Shared Memory Supercomputing as Technique for Computational Electromagnetics

César Gómez-Martín, José-Luis González-Sánchez, Javier Corral-García, Ángel Bejarano-Borrega, Javier Lázaro-Jareño
[cesar.gomez, joseluis.gonzalez, javier.corral, angel.bejarano, javier.lazaro]@cenits.es

***Abstract*—Nowadays, it is not possible to innovate and investigate computational electromagnetics without being able to process and store huge amounts of data. Supercomputers along with high-performance computing techniques are aimed to provide methodologies and tools for computational electromagnetics researchers that will help them to solve problems in a more efficient and easy way. In order to get the best of those supercomputers it is important to know which kind of computer architecture is going to be used and how to achieve the best performance applying the appropriate programming model. Shared memory supercomputing is probably the easiest way of exploiting parallelism in computational electromagnetics, the combination of shared memory programming techniques like OpenMP with other distributed programming techniques will allow the improvement of supercomputer performance. Is is also very important to track and test new parallel programming models such as CUDA, Intel Ct or OpenCL because they are meant to exploit modern supercomputers efficiently. Automatic parallelization models are improving performance quickly but there is not a definitive model, they all have pros and cons and whether to use one programming model or another should be thoroughly studied.**

***Index Terms*—HPC, Supercomputing, Electromagnetism, CénitS, LUSITANIA**

I. Introduction

SUPERCOMPUTING is a widely used technology in all research areas. Most of the scientific investigations require simulations that let researchers know beforehand how a random experiment will behave, how the climate changes will affect farming, what is the impact of an industry or a refinery, what will happen in case of a nuclear/chemical disaster, etc. Summing up, supercomputing will help us know how, in every branch of science, certain behaviours will affect our way of living, i.e., R&D&I is not feasible without supercomputers and high-performance computing techniques. The performance of shared memory supercomputers, such as the supercomputer of Extremadura (LUSITANIA), will help researchers in obtaining results quicker than what they have never imagined. It is not possible to innovate today without a big computation capability or without being able to process and store huge amounts of data. This also applies to computational electromagnetics. Supercomputers along with high-performance computing techniques are aimed to provide methodologies and tools for researchers that will allow to solve problems in a more efficient and easy way. A set of concepts have to be addressed in order to understand how to properly optimize all sort of application and software.

César Gómez-Martín, José-Luis González-Sánchez, Javier Corral-García, Ángel Bejarano-Borrega, Javier Lázaro-Jareño are with CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura)

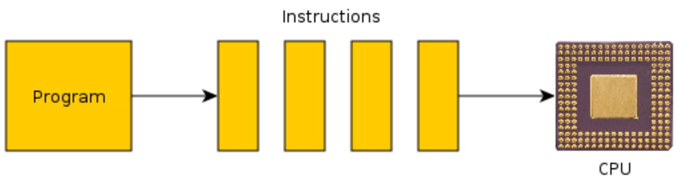

Fig. 1. Sequential Programming

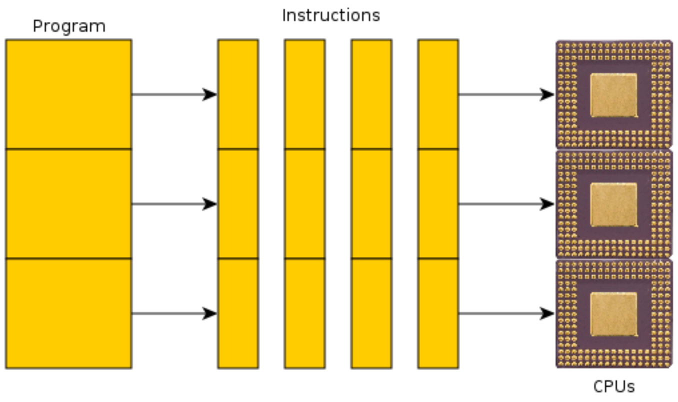

Fig. 2. Distributed and Parallel Programming

A. Programming Paradigms

1) Sequential Programming: The traditional way of solving problems with a computer is based on the execution of serial operations. Those calculations were usually executed on single processor computers, their instructions were processed in a sequential way (Fig. 1), i.e. one after another and only one instruction at the same time.

2) Distributed and Parallel Programming: Distributed and parallel programming [1] consists of using several resources simultaneously to solve specific problems (Fig.2). Instructions are executed on multi-core computers, the problem is divided into independent parts which are parallely executed on each processing unit.

