Modélisation Multi-échelles : de l'Electromagnétisme à la Grille

Fadi Khalil

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'INPT - ENSEEIHT (Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Telecommunications)*
Discipline ou spécialité : *Micro-ondes Electro-Magnétisme Opto-électronique (MEMO)*

Présentée et soutenue par *Fadi KHALIL*
Le *14 Décembre 2009*

Titre : *Modélisation Multi-échelles : de l'Electromagnétisme à la Grille*
(Multi-scale Modeling: from Electromagnetism to Grid)

JURY
M. Hervé Aubert, M. Fabio Coccetti
M. Renaud Loison, M. Christian Perez
M. Michel Daydé, Président
M. Yves denneulin, examinateur
M. Thierry Monteil, examinateur
M. Luciano Tarricone, invité
M. Petr Lorenz, invité

Ecole doctorale : *Génie Electrique, Electronique, Télécommunications (GEET)*
Unité de recherche : *Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS -- CNRS)*
Directeur(s) de Thèse : *M. Hervé Aubert et M. Fabio Coccetti*
Rapporteurs : *M. Renaud Loison et M. Christian Perez*

UNIVERSITY OF TOULOUSE

Doctoral School GEET

GENIE ELECTRIQUE ELECTRONQIUE TELECOMMUNICATIONS

P H D T H E S I S

to obtain the title of

PhD of Science

of INPT - ENSEEIHT

Specialty : MicroOndes, ElectroMagnétisme et Optoélectronique (MEMO)

Defended on December 14, 2009 by

Fadi KHALIL

Multi-scale Modeling: from Electromagnetism to Grid

Thesis Advisors: Hervé Aubert and Fabio Coccetti

prepared at the Laboratory of Analysis and Architecture of Systems (LAAS – CNRS, UPR 8001), MINC Team

Jury :

Reviewers :	Renaud Loison	-	IETR-INSA Rennes
	Christian Perez	-	LIP-ENS Lyon
Members :	Hervé Aubert	-	University of Toulouse, LAAS, INPT-ENSEEIHT
	Fabio Coccetti	-	University of Toulouse, Novamems, LAAS
	Michel Daydé	-	University of Toulouse, IRIT, INPT-ENSEEIHT
	Yves Denneulin	-	LIG-ENSIMAG Monbonnot
	Thierry Monteil	-	University of Toulouse, INSA, LAAS, IRIT
Invited :	Petr Lorenz	-	Lorenz Solutions
	Luciano Tarricone	-	University of Lecce

Acknowledgments

First and foremost, to my thesis advisors, Prof. Hervé Aubert, and Dr. Fabio Coccetti, for supporting this research and for providing an excellent working environment, for dedicated help, inspiration and encouragement throughout my PhD, for providing sound advice and lots of good ideas, and for good company within and outside the laboratory.

My appreciation goes to my other committee members as well: Prof. Renaud Loison, and Prof. Christian Perez, for having accepted to examine this work and for having provided valuable insights and contributed to the improvement of the quality of this thesis. Thanks are also due to Prof. Michel Daydé for chairing my thesis committee, Prof. Yves Denneulin and Prof. Thierry Monteil.

I appreciate very much the presence, as member of committee, of Prof. Luciano Tarricone and Dr. Petr Lorenz.

My gratitude to Prof. Robert Plana for all of his guidance, assistance, and subtle sense of humor.

I would like to acknowledge the National Research Agency (ANR) for support of MEG Project (ANR-06-BLAN-0006, 2006-2009), and the collaboration of Carlos-Jaime Barrios-Hernandez, and Luis Melo from LIG Laboratory - ENSIMAG.

I firmly believe that the work environment makes the greater part of the learning experience and for this I would like to thank my colleagues in the Laboratory of Architecture and Analysis of Systems. Thank you especially to Bernard Miegemolle, Rémi Sharrock, and Tom Guérout from MRS research group.

My thanks also to Aamir Rashid and Euloge Budet Tchikaya for having provided me with the Scale-Changing Technique modeling codes used in this thesis on Grid platform. I have had three office mates, Jinyu (jason) Ruan, Badreddine Ouagague and Ali Kara Omar, all of them have freely shared their time, opinions and expertise.

In addition to my office mates, I would like to extend my gratitude to all MINC research group members for their generous company.

I am grateful to the secretary Mrs. Brigitte Ducroq for helping the lab to run smoothly and for assisting me in many different ways.

Out with the work setting, I would like to offer my fondest regards to my friends: Phélomène Makhraz, Youssef El Rayess, Dalal Boutros, Joseph Chemaly, Georges Khalil, Wissam Karam, Serge Karboyan, Issam Tawk, Florence Freyss, Nancy Nehme, Rania Azar, Micheline Abbas and Hikmat Achkar.

Finally, I would like to mention my family. I wish to thank my parents who raised me, supported me, taught me, and loved me. Many Thanks to my love Nadine Makhraz. To them all I dedicate this thesis.

Multi-scale Modeling: from Electromagnetism to Grid

Abstract: The numerical electromagnetic tools for complex structures simulation, i.e. multi-scale, are often limited by available computation resources. Nowadays, Grid computing has emerged as an important new field, based on shared distributed computing resources of Universities and laboratories.

Using these shared resources, this study is focusing on grid computing potential for electromagnetic simulation of multi-scale structure. Since the numerical simulations tools codes are not initially written for distributed environment, the first step consists to adapt and deploy them in Grid computing environment. A performance study is then realized in order to evaluate the efficiency of execution on the test-bed infrastructure.

New approaches for distributing the electromagnetic computations on the grid are presented and validated. These approaches allow a very remarkable simulation time reduction for multi-scale structures and friendly-user interfaces.

Keywords: computational electromagnetics, Transmission Line Matrix (TLM), Scale Changing Technique (SCT), Grid computing, distributed computing, performance

Modélisation Multi-échelles : de l'électromagnétisme à la Grille

Résumé: Les performances des outils numériques de simulation électromagnétique de structures complexes, i.e., échelles multiples, sont souvent limitées par les ressources informatiques disponibles. De nombreux méso-centres, fermes et grilles de calcul, se créent actuellement sur les campus universitaires.

Utilisant ces ressources informatiques mutualisées, ce travail de thèse s'attache à évaluer les potentialités du concept de grille de calcul (Grid Computing) pour la simulation électromagnétique de structures multi-échelles. Les outils numériques de simulation électromagnétique n'étant pas conçus pour être utilisés dans un environnement distribué, la première étape consistait donc à les modifier afin de les déployer sur une grille de calcul. Une analyse approfondie a ensuite été menée pour évaluer les performances des outils de simulation ainsi déployés sur l'infrastructure informatique.

Des nouvelles approches pour le calcul électromagnétique distribué avec ces outils sont présentées et validées. En particulier, ces approches permettent la réalisation de simulation électromagnétique de structures à échelles multiples en un temps record et avec une souplesse d'utilisation.

Mots Clés: modélisation électromagnétique, modélisation par lignes de transmission (TLM), modélisation par changements d'échelles (SCT), grille de calcul, calcul distribué, performance

Contents

Chapter 1

Introduction

Contents

1.1 Numerical Techniques in CEM

Modern microwave and radio frequency (RF) engineering is an exciting and dynamic field, due in large part to the symbiosis between recent advances in modern electronic device technology and the current explosion in demand for voice, data, and video communication capacity. Prior to this revolution in communications, microwave technology was the nearly exclusive domain of the defense industry; the recent and dramatic increase in demand for communication systems for such applications as wireless paging, mobile telephony, broadcast video, and computer networks is revolutionizing the industry. These systems are being employed across a broad range of environments including corporate offices, industrial and manufacturing facilities, and infrastructure for municipalities, as well as private homes.

Electromagnetic analysis, a discipline whereby one solves Maxwell's equations [1] - [3] to obtain better understanding of a complex system, is a critical part of the microwave design cycle. One reason is that Maxwell's theory is essential for the manipulation of electricity and hence is indispensable. Another reason is that Maxwell's theory has proven to have strong predictive power. This strong predictive power, together with the advent of computer technology, has changed the practice of electrical engineering in recent years. A complete solution to Maxwell's equations can expedite many electrical engineering design properties.

Electromagnetic analysis methods can be classified by analytical, semi-analytical and numerical methods. Closed-form solutions in terms of analytical functions can only be found for a few special geometries (for example in rectangular, elliptical, and spherical waveguides and resonators). In spite of their limited practical applicability, analytical solutions are extremely useful for the purpose of validating numerical methods since they provide error-free reference solutions.

Semi-analytical methods were developed before the advent of powerful computers. They involve extensive analytical processing of a field problem resulting in a

complicated integral, an infinite series, a variational formula, an asymptotic approximation, in short, an expression that requires a final computational treatment to yield a quantitative solution. The analytical preprocessing often leads to rather fast and efficient computer algorithms, but the resulting programs are necessarily specialized since specific types of boundary and material conditions have been incorporated in the formulation.

Several real-world electromagnetic problems like scattering, radiation, waveguiding etc, are not analytically calculable, for the multitude of irregular geometries designed and used. The inability to derive closed form solutions of Maxwell's equations under various constitutive relations of media, and boundary conditions, is overcome by computational numerical techniques. Numerical methods transform the continuous integral or differential equations of Maxwell into an approximate discrete formulation that requires either the inversion of a large matrix or an iterative procedure. There exist many ways to discretize an electromagnetic problem, ranging from very problem- specific to very general purpose approaches.

This makes computational electromagnetics (CEM), an important field in the design, and modeling of antenna, radar, satellite and other such communication systems, nanophotonic devices and high speed silicon electronics, medical imaging, cell-phone antenna design, among other applications (see Figure 1.1).

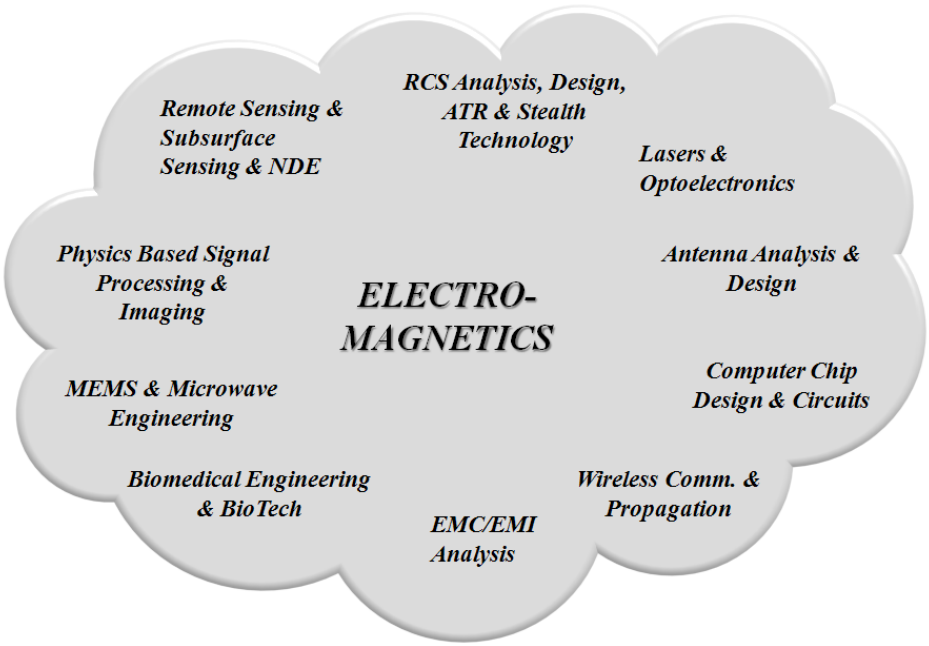


Figure 1.1: Impact of Electromagnetics.

Computer-based analysis is at the core of modern simulation tools, and it has revolutionized engineering design, even more so in microwave engineering where these tools allow us to "see" the electromagnetic field and its effects such as current and charge distributions. It reflects the general trend in science and engineering to

formulate laws of nature as computer algorithms and to simulate physical processes on digitals computers.

Classification

The purpose of all numerical methods in electromagnetics is to find approximate solutions to Maxwell's equations (or of equations derived from them) that satisfy given boundary and initial conditions, formulating an electromagnetic problem amounts to specifying the properties that a solution must have in order to qualify. These properties can be specified as local (differential) or global (integral) properties, both in the field space and its boundaries.

When discussing the properties of the different methods, it is necessary to classify them. A major point of difference is the domain they are working in, which is either time domain (TD) or frequency domain (FD). The perceived differences between these two categories are better captured by the terms time-harmonic and transient methods. However, in the formal sense, frequency domain formulations are time domain formulations in which the time dimension has been subject to a Fourier transform, thus reducing the number of independent variables by one. Expressed in a simplistic way, frequency domain formulations are obtained by replacing the time differential operator d/dt by jw, and the time integration operator by $-j/w$, thus effectively transforming a time differentiation into a multiplication, and a time integration into a division by jw.

Another way of categorizing both the numerical techniques and the computer tools based on them relies on the number of independent space variables upon which the field and source functions depend. In all categories we can again distinguish between frequency domain and time domain formulations.

- 1D Methods: These are methods for solving problems where the field and source functions depend on one space dimension only. Typical applications are transmission line problems, uniform plane wave propagation, and spherically or cylindrically symmetrical problems with only radial dependence. Transmission-line circuit solvers and the SPICE program are well-known examples of 1D solvers.

- 2D Methods: These are methods for solving problems where the field and source functions depend on two space dimensions. Typical applications are crosssection problems in transmission lines and waveguides, TEn0 propagation in rectangular waveguide structures, coaxial TEM problems, and spherical problems depending only on radius and azimuth or radius and elevation.

- 2.5D Methods: These are methods for solving problems where the fields depend on three space dimensions, while their sources (the currents) are mainly confined planes with two space dimensions. Typical examples are planar structures such as microstrip circuits, co-planar circuits, patch antennas, and general multilayer structures that contain planar conductor pattern. The pre-

dominant solution method for such structures is the method of moments in the space and spectral domains; however, the method of lines is also suitable for planar and quasi-planar structures.

- 3D Methods: These are methods for solving problems where both the field and source functions depend on three space dimensions. This category comprises all volumetric full-wave general-purpose formulations.

Hybrid formulations combining two or more different numerical techniques have also been developed and implemented for particular applications.

Figure 1.2 shows main numerical modeling methods with respect to discretization category, distinguishing between volumetric discretization methods and surface discretization methods.

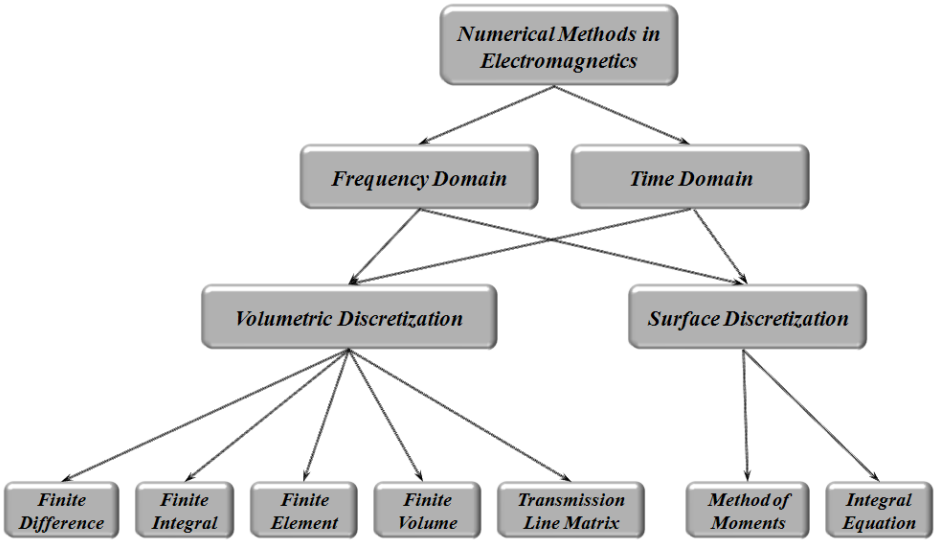


Figure 1.2: Classification of main numerical methods in electromagnetics.

As we can see, there is a vast number of numerical methods one can use to solve the electromagnetic problem. However, not every method is suitable for a particular problem.

1.2 Objectives and Contribution presented in this Thesis

The area of CEM has undergone a vast development in the past years - giving rise to powerful electromagnetic simulation tools and techniques. The goal of all these techniques is to determine and predict performances of RF and microwave components, networks and machines in a very accurate and reliable way.

Recent advances in wireless and microwave communication systems (higher operation frequency bands, more compact topologies containing MMICs and MEMS)

have increased the necessity of fast and accurate numerical simulation techniques.

The advent of powerful computers and computer systems allows us to analyze some large and complicated structures, with the numerical efficiency and accuracy remaining at the focal point of today's research interests. The amount of addressable memory for 64-bit machines goes far beyond the memory addressing limit of 32-bit machines. In theory, the emergence of the 64-bit architecture effectively increases the memory ceiling to 264 addresses, equivalent to approximately 17.2 billion gigabytes.

However, this addressing capability is a theoretical number. The support of the memory addressing is still limited by the bandwidth of the bus, the operating system and other hardware and software issues. Most 64-bit microprocessors on the market today have an artificial limit on the amount of memory they can address, because physical constraints make it impossible to support the full 16.8 million terabyte capacity.

Most of the modern full-wave CEM techniques require discretization of the whole computational domain. In order to achieve numerically efficient computations with the available computer memory, one must define a finite computational domain enclosing the problem under analysis and rely on the ways to terminate the computational domain at the boundaries.

Despite the recent hardware progress, the computational resources of today's computers seems to be very limited. It is often impossible to model large volumes of space and/or multi-scale structures with high aspect ratio. The limits up to which we are able to solve a problem are set by:

- memory requirements,
- time needed to process the information.

To get a better feeling for what are the memory requirements to solve an electromagnetic problem, let us take a look at the following example. There is given a structure assumed to be perfect electric conductor (PEC) in air and we wish to compute the scattered electromagnetic field. A three-dimensional computational domain of $2000 \times 2000 \times 2000$ TLM cells requires 6144000000000 bits [1] to hold the information about the electromagnetic field. This amount of information means 715.25 GBytes of memory.

Increasing the spatial resolution just twice in every direction results in eight times larger memory requirements. Furthermore, the time needed to process this amount of information is inverse proportional to the speed of information processing.

The need for solving the CEM problem for ever increasing meshes leads to the idea of running these applications in Grid environments. The example shows the reasons why we are not able to discretize very large structures. Consequently, to overcome these problems, large scale computer systems, as Grid Computing, must be used.

[1] For the modeling of free-space we need 12 variables per TLM cell. We assume a TLM variable to be represented by 64 bits (double precision). For more information, please refer to Chapter 3.

Grid computing is currently the subject of a lot of research activities world-wide. Most of these activities aim at providing the necessary tools (for dealing with aspects such as administration, security, performance simulation, discovery, scheduling and volatility of resources, etc.) and programming methodologies. Some studies are centred on applications.

In this thesis, a new methodology is presented for conducting complex multi-scale numerical electromagnetic simulations. Technologies from several scientific disciplines, including computational electromagnetics, and parallel computing, are combined to form a simulation capability that is both versatile and practical.

In the process of creating this capability, work is accomplished to conduct first a study designed to adapt the electromagnetic solvers to distributed computing, and to provide an assessment of the applicability of Grid Computing to the field of computational electromagnetics.

Two modeling methods which can be useful to be coupled in a hybrid method have been chosen: the Transmission Line Matrix (TLM) modeling method (3D) and the Scale Changing Technique (SCT) modeling method (2.5D). The efficiency of the modern CEM software to analyze complex microwave structures in Grid Computing environments are investigated with real life examples. Multiscale strutures, i.e. planar reflectarrays, have been simulated and the radiation patterns plotted. After a performance study, well adapted approaches are proposed and tested on Grid nodes.

In order to keep the use of Grid Computing transparent to electronic engineer that are not probably computer science specialists, friendly solutions as graphical user interfaces (GUI) have been used.

1.3 Organization of the Thesis

The thesis is organized as follows. Chapter 2 introduces the Grid Computing and distributed systems, and a brief overview of Grid applications. Grid'5000, the test-bed platform used to run all the experiments in this thesis, is also presented in a way to show underlying hardware, network and software. This chapter is then concluded by the definition of essential terms used the following chapters.

In Chapter 3, the Transmission Line Matrix (TLM) method is described. The principles of the method are shown as well as the interconnection of TLM nodes and their contributions in the algorithm. A quick overview of past works on parallel TLM shows different approaches that have been used. Then, both proposed approaches (distributed parallel and parametric TLM computing) are investigated and perfomance evaluated. Simulations of a coplanar phase-shifter based on MEMS and reflectarrays are shown.

Chapter 4 is devoted to the Scale-Changing Technique (SCT) modeling method. The distribution of the independant tasks are described in detail. Reflectarrays are simulated in parallel on Grid'5000 nodes. Parametric analysis, for frequency sweep and convergence, shows the efficiency of using large scale Grids.

Finally, the conclusions of the thesis are presented in Chapter 5. Appendix A and Appendix B describes two electromagnetic tools based on TLM and used in this thesis. The theory used in this thesis to design the array antennas with steered beams is presented in Appendix C. Appendix D lists different acronyms used.

Chapter 2

Grid Computing

Contents

This chapter provides an overview of Grid Computing, its different application domains and some related terminology. It presents also the testbed platform, Grid'5000, used along this thesis to run computational electromagnetics experiments.

2.1 What is the Grid?

The popularity of the Internet as well as the availability of powerful computers and high-speed network technologies as low-cost commodity components is changing the way we use computers today. These technology opportunities have led to the possibility of using distributed computers (*nodes*) as a single, unified computing resource, leading to what is popularly known as Grid computing (GC) [4] - [9]. Grid Computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing [1], innovative applications, and, in some cases, high-performance orientation.

The term "the Grid" was coined in the mid of 1999s [2] to denote a proposed distributed computing infrastructure for advanced science and engineering. In 1997,

[1] In Grid terms, a resource is any kind of software (a piece of application, a file, a database, etc.) or hardware entity (an electrical device, a storage card, etc.) accessible through the network.

[2] The name "Grid Computing" is inspired by the electrical grid.

CPU scavenging and volunteer computing were popularized by distributed.net [10] and later in 1999 by SETI@home [11] to harness the power of networked PCs worldwide, in order to solve CPU-intensive research problems.

The ideas of the grid (including those from distributed computing, object-oriented programming and Web services) were brought together by Ian Foster, Carl Kesselman, and Steve Tuecke, widely regarded as the "fathers of the grid" [5] - [9]. They led the effort to create the Globus Toolkit [3] incorporating not just computation management but also storage management, security provisioning, data movement, monitoring, and a toolkit for developing additional services based on the same infrastructure, including agreement negotiation, notification mechanisms, trigger services, and information aggregation [12]. While the Globus Toolkit remains the de facto standard for building Grid solutions, a number of other tools have been built that answer some subset of services needed to create an enterprise or global Grid.

Computational Grids may span domains of different dimensions, starting from local Grids, where the nodes belong to a single organization via a LAN connection, to global Grids, where the nodes are owned by different organizations and linked via Internet.

Grid applications (typically multidisciplinary and large-scale processing applications) often couple resources that cannot be replicated at a single site, or which may be globally located for other practical reasons, and belonging to multiple individuals or organizations (known as multiple administrative domains), in a flexible and secured environment. These are some of the driving forces behind the foundation of global Grids. In this light, the Grid allows users to solve larger or new problems by pooling together resources that could not be easily coupled before. Hence, the Grid is not only a computing infrastructure, for large applications, it is a technology that can bond and unify remote and diverse distributed resources ranging from meteorological sensors to data vaults, and from parallel supercomputers to personal digital organizers.

A Grid could be characterized by four main aspects [13]:

- ***Multiple administrative domains and autonomy***. Grid resources are geographically distributed across multiple administrative domains and owned by different organizations. The autonomy of resource owners needs to be honored along with their local resource management and usage policies.
- ***Heterogeneity***. A Grid involves a multiplicity of resources that are heterogeneous in nature and will encompass a vast range of technologies.
- ***Scalability***. A Grid might grow from a few integrated resources to millions. This raises the problem of potential performance degradation as the size of Grids increases. Consequently, applications that require a large number of geographically located resources must be designed to be latency and bandwidth tolerant.

[3] Globus Toolkit. http://www.globus.org

- ***Dynamicity*** or ***adaptability***. In a Grid, resource failure is the rule rather than the exception. In fact, with so many resources in a Grid, the probability of some resource failing is high. Resource managers or applications must tailor their behavior dynamically and use the available resources and services efficiently and effectively.

Over the last decade, an increasing number of scientists have run their workloads on large-scale distributed computing systems such as Grids. The concept of Grid computing started as a project to link geographically dispersed supercomputers, but now it has grown far beyond its original intent. The Grid infrastructure can benefit many applications, including among others collaborative engineering, data exploration, high-throughput computing, and distributed supercomputing.

2.2 Grids Projects and Applications Area

There are currently a large number of international projects and a diverse range of new and emerging Grid developmental approaches being pursued worldwide. These systems range from Grid frameworks to application testbeds, and from collaborative environments to batch submission mechanisms (integrated Grid systems, core middleware, user-level middleware, and applications/application driven efforts ...) [14, 15].

Grid resources can be used to solve grand challenge problems in areas such as biophysics, chemistry, biology, scientific instrumentation [16], drug design [17, 18], tomography [19], high energy physics [20], data mining, financial analysis, nuclear simulations, material science, chemical engineering, environmental studies, climate modeling [21], weather prediction, molecular biology, neuroscience/brain activity analysis [22], structural analysis, mechanical CAD/CAM, and astrophysics.

In the following, two applications of interest to NASA [23] are illustrated. Key aspects of the design of a complete aircraft (airframe, wing, stabilizer, engine, landing gear and human factors) are depicted in Figure 2.1.

Each part could be the responsibility of a distinct, possibly geographically distributed, engineering team whose work is integrated together by a Grid realizing the concept of concurrent engineering. Figure 2.2 depicts possible Grid controlling satellites and the data streaming from them. Shown are a set of Web (OGSA [25]) services for satellite control, data acquisition, analysis, visualization and linkage (assimilation) with simulations as well as two of the Web services broken up into multiple constituent services.

Meanwhile, the community of electromagnetics (EM) research has been only peripherally interested in Grid Computing (GC) until 2004 [26] [4]. Few works in electromagnetics could be found.

In 2005, Cardiff University and the University of Wales, Swansea, have teamed with Hewlett-Packard, BAE SYSTEMS and the Institute of High Performance Com-

[4] A practical and comprehensive guidance needed to use the Grid.

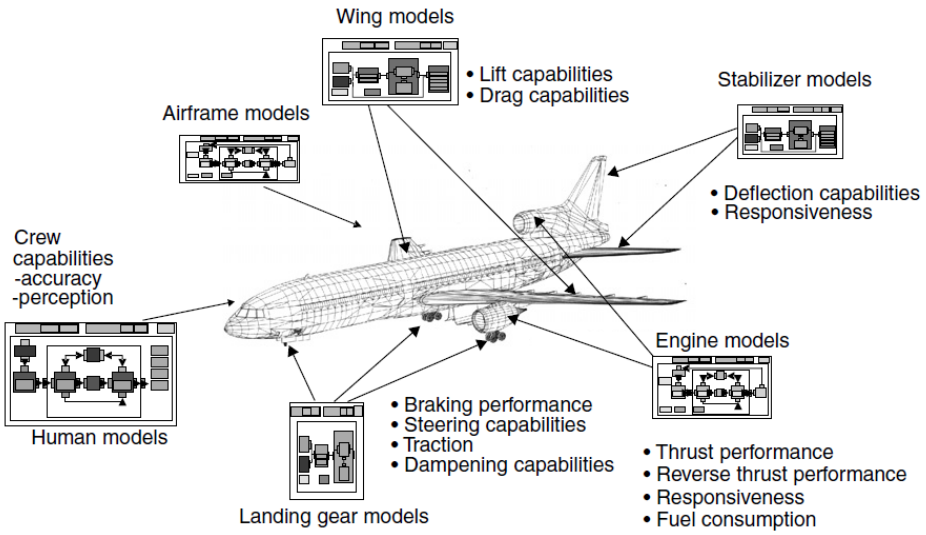


Figure 2.1: A Grid for aerospace engineering showing linkage of geographically separated subsystems needed by an aircraft [24].

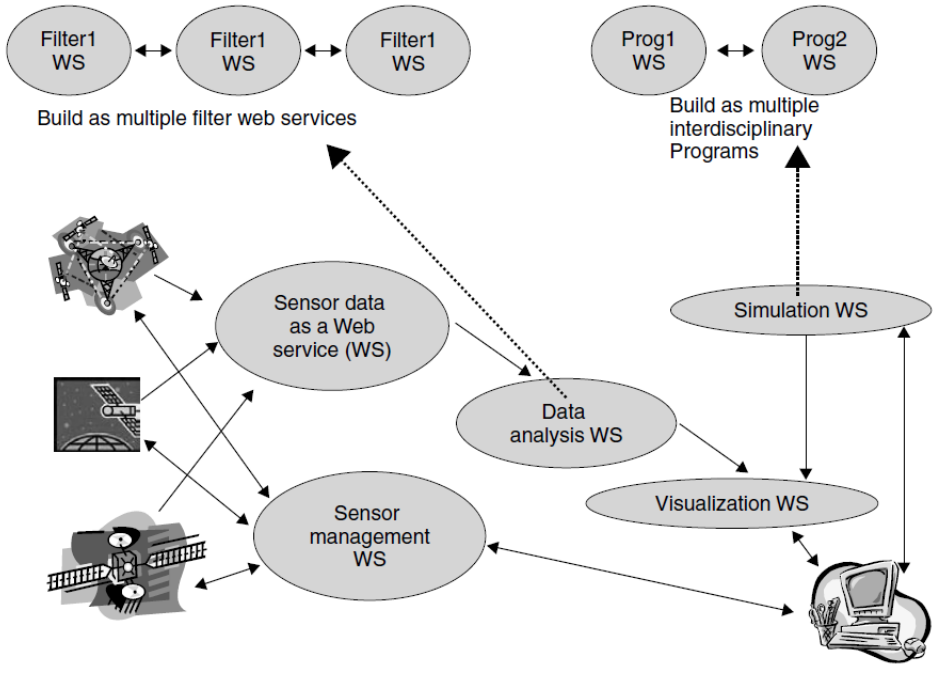


Figure 2.2: A possible Grid for satellite operation showing both spacecraft operation and data analysis. The system is built from Web services (WS) and it shows how data analysis and simulation services are composed from smaller WS's [24].

puting in Singapore, to use Grid computing for the exploration of advanced, collaborative simulation and visualization in aerospace and defense design [27].

One year after, Lumerical launches parallel FDTD solutions on the supercomputing infrastructure of WestGrid in Canada [28], even though their field of interest is more in optical and quantum devices rather than in RF applications.

Another recent example of distributed computing using a code based on the TLM (Transmission Line Matrix, defined in chapter 3) method has been demonstrated in Germany [29].

In 2006, computer scientists from different french labs started the DiscoGrid project [30]. It aims at studying and promoting a new paradigm for programming non-embarrassingly parallel scientific computing applications on distributed, heterogeneous, computing platforms. The target applications require the numerical resolution of systems of partial differential equations (PDEs) for computational electromagnetism (CEM) [31] and computational fluid dynamics (CFD).

In [32], EM researchers can identify new and promising Information and Communication Technologies (ICT) tools (already available, or to be consolidated in the immediate future), expected to significantly improve their daily EM investigation.

Later, a Web-based, Grid-enabled environment for wideband code-division multiple-access (WCDMA) system simulations, based on Monte Carlo methods, has been implemented and evaluated on the production Grid infrastructure deployed by the Enabling Grids for E-sciencE (EGEE) project [33].

