POLITECNICO DI TORINO
Repository ISTITUZIONALE

Grid Infrastructure for Domain Decomposition Methods in Computational ElectroMagnetics

Original
Grid Infrastructure for Domain Decomposition Methods in Computational ElectroMagnetics / Terzo, O., Ruiu, P., Mossucca, L., Francavilla, M.A., Vipiana, F. - In: Grid Computing - Technology and Applications, Widespread Coverage and New HorizonsELETTRONICO. - Croazia : Soha Maad (Ed.), 2012. - ISBN 9789535106043. - pp. 247-266 [10.5772/36691]

Availability:
This version is available at: 11583/2497854 since:

Publisher:
Soha Maad (Ed.)

Published
DOI:10.5772/36691

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

06 September 2026

11

Grid Infrastructure for Domain Decomposition Methods in Computational ElectroMagnetics

Olivier Terzo, Pietro Ruiu, Lorenzo Mossucca,
Matteo Alessandro Francavilla and Francesca Vipiana
Istituto Superiore Mario Boella (ISMB)
Italy

1. Introduction

The accurate and efficient solution of Maxwell's equation is the problem addressed by the scientific discipline called Computational ElectroMagnetics (CEM). Many macroscopic phenomena in a great number of fields are governed by this set of differential equations: electronic, geophysics, medical and biomedical technologies, virtual EM prototyping, besides the traditional antenna and propagation applications. Therefore, many efforts are focussed on the development of new and more efficient approach to solve Maxwell's equation. The interest in CEM applications is growing on. Several problems, hard to figure out few years ago, can now be easily addressed thanks to the reliability and flexibility of new technologies, together with the increased computational power. This technology evolution opens the possibility to address large and complex tasks. Many of these applications aim to simulate the electromagnetic behavior, for example in terms of input impedance and radiation pattern in antenna problems, or Radar Cross Section for scattering applications. Instead, problems, which solution requires high accuracy, need to implement full wave analysis techniques, e.g., virtual prototyping context, where the objective is to obtain reliable simulations in order to minimize measurement number, and as consequence their cost. Besides, other tasks require the analysis of complete structures (that include an high number of details) by directly simulating a CAD Model. This approach allows to relieve researcher of the burden of removing useless details, while maintaining the original complexity and taking into account all details. Unfortunately, this reduction implies: (a) high computational effort, due to the increased number of degrees of freedom, and (b) worsening of spectral properties of the linear system during complex analysis. The above considerations underline the needs to identify appropriate information technologies that ease solution achievement and fasten required elaborations. The authors analysis and expertise infer that Grid Computing techniques can be very useful to these purposes. Grids appear mainly in high performance computing environments. In this context, hundreds of off-the-shelf nodes are linked together and work in parallel to solve problems, that, previously, could be addressed sequentially or by using supercomputers. Grid Computing is a technique developed to elaborate enormous amounts of data and enables large-scale resource sharing to solve problem by exploiting distributed scenarios. The main advantage of Grid is due to parallel computing, indeed if a problem can be split in smaller tasks, that can be executed independently, its solution calculation fasten up considerably. To exploit this advantage, it is necessary to identify a technique able to split original electromagnetic task into a set of smaller subproblems. The Domain Decomposition (DD) technique, based on the block generation algorithm introduced in Matekovits et al.

(2007) and Francavilla et al. (2011), perfectly addresses our requirements (see Section 3.4 for details). In this chapter, a Grid Computing infrastructure is presented. This architecture allows parallel block execution by distributing tasks to nodes that belong to the Grid. The set of nodes is composed by physical machines and virtualized ones. This feature enables great flexibility and increase available computational power. Furthermore, the presence of virtual nodes allows a full and efficient Grid usage, indeed the presented architecture can be used by different users that run different applications.

The chapter is organized as follow, Section 2 briefly explains author's contribution. Section 3 describes technologies used and summarized Domain Decomposition principles. The architecture is shown in Section 4, while the advantages derived by its adoption are illustrated in Section 5 and Section 6 draws conclusions and gives hints for future works.

2. Motivation

Due to the decomposition of the original problems into a larger number of subproblems, the scheme is well suited to a parallelization approach, since the subproblems (which will be referred to as blocks in the following) are disjoint, and the elaboration of the blocks is intrinsically a parallelizable operation. Since the generation of the entire domain basis functions on each block is independent from the other blocks, a high scalability is expected. As an example, Figure 1 shows a fighter subdivided into 4 different blocks, identified by different colors. The four blocks can be processed in parallel by four different processes, in order to generate the basis functions describing each block. Parallelization can be achieved using two different techniques: parallel programming or Grid Computing. Parallel programming is the technique by which have been obtained the best results in this field (500 billion of unknowns) Mouriño et al. (2009). There are important barriers to the broader adoption of these methodologies though: first, the algorithms must be modified using parallel programming API like MPI (2011) and OpenMP (2011), in order to run jobs on selected cores. The second aspect to consider is the hardware: to obtain results of interest supercomputers with thousands of cores and thousands of GB of RAM are required. Typically these machines are made by institutions to meet specific experiments and do not always allow public access. Grid Computing is a rather more easily applicable model for those who do not fulfill the requirements listed above Foster & Kesselman (2003). For its adoption the only requirement is to have at disposal computers to use within the grid. The machines may be heterogeneous, both in terms of the types (workstations, servers) and hardware resources of each machine (RAM, CPU, HD); besides they can also be recovery machines. With the Virtualization technology the number of processes running on each machine (for multicore ones) can be optimized by instantiating multiple virtual nodes on the same physical machine. The real advantage of this model, however, is that researchers do not have to worry about rewriting code to make their application parallelizable. One of the drawbacks in the adoption of this model is the addition of overhead introduced by the grid (e.g., inputs time transfer) and by the hypervisor that manages the virtual machines. It has been shown, however, that the loss of performance due to the hypervisor is not particularly high, usually not exceeding 5% Chierici & Verald (2010). For the above reasons, in our case it was decided to grid infrastructure, composed of heterogeneous recovery machines, both physical and virtual, on which to run scientific applications without the need to modify the source codes of the used algorithms. The ultimate goal is the reduction of execution times of CEM applications.

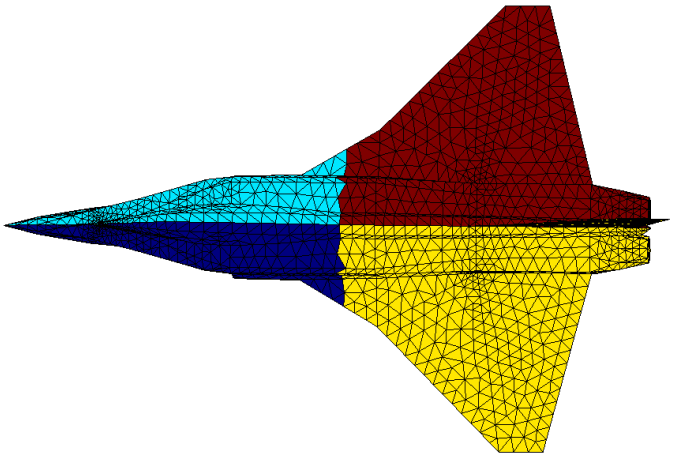

Fig. 1. A fighter discretized with 4774 triangles is subdivided into 4 blocks, shown with different colors.