Parallel programming is frequently mistaken with distributed programming because they have similar philosophies.

Despite that they both are based on the simultaneous use of several resources, parallel programming is not similar to distributed programming in the sense that the problem is solved in the same computer on a parallel approach, and, in a distributed environment, it is not necessary to use the same hardware with the same architecture or the same programming language. These are the pros of both programming paradigms:

- They can solve problems quicker than sequential programming.
- They provide solutions for bigger and more complex problems.
- They can study different variants of a problem parallely.
- Current processors come with more than one core, thus they are able to optimize the use of modern hardware.

B. Computer Classification

There are several [2][3] taxonomies that try to classify computers, these are the most popular:

- Flynn's Taxonomy: divides the computer universe into four classes depending on the number of concurrent instruction and data streams that an architecture can process simultaneously:
  - SISD (Single Instruction Single Data stream): computer with one sequential processor which exploits no parallelism.
  - SIMD (Single Instruction Multiple Data stream): array processors, vector processors or GPUs that exploit multiple data streams that can be naturally parallelized.
  - MIMD (Multiple Instruction Multiple Data stream): Multiple autonomous computers or processors executing different instructions among different data (e.g. Distributed and parallel systems).
  - MISD (Multiple Instruction Single Data stream): Multiple instructions operate on a single data stream. It is not a common architecture but it is usually used to run fault tolerance tests (e.g. Space Shuttle flight control computer).
- Feng's Taxonomy: Tsen-yun-Fen uses parallelism levels to sort computer architectures. The maximum parallelism level is the maximum number of bits that can be processed per time unit.
- Händler's Taxonomy: Wolfgang Händler uses not only parallelism level to classify computers, he also includes the number of paths inside computer hardware structures as a sorting factor.

Although those taxonomies are the most studied, the common way of classifying is the memory distribution of the computer, i.e: shared memory, distributed memory or hybrid.

1) Shared Memory: The same memory block may be simultaneously accessed by multiple processors (Fig.3), thus the changes applied to that block will affect all and every processor. There are two types of shared memory computers:

- UMA (Uniform Memory Access): processors are at the same distance from memory, they are pure SMP (Symmetric MultiProcessors) computers.

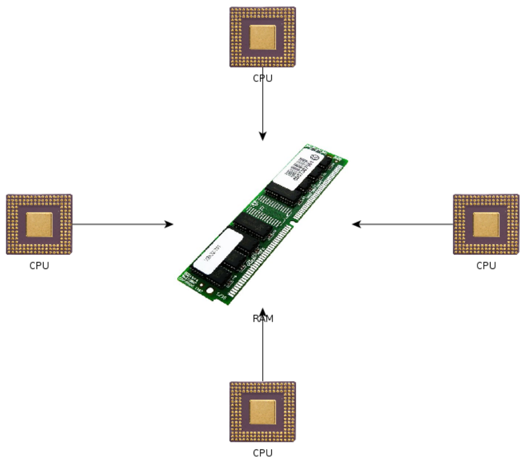

Fig. 3. Shared Memory

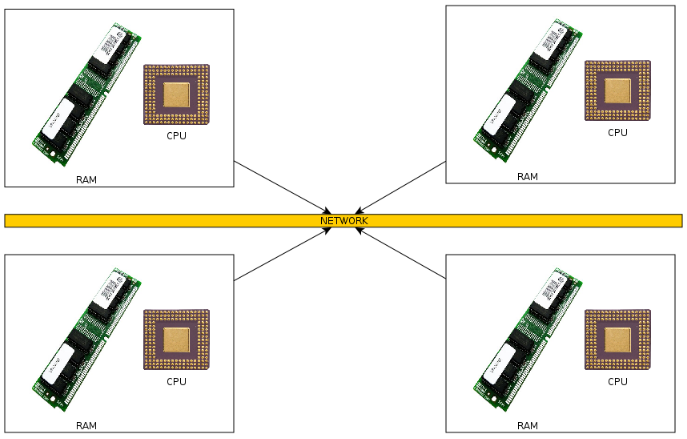

Fig. 4. Distributed Memory

- NUMA (Non-Uniform Memory Access): processors are at different distances from memory, they are mostly interconnected SMP computers.

One of the most important advantage of shared memory computers is that they are easy to program and quick when sharing data between processes or threads. The main disadvantage is the price of manufacturing computers with a lot of processors.

