focus on the design, implementation and analysis of just one animation: "measurement of vibrations in a motor-bomb using the VibroTest VT60". This animation was implemented in Blender 2.64a. The final design contains: 20 objects, 4 lights (2 point-light, 1 area light and 1 spotlight), the environment was filled with solid black and it has five textures: wall (691 kB), wood base (2,675kB), green metal table (285kB), floor (156kB) and the vibrotest (signal transducer) (31kB). This animation has the following features activated: anti-aliasing, textures, shadows, subsurface, environment map and ray-tracing. Figure 1 shows a screenshot of the animation in Blender. Two render motors were used to generate the animations: (a) Blender render. The native render engine that exploits multithreading and (b) NetRender. It configures computer to be used as a farm. The animations were created in an 8-core MACPRO 2.4Ghz, 12MB cache L3, 8GB SDRAM ECC DDR2 1024Mhz, GDDR5 1Gb ATI Radeon HD 5870 running OSX server 10.7 (see Figure 2).

V. Speeding Up Rendering

In Figure 3 the time spent for rendering using the Blender-Render engine is presented. In this figure it is possible to see that in both cases rendering time reduces as more threads are introduced for resolution DVCPRO-HD1080p there is no relevant time reduction after 4 threads working during the rendering. The same happened for resolution HDTV-720p after 12 threads.

In Figure 4 the speedup achieved when varying the number of threads is presented. The maximum speed up for resolution DVCPRO-HD1080p is 4.63 and the maximum speed up for resolution HDTV-720p is 5.33.

Figure 5 shows the KW/h spent for rendering; this measurements were obtained according to the formula (W=V*I) where "V" is 126V registered by the UPS hardware during the rendering and "I" is set to 12 amperes according to the hardware technical specifications. Then, the power needed by the MACPRO is the 1512W. The value shown in Figure 5 is the multiplication of hours spent for rendering by 1512W. In figure 6, it is possible to see the cost for rendering in each resolution and varying the number of threads. This value was obtained by multiplying the KW/h by 0.9318 which is the rate in Mexican pesos for the region where the Petrochemical Complex "Escolín" is located.

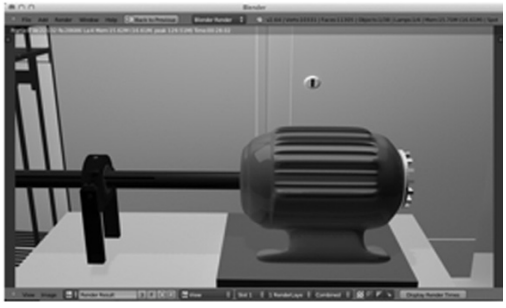


Figure 1. Design of the animation: "measurement of vibrations in a motor-bomb using the VibroTest VT60" using Blender 2.64a

Figure 2. Hardware used to run animations.

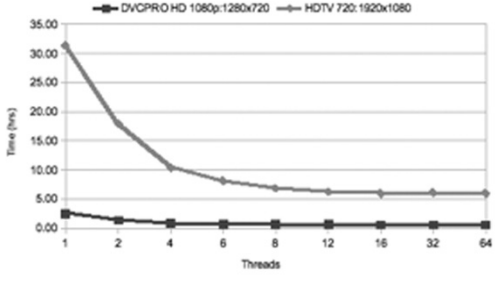


Figure 3. Time reduction when increasing the number of threads using Blender-Render.

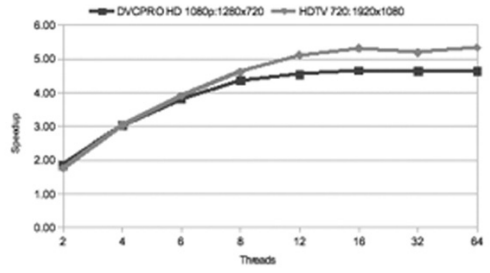


Figure 4. Speedup achieved when varying the number of threads using Blender-Render.