3. Background

The rapid technology growth in the recent years has helped the development and rapid expansion of Grid Computing. "The roots of Grid Computing can be traced back from the late 80s when the research about scheduling algorithms for intensive applications in distributed environments accelerated considerably" Kourpas (2006). In the late 1990s began to emerge, a more generalized framework for accessing high performance computing systems, and at the turn of the millennium, the pace of change was accelerated by the recognition of potential synergies between the grid and the emerging Service Oriented Architectures (SOA) through the creation of the Open Grid Services Architecture (OGSA) Kourpas (2006). "The Open Grid Services Architecture is a set of standards defining the way in which information is shared among several components of large, heterogeneous grid systems. The OGSA is, in effect, an extension and refinement of the Service Oriented Architecture" OGSA (2007). The Open Grid Services Architecture is a standard created by the Open Grid Forum (OGF) OGF (2011). OGF was founded in 2006 from the Global Grid Forum (GGF) and the Enterprise Grid Alliance (EGA). GGF had a rich background and established international presence within the academic and research communities while EGA was focused on developing and promoting enterprise grid solutions. The OGF community now counts thousands of members working in research and industry, representing more than 400 organizations in 50 countries.

3.1 Grid Computing technologies

Grid Computing is often described by referring to the analogy between electrical networks and grid. When people access to electric network they use wall sockets with no care about where or how electricity is actually generated. This relation underlies that computing becomes pervasive thanks to Grid Computing diffusion. Therefore, individual users (or client applications) can access computing resources (processors, storage, data, applications, etc..) when needed with little or no knowledge about where those resources are located or what underlying technologies, hardware, operating system are used. A further definition is given by "Grid is an infrastructure that involves the integrated and collaborative use of

computers, networks, databases and scientific instruments owned and managed by multiple organizations" Asadzadeh et al. (2005). Grid Computing is based on these technology principles Oracle (2009):

Standardization on operating systems, servers, storage hardware, middleware components, and network components extends interoperability and reduce system management overhead. It also improves operational complexity reduction in data center by simplifying application deployment, configuration, and integration.

Virtualization of resources means that applications are not bound to a specific server, storage, or network components but can be used in any virtualized resource. Virtualization is realized thanks to a sophisticated software layer that hides the underlying complexity of hardware resources and presents a simplified interface used by applications and other resources (see Section 3.2).

Automation Grid Computing requires large-scale automation of IT operations due to the potentially very high number of components, both virtual and physical. Each component requires configuration management, on-demand provisioning, top-down monitoring, and other management tasks. Combining these capabilities into a single, automated, integrated solution for managing grids enables organizations to maximize their return of investment.

3.1.1 The Globus Toolkit

The Globus Toolkit (Globus (2010)) developed by the Globus Alliance, is an open source software toolkit used for building grid systems and applications. The Globus Alliance includes ISI, the University of Chicago, the University of Edinburgh, the Royal Institute of Technology in Sweden, the National Center for Supercomputing Applications, and Univa Corporation. Sponsors include federal agencies such as DOE, NSF, DARPA, and NASA, along with commercial partners such as IBM and Microsoft.

The Globus Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop applications. It is used by various companies and organizations as the basis for grid implementations of various types.

Globus Toolkit is composed by four main components (shown in Figure 2):

- Security (GSI: Grid Security Infrastructure). It is a set of tools, libraries and protocols used in Globus to allow users and applications to securely access resources. GSI is based on a public key infrastructure, (PKI) with certificate authorities (CA) and (X509) certificates. GSI uses (SSL) for authentication and message protection and enables user to create and delegate proxy credentials to processes running on remote resources;
- Resource Management (GRAM: Grid Resource Allocation Manager). It provides the user to access the grid in order to run, terminate and monitor jobs remotely;
- Information Services are for providing configuration and adaptation capabilities for heterogeneous resources:
  - MDS: Monitoring and Discovery Service: provides information about the available resources on the Grid and their status;
  - GRIS: Grid Resource Information Service: is associated with each resource and answers queries from client for their current configuration, capabilities and status;

  - GIIS: Grid Index Information Service: is a directory service that pulls information for GRIS's. It is a "caching" service which provides indexing and searching functions.
- Data Management GridFTP: is a protocol that provides for the secure, robust, fast and efficient transfer of data.

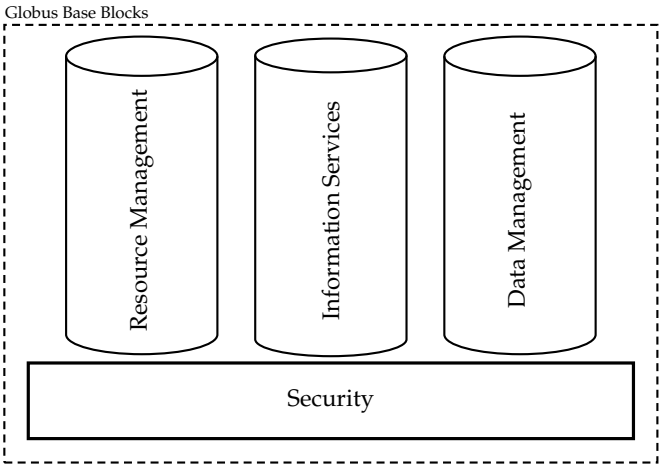

Fig. 2. Globus Toolkit Protocols.

3.2 Virtualization technologies

The idea of computer system virtualization is not new, the earliest examples dates back to the 60s when IBM introduced the IBM 7044 and when Manchester University presented the Atlas project (one of the first world's supercomputers). These systems present the first rudimental example of demand paging and supervisor calls.

However, in the last ten years, the use of virtualization in modern data centers increased mainly due to contain operating. Virtualization is a technology that allows running several concurrent Operating System (OS) instances inside a single physical machine called host. The physical device is divided into multiple isolated virtual environments called guest system. A Virtual Machine (VM) is an instance of the physical machine and gives users the illusion of accessing the physical machine directly and each VM is a fully protected and isolated copy of the underlying system. Virtualization is thus used to reduce the hardware costs on one side and to improve the overall productivity by letting many more users work on it simultaneously. Moreover the global cost and electricity power consumption makes the virtualization adoption convenient. Furthermore, the server number can rationalized, while maintaining the functional capacity of the system. A virtualization layer provides the required infrastructural support exploiting lower-level hardware resources in order to create multiple independent virtual machines that are isolated from each other. This layer, traditionally called Virtual Machine Monitor, usually sits on top of the hardware and below the operating system.