2.3 Grid'5000

2.3.1 Testbed Description

Grid'5000 [34] is a research effort developing a large scale nationwide infrastructure for large scale parallel and distributed computing research [5]. It aims at providing a highly reconfigurable, controllable and monitorable experimental platform to its users [35]. The initial aim (circa 2003) was to reach 5000 processors in the platform. It has been reframed at 5000 cores, and was reached during winter 2008-2009. For the 2008-2012 period, engineers of ADT ALADDIN-G5K initiative are ensuring the development and day to day support of the infrastructure.

The infrastructure of Grid'5000, which is a Multi-clusters Grid, is geographically distributed on different sites hosting the instrument, initially 9 in France (17 Laboratories). The current plans are to extend from the 9 initial sites each with 100 to a thousand PCs, connected with a 10Gb/s link by the RENATER (v5) Education and Research Network [36] to a bigger platform including a few sites outside France not necessarily connected through a dedicated network connection. Porto Alegre, in Brazil, is now officially becoming the 10th site and Luxembourg should join shortly.

Figure 2.3 shows the Grid'5000 backbone connecting the nine sites of this Grid (Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia and Toulouse).

[5] Other testbeds for experiments: PLANETLAB (http://www.planet-lab.org/), Emulab (http://www.emulab.net/), DAS-3 (http://www.starplane.org/das3/), GENIE (http://www.genie.ac.uk/)

Each site is composed of a heterogeneous set of nodes and local networks (High performance networks: Infiniband 10G (161 cards), Myrinet 2000 (222 cards), Myrinet 10G (423 cards), Gigabit Ethernet 1 or 10 Gb/s). The "standard" architecture is based on 10Gbit/s dark fibers and provides IP transit connectivity, interconnection with GEANT-2 [37], overseas territories and the SFINX (Global Internet exchange).

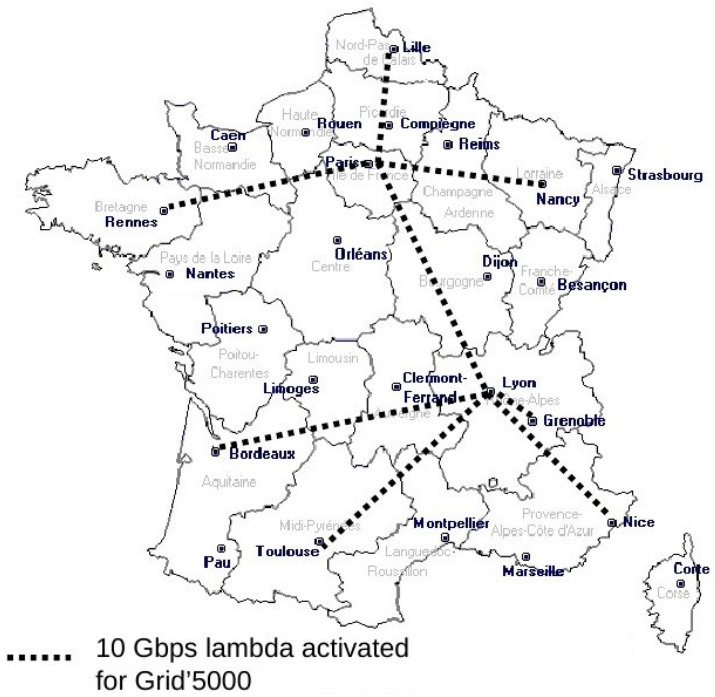


Figure 2.3: Grid'5000 Backbone.

Grid'5000 clusters have been built mainly upon 64 bits bi-processors architectures (AMD Opteron (78%) and Intel Xeon EMT64 (22%)), but a certain degree of (desired) heterogeneity appears from cluster to cluster [6]. The user of this architecture can choose the CPU family in order to run its own experiments, which could be MonoCore (41%), DualCore (46%), QuadCore (13%). All computing nodes have local disks (generally 80 gigabytes IDE or SATA) that are all partitioned the same way (for all clusters at all sites). The frontends (Figure 2.5) generally feature huge storage space (0.5 or 1 terabyte) mainly for the home directories (note that these homes are in general not backuped nor synchronized from cluster to cluster).

[6] More informations on: https://www.grid5000.fr/mediawiki/index.php/Special:G5KHardware

2.3.2 Grid'5000 Experimental Activities

This platform is used for large research applicability Grid experiments [7] (see Figure 2.4) to address critical issues of Grid system/middleware (Programming, scalability, fault tolerance, scheduling), of Grid networking (High performance transport protocols, QoS), to gridify and test real life applications and investigate original mechanisms (P2P resources discovery, desktop Grids).

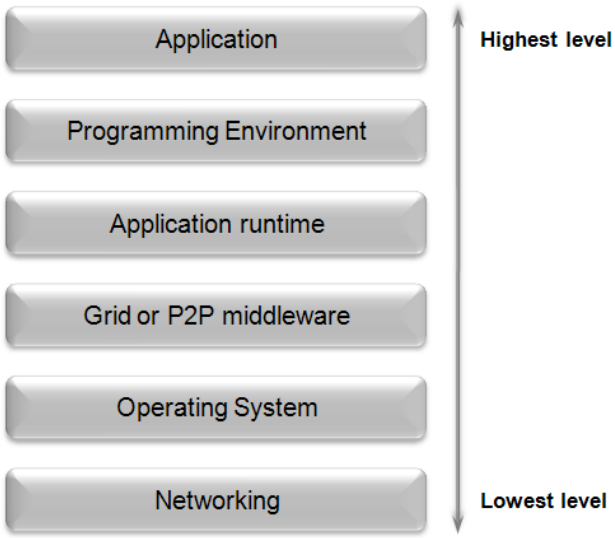


Figure 2.4: Grid'5000 Applications.

2.3.3 Cluster Definition

All computers from a given site (a geographical place where a set of computing resources shares the same administration policy), which are connected to a given network architecture, define a cluster. Every cluster basically consists of:

- one (or more) frontend machine(s), that serve(s) as resource allocation platform(s), so that each user may request usage of whole or part of the other nodes;
  - some of these frontends are accessible from the public internet domain, generally through a firewall, depending on local security policies;
- the computing nodes themselves that form the main cluster, the main computing power; each node may have several CPUs (and each CPU possibly several cores); depending on the resource manager middleware configuration, it is possible to request resource allocation at the node level or at the CPU level;

[7] A summary of the domains of experiment which Grid'5000 is providing a research platform for can be found on https://www.grid5000.fr/mediawiki/index.php/Grid5000:Experiments.

- for security and/or redundancy reasons, there may be servers besides the frontends to isolate critical software and services;
  - in principle, servers and frontends do not take part to the computing power of the cluster and therefore are not counted as computing nodes in the cluster hardware description;
- the network architecture onto which all nodes, frontends and servers are physically connected; it is in principle isolated from the public internet domain (private IP addresses), but is interlinked with the other sites through a privileged RENATER network.

2.3.4 Software and Middleware

The middleware is the software suite that performs the main tasks during computing nodes usage. It mainly consists of a resource allocation manager involving:

- node(s) reservation for a given user and a given duration (with possible environment customization);
- task(s) scheduling over reserved nodes (with possible results retrieval);
- resource deallocation after tasks completed to make them available again for new tasks.

In the case of Grid'5000 [8], job scheduling and resource allocation and deallocation is performed by OAR [38] at the cluster level.

Grid'5000 has also made the choice to allow users to customize their environment and even install their own preferred operating system on the reserved nodes. This implies some preliminary work to build this kind of customized environment. It is generally a hard work, but allows for optimum experiment conditions and reproductability (as long as the hardware does not change).

However, a default Linux environment is available in case of experiments that do not need customization. The KaTools (kadeploy, karun, kaenvironments, karecordenv) [39] have been designed in this purpose of custom environment deployment.

2.3.5 Grid View

The Grid architecture for Grid'5000 is achieved by the privileged interlinks existing between the different clusters, using the local network architecture for clusters at a given site, or provided by RENATER from site to site.

The Grid middleware layer consists of tools able to request resource allocation on several clusters at the same time, using a simple synchronization mechanism to associate what are basically separate resources. The privileged links between sites

[8] Software mainly developed in Grid'5000 and available for its users: https://www.grid5000.fr/mediawiki/index.php/Grid5000:Software

formerly used 10 gigabit optic fibers technology (RENATER v5 network) dedicated only to the Grid'5000 traffic.

To briefly summarize the common general security policy overall Grid'5000 network, it is allowed to perform SSH connections inbound frontends (and not elsewhere) and outbound nowhere. In other words, intrusions are allowed through narrow one-way windows, but the huge computing power from Grid'5000 may not be used to attack any server on the public internet side (also known as the outside)... Local policies may differ about the inbound connections from the outside. Internal connections from one site to another are completely free, either inbound or outbound nodes as well as frontends.

The main internal network services (DNS directories for the machines, LDAP directories for the users' accounts...) are common and distributed among all sites across this dedicated internal network (private IP addresses). External network services (WiKi website, email lists manager...) are centralized on dedicated hosts belonging to the outside (public IP addresses).

2.3.6 Typical use case

An experiment could be resumed in following six steps:

1. Connect to the platform on a site
2. Reserve some resources
3. Configure the resources (optional)
4. Run experiment
5. Grab the results
6. Free the resources

A Grid'5000 Experiment Workflow is sketched in Figure 2.6. In fact, when entering a site, user is connected to its access frontend (Figure 2.5). Thus he is not on its submission frontend. Some sites possess only one frontend to do external access and job submission, but this is not the case everywhere and other sites possess one frontend for external access and another one for job submission.

On each Grid'5000 site, user possesses a Grid-independent home directory. Thus before submitting jobs over the Grid, user must be sure that code and configuration are available on each used cluster. SSH publickey and configuration must be synchronized.

OAR [38] is the resource manager (or batch scheduler for large clusters) used to manage Grid'5000 platform's resources which support KADEPLOY [39]. It allows cluster users to submit or reserve nodes either in an interactive or a batch mode.

OAR is an open source batch scheduler which provides a simple and flexible exploitation of a cluster. It manages resources of clusters as a traditional batch

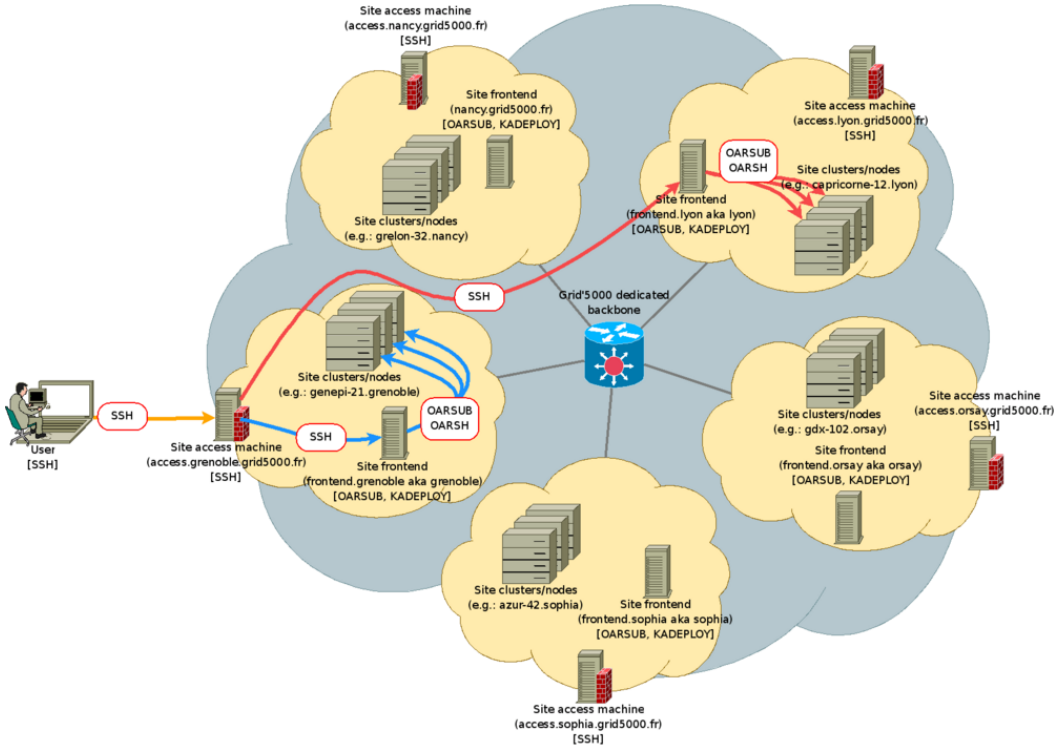


Figure 2.5: Grid'5000 access and job submissions.

scheduler [9]. It is flexible enough to be suitable for production clusters and research experiments. It currently manages all the Grid'5000 nodes and has executed millions of jobs. When making a reservation, one specifies which nodes to use. Nodes can be chosen by the user, or automatically selected.

Note that for Grid experiment, a really simple tool named OARGRID was built upon OAR to help you using distant resources. OARGRID is a Grid version of OAR. It provides the capability of globally reserving nodes on the whole Grid'5000 platform (several sites at once). *Oargridsub* perform an OAR reservation on each specified clusters. An identity number is given by OARGRID, so with it all reservation information could be found. It returns a Grid job id to bind cluster jobs together if the inherent cluster reservations succeed. If one of them did not succeed then all previous reservations are canceled and also the global operation.

The reservations and submissions could be monitored by graphical tools Current and scheduled jobs could be displayed by Monika [40] or by *oarstat*, a command-line tool to view current or planned job submission (running or waiting). *oarnodes* is also a command-line tool. It shows cluster node properties (list states and informations of all nodes). Among returned information there is current node state. This state

[9] as PBS (http://www.pbsgridworks.com/), Torque (http://www.clusterresources.com/products/torque-resource-manager.php), LSF (http://www.platform.com/products/LSF/), SGE (http://gridengine.sunsource.net/)

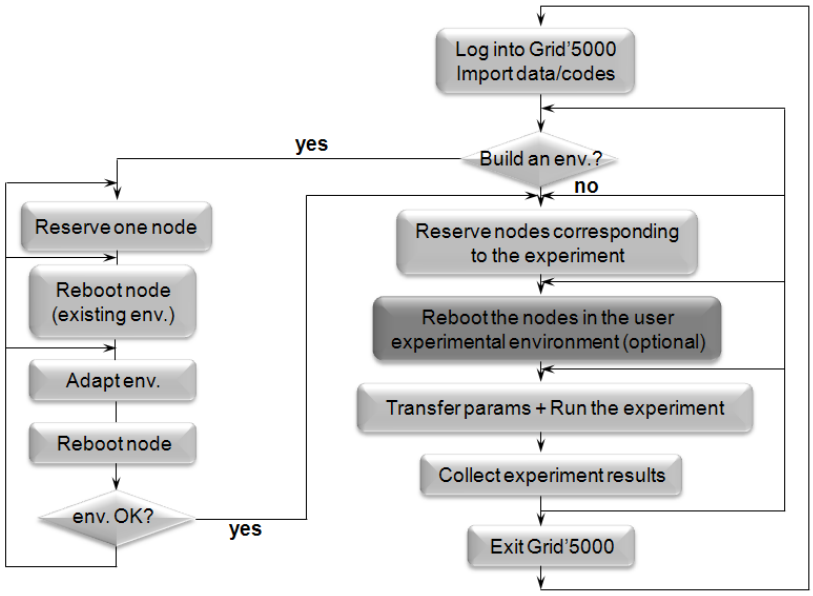


Figure 2.6: Grid'5000 Experiment Workflow.

is generally free, job or Absent. When nodes are sick, their state is Suspected or Dead.

DrawOARGantt displays past, current and scheduled OAR jobs, while Ganglia [41] provides resources usage metrics (memory, cpu, jobs...) for individual sites or the whole Grid.

To analyze standard output stdout and standard error output stderr of the main node, where script is run, OAR puts theses output in files named *OAR.scriptname.IdJob.stdname*.

In order to show who is using the platform, as well as the Number of usage hour (per month per cluster or per month per site), Kaspied is a statistic tool provided. Nagios [42] monitors critical Grid servers and services and automatically reports incidents and failures.

Note that while using above mentioned tools, user can choose individual site or cluster, or have a global view of the Grid.

2.3.7 Deploying an Environment

As mentioned in previous sections, KADEPLOY is the deployment system which is used by Grid'5000. It is a fast and scalable deployment system towards cluster and Grid computing. It provides a set of tools, for cloning, configure (post installation) and manage a set of nodes. Currently it deploys successfully Linux, *BSD, Windows, Solaris on x86 and 64 bits computers.

Any user may deploy his own environment (see Figure 2.6), which actually means any OS adapted by him to suit his experiments needs, on his reserved computational

nodes [10]. This way, he has a full control over which software or library is used, what kind of kernel or OS is running and how the system is configured for instance. Each site maintains an environment library in the **/grid5000/images** directory of the OAR node. To deploy an environment, by using a simple command *kaenvironments* on the frontend, user can know its name as registered in the KADEPLOY database.

2.4 Terminology

In order to have a clearer understanding of the upcoming discussion, the following terminology is introduced:

Virtual Organization

In grid computing, a Virtual Organization (VO) is a set of users working on similar topics and whose resources requirements are quite comparable. The users usually share their resources between them (data, software, CPU, storage space). The collaborations involved in Grid computing lead to the emergence of multiple organizations that function as one unit through the use of their shared competencies and resources for the purpose of one or more identified goals.

A user needs to be a member of a VO before he is allowed to submit jobs to the Grid. Moreover, a VO is usually defined by a name, and a scope which represents a geographical location constraint.

Grid Application

A Grid application is a collection of work items to solve a certain problem or to achieve desired results using a Grid infrastructure. For example, a Grid application can be the simulation of electromagnetic structure, like a planar antenna, that require a large amount of data as well as a high demand for computing resources in order to calculate and handle the large number of variables and their effects. For each set of parameters a complex calculation can be executed. The simulation of a large scale scenario then consists of a larger number of such steps. In other words, a Grid application may consist of a number of jobs that together fulfill the whole task.

Job

A job is considered as a single unit of work within a Grid application. It is typically submitted for execution on the Grid, has defined input and output data, and execution requirements in order to complete its task. A single job can launch one or many processes on a specified node. It can perform complex calculations on

[10] In this thesis, a customized environment called MEG, including an OS in addition to some needed libraries and legacy files, is used to run TLM and SCT experiments. (see Chapter 3 and 4)

large amounts of data or might be relatively simple in nature.

Parallel computing

The simultaneous execution of the same task (split up and specially adapted) on multiple processors in order to obtain results faster. The idea is based on the fact that the process of solving a problem usually can be divided into smaller tasks (see Figure 2.7), which may be carried out simultaneously with some coordination.

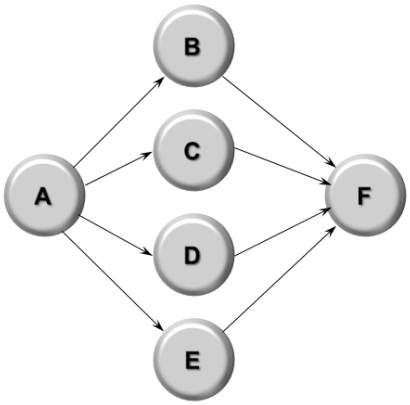


Figure 2.7: Parallel application flow.

To take advantage of parallel execution in a Grid, it is important to analyze tasks within an application to determine whether they can be broken down into individual and atomic units of work that can be run as individual jobs.

Serial or sequential Computing

In contrast to the parallel flow is the serial application flow. In this case there is a single thread of job execution where each of the subsequent jobs has to wait for its predecessor to end (see Figure 2.8) and deliver output data as input to the next job. This means any job is a consumer of its predecessor, the data producer.

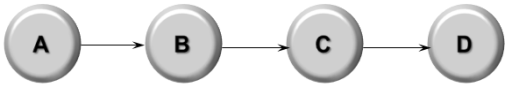


Figure 2.8: Serial job flow.

In this case, the advantages of running in a Grid environment are not based on access to multiple systems in parallel, but rather on the ability to use any of several appropriate and available resources. Note that each job does not necessarily have

to run on the same resource, so if a particular job requires specialized resources that can be accommodated, while the other jobs may run on more standard and inexpensive resources.

Parametric Computing

An application may consist of a large number of such calculations where the start parameters are taken from a discrete set of values. Each resulting serial application flow then could be launched in parallel on a Grid in order to utilize more resources. The serial flow A through D in Figure 2.8 is then replicated to A' through D', A" through D", and so forth.

Distributed computing

Distributed computing is a field of computer science that studies distributed systems, which consist of multiple autonomous computers (with onboard CPU, storage, power supply, network interface, etc.) that communicate through a computer network. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one computer.

Speedup

There are always parts of a program that cannot run in parallel, where code must run in serial. In parallel computing, speedup refers to how much a parallel algorithm is faster than a corresponding sequential algorithm.

The serial parts of the program cannot be speedup by concurrency. Let p be the fraction of the program's code that can be made parallel (p is always a fraction less than 1.0). The remaining fraction (1-p) of the code must run in serial. In practical cases, p ranges from 0.2 to 0.99.

The potential speedup for a program is proportional to p divided by the CPUs applied, plus the remaining serial part, 1-p.

As an equation, Amdahl's law could be expressed as:

$$Speedup(N) = \frac{1}{\frac{p}{N} + (1 - p)} \tag{2.1}$$

The maximum possible speedup (if applying an infinite number of CPUs) would be 1/(1-p). The fraction p has a strong effect on the possible speedup.

When the value of p could not be defined for a given problem, the following equation could be used:

$$Speedup(N) = \frac{Ts}{T_N} \tag{2.2}$$

where N is the number of processors, Ts is the execution time of the sequential algorithm and T_N is the execution time of the parallel algorithm with N processors.

Linear speedup or ideal speedup is obtained when Speedup(N) = N. When running an algorithm with linear speedup, doubling the number of processors doubles the speed. As this is ideal, it is considered very good scalability.

Granularity

Granularity is a measure of the size of the components (the size of the units of code under consideration in some context), or descriptions of components, that make up a system. Systems of, or description in terms of, large components are called ***coarse-grained***, and systems of small components are called ***fine-grained***. In parallel computing, granularity means the amount of computation in relation to communication, i.e., the ratio of computation to the amount of communication.

Fine-grained, or "tightly coupled, parallelism" means individual tasks are relatively small in terms of code size and execution time. The data are transferred among processors frequently in small amounts of messages.

Coarse-grained, or loosely coupled, is the opposite: data are communicated infrequently, after larger amounts of computation.

Execution Time

The execution time of a parallel program is defined here as the time that elapses from when the first processor starts executing on the problem to when the last processor completes execution, while one job is assigned to one processor.

During execution, each processor is computing or communicating. The computation time of an algorithm (T^icomp) is the time spent performing computation rather than communicating, while the communication time of an algorithm (T^icomm) is the time that its tasks spend sending and receiving messages.

On the ith processor, $T^i = T^icomp + T^icomm$.

Hence, to determine the total computation and communication performed by a parallel algorithm rather than the time spent computing and communicating on individual processors, total execution time T can be defined as the biggest value of T^i:

T = max (i=[0, N-1])($T^icomp + T^icomm$), while N is the number of processors

Both computation and communication times are specified explicitly in a parallel algorithm; hence, it is generally straightforward to determine their contribution to

execution time.

On the Grid, two distinct types of nodes communication will be distinguished: *internodes* communication and *intranode* communication. In *internodes* communication, two communicating tasks are located on different Grid nodes. This will be the case of distributed computing. In *intranode* communication, two communicating tasks are located on the same Grid node (via local bus).

Note that only the internodes communication will be highlighted later in the Grid experiments. This kind of communication could be established between nodes from the same cluster, called *intracluster* communication (i.e. two nodes of Pastel cluster in Grid'5000 Toulouse site), nodes from different clusters but in the same Grid site, called *intrasite* communication (i.e. one node of Pastel cluster and one node of Violette cluster in Grid'5000 Toulouse site), or nodes belonging to geographically remote sites, called *intersite* communication (i.e. one node from Toulouse site and another from Bordeaux site).

Bandwidth and Latency

Bandwidth in computer networking refers to the data rate (overall capacity of the connection) supported by a network connection or interface. The greater the capacity, the more likely that better performance will result.

Network bandwidth is not the only factor that contributes to the perceived speed of a network. A lesser known but other key element of network performance - latency - also plays an important role.

Latency could be defined as the delay between the initiation of a network transmission by a sender and the receipt of that transmission by a receiver. In a two-way communication, it may be measured as the time from the transmission of a request for a message, to the time when the message is successfully received by the requester.

The basic formula used to estimate communication time is $Tcomm = L + s/B$, where $Tcomm$ is the elapsed time to finish the communication, L is the latency of the network between the sender and the receiver, s is the size of the message and B is the available bandwidth.

2.5 Conclusion

Grid computing offers a model for solving computational problems by making use of resources (CPU cycles and/or disk storage) of large numbers of disparate computers, treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grid computing's focus on the ability to support computation across administrative domains sets it apart from traditional computer clusters or traditional distributed computing. This approach offers a way to solve Grand Challenge problems like protein folding, financial modelling, earthquake simulation, and climate/weather modeling.

But the use the Grid means dealing with aspects such as performance simulation,

scheduling and volatility of resources, etc. An application that runs on a stand-alone computer must be "gridified" before it can run on a Grid. In this work, the Grid is used to run electromagnetic simulations of complex structures. Modeling methods will be deployed on Grid'5000 nodes and the performance will be evaluated.

Chapter 3

TLM Modeling Method in Grid Environment

Contents

The Transmission Line Matrix (TLM) method, a flexible method used to model arbitrary and complex electromagnetic structures, is introduced in this chapter. The "gridification" of this method is then discussed and experiments presented.

3.1 Overview of the Transmission Line Matrix (TLM) Modeling Method

The TLM method is a key numerical method in computational electromagnetics. It is based on the equivalence between Maxwell's equations and the equations for voltages and currents on a mesh of continuous two-wire transmission lines.

This method is based on the analogy between the electromagnetic field and a mesh of transmission lines [43]. As a network model of Maxwell's equations formulated in terms of the scattering of impulses, it possesses exceptional versatility, numerical stability, robustness and isotropic wave properties.

The main feature of this method is the simplicity of formulation and programming for a wide range of applications [44, 45]. As compared to the lumped network model, the transmission line model is more general and performs better at high frequencies where transmission and reflection properties of geometrical discontinuities cannot be regarded as lumped [46].

The TLM was originally used for modeling electromagnetic wave propagation [44], [47], [48] but since it is based on Huygens principle it could be used for modeling any phenomena which obeys this principle. Researchers showed that TLM can be used to solve the following problems: Diffusion problem [49], Vibration [50], Heat transfer [51], Radar [52], Electromagnetic compatibility [53].

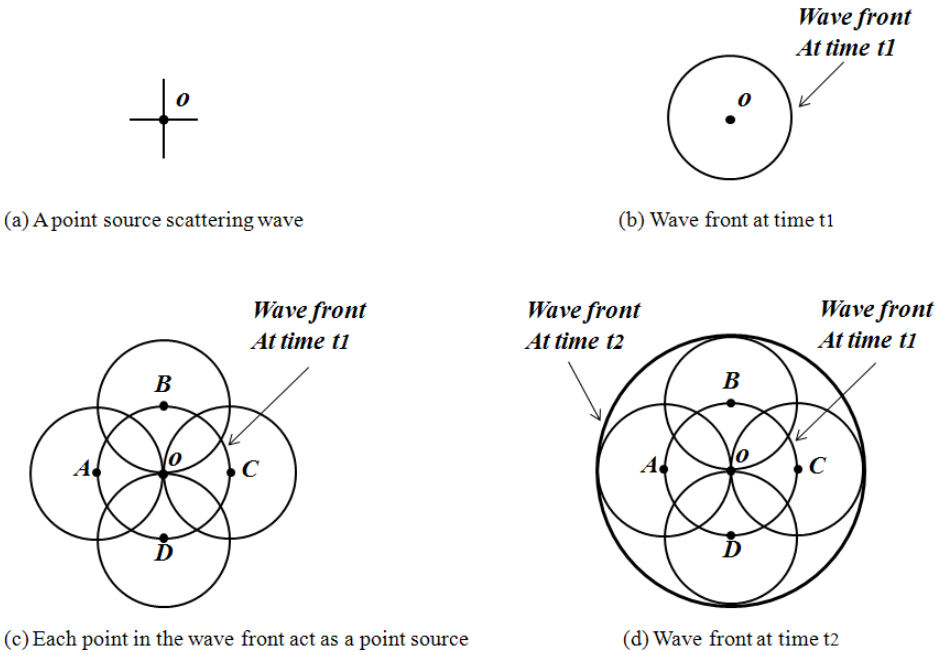


Figure 3.1: Huygens Principle.

3.2 From the Huygens principle to TLM modeling

Two distinct models describing the phenomenon of light were developed in the seventeenth century: the corpuscular model by Isaac Newton and the wave model by Christian Huygens. At the time of their conception, these models were considered incompatible. However, modern quantum physics has demonstrated that light in particular, and electromagnetic (EM) radiation in general, posses both granular (photon) and wave properties. These aspects are complementary, and one or the other dominates, depending on the phenomenon under study.

TLM modeling is based on the Huygens principle which is [54] - [56]:

All points on a wave front serve as point sources of spherical secondary wavelets. After a time T the new position of the wave front will be the surface of tangency to these secondary wavelets

This principle is shown in Fig 3.1. At time 0 the central point scatters a wave. The wave front at time ***t1*** is shown in Figure 3.1(b). At this time (time = ***t1***) we can assume that all points on the wave front are acting as a point sources (shown in Figure 3.1(c)) and the wave front at any time later (for example ***t2***) is the wave front from these secondary point sources (Figure 3.1(d)).

In order to implement Huygens's model on a digital computer, one must formulate it in discretized form.

3.3 TLM Basics

Johns and Beurle [47] modeled this principle in 1971 by sampling the space and representing it with a mesh of passive transmission line components. The wave propagation was modeled as voltage and current travelling in this mesh. Time was also sampled and the relationship between Δt, the sample interval and Δl, the sample space, is:

$$\Delta l = \Delta t.c$$

where $\boldsymbol{c}$ is the wave speed in the medium.

In Figure 3.2 wave propagation in a two dimensional TLM mesh is shown. Assume that at time zero, an impulse is incident to the middle node (Figure 3.2(a)). This node scatters the wave to its 4 neighboring nodes. The scattered wave reaches the neighboring nodes at time = Δt (Figure 3.2(b)). Now these 4 nodes scatter waves to their neighboring nodes (Figure 3.2(c)). At time = $2\Delta t$ the wave front can be found by finding waves scattered from points in Figure 3.2(b) as shown in Fiure 3.2(d). At each time step, each node receives an incident wave from its neighbors and scatters it to its neighbors. By repeating the above calculation for each node, the wave distribution on the medium can be calculated. Subsequent papers by Johns and Akhtarzad [57] - [63] extended the method to three dimensions and included the effect of dielectric loading and losses. Building upon the groundwork laid by these original authors, other researchers [64] - [87] added various features and improvements such as variable mesh size, simplified nodes, error correction techniques, and extension to anisotropic media.