2) Distributed Memory: Each processor has its own local memory and it is not visible and also not accessible by the rest of processors (Fig.4). Obviously, there have to be communication between processors in order to share data, but the communication is done via network instead of using internal buses. The main advantage of this kind of architecture is that if more memories or processors are needed the cost is linear, it does not grow exponentially like in shared memory computers. Disadvantages are mostly due to the use of an external communication network to connect all the nodes:

- Network is usually a bottleneck if there are intensive communications between nodes.
- The programmer has to deal with communications and must synchronize all the different threads that are running parallely on the nodes.

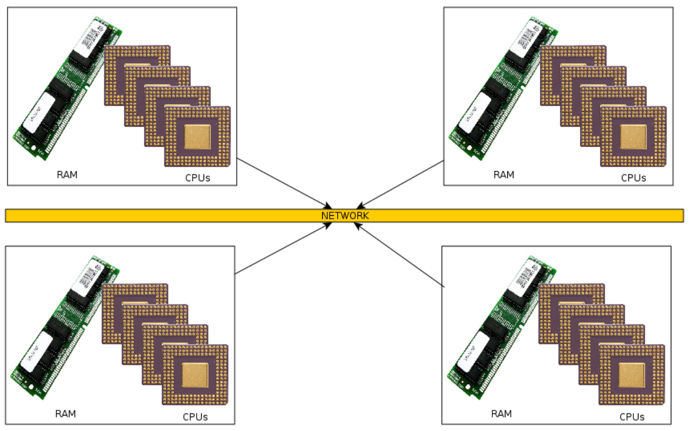

Fig. 5. Hybrid Computers

- Program parallelization could be hard or even impossible. (e.g. Mathematical functions with data interdependence - Fibonacci Series -).

3) Hybrid Computers: Hybrid computers share the advantages and disadvantages of shared memory and distributed memory computers. Hybrid computers are composed of interconnected sets of processors sharing the same memory, every set uses a communication network to send and receive data . They are usually interconnected SMP computers (Fig.5). These are their pros:

- Processor and memory scalability.
- Upgrading costs are linear.
- Communication network is not critical like in distributed memory computers.

Their cons:

- Still programmers have to manage communication among computing nodes.
- Parallelization could be hard or even impossible to achieve.

C. *What is the best technique for Computational Electromagnetics?*

Using one or another computing technique is not an exact science, but the fact is that most of the computational electromagnetics problems are very big, they need a lot of primary memory and most of them are hard or even impossible to partitionate or divide. For those cases, shared memory computers are the best choice. But hybrid computers have to be also taken in account, some problems do not fit in big shared memory supercomputers and the only possibility is to split the problem and use a hybrid approach. On the other hand, distributed memory supercomputers are less expensive and also a good choice if the problem does not fit in shared memory supercomputers and is also easy or feasible to divide it into smaller code portions.

II. New Trends in Computational Electromagnetics

A. *Supercomputing vs. GRID Computing vs. Cloud Computing*

Nowadays, there are multiple terms that are used when we address computational electromagnetics (Supercomputing, GRID Computing and Cloud Computing). It is important to know which technique fits better when facing a certain electromagnetic problem.

1) Supercomputing: A supercomputer has capacities and performances hard to achieve using common computers. The key feature is a very well optimized memory hierarchy [4], a supercomputer is carefully designed to be able to feed processors with instructions and data uninterruptedly, this is the main difference between supercomputers and normal computers. Their input/output system is also designed to handle high bandwidths continuously.

2) GRID Computing: The main difference between GRID computing and supercomputing is the way in which GRID computing uses all kind of resources simultaneously without having a central control. GRID computing is a new distributed computing technique that allows the use of heterogeneous computers connected via Internet [5]. GRID computing doesn't need to have all the resources in the same datacenter, this is its main advantage but also the biggest disadvantage. Pros of GRID computing:

- It is good to solve a lot of small problems.
- Scalability is virtually unlimited.
- All the nodes of the GRID will never be obsolete because they could be integrated with much modern technologies and all the components could be modified without affecting the correct operation.

Cons of GRID computing:

- It is not good to solve very big problems.
- Processes may not be interdependent, if there were intensive communications via Internet the system performance would degrade exponentially.
- System management is not trivial because public communication networks are being used, access and security policies are controlled with a very difficult to manage and configure middleware.

3) Cloud Computing: Similarly to GRID computing, cloud computing is an Internet based computation model. Cloud computing users are not generally infrastructure owners, they just consume third party resources and pay for those services. Analogously to power industry, cloud computing users only pay for what they use (computation time, storage and/or power consumption) [6]. Cloud computing should not be mistaken with other technologies but it combine derived features from supercomputing and GRID computing.