The hypervisor, or Virtual Machine Manager (VMM), allows multiple operating systems to concurrently run on a single host computer. It is so named because it is conceptually one level higher than a supervisory program. A supervisory program or supervisor is usually part of an operating system, that controls the execution of other routines and regulates work

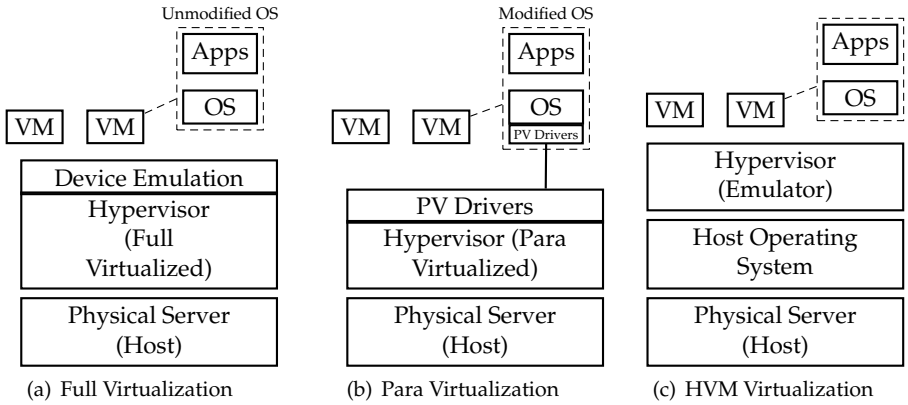

Fig. 3. Virtualization approaches.

scheduling, input/output operations, error actions, and similar functions and regulates the flow of work in a data processing system. The supervisor is generally called kernel. The hypervisor presents to the guest OS a virtual hardware platform and manages the execution of the guest OS. Multiple instances of different operating systems may share the virtualized hardware resources. Hypervisors are installed on server hardware whose only task is to run guest operating systems. Non hypervisor virtualization systems are used for similar tasks on dedicated server hardware, but also commonly on desktop, portable and even handheld computers. There are three popular approaches to server virtualization: Full Virtualization, Para Virtualization and virtualization with hardware support (Hardware Virtual Machine, or HVM).

Full Virtualization provides emulation of the underlying platform on which a guest operating system and application set run without modifications and unaware that the platform is virtualized. This approach is idealistic, in real world scenarios virtualization comes with costs.

Providing a Full Virtualization implies that every platform device is emulated with enough details to permit the guest OS to manipulate them at their native level (such as register-level interfaces). Moreover, it allows administrators to create guests that use different operating systems. These guests have no knowledge about the host OS since they are not aware that the hardware they see is not real but emulated. The guests, however, require real computing resources from the host, so they use a hypervisor to coordinate instructions to the CPU. The hypervisor provides virtual machine to show all the needed hardware to start and run the operating system. Via a virtual bios, it shows CPU, RAM and storage devices. The main advantage of this paradigm concerns the ability to run virtual machines on all popular operating systems without requiring them to be modified since the emulated hardware is completely transparent.

Para Virtualization Machine approach is based on the host-guest paradigm and uses a virtual machine monitor. In this model the hypervisor modifies the guest operating system's code in order to run as a guest operating system in a specific virtual machine environment. Like virtual machines, Para Virtual Machines are able to run multiple operating systems. The main advantage of this approach is the execution speed, always faster than HVM and Full Virtualization approach. The Para Virtualization method uses

a hypervisor for shared access to the underlying hardware but integrates virtualization aware code into the OS itself. In a context of Para Virtualization the guest operating system must be aware of being run in a virtual environment. So the original operating system, in particular its kernel, is modified to run in a Para Virtualized environment.

Hardware Virtual Machine (HVM) Hardware-assisted virtualization is a virtualization approach that enables efficient Full Virtualization using help from hardware capabilities. Last innovations in hardware, mainly in CPU, MMU and memory components (notably the Intel VT-x and AMD-V architectures), provide direct platform-level architectural support for OS virtualization. For some hypervisors (like Xen and KVM) it is possible to recompile Para Virtualized drivers inside the guest machine running in HVM environment and load those drivers into the running kernel to achieve Para Virtualized I/O performance within an HVM guest.

3.2.1 Virtualization considerations

As anticipated, several advantages ensue from virtualization use. The number reduction of physical servers is one of the most evident: the hardware (software) solution for virtualization allows multiple virtual machines running on one physical machine, moreover additional advantages are power consumption reduction, better fault tolerance, optimization of time needed for device installation and of the number of cabinets. Another advantage is that virtualization can be used to abstract hardware resources, since operating systems are closely related to the underlying hardware, several issues need to be taken into account when a server is moved or cloned, e.g., incompatible hardware, different drivers and so on. Another virtualization feature is the creation of abstraction levels such that the operating system does not see the actual physical hardware, but a virtualized one. Administrator can move or clone a system on other machines that run the same virtualization environment without worrying about physical details (network and graphics cards, chipsets, etc.).

Another important aspect is adaptability: if a company changes its priorities and needs, a service may become more important than another or may require more resources. Virtualization allows resource allocation to virtual hardware easily and quickly. Some companies maintain old servers with obsolete operating systems that cannot be moved to new servers as these OS would not be supported. In virtualized environments it is possible to run legacy systems allowing IT managers to get rid of old hardware no longer supported, and more prone to failure. In many cases it is appropriate to use virtualization to create test environments. It frequently happens that production systems need to be changed without knowledge about consequences, i.e., installing an operating system upgrade or a particular service pack is not a risk-free. Virtualization allows immediate replication of virtual machines in order to run all necessary tests.

Like any other technology, the virtualization gives disadvantages that depends on the application scenario. The most important are performance overhead and presence of hardware virtualization. Each virtualization solution is decreasing the overall performance, such as disk access times or access to memory. Some critical applications may suffer from this overhead introduced by virtualization. Depending on the products used, some devices may not be used by virtual machines, such as serial and parallel ports, USB and Bluetooth interfaces, or graphics acceleration hardware.

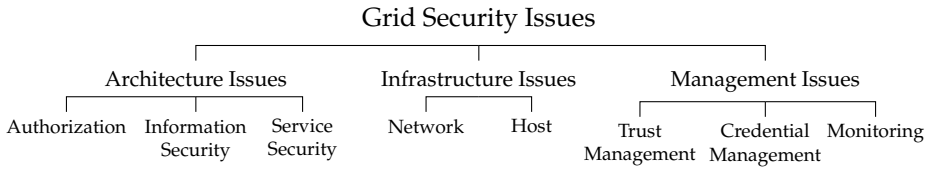

Fig. 4. Taxonomy of grid security issues.

3.2.2 Xen and KVM based virtualization

Xen is a Virtual Machine Monitor that allows several guest operating systems to be executed on the same computer hardware concurrently. A Xen system is structured with the Xen hypervisor as the lowest and most privileged layer. Above this layer are located one or more guest operating systems, which the hypervisor schedules across the physical CPUs. Xen can work both in Para Virtualized or HVM mode; in the first the guest operating system must be modified to be executed. Through Para Virtualization, Xen can achieve very high performance. The HVM mode offers new instructions to support direct calls by a Para Virtualized guest/driver into the hypervisor, typically used for I/O or other so-called hypercalls. KVM is a Full Virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). KVM is implemented as a module within the Linux kernel. A hypervisor hosts the virtual machine images as regular Linux processes, so that each virtual machine image can use all of the features of the Linux kernel, including hardware, security, storage, and applications. KVM supports I/O Para Virtualization using the so called VIRTIO subsystem consisting of 5 kernel modules IBM (2010).