3.4 TLM Algorithm

As explained above, the TLM method, like other numerical techniques, is a discretization process. Unlike other methods such as finite difference and finite element

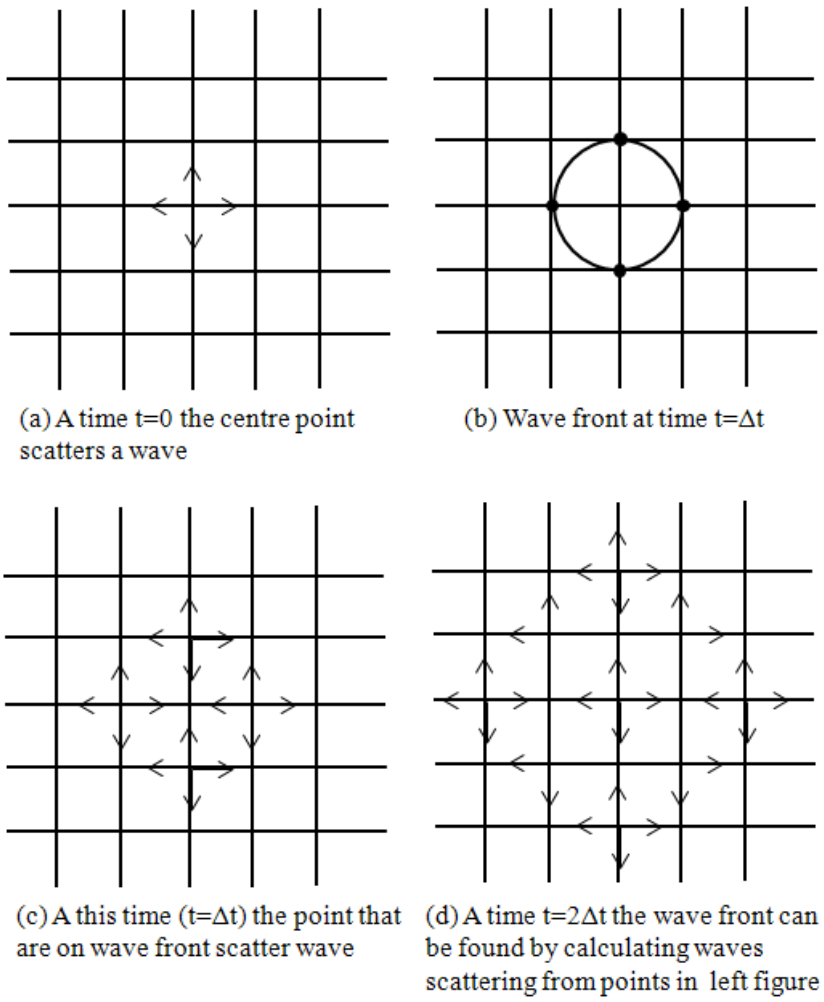


Figure 3.2: Wave propagation in a two dimensional TLM mesh.

methods, which are mathematical discretization approaches, the TLM is a physical discretization approach.

For example, consider the structure to simulate in Figure 3.3. The TLM method involves dividing the solution region into a rectangular mesh of transmission lines. Junctions are formed where the lines cross forming the impedance discontinuities.

A comparison between the transmission line equations and Maxwell's equations allows equivalences to be drawn between voltages and currents on the lines and electromagnetic fields in the solution region.

When all sections of a medium have the same properties, the medium is referred as homogeneous. For modeling homogeneous media it does not need to consider the medium properties and hence it is possible to use the simple mesh.

The relationship between the incident wave (voltage in transmission line) and the scattered wave is:

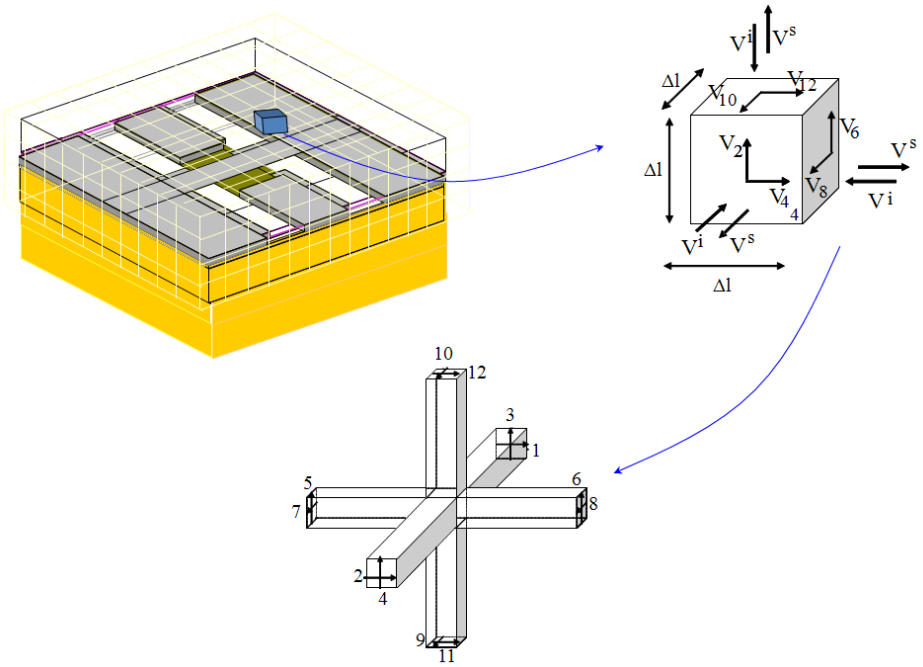


Figure 3.3: Entire space is discretized. On each face of each cell (cuboïd) the field is decomposed in the two polarization components (***i*** stand for incident, ***s*** stands for scattered). The cubic cell is modeled by the three-dimensional symmetrical condensed (SCN) TLM node proposed by P. B. Johns in 1986.

$$V_k^s = \boldsymbol{S} \cdot V_k^i \; ; \; V_{k+1}^i = \boldsymbol{C} \cdot V_k^s$$

In this figure, V^i represents the incident wave (voltage in the transmission line) and V^s represents the scattered wave. $\boldsymbol{S}$ is the impulsive scattering matrix of the node, and $\boldsymbol{C}$ is a connection matrix describing the topology of the TLM mesh. $\boldsymbol{k}$ and $\boldsymbol{k+1}$ are arbitrary consecutive time steps separated by the sample interval Δt. Based on this equation, if the magnitude of the wave (voltage in the TLM modeling) is known at any time $\boldsymbol{k}$. Δt then the magnitude of wave in the mesh could be found at time $\boldsymbol{(k+1)}$. Δt. By repeating this for each time step, wave propagation could be modeled.

In the case of square cell (uniform mesh) and loss-free propagation wave, the scattering matrix $\boldsymbol{S}$ is a 12x12 sparse matrix.

When modeling a non homogeneous medium, one should consider the properties of the medium in the model. For this reason a new model for a node is created by adding a capacity [57], [58]. This model is valid when the medium is lossless. When there are some losses in the medium, there is a resistor in parallel to the capacitor to model the loss [61].

In case of non uniform mesh (cell with $\Delta x \neq \Delta y \neq \Delta z$) where stub are introduced to compensate line delay in the scattering, the resulting is a scattering matrix $\boldsymbol{S}$ of 18x18. If it is completely equipped with permittivity, permeability,

and loss stubs, the matrix goes up to 30x30.

Thus, the TLM method involves two basic steps [88]:
- Replacing the field problem by the equivalent network and deriving the analogy between the field and network quantities.
- Solving the equivalent network by iterative methods.

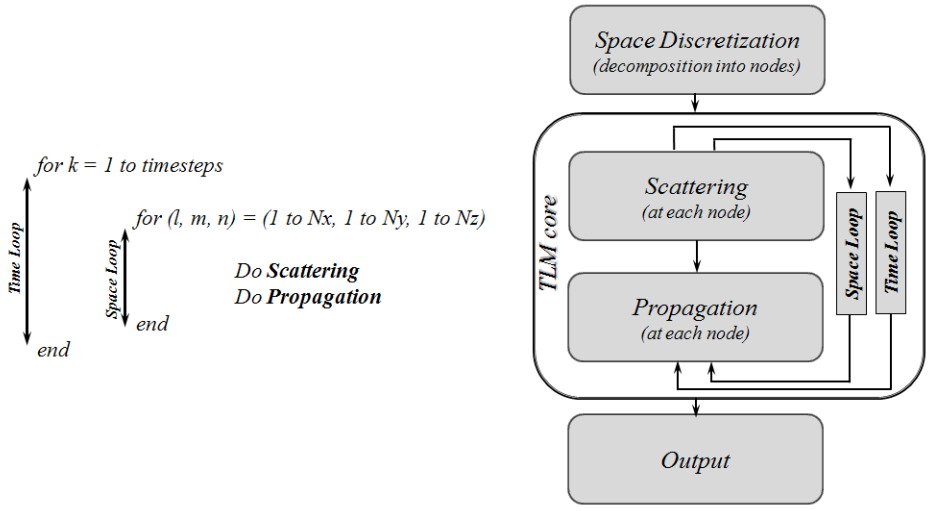


Figure 3.4: TLM Algorithm.

3.5 Implementation of TLM in Parallel computers

Since TLM is a numerical model, it should be implemented in software and executed in digital computers. Since TLM requires that the entire computational domain be gridded, and the grid spatial discretization must be sufficiently fine to resolve both the smallest electromagnetic wavelength and the smallest geometrical feature in the model, very large computational domains can be developed, which results in very long solution times (which is also the case of the Finite-difference time-domain (FDTD)). Thus it needs a large amount of memory and CPU power to model wave propagation, and consequently it is hard to model a real wave propagation problem with normal computers.

Since the TLM algorithm is based on local operations at neighboring mesh nodes only, the TLM method seems to be adapted for parallel computing. Clearly, to have a single processor working though the computations for each node in turn is therefore a suboptimal solution. Furthermore, since a speed increase is required, parallel processing is a logical step even if the communication is a critical point.

There are a number of different models of parallelism that has been considered. Past work on parallel TLM has often used special hardware. A number of papers discuss using Transputers (Transistor Computer). In [89], Transputers have

been used for parallel simulation of a multistage power electronic circuit. In [90], a distributed array processor (SIMD - AMT DAP 510 machine) has been used to reduce computational time when solving a two-dimensional TLM electromagnetic field formulation.

Massively parallel computers (MPP) have been used in [91] - [95]. These computers all have a large number (8 to 32K) of CPUs; each CPU is used to model a TLM node. In order to take advantage of these massively parallel computers, the scattering and transfer operations must be implemented using parallel languages designed for those computers.

Since, these computers were expensive and not widely available for general use, another approach [96] based on multithreading has been tested with multi-processor computers (8-processor IBM RS/6000 SMP).

In [91] - [96] a fine grain approach (ref. Chapter 2) was proposed. It demands specialized computer hardware employing thousands of processing units. Moreover a high communication bandwidth is required.

In [97] a coarse grain approach (ref. Chapter 2) was introduced, featuring interconnected standard RISC workstations. This approach needs only small communication bandwidth and makes best use of the available resources. As a parallel computing environment, the Parallel Virtual Machine (PVM) was used [98]. Within the PVM the processing units interchange their information by message passing techniques, which can be implemented into TLM. A static load balancing was achieved by an optimum segmentation of the mesh based on an initial guess of the available computer power at the particular machines and with respect to minimum communication traffic between the computing nodes. Up to six HP-715 and HP-720 workstations were used.

In [99] and [100], for the simulation of microwave circuits, interconnected workstations as well as an IBM-SP2 parallel computer consisting of 56 computing nodes were employed using PVM. As a result of the synchronization required throughout the mesh after each TLM time step, each working client has to await the completion of the scattering and propagation processes at the adjacent sub meshes. The access mode in commonly used Ethernet-LANs limits the throughput within the PVM, and thus, the suitable number of computing nodes within a loosely coupled network. The time required for the communication between all adjacent, clients is considerable larger than the accumulated communication time between each two clients operating on adjacent sub meshes.

In [101] an approach has been investigated using the Ruby programming language [102].

A Grid-enabled time-domain transmission line matrix (TLM-G) system for full-wave analysis of complex electromagnetic (EM) structures is presented in [29]. The system has been tested on 7 computers network for the computation of the input impedance of bowtie antenna.

All these applications mentioned above are executed in parallel computers or supercomputers, which mean that most of these cases are based on shared memory parallelism. Besides, the cases of distributed computing shown have been executed

on tight coupled computing nodes interconnected by local network.

The purpose now is to estimate the efficiency of implementing the TLM method in Grid Computing Environment, on (geographically) distributed computing nodes belonging to different Grid clusters.

3.6 Distributed Parallel TLM Simulations in Grid Environment

Grids differ from traditional parallel production environments (PPEs) in both structure and typical use. While a PPE consists of a single (large) tightly coupled supercomputer or a well-interconnected cluster, a Grid consists of multiple sites, each of which in the type of Grids we consider typically comprising a commodity cluster with a more common network.

Distributed computing techniques (ref. Chapter 2) offer the potential to significantly reduce the run time of TLM calculations. To get better performance by using more resources, users may want to execute their applications, written for clusters, on Grids.

Today, clusters are often interconnected by long distance networks within Grids to offer a huge number of available resources to a range of users. It is therefore very important to test the influence of network on the performance of TLM computation when tested on such platform.

Consider a simple air-filled cavity of size Δx * Δy * Δz = 400 mm * 400 mm * 10 mm. This cavity is discretized using a three-dimensional TLM cartesian uniform mesh of dl = 0.5 mm.

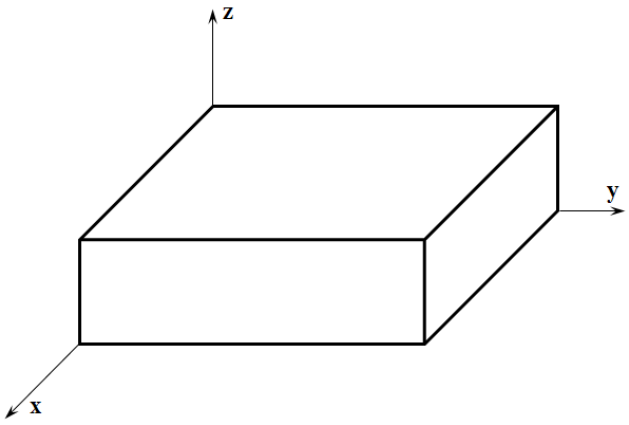


Figure 3.5: The three-dimensional medium modeled with TLM method.

If the speed in the medium is considered equal to $3 * 10^8$ m/s and the wave frequency 12.5 GHz then:

Wavelength = Speed / Frequency = 24 mm

and Number of TLM nodes = (Δx/dl) * (Δy/dl) * (Δz/dl) = 12 800 000 nodes.

If for storing each data we use 8 bytes (double precision) then each node needs 12 * 8 bytes of memory. Consequently, the memory needed for modeling the entire medium is tens of Gigabytes for fine mesh.

The use of **MPI** to distribute the TLM computations on Grids nodes will be investigated in next sections since MPI remains the dominant model used in high-performance computing today.

3.6.1 Message Passing Interface (MPI)

MPI is proposed as a standard to write parallel applications by a broadly based committee of vendors, implementers, and users. MPI is a specification for an *application programming interface* (API) that allows many computers to communicate with one another by message-passing [1]. Each node has its local memory to be used for computation.

Before MPI, PVM [98] was the reference on message passing environment, but with a stronger focus on resources/process management, dynamicity, the idea of a virtual parallel machine and transparency.

On the other side, MPI focus in performance, a clear interface featuring a powerful support to collective communication and different parallel machine architectures, from shared-memory multiprocessor machines to clusters. Also, the support to fast interconnection networks has being one of the main keystones.

Most MPI implementations consist of a specific set of routines (i.e., an API) callable from Fortran, C, C++ or Java and from any language capable of interfacing with such routine libraries. The advantages of MPI over older message passing libraries are portability (because MPI has been implemented on almost every distributed memory architecture) and speed (because each implementation is in principle optimized for the hardware on which it runs).

MPI implementations are initially written for high performance on both massively parallel machines and on workstation clusters and do not consider the specificities of Grid interconnections. The main feature of Grid, which is the long distance networks, raises the question of MPI efficiency in Grids.

3.6.2 MPI on Computing Grids

The Grid raises mainly three problems to execute MPI applications. First, MPI implementations have to manage efficiently the long-distance between sites. Actually, the high latency between sites is very costly, especially for little messages. Inter-sites communications take more time than intra-sites ones. In a Grid, inter-sites links may offer higher capacities than cluster links.

[1] more informations could be found on: https://computing.llnl.gov/tutorials/mpi/ by Blaise Barney, Lawrence Livermore National Laboratory.

Several MPI implementations are taking some characteristics of Grids into account, to offer better execution of MPI applications on Grids. The MPICH2 [103], GridMPI [104], MPICH-Madeleine [105], OpenMPI [106], MPICH-G2 [107] implementations can be used on a Grid but are not similarly optimized to this specific context.

In order to identify which ones are efficient on a real Grid and in which conditions, how to best configure these implementations to achieve good performance on the Grid, and to have a better view of which communication patterns better fit Grid context, in [108] experiments in Grid'5000 have been conducted. The paper details the tuning required on each implementation to get the best performances.

Since this study has been performed on the actual Grid'5000 architecture and network, there no need to re-validate the results (bandwidth and latency of MPI implementations) and they can be used in this thesis.

The comparison is based on the execution of **pingpong** between two nodes. The results within a cluster are done between two nodes of the Rennes cluster (intra-site / one cluster) corresponding to P_{R1} and P_{R2} on Figure 3.6. The experiments in the Grid (inter-sites) are done between P_{R1} and P_{N1}.

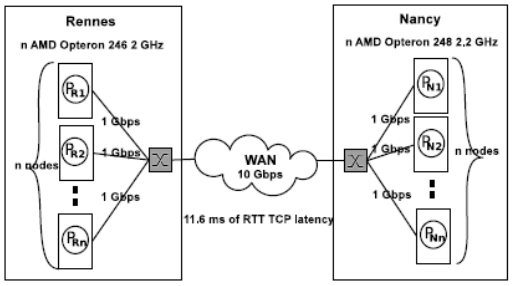


Figure 3.6: Configuration used.

Table 3.1 shows the latency comparison between the different MPI implementations in the Rennes cluster or in the Grid (between Rennes and Nancy).

	Cluster: Rennes	Grid: Rennes-Nancy
TCP	41	5812
MPICH-2	46(+5)	5818(+6)
GridMPI	46(+5)	5819(+7)
MPICH-Madeleine	62(+21)	5826(+14)
OpenMPI	46(+5)	5820(+8)

Table 3.1: Comparison of latency in a cluster and in a Grid (in μs).

The bandwidth obtained on a cluster (Intra-site) and on Grid (Inter-sites: Rennes-Nancy) are shown respectively in Figure 3.7 and Figure 3.8.

Using these results on network communications, the TLM parallel application efficiency will be investigated.

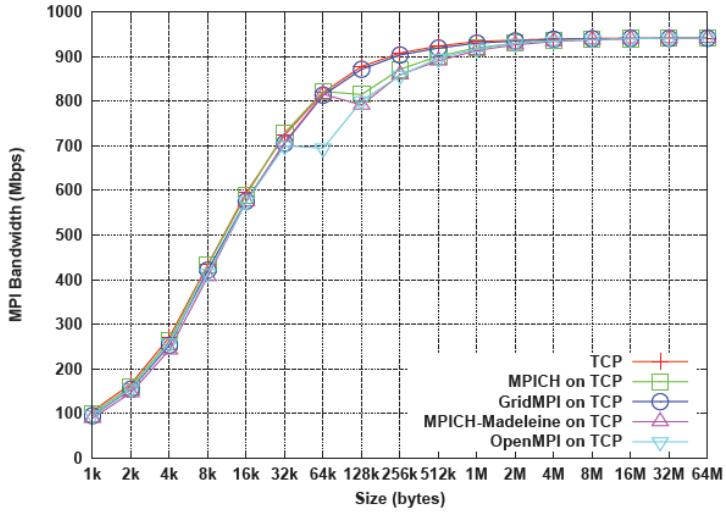


Figure 3.7: Comparison of MPI bandwidth of the different MPI implementations on a local network (cluster).

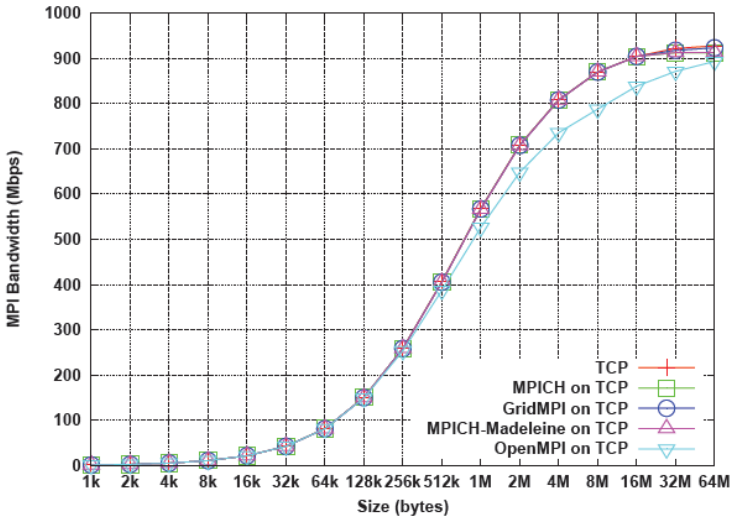


Figure 3.8: Comparison of MPI bandwidth of the different MPI implementations on a distant network (Grid).

3.6.3 Efficiency of using MPI for TLM

The method is limited by the amount of memory storage required, which depends on the complexity of the structure and the non uniformity of fields set up in it. In general, the smallest feature in the structure should at least contain three nodes for good resolution. The total storage requirement for a given computation can be determined by considering that each basic three-dimensional node requires twelve number locations; if it is equipped with permittivity, permeability, and loss stubs,

the required number of stores will increase. Again, one real number must be stored per output function and per iteration. The number of iterations required varies between several hundred and several thousand, depending on the size and complexity of the TLM mesh.

3.6.3.1 Division of the TLM medium in sub-media

For modeling in parallel, the medium is divided on some regions (Figure 3.9) and then each region is processed on one Grid computing node. These processes should be communicating (*Send* and *Receive*) with each other computer in use.

If N Grid computing nodes available, medium shall be divided into N sections. Each computing node models only one sub-medium. At the boundary (Figure 3.10), it is necessary to exchange data between sub-media (incidence and scattering for each iteration (ref. TLM algorithm)). Each TLM node at the boundary interface exchanges two values. Figure 3.9(a) shows two sub-media on two computing nodes that communicate via one boundary.

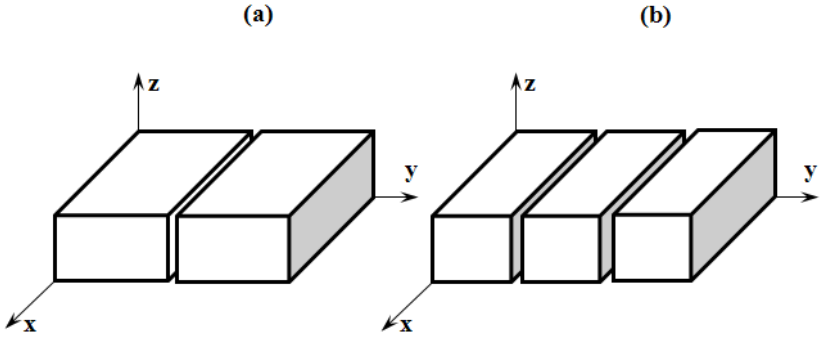


Figure 3.9: Division of the medium to sub-media in one direction on a (a) two and (b) three Grid computing nodes.

When the medium is subdivided to three, the number of nodes on the boundary does not change comparing to the case of two sub-media, but in this case two boundaries are considered (Figure 3.9(b)). A Grid node in charge of a border area communicates only with the Grid node running the one of the center. Meanwhile, the Grid node in charge of the center process must communicate with both other Grid nodes.

When N, the number of used computing nodes, increases, the number of boundaries increases also (Figure 3.11). The sub-medium represented in black is the most communicating one since it has an iterface boundary with six neighbours.

TLM algorithms are iterative and the time loop (see Figure 3.4) is essential. Before beginning a new (next) iteration, the values of the TLM node must be computed and ready to be communicated. Synchronism is critical in such cases. Note that some areas consuming more communication time than others, i.e. the medium in the center in figure 3.11, block the others (which are faster).

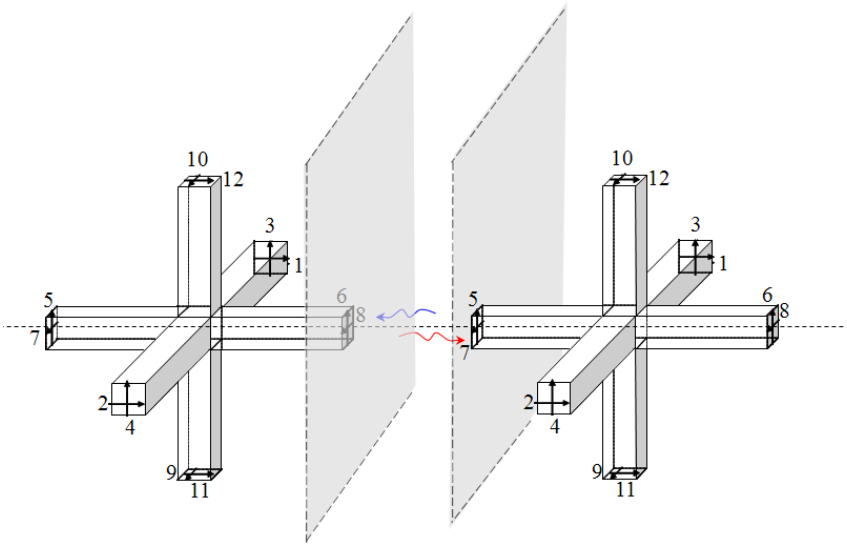


Figure 3.10: Two TLM nodes exchanging at the boundary of the sub-media.

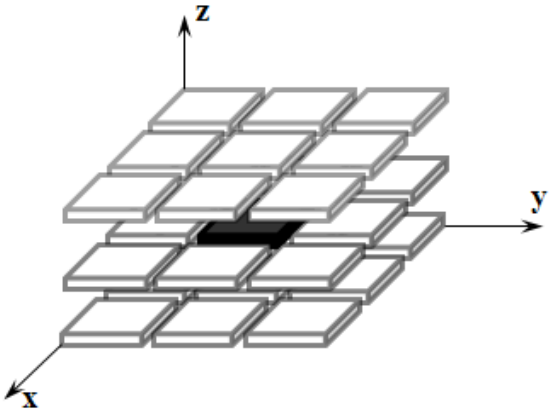


Figure 3.11: Division of the medium to sub-media in all directions. The sub-medium represented in black communicates with six neighbours.

In the following, a full duplex system will be considered (optimal case). The sub-medium is supposed to send and receive messages at the same time.

3.6.3.2 Performance

As defined in previous chapter, the execution time is the sum of computation time ($T^i comp$) and communication time ($T^i comm$). To find the relation between the computation time and the computing volume (number of TLM nodes), the TLM solver of YATPAC (Appendix A) has been used with different structures and mesh

grids [2].

Based on curves in Figure 3.7 and in Figure 3.8, the latency and bandwidth could be deduced and used in equations defined in the last section of chapter 2 to calculate the communication time for a specific weight of message. The number of TLM nodes on the communication interface defines the size of the message (s).

Figure 3.12 shows the estimated speedup when distributing the computing on the same cluster nodes and the speedup when using computing nodes from different Grid site, compared to the optimal linear speedup.

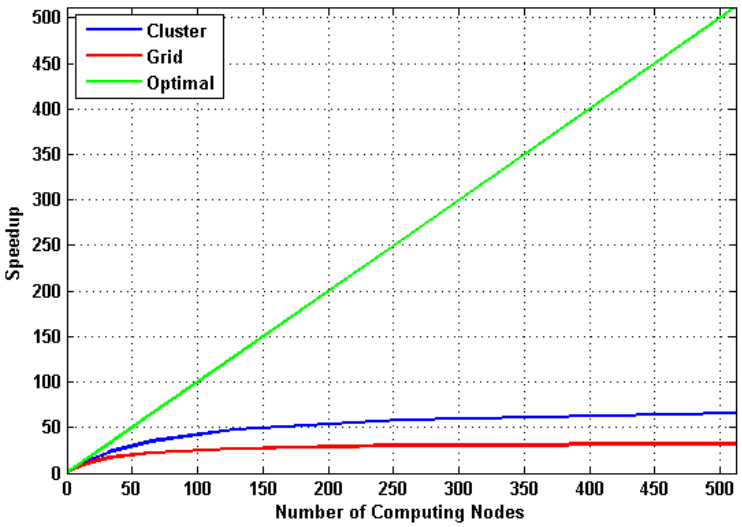


Figure 3.12: Estimated speedup when dividing the medium to sub-media in one direction.

The speedup has been also estimated in the other partitionning case (Figure 3.13) while considering the same division coefficient for the three dimensions.

[2]The method of least squares has been used as a method of fitting data.

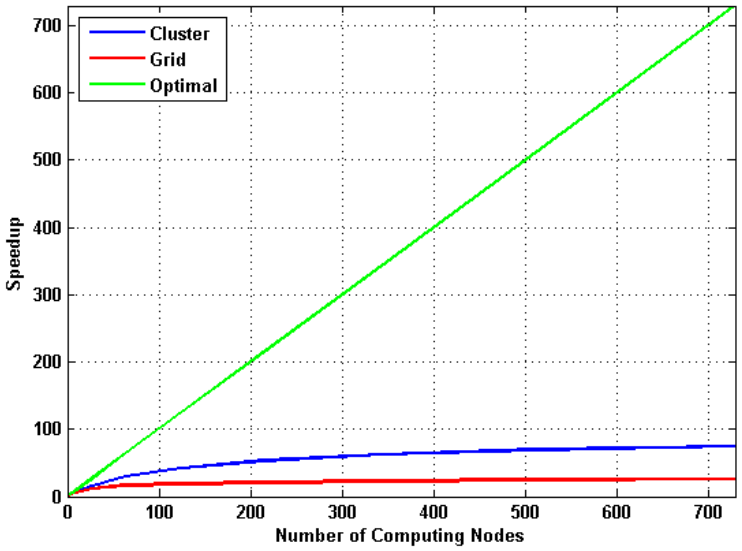


Figure 3.13: Estimated speedup when dividing the medium to sub-media in three directions.

The speedup reaches a stable state rapidly in both cases. Even if the cluster speedup is little bit better than the Grid one, it remains too low. When the number of TLM nodes increases (finer mesh of the same cavity (dl = 0.1 mm) or bigger struture), bigger speedup can be noticed for the division in one direction (Figure 3.14) and for the division in all directions(Figure 3.15).

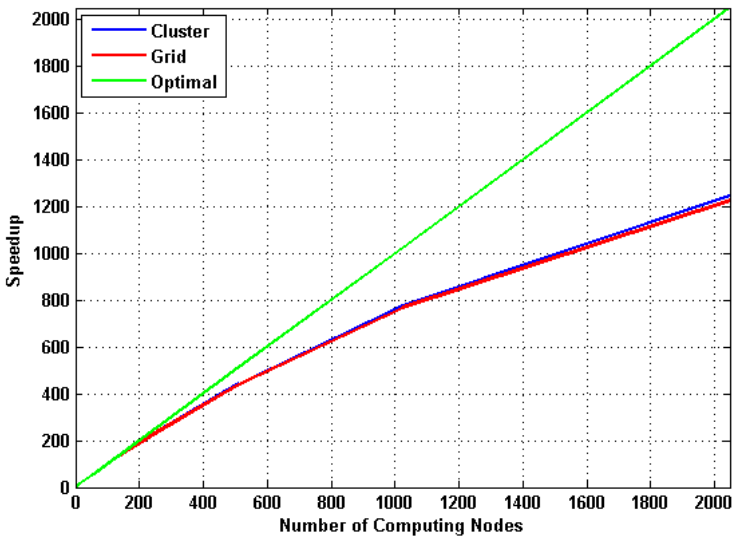


Figure 3.14: Estimated speedup when dividing the medium to sub-media in one direction.