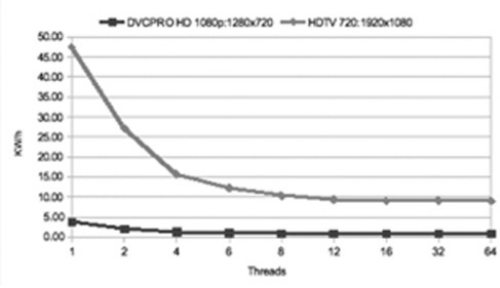


Figure 5. Energy consumption in KW/h during rendering varying the number of threads using Blender-Render.

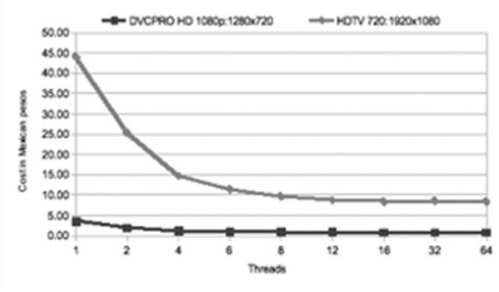


Figure 6. Approximation of Cost of rendering measured in mexican pesos for the energy-cost for gulf of Mexico region using Blender-Render.

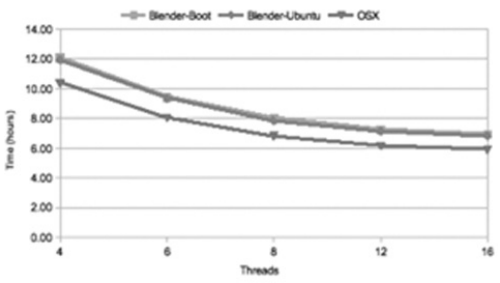


Figure 7. Time reduction obtained when varying the number of threads for the three Blender installation: Blender-Boot, Blender-Ubuntu and OSX

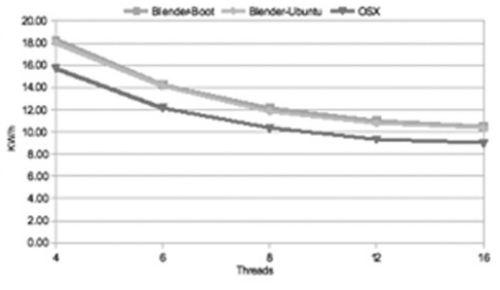


Figure 8. Comparison of the KW/h spent in each Blender installation: Blender-Boot, Blender-Ubuntu and OSX

As additional techniques to reduce rendering time, the previous results were compared with Blender software running on two GNU/Linux distributions: Blender-Boot and Blender-Ubuntu. Blender-boot is designed to be a dual boot Linux distro. It boots straight into a light environment with Blender already open. It is based on Ubuntu 11.10 for ease of use and the Software Center for installing more applications. Blender-Ubuntu is the implementation of Blender software in a regular Ubuntu distribution. These two linux distributions were installed in the same MACPRO workstation were previous renderings were made.

Figure 7 shows the reduction time for each of the three installations of Blender software. In this figure, it is possible to see that Blender software running on the native OSX have better performance (almost for one hour) than both Blender installations running on linux OS. It is also important to mention that for both Blender installations running on linux OS there is no significant difference in time reduction.

Figure 8 shows the KW/h (computed as mentioned before) spent in each of three installations of Blender software when varying the number of threads.

Acknowledgement

Authors want to thank to Universidad Veracruzana through Facultad de Ingeniería en Electrónica y Comunicaciones and the Red Iberoamericana de Supercómputo through the project #FP7-288883: “A network for supporting the coordination of Supercomputing research between Europe and Latin America” for the logistic and the financial support to accomplish this project.

VI. Conclusions

In this paper the experience of designing a high definition 3D animations using Blender Software is presented. Several experiments are presented when varying the number of threads used for the rendering process. Also, a comparison of the same software with the same animation and running on the same hardware but varying the operative system is also presented.