3.3 Security overview

The basis idea of pooling has been always employed by humans. Its most evident and valuable advantage is cost and resource optimization, but it hides facets that may shadows its benefits. Whenever we share something we are worried since our goods may not be handle properly and may be manipulated by strangers. Moreover when we use someone else stuff we worry about safety since objects may be dangerous, broken or compromised.

The above concept perfectly fits grid system since grid can be seen as a mechanism to pool resources to optimize system utilization. Therefore we can state that security and trust, together with resource monitoring, authentication and authorization of users, and data protection, are essential to Grid Computing. In the following we give an overview of security in Grid Computing by using the classification proposed by Chakrabarti (2007). As shown in Figure 4, he categorized grid security issues into three main categories: architecture, infrastructure, and management related issues.

3.3.1 Architecture issues

Architecture level issues concerns threats - pertaining information security (e.g., data confidentiality and integrity), authorization, and service level security - that impact the whole grid system.

Information Security is defined as the security properties of data transmission between hosts, that is, mainly, secure communication and authentication. These aspects involve confidentiality (i.e., the data can be accessed only by legitimate users) and integrity (i.e., the data has not been modified during transmission). These aspects are essential security

requirements in all information and communication areas, but become extremely critical in distributed and heterogeneous environment like grids. The Open Grid Forum released an open standard called Open Grid Standards Architecture OGSA (2007) which is the referring point for worldwide researchers. OGSA specifies a layer called Grid Security Infrastructure GSI (2010) which aim is to undertake these matters, for further details see Section 3.3.4. The GSI is based on X.509 infrastructure and Secure Socket Layer (SSL) protocol, and uses public key cryptography and certificates for creating secure grid and application level data encryption. The X.509 certificates are used to ensure authentication: every user or service owns a certificate which contains needed information to identify and authenticate the owner.

Authorization is a security feature that implies the definition of "who" can access "what", clearly this definition can be extended by adding "when", "how", "how long", and so on. In the grid environment authorization is very important as resources are shared between several users and services. The Authorization Systems (AS) can be classified into two groups: (1) Virtual Organization (VO) Level and (2) Resource Level Systems. The former is centralized AS for an entire VO: when an user needs to access a resource of Resource Provider (RP) he requests for credentials to the AS and presents them to the RP which allows/denies rights to the user. The latter, instead, specifies the authorization to a set of resources, these systems allow the (authenticated) users to access the resources based on the credentials he presents. Examples of the first authorization systems are Community Authorization Service CAS (2010) and Virtual Organization Membership Service VOMS (2003), while example of the second are Privilege and Role Management Infrastructure Standards Validation PERMIS (2002), and the GridMap system.

Service Security entails the implementation of protection mechanisms focussed to guarantee the availability of services. Attack class examples are QoS violation and Denial-of-Service (DoS). The first is achieved through traffic congestion, packet delay or drop, while the second aims to cause software crash and to completely disrupt the service. The countermeasures to contain these attacks are mainly based on prevention and monitoring in order to limit damages and to raise alarms. Some approaches also try to detect the source, but an effective defense against these attacks is really hard to achieve.

3.3.2 Infrastructure issues

Infrastructure threats regards network and host devices that form the grid infrastructure and are classified in Host level and Network level. These issues impact data protection, job protection, host availability, access control functionalities, secure routing and all the communication aspects.

Host Security impacts data protection and job starvation. The former regards the protection of data already stored in the host, in fact the host submitting the job may be untrusted and the job code may be malicious. The latter is a scenario in which resources assigned to a job are denied to the original job and assigned to a different (malicious) job. The most effective countermeasures to limit data threats are: (1) application level sandboxing, (2) virtualization, and (3) sandboxing. The first approach uses proof carrying code (PCC), the compiler creates proofs of code-safeness and embed those in the executable binary. The second solution is based on the creation of Virtual Machines upon the physical host, this technique ensures strong isolation between different VMs and the host system. The third method confines system calls and sandboxes (isolates) the applications to avoid data and memory access not allowed. The solution adopted to avoid job starvation are based on

resource booking or priority reduction for long running jobs, in order to reduce starvation likelyhood.

Network Security is a core requirement in grid scenario due to high speed needs and host heterogeneity. Access control and isolation are fundamental to the grid networks. Solutions for Grid Computing may not work effectively with existing firewalls and virtual private networks (VPN), for this reason researcher developed solutions like Adaptive Grid Firewalls (AGF) and Hose. Moreover routing attacks can be very dangerous for grid working, countermeasures to these threats came from the traditional networking research and foresee the deploy of secure routing protocol.

3.3.3 Management issues

Management issues are very delicate as the grid is an heterogeneous environment composed of several entities, users, domains, and policies. The management problem can be seen as three distinct, but correlated, points: (1) credential management, (2) trust management, and (3) monitoring.

Credential Management is a complex and delicate task due to the distributed and numerous components that form the grid. Each of these requires rights to access resources that need to be trusted and non compromised. This aim is achieved by using credential management mechanisms that securely store, grant, revoke, and renew credentials for user and system. Some solutions move the burden to store credential from the user to the system, e.g., by using smart cards. Other approaches resort to the federated identity paradigm to manage credentials, across different systems, domains, and frameworks. Implementation example of the first family is MyProxy, while KX.509 (a protocol which enables interoperability between X.509 and Kerberos), Liberty Framework and Shibboleth are examples of the second one.

Trust Management is a critical aspect in grid since nodes and users continuously join and leave the system. Therefore a mechanism to manage trust levels of users, nodes and the grid itself is mandatory. Different trust management solutions have been developed, their key features are scalability, reliability, and security and can be grouped into two main categories: reputation based and policy-based systems. The formers are based on trust metrics taken from local and global reputation of a resource or an host. In the latter approach, the different units that compose the system, exchange and manage credentials to create trust connections given a set policies.

Monitoring of resources is necessary in grid due to two main reasons. Firstly, organizations can be charged according to grid utilization, and, secondly, resource information can be logged for auditing, debugging, testing and security purposes. Different monitoring system are available in literature and can be grouped into three categories: (1) system level, (2) cluster level, and (3) grid level. The first systems collect and transmit data related to standalone systems or networks. The second ones require deployment across clusters and gather information upon the cluster itself. The thirds are more flexible than the formers because they can be deployed on top of other monitoring systems and may provide interfaces for querying, and displaying data in standard formats.

3.3.4 Grid Security Infrastructure (GSI)

The definition and implementation of a robust infrastructure is one of the main issue when the problem of securing grid is investigated. The Grid Security Infrastructure GSI (2010)

addresses exactly this issue, it defines security requirements and provides a framework to provide security in Virtual Organization based grid systems. GSI has been provided by the Global Grid Forum (GGF) that is a forum of researchers and practitioners with the aim to exchange information and to define standards for Grid Computing. One of the most important aspects of GSI is that it is not only a theoretical definition, but it is implemented and used worldwide thanks to Globus Toolkit Globus (2010). GSI handles different security requirements, that can be summarized in: authentication, integrity, confidentiality, and delegation. The most prevalent mechanisms of authentication in a GSI based on grid is the certificate based on authentication (X.509) mechanism where a public key infrastructure (PKI) is assumed to exist which allows the trusted authority to sign information to be used for authentication purposes, by using these mechanisms it is also possible to ensure integrity. In addition to certificate based mechanism, it supports password based authentication, and research efforts are underway to integrate One Time Password (OTP) and Kerberos authentication with GSI.