B. *MPI, OpenMP and New Parallel Programming Models*

Due to the difficulties in automatic parallelization today, people have to choose a proper parallel programming model to develop their parallel applications for a particular platform.

1) MPI: The Message Passing Interface (MPI) is a language-independent communications protocol used to program parallel computers. Pros of MPI:

- Runs on either shared or distributed memory architectures.
- Can be used on a wider range of problems than OpenMP.
- Each process has its own local variables.
- Distributed memory computers are less expensive than large shared memory computers.

Cons of MPI:

- Requires more programming changes to go from serial to parallel version.
- Can be harder to debug.
- Performance is limited by the communication network between the nodes.

2) OpenMP: OpenMP is a method of parallelization that exploits multithreading and it is mostly used for loop parallelization. Pros of OpenMP:

- Easier to program and debug than MPI.
- Directives can be added incrementally (gradual parallelization).
- Can still run the program as a serial code.
- Serial code statements usually don't need modification.
- Code is easier to understand and maybe more easily maintained.

Cons of OpenMP:

- Can only be executed in shared memory computers.
- Requires a compiler that supports OpenMP.
- Mostly used for loop parallelization.

3) New Parallel Programming Models: C/C++ will probably continue to be the dominant language for high performance programming for the next decade, that is the reason why most vendors are focusing on optimizing and inventing new C++ based parallel programming models.

- Compute Unified Device Architecture (CUDA) is a compiler and a set of development tools created by nVidia that allow programmers to use a C variation in order to code nVidia Graphics Processing Units (GPUs).
- Rapid Mind is a data parallel programming mode, it is exposed as a set of C++ libraries that currently target multi-core x86 processors, GPUs via OpenGL and Cell processors.
- Intel Threading Building Blocks (Intel TBB) is a template based programming library for C++ developed by Intel to facilitate parallel code writing. TBB provides advanced C++ abstraction concepts particularly suited when parallelism is hidden in generic C++ structures like containers and iterators. Task patterns are specified instead of threads and a task scheduler does the mapping to the threads.
- Cilk++ offers a compiler keyword alternative to TBB, every program preserves the serial semantic providing performance guarantees based on a theoretically efficient scheduler. The programmer should be responsible for identifying elements that can be parallely executed and left the decision of how to divide the work between processors to the run-time environment.

Fig. 6. LUSITANIA

- Intel Ct is a C++ based programming model developed by Intel to ease the exploitation of its future multicore chips (Rapid Mind was bought by Intel in 2009 and it is currently integrated into Intel Ct). Once compiled it will generate optimized and native IA code and, without recompiling, it will reoptimize for more cores, more cache, more bandwidth and even more instruction set enhancements.
- Open Computing Language (OpenCL) is a portable intermediate low-level language layer for a wide variety of different hardware (CPUs, GPUs, Floating Point Accelerators (FPAs), etc.), OpenCL provides parallel computing using task-based and data-based parallelism.

III. LUSITANIA/CÉNITS

LUSITANIA is a SMP-ccNUMA system with 2 HP SuperDomes SX2000 nodes installed at Extremadura Supercomputing Center (CénitS) [10] in Cáceres, Spain (Fig. 6). The demanded applications in this Supercomputer are multidisciplinary and heterogeneous so, it was very important to know its configuration to slightly adjust all the parameters of the applications to improve the performance of the system. LUSITANIA was recently used to analyze a massive computational electromagnetics problem, the largest with more than 620 million unknowns [7].

A. Hardware Configuration

LUSITANIA is the Supercomputer of Extremadura (Spain), it has some of the biggest shared-memory nodes of Spain and Europe. The solution is based on two shared-memory HP Integrity SuperDome SX2000 supernodes:

- They both are equipped with 64 dual-core Intel Itanium2 Montvale processors running at 1.6GHz with 18 MB cache, 1TB memory (upgradeable) on a single image and SX2000 chipsets designed to take advantage of Itanium2 Montvale CPUs [8].
- The Itanium architecture is based on explicit ILP, in which the compiler makes the decisions about which instructions will execute in parallel. This alternative approach helps Itanium processors execute up to six instructions per clock cycle.