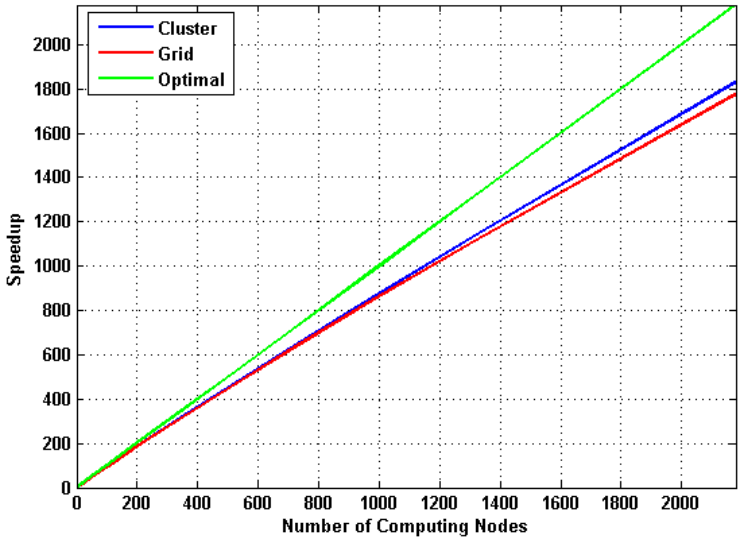


Figure 3.15: Estimated speedup when dividing the medium to sub-media in three directions.

3.7 Distributed Parametric TLM Simulations in Grid Environment

This section discusses much simpler parallel applications, known as "parametric" or "embarrassingly parallel", which do not request data transfer during executions. This kind of computer programs involves execution of the same code with different input parameters. Some typical examples include frequency sweeps for antenna characterization, or design of optimal antenna geometries.

Since parametric problems are the simplest kind of parallel distributed applications, there is no need to face the complexity and restrictions of traditional network computing libraries for developing an execution framework. On the other hand, novel network technologies greatly simplify the deployment of highly scalable distributed systems at relatively small bandwidth and computing cost.

Parametric distributed study application is a good candidate to execute in a Grid environment [32]. In distributed parametric TLM simulations, a wide range of design parameters are evaluated in a single analysis run with the goal of exploring the entire design space and selecting the optimized design without need for the normal iterative process (Figure 3.16).

This Grid-based electromagnetic approach, exploits the availability of computing node at disposal through the Grid to face the demand of arbitrary large simulations by allocating a corresponding amount of resources hence minimizing the overall elapse time.

Full-wave electromagnetic solvers based on the Transmission Line Matrix Method have been deployed on Grid test-bed.

Different approaches to run distributed parametric experiments have been used

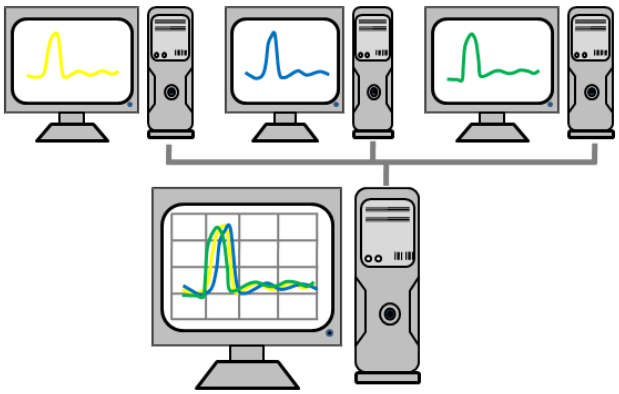


Figure 3.16: Distributed parametric computing.

along this thesis. The first one was the simplest with bash scripts and YATPAC (Appendix A)[3]. Since the difficulty for non expert in computer science to run those applications on high number of computing nodes discourages them, a second approach is based on the deployment and management of the electromagnetic solver yatsim by using the TUNe [4] autonomic middleware. The last presented approach is the most developed one. It is based on TUNe that deploys emGine environment (Appendix B) on the Grid'5000 nodes with a graphical user interface (GUI).

In order to deploy YATPAC on Grid'5000 nodes (with shell scripts and TUNe), a customized Linux OS environment, called MEG was prepared for KADEPLOY. It should be mentionned that the generation of MEG was not an easy step and took a considerable time because of problems with YATPAC tools installation.

Usually, to create such environment, user reserves a Grid node, deploy an exisisting basic OS (Fedora, Debian, Ubuntu, ...), add needed software by installing them, and save this customized environment to be used later (see section 2.3.6 of chapter 2). In fact, YATPAC presents lot of libraries dependencies (atlas, bison, blas, fftw, lesstif, mesa, zlib, gcc, ...) and the makefiles could not make the job easily (problems of path, ...).

Note that, in addition to TLM solver and software, MEG also contains the libraries and codes necessary for SCT simulations on the Grid (see chapter 4).

3.7.1 First Approach: Shell Scripts + YATPAC

The methodology proposed is experimented on a coplanar 6-bits phase-shifter based on MicroElectroMechanical switches (MEMS)[5]. The circuit design proposed here is based on a truetime delay phase shifter which consists of a coplanar waveguide

[3] An easy deployment system, MEG GUI, based on Taktuk has also has been developed and tested. It can launch TLM (with YATPAC) and SCT experiments, and could be found in Chapter 4.

[4] Toulouse University Network. http://hagimont.perso.enseeiht.fr/transp-jte/hagimont.pdf# 5

[5] The phase shifter was designed by VTT (Finland) and realized by FHG-ISIT (Germany) in the framework of the first AMICOM multi project wafer.

(CPW) transmission line loaded periodically with several shunt MEMS capacitors (Figure 3.17).

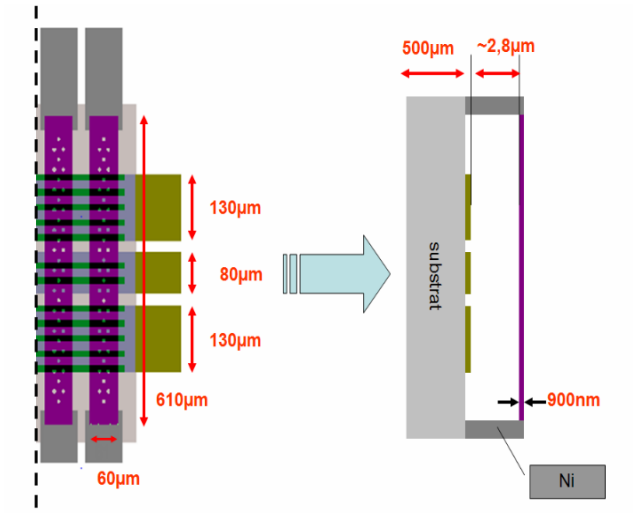


Figure 3.17: RF MEMS planar phase-shifter cross section.

Thus the circuit can be considered as a synthetic transmission line whose phase velocity can be varied by switching the MEMS capacitive switches up and down [109].

For illustration purposes, this structure has been used in a parametric design sweep, and the phase shift of each configuration have be simulated.

As mentioned in appendix A, in the preprocessing step, the engineered structure and the discretization of each model must be prepared. To generate automatically the different prototypes with its proper specification, a C++ code has been implemented. This generator needs to be run once, and all the simulator input files are ready-to-use.

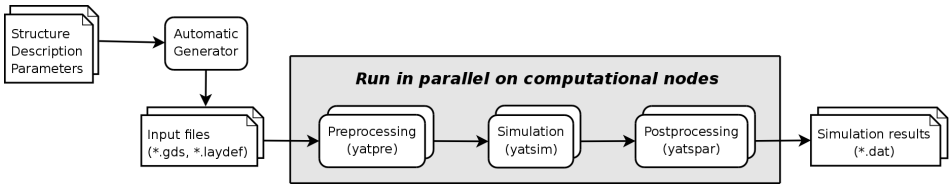


Figure 3.18: Simulation workflow.

Simulation steps take long computational time, and must be repeated for each generated input file. Using a Grid infrastructure enables to parallelize this work (Figure 3.18) by reducing the number of simulations treated by each node.

Once all of the models are analyzed, response data are automatically collected for the postprocessing step. The simulation executions on the Grid will be detailed in the following.

In order to test the behaviour of the infrastructure, an experiment composed of 192 TLM heteregeneous simulations will be executed several times, for different numbers of computational nodes to use. The number of nodes used will be increased gradually, in order to measure the impact of this metric on the performance of each step of the simulations.

The experiments use up to 3 clusters (the ones of Nancy, Toulouse and Sophia-Antipolis), with a maximum of 64 nodes per cluster. The cluster characteristics are the following:

- **Cluster of Nancy:** Intel Xeon 5110, 1.6GHz
- **Cluster of Toulouse:** AMD Opteron 2218, 2.6GHz
- **Cluster of Sophia-Antipolis:** AMD Opteron 2218, 2.6GHz

The number of nodes used is gradually increased. First, 4 nodes of the cluster in Nancy are used. When the number of nodes reaches 64, a second cluster is used (the one of Toulouse); the 64 nodes of the first cluster are still used, and the number of nodes used on the second cluster is gradually increased to reach 64 as well. Finally, the third cluster (Nancy one) is included when more than 128 nodes are required.

The simulations were divided into 5 main parts, namely: Initialization (i.e. Generation of the Simulations Input Files), Deployment of a Customized Linux operating system that bundles YATPAC Simulation Package, Deployment of the Input Files over the Computational Nodes, Execution of the Simulations, and Results Recovery.

3.7.1.1 Initialization Phase.

During this phase, the simulation input files are generated on each Grid'5000 sites, using the C++ prototypes generator (Figure 3.18), and then stored in the local NFS server. The generated files (`PS.laydef` files), particularly their number and their content, depend on the simulations that have to be performed. This is done independently but simultaneously on each site.

This phase occurs at the beginning of the experiments. It is executed on one host per cluster (typically the OAR server node), and generated files are stored on the NFS server of each cluster. When several clusters are used, the execution on each cluster is done in parallel at the same time. Thus, the duration of this phase is almost constant, and depends not on the number of nodes used, but only on the load of OAR server, NFS server and network. The experiments show that this phase lasts between 1.5 and 2 seconds.

3.7.1.2 Deployment of the Customized Linux Image.

The KADEPLOY tool is used to reboot all the computational nodes in order to run MEG, the operating system that bundles the YATPAC Simulation Package (see section 2.3.6 of chapter 2). This operation is done for all nodes used during the experiment. Note that, since there is no file sharing between the different sites of

Grid'5000, a copy of the Linux image is stored in each local NFS server of each of them, in order to be used by KADEPLOY.

Before launching the simulations, the Linux image that contains YATPAC is deployed over the different nodes. Figure 3.19 shows the duration of this phase for the different numbers of nodes used.

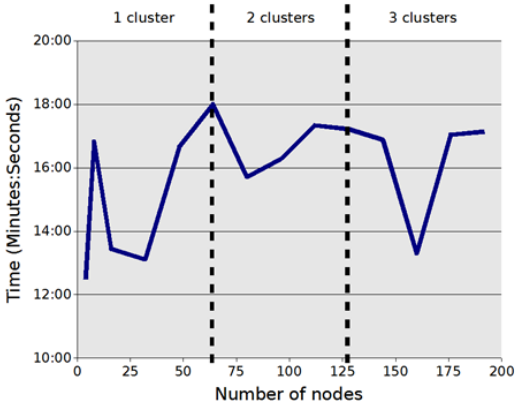


Figure 3.19: Duration of MEG deployment phase.

The duration of this phase is between 12 and 18 minutes, but it is not accurately predictable. The deployment is performed in parallel on each node, so that the number of nodes used would not affect the performances of the system during this phase. Actually, this is not so simple because the Linux image to deploy is stored in the NFS server, and the performances may depend on the number of requests, and thus on the number of nodes used. The activity of other Grid users can also have an impact on the performances of the NFS server, and so on the duration of the deployment of the image.

3.7.1.3 Deployment of the Simulation Input Files.

When all computational nodes have been rebooted with MEG image that contains YATPAC, the simulation input files generated in first step and stored in the local file servers, must be deployed on each of them. Indeed, even if these files are accessible from every node on the NFS server, it is better to make a local copy of them in order to optimize their access during the simulations, avoiding simulation slowdowns due to multiple networking accesses. The files that are transferred to a computational node are filtered in order to copy only the files corresponding to the simulations that will be run on this node. Two files are required to perform a simulation: `PS.laydef` (specific to each simulation) and the corresponding `PS.gds`. The duration of this phase is represented by Figure 3.20.

The profile of the measured performance points out two different cases. First, when only one cluster is used, the duration of this phase depends directly on the number of nodes used, in a linear way. The copy of the input files is performed by

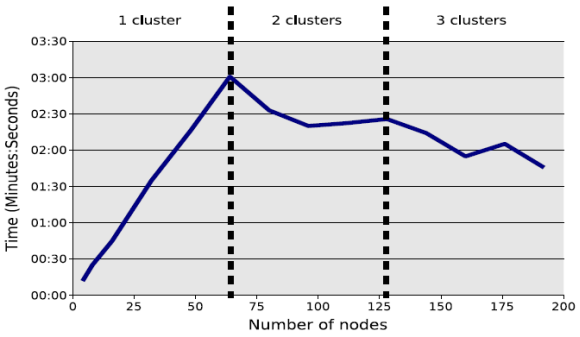


Figure 3.20: Duration of the input files deployment phase.

the OAR server node, which sends these files to each computational node. Thus, the duration of this phase increase with the number of nodes used. Note that both network load and NFS server performances have an impact on this duration.

If several clusters are used, this assertion is not verified anymore. In the case of two clusters, the number of used nodes is gradually increased. When 64 nodes are used on the first cluster, the nodes of second cluster is then used. Thus, in this case, there is a cluster for which the deployment of the input files must be performed for 64 nodes. The duration of this period is almost constant, and corresponds to the time needed for the copy of the input files on the 64 computational nodes of the cluster.

So when the number of hosts increases, it is interesting to distribute them over different sites, in order to bound the duration of this phase.

3.7.1.4 Execution of the Simulations.

All simulations are executed on the computational nodes. Each node executes a subset of simulations, corresponding to a limited number of input files. All nodes execute the same number of simulations, since spreading them equally over the nodes leads to optimize the duration of this phase. The simulations are independent, so they can be run in parallel. They use the input files generated in previous steps, and produce new result files. The following commands enable to run the simulation using YATPAC:

```
yatpre -t t -i PS.laydef PS.gds I3D.Temp
yatsim Temp
yatpre -t ref -i PS.laydef PS.gds I3D.Ref
yatsim Ref
yatpre -t C -i PS.laydef PS.gds I3D.Dev
yatsim Dev
yatpre -t C -i PS.laydef -M PS.gds TLMSTRUCT
yatspar -v -h 8.1 -l 3e-6 -f 25e7 -N 200 2:EH.Ref 8:EH.Dev S_Parameters_PS.dat
```

This phase is the longest one, since it corresponds to the execution of the YATPAC tools, mainly the *yatsim* the simulator, in order to explore all possible configurations of the simulated problem.

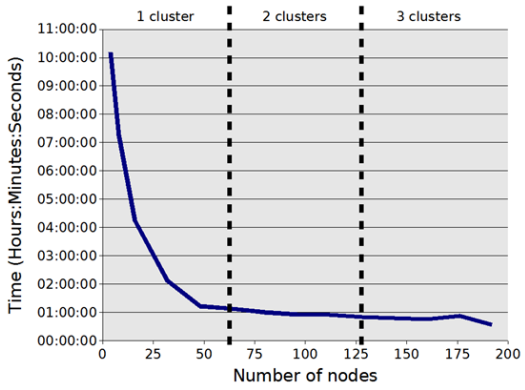


Figure 3.21: Duration of the simulation execution phase.

The duration profile of this phase (Figure 3.21) presents a good regularity. The execution time of the simulations depends on the number of nodes that are used, or more precisely on the number of simulations that are executed on each node. Moreover, the experiments have been designed in order to obtain an homogeneous distribution of the simulations over the computational nodes.

Since the experiments are composed of 192 simulations, the mean number of simulations per node is: $\frac{192}{Number\ of\ nodes}$. This expression matches the hyperbolic appearance of the measured execution times. For example, if only 4 nodes are used, each of them executes 48 simulations. More than 10 hours are then needed to complete this phase. If 192 nodes are used (1 simulation per node), the execution of the simulations only takes 34 minutes.

This result points out the benefit from using a large number of computational nodes for running TLM simulations.

3.7.1.5 Retrieval of Result Files.

The generated result files (`S_Parameters_PS.dat`), present on each computational node, must be retrieved and stored on the NFS servers. Indeed, when the experiments end, all nodes are rebooted to run a default operating system, so that other users are enabled to use them for their own experiments; this operation deletes all files stored on each node.

The last step of the experiments is to retrieve the results from the computational nodes and to store them on the local NFS servers. The two-port scattering parameters of the circuit were recorded up to 40GHz (Figure 3.22). From 30 GHz to 40GHz the structure could cover almost an entire interval of 360°.

Figure 3.23 shows that the duration of this phase decrease with the number of nodes used. Some irregularities are present, since the performance of the results recovery depends on the NFS servers and the network load, and so on the other users activities.

Note that the MEG OS environment, to deploy with KADEPLOY on Grid

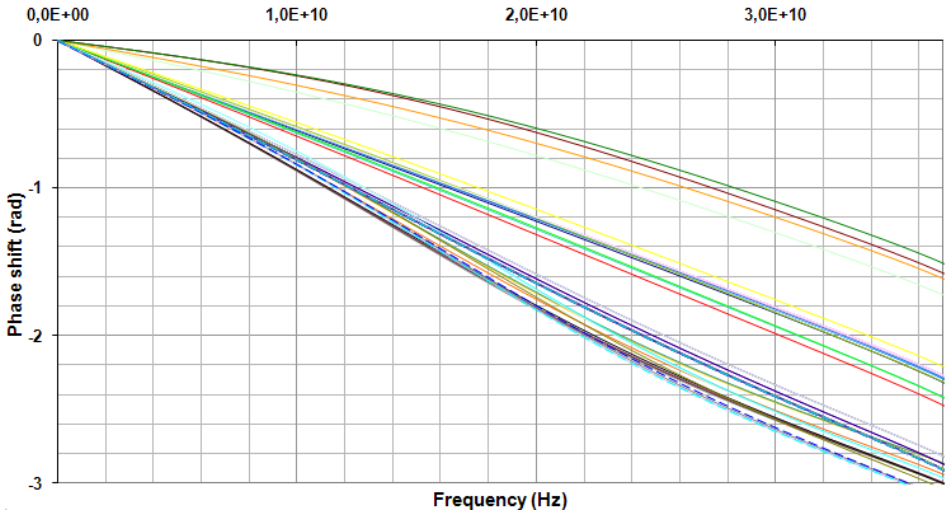


Figure 3.22: Simulated phase shift of the MEMS phase shifter versus frequency.

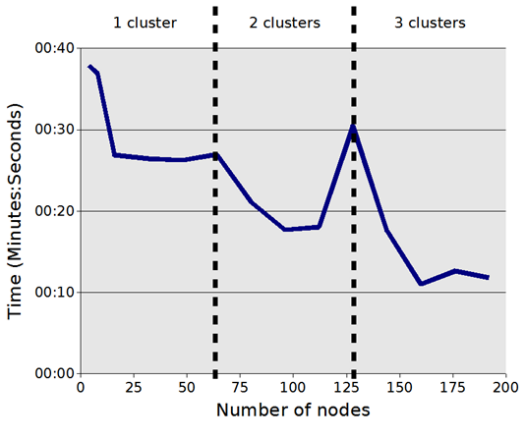


Figure 3.23: Duration of the result recovery phase.

computing nodes in order to run TLM simulations, could be used more easily with MEG GUI, presented later in chapter 4.

3.7.2 Second Approach: TUNe + YATPAC

For the sake of demonstration, a reflectarray made of various rectangular microstrip patches loaded with slots, proposed in [110], is simulated in the band of interest namely the Ku Band (Figure 3.24). Reflectarrays combines key features of large reflectors and phased array elements to generate a collimated beam as required in high gain antennas [111].

Here, the presented non-uniform planar array is illuminated by a normally and linearly polarized incident plane wave (In [110], the primary source is a 12-18 GHz feed horn). The patches are printed on one side of a single dielectric slab of 4

mm thickness with relative permittivity of $\varepsilon = 2.17$. The other side of the slab is completely metalized. The elementary cell is a 16.8 mm x 16.8 mm square containing rectangular metallic patch loaded with a centered slot. The center-to-center distance between elements is 0.7λ. The dimensions of the reflectarray are 7λ x 7λ where λ is the freespace wavelength.

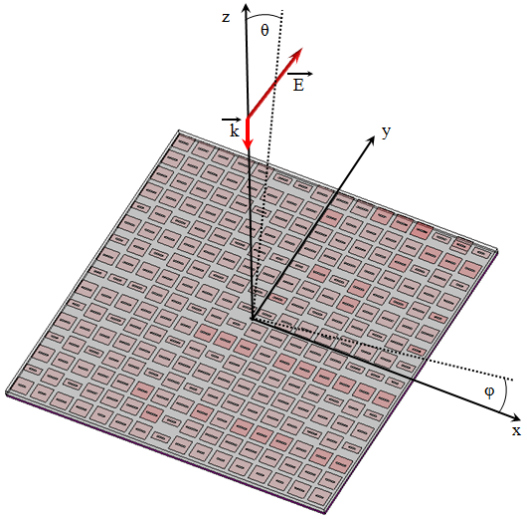


Figure 3.24: Non-uniform reflectarray antenna illuminated by a normally and linearly polarized incident plane wave (Φ = Azimuth angle/+ or - rotation away from the x-axis; Θ = Elevation Angle/+ or - rotation away from the zaxis).

Since the slot aperture lengthens the path of the electric currents on the metallic patch of the array cell, a phase shift is introduced locally for the reflected wave and the main-lobe direction of the backscattered field is then controlled. In order to choose the slot and patch dimensions, and consequently adjust the phase distribution, parametric simulations of the reflectarray can be performed in the design process.

The TLM Method developed here for the full-wave electromagnetic simulation of such reflectarray uses a uniform mesh grid cell dimensions of Δx= Δy= Δz= 0.5 mm. Absorbing boundary conditions are assigned to a large box 7λ x 7λ x λ that artificially encloses the structure.

Parametric studies are used here on geometrical dimensions as well as frequency points of interest to help the designer choosing the most appropriate elementary cell dimensions.

3.7.2.1 TUNe Principle

TUNe is a project that aims to give a solution to the increasing complexity in the domain of distributed software execution management. It is based on the concept of autonomous computing [112]. TUNe can be used in different domains: web architecture, middleware architecture [113], Grid computing. Users give an abstraction

of the platform execution (hardware) and application (software). A subset of UML graphical language is used to describe the global view in a high level of abstraction. The main idea is then to automatically create a representation based on fractal components [114] of the real system, with:

- Application components, also called legacy components because they can wrap legacy software pieces.
- Running platform components, each one representing a node on which the legacy software piece can run.

The application components are connected to the real system with a wrapping language based on XML, describing methods that are reflected on the real system by making action commands. The global dynamic behavior of the system is expressed with UML state charts or activity diagrams, some steps of these diagrams referring to the methods in the wrapping XML files.

3.7.2.2 Platform description

The platform (hardware) is described with a UML class diagram (see Figure 3.25). Each class represents a family of nodes (called host-family). The difference between the families is up to the user level, so different users could have different families. For example, a user can specify with one class one powerful local machine in its network that is not managed (directly accessible without asking a resource scheduler), or one local cluster (a set of directly accessible machines) in the same way. One class can also specify Grid platforms (managed resources, thus they are not directly accessible) which use a resource scheduler to access the machines. Each family has different parameters which specify its particularities on how to get resources, or how to access them. For Grid utilization, the following parameters are useful:

- *type*: this parameter is used in Grid environments. It makes the relation with the batch scheduler. For example with Grid'5000, the oargrid tool is used for multi-clusters on multi-sites reservations, but the OAR tool could also be used for multi-clusters on one geographical site. On the diagram 3.25, the *allsites* class represents a Grid-level reservation using the oargrid tool, and the *toulouse* class represents a site-level reservation of the nodes for the city of Toulouse using the OAR tool.
- *sites*: is a specific parameter for the batch scheduler, representing the sites (city names) and the number of nodes to reserve for each site. An automatic way is actually sought to calculate the best number of nodes for each site based on profiling applications and analyzing their logs.
- *walltime*: is the duration of the nodes reservation, here 2 hours.
- *user*: allows to have different login on different families.

- *keypath*: facilitates the remote login with ssh keys on the Grid to avoid password typing.
- *javahome*: is the directory where java is located for this family. This is the only necessary software to make TUNe work.
- *dirlocal*: is the working directory for the application (where it will be installed and started).
- *protocole*: allows to have different remote protocol connections (local commands, ssh, via oarsh specific to Grid'5000).

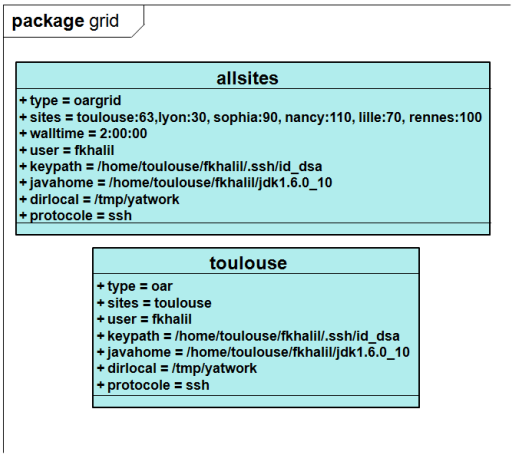


Figure 3.25: Platform description.

With the actual version of TUNe, the javahome property has to be defined on each class of the Grid diagram. Java is indeed needed on each node for the RMI process (java's remote procedure calls) to send events to the TUNe manager eventually from every computing node. If java is not present on the node, TUNe will consider this node in a special deploy error state and will not try to redeploy it. There is an ongoing work on this issue and will use the following steps:

- *1*: use of the javahome property defined in the class of the Grid diagram to find the java binary
- *2*: if the java binary is not present, TUNe will use the JAVA_HOME environment variable of the node
- *3*: if this variable is not set, TUNe will use the first java binary found in the PATH environment variable of the node
- *4*: if the java binary cannot be found in any of these steps, TUNe will engage a java runtime environment copy. The difficult part is to define where and how TUNe will get the good java version for the good architecture.

Anyway, any java runtime version with support of RMI is acceptable for TUNe (for instance any java version superior to 1.4.2).

Altought this concept is not presented here, the user would be capable of describing if he is a member of a Virtual Organization (see Chapter 2). Within the platform description diagram, a VO could be represented by a class. Indeed, the name of the VO would be the name of the class, and the geographical constraints would be represented as for the sites locations for Grid'5000. Furthermore, this diagram could also be extended to take into account specific VO policies constraints. Each constraint would be expressed with a property of the class. These properties could then be used to dynamically generate commands to communicate with particular schedulers to get the resources. This set of specific commands would finally describe a specialized protocol to get the resources, connect to them, as for the specific OAR tools on Grid'5000.

Finally, if the user wanted to target production Grids like EGEE [6], he would have to define a specialized protocol by extending the generic protocol java class within TUNe and write the different commands to get and to connect to the resources (like the specialized protocols for OAR tools on Grid'5000). TUNe is also able to read a set of nodes from the "nodefile". In this case, the user has to construct this file by hand by manually getting the resources from the specific scheduler. Another option is again to describe the reservation mechanism specifying which commands have to be launched and where to get the resources, by extending the generic node java class within TUNe.

For now, the platform description diagram is used to construct the reservation commands; therefore it includes an exhaustive description of the resources allocated by the resource manager. This allows the user to fix a maximum limit for the reservation process and if the application needs more resources, it forces TUNe to aggregate some of the processes on same nodes. An evolution of these diagrams is under development: an improved application description diagram will be used with more specific QoS constraints and the platform diagram will permit a more precise description of the hardware architecture. The purpose is to define the way to calculate the best fit between the application demands and the actual availabilities of the resources.

3.7.2.3 Application description

The goal of this diagram is to describe what should be the running distributed application at its best (with no failures). As for the platform description, it is also based on a UML class diagram. The user has to create the different families of its program. Two kinds of programs can be distinguished:

- the architecture and information specific to its application. This part requires an abstraction work for the user.

[6] Enabling Grids for E-sciencE (EGEE). http://www.eu-egee.org/

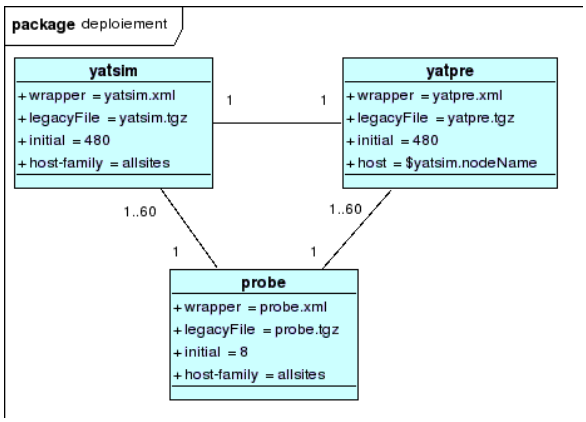


Figure 3.26: Application description.

- the probing system which senses the state of the application and therefore can asks TUNe to change the behavior of the application during its execution by sending events. TUNe provides a set of different probes (to check if a PID is alive, a node is alive) but the users can create their own programs (that will also be automatically deployed by TUNe) to observe specific values of their applications and send events to the TUNe manager in a very simple way: they only have to write into a pipe.

In diagram 3.26, three families of program are described: yatsim, yatpre and a probe that can observe these two programs.
Each class contains TUNe specific parameters or legacy specific parameters. A minimum of parameters for the yatpac simulation are:

- *legacyFile*: is an archive of all necessary files to execute this program.
- *initial*: is an integer which represents the desired number of running processes of this program.
- *host-family or host*: represents how TUNe maps the software component with the platform component. It can be a family in the Platform description (see Figure 3.25) or a specific host in the family. Those values can be estimated by using the batch scheduler or by taking the host name of another running process. Note that the yatpre program should be on the same node than the yatsim program that it is linked to. The yatsim node is therefore chosen by the scheduler and the linked instance of yatpre will also take the same node to execute itself.
- *wrapper* is the name of the wrapping XML file which contains all the methods that can be called for the legacy, and will be explain further.

The user should also create relations between program parts by creating links between them. Those links can also be named, and are useful for two main reasons.

First, it allows a component to get some information (value of parameters, value of TUNe variables) about the components it is connected to. Secondly, it also creates a cardinality between the different programs. In Figure 3.26, it is expressed that when TUNe creates a yatsim program, a yatpre program should also be created and reciprocally (cardinality 1-1). For the probe, it limits for one instance of the probe program the number of observed components. Here there is a maximum limit of 60 yatpre and 60 yatsim for one probe.

3.7.2.4 Behavior description

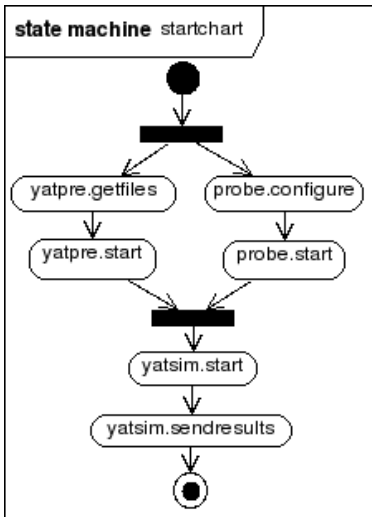


Figure 3.27: Start behavior description.

Two types of diagram are used to express the behavior of the application: state chart diagrams and activity diagrams. Activity diagrams are still under development and allows to create more sophisticated reactions. The minimal number of state chart diagrams needed to start and stop the application are presented here:

- *The startchart*: describes what should be done to start the distributed application, shown by Figure 3.27.
- *The stopchart*: gives the different actions to stop the distributed application.

The user can also create some other state chart diagrams which are run when the TUNe manager receives an event from the probe for example. In this case, the name of the diagram is the name of the received event. That is the case in Figure 3.28 for the state chart "repair", which makes TUNe sensible to the event "repair". This diagram is started when a probe detects a yatpre or a yatsim failed execution. A state chart begins at the initial node and finishes when it arrives at an ending node. Some of the actions can be executed in parallel with the use of fork junctions. The join junctions wait for the different forked paths to arrive before the execution

of the diagram can be continued.
State charts use a language to manipulate the components and to call the methods in the wrapping XML file. The first part of a command represents one component or a set of components. For instance, it can be a family name (in Figure 3.27 "yatpre"), a variable defined by TUNe (in Figure 3.28 "arg" which is an argument passed to the state chart) or by the user (in Figure 3.28 "new" represents the last component created by the user). The second part of a command is the action to be executed on these components. It can be a TUNe operator (like "−−" in Figure 3.28) or a call to a method defined in the wrapping XML file (in Figure 3.27 "sendresults"). A more detailed execution is presented in the Experiments section.