Preliminary results indicate that the use of these techniques improve performance of the rendering process. Results analysis reveals that it is possible to understand that installing Linux OS on a MACPRO hardware does not provide better performance. Another conclusion is that when dealing with HD animations it is convenient to use between 8 and 16 threads for rendering. These results agree with the theory that state that in an 8-cores computer, its best performance is achieved when there are one or as much as two threads per core. Finally, these optimizations allows to improve the degree of realism in teaching mechanical maintenance. Now it is necessary to research on the impact this 3D animations have on the participants and how the course is improved with the use of multimedia didactic material.

VII. References

[1] Adams GL. (1992). Why Interactive?. Multimedia & videodisc Monitor. pp. 20-25.

[2] Bautista García-Vera, A. (2004). Calidad de la educación en la sociedad de la información. Revista Complutense de Educación, 15, (2), 509-520.

[3] Cabero, J. (1999). Evaluación de medios y materiales de enseñanza en soporte multimedia. Pixel-Bit, 13, Artículo 3. Extraído 10 de Julio, 2009 de http://www.sav.us.es/pixelbit/articulos/n13/n13art/art133.htm

[4] Carballo Santaolalla, R. y M.J. Fernández Díaz (2005). "La actitud del profesorado de primaria y secundaria de la Comunidad de Madrid ante las TIC: problemática y claves para su integración". Actas del XII Congreso de Investigación Educativa: Investigación en Innovación Educativa. http://www.uv.es/aidipe/XIICongreso/ActasXIIcongreso.pdf

[5] Cataldi, Zulma (2000). Metodología de diseño, desarrollo y evaluación de software educativo. Argentina, 75 p Trabajo de grado (Tesis de Magister en Informática.Versión resumida). Universidad Nacional de la Plata. Facultad de Informática.

[6] Diez, Jennifer y José Luis Abreu* (2009). "Impacto de la capacitación interna en la productividad y estandarización de procesos productivos: un estudio de caso", en International Journal of Good Conscience, núm.2, Vol.4, México, pp 97-144.

[7] Esteban, M. (2002). El diseño de entornos de aprendizaje constructivista. Revista de Educación a distancia, 6 http://www.sav.us.es/pixelbit/articulos/n15/n15art/art158.htm

[8] Estupiñán, P Edgar, et. al., (2006), "Diseño e implementación de un analizador virtual de vibraciones mecánicas", en Revista Facultad de Ingeniería, núm. 01, vol.14, pp. 7-15.

[9] García, A. y M.R. Gil (2006). Entornos constructivistas de aprendizaje basados en simulaciones informáticas. Revista Electrónica de Enseñanza de las Ciencias, 5, (2) http://www.saum.uvigo.es/reec/volumenes/volumen5/ART6_Vol5_N2.pdf

[10] García, Garrido Santiago (2003),"La función mantenimiento", en Garcia, Organización y gestión integral de mantenimiento, Madrid España, Ediciones

Diaz de Santos, S. A., pp1.

[11] López-García y Morcillo (2007), "Las TIC en la enseñanza de Biología en la educación secundaria: los Laboratorios virtuales", en Revista Electrónica de Enseñanza de las Ciencias, núm. 3, vol. 6, lugar, 562-576.

[12] Orellana, N., Almerich, G., Belloch, C. y I. Díaz (2004). La actitud del profesorado ante las TIC: un aspecto clave para la integración. Actas del V Encuentro Internacional Anual sobre Educación, Capacitación Profesional y Tecnologías de la Educación, Virtual Educa http://www.uv.es/~bellochc/doc%20UTE/VE2004_5_6.pdf

[13] Perkins, D. (1995): "La enseñanza y el aprendizaje. La Teoría Uno y más allá de la Teoría Uno", en La escuela inteligente, cap. 3, pp. 68, 70 y 75, Barcelona, Gedisa.

[14] Salinas, J. (1994). Hipertexto e Hipermedia en la enseñanza universitaria. Pixel-Bit, 1, Artículo 2. Extraído el 10 de Julio, 2007 de http://www.sav.us.es/pixelbit/articulos/n1/n1art/art12.htm

[15] Sánchez, Castañeda Alfredo (2007) La capacitación y adiestramiento en México: Regulación, realidades y retos, en Revista Latinoamericana de derecho social Num.5, julio-diciembre de 2007, pp.191-2008.

[16] Sakurai, Kunitoshi (1980). "Mantenimiento de equipos" en Biblioteca virtual de desarrollo sostenible y salud ambiental, Lima, http://www.bvsde.paho.org/bvsacd/scan2/012012/012012-06.pdf, consultado el 01 de marzo de 2009.

Appendix I
Conference Keynote Speakers

Dr. Carl Kesselman
University of Southern California
United States of America
http://www.isi.edu/~carl/
Email: carl@isi.edu

Increasingly, management, analysis and sharing of large-scale data sets, so called big-data, plays a central role in scientific discovery. In large, well organized projects, the services and infrastructure required for big-data analysis is often a well defined, and consequently funded part of the operation of the project. However, in smaller teams, the up front planning for data management, analysis and sharing is the exception and not the rule. Small teams need overhead methods to define domain specific and use case specific approaches for accessing integrating and analyzing these data sources. In recognition of the specialized needs of small projects, we have receintly established the Institute for Empowering Long Tail Research with collaborators from the University of Chicago, the University of Washington, the University of California at Los Angles and the University of Arizona. In my talk, I will describe the work of the institute and illustrate the challenges and needs of small teams with use cases from a number of different scientific domains. I will present some recent work that we have done to address the problems of big-data and small teams and illustrate solutions for a number of different application domains.

Dr. Jack Dongarra
University of Tennessee
United States of America
http://www.cs.utk.edu/~dongarra
http://www.netlib.org/utk/people/JackDongarra/
Email: dongarra@cs.utk.edu

Algorithmic and Software Challenges when Moving Towards Exascale.

In this talk we examine how high performance computing has changed over the last 10-year and look toward the future in terms of trends. These changes have had and will continue to have a major impact on our software. Some of the software and algorithm challenges have already been encountered, such as management of communication and memory hierarchies through a combination of compile--time and run--time techniques, but the increased scale of computation, depth of memory hierarchies, range of latencies, and increased run--time environment variability will make these problems much harder.

Prof. Mateo Valero
Centro Nacional de Supercomputación
España
http://www.bsc.es/
http://www.personals.ac.upc.edu/mateo
Email: mateo@ac.upc.edu

Killer-mobiles: the way towards energy efficient High Performance Computers?

It is widely recognized that Exascale systems will be constrained by power. The Mont-Blanc project aims to build an alternative approach towards Exascale based on aggregating parts from the embedded and mobile market, which offer a better FLOPS/Watt ratio and a lower unit cost, at the expense of lower peak performance per chip. HPC systems built from these parts will require a higher number of processors, or resort to extensive use of compute accelerators. Using a higher number of chips increases the available memory bandwidth, alleviating the bandwidth wall, but increases the pressure on the interconnection network. The use of a high number of processors and accelerators, and the increased pressure on the interconnect require extensive code optimizations to achieve strong scaling, point to point synchronizations, and overlap data transfer with computation. The role of the OmpSs parallel programming model is paramount, as the key enabling technology that hides the complexity from the programmer, and transparently performs all the required optimizations. In this talk, we will review the design philosophies of several vendors, including HPC compute accelerators, and ARM-based mobile application processors in terms of peak performance, memory bandwidth, and energy efficiency; and we will review how the OmpSs programming models exploits the benefits of the Mont-Blanc approach while overcoming the drawbacks.

Dr. Rajkumar Buyya
Lab Department of Computing
and Information Systems
University of Melbourne
http://www.cloudbus.org/~raj/

High-Performance Cloud Computing

Computing is being transformed to a model consisting of services that are commoditised and delivered in a manner similar to utilities such as water, electricity, gas, and telephony. In such a model, users access services based on their requirements without regard to where the services are hosted. Several computing paradigms have promised to deliver this utility computing vision. Cloud computing is the most recent emerging paradigm promising to turn the vision of "computing utilities" into a reality. Cloud computing has emerged as one of the buzzwords in the IT industry. Several IT vendors are promising to offer storage, computation and application hosting services, and provide coverage in several continents, offering Service-Level Agreements (SLA) backed performance and uptime promises for their services. It delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. The price that Cloud Service Providers charge can vary with time and the quality of service (QoS) expectations of consumers. This keynote talk will cover (a) 21st century vision of computing and identifies various IT paradigms promising to deliver the vision of computing utilities; (b) the architecture for creating market-oriented Clouds by leveraging technologies such as VMs; (c) market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation; (d) Aneka, a software system for rapid development of Cloud applications and their deployment on private/public Clouds with resource provisioning driven by SLAs and user QoS requirements, (e) experimental results on deploying Cloud applications in engineering, gaming, and health care domains (integrating sensors networks, mobile devices), ISRO satellite image processing on elastic Clouds, and (f) need for convergence of competing IT paradigms for delivering our 21st century vision along with pathways for future research.

Dr. Carlos Jaime Barrios
Director of the High Peformance and Scientific Computing Unit
Universidad Industrial de Santander in Bucaramanga
Colombia

A Navigation Chart on Advanced Computing by/ for Latin American face to Exascale Era

Nowadays, the research and development of Latin-American countries have an important impact in the worldwide science and technology innovation. With an interesting development, Latin-American scientists and engineers perform communities with specific requirements in High performance computing architectures and platforms, from European and US experiences. Countries like Brazil and México led the others in some continental projects to propose ways to make Advanced-computing contributions to academic and scientific needs. The derived experience of these projects allows implementing infrastructure across the continent and more important, train minds in technical and scientific competences in high performance and advanced computing. In consequence, Latin-American scientists and engineers propose contributions based more in expectations and prospective than in needs, and moreover with a highest economical and social impact. This talk proposes a discussion from a roadmap on advanced computing face to challenges of the Exascale questions and how Latin America responds to these challenges observing energy-aware impact, environment, usability and scientific/technological realities.

Dr. Padmanabhan Iyer
Intel Corporation

HPC – An Intel Perspective

The race to Exa-scale is spurring exciting developments in CPU and interconnect architectures. While the Intel Xeon architecture continues to evolve in performance and applicability to a broad range of computing applications, the new Xeon Phi architecture shows an aggressive path forward in parallelism and energy efficiency. This talk will highlight some of the features of the Xeon Phi architecture and the programming models to utilize the same for High Performance Computing.

Dr. Jaime Klapp
Instituto Nacional de Investigaciones Nucleares y Cinvestav
México
Email: jaime.klapp@inin.gob.mx

On the evolution of HPC in México: are we finally approaching a new expanding era?

In this talk I describe the evolution of High Performance Computing (HPC) in México during the past three decades. A major problem in México for the creation and usage of large supercomputer sites has been the almost complete lack of very good connectivity between most academic institutions and supercomputer sites. This situation is changing very fast and by mid 2013 more than 1,100 academic institutions will be connected by the red Niba project (e-México project of the SCT). This will change the fate of HPC in México because potential users from all over the country (and the world) could efficiently use large supercomputer sites in México, some of which are currently being developed. I describe a specific case of a new large HPC project in which I have been involved that has been designed so that it will be self-sustainable.

Bob Anderson, Claude Paquette
Cray and Xyratex
Email: bob_anderson@xyratex.com , www.xyratex.com
Email: claudep@cray.com , www.cray.com

Cray and Xyratex

For more than three decades, Cray experts have been helping scientists and engineers solve challenging, complex computational problems. Claude Paquette will describe how Cray's Supercomputer and Cluster systems have evolved as users high performance requirements have grown. The same real-world performance that defines Cray's leadership in high-end supercomputing is also available in smaller systems that meet the needs of the expanding market for superior sustained performance and reliability. As the number of cores and processors have climbed, the I/O bottleneck has become a limiting factor in how fast data could be accessed from the storage systems. Cray introduced Sonexion, a radically new storage system to meet the demands of HPC. With over 150 engineers led by Dr. Peter Braam, (inventor of the Lustre file system), they built the fastest, most compact parallel storage appliance capable of delivering over 1TB/s and up to 100PB of storage, in less than 50% of the foot space of existing storage systems. Bob Anderson will take us through this new paradigm in storage technology

Boris Cortes Silva
NetApp / EMTEC
Email: bcortes@emtecgroup.net

NetApp / EMTEC

The NetApp® High-Performance Computing Solution for Lustre is purpose-built to efficiently scale bandwidth and density with uncompromised reliability, solving difficult research, modeling, and simulation problems.

Based on the NetApp E-Series storage platform and Whamcloud's Lustre highperformance file system, it offers an infrastructure that is designed for the flexibility, performance, and scalability required by the most demanding workflows to meet storage requirements today and in the future:

- *Proven performance. Delivers a dramatic increase in throughput and I/O, allowing massively parallel file access.*
- *Excellent scalability. Scales to tens of petabytes of data and tens of thousands of clients.*
- *Reliability. Meets uptime requirements of both business and research applications.*
- *Decreased costs. Significantly reduces deployment and support costs with preconfigured and pretested.*

Comité Organizador

UDG
Moisés Torres Martínez
Verónica Lizette Robles Dueñas

CNS - IPICYT
César Carlos Díaz Torrejón
Cynthia Lynnette Lezama Canizales

CINVESTAV
Mauricio Carrillo Tripp

CIMAT
Alonso Ramírez Manzares
Salvador Botello Rionda

UNISON
María del Carmen Heras Sánchez
Yessica Vidal Quintanar
Daniel Mendoza Camacho
Raul Gilberto Hazas Izquierdo

UCOL
Juan Manuel Ramírez Alcaraz
Carlos Alberto Flores Cortés

IPN
Juan Carlos Chimal Enguía
René Luna García

CICESE
Salvador Castañeda
José Lozano
Andrei Tchernykh

CUDI
Salma Jalife

UAM
Juan Carlos Rosas Cabrera
Manuel Aguilar Cornejo
Jaqueline Posadas Rodríguez

ININ
Jaime Klapp

UNAM
Lizbet Heras Lara
Jesús Cruz Guzmán

U. C. Berkeley
Nicholas Cardo

Author Index

Author Index

Printed in:
Los Talleres Gráficos de Transición
Mezquitan 465
Col. Artesanos
University of Guadalajara
December 2013
Printing 700 Issues/ 700 Ejemplares

www.isum.mx

This edition of **Supercomputing in México: A Navigation through Science and Technology** presents 19 original peer reviewed research papers written by 67 authors from México, Europe, United States, Asia and Latin America.

These works cover four thematic fields in the area of supercomputing. The four thematic fields include:

Applications
Architectures
Parallel Computing
Scientific Visualization

The contents of this volume 4, first edition can be of interest to computer science and engineering researchers, undergraduate and graduate students, professionals, and researchers working on HPC, as well as the general public interested in current research in Supercomputing.

Dr. Moisés Torres Martínez, Editor

ISBN: 978-607-450-919-9

ISBN: 978-607-450-919-9
9 786074 509199

CNS CENTRO NACIONAL DE SUPERCÓMPUTO
UNIVERSIDAD DE COLIMA
UNIVERSIDAD DE GUADALAJARA
CGTI
CONACYT
cudi