Confidentiality are supported through transport level security using SSL/TLS protocols, and message level security using Web services standards. It is worth notice that Globus Toolkit is one of the few implementations where message level security is used for grid confidentiality purposes.

Delegation is especially important in case of grid because of the possibility of multiple resources involved in grid based transactions. It may be unnecessary or very expensive to authenticate each and every time a resource is accessed. On the other hand, if the user issues a certificate allowing the resource to act on its behalf then the process will become a lot simpler. This type of certificate issued by the user to be used by some other entity is called a proxy certificate. A proxy is made up of a new certificate containing two parts, a new public and a new private key. The proxy certificate has the owner's identity, with a slight change to show that it is a proxy certificate. The certificate owner will sign the proxy certificate. As part of the proxy certificate there is an entry with a timestamp, which indicates at what time the proxy certificate expires.

3.4 Computational ElectroMagnetics description

A complete discussion about the EM modeling and the mathematical aspects of the formulation goes beyond the scope of this paper; only a short summary of the problem will be presented in the following. In order to simplify the mathematical model, in the following the analysis of the electromagnetic behavior of Perfectly Electric Conducting (PEC) objects in a free space environment will be briefly introduced. Nevertheless, the authors would like to point out that the described approach is not limited to PEC objects in free space, in fact it is applicable to different formulations as well (dielectric objects or layered media problems for instance): in other terms it is a kernel free method. Besides, the focus of this chapter will be on a Grid Computing approach applied to computationally demanding electromagnetic problems: rather than considering more complicate approaches, that would divert the attention from the subject of this chapter, we prefer to introduce and apply the method to PEC objects in a homogeneous background, but we stress that it can be applied to other formulations as well. The Electric Field Integral Equation (EFIE) is a very versatile approach to the full-wave analysis of complex electromagnetic problems: for PEC objects the EFIE can be written by enforcing the boundary condition on the surface of the object, i.e. the

tangential component of the electric field vanishes on the surface S:

$$\hat{n} \times \underline{E}|_{\Sigma} = \hat{n} \times \left(\underline{E}^{scat} + \underline{E}^{inc} \right) |_{\Sigma} = 0 \tag{1}$$

The surface S is discretized by a mesh with triangular cells, over which a usual system of RWG functions $\underline{f}_n$ is defined. The unknown surface current $\underline{J}$ is approximated by the above set of RWG basis functions

$$\underline{J}(\underline{r}) \approx \sum_{n=1}^{N} I_n \underline{f}_n(\underline{r}) \tag{2}$$

A Galerkin testing is used to convert the EFIE into the MoM linear system; hence we obtain the matrix equation

$$[Z] \cdot [I] = \left(\left[Z^{\phi} \right] \left[Z^{A} \right] \right) \cdot [I] = [V] \tag{3}$$

where a generic element of the scalar potential and of the vector potential matrix, $\left[Z^{\phi} \right]$ and $\left[Z^{A} \right]$ respectively, is expressed as

$$Z^{\phi}_{m,n} = \frac{1}{4\pi j \omega \epsilon_0} \iint_{S_m} dS \; \nabla_s \cdot \underline{f}_m(\underline{r}) \iint_{S_n} dS' \; G(\underline{r}, \underline{r}') \; \nabla_s \cdot \underline{f}_n(\underline{r}') \tag{4}$$

$$Z^{A}_{m,n} = \frac{j \omega \mu_0}{4\pi} \iint_{S_m} dS \; \underline{f}_m(\underline{r}) \cdot \iint_{S_n} dS' \; G(\underline{r}, \underline{r}') \; \underline{f}_n(\underline{r}') \tag{5}$$

where: $G(\underline{r}, \underline{r}') = \frac{e^{-jk_0 R}}{R}$, $k_0 = \omega \sqrt{\epsilon_0 \mu_0}$, $R = |\underline{r} - \underline{r}'|$, and S_m is the definition domain of the function $\underline{f}_m$. The coefficients I_n in (2) are collected in the vector $[I]$, and the mth element of $[V]$ is equal to

$$V_m = - \iint_{S_m} dS \; \underline{f}_m(\underline{r}) \cdot \underline{E}^{i}(\underline{r}) \tag{6}$$

where $\underline{E}^{i}$ is the impressed field in absence of bodies.

The first step of the Domain Decomposition approach is a subdivision of the overall geometry, breaking down the scatterer surface S into N_B sub-scatterers, which will be referred as *blocks* in the following. An example of the splitting of a sphere into 8 blocks is shown in Figure 5.

To subdivide the structure in blocks, on which entire domain synthetic functions will be generated, we use a fully automatic procedure. We would like to stress this aspect of our implementation, as it is of vital importance to be able to properly generate the subdomains for the synthetic function approach, and it is especially critical for arbitrary and complex structures. The block generation algorithm is based on the multi-level cell grouping described in Andriulli et al. (2008); Vipiana et al. (2009) for the generation of the Multi-Resolution basis functions. The starting point is the usual mesh for the analysis of the considered structure, without any constraint on the mesh properties and on the topology of the structure. Then a nested family of meshes, with non-simplex cells, is generated through subsequent groupings of the initial triangular mesh cells. We denote this initial mesh with M_0, and call it level-0 mesh. All other meshes will be composed of groups of adjacent cells of the initial mesh. We start by considering groupings of adjacent cells in M_0 formed so that their average area is about four times the average area of the cells in M_0. This covering will be called the level-1 mesh, M_1. The same procedure applied to M_1 will generate the generalized mesh M_2, and so forth. We stop grouping a cell with its adjacent cells when its size reaches a chosen threshold

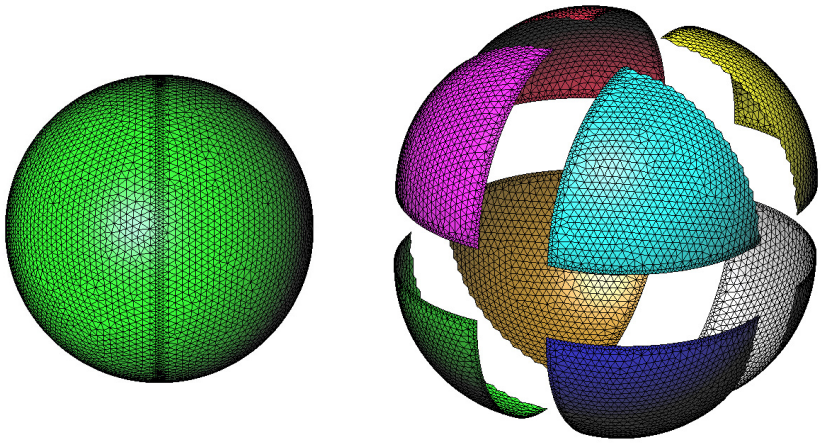

Fig. 5. Example of geometric block partitioning and definitions; a sphere is split into 8 blocks.

Δ; the algorithm stops when the last level L contains non-simplex cells with (linear) dimension around Δ. The non overlapping blocks, on which the synthetic functions will be generated, will then be the cells of the last level M_L. The only parameter the user has to control for the algorithm is the threshold where the grouping has to be stopped. We underline that the grouping procedure used to find the domains where the Synthetic Functions are defined has a $O(NlogN)$ complexity, where N is the number of initial mesh cells.

The next step consists in the generation of the basis functions to model the current distribution over each block; these will be referred as *synthetic functions* (SFs), whose support extends over the entire block. The synthetic functions are chosen in the set of responses to the required incident field, and to other sources placed at the borders of the block and around it, to make up the space of all (rigorous) solutions restricted to that block. Once the set of the solutions to all these sources is computed, the minimum number of necessary responses has to be determined. This is done through a combination of a Singular Value Decomposition (SVD) and a Gram-Schmidt (GS) procedure, applied to the matrix that collects the responses from all the sources: the functions are then selected with a proper thresholding on the associated singular value. Finally, the set of RWG functions defined over the connections of contacting blocks is added to the set of SFs in order to guarantee continuity across blocks. Since the SFs are expressed as a linear combination of the initial RWG functions, the scheme can be seen as a purely multiplicative algebraic compression of the standard MoM matrix.

It should be clear now that the process of generation of the synthetic functions is carried out independently on different blocks, i.e. the set of SFs is generated by solving the electromagnetic problem on each block in *isolation*. As a consequence, a Grid Computing approach is particularly well suited for the generation of the SFs: each block can be processed by a different node of the grid.

Finally, after the complete set of SFs is generated, the original system matrix is compressed, and the compressed system is inverted to yield the solution of the full problem. We underline that the goal of the synthetic functions approach is to accelerate the solution of the problem when a large number of excitation vectors (right hand sides, RHSs) is present, which is typical

for Radar Cross Section (RCS) calculations. On the other hand, when a single RHS is present, approaches based on the combination of a fast factorization scheme and an iterative solver, are more efficient.

In order to exploit the Grid Computing approach, each node of the grid has to sequentially solve a number of blocks. The synthetic functions are chosen in the set of responses to the required incident field, and to other sources placed at the borders of the block and around it, to make up the space of all (rigorous) solutions restricted to that block. Since the grid is in general heterogeneous, i.e., the nodes have different processing and memory characteristics, the blocks should be properly dimensioned and assigned to the computing nodes. When the block dimensions are properly taken into account, the computation of the full MoM matrix and its handling do not pose a limitation within a single block.

However, once the full set of synthetic functions is generated, one needs to apply the basis change to the full system matrix related to the original structure, in order to compress the system and invert it. For practical problems this is not feasible though, due to the total dimension of the problem. Therefore we perform the compression within a fast scheme, which avoids computing the full system matrix and makes the solution of large problems possible. The compressed matrix in the new basis can be written as:

$$[Z_{SF}] = [T]\,[Z]\,[T]^H \tag{7}$$

where $[Z_{SF}]$ is the compressed matrix, $[T]$ is the change of basis matrix, whose dimensions are $N_{SF} \times N_{RWG}$ (the total number of synthetic functions and the total number of RWG functions, respectively), $[Z]$ is the MoM matrix in the RWG basis, and $[]^H$ represents the hermitian operator. The same change of basis is performed on the RHS, namely:

$$[V_{SF}] = [T]\,[V] \tag{8}$$

where $[V_{SF}]$ is the compressed RHS in the SF basis. Finally the compressed system is solved. At the present stage of the work, the compression and the solution of the complete system is not carried out in the grid though; this will be object of future research.

4. GridCEM architecture

The GridCEM project aims to improve performance of CEM application. This objective is achieved by the adoption of a grid architecture that exploits the benefits derived from the parallel computation. The proposed system manages data automatically and without impact for users. Furthermore, virtualized resources make system flexible, simplify the underlying infrastructure and improve scalability. To this aim an hybrid grid infrastructure was built and tests were performed in order to compare grid results to the ones of the same task executed on a single machine. The grid is composed of two types of nodes, a Master Node (MN) and Worker Nodes (WN). The Master Node is in charge to manage, control, and grant the security of the entire infrastructure. In addition, it coordinates other node operations. The middleware used to create the grid infrastructure is the Globus Toolkit (Globus (2010)). The toolkit, as explained in Section 3.1.1, includes functionalities for security, information systems, resource and data management, node communication, fault detection, portability and all the features need to safely deploy grid. The input data is a file that contains the mesh structure to analysis. It is split by the MN in smaller files, called blocks, that are distributed to WN. Since these blocks are generated according to Domain Decomposition technique and are independent, they can executed in parallel. The size of split blocks is not uniform but

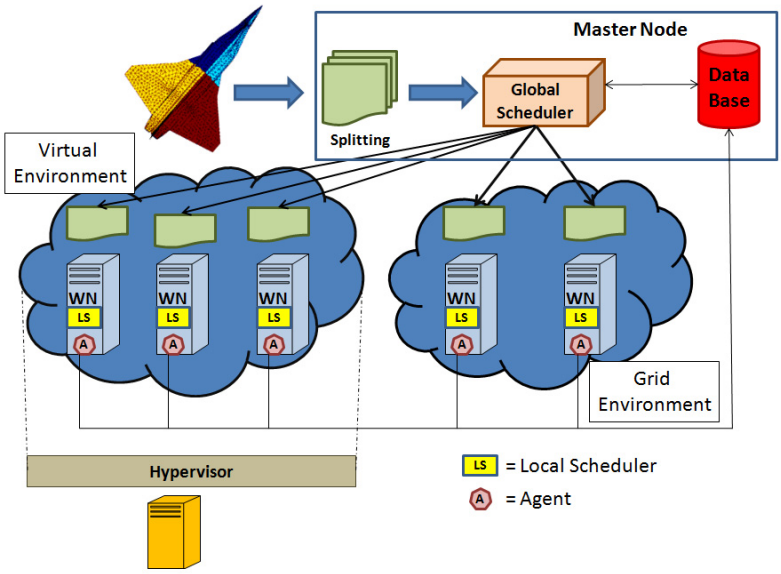

Fig. 6. GridCEM Architecture.

depends on the geometry of the structure. Moreover, the MN hosts the Global Scheduler (GS), that represents the smart component of the grid, and it holds information related to grid status, that are sent from each Worker Node, and stored in a database. The WN, instead, receives blocks, elaborates them and sends results to MN. The execution phase is managed by the Local Scheduler (LS) and the node status is monitored by an always active agent. The agent monitors the availability of each service on the node and sends periodically its status to the database on the MN. The agent role is crucial since the choice to send a job to a node depends on the information it gathers: if all services are available, the node is in condition to receive a job. All Worker Nodes have same operating system and are equipped with the same software: the middleware, the monitoring agents, the Local Scheduler and the code for execution.

4.1 Global Scheduler

As mentioned before the Master Node holds the brain of the grid that is represented by the Global Scheduler. It is a software module developed in Java responsible for the distribution of smaller parts of the input data (blocks) to each Worker Nodes. It communicates with other nodes thanks to grid services provided by the Globus Toolkit. The files are sent by the scheduler in order to balance the execution on each node: this is achieved by checking WNs availability and the number of files to sent. In order to know the overall status of the grid, the GS queries the database and transfers file only to nodes that have communicate their status within a specific time period. It is worth noting that this monitoring system that periodically push data into the database instead of gather information from each machine, allows to reduce time and computational wastes Gradwell (2003).

4.2 Local Scheduler

Each WN is provided with a Local Scheduler, developed in Java, that checks contents of its input folder: every time a new job file is detected it executes the task. In order to be recognized the filename must follow a predefined naming convention rules. During the execution the analyzed block is transformed into a Method of Moments (MOM) matrix. If the execution terminates successfully, the output is sent to the Master Node, that reassembles it with the outputs received from other nodes. Also the LS is in charge of tracking the status of job execution and sends information about start and stop time of each process.

4.3 Virtual environment

Since the code was not developed for parallel execution, it was decided to optimize resources by virtualizing nodes in order to run multiple processes on the same physical machine. In this way it was possible to create multiple virtual nodes on the same resource and increase the available nodes number, e.g., the parallelization of the system, instead of coding parallelization, to improve the overall performance. Virtualized systems also help to improve infrastructure management, allowing the use of virtual node template to create virtual nodes in a short time, speeding up the integration of new nodes on the grid and, therefore, improving the reactivity and the scalability of the infrastructure. Another advantage of virtual environment is the availability improvement, since in case of damage of a virtual node the system will be able to quickly restore it, reducing the downtime due to his recovery. The open source KVM (Hirt (2010)), has been used as hypervisor. It allows to create fully virtualized machines. The kernel component of KVM is included in mainline Linux. The basic requirements for the installation of this hypervisor is that the processor of the machine supports virtualization technology (Intel VT or AMD-V).

5. Test environment

In order to measure the gain in terms of time, some performances test were conducted. It has been compared the result times of the analysis of a model executed on the grid infrastructure with the sequential execution of the same model on a single computer. The experiment allowed verifying the practical usefulness of the adoption of distributed infrastructures in this kind of applications.

Node	Type	CPU model	Virtual CPU	RAM[GB]
master	physical	Intel Core Duo 2.66GHz		4
wn1	physical	Intel Pentium 4 @ 3.20GHz		3.5
wn2	physical	Intel Core 2 6420 @ 2.13GHz		2
wn3	virtualized	Intel Xeon E5440 @ 2.83GHz	2	4
wn4	virtualized	Intel Xeon E5440 @ 2.83GHz	2	4
wn5	virtualized	Intel Xeon X3470 @ 2.93GHz	2	4

Table 1. Grid Nodes Specifications.

The grid used for performance testing consists of hardware not purchased specifically for this purpose, but of computer available to researchers of Istituto Superiore Mario Boella(ISMB). For this reason, its composition is heterogeneous in terms of the machines types (workstations, servers) and in terms of view of hardware resources of each machine (RAM, CPU, HD). Within the pool of available machines, two machines met the criteria for virtualization: on

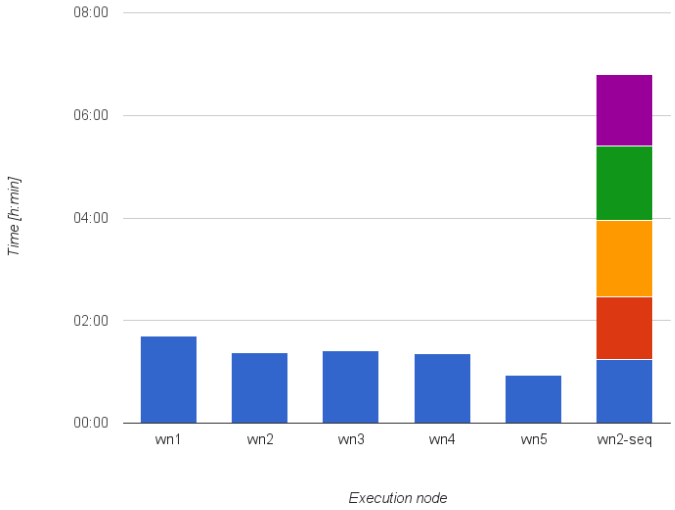

Fig. 7. Total execution time comparison between grid enviroment and sequential execution

these machines three VMs have been created. To run the tests six nodes with the following configuration have been used: a Master Node, two physical Worker Nodes (wn1, wn2) and three virtualized Worker Nodes (wn3, wn4, wn5). The details of the machines used for the experiment are shown in Table 1.

As a test-case a jet fighter aircraft (shown in Figure 1) has been discretized with a linear mesh density around 5 cm. The input file has a size of 17 MB and consists of about 156K unknowns. The plane is illuminated with a wave at the frequency of 600 MHz. The geometry has been subdivided into 67 blocks, each one described in a mesh file of about 7.5 MB. The first test performed was the execution of the entire process on a single physical machine. The node chosen for this test was the wn2: despite the smaller amount of RAM, the processing time of this machine is comparable to the average processing time of the other nodes. This node splits

Node	Executed blocks	Number of executed blocks	Time [h:min:s]
master		0	0:02:55
wn1	5 10 15 20 25 30 35 40 45 50 55 60 65	13	1:42:19
wn2	1 6 11 16 21 26 31 36 41 46 51 56 61 66	14	1:22:33
wn3	2 7 12 17 22 27 32 37 42 47 52 57 62 67	14	1:24:50
wn4	3 8 13 18 23 28 33 38 43 48 53 58 63	13	1:21:38
wn5	4 9 14 19 24 29 34 39 44 49 54 59 64	13	0:56:22
wn2	all	67	6:56:51

Table 2. Nodes Execution Time.

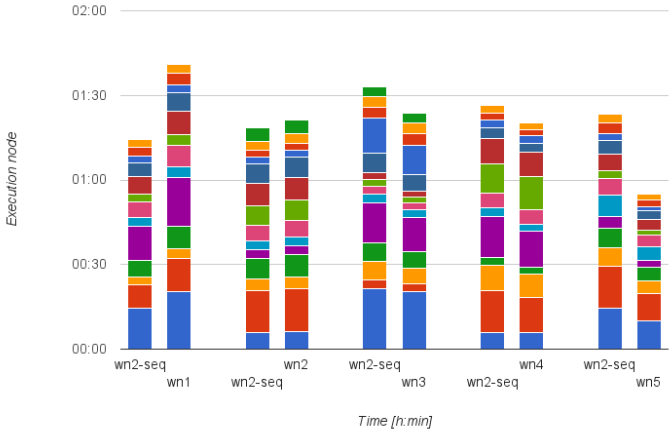

Fig. 8. Block execution time of each node compared to the same block executed in sequential mode (wn2-seq)

the main input into smaller chunks and ran the sequential execution of individual blocks: the total processing time for 67 blocks on wn2 was equal to 6h 56min 51s. In the second test the splitting was delegated to the Master Node, which has also been responsible for the distribution of the single blocks to different nodes. The execution time of the build process of the blocks and the file transfer is 2min 55s, and therefore it is negligible if compared to the total execution time. The execution times of different nodes are very similar, only the virtual machine wn5 has slightly better performance, probably due to the higher performance processor. The total execution time of the grid is equal to the maximum execution time of individual nodes, i.e., 1h 42min 19s of wn1. Table 2 summarizes the execution times. Figure 7 depicts total execution time comparison between grid environment and sequential execution on different nodes, on the grid and on sequential execution. The total time reduction is of 75%. Figure 7 shows the comparison between the processing times of individual nodes on the grid and the sequential execution: the colors of the column wn2-seq, correspond to the processing time of each node, through this comparison it can be appreciate the performance gain of the grid. In Figure 8 the comparison between the execution of individual blocks on the grid and in sequential mode is represented. The columns are divided into small parts that correspond to different executed blocks, each block is identified by a different color. The graph is useful for reasoning on the overhead introduced both by the grid and by the virtual machines. From the comparison between the execution of a group of blocks on the grid on a given node and sequentially on the same machine (i.e., the second pair of columns) it can be deduced that the grid introduces an overhead (equal to 2min 38s in this case) due to the transfer of files output, but negligible compared to the total execution time.

5.1 Application domains overview

More and more applications require high performance computing, flexibility and reduced processing time. The architecture explained before, can be useful in many fields i.e., in

e-science applications (electronic, geophysics, biomedical domains). In our past studies, a similar grid infrastructure has been created for the analysis of Radio Occultation data. This project for the Italian Space Agency (ASI), allows to characterize the temperature, pressure and humidity (further details are available at Terzo et al. (2011)). In this project we have focused the attention on configuration and management problems due to distributed environment, scheduling of processes and resources, virtualization and Cloud environment. Another type of application, where the Grid Computing technique can be useful is in bioinformatic field, i.e., biological laboratories are producing a huge amount of DNA/RNA sequencing data and Next Generation Sequencing has proved to be extremely helpful in making the detection of various forms of disease. Unfortunately, this is reflected in a higher computational effort that must be faced by innovative computing infrastructure but in this case an hybrid infrastructure, composed of physical and virtualized resources and when it is necessary a Public Cloud (i.e., Amazon), is the best choice. An important feature is the application type that must allow to split a process in a smaller independent jobs in order to elaborate each single job in several resources reducing the elaboration time and increase the system flexibility.

6. Conclusion

The solution for CEM problems requires infrastructures based on high performance computing in order to reduce the execution time. However, it is not always possible to use supercomputers or parallelize algorithms code used for the calculation, a good solution may be represented by the Domain Decomposition technique, which allows the subdivision of the original complex problem into a set of smaller subproblems in a grid environment. In fact for this project, a grid infrastructure was carried out, that consists of 6 nodes (both physical and virtual). The results showed that the adoption of this infrastructure has reduced the execution time of 75% with respect to the sequential execution on a single machine. It was also noted that the overhead introduced by the grid is negligible when compared to the total execution time. The Virtualization has allowed optimizing the hardware resources of the machines, letting to run multiple blocks in parallel on the same physical machine, not introducing a significant overhead compared to the overall system performance. Studies are underway to improve the implementation of the scheduler, ensuring that blocks will be distributed to nodes most suitable for their execution. In particular, it will be developed a system that takes into account the weights of the jobs and the weights of the nodes available, assigned on the basis of predetermined criteria (e.g., file size, hardware resources, nodes availability, the average time of execution). The Scheduler will also be able to turn on and off virtual nodes according to the needs of the system, considering the load level of the grid. Furthermore the virtual nodes will be configured dynamically as needed to perform specific jobs (RAM, CPU and storage). A further improvement will be the integration of the system with Public Cloud Platforms (e.g., Amazon EC2) in order to increase the system scalability, asking for virtual nodes usage only when necessary.

7. References

Andriulli, F., Vipiana, F. & Vecchi, G. (2008). *Hierarchical bases for non-hierarchic 3D triangular meshes*, IEEE Trans. on Antennas and Propagation.

Asadzadeh, P., Buyya, R., Kei, C. L., Nayar, D. & Venugopal, S. (2005). *Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies*, available: http://www.buyya.com/papers/gmchapter.pdf.

CAS (2010). *Community Authorization Service*, available:
http://globus.org/toolkit/ docs/4.0/security/cas.
Chakrabarti, A. (2007). *Grid computing security*, Springer.
Chierici, A. & Verald, R. (2010). *A quantitative comparison between xen and kvm*, 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09), IOP Publishing Journal of Physics.
Foster, I. & Kesselman, C. (2003). *The Grid2: Blueprint for a New Computing Infrastructure*, Morgan Kauffman.
Francavilla, M. A., Vasquez, J. A. T., Vipiana, F., Matekovits, L. & Vecchi, G. (2011). *Multi-level cell grouping scheme for an SFX/GIFFT approach for the analysis of electrically large 3D structures*, IEEE International Symposium on Antennas and Propagation.
Globus (2010). *Globus Toolkit*, available: http://www.globus.org.
Gradwell, P. (2003). *Grid scheduling with agents*, Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems.
GSI (2010). *Grid Security Infrastructure*, available:
http://www.globus.org/security/overview.html.
Hirt, T. (2010). *KVM-The Kernel-Based virtual machine*, available: http://www.cs. hs-rm. de/ linn/fachsem0910/hirt/KVM.pdf.
IBM (2010). *Virtual Linux*, available: http://www.ibm.com/developerworks/linux/library /l-linuxvirt/.
Kourpas, E. (2006). *Grid Computing:Past, Present and Future, An Innovation Perspective*, available: http://www.cnw.com.cn/cnw07/download/IBM.pdf.
Matekovits, L., Laza, V. A. & Vecchi, G. (2007). *Analysis of Large Complex Structures With the Synthetic-Functions Approach*, IEEE Trans on Antennas and Propagation.
Mouriño, J. C., Gómez, A., Taboada, J. M., Landesa, L., Bértolo, J. M., Obelleiro, F. & Rodríguez, J. L. (2009). *High scalability multipole method. Solving half billion of unknowns*.
MPI (2011). *Message Passing Interface Forum*, available:
http://www.mpi-forum.org/docs/docs.html.
OGF (2011). *OGF Introduction*, available:
http://www.gridforum.org/About/abt_introduction.php.
OGSA (2007). *Open Grid Services Architecture*, available:
http://searchsoa.techtarget.com/definition/Open-Grid-Services-Architecture.
OpenMP (2011). *The OpenMP, API specification for parallel programming*, available:
http://openmp.org/wp/.
Oracle (2009). *Oracle Grid Computing*, available: http://www.oracle.com/us/technologies/ grid/057123.pdf.
PERMIS (2002). *Privilege and Role Management Infrastructure Standards Validation*, available: www.permis.org.
Terzo, O., Mossucca, L., Cucca, M. & Notarpietro, R. (2011). *Data intensive scientific analysis with Grid Computing*, International Journal of "Applied Mathematics and Computer Science".
Vipiana, F., Andriulli, F. & Vecchi, G. (2009). *Two-tier non-simplex grid hierarchic basis for general 3D meshes*, Waves in Random and Complex Media.
VOMS (2003). *Virtual Organization Membership Service*, available:
http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html.