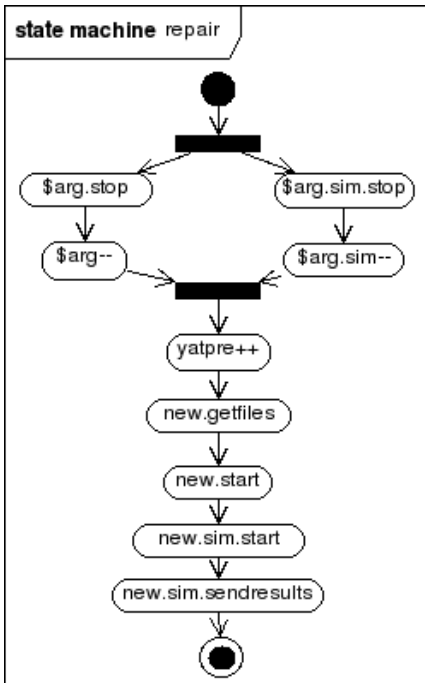


Figure 3.28: Example of dynamic behavior description.

3.7.2.5 Wrapping files

One of the most promising approaches with TUNe is the encapsulation mechanism of legacy software into generic wrappers. This permits any legacy software to be wrapped without modifying any line of its code. The user simply has to describe the wrapper using a simple XML syntax. However this limits the controls on the component: basic controls are starting the component, stopping it and dynamically generate configuration files. Other available controls of the legacy can be described, but TUNe does not offer a controller API to integrate to the codes of the legacies. First, the wrapper is associated to a class with the "wrapper" parameter of the application diagram. All the methods are listed and the commands to run effectively on the nodes, including their parameters, are described. The character $ allows to

get the different values from component instances of the class diagram, or from internal variables of TUNe.

example of wrapping file:

```
<wrapper name='yatsim'>
  .....
  <method name="sendresults"
  key="extension.GenericCommands"
  method="start" >
    <param value="./sendresults.sh"/>
    <param value="\$dirLocal"/>
    <param value="\$node.user"/>
    <param value="\$node.keypath"/>
    <param value="\$nodeName"/>
  </method>

  <method name="start"
   key="extension.GenericCommands"
   method="start" >
    <param value="./yatsim Temp"/>
    <param value="\$dirLocal"/>
    <param value="\$node.user"/>
    <param value="\$node.keypath"/>
    <param value="\$nodeName"/>
    <param value="LD_LIBRARY_PATH
      =\$dirLocal"/>
  </method>
......
</wrapper>
```

3.7.2.6 Experiments

Experiment conditions

In this section, a short review of all the steps that the end user has to achieve in order to launch effectively a yatpac simulation with TUNe. These steps include:

- *Preparing locally the simulation*: First of all, the user has to create compressed archives (tgz) containing the legacy binary files and all the libraries they use. A recursive ldd script copies all the libraries used by a binary, including libraries used by libraries, and creates the archive. The generic probe.tgz to detect the presence of the pid process has been chosen in the following simulations. The user also has to prepare all the necessary input files for its simulation and number them.

- *Describing the application for TUNe*: the user creates the three types of diagrams for TUNe: platform description (Figure 3.25), application description (Figure 3.26), and behavior description (Figures 3.27 and 3.28) using a graphical UML editor like Eclipse or Topcased. He then writes the wrapping files to describe what commands should be launched by the behavior description diagrams.

- *Launching TUNe*: The user has to send all the files (application files, TUNe software and the UML file containing the diagrams) to the Grid, for instance the frontnode of its site on Grid'5000. TUNe can then be launched manually and it prompts the user for the commands interactively, or it can be launched in remote control mode. In this case, it can be controlled using telnet or writing into a pipe. To deploy and start the application, one simple command is given to TUNe: `deploy <path to application files> <uml file>`.

Two experiments were conducted with different numbers of sites and nodes as shown by table 3.2. The first experiment is a small scale deployment, concerning 8 sites and 20 nodes on each site. The second experiment is a large scale deployment, concerning 6 sites and between 30 to 110 nodes on each site. For these experiments, the purpose is to show how fast TUNe can deploy and start the simulation on all nodes, and also how fast it can undeploy everything. To do this, the sites are gradually added one by one, relaunching TUNe everytime a new site is added in order to always mesure the global times. Thus, the first step will concern only one site, the second step two sites etc. On the following figures 3.29, 3.30 and 3.31, the first point of an experiment represents the first step (deployment on one site) and the last point represents the last step (deployment on all sites). To choose which sites are concerned, there is only need to modify for each step the platform description diagram, adding one site to the list of sites (Figure 3.25), and changing the initial parameter values of the application diagram to the total number of nodes. This is a very simple way to add or remove one site without touching any other file or configuration.

Exp	Rennes	Nancy	Nice	Toulouse	Lyon	Lille
exp 1	20	20	20	20	20	20
exp 2	100	110	90	63	30	70

Table 3.2: Nodes and location

Note that the sites of Paris and Bordeaux are also used with 20 nodes in the experiment 1.

Deployment

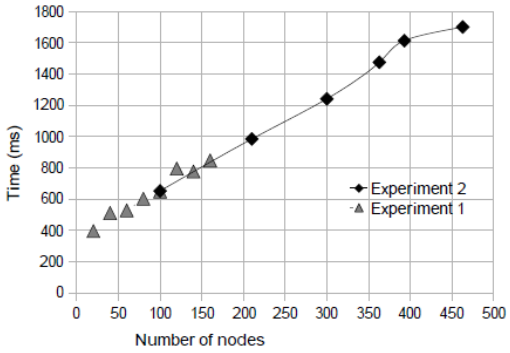


Figure 3.29: Creation of system representation.

The deployment consists of three phases:

- *The creation of the internal system represention of TUNe*: illustrated by Figure 3.29.
- *The deployment of the application on the nodes*: shown in Figure 3.30.
- *The initialisation of the software components*: shown in Figure 3.31.

Initialization of the deployment

When TUNe receives the "deploy" command, it parses the UML file and eventually sends a request to the resource scheduler of the Grid if the resources are not directly listed in a nodefile. For this experiment, the Grid'5000 OAR resource scheduler has been actually used, that's why on Figure 3.25 the type is equal to oargrid. This means that TUNe will get the resources needed using the oargrid tool. Note that the time spent by the oargrid tool to achieve TUNe's request has not been measured. Indeed, depending on the tool used to schedule the Grid resources and depending on TUNe's demand, the time can vary a lot and the purpose is to show TUNe performances. The time was measured as soon as all the resources needed are taken.

Moreover, the TLM simulations deployed include time range and time step parameters that can vary from one simulation to the other. For instance, the user can specify a less time precision for some area of the electronic circuit. He can therefore focus on some interesting parts of the electronic circuit and ask for a very precise time step and time range on a specific area. Depending on these parameters, the running time for one simulation can vary from 5 minutes to a few hours.

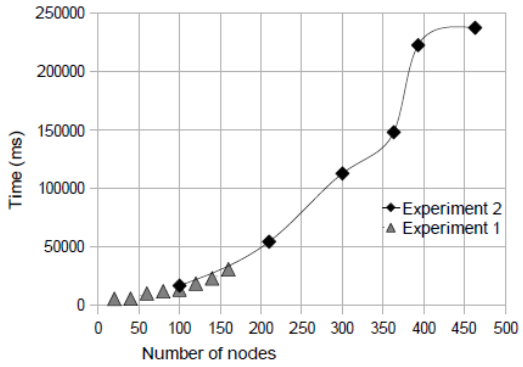


Figure 3.30: Deployment of application (Table 3.2.

Creation of the internal system represention of TUNe

Figure 3.29 shows how much time TUNe spends in creating its internal representation of the global system to be deployed. This represents the time needed by TUNe to create an instanciation of the application description diagram (Figure 3.26) mapped to the platform description diagram (Figure 3.25). For each instance of the classes in the application desciption diagram, TUNe creates a fractal component. All components are then linked in accordance with the links between the classes and their cardinalities. The same goes for each node, TUNe creates the fractal components in accordance with the result of the oargrid resource scheduler. All node components are then linked to software components depending on the host-family or the host parameter of the classes. Figure 3.29 clearly shows that the time needed by this process varies in a linear manner depending on the number of nodes involved, starting at 400 ms for 20 nodes on one site up to 820 ms for 160 nodes on 8 sites (experiment 1) or from 650 ms for 110 nodes on one site up to 1700 ms for 460 nodes on 6 sites (experiment 2). Note that the nodes could be monocore or multicore (dual, quad).

Deployment of the application on the nodes

Figure 3.30 shows the time needed by TUNe to deploy the application without launching it. This includes: the time to create the remote working directory (very fast), the RMI communication sockets (java's remote procedure call system) in order to communicate between the node and TUNe, the archive copy and decompression.

The two latters share their time usage depending on the archive size and the decompression speed on the node. Around 80% are observed for the creation of the TUNe communication system and 20% for the archives copy

and decompression if the total archives sizes were less than 1Mb (which was the case for these experiments with a total of 750k). This time sharing can go up to 95% for the copy and decompression of big archives (100Mb and up) on the Grid'5000 network (10Gb/s links between most nodes).

On Figure 3.30, notice that the total time for the deployment on 20 nodes on 1 site is 5500 ms and goes up to 30800 ms for 160 nodes on 8 sites (experiment 1). This time is 16800 ms for 100 nodes on 1 site and 237500 ms (4 minutes) for 463 nodes on 6 sites (experiment 2). It can be also observed that the two experiments stick together between 100 and 160 nodes and that in average, the deployment time increased in an exponential way. This is due to the fact that the deployment is multithreaded and multiple archives copies (using scp) are performed in parallel, as well as multiple tcp sockets initiations.

Note that for a massive deployment, the node on which TUNe is working has to have a high limit for maximum tcp concurrent connections (this can be configured in the OS kernel and must be higher than the number of nodes to be deployed). Note also that this node must have an increased tcp timeout value to overpass the possible congestion when a large amount of tcp connections are initiated in parallel. One possible way to improve TUNe's performance would be to automatically find the best number of parallel copies to maximize the throughput, or even better, to use multiple nodes as sources for the copy of files.

It could be possible to integrate a more scalable copy tool like taktuk [115] to broadcast the files on the nodes from multiple nodes. The first way to use taktuk with TUNe would be to launch one kanif command instead of launching multiple scp commands. One other possible way to use taktuk or kanif within TUNe would be to define a class in the application description diagram with this tool. This tool would be encapsulated in a generic component and could even be deployed on Grids without it installed. The starting diagram would then first use this tool to deploy the entire system and continue normally. Thus, the deploying tasks and their burden would be given to this tool.

Notice that depending on the simulation parameters, their times can vary from 5 minutes to a few hours. Given these times and the deployment times leads to conclude that if all the deployed simulations have a minimum running time of 5 minutes, TUNe will add a significant overhead if all the nodes are used (45% of the global simulation time is used by TUNe to deploy it on 463 nodes). In this case, the user is recommanded to aggregate some of the simulations on some nodes. Indeed, it is possible to limit the number of nodes in the Grid diagram while the number of simulations is kept unchanged. This forces TUNe to copy the simulation program one time on one node and to copy multiple input files on that node. Therefore, the global simulation time would increase (because of the aggregation) but the deployment time would

decrease, and as a result the overhead induces by TUNe would also globally drop. Still given these times, it can concluded that if all the simulations have a maximum running time of 6 hours, TUNe do not add a significant overhead if all the nodes are used (1, 1% of the global simulation time is used by TUNe to deploy it on 463 nodes).

In section 3.7.1, a script was created to deploy quite similar simulations with identical inputs. The deploying times were a little bit shorter but very comparable to those measured with TUNe. However, the large benefit of using TUNe comes from the ease of use with UML diagrams and automatic repair of crashed nodes.

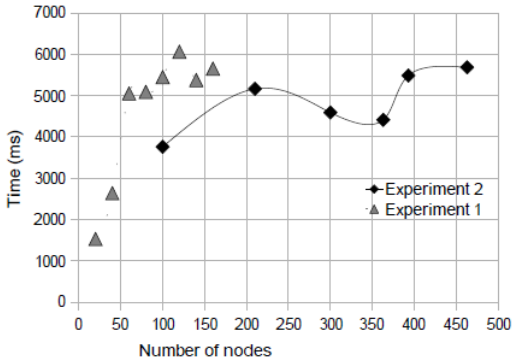


Figure 3.31: Starting application.

Initialization of the software components

Figure 3.31 shows how much time TUNe spends in sending which commands to execute on which nodes to start the application. This includes the time needed by TUNe to generate the commands from the wrapping files (including the time to interpret the variables with their dynamic values), to generate the configuration files, to connect to the nodes, write the configuration files and launch the commands. Note that the electromagnetic simulation times or other times spent by the legagy applications are not included, but only the times spent by the TUNe processes to launch the commands. This time goes from 1500 ms for 20 nodes on 1 site to an average of 5000 ms if the number of nodes is greater than 60 on any number of sites. This means that this time does not increase dramatically with the number of nodes involved, unlike the time of deployment. It can however be observed that for the same number of nodes the starting time is lower when the nodes are concentrated on a lower number of sites. For example, with 100 nodes on 1 site (first point of experiment 2), the starting time is 3760 ms whereas with 100 nodes on 5 sites (fifth point of experiment 1) the starting time is 5440

ms.

Undeploying times

Figure 3.32 shows the time needed by TUNe to undeploy completely the application. This includes the time to clean working directories and cleanly stop all the network connections. This figure shows that this time increases with the number of nodes involved, starting from 349 ms with 20 nodes on 1 site to 7845 ms with 160 nodes on 8 sites for the experiment 1, and from 2863 ms with 100 nodes on 1 site to 32568 ms with 463 nodes on 6 sites for experiment 2. As for the deployment, an important number of tcp connections are initiated during this phase.

Note again that the time to retreive the simulation results is not taken into account as it can vary a lot depending on the output files size created by the application deployed. With TUNe, it is possible to use pulling or pushing for the result retrieval. In the pushing case, and using the application description diagram, it is possible for the user to define a different target for a set of nodes. For example, the 50 first simulations can push the results on a certain node, the 50 next on another node etc. It is also possible to define a single target for each site, and the nodes from that site would push the results to that particular node. Taktuk could again be use for a more scalable copy.

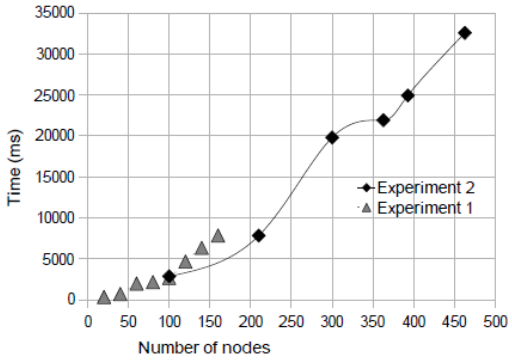


Figure 3.32: Undeployment of application.

Simulations results

Parametric studies are used here on geometrical dimensions as well as frequency points of interest to help the designer choosing the most appropriate elementary cell dimensions. In this section, an example of numerical results obtained using the Grid-based approach for electromagnetic simulations of the

reflectarrays in Figure 3.24 is presented.

The E-phi and E-theta simulated components of the backscattered electric field are used to visualize the radiation pattern.

Note that the calculation of near field values and the near field-to-far field transformation are not delivered in the YATPAC installation source. These features have been adapted and validated during the thesis due to the collaboration of Dr. Petr Lorenz (one of the developers of the Yatsim codes).

Figure 3.33 shows the radiation patterns in the case of the non-uniform reflectarrays shown in Figure 3.24 simulated at a single point of frequency.

These reflectarrays have different patch dimensions configurations, 13.5 mm in x-direction, 2 mm to 12 mm in y-direction, while varying slot dimension in x-direction from 2 mm to 12 mm and keeping on 1 mm in y-direction. The design of such arrays used to steer the main reflectarray beam in a specified direction is detailed in Appendix C.

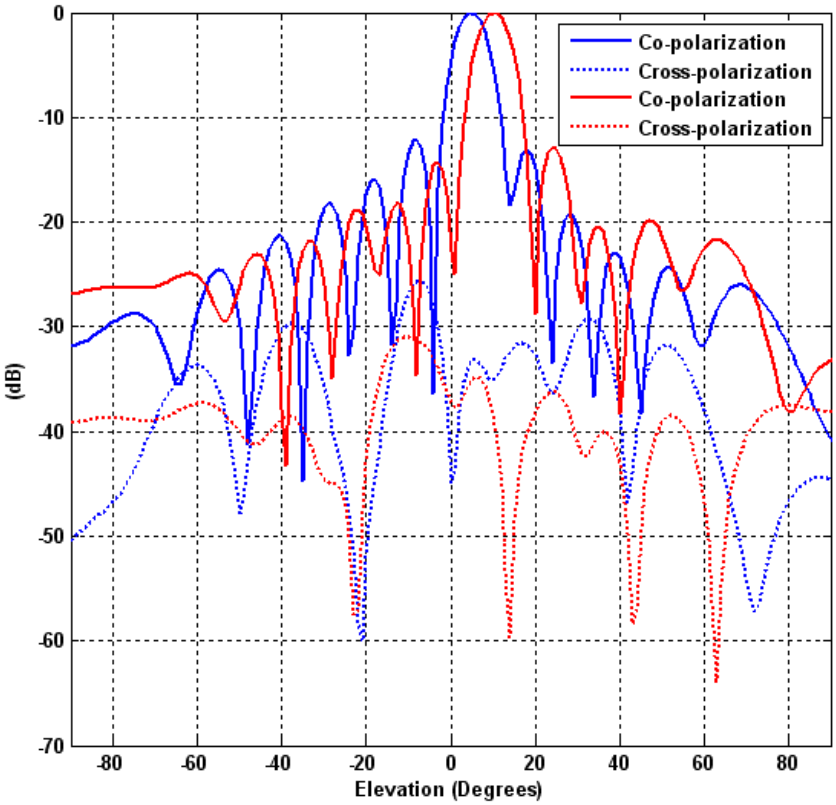


Figure 3.33: The simulated radiation patterns in H-plane for two non-uniform reflectarrays at 12.5 GHz.

The results of Yatsim are found to be in good agreement with HFSS [116] simulation results. The radiation patterns of Figure 3.33 show that the main beam position steers toward 5° and 10° (Φ =0° cut).

Repair

It has been shown that TUNe is a software that can be used to deploy and configure applications in a distributed environment. It can also monitor the environment and react to events such as failures or overloads and reconfigure applications accordingly and autonomously. Management solutions for legacy systems are usually proposed as ad-hoc solutions that are tied to particular legacy system implementations.

This unfortunately reduces reusability and requires autonomic management procedures to be re-implemented each time a legacy system is taken into account in a particular context. Moreover, the architecture of managed systems is often very complex (e.g. multi-tier architectures), which requires advanced support for its management. Still to propose a high level interface, a UML profile for specifying reconfiguration policies is used. The main benefit of this approach is to provide a higher level interface to the software environment administrator to autonomously manage its distributed applications.

Indeed, when deploying on a large amount of nodes a yatpac simulation that can run for hours, it often happens that some of the nodes crash or some of the processes hang. In the experiments presented above, it is possible to restart one simulation with the same input files in case of failure because the simulation tasks do not communicate between them and all the input files are not collerated. In certain conditions, TUNe is able to restart communicative tasks (for web services architectures for example).

In order to detect these failures, the monitoring system consisting of a number of generic probes given by default by TUNe was used. These probes regularly connect to the nodes they are in charge and scan for the PID of the processes to see if they are alive and not in a blocked state. When a simulation finishies successfully and the results have been sent back, the monitoring is stopped by unregistering its PID. When such failures occurs, a *repair* event is sent to TUNe with the name of the component involved in the failure, and the corresponding state chart being executed. For instance if a yatsim process is dead, the probe sends the *repair* event to TUNe with the instance name of the yatsim process involved, and the state chart shown in Figure 3.28 is executed. Following this chart, it could be noticed that TUNe tries to stop the failed process ($arg.stop) and tries to stop the process connected by the sim link on the class diagram ($arg.sim.stop, the *sim* link is shown on Figure 3.26). Indeed, if a yatpre failure occurs, then the yatsim will fail anyway so it's better not to start the yatsim process in this case. Then TUNe tries to undeploy the yatpre and yatsim files ($arg−− and $arg.sim−−), as well as cleaning completely the node. Anyway if it was a node failure, all of that would have no consequence and the statechart would continue normally.

After the join junction, the next action is *yatpre++*, that creates and deploys a new yatpre component and also creates and deploys a new yatsim component because they are connected with a *1-1* cardinality. *New.getfiles* gets the input files for the newly created simulation and *new.start* is the equivalent for *yatpre.start* in the startchart diagram (Figure 3.31). The same goes for *new.sim.start* that is the equivalent of *yatsim.start* and *new.sim.sendresults* that is the equivalent of *yatsim.sendresults*.

Different time measurements for a repair process in real conditions are presented in table 3.3. To get these values, a probe was modified to communicate with it during the execution process and simulated a node failure by forcing the event node failure by hand. However, there is regularly a difference between the number of nodes asked to the scheduler and the number of nodes actually ready to receive the simulation binaries and input files. For instance, during the experiments, when TUNe asked for 105 nodes, it received 100 ready nodes. The same went when TUNe asked for 480 nodes and received 463 ready nodes. In that case, TUNe continued the deployment normally but the probes would eventually detect those failures and TUNe would redeploy the failed parts on new nodes. On the other hand, process hangs were very rare cases (1 of 463 nodes on long simulations running times) and often related to hardware overloads. The oargrid request time is also consider here in order to get an idea of the global repair time in real conditions. The average total time measured for one repair process is around 12 seconds.

action	time (ms)
undeploying	683
component removals	126
oargrid request	4904
component creation	167
deployment	4525
start	1855

Table 3.3: Time measurement for repairing the application for the given experiment.

3.7.3 Third Approach: TUNe + emGine environment

emGine environment (Appendix B), based on TLM modeling method, present more advantages than YATPAC. Among these, the most important is that it have a friendly-user interface that allow user to build the model and visualize the results of time-domain simulations in the GUI in real time. Besides, tlmGine exists in 32-bit and 64-bit versions (while yatsim could only be compiled on 32-bit computer) which is suitable with the heterogeneity of Grids. Note that YATPAC is not maintained or updated since 2006 while emGine environment is under continuous devlopment.

Using emGine with TUNe gives the user a very powerful tool for electromagnetic simulations in Grid computing environment but at the same time too easy to use. In fact, from the use point of view, the user does not feel the difference between local simulations and remote simulations. The simulation flow (described in Appendix B) is the same for distributed simulations: prepare EMM file, simulate with ***tlmGine***, and visualize the time domain signals with the GUI.

In order to control TUNe (defined in previous sections), a folder named emCluster is placed in the Grid'5000 user home directory (see Figure 3.34). The main component is *emServer* which is the entry point for the user (it defines a base class emServer). The *emServer* is command line driven. The user can start/stop and watch progress of ***tlmGine*** simulations.

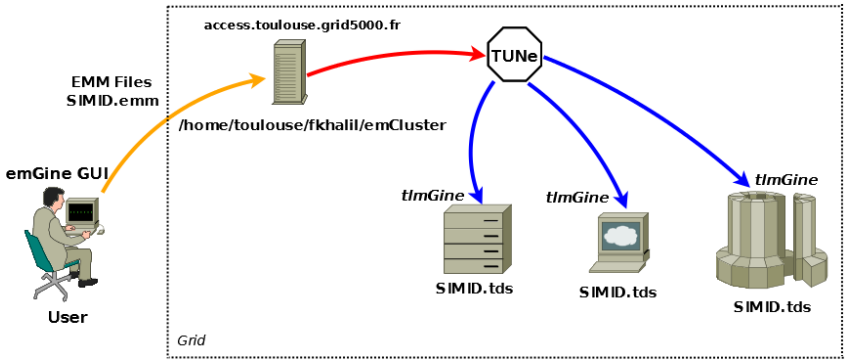


Figure 3.34: Architecture of the proposed approach.

First user prepares the model he wants to analyze (EMM file) with emGine GUI on local machine. To launch the simulation on Grid nodes, "remote simulations" has to be chosen in the menu instead of local one (see Figure 3.35).

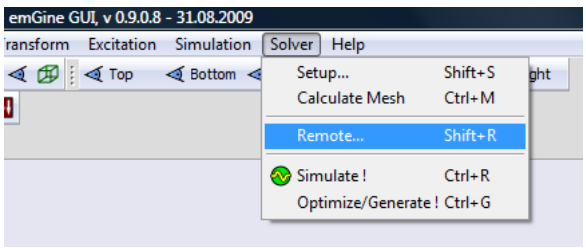


Figure 3.35: Distributed computing menu.

User enters its Grid'5000 login informations into specific fields (Figure 3.36). Three simple lines are only needed to specify the Grid'5000 front-end (that contains user home directory and ***emCluster*** files), login and the path

to its public key file [7].

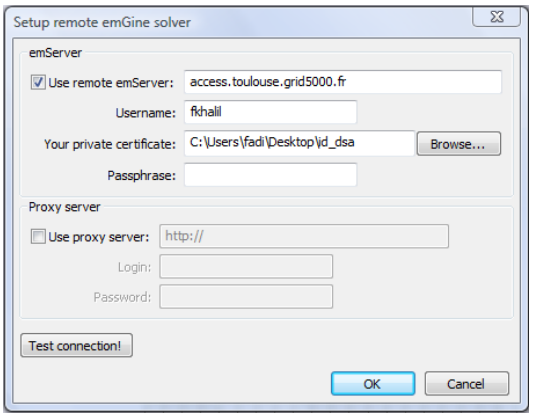


Figure 3.36: Grid'5000 Autentication.

Before clicking on the execution button, user has the possibility to check the network connection between its local computer (where emGine GUI is running) and the Grid (see Figure 3.37).

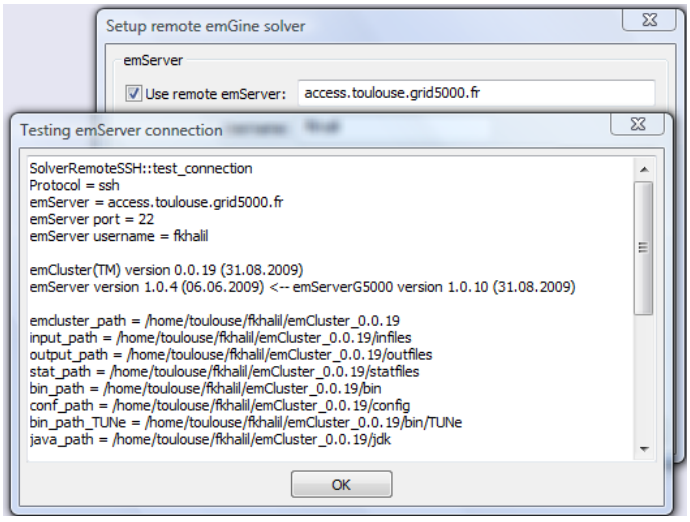


Figure 3.37: Test of the network connection between local computer (where emGine GUI is being used) and the Grid.

The TLM simulations executing on Grid'5000 nodes can be monitored on user local computer screen in real time (see Figure 3.38).

The TDS files (SIMID.tds in Figure 3.34) are redirected to the local emGine GUI in order to visualize the voltages, S-parameters and other sim-

[7] The user must have an ssh access to the front-node configured with a public/private key without password, so that ***emServer*** can be called without entering a password each time.

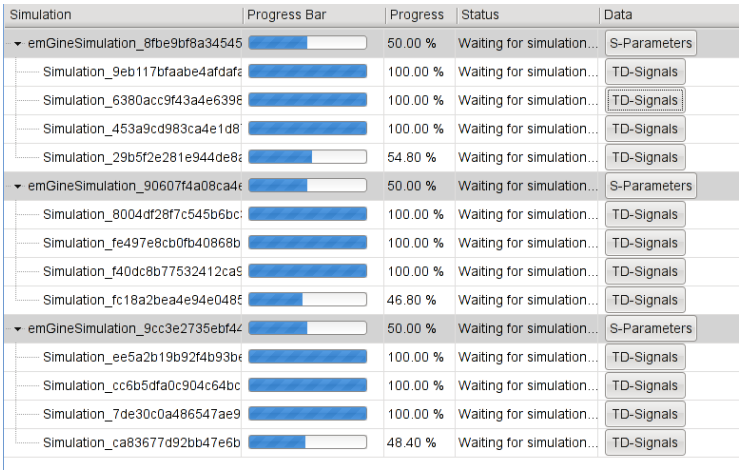


Figure 3.38: Progress visualization of remote simulations on local GUI.

ulated data. User needs only to click the button of S-Parameters of the concerned SIMID simulation.

The parallel execution of tlmGine is based on "Startchart" and "Stopchart" diagrams which are very simple. Once the nodes are reserved, "Startchart" copies TUNe and tlmGine on the nodes, and launches the execution. A special deamon monitors the execution and progess. When the execution is finished, this deamon start the "Stopchart" that copies all the needed output results in the user directory before the node is liberated (the restart on default OS cleans all the folders).

3.8 Conclusion

In this chapter, the use of TLM modeling method in Grid environment has been discussed. Powerful developed tools for distributing parametric TLM applications on Grid'5000 nodes have been presented. The efficiency of these tools, with friendly interfaces, have been demonstrated in the simulation of real life electromagnetic structures. Using a large number of nodes enables to decrease significantly the execution time of the simulations. Distributing the simulations on several sites does not have an impact on the simulations execution time, since simulations are independent and do not communicate between each other. However, it enables to optimize the input files deployment by decreasing the network bandwidth consumption and file server requests for each site.

Due to the nature of computations, TLM algorithm is not trivially parallelizable, as data dependency inside the meshes implies communication. To avoid time losses in communications, it would be better to split the TLM computational region into big subregions, to perform the TLM algorithm inside

each subregion independently of other subregions and to exchange the values on the boundaries common to the subregions.

Chapter 4

SCT Modeling Method in Grid Environment

Contents

The Scale Changing Technique (SCT) [117] is an efficient monolithic (unique) formulation for the electromagnetic modeling of complex (multi-scale) structures i.e., structures that exhibit multiple metallic patterns whose sizes cover a large range of scales.

The SCT Method will be briefly introduced in the following section. After that, the algorithm and the decomposition of SCT programs will be detailed. The deployement of a SCT application on Grid'5000 nodes shows then the different proposed distributed computing approaches.

4.1 Overview of the SCT Modeling Method

Nowadays global electromagnetic simulators are essential for accurate predictions of the overall electromagnetic performances of radiofrequency systems. With the increase demand of miniaturization of systems, complex structures involving both large structures (in terms of wavelength) and fine details could be found in most of modern designs. The complexity of the problem depends usually on the ratio between the biggest scale and the smallest scale. Well-known examples of complex structures are provided by multi-band frequency-selective surfaces, finite-size arrays of non-identical cells and fractal planar objects.

SCT consists of interconnecting Scale-Changing Networks (SCN) (Figure 4.1), each network models the electromagnetic coupling between adjacent scale levels [117]. The cascade of Scale Changing Networks allows the global electromagnetic simulation of multi-scale structures, from the smallest to the highest scale.

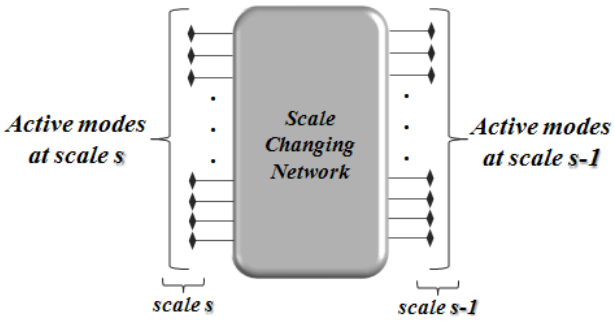


Figure 4.1: The Scale-Changing Network modeling the electromagnetic coupling between the scales s and $s-1$.

Consider a Frequency Selective Surface (FSS) consisting of non-uniform rectangular apertures or waveguides perforating a perfectly conducting and thick metallic plate (see Figure 4.2), illuminated by a normal incident plane wave. The frequency selective surfaces play a key role in many antenna systems for modern fixed and mobile communication services. As mentioned in [118, 119] applications include radomes, frequency separation in quasi-optical beam splitters, Cassegrain reflectors, and phase screens for beam steering. FSS are used in various applications to act as spatial filters, allowing transmission at certain frequencies and reflection at others.

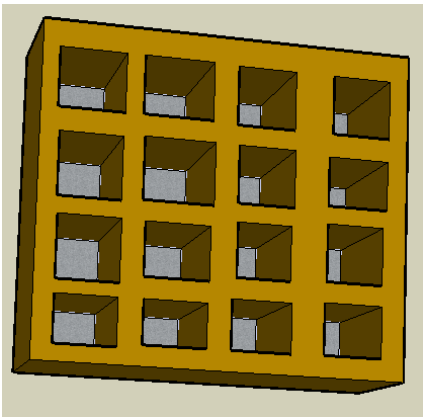


Figure 4.2: FSS consisting of non-uniform rectangular waveguides perforating a thick metallic plate.

The SCT is applied here to calculate the transmission and reflection coefficients of this finite size non-uniform FSS.

In such structures, a high scale ratio exists between the highest and the smallest dimensions in the discontinuity plane. The starting point of the proposed approach consists of the coarse partitioning of this (complex) discontinuity plane into large-scale (called scale level s_{max}) sub-domains of arbitrary shape; in each sub-domain a second partitioning is then performed by introducing smaller sub-domains at scale level $s_{max} - 1$; again, in each sub-domain introduced at scale level $s_{max} - 1$ a third partitioning is performed by introducing smaller sub-domains at scale level $s_{max} - 2$; and so on ...

Such hierarchical domain-decomposition, which allows to focus rapidly on increasing detail in the discontinuity plane, is stopped when the finest partitioning (scale level s=0) is reached.

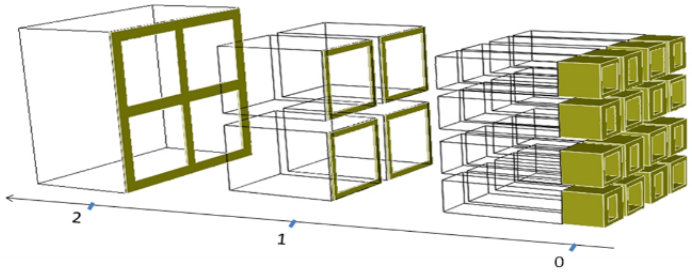


Figure 4.3: Partitioning into multiple scale levels of the discontinuity plane.

An illustration of the partitioning of a metallic grid of 16 cells is sketched in Figure 4.3. Three different scale levels exist when cells are grouped by four.

Boundary conditions are artificially introduced at the contour of all the sub-domains generated by the partitioning process. The physics of the problem may be incorporated into the choice of these conditions in order to avoid perturbation of the actual electromagnetic field.

In practice a type of boundary conditions (e.g., perfect electric or magnetic conducting condition) has to be tried on the contour of each sub-domains and the quality in terms of accuracy, execution time, numerical convergence of the numerical solution has to be checked a posteriori (see section 4.3).

Multi-modal sources, called Scale-Changing Sources, are artificially incorporated at all scale levels for the derivation of the network. When the complex surface presents both large regions and fine details - but no structures at intermediary scale levels-, mono-modal sources are able to model the electromagnetic coupling between the disparate scale levels [120] - [122].

However, for objects involving multiple structures whose size covers a large range of scale, mono-modal sources fail to provide accurate numerical results while the Scale-Changing Sources allow the modeling of the scale crossing

from the smallest to the highest scale (the number of modes in these sources can be derived from numerical convergence criteria).

The global electromagnetic simulation of multi-scale structures via the cascade of Scale Changing Networks has been applied with success to the design and electromagnetic simulation of specific planar structures such as reconfigurable phase-shifters [123, 124], multi-frequency selective surfaces [125], discrete self-similar (pre-fractal) scatterers [126, 127] and patch antennas [128, 129]. This Scale-Changing technique is a very fast technique and this makes it a very powerful investigation, design and optimization tool for engineers who design complex circuits (see, e.g., [130] - [132]).

4.2 Distributed Parallel SCT Simulations in Grid Environment

Distributed parallel and parametric SCT applications based on arrays [133] - [135] and multi-frequency FSSs [136] have been presented. The SCT application shown here consists of a planar array of 256 (16x16) non-uniform cells illuminated by a normally and linearly polarized incident plane wave. This structure is almost the same presented in chapter 3, but the elementary cell here is a 16.8 mm x 16.8 mm square containing rectangular metallic patch (with no slots). The center-to-center distance between elements is 0.7λ. The radiation patterns of this structure will be simulated with the SCT method.

4.2.1 Optimization of SCT Computing Codes

In order to parallelize the SCT applications, the programs must be analyzed to localize parts of codes that can be run in parallel (in other words identify the SCT modules), and others that must be run in serial. During this phase of analyze, an optimization of serial codes have been done.

The SCT was implemented on digital computers as MATLAB [137] in-house codes [1]. Since these programs were focused on the "translation" of mathematical equations and physical problems, they were not optimized to run efficiently even in sequential (serial) execution on a single computer.

These programs often have several layers which might be calling other time-consuming functions that can be several layers down in the code. In this case it is important to determine which functions are responsible for such calls. Besides, because memory performance is not increased at the same rate as CPU performance, code today is often "memory-bound", its overall performance limited by the time it takes to access memory.

In order to debug and optimize the SCT programs, a profile operation have been realized to track their execution time. For each function and subfunc-

[1] developed actually by LAAS-CNRS PhD students Aamir Rashid and Euloge B. Tchikaya.

tion in the code, information about execution time, number of calls, parent functions, child functions, code line hit count, and code line execution time have been recorded.

Once identifying which functions are consuming the most time, the reason of calling them is determined and possible ways to minimize their use and thus improve performance were sought.

This profile step helped to uncover performance problems that it can solve by avoiding unnecessary computation (which can arise from oversight), re-computation by storing results for future use, and to change the algorithm to avoid costly functions. Storing and accessing the data in columns helps also to improve the speed of the execution.

After some modifications (i.e. pre-allocating arrays before accessing them within loops, avoiding creating unnecessary variables), the code was optimized as much as expected, and the execution time of the same program was reduced by 7% (compared to original one executed on same workstation).

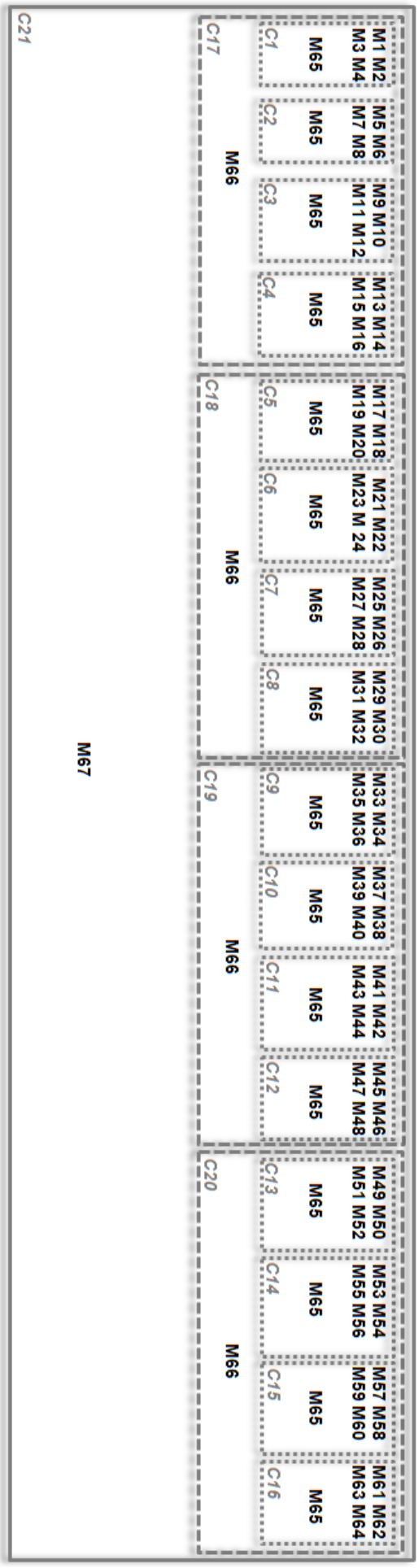
4.2.2 SCT Algorithm

First, the dimensions of the studied structure, frequency of work, number of modes calculated in convergence study must be defined. After this initialization phase, the different SCNs are computed independently. They are represented by the computing modules M1, M2, ..., M64, M65, M66 and M67 in Figure 4.4.

Once these modules are computed, the global electromagnetic simulation of multi-scale structures is done via the cascade represented by C1, C2, ..., C20 and C21. C1, ..., C16 are executed at the first level, C17, ..., C20 at the next level and C21 at the final one. The number of cascades and SCNs depends on the partitioning of the problem and chosen sub-domains defined by user.

SCT program serial execution:

```
Initialization
M1
M2
:
:
M67
C1
:
C21
postprocess (optional)
```


C21
M67
C17
M66
C1
M65
M1 M2
M3 M4
C2
M65
M5 M6
M7 M8
C3
M65
M9 M10
M11 M12
C4
M65
M13 M14
M15 M16
C18
M66
C5
M65
M17 M18
M19 M20
C6
M65
M21 M22
M23 M 24
C7
M65
M25 M26
M27 M28
C8
M65
M29 M30
M31 M32
C19
M66
C9
M65
M33 M34
M35 M36
C10
M65
M37 M38
M39 M40
C11
M65
M41 M42
M43 M44
C12
M65
M45 M46
M47 M48
C20
M66
C13
M65
M49 M50
M51 M52
C14
M65
M53 M54
M55 M56
C15
M65
M57 M58
M59 M60
C16
M65
M61 M62
M63 M64

4.2.3 Parallel Model

One of the first steps in designing a parallel program is to break the problem into discrete "chunks" of work that can be distributed. This is known as decomposition or partitioning. A basic way to partition computational work among parallel tasks is the functional decomposition (Figure 4.5). In this approach, the focus is on the computation that is to be performed rather than on the data manipulated by the computation.

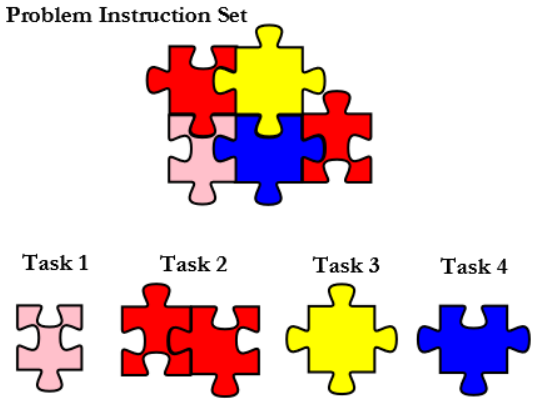


Figure 4.5: Example of functional decomposition where the problem is decomposed to four tasks.

The computing modules M1, M2, ..., M64, M65, M66 and M67 represent individual tasks and are totally independent, and therefore can be potentially run simultaneously (in parallel) on different nodes. The cascade modules, since they present inter-dependencies, represent for the moment the serial part of SCT program.

Several programming paradigms are commonly used to develop parallel programs on distributed architectures. In this case, the *master-worker* model may be a suited programming paradigm. The Master-Worker paradigm (also known as task farming) is especially attractive because it can be easily adapted to run on a Grid platform.

The *Master-Worker* paradigm consists of two entities: a *master* and multiple *workers* (Figure 4.6). The *master* is responsible for initializations, decomposing the problem into small tasks (and distributes these tasks among a farm of *worker* processes), as well as for gathering the partial results in order to produce the final result of the computation. The *worker* processes execute in a very simple cycle: receive a message from the *master* with the next task, process the task, and send back the result to the *master*.

Usually, the communication takes place only between the *master* and the *workers* at the beginning and at the end of the processing of each task. This means that, *master-worker* applications usually exhibit a weak synchroniza-

tion between the *master* and the *workers*, they are not communication intensive and they can be run without significant loss of performance in a Grid environment.

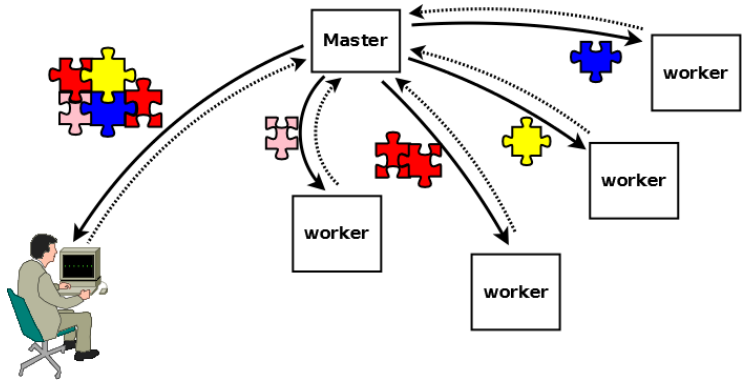


Figure 4.6: Master-worker model. The master is responsible for partitioning the domain and distributing subdomains to workers. Workers elaborate in parallel the subdomain they have been assigned and return results to the master, which elaborates the returned results in a form acceptable for the end-user.

Due to these characteristics, this paradigm can respond quite well to an opportunistic environment like the Grid. The number of workers can be adapted dynamically to the number of available resources so that, if new resources appear they are incorporated as new workers in the application. When a resource is reclaimed by its owner, the task that was computed by the corresponding worker may be reallocated to another worker.

A Grid application must be portable and license-free in order to be deployed on the nodes. As mentioned before, the SCT code is written with MATLAB (a registered trade mark of The MathWorks Inc.). In order to be installed and used, this software needs once license per computer. When the SCT independent parts (modules representing the SCNs) are well defined, they are packaged in separate compiled standalone executables that could be used on any computer even if it does not have MATLAB installed.

The modules M1, M2, ..., M64 are based on the same computing module but each one takes different input parameters. Thus they will be replaced in the parallel SCT by $exe1$. Same for M65, M66, and M67 that will be replaced by $exe2$. $exe3$ represent the 21 cascade executed in serial.

$$
\begin{aligned}
WorkerTasks &\begin{cases} exe1 \\ exe1 \\ exe1 \\ : \\ : \\ : \\ exe1 \\ exe2 \\ exe2 \\ exe2 \end{cases} \\
MasterTasks &\begin{cases} exe3 \\ exe3 \\ : \\ : \\ : \\ exe3 \end{cases}
\end{aligned}
$$

This SCT parallel application has been deployed with two different developed tools, MEG GUI and TUNe-DIET, both of them designed to make the use of the Grid transparent.

4.2.4 SCT deployment on Grid with MEG GUI

The purpose of this collection of scripts is to hide the complexity of using the Grid by a non expert in computer science and minimize entering command lines. It can be used for SCT and TLM simulations since the MEG environment is built with all the needed libraries and legacy files for both of them.

The user prepares its SCT experiment locally on its personal computer or laptop (that contains the folder of SCT scripts) which is connected to internet. To launch the procedure, user opens a (shell) terminal and execute a script called *start.sh*. Consequently, a friendly user interface opens and asks the user to enter its own personal Grid'5000 login informations.

After the authentication step, two kinds of scripts could be executed [2] to have a reservation "screen":

- Cluster scripts: dedicated to launch experiments on a specified cluster,

[2] execution based on expect tool (programmed dialogue with interactive programs) [138]

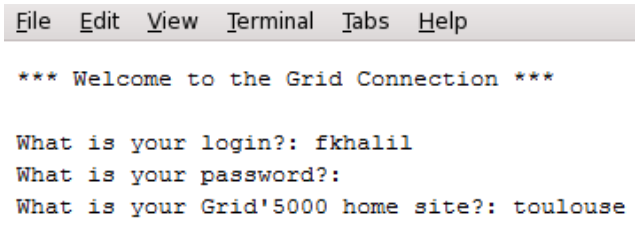


Figure 4.7: External access to Grid'5000 from local computer.

- Grid scripts: dedicated to launch experiments on the entire Grid with its sites and clusters.

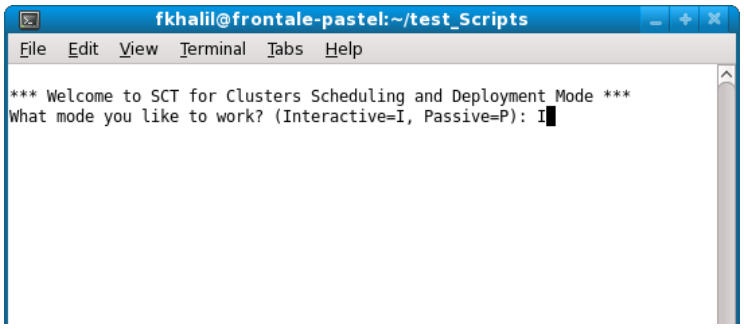


Figure 4.8: Interface for active or passive reservation on Clusters.

To reserve computing nodes, the user defines few parameters: Interactive or Passive mode (see OAR user manual), number of nodes on which the experiments will run, and an estimated needed time.

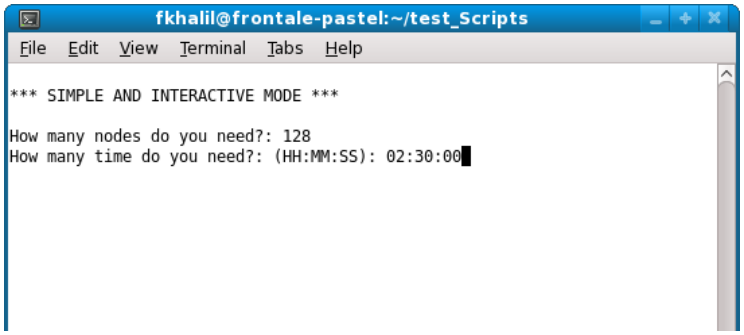


Figure 4.9: Define number of nodes and estimated needed time.

If passive mode has been chosen by user, a script called *ClusterDeployment.sh* will be executed to deploy the MEG environment on the reserved nodes. This step could take several minutes, since it installs a complete OS on reserved nodes.

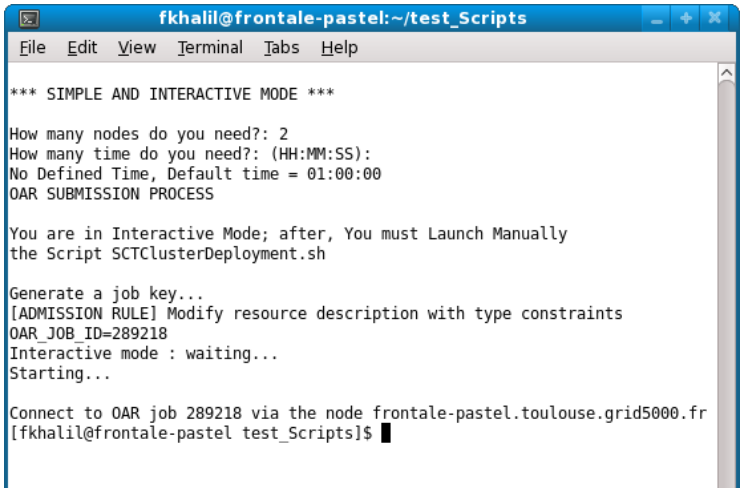


Figure 4.10: Confirmation of the reservation of nodes.

When interactive mode is chosen, the user must launch this procedure manually as seen in figures 4.10 and 4.11.

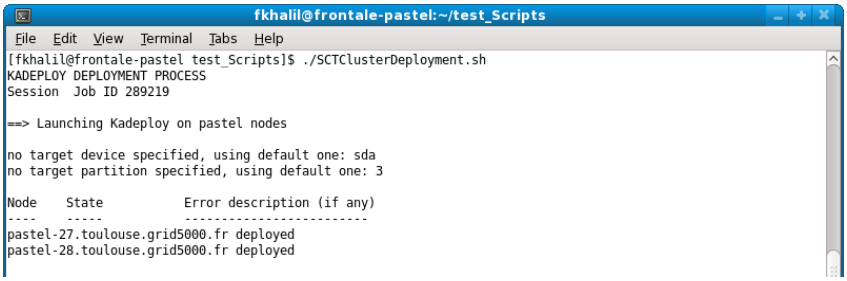


Figure 4.11: Deployment of MEG environment.

For reservation on several Grid'5000 sites, it is almost the same procedure (Figure 4.12).

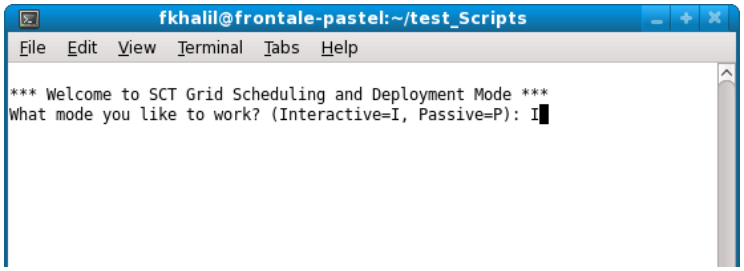


Figure 4.12: Choosing reservation mode on the Grid.

But in this case, user must choose on which cluster(s) the experiments will run (Figure 4.13).

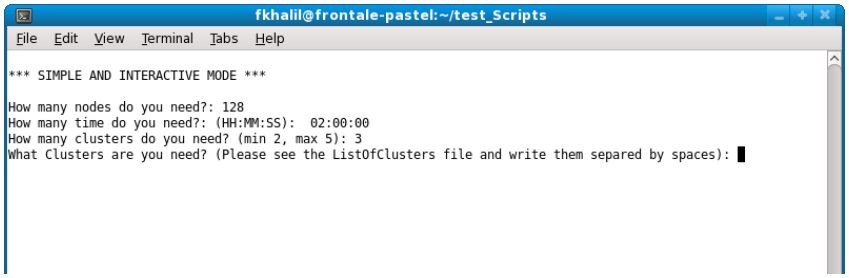


Figure 4.13: User input for Grid experiments.

The clusters names belonging to the Grid'5000 sites will be listed in order to choose (Figure 4.14).

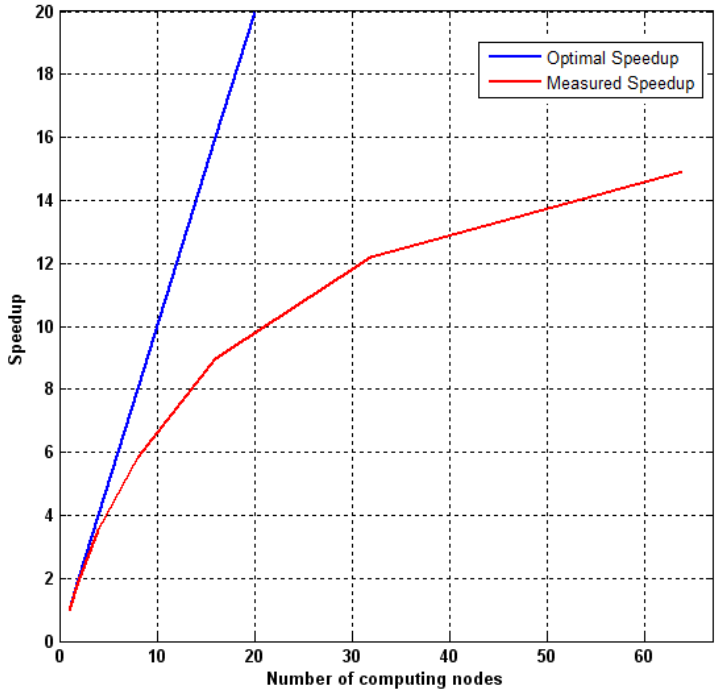
After deploying MEG on the nodes, the user will be redirected automatically by an ssh connection to the first reserved node, where all the needed files and scripts to run SCT simulation have been copied. The parallel execution of SCT is launched with Taktuk [115]. During this, the user has to do nothing else waiting the final results.

A Master process passes a description (input) of the task to each Worker process in order to solve it. Upon the completion of a task, the Worker passes the result (output) of the task back to the Master.

When executed, the Taktuk engine establishes a logical interconnection network between remote hosts. It is a tool for deploying parallel remote executions of commands to a potentially large set of remote nodes. It spreads itself using an adaptive algorithm and sets up an interconnection network to transport commands and perform I/Os multiplexing/demultiplexing.

Figure 4.15 shows the speedup calculated for the SCT application with respect to the number of nodes used in the analysis. It can be seen that is bounded because of the serial part of the cascade. The speedup could be enhanced in the future by parallelizing the cascades at each level and optimizing the scheduling and execution of the SCT application.

Figure 4.14: List of Grid'5000 clusters.



4.2.5 SCT deployment on Grid with TUNe-DIET

Among existing grid middleware approaches, one simple, powerful, and flexible approach consists of using servers available in different administrative domains through the classic client-server or Remote Procedure Call (RPC) paradigm [3]. Network Enabled Servers (NES) implement this model also called Grid-RPC [139]. Clients submit computation requests to a scheduler whose goal is to find a server available on the grid.

In order to simplify the distribution of the different SCT modules on Grid'5000 nodes, TUNe will be applied to DIET [4], an Application Service Provider (ASP) platform providing remote execution of computational problems on distributed, heterogeneous resources.

4.2.5.1 DIET

The Distributed Interactive Engineering Toolbox (DIET) project is focused on the development of scalable middleware by distributing the scheduling problem across multiple agents.

Several approaches exist for porting applications to grid platforms; examples include classic message-passing, batch processing, web portals, and Grid-RPC systems. This last approach implements a grid version of the classic Remote Procedure Call (RPC) model.

DIET consists of a set of elements that can be used together to build applications using the GridRPC paradigm [139]. This middleware is able to find an appropriate server according to the information given in the client's request (problem to be solved, size of the data involved), the performance of the target platform (server load, available memory, communication performance) and the local availability of data stored during previous computations.

The scheduler is distributed using several collaborating hierarchies connected either statically or dynamically (in a peer-to-peer fashion). Data management is provided to allow persistent data to stay within the system for future re-use. This feature avoids unnecessary communication when dependences exist between different requests. The DIET architecture is based on a hierarchical approach to provide scalability. The architecture is flexible and can be adapted to diverse environments including heterogeneous network hierarchies.

DIET is implemented in Corba [140] and thus benefits from the many standardized, stable services provided by freely-available and high performance Corba implementations.

DIET is based on several components. A Client is an application that uses

[3] Remote Procedure Call (RPC) is a technique for constrcuting distributed, client-server applications.

[4] DIET. http://graal.ens-lyon.fr/ diet/

DIET to solve problems using an RPC approach. Users can access DIET via different kinds of client interfaces: web portals, PSEs such as Scilab [5], or from programs written in C or C++.

A **SeD**, or server daemon, provides the interface to computational servers and can offer any number of application specific computational services. A **SeD** can serve as the interface and execution mechanism for a stand-alone interactive machine, or it can serve as the interface to a parallel supercomputer by providing submission services to a batch scheduler.

Agents provide higher-level services such as scheduling and data management. These services are made scalable by distributing them across a hierarchy of agents composed of a single Master Agent (**MA**), several Agents (**A**), and Local Agents (**LA**). Figure 4.16 shows an example of a DIET hierarchy treating a SCT simulation.

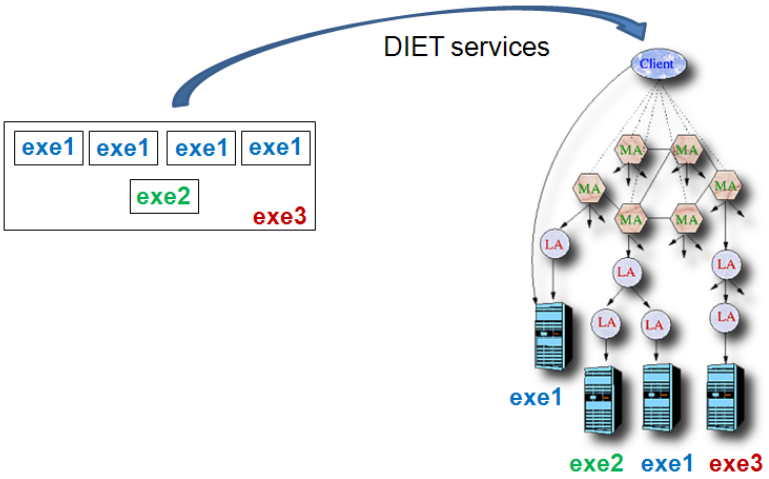


Figure 4.16: Executables distribution architecture to DIET hierarchical organization.

A Master Agent is the entry point of the environment. In order to access DIET scheduling services, clients only need a string-based name for the **MA** (e.g. "MA1") they wish to access; this **MA** name is matched with a Corba identifier object via a standard Corba naming service.

Clients submit requests for a specific computational service to the **MA**. The **MA** then forwards the request in the DIET hierarchy and the child agents, if any exist, forward the request onwards until the request reaches the **SeD**s. The **SeD**s then evaluate their own capacity to perform the re-

[5]Scilab Home page. http://www.scilab.org/

quested service; capacity can be measured in a variety of ways including an application-specific performance prediction, general server load, or local availability of data-sets specifically needed by the application. The **SeD**s forward their responses back up the agent hierarchy. The agents perform a distributed collation and reduction of server responses until finally the **MA** returns to the client a list of possible server choices sorted using an objective function (computation cost, communication cost, machine load, ...). The client program may then submit the request directly to any of the proposed servers, though typically the first server will be preferred as it is predicted to be the most appropriate server.

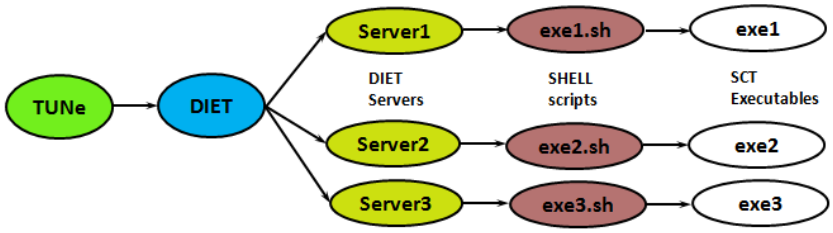


Figure 4.17: DIET deployment with TUNe.

The primary interest of the DIET scheduling approach lies in its distribution, both in terms of collaborative decision making and in terms of distribution of information important to the scheduling decision. When the **MA** receives a client request, it (1) verifies that the service requested exists in the hierarchy, (2) collects a list of its children that are thought to offer the service, and (3) forwards the request on those subtrees. Local agents use the same approach for forwarding the request to their children, whether the children are other agents or **SeD**s. Agents obtain information on services available in subtrees during the deployment process. When a **Se**D or agent starts up, it joins the DIET hierarchy by contacting its parent agent (located by a string-based name in a naming service). The parent adds the new child to its list of children and records which services are available via that child. The parent need not track whether the service is provided directly by the child (if the child is a server) or by another server in the child's subtree (if the child is an agent); it suffices to know the service is available via the child. Thus if an agent has **N** children and the DIET hierarchy offers a total of **M** services, the most hierarchy information any agent in the tree will store is **N*M** service/child mappings.

When an agent forwards a request to its children, it sets a timer restricting the amount of time to wait for child responses. This avoids a deadlock in the hierarchy based on one failed or slow-to-respond server. Eventually, a child will be forgotten if it is unresponsive for long enough.

SeDs are responsible for collecting and storing all of their own performance

and status data. Specifically, the **SeD** stores a list of problems that can be solved on it, a list of any persistent data that are available locally to the server, and status information such as the number of requests currently running on the **SeD** and the amount of time elapsed since the last request. When a request arrives at a **SeD**, the **SeD** creates a response object containing both status information and performance data.

4.2.5.2 Simulation workflow

TUNe is responsible of deploying DIET on the Grid. This step is done after the expression of global dynamic behavior of the system with UML state charts or activity diagrams on local machine [6]. Once DIET is deployed on the Grid, the $diet - sct.uml$ is used.

The DIET client (written in C++) calls Server1 sixty four (64) times in order to execute $exe1$ and Server2 three (3) times to execute $exe2$. Usually the server launches the executables directly, but in this case, a shell script makes the relation between the server and the executable (Figure 4.17). This script is used in order to initialize environment variables that are necessary for the application, and then launch the simualtion.

It can be noticed in Figure 4.3 that the cascade at each level can also run in parallel. For the moment this feature is not implemented since it needs a high level of synchronism. Consider *N1* executions of $exe1$ and *N2* executions of $exe2$, $exe3$ will have to wait the *N1+N2* output files in order to be generated. The synchronism needs a special activity diagram for TUNe.

4.3 Distributed Parametric SCT Simulations in Grid Environment

As mentioned at the beginning of this chapter, a convergence study should fix the appropriate number of modes (active and passive) expressing each domain.

Since this study generates lot of computing cases, it is ideal to run in distributed environments where a high of computing nodes are available. In the case of the reflectarray (of 256 non-uniform cells), a remarkable time reduction is reached while distributing the independent simulations on Grid'5000 nodes. In fact, the ideal case is to run one simulation per node. Thus the global time needed in order to have final results is equal to the longest simulation time. Consider M simulations with the same execution time t, if distributed on N computing nodes, the global time T is expressed by $T = M * t / N$. when $M = N$, the global time T will be equal to t [7].

[6] See Chapter 3 for more details.

[7] Tasks are independent and no network losses

The results helps the user to fix the number of modes. Figure 4.18 shows an example of the different curves calculated in the distributed simulations. When increasing the number of modes, the different curves will converge to some point where the difference is too small between them (Δ «). In Figure 4.18, clearly not all the curves/colors can be identified. At this moment, the correspondant number of modes is chosen.

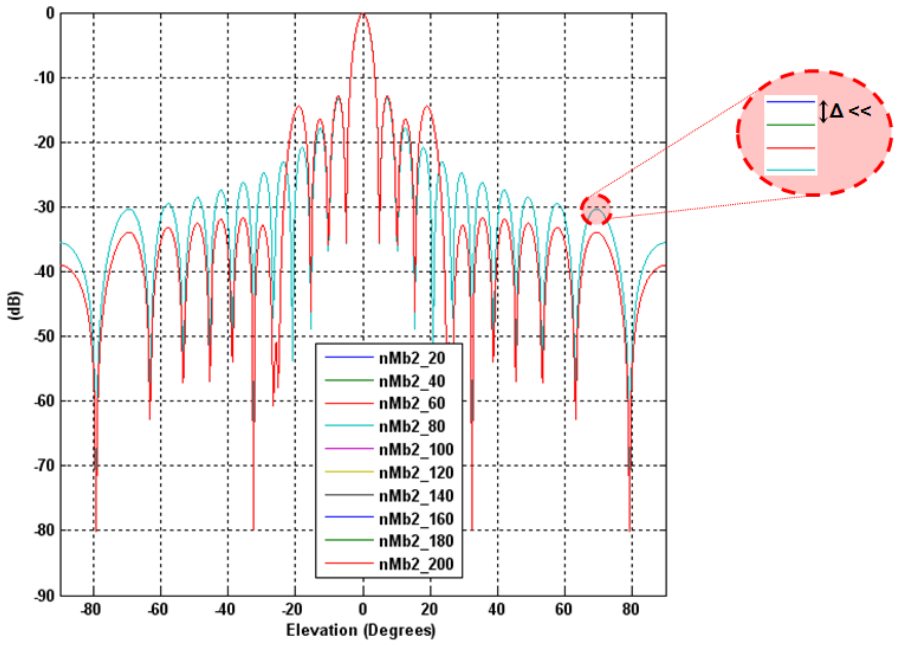


Figure 4.18: Example of convergence study where nMb2 is the number of modes.

While using this technique to fix the number of modes, SCT resuls are found to be in good agreement with simulation results of electromagnetic software HFSS [116] and IE3D [141] (Figure 4.19).

In addition to the utility shown in convergence studies, the distributed parametric approach applied in chapter 3 on design dimensions and frequency sweep could be applied also for SCT applications.

4.4 Conclusion

The Scale Changing Technique modeling method was deployed on Grid'5000 nodes. Two different approaches of distributed computing have been tested in the case of electromagnetic simulation of a planar array antenna. The SCT shows a good intrinsic scalabilty. The fact of partitioning the electromagnetic simulation based on multiscale modeling generates independent jobs that can run in parallel in distributed environments. An acceptable speedup

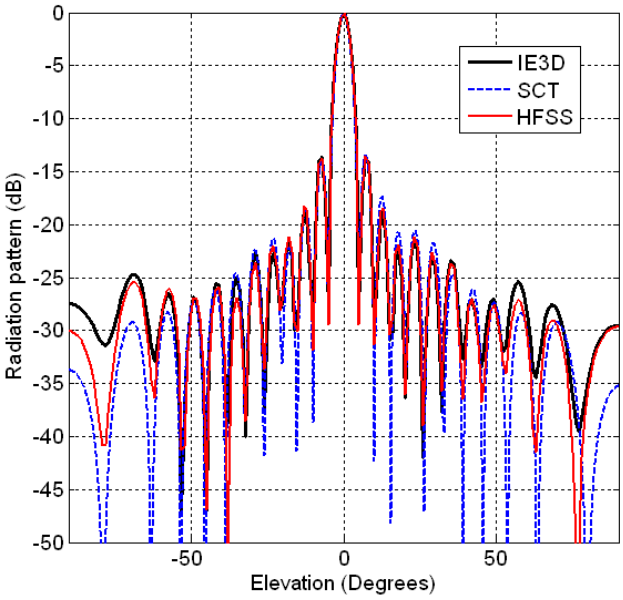


Figure 4.19: The radiation pattern in E-Plane of the array simulated by SCT (considering convergence study) in comparison to HFSS and IE3D results.

has been calculated while keeping on the accuracy of results. The performance is bounded due to the serial part of the algorithm consisting of the cascade phase. The speedup could be enhanced in the future by parallelizing the cascade operations at each level.

To provide transparent access to a pool of computational servers at a very large scale and run the SCT computing modules in parallel, TUNe was used here to submit computation requests to DIET scheduler whose goal is to find a server available on the grid.

The parallel case with TUNe and DIET presented in this chapter is a preliminary study of the distribution of SCT jobs. This approach has proved to be simple, powerful, and flexible. While deploying DIET with TUNe, a small overhead is added to the parallel application (explained in previous chapter). Eventhough, this approach seems so atractive. A graphical interface of TUNe is under development and will be used in the near future to deploy DIET on the Grid nodes more transparently.

On the other hand, the parametric distributed computing is very useful for SCT convergence studies and for design parametric sweeps (i.e. frequency).

Chapter 5

Conclusions and Perspective

In computational electromagnetics, the ever increasing need for more precision and larger meshes raises the natural question whether it is worth porting algorithms to computer grids and the way to use such infrastructures.

The scope of the present work is to define an environment and tools for the electromagnetic simulation of future radio frequency systems. This objective is pursued by formulating an adaptive use of the distributed architecture. Friendly-user solutions as graphical user interfaces (GUIs)have been used in order to keep the use of Grid Computing transparent to electronic engineers that are not probably computer science specialists. Distributed simulations of modern electromagnetic structures (i.e. large planar reflectarrays and phase shifters based on MEMS switches) have been presented. The efficiency of the different proposed approaches has been evaluated.

In fact, Grid computing is not a "silver bullet". It offers a great opportunity for scientists to access a large amount of shared resources, but at the same time distributed software on a large scale environment are increasingly complex and difficult to manage. Besides, the (hardware and software) heterogeneity of the Grid is sometimes a disadvantage. For example, the user must change the used compilers and libraries when using AMD architecture or INTEL architecture, or re-write parts of a programm code or script beacause of the upgrade of an installed software.

The Grid offers availability of a high number of computing nodes but resource managers must tailor their behavior dynamically and use the available resources and services efficiently and effectively.

To address these issues, a promising approach based on autonomic computing has been used. TUNe middleware has simplified the use of electromagnetic solvers. It has also increased the robustness of the application execution by automatically restarting parts of the simulation that failed and redeploying them on new nodes.

Due to the nature of computations, TLM algorithm is not trivially parallelizable, as data dependency inside the meshes implies communication. Since adopted communication strategy depend on network, in the case of TLM computing, it would be better to send a few large messages. Applications that require a large number of geographically located resources must be designed to be latency and bandwidth tolerant.

The TLM method deployed on the Grid'5000 nodes has shown good speedup and scalability. The memory problem has been solved by using a high number of computing nodes having normal hardware characteristics, i.e. 2GB of RAM.

An hybrid parallel programming model could be the most adapted for TLM in distributed environments, i.e. a threaded programming on multiprocessor nodes (based on OpenMP [142]) for each sub-region independently of other sub-regions with message passing on the boundaries (common to sub-regions) between Grid nodes (based on MPI [107]).

Meanwhile, the distributed computing, parallel and parametric applications, of Scale Changing Technique in Grid environments seems to be very promising. Accurate and fast electromagnetic simulation results have been reported. The good speedup shown here could be enhanced by parallelizing the cascade operations at each level.

Based on the SCT, the modeling of more complex structures are being developed actually requiring more computational resources. Using the proposed approaches with these future applications is so easy, since the tools developed in this thesis have been adapted to the methodology of the SCT method and they do not only treat specific problems.

To provide transparent access to a pool of computational servers at a very large scale and run the SCT computing modules in parallel, TUNe was used here to submit computation requests to DIET scheduler whose goal is to find a server available on the grid. The Diet component architecture is structured hierarchically for improved scalability. Such an architecture is fexible and can be adapted to diverse environments, including arbitrary heterogeneous computing platforms.

The DIET middleware represents an interesting example for the TUNe autonomic management environment, since it brings together many of the challenges addressed by TUNe:

- the management of a distributed organization of legacy software (MA, LA and SeD).
- distributed configuration, since DIET requires the configuration of these software (through a set of configuration files) to implement a consistent hierarchical structure.
- self-repair: due to several reasons (hardware or software), a server may stop functioning. The goal is to detect the failure and repair it automatically.

Moreover, using the TUNe GUI to deploy DIET (Autonomic administration of DIET with TUNe) and distribute the SCT jobs on Grid nodes renders

the tasks of user so easy.

In addition to these solutions, it could be very useful to combine the SCT method with the TLM method in hybrid simulations in the case of electromagnetic strutures combing at the same time planar components (ideal to simulate with SCT i.e. small front-end antennas) with complex circuits (to be used with fullwave 3D TLM method, i.e. circuits containing MEMS switches and CMOS integrated circuits). This feature could be realized with emGine environment since it support imported electromagnetic data and will support soon genetic optimizations.

Implementing distributed genetic algorithms for CEM applications is a very active research area, especially in Computer-Aided Engineering (CAE) for antenna design and simulations. In this context, the case of distributed Genetic Search Optimisation (Figure 5.1) with emGine environment is being actually developped. The theory of genetic algorithms is based upon evaluations of the quality (fitness) of multiple potential solutions and mixing of input parameters for the production of new solutions, following the laws of natural evolution (survival of the fittest).

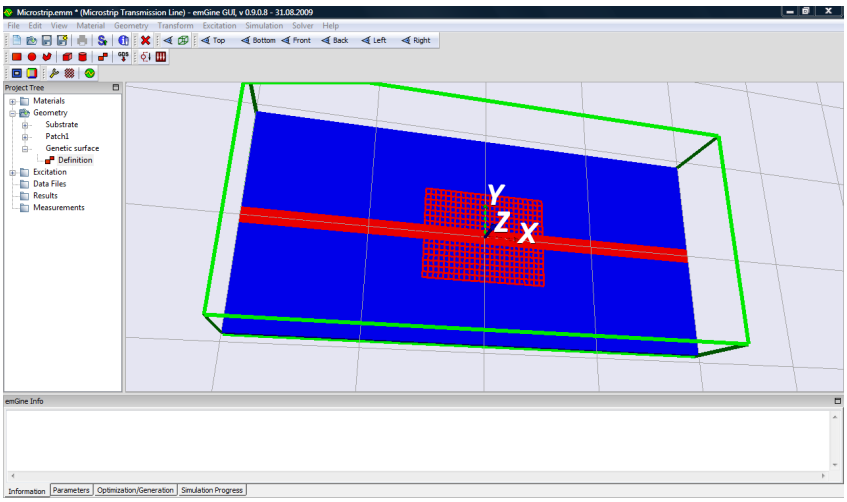


Figure 5.1: Genetic surfaces in emGine environment.

A genetic algorithm performs a guided, intelligent random search in the multi-dimensional space of optimization parameters, for the global optimum. The genetic algorithm finds tentative solutions (called chromosomes or individuals) and classifies their quality according to a fitness function, defined according to the desired characteristics. The optimization parameter values are produced by combining the simulation of the survival-of-the-fittest natural mechanisms (crossover, mating, mutation, population decimation) with random processes. It must be noted that the estimation of the fitness function

is usually a complex and resource-demanding task, since it typically involves large-scale electromagnetic field calculations.

The distributed genetic algorithms practically provides scalability to the classic genetic algorithms by dividing the costly fitness evaluations into several interconnected processing nodes.

On another hand, exciting opportunities arise from new hardware architectures including graphics processing units (GPUs). This kind of computing is known as GPU Computing or GPGPU [1]. Graphics Processing Units (GPUs) are high-performance many-core processors that can be used to accelerate a wide range of applications. Such systems, originally designed for gaming and graphics processing, comprise hundreds and even thousands stream processors in a single envelope at a very low cost. GPU systems offer the power of a CPU based cluster but at a cost of a simple desktop computer. Moreover, the recent introduction of the CUDA extension of the C programming language [143] allowed writing general purpose high-level codes without considering specifics of a particular graphics processor.

Since the fundamental idea behind using GPUs for TLM and SCT electromagnetic numerical techniques is parallelization. The parallelization of the program is done by launching execution threads simultaneously. A batch of fixed number of threads executed on a multiprocessor are labeled as a thread block. Threads within the same block can be synchronized at run time and can cooperate with each other via GPU's shared memory, which can be as fast as registers. Each thread is identified via its thread ID and thus can access different parts of the memory. However, certain requirements have to be met to achieve high efficiency due to the hardware architecture of GPU. Understanding how to map algorithms appropriately to the GPU is of vital importance to achieving the maximum gains in speed.

Finally, it would be interesting in the future to test bigger deployments on the Grid with more scalable tools, i.e. Taktuk [115], instead of scp (secure copy), and to detect bottleneck simulations, interrupt them and migrate them to faster computing nodes in order to enhance the actual performance.

[1] GPGPU stands for General-Purpose computation on Graphics Processing Units. http://gpgpu.org/about is a central resource for GPGPU news and information.

Appendix A

YATPAC

A.1 The Ultimate Open Source TLM Simulation Package

YATPAC (Yet Another TLM Package) [1] is a free (open source project licensed under the GNU General Public License) TLM-based full-wave electromagnetic simulation package developed at the Institute for High-Frequency Engineering of the Technische Universitat Munchen in Germany. The equation solver of the electrostatic field solver was developed by the Weierstrass Institute for applied analysis and stochastics in Berlin and is licensed to use it for free in YATSIM.

With YATPAC various electromagnetic structures can be characterized in time-domain like hollow waveguides, transmission lines, planar microwave circuits and antennas. It is best suited for layer oriented MMIC structures and people, who want to know, what the software is doing in the background.

Perhaps YATPAC may not be the best for people who want to click every action, but compared with commercial software tools one of the best in time domain with excellent results.

A.2 Overview of the YATSIM Simulation Package

The YATSIM package combines various UNIX/Linux based programs developed and used for solving three dimensional field problems using the TLM method.

The most important YATPAC's components are:

- ***yatpre*** - the preprocessor
- ***yatsim*** (Yet Another TLM Simulator) - the core computational engine
- ***yati2of*** - the visualization component of the TLM model file
- ***yatvis5d*** - the visualization component of the computed electromagnetic field
- ***yatspar*** - the calculation of S-Parameters

[1] YATPAC Homepage. http://www.yatpac.org/

A simulation is performed by three steps: preprocessing, simulation, and postprocessing. The next sections should give a short overview about the different programs, which are used for performing these three steps.

A.2.1 Preprocessing

At this step, the design of the structure is prepared for the simulator kernel. The preprocessing with ***yatpre*** consists of two steps. First, the preparation of the layout file, containing the information of the engineered HF structure, in GDSII file format, using **KIC** (free layout editor distributed by Whiteley Research Inc.), **ADS** (Advanced Design System commercialised by Agilent) or **CleWin** (commercialised by PhoeniX) or any other software supporting the GDSII file format. This file must have the extension "**.gds**".

Second, the preparation of the laydef file (an ASCII file with a C++ like syntax containing informations on the discretization of the simulation object), using any editor, like Kwrite, Vi or Emacs. This file must have the extension "**.laydef**".

Than, an I3D model file, named **I3D.**outfname must be generated from the two input files, by calling yatpre with his specific command (***yatpre*** -i *fname*.**laydef** *fname*.**gds I3D.***outfname*). This ASCII output file contains all needed input datas for the simulator ***yatsim*** in a lower language format.

A.2.2 Simulation

yatsim represents the simulator itself. As input, the I3D-file is used. As output three files are generated. The first, **VIXY01.**<*name*>**.v5d** contains all 6 field components of each cell in the simulation area at every time step.

The second file, **EH.**<*name*>, contains certain current and voltage-signals in time domain, computed by integral paths over E- and H-field in defined planes in the laydef file. In an additional <*name*>**.logfile**, performance information and other datas concerning the simulation run are saved.

The third file, **NF_**<*structure*>**.m** contains the computed nearfields of the simulated structure.

The simulation of an I3D model file **I3D.***outfname* with yatsim can be started using the command ***yatsim*** *outfname*.

A.2.3 Postprocessing

The visualization of the discretized structure is done with **Geomview** (written at the Geometry Center at the University of Minnesota).

The computed time-domain signals in EH-file (ASCII code) or calculated S-Parameters can be regarded and processed by tools like **Grace** freeware graph plotting program (developed by Grace Development Team), which can

be called by a simple command: ***xmgrace*** -nxy **EH.***outfname* , or math computing programs like **Octave** or **MATLAB**.

The visualization of the computed electromagnetic field is done with the ***yatvis5d*** component. It is derived from the great **vis5d** visualization software (originally developed by the University of Wisconsin-Madison for meteorological applications). As input, the v5d-file is used (command: ***yatvis5d*** **VIXY01.***outfname***.v5d**). Isoclines, vector plots, intensity plots and field lines can be plotted time continuously. Features to create movies for presentations are possible too.

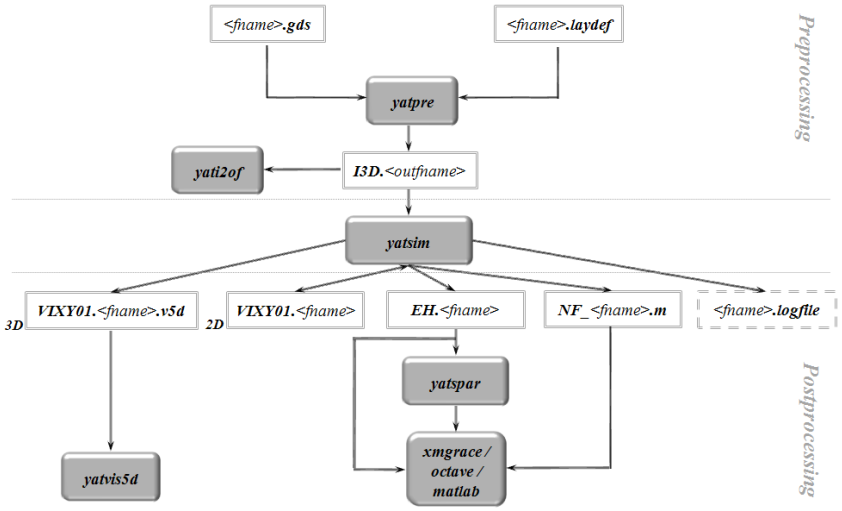


Figure A.1: Overview of all programs and files in the simulation flow of Yatsim.

A.2.3.1 S-Parameters

The S-Parameters in frequency-domain can be obtained by using the program ***yatspar***. They are computed from the current and voltage time domain signals in the EH-files generated by the simulator kernel. During the program run, different inputs have to be stated via a terminal user interface. They refer to settings, placed in the laydef-file. The output is an ASCII file in touchstone sNp format (N is the port number of the device), which contains the S-Parameters in magnitude and phase. The sNp format can be easily imported to the **ADS** data display module or grace for printing and presenting results.

A.2.3.2 Far Fields

The far fields at a defined frequency can be obtained by a near field-to-far field transformation (NFFFT) from near field values (saved in the file

NF_<*structure*>**.m**) using **MATLAB** or **octave**, and different components of the field (Ephi, Etheta ...) could be plotted for a desired angle.

Figure A.1 shows a flow chart of the simulation process and the different connections between files and programs.

Appendix B

emGine Environment

The emGine Environment [1] is a time-domain full-wave 3-D electromagnetic simulator based on the transmission line matrix (TLM) method. It is used for the modeling of high-frequency electromagnetic field in microwave circuits, antennas, resonators, hollow waveguides, etc.

The emGine Environment is provided free for non-commercial purposes (see the licenses for more details) and consists of the following components:

- ***emGine GUI*** : open source Graphical User Interface (written in Python) for the input, pre-processing and post-processing of electromagnetic models
- ***tlmGine***: time-domain full-wave 3D electromagnetic simulator (written in C++)

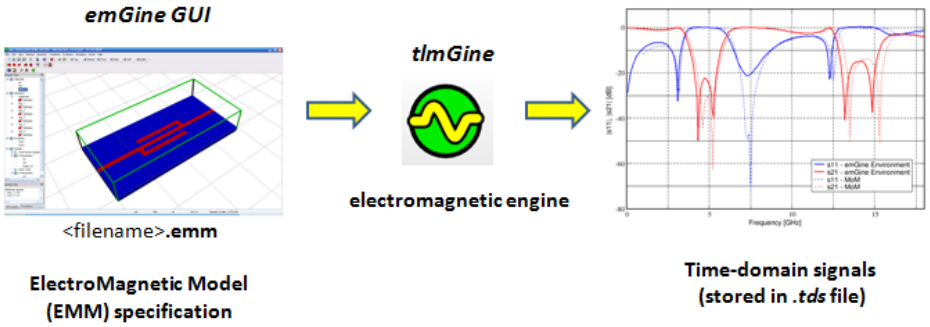


Figure B.1: emGine Environment simulation flow.

The development of the emGine Environment is focused to have following features:

- Accuracy, Speed, and support for multi-core computations
- Portability - the emGine Environment is running on multiple platforms (Windows, Linux, MacOS X)
- Open standards and interoperability, e.g., Open Source GUI, XML-based EMM specification
- User-friendly handling

[1] http://www.petr-lorenz.com/emgine/

Appendix C

Beam steering of planar arrays

The reflectarray concept is based on the scattering properties of microstrip patches. Each array element printed onto the reflecting surface is designed to reradiate the incident field with a phase delay suitable to steer the main reflectarray beam in a specified direction [144].

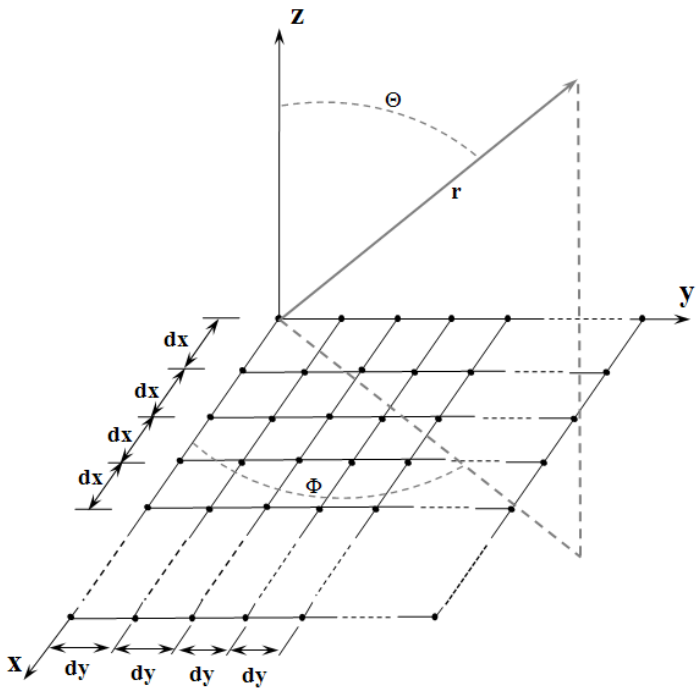


Figure C.1: Rectangular planar array geometry.

To design a microstrip reflectarray, the scattering properties of each patch have to be accurately estimated so that the desired phase distribution over the whole array surface can be achieved.

Here, the relationship between the reflection phase and the patch configuration has been obtained through extensive simulations. To estimate the phase of the reflected field versus patch and slot length, a variable-sized microstrip antenna element has been analyzed.

A set of isolated microstrip patches (loaded by slots) printed on a 16.8 mm x 16.8 mm grounded dielectric slab of 3.175 mm height and with $\varepsilon r = 2.17$

has been simulated at 12.5 GHz (Figure C.2), illuminated by a normally and linearly polarized incident plane wave.

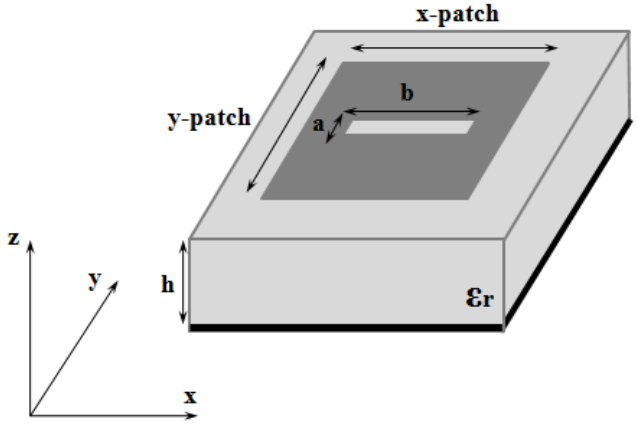


Figure C.2: Individual array cell.

In Figure C.3, the simulated curves that relate the phase of the field scattered by the isolated patch and the antenna size are presented.

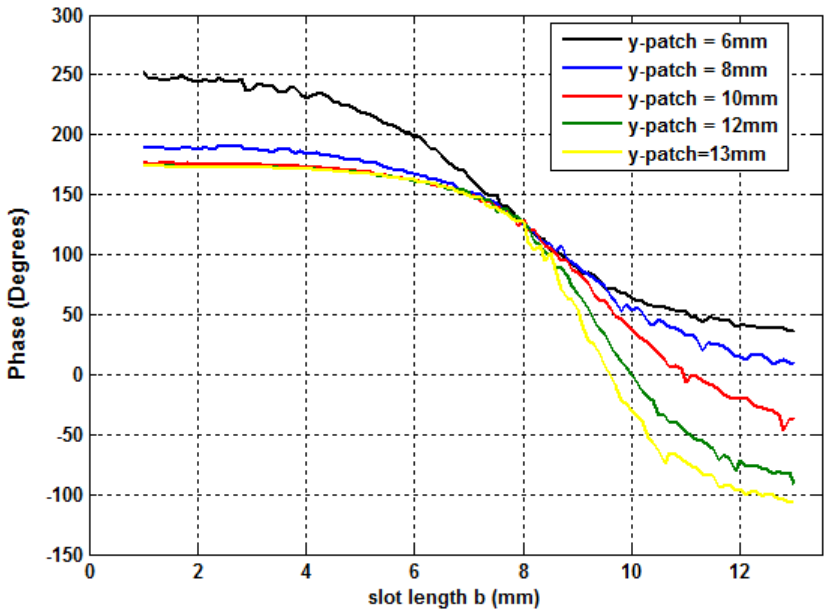


Figure C.3: Phase of the field reradiated by reflectarray samples versus slot length for different patches size (x-patch = 13.5 mm and a = 1 mm).

If it is desired to have only one main beam that is directed along $\Theta = \Theta_0$ and $\phi = \phi_0$, the progressive phase shift between the elements in the x- and y-directions must be equal to

β_x = -k.d_x.sinΘ_0.cosϕ_0 ; β_y = -k.d_y.sinΘ_0.sinϕ_0

where k is wave number of free space (k = $2\pi/\lambda$).

By controlling the progressive phase difference between the elements, β_x and β_y, the maximum radiation can be squinted in any desired direction to form a scanning array.

In the case of the array used in chapter 3 and 4, the inter-elements space is: dx = dy = d = 0.7λ. To change the direction of the main lobe of the radiation pattern 5° in the $\phi = 0°$ cut, β_x is equal to $-21.96°$ while β_y is equal to zero. After checking the desired phase of each element of the array represented in Figure C.1, the dimensions could be defined from the curves of Figure C.3.

Appendix D

Acronyms

3D Three Dimensional
API Application Programming Interface
ASP Application Service Provider
CAD Computer-Aided Design
CAM Computer-Aided Manufacturing
CEM Computational ElectroMagnetics
CFD Computational Fluid Dynamics
CPU Central Processing Unit
DIET Distributed Interactive Engineering Toolbox project
DNS Domain Name System
EGEE Enabling Grids for E-sciencE
EM ElectroMagnetic
EMM ElectroMagnetic Model
FD Frequency Domain
FDTD Finite-Difference Time-Domain
FSS Frequency Selective Surface
GC Grid Computing
GNU GPL General Public License
GPU Graphical Processing Unit
GPGPU General-Purpose computation on Graphics Processing Units
GUI Graphical User Interface
IP Internet Protocol
IDE Integrated Development Environment
LA Local Agent
LAN Local Area Network
LDAP Lightweight Directory Access Protocol
MA Master Agent
MEMS Micro-Electro-Mechanical Systems
MPI Message Passing Interface
MPP Massive Parallel Processing
NES Network Enabled Servers
OGSA Open Grid Services Architecture
OS Operating System
P2P Peer-to-Peer
PDE Partial Differential Equation

PEC Perfect Electric Condition
PID Process Identifier
PPE Parallel Production Environments
PVM Parallel Virtual Machine
QoS Quality of Service
RENATER Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche
RF Radio Frequency
RISC Reduced Instruction Set Computer
RPC Remote Procedure Call paradigm
RMI Remote Method Invocation
SATA Serial Advanced Technology Attachment
SCP Secure Copy Protocol
SCN Scale Changing Network
SCT Scale Changing Technique
SeD Server Daemon
SIMD Single Instruction Multiple Data
SIMID SIMulation ID
TCP Transmission Control Protocol
TD Time Domain
TLM Transmission Line Matrix
TUNe Toulouse University Network
UML Unified Modeling Language
XML eXtensible Markup Language
YATPAC Yet Another Tlm simulation PACkage

Appendix E

Author Biography

Fadi Khalil was born in Beirut, Lebanon, on March 1983. He received his BS degree in Electrical Engineering from the Lebanese University, Lebanon in 2005, and the MS degree in Modelling, Information and Systems, in 2006 from Université Paul Sabatier, Toulouse, France.

From November 2006 to November 2009, Fadi was working toward the PhD degree in Micro and Nanosystem for wireless Communications Research Group at the Laboratoire d'Analyse et d'Architectures des Systèmes at the Centre National de la Recherche Scientifique (LAAS-CNRS) in Toulouse, France.

His research interests include Computational Electromagnetics, modelling of complex (multi-scale) structures and reconfigurable circuits for RF and microwave applications, High Performance Computing and Grid Computing.

During the thesis, author was affiliated to IEEE MTT (Microwave Theory and Techniques) Society and the IEEE AP (Antennas and Propagation), and was an ACES (Applied Computational Electromagnetics Society) fellow. He is a reviewer of CLCAR 2009 conference and CLCAR 2010 conference, and a regular reviewer for the International Journal of Computer Science Issues (IJSCI).

To send an e-mail: **fadi.khalil@yahoo.fr**
Web page: **http://fadi.khalil.perso.sfr.fr**

Appendix F

List of Publications

Journals

- F. Khalil, H. Aubert, F. Coccetti, P. Lorenz, and R. Plana, "Grid- Based Global Electromagnetic Simulation Tool for Parametric Distributed Analysis of Array Antennas," *Progress In Electromagnetics Research M*, Vol. 10, 1-12, 2009.

- F. Khalil, C. J. Barrios-Hernandez, A. Rashid, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana, "Parallelization of the Scale Changing Technique in Grid Computing environment for the Electromagnetic Simulation of Multiscale Structures," *International Journal of Numrical Modeling: Electronic Networks, Devices And Fields*, John Wiley & Sons, 2010.

- F. Khalil, "Young Professional and Graduate in Actual Global Financial Crisis," *IEEE GOLDRush 2009 Newsletter* (Under Review).

International Conferences

- F. Khalil, H. Aubert, F. Coccetti, R. Plana, Y. Deunneulin, B. Miegemolle, T. Monteil, and H. Legay, "Electromagnetic Simulation of MEMS-Controlled Reflectarrays based on SCT in Grid Environment," *IEEE International Symposium on Antennas and Propagation*, Honolulu, Hawaii, USA, pp. 49-52, June 10-15, 2007.

- F. Khalil, H. Aubert, F. Coccetti, R. Plana, T. Vaha-Heikkila, and T. Lisec, "Distributed Parametric Simulation of Complex Electromagnetic Structures in Grid Computing Environment," *8th International Symposium on RF MEMS and RF Microsystems MEMSWAVE*, Barcelona, Spain, pp. 215-218, June 26-29, 2007.

- F. Khalil, C. J. Barrios-Hernandez, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana, "Multiscale modeling: from Electromagnetism to the GRID," *European Project Showcase, IEEE CCGrid 2008*, Lyon, France, May 19-22, 2008.

- F. Khalil, B. Miegemolle, T. Monteil, H. Aubert, F. Coccetti, and R. Plana, "Simulation of Micro Electro-Mechanical Systems (MEMS) on Grid," *8th International Meeting High Performance Computing for Computational Science (VECPAR'08)*, Toulouse, France, pp. 614-627, June 24-27, 2008.

- F. Khalil, C. J. Barrios-Hernandez, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana, "Electromagnetic Simulations via Parallel Computing: an Application Using Scale Changing Technique for Modeling of Passive Planar Reflectarrays in Grid Environment," *2008 IEEE International Symposium on Antennas and Propagation and the 2008 USNC/URSI National Radio Science meeting*, San Diego, California, July 5-12, 2008.

- R. Sharrock, F. Khalil, T. Monteil, H. Aubert, F. Coccetti, P. Stolf, L. Broto, and R. Plana, "Deployment and management of large planar reflectarray antennas simulation on grid," *International Symposium on High Performance Distributed Computing, CLADE 2009*, Munich, Germany, June 11-13, 2009.

- F. Khalil, A. Rashid, H. Aubert, F. Coccetti, R. Plana, C.-J. Barrios-Hernandez, and Y. Denneulin, "Application of scale changing technique-grid computing to the electromagnetic simulation of reflectarrays," *2009 IEEE International Symposium on Antennas and Propagation and the 2009 USNC/URSI National Radio Science meeting*, Charleston, South Carolina, June 1-7, 2009.

- C. J. Barrios-Hernandez, F. Khalil, Y. Denneulin, H. Aubert, F. Coccetti, and R. Plana, "Deployment of CEM Applications On Large Scale Architectures," *Latin American Conference on High Performance Computing CLCAR 2009*, Merida state, Venezuela, September 21-25, 2009.

- E. B. Tchikaya, A. Rashid, F. Khalil, H. Aubert, H. Legay, and N. J.G. Fonseca, "Multi-scale Approach for the Electromagnetic Modeling of Metallic FSS Grids of Finite Thickness with Non-uniform Cells," *2009 Asia-Pacific Microwave Conference (APMC 2009)*, Singapore, December 7-10, 2009.

- F. Khalil, R. Sharrock, H. Aubert, F. Coccetti, R. Plana, T. Monteil and Y. Denneulin, "Distributed Electromagnetic Analysis of Reflectarrays," *The 2010 annual conference of the Applied Computational Electromagnetics Society (ACES)*, Tampere, Finland, April 26-29, 2010.

National Conferences

- F. Khalil, F. Coccetti, H. Aubert, R. Plana, Y. Denneulin, B. Miegemolle, and T. Monteil, "Etude des potentialités du concept de Grille de Calcul pour la Simulation Electromagnetique de Micro-Systèmes Complexes," *15èmes Journées Nationales Micro-ondes*, Toulouse, France, pp. 92, May 23-24-25, 2007.

- F. Khalil, C. J. Barrios-Hernandez, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana, " Simulations électromagnetiques distribuées sur une grille de calcul," *16èmes Journées Nationales Micro-ondes*, Grenoble, France, May 25-28, 2009.

- F. Khalil, "Modélisation électromagnetique des Antennes à Réseaux Réflecteurs," *Smart Engineering Simulation ANSYS 2009*, Paris, France, October 14-15, 2009.

Miscellaneous

- F. Khalil, "Coupling bi- and tri-dimensional electromagnetic modeling methods with computing grid," *LAAS Report* (06263), September 2006, 56 pages.

- F. Khalil, "Transmission line matrix modeling method," *LAAS Report* (07820), August 2007, 40 pages.

- F. Khalil, "Modélisation multi-échelles: de l'électromagnétisme à la grille," *Journée doctorale GEET*, Mars 2009.

- P. Lorenz and F. Khalil, User Manual of Electromagnetic Modeling Software emGine Environment v0.8.0.

- F. Khalil, H. Aubert, "La puissance d'une infrastructure virtuelle pour la résolution de problèmes électromagnétique dans le monde réel," Cahier N°3 de l'ANR, Calcul Haute Performance: une technologie clé pour des multiples applications.

Bibliography

[1] R. E. Collins. *Foundations of Microwave Engineering.* New York: McGraw-Hill, 1966.

[2] J. A. Kong. *Electromagnetic Wave Theory.* New York: John Wiley and Sons, 1986.

[3] F. El Dabaghi. *Approximations and Numerical Methods for the Solution of Maxwell Equations.* Oxford University Press, November 1997.

[4] M. Chetty and R. Buyya. Weaving computational grids: How analogous are they with electrical grids? *J. R. Statistical Society*, July - August, 2001.

[5] I. Foster and C. Kesselman. *The Grid 2 : Blueprint for a computing Infrastructure.* Morgan Kaufmann: San Fransisco, 2003.

[6] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid : Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 2001.

[7] I. Foster, C. Kesselman, J. Nick, and Tuecke S. The physiology of the grid : An open grid services architecture for distributed systems integration. *WG, Global Grid Forum*, June 22, 2002.

[8] I. Foster. What is the grid? a three point checklist. *Grid Today*, 12002.

[9] I. Foster. The grid: A new infrastructure for 21st century science. *Physics Today*, 55(2):42–47, 2002.

[10] Distributed.net. http://www.distributed.net/.

[11] Seti@home. http://setiathome.ssl.berekley.edu/.

[12] L. Ferreira, V. Berstis, J. Armstrong, M. Kendziersky, A. Noeukotter, M. Takagi, R. Bing-Wo, A. Amir, R. Mukarakawa, O. Hernandez, J. Magowan, and N. Bierberstein. Introduction to the grid computing with globus. *IBM Redbook*, 2002.

[13] V. Berstis. Fundamentals of grid computing. *IBM Redbooks paper*, 2002.

[14] M. Baker, R. Buyya, and Laforenza D. Grids and grid technologies for wide-area distributed computing. *Soft. Pract. Exper.*, 2002.

[15] An introduction to grid computing from cern. http://www.gridcafe.org/grid-powered-project.html.

[16] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with nimrod/g: Killer application for the global grid? *International Parallel and distributed Processing Symposium (IPDPS). IEEE Computer Society Press: Los Alamitos, CA*, 2000.

[17] R. Buyya. The virtual laboratory project: Molecular modeling for drug design on grid (http://www.buyya.com/vlab/). *IEEE Distributed Systems Online*, 5(2), 2001.

[18] R. Buyya, K. Branson, J. Griddy, and D. Abramson. The virtual laboratory: A toolset to enable distributed molecular modeling of drug design on the world-wide grid. *Concurrency and Computation: Practice and Experience 2002*, 2002.

[19] S. Smallen et al. Combining workstations and supercomputers to support grid applications: The parallel tomography experience. *The 9th Heteregeneous Computing Workshop (HW 2000, IPDPS), Cancun, Mexico*, 61(2):479–482, April 2000.

[20] K. Holtman. Cms datagrid system overview and requirements. *The compact Muon Solenoid (CMS) Experiment Note 2001/037, CERN, Switzerland*, 2001.

[21] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-performance remote access to climate simulation data: A challenge problem for data grid technologies. *proceedings od SC2001 conference, denver, CO*, November 2001.

[22] S. Date and R. Buyya. Economic and on demand brain activity analysis on the grid. *The 2nd Pacific Rim Application and Grid Middleware Assembly Workshop, Seoul, Korea*, July 2002.

[23] Nasa information power grid. http://www.ipg.nasa.gov/.

[24] F. Berman, G. Fox, and T. Hey. Grid computing: Making the global infrastructure a reality. *Wiley, chapter1*, march2003.

[25] The open grid services architecture (ogsa). http://www.globus.org/ogsa/.

[26] L. Tarricone and A. Esposito. Grid computing for electromagnetics. *Artech House*, September 2004.

[27] Gecem - electromagnetic compatibility in aerospace design. http://www.wesc.ac.uk/projectsite/gecem/.

[28] The lumerical/westgrid partnership. http://www.westgrid.ca/files/webfm/-aboutdocs/Lumerical7.pdf.

[29] P. Lorenz, J. Vagner Vital, B. Biscontini, and P. Russer. Tlm-g: A grid-enabled time-domain transmission-line-matrix system for the analysis of complex electromagnetic structures. *IEEE Transactions on Microwave Theory and Techniques*, 53(11):3631–3637, 2005.

[30] Discogrid. http://www-sop.inria.fr/nachos/teammembers/Stephane.Lanteri/DiscoGrid/.

[31] E. Caron, C. Klein, and C. Perez. Efficient grid resource selection for a cem application. *RenPar'19, SympA'13, CFSE'7, Toulouse, France*, Sept. 2009.

[32] L. Tarricone and A. Esposito. Advances in information technologies for electromagnetics. *Springer Publishers*, September 2006.

[33] T. E. Athanaileas, P. K. Gkonis, G. E. Athanasiadou, F G. V. Tsoulos, and D. L. Kaklamani. Implementation and evaluation of a web-based grid-enabled environment for wcdma multibeam system simulations. *IEEE Antennas and Propagation Magazine*, 50(3), 2008.

[34] Grid'5000 homepage. http://www.grid5000.fr/.

[35] F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, M. Nouredine, G. Mornet, R. Namyst, P. Primet, and O. Richard. Grid'5000: a large scale, reconfigurable, controlable and monitorable grid platform. *Grid'2005 Workshop, Seattle, USA*, November 13-14, 2005.

[36] National telecommunication network for technology, education and research. http://www.renater.fr/.

[37] European high-bandwidth, academic internet geant2. http://www.geant2.net/.

[38] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. *Proceedings of IEEE International Symposium on Cluster Computing and the Grid, 2005, CCGrid 2005, Cardiff, United Kingdom*, pages 776–783, 2005.

[39] Y. Georgiou, J. Leduc, B. Videau, J. Peyrard, and O. Richard. A tool for environment deployment in clusters and light grids. *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 25-29 April 2006.

[40] Monika. http://oar.imag.fr/.

[41] Ganglia. http://ganglia.info/.

[42] Nagios. http://nagios.org/.

[43] G. Kron. Equivalent circuit of the field equations of maxwell i. *Proc. IRE*, 32:289–290, May 1944.

[44] W. J. R. Hoefer. The transmission-line matrix methodtheory and applications. *IEEE Trans. Microwave Theory Tech.*, MTT-33(10), Oct. 1985.

[45] C. Christopoulos. *The Transmission-Line Modeling Method (TLM).* New York: IEEE Press, 1995.

[46] N. Marcovitz and J. Schwinger. On the reproduction of the electric and magnetic fields produced by currents and discontinuities in wave guides, i. *J. Appl. Phys.*, 22(6):806–819, June 1951.

[47] P. B. Johns and R. L. Beurle. Numerical solution of 2-dimensionnal scattering problems using transmission line matrix. *Proc. IEEE*, 118(9):1203–1208, Sept. 1971.

[48] W. J. Hoefer. Huygens and the computer - a powerful alliance in numerical electromagnetic. *Proceeding of the IEEE*, 79:1459–1471, October 1991.

[49] P. B. Johns. A simple explicit and unconditionally stable numerical routine for the solution of the diffusion equation. *Int. J. Num. Meth. Eng.*, 11:1307–1328, 1977.

[50] GJ. Partridge, C. Christopoulos, and P.B. Johns. Transmission line modelling of shaft dynamic systems. *Proceedings of the institute of mechanical engineers*, 201:271–278, 1987.

[51] V. Trenkic, C. Christopoulos, and J.G.P. Binner. The application of the transmission line modelling (tlm) method in combined thermal and electromagnetic problem. *Proceedings of the international conference on numerical methods for thermal problems*, pages 1263–1274, 1993.

[52] F. J. German, G. K. Gothard, L. S. Riggs, and P. M. Goggans. The calculation of radar crosssection (rcs) using the tlm method. *Int. J. Numerical Modeling*, 2:267– 278, 1989.

[53] K. Umashankar and A. Taflove. A novel method to analyze electromagnetic scattering of complex objects. *IEEE transactions on electromagnetic compatibility*, EMC-24:397–405, November 1982.

[54] C. Huygens. Traité de la lumière. *Leiden*, 1690.

[55] D. Halliday and R. Resnick. Fundamental of physics. *John Wiley and Sons, third ed.*, 77, 1988.

[56] E. H. Barton. Huygence principle. *ch. 2*, pages 71–74, 1963.

[57] P. B. Johns. Application of the transmission-line matrix method to homogeneous waveguides of arbitrary cross-section. *Proc. Inst. Elec. Eng.*, 119(8):1086–1091, Aug. 1972.

[58] P. B. Johns. The solution of inhomogeneous waveguide problems using a transmission-line matrix. *IEEE Trans. Microwave Theory Tech.*, MTT-22:209–215, Mar. 1974.

[59] S. Akhtarzad and P. B. Johns. Numerical solution of lossy waveguides: T.l.m. computer program. *Electron. Lett.*, 10(15):309311, July 25, 1974.

[60] P. B. Johns. A new mathematical model to describe the physics of propagation. *Radio Electron. Eng.*, 44(12):657–666, Dec. 1974.

[61] S. Akhtarzad. Analysis of lossy microwave structures and microstrip resonators by the tlm method. *Ph.D dissertation, Univ. of Nottingham, England*, July 1975.

[62] S. Akhtarzad and P. B. Johns. Solution of maxwell's equations in three space dimensions and time by the t.l.m. method of analysis. *Proc. Inst. Elec. Eng.*, 112(12):13441348, Dec. 1975.

[63] S. Akhtarzad and P. B. Johns. Generalized elements for t.l.m. method of numerical analysis. *Proc. Inst. Elec. Eng.*, 112(12):13481352, Dec. 1975.

[64] G. E. Mariki. Analysis of microstrip lines on homogeneous anisotropic substrates by the tlm numerical technique. *Ph.D. thesis, Univ. of California, Los Angeles*, June 1978.

[65] W. J. R. Hoefer and A. Ros. Fin line parameters calculated with the tlm method. *IEEE MTT Int. Microwave Symp. Dig. (Orlando, FL)*, Apr. 28-May 2, 1979.

[66] N. Yoshida, L. Fukai, and J. Fukuoka. Transient analysis of two-dimensional maxwell's equations by bergeron's method. *Trans. IECE Japan*, J62B:511518, June 1979.

[67] P. Saguet and E. Pie. An improvement for the tlm method. *Electron. Lett.*, 16(7):247–248, Mar, 27, 1980.

[68] Y.-C. Shih, W. J. R. Hoefer, and A. Ros. Cutoff frequencies in fin lines calculated with a two-dimensional tlm-program. *IEEE MTT Int. microwave Symp. Dig. (Washington, DC)*, pages 261–263, June 1980.

[69] W. J R. Hoefer and Y.-C. Shih. Field configuration of fundamental and higher order modes in fin lines obtained with the tlm method. *URSI and Int. IEEE-AP Symp., Quebec, Canada*, June 2-6, 1980.

[70] N. Yoshida, I. Fukai, and J. Fukuoka. Transient analysis of three-dimensional electromagnetic fields by nodal equations. *Trans. IECE Japan*, J63B:876–883, Sept. 1980.

[71] A. Ros, Y.-C. Shih, and W. J. R. Hoefer. Application of an accelerated tlm method to microwave systems. *10fh Eur. Microwave Conf. Dig. (Warszawa, Poland)*, pages 382–388, Sept. 8-11, 1980.

[72] Y.-C. Shih and W. J. R. Hoefer. Dominant and second-order mode cutoff frequencies in fin lines calculated with a two-dimensional tlm program. *IEEE Trans. Microwave Theory Tech.*, MTT-28:14431448, Dec. 1980.

[73] Y.-C. Shih. The analysis of fin lines using transmission line matrix and transverse resonance methods. *M. A. SC. thesis, Univ. of Ottawa, Canada*, 1980.

[74] D. Al-Mukhtar. A transmission line matrix with irregularly graded space. *Ph.D. thesis, Univ. of Sheffield, England*, Aug. 1980.

[75] P. Saguet and E. Pie. Le maillage rectangulaire et le changement de maille dans la méthode tlm en deux dimensions. *Electron. Lett.*, 17(7):277278, Apr. 23 1981.

[76] D. A. Al-Mukhtar and J. E. Sitch. Transmission-line matrix method with irregularly graded space. *Proc. Inst. Elec. Eng.*, 128(6):299305, Dec. 1981.

[77] N. Yoshida, I. Fukai, and J. Fukuoka. Application of bergeron's method to anisotropic media. *Trans. IECE Japan*, J64B:12421249, Nov. 1981.

[78] P. Saguet and E. Pie. Utilisation d'un nouveau type de noeud dans la méthode tlm en 3 dimensions. *Electron. Lett.*, 18(11):478–480, May 1982.

[79] P. Saguet. Le maillage parallélipipédique et le changement de maille dans la méthode tlm en trois dimensions. *Electron. Lett.*, 20(5):222224, Mar. 15, 1984.

[80] N. Yoshida and I. Fukai. Transient analysis of a strip line having a corner in three-dimensional space. *IEEE Tran. Microwave Theory Tech.*, MTT-32:491–498, May 1984.

[81] D. H. Choi and W. J. R. Hoefer. The simulation of three-dimensional wave propagation by a scalar tlm model. *in IEEE MTT Int. Microwave Symp. Dig. (San Francisco)*, May 1984.

[82] S. Lindenmeier, L. Pierantoni, and P. Russer. Hybrid space discretizing integral equation methods for numerical modeling of transient interference. *IEEE Transactions on Electromagnetic Compatibility*, 41(4):425–430, November 1999.

[83] L. Pierantoni, G. Cerri, S. Lindenmeier, and P. Russer. Theoretical and numerical aspects of the hybrid mom-fdtd, tlm-ie and arb methods for the efficient modeling of emc problems. *Proc. 29th European Microwave Conference, Munich, Germany*, 1999.

[84] P. Lorenz and P. Russer. Characterization of complex 2d surface objects using the 3d transmission line matrix (tlm) method with a high-resolution mesh. *Proc. AP-S/URSI Symp., Washington D.C., USA*, July 3-8, 2005.

[85] P. Russer and U. Siart. Time domain methods in electrodynamics: A tribute to wolfgang j. r. hoefer. *New York: Springer-Verlag*, Oct. 2008.

[86] N. Fichtner and P. Russer. A total-field/scattered-field technique applied for the tlm-integral equation method. *IEEE MTT-S Int. Microwave Symp. Dig, Boston, USA*, pages 325–328, June 2009.

[87] P. Russer. Electromagnetic field computation by network methods. *Proc. of the 25th Annual Review of Progress in Applied Computational Electromagnetics ACES, Monterey, California USA*, March 2009.

[88] M. N. O. Sadiku and L. C. Agba. A simple introduction to the transmission line modeling. *IEEE Trans. Cir. Sys.*, CAS-37(8):991–999, Aug. 1990.

[89] KK. Fung, SYR. Hui, and C. Christopoulos. Concurrent programming and simulation of decoupled power electronic circuits. *IEE Proceedings Science Measurement and Technology*, 49:1–13, 1996.

[90] C. C. Tan and V. F. Fusco. Tlm modeling using an simd computer. *International journal of numerical modeling*, 6:299–304, 1993.

[91] P.P.M. So, C. Eswarappa, and W.J.R. Hoefer. Transmission line matrix on massively parallel processor computers. *9th Annual Review of*

Progress in Applied Computational Electromagnetics Monterey, California USA, pages 467–474, March 1993.

[92] P.P.M. So, C. Eswarappa, and W.J.R. Hoefer. Distributed computing for transmission line matrix. *Second International Workshop on Time Domain Modeling of Field and Networks, Berlin Germany*, October 1993.

[93] P.P.M. So, C. Eswarappa, and W.J.R. Hoefer. Optimization of microwave structures using a parallel tlm module. *10th Annual Review of Progress in Applied Computational Electromagnetics Digest, Monterey, California USA*, pages 546–553, March 1994.

[94] P.P.M. So, C. Eswarappa, and W.J.R. Hoefer. Massively parallel and distributed computing and digital signal processing for tlm electromagnetic field modeling. *IEEE APS, Seattle, Washnigton USA*, pages 546–553, June 1994.

[95] P.P.M. So, C. Eswarappa, and W.J.R. Hoefer. Distributed parallel tlm computation and signal processing for electromagnetic field modeling. *Invited Paper, International journal of numerical modeling: Electronic network, device and fields, John Wiley and Sons Inc*, 8(3/4):169–185, August 1995.

[96] P.P.M. So and W.J.R. Hoefer. A multi-threaded time domain tlm algorithm for symmetric multi-processing computers. *IEEE MTT-S, Phoenix, Arizona USA*, pages 2007– 2010, May 2001.

[97] B. Isele, J. Schmoller, and P. Russer. Simulation of coplanar resonators with tlm method in a parallel computing environment. *PIERS 95 Seattle*, page 733, July 1995.

[98] Parallel virtual machine (pvm). http://www.csm.ornl.gov/pvm/pvm_home.html.

[99] B. Isele and P. Russer. Tlm modelng of microwave circuits by distributed computing. *PIERS 96 Innsbruck*, 145, July 1996.

[100] B. Isele and P. Russer. The modeling of coplanar circuits in a parallel computing environment. *MTT-Symposium San Francisco*, 2:1035–1038, June 1996.

[101] H.G. Sasse and A.P. Duffy. Implementation of a parallel distributed tlm solver. *Fifth IEEE International Conference on Computation in Electromagnetics*, pages 29 –30, 2004.

[102] D. Thomas and A. Hunt. *Programming Ruby: The Pragmatic Programmer's Guide*. Addison Wesley, 2001.

[103] W. Gropp. Mpich2: A new start for mpi implementations. *In Recent Advances in Parallel Virtual Machine and Message Passing Interface: 9th European PVM/MPI Users Group Meeting, Linz, Austria*, October 2002.

[104] Gridmpi project. http://www.gridmpi.org/gridmpi.jsp.

[105] O. Aumage and G. Mercier. Mpich/madiii : a cluster of clusters enabled mpi implementation. *Proceedings of 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan*, 2003.

[106] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, Ra. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open mpi: Goals, concept, and design of a next generation mpi implementation. *In Proceedings, 11th European PVM/MPI Users Group Meeting, Budapest, Hungary*, pages 97–104, September 2004.

[107] I. Foster, N. Karonis, and B. Toonen. Mpich-g2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, pages 551–563, 2003.

[108] L. Hablot, O. Gluck, J-C. Mignot, and P. Vicat-Blanc Primet. Etude d'implémentations mpi pour une grille de calcul. *RenPar'18, SympA'2008, CFSE'6, Fribourg, Suisse*, 11-13 Feb. 2008.

[109] G. M. Rebeiz. *RF MEMS: Theory, Design, and Technology.* New York: J. Wiley and Sons, 2003.

[110] D. Cadoret, A. Laisne, R. Gillard, and H. Legay. Design and measurement of new reflectarray antenna using microstrip patches loaded with slots. *Electronic Letters*, 41(11):623–624, May 2005.

[111] J. Huang. Microstrip reflectarray. *IEEE APS, Ontario, Canada*, 1991.

[112] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer (http://www.research.ibm.com/autonomic/research/papers/ACVisionComputerJan2003.pdf)*, 3(1):41–50, 2003.

[113] L. Broto, D. Hagimont, P. Stolf, N. Depalma, and S. Temate. Autonomic management policy specification in tune. *23rd Annual ACM Symposium on Applied Computing, Fortaleza, Brazil*, 1March 2008.

[114] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, and J.B. Stefani. An open component model and its support in java. *Proceedings of the 7th International Symposium on Component-Based Software Engineering (CBSE-7). Lecture Notes in Computer Science, Springer (2004)*, 3054(2):7–24, 2004.

[115] G. Huard and C. Martin. Taktuk. http://taktuk.gforge.inria.fr/.

[116] Ansoft hfss homepage. http://www.ansoft.com/products/hf/hfss/.

[117] H. Aubert. The concept of scale-changing network in the gobal electromagnetic simulation of complex structures. *Progress Electromagnetics Research B*, 16:127–154, 2009.

[118] B. A. Munk. *Frequency Selective Surfaces: Theory and Design*. First Edition, Wiley, New York, 2000.

[119] W.L. Ko and R. Mittra. Implementation of floquet boundary condition in fdte for fss analysis. *IEEE APS, Int. Symp.Dig.*, June28-july 2 1993.

[120] M. Nadarassin, H. Aubert, and H. Baudrand. Analysis of planar structures by an integral multi-scale approach. *IEEE MTT-S International Microwave Symposium, Orlando, Florida, USA*, 2:653–656, May 14-19, 1995.

[121] H. Baudrand and S. Wane. Circuits multi-échelles: Utilisation des sources auxiliaires. *Modélisation Caractérisation et Mesures de Circuits Intégrés Passifs R.F, Hermès*, 3:75–108, 2003.

[122] H. Baudrand. study of coupling between active and passive circuits. *Microwave and Optoelectronics Conference*, 1:143–152, 1997.

[123] E. Perret, H. Aubert, and H. Legay. Scale-changing technique for the electromagnetic modeling of mems-controlled planar phase-shifters. *IEEE Trans. Microwave Theory and Tech.*, 54(9):3594–3601, Sept. 2006.

[124] E. Perret, N. Raveu, H. Aubert, and H. Legay. Scale-changing technique for mems-controlled phase-shifters. *36th European Microwave Week, Manchester, United Kingdom*, pages 866–869, Sep. 10-15, 2006.

[125] D. Voyer, H. Aubert, and J. David. Scale-changing technique for the electromagnetic modeling of planar self-similar structures. *IEEE Trans. Antennas Propagat.*, 54(6):2783–2789, Oct. 2006.

[126] D. Voyer, H. Aubert, and J. David. Radar cross section of discrete self-similar objects using a recursive electromagnetic analysis. *IEEE Antennas and Propagation Society International Symposium, Monterey, California, USA*, 4:4260–4263, Jun. 20-26, 2004.

[127] D. Voyer, H. Aubert, and J. David. Electronics letters. *Radar cross section of self-similar targets*, 41(4):215–217, Feb. 17, 2005.

[128] E. Perret and H. Aubert. A multi-scale technique for the electromagnetic modeling of active antennas. *IEEE Antennas and Propagation Society International Symposium, Monterey, California, USA*, 4:3923–3926, Jun. 20-25, 2004.

[129] E. Perret and H. Aubert. Scale-changing technique for the computation of the input impedance of active patch antennas. *IEEE Antennas and Wireless Propagation Letters*, 4:326–328, 2005.

[130] N. Raveu, G. Prigent, H. Aubert, P. Pons, and H. Legay. Scale-changing technique design and optimization tool for active reflect-arrays cell. *37th European Microwave Conference, Munchen, Germany*, pages 736–739, Oct. 9-12, 2007.

[131] N. Raveu, E. Perret, H. Aubert, and H. Legay. Design of mems controlled phase shifter using sct. *PIERS Online*, 3(2):230–232, Mar. 26-30, 2007.

[132] N. Raveu, E. Perret, H. Aubert, and H. Legay. Scale-changing technique: A design tool for reflectarrays active cells. *Proceedings of the European Microwave Association*, 4(2):163–168, Jun. 2008.

[133] F. Khalil, H. Aubert, F. Coccetti, R. Plana, Y. Deunneulin, B. Miegemolle, T. Monteil, and H. Legay. Electromagnetic simulation of mems-controlled reflectarrays based on sct in grid environment. *IEEE International Symposium on Antennas and Propagation, Honolulu, Hawaii, USA*, June 10-15, 2007.

[134] F. Khalil, C. J. Barrios-Hernandez, H. Aubert, Y. Denneulin, F. Coccetti, and R. Plana. Electromagnetic simulations via parallel computing: an application using scale changing technique for modeling of passive planar reflectarrays in grid environment. *2008 IEEE International Symposium on Antennas and Propagation and the 2008 USNC/URSI National Radio Science meeting, San Diego, California*, July 5-12, 2008.

[135] F. Khalil, A. Rashid, H. Aubert, F. Coccetti, R. Plana, C.-J. Barrios-Hernandez, and Y. Denneulin. Application of scale changing technique-grid computing to the electromagnetic simulation of reflectarrays. *2009 IEEE International Symposium on Antennas and Propagation and the 2009 USNC/URSI National Radio Science meeting, Charleston, South Carolina*, June 1-7, 2009.

[136] E. B. Tchikaya, A. Rashid, F. Khalil, H. Aubert, H. Legay, and N. J.G. Fonseca. Multi-scale approach for the electromagnetic modeling of metallic fss grids of finite thickness with non-uniform cells. *2009 Asia-Pacific Microwave Conference (APMC 2009), Singapore*, Dec. 7-10, 2009.

[137] Matlab. mathworks inc. internet site. http://www.mathworks.com/.

[138] V. Saladino. Automating tasks with expect. *LinuxJournal*, October 1st, 1998.

[139] K. Seymour, C. Lee, F. Desprez, H. Nakada, and Y. Tanaka. The end-user and middleware apis for gridrpc. *Workshop on Grid Application Programming Interfaces, In conjunction with GGF12, Brussels, Belgium*, Sep. 2004.

[140] M. Henning and S. Vinoski. *Advanced CORBA(R) Programming with C++*. Addison-Wesley Pub Co, 1999.

[141] Ie3d-zeland software. http://www.zeland.com/ie3d.htm.

[142] The official site of openmp. http://openmp.org/wp/.

[143] NVIDIA CUDA. *Compute Unified Device Architecture Programming Guide*. NVIDIA Corp., 2008.

[144] C. Balanis. *Antenna Theory: Analysis and Design*. John wiley and sons, 2005.