- SX2000 chipsets are interconnected via crossbar switches with three independent connections to ensure the best performance by using multipathing, ECC protection and load-balancing.
- The well-balanced architecture of the HP Super-Scalable Processor Chipset SX2000 makes the Dual-Core Intel Itanium2 Montvale more powerful.
- The system is also designed to derive significantly greater performance from these existing processors by providing systems with enhanced bandwidth, high memory capacity and reduced memory latency.
- HP SX2000 Chipset is also built to support Intel Itanium processors with multithreading for enhanced performance and scalability [8]. Its VLSI components consist of a cell controller, a memory buffer, a crossbar switch, a PCI-X system bus adapter and a PCI-X host bridge. The chipset enhances interconnectivity between processors, memory and I/O cards, providing you with a high-performance computer system.
- HP Superdome SX2000 consists on 16 cells with interleaved memory for shared objects or data structures. A portion of memory is taken from cells of the system (typically all of the cells) and is mixed together in a round robin fashion of cache-line-size chunks. It has the characteristic that memory accesses take a uniform amount of time. In other words, it has uniform latency no matter which processor accesses it.
- The cell controller (CC) maintains a cache-coherent memory system (ccNUMA) using a directory-based memory controller [9]. The CC's memory controller is combined with memory buffers and DIMMs to create four independent memory systems (quadrants). The memory buffers enable streaming of data between the CC and DIMMs at 533 MT/s. These memory buffers accelerate access to memory, enable memory ECC and can buffer several cache lines going both to and from memory. Up to 32 DIMMs can be controlled by the CC's memory controller [8].

IV. Conclusion

Although modern supercomputers are able to solve the most complex electromagnetism problems never imagined, not all of them are prepared to solve big problems because of their nature and some of them are not able to execute computing algorithms in an efficient manner. Shared memory supercomputers are probably the best choice for electromagnetism problems but this is not an exact science, researchers should test which is the best supercomputer architecture for their problems by running their code in different computing environments. Furthermore, it is important to track and test new parallel programming models, they are meant to exploit all new hardware resources in a more transparent, scalable and efficient way.

References

[1] Ananth Grama, Anshul Gupta, George Karypis and Vipin Kumar. "Introduction to parallel computing". Pearson, 2003. Chapter 1

[2] Julio Ortega, Mancia Anguita and Alberto Prieto. "Computer Architecture". Thomson, 2004. Chapter 1

[3] Kai Hwang and Faye A. Briggs. "Computer Architecture And Parallel Processing". MCGraw-Hill, 1990. Chapter 1

[4] Julio Ortega, Mancia Anguita and Alberto Prieto. "Computer Architecture". Thomson, 2004. Chapter 7

[5] Ian Foster. "What is the Grid? A Three Point Checklist". GridToday, July 2002.

[6] Gruman, Galen. "What cloud computing really means". InfoWorld April 2008.

[7] World Record in Electromagnetics Supercomputing - http://www.cenits.es/index.php/noticias/1-latest-news/78-world-record

[8] Hewlett Packard "HP Super-Scalable Processor Chipset sx2000", 2007.

[9] Hewlett Packard "ccNUMA White Paper", 2003.

[10] CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura) - http://www.cenits.es

César Gómez-Martín received B.Sc., M.Sc. and M.Phil. degrees in Computer Science as well as a Master degree in GRID Computing and parallelism at the University of Extremadura. He is actually researching and working at CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura) and as part-time Adjunct professor at the University of Extremadura.

José-Luis González-Sánchez is a full time Associate Professor of the Computing Systems and Telematics Engineering department at the University of Extremadura, Spain. He received his Engineering degree in Computer Science and his Ph.D degree in Computer Science (2001) at the Polytechnic University of Cataluña, Barcelona, Spain. He has worked, for years, at several private and public organizations, accomplishing functions of System and Network Manager. He has published many articles and books and directed research projects related to computing and networking. He was the main researcher of the Advanced and Applied Communications Engineering Research Group (GÍTACA) of the University of Extremadura and, currently, he is the general manager of the Foundation COMPUTAEX and the Center CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura).

Javier Corral-García received B.Sc. and M.Sc. degrees in Computer Science at the University of Extremadura as well as Ph.D. student at the same University. He is also a legal expert at the Computer Science Association of Extremadura and a supercomputing analyst at CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura).

Ángel Bejarano-Borrega received his M.Sc. degree in Computer Science at the University of Extremadura. He is system administrator at CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura). He has been working as a system engineer at COMPAREX Spain and as a Java programmer for the TESEO project at the University of Extremadura.

Javier Lázaro-Jareño received his M.Sc. degree in Computer Science at the University of Extremadura. He has worked as system administration at the University of Extremadura and now he works as web and system administrator at CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura).