Combining Grid and Cloud Resources by Use of Middleware for SPMD Application

Brian Amedro, Françoise Baude, Fabrice Huet, Elton Mathias

Combining Grid and Cloud Resources by Use of Middleware for SPMD Applications

Brian Amedro, Françoise Baude, Fabrice Huet and Elton Mathias
INRIA Sophia-Antipolis, CNRS, I3S, UNSA.
2004, Route des Lucioles, BP 93
F-06902 Sophia-Antipolis Cedex, France.
{First.Last}@inria.fr

***Abstract*—Distributed computing environments have evolved from in-house clusters to Grids and now Cloud platforms. We, as others, provide HPC benchmarks results over Amazon EC2 that show a lower performance of Cloud resources compared to private resources. So, it is not yet clear how much of impact Clouds will have in high performance computing (HPC). But hybrid Grid/Cloud computing may offer opportunities to increase overall applications performance, while benefiting from in-house computational resources extending them by Cloud ones only whenever needed. In this paper, we advocate the usage of ProActive, a well established middleware in the grid community, for mixed Grid/Cloud computing, extended with features to address Grid/Cloud issues with little or no effort for application developers. We also introduce a framework, developed in the context of the DiscoGrid project, based upon the ProActive middleware to couple HPC domain-decomposition SPMD applications in heterogeneous multi-domain environments. Performance results coupling Grid and Cloud resources for the execution of such kind of highly communicating and processing intensive applications have shown an overhead of about 15%, which is a non-negligible value, but lower enough to consider using such environments to achieve a better cost-performance trade-off than using exclusively Cloud resources.**

I. Introduction

Cloud computing is being considered a disruptive technology as it is changing the way to manage resource provisioning [1]. The idea of extending owned computational resources with Cloud computing resources, or alternatively to partly outsource some of the additional computing power needs to a Cloud has gained attention.

It is not yet clear, however, how much of impact Clouds will have in HPC [2][3]. In fact, it is very unlikely that Cloud resources will outperform highly optimized hardware (e.g clusters with high performance networks, GPUs and specialized peripherals) in a near future. Besides, most enterprises, research institutes, government agencies, etc. already count with in-house HPC resources, which should be used anyway.

For these reasons, hybrid distributed computing environments, built mixing clusters, computing Grids, and Cloud resources may provide a better cost-performance trade-off or, at least, a temporary solution until a viable (i.e. performant and cost-effective) 100% Cloud usage for HPC be promoted. The usage of such hybrid environment raises issues related to multi-domain infrastructures that pertains to: deployment and dynamic provisioning of resources, multi-protocol and multi-domain communication, and flexible programming models capable to adapt to the execution in different environments. A straightforward solution, inherited from the Grid computing world is to introduce a middleware that address these issues given the targeted applications to be supported, like SPMD ones as considered in this paper.

HPC applications, in particular non embarrassingly parallel legacy SPMD applications, have specific requirements, which are harder to handle in heterogeneous multi-domain environments compared to homogeneous ones: load balancing across heterogeneous resources, topology-aware point-to-point and collective communication, management of heterogeneous network characteristics and so on. A Grid/Cloud middleware that handles basic multi-domain issues (like the one we propose in section III) may not be enough to address all the requirements of non-embarrassingly parallel applications, but it is a good starting point to develop a more advanced and integrated framework (like the one we propose in section IV) allowing coupling of such legacy SPMD applications.

The contributions of this paper are threefold. First, we assess the worthiness of using Amazon EC2 Cloud platform to run HPC applications in different instance configurations, compared to private resources. Second, motivated by benchmark results and the potential benefit of combining private and Cloud resources for HPC, we advocate the usage of the well established ProActive Grid middleware [4] and describe how it had to be extended so to address Grid/Cloud issues. Third, we provide a solution in the form of a framework, allowing to deploy and execute coupled domain decomposition legacy SPMD applications on heterogeneous multi-domain environments. Its distinctive aim is in the coupling of SPMD applications that are moreover legacy. However, it takes benefit of features offered by a lower layer middleware, in our case, the ProActive middleware extended in section III. That's why we advocate that such kind of framework could also be built by leveraging alternate Grid/Cloud low level middlewares. Finally we present experimental performance results and cost-performance analysis of such framework (and, by extension, of ProActive for mixed Grid/Cloud computing) over a hybrid Grid/Cloud multi-domain platform.

The remainder of the paper is organized as follows. Section II presents the evaluation of the performance offered by Amazon EC2 Cloud resources to execute HPC applications in comparison with private resources. Section III presents the ProActive middleware and the main mechanisms included

to support the development and execution of applications in Cloud platforms: resource management and associated deployment, and multi-protocol and multi-domain communication mechanisms. Section IV presents the ProActive-based framework for legacy domain-decomposition applications that integrates ProActive features, performance and cost-performance analysis of the framework on hybrid Grid/Cloud environments. Section V presents related works and our positioning with respect to them. Section VI concludes.

II. Benchmarking Amazon EC2 Against Private Clusters

Cloud platforms are gaining popularity due to their pay-as-you-use and simplified interface. Besides of being one of the main IaaS Cloud infrastructures, Amazon EC2 is one of the best adapted Cloud alternative for HPC for its openness in configuration. In order to assess the worthiness of using Amazon EC2 Cloud as an HPC platform, we have deployed a series of HPC benchmarks, the MPI NAS Parallel Benchmarks. Five different architectures, described in Table I, are compared: a private cluster and four types of Amazon EC2 instances, including the *Cluster Compute Instances* (CCI) which were designed for HPC applications and comes with a detailed hardware architecture, unlike other instances [5].

	Private Cluster	cc1.4xlarge
Proc.	Intel Xeon L5420 2*4 cores @ 2.5 GHz	Intel Xeon X5570 2*4 cores @ 2.93 GHz 33.5 EC2 Units
Arch.	64 bits	64 bits
Memory	16 GB	23 GB
HDD	320 GB	1,690 GB
I/O Perf.	Gigabit Ethernet	Very high (10GbE)

	m1.small	c1.medium	c1.xlarge
EC2 units	1	5 (2*2.5)	20 (8*2.5)
Arch.	32 bits	32 bits	64 bits
Memory	1.7 GB	1.7 GB	7 GB
HDD	160 GB	350 GB	1,690 GB
I/O Perf.	Moderate	Moderate	High

TABLE I
Benchmarked Resources

Figure 1 shows a comparison of I/O performances for the given distributed architectures. Our private cluster showed standard results for a Gigabit Ethernet network (900MB/sec for throughput and 55μsec for latency). Regarding Cloud instances, Amazon defines the I/O performances of its instances as "Moderate" (for Small and Medium instances), "High" (for XLarge instances) and "Very High" (for Cluster Compute instances) and both throughput and latency reflect this classification. However, the "Very High" performance network, which is a 10 Gigabit Ethernet, provides a higher latency (80μsec) than the Gigabit Ethernet network of our private cluster.

Figures 2, 3 and 4 show the performance (mflops) of three of the NAS Parallel Benchmarks on each architecture, varying the number of processes, up to 1024 with the *Compute Cluster Instances*. Results average 10 runs. We used Intel Fortran 10.1 for compilation and OpenMPI 1.4.2 for distribution.

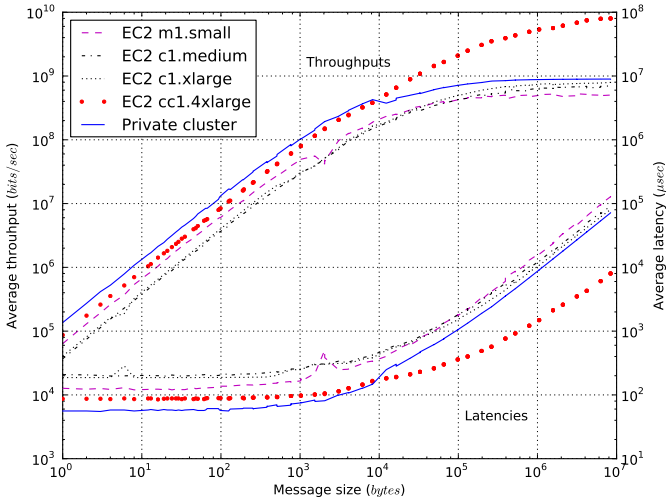

Fig. 1. I/O performance comparison between our private cluster and EC2 instances

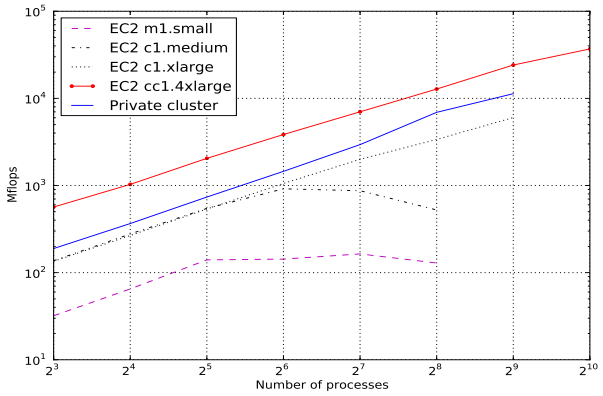

Fig. 2. Kernel EP class C

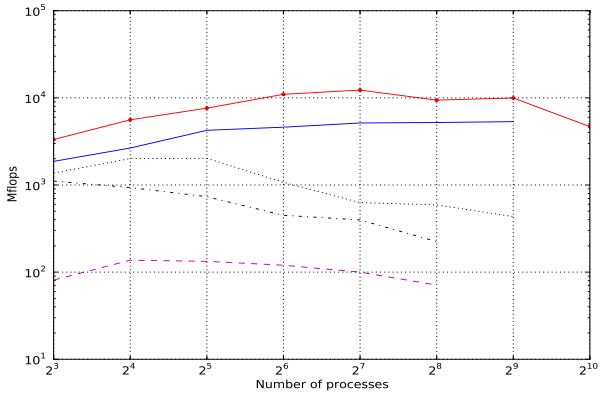

Fig. 3. Kernel CG class C

Kernel EP (Fig.2) is an embarrassingly parallel problem which involves almost no communication between processes. It is a strong test for pure computational speed. This test clearly shows the speed difference between all the architectures.

Kernel CG (Fig.3) computes a conjugate gradient involving

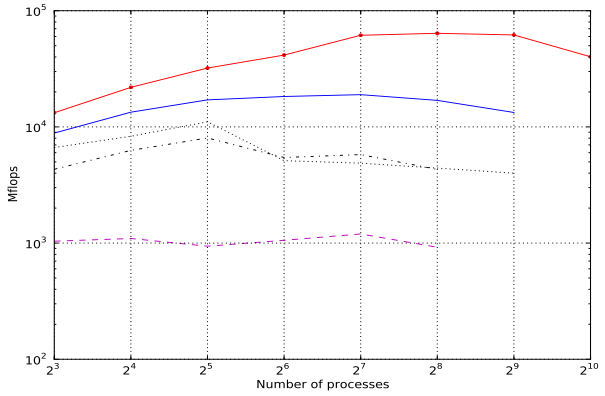

Fig. 4. Kernel MG class C

a large number of small messages, and is a strong test for communication performance. It confirms the results shown in Fig.1. The Amazon EC2 instances performances are, in fact, below what we get with our private cluster, except with the *CCIs*.

Kernel MG (Fig.4) is a multi-grid problem. It is a test for both, computation and communication speed involving large data transfers. With such a problem, the gap between our private architecture with standard Amazon instances narrows, while the 10 Gigabit Ethernet network provided by the *CCIs* takes advantage of its higher throughput.

As shown by the previous experiments[3], standard EC2 does not offer good performance for communication intensive applications, compared to a local cluster. However, recent benchmarks using the newly introduced Amazon EC2 Cluster Compute Instances (CCI) indicate that their performance is comparable with private resources and also exhibit a much smaller instances deployment time. Results showed, however, that communication latencies between CCIs are still higher than within private clusters (70-80 μsec for a 10Gb Ethernet network Amazon EC2 cluster instance, against 50-55 μsec of latency on a private 1Gb Ethernet cluster). This can be annoying with some tightly-coupled processes, but the higher throughput can lead to better program performance, depending on the application.

Also, renting costs associated with execution in Amazon EC2 could be lessen by integrating private resources to the computation. Based on this assumption, when dealing with applications requiring a large amount of computational resources, it might be interesting to have a part on a cluster or Grid and another on a Cloud, given the application characteristics and the possibility to decompose the application in separate parts.

If, on one side, the usage of multi-domain environments may improve cost-performance trade-off, on the other side, hybrid multi-domain platforms make deployment and execution of applications more complex. In next section, we present different mechanisms introduced in the ProActive middleware that may allow the execution of applications over such hybrid environments.

III. ProActive: Towards Hybrid Grid/Cloud Computing

Complex multi-domain platforms raise aspects that are hard to handle at the application level: firewalls, NAT-based networks, heterogeneity of network protocols and resources performance (computing nodes and network). The introduction of a middleware to handle these aspects seems to be the natural approach, inherited from the Grid computing world.

ProActive [4] is a Java middleware which aims to achieve seamless programming for concurrent, parallel and distributed computing. It offers both an uniform active object programming model in which objects are remotely accessible via asynchronous method invocation and an implementation of the GCM[1]. Along with programming models, ProActive offers features which also makes it a middleware. Next subsections present some of the main features that we contributed to add/extend to ProActive, which may help users to develop and deploy mixed Grid/Cloud applications: the ProActive/GCM Deployment framework, the support to multi-protocol and multi-domain communication, and the ProActive Resource Manager.

A. ProActive/GCM Deployment

The ProActive middleware provides an abstract descriptor-based deployment model and framework [6], giving users the capability to deploy applications on different platforms without changing the source code. ProActive/GCM Deployment allows allocation of resources, creation of remote processes and input/output data handling. The definition of the deployment can also encompass security, tunneling of communications, fault tolerance and support of portable file transfer operations. The deployment process is defined by means of two XML descriptors:

- The GCM Application Descriptor (GCMA). A GCMA descriptor defines applications-related properties, such as localization of libraries, file transfer, application parameters and non-functional services requirements and configurations(logging, security, checkpoint and fault tolerance). Once the deployment is done, GCMA descriptors expose the resulting physical environment as a logical network of virtual nodes (VNs) which are used by applications as an abstract representation of computing nodes. The GCMA also defines one or multiple resource providers, described by GCMD descriptors;
- The GCM Deployment Descriptor (GCMD). A GCMD descriptor defines and configures access protocols to reach the resources (e.g. SSH, RSH, GSISSH, etc.), acquisition protocols and tools which are required to acquire resources (e.g. Amazon EC2, PBS, LSF, Sun Grid Engine, OAR, etc.), creation protocols which pertain to "how to launch processes" i.e. ProActive Runtimes (e.g. SSH, OAR, Globus) on these resources, and communication protocols for inter-runtime communication(e.g. RMI, RMISSH and HTTP).

Thanks to the VN abstraction and separation between GCMA and GCMD, the ProActive deployment process is

[1]Grid Component Model

independent of application code. By simply changing resource providers (i.e. add or replacing GCMDs in GCMA files) the same application can be deployed in a different set of resources.

1) ProActive/Amazon EC2 Deployment: In order to ease Amazon EC2 deployment, the ProActive/GCM deployment framework was extended with the support to Amazon EC2 deployment. Users willing to deploy ProActive applications over Amazon EC2 resources need an Amazon Web Services account, the associated credentials and a pre-configured Amazon Machine Image (AMI). A couple of public images are provided by the ProActive development team to simplify this task. Once these requirements are fulfilled, the user can then configure the deployment process through a GCMD descriptor, defining the AMI name, the instance type, specifying the type of Amazon EC2 virtual instance configuration and the maximum number of instances that can be deployed simultaneously. In addition to this configuration expressed in the GCMD descriptor, the user has also to configure how these Cloud hosted resources will be able to interact with non Cloud hosted ones in a way that handles firewalls and NAT issues. The sub-sections III-B and III-C present details about these new ProActive built-in mechanisms for inter ProActive runtimes communications.

2) Resource Manager: The basic approach for ProActive applications deployment is to programmatically load GCMA and GCMD files to first deploy ProActive runtimes (i.e. Virtual Nodes) and then deploy and launch applications over these VNs. In cases where users do not want to take care of configuration of the whole deployment process in their application code and associated deployment configuration, a more high-level deployment approach is offered by ProActive.

In fact, users can rely upon the ProActive Resource Manager. The ProActive Resource Manager is a full-featured resource manager, itself programmed as a ProActive application, which retrieves and releases computing nodes on demand from any kind of resource provider supported by the ProActive deployment framework. In this sense, it completely hides to users the complexity of using the underlying distributed platform resources.

The *Resource Manager Core* is responsible for handling all requests from clients and delegating them to other components. Once the request of getting new nodes for computation, expressed using a resource manager specific API, is received, the core redirects it to a *selection manager*, which finds appropriate nodes in a pool of available nodes based on criteria provided by the client application (e.g. a specific CPU architecture). The pool of nodes is formed by one or several *node aggregators*. Each aggregator (also named node source) is in charge of nodes acquisition, deployment and monitoring from a dedicated infrastructure in line with this infrastructure usage policies. As an example, by using the Amazon EC2 deployment extension (presented in section III-A1), management of Amazon EC2 instances is achieved: this includes the on-demand creation of instances and deployment of applications over these instances. Aggregators rely on some predefined GCMDs descriptors, defined by the administrator in charge of the ProActive resource manager, and not by end-users that just wish to run an application on the infrastructure.

ProActive/GCM deployment framework has already been proved to be useful for deploying and then launch ProActive-based applications in Cloud platforms. In particular, [7] describes ProActive-based applications which are ProActive active object based implementations of the NAS benchmarks we discussed in section II. Besides, performances of the NAS benchmarks programmed along the ProActive active object model are close and, in some cases, better compared to those obtained by the corresponding Fortran-MPI version we evaluated in II.

In this paper, we address mechanisms to allow seamless multi-domain Grid/Cloud computing. Deployment plays an important role in this context, but communication mechanisms we present in next subsections are equally important.

B. Multi-protocol ProActive Communication

As already mentioned in the section III-A, ProActive deployment includes the definition of the point-to-point communication protocols, which are to be used to transport messages among application processes through their corresponding hosting ProActive runtime. Currently, the supported communication protocols are: Java RMI, HTTP, SSH tunneling for RMI (also called RMISSH), HTTP and SOAP.

The basic principle of the ProActive communication mechanism is that ProActive Runtime references hold the information about the supported communication protocol. This makes possible for the ProActive middleware to be agnostic in relation to the communication protocols, i.e. it can seamlessly perform communications between every pair of ProActive nodes, despite of the need of relying on different communication protocols and message forwarding. The multi-protocol communication, in this context, also means multi-hop communications which might depend upon multiple protocols to apply from the origin to the destination, as explained in more details in the next subsection.

C. Multi-domain Communication: Message Forwarding and Tunneling

Connecting computing resources gained from more than one administrative domain raises connectivity problems: resources are generally not directly accessible and do not possess a public IP address. Main connectivity problems include firewalls, NAT-based addressing and multihoming.

ProActive middleware includes a built-in communication protocol called ProActive Message Routing (PAMR), that enables message routing and forwarding. This communication protocol is very versatile, but requires a single communication routing node accessible by all nodes of the application. Besides of network restrictions, having a single forwarding node would certainly generate bottlenecks in highly-communicating applications composed by a large number of nodes.

For this reason, we decided to introduce a lightweight solution based on SSH tunneling and forwarding to address network restrictions like firewalls and NAT. This solution provides a seamless integration of forwarding and tunneling,

and it is managed at the application level (i.e. no need of privileged user account to configure routing at the OS and network levels).

Figure 5 shows a scenario where a single application runs over a set of nodes distributed in Amazon EC2 Cloud and Grid5000[2]. In this scenario, all the nodes located in Amazon EC2 Cloud offer inbound and outbound communication, but nodes located on Grid5000 are isolated from the external network. However ProActive extension for multi-domain communication enables the usage of these resources as if every node would be accessible by every other node by forwarding incoming and outgoing messages through the use of the adequately configured Grid5000 gateway.

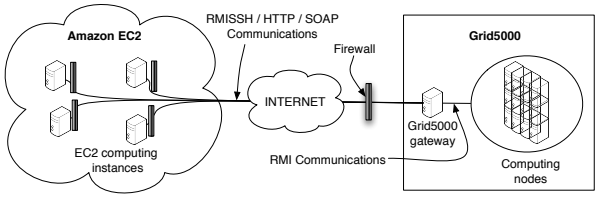

Fig. 5. Tunneling and forwarding communications on an heterogeneous Cloud-Grid environment

In a more protected environment, nodes might be also isolated in both sides. The tunneling/forwarding mechanism can be configured to act as a double forwarding process to handle such a situation, only requiring the modification of the configuration file associated to ProActive Runtimes configuration files.

In any case, an application remains unchanged and its execution in different distributed resources settings only requires the modification of the configuration file associated to each ProActive Runtime to be deployed on the specific node acting as entry and outgoing point of each involved domain. Moreover, a ProActive user has the capability to himself populate those files with such user/application level specific needs.

IV. DiscoGrid: Coupling Legacy SPMD Applications in Grid/Cloud Platforms

Features presented in previous sections solve most of the issues related to high performance Grid/Cloud computing in a separated manner. By integrating these features in Java ProActive applications, users are capable of developing distributed applications for multi-domain (e.g. Grid/Cloud) platforms. However, depending on applications characteristics, other issues that may impact applications performance must be treated like heterogeneity of resources, heterogeneous network performance (latency and lower bandwidth).

Keeping performance of highly communicating legacy SPMD applications in multi-domain environments is a challenging task. Besides the solution of basic deployment and communication issues, these applications are very sensitive to heterogeneity in resources (i.e. performance of machines and network). Even if it would be possible to treat these issues directly in application business code, this would increase the complexity of applications and the solution to these issues would be hardly reusable.

In this section, we present a GCM component-based framework developed in the context of the DiscoGrid project. This framework put together most of the features presented in section III and support coupling of domain-decomposition legacy Fortran C/C++ applications in heterogeneous multi-domain platforms through a new paradigm called hierarchical SPMD (HSPMD).

A. General Approach

Heterogeneity in network and resources is a challenging issue for domain decomposition based scientific applications. The main reason comes from the fact that these applications rely upon a bulk synchronous iterative approach, where applications loop at the pace of the slowest process. The hierarchical network topology and computing power heterogeneity must, therefore, be considered in the mesh partitioning and communication process.

The traditional way of designing domain decomposition based applications is to adopt an SPMD technique combining mesh partitioning and the message passing programming model. This approach was extended in the context of the DiscoGrid project with the hierarchical SPMD (HSPMD) paradigm and a topology-aware partitioning mechanism. HSPMD [8] is an evolution of the traditional flat SPMD parallel programming. It consists in assigning hierarchical identifiers to processes and treating collective communications in a topology-aware manner. The implementation we propose consists in intuitive extensions of the MPI API.

Partitioning is also important to improve load balancing and the communication process. The multi-level partitioner is capable of taking into account resources characteristics (CPU power, amount of memory) and topology to partition a global mesh so that each process presents an equivalent processing time, yet minimizing the amount of communication through slower links [8].

B. Automated Multi-domain Deployment

Deployment using ProActive/GCM in multi-domain environments requires a knowledge of resources configuration and the ProActive/GCM deployment framework. The framework we propose includes an automated deployment mechanism (based on the ProActive Deployment presented in section III-A), which simplifies the deployment task.

The automated deployment mechanism consists of a set of scripts that allows users to reserve resources from multiple domains, generate automatically ProActive deployment descriptors and then use the acquired resources. The configuration of deployment also encompasses the configuration of multi-protocol communication, forwarding and communication tunneling, once acquired resources are known.

C. Hierarchically Organized Applications and Communication

The GCM/ProActive-based framework that supports the DiscoGrid framework is a modular infrastructure composed

[2]Grid5000 is a national French Grid of 5000 cores, in 9 sites

accordingly to the resources hierarchy. It gives the applications a view of a unique global computing infrastructure, despite of the localization and access restrictions of resources. In practice, a DiscoGrid application is composed by independent MPI applications, each of them running in a different administrative domain (as shown in Figure 6).

The framework we propose connects independent applications and offers communication channels to processes of independent applications. Multi-domain resources present at least two logical levels: in the first level, the applications running on physical (or virtual in case of Clouds) nodes , and in the second level, components that route messages between the different domains. The framework also offer the possibility of including more levels, which would allow the representation (and connection) of more complex multi-domain environments, e.g. connecting private resources organized in multiple sites and an already configured multi-domain composed by Grid and Cloud resources.

Figure 6 shows an example of a two-level component-based overlay created by the DiscoGrid framework leveraging to Cloud and Grid resources. On the left, we have a standalone MPI application running on a Cloud (e.g. a set of Amazon EC2 instances) and on the right another standalone MPI application running over a multi-cluster based Grid (e.g. the Grid5000). Each of the MPI processes is wrapped by a *GCM wrapper component* which is connected to the encapsulating component named *router component*, which represents the next level up in the infrastructure. Due to the hierarchical composition and the routing interfaces associated to higher levels, all the nodes are logically connected, even if indirectly, to every other in the multi-domain and, consequently, independent MPI executions are coupled to form a single parallel application along the HSPMD concept.

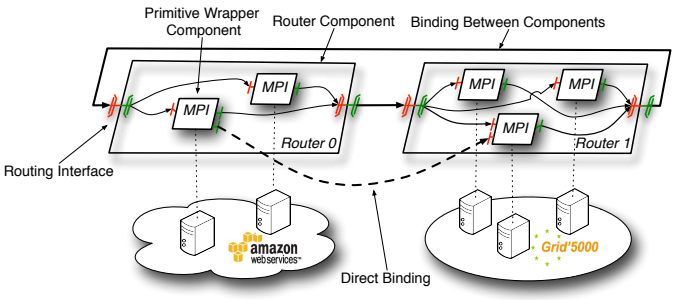

Fig. 6. Typical GCM/ProActive-Based Multi-domain Runtime Support for HPC

Collective communications, as required by the HSPMD API, take profit of the topology to be staged and parallelized. Whenever possible (e.g. no firewalls to cross) for optimizations purposes, the GCM component model allows us to create on-demand direct bindings to perform point-to-point communications, thus bypassing the component hierarchy, e.g. dashed line on Figure 6 directly connects processes of the different domains.

D. Performance Evaluation

In this section, we compare the performance of the component-based framework supporting the HSPMD model in three scenarios: a multi-cluster Grid (Grid5000), the Amazon EC2 Cloud and a multi-domain environment which integrates resources from both platforms. The comparison is based on a non-trivial simulation of electromagnetic wave propagation in three-space dimensions.

Table II presents resources used in these experiments.

	Grid5000 Cluster (Grelon)	
Processors	2 Intel Xeon 5110 (1.6GHz/64b)	
Memory	2 GB	
I/O Perfor.	Gigabit Ethernet	
	Small	**High-CPU XLarge**
EC2 Units	1 /32-bits	20 (8*2.5) / 64b
Memory	1.7 GB	7 GB
I/O Perfor.	Moderate	High

TABLE II
GRID/CLOUD RESOURCES

This simulation is based on a finite element method working on arbitrarily unstructured tetrahedral meshes for solving a system of Maxwell equations. From the computational point of view, the execution is characterized by two types of operations: purely local operations on tetrahedra for computing integral values and a calculation involving neighbor subdomains which involves a *gather-compute-scatter* sequence. Formulations are described in more details in [9].

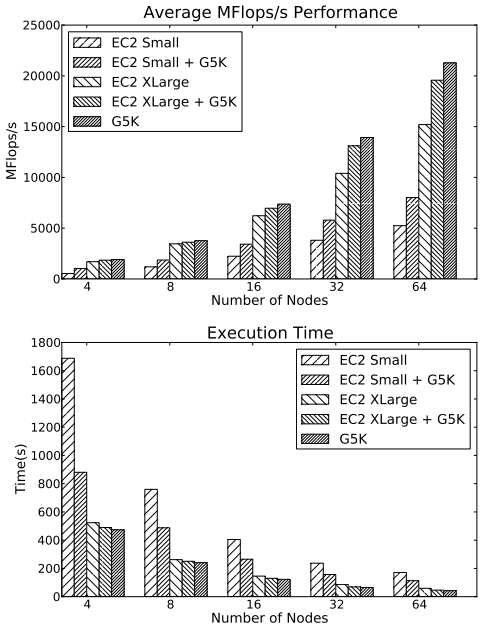

Fig. 7. Performance over Grid5000, Amazon EC2 and resource mix

Figure 7 presents the overall execution times and MFlops/s obtained in the different scenarios. The application being network and CPU intensive, both CPU and network affect

the overall performance. In average, exclusive use of small Amazon EC2 instances presents a performance four times smaller than the one using standard cluster of Grid5000. While Extra Large instances present a CPU performance close to the performance of Grid5000 machines and a slower network interconnection, the resulting application performance using only this kind of node is comparable to the pure Grid5000' one. A mix of Grid5000 resources and Small Amazon EC2 does not perform well, compared to single-site execution over Grid5000, because of the heterogeneous network and load unbalances, even with the usage of the load-balancing partitioner. Adding to Grid5000 resources some Extra Large EC2 instances is the most effective configuration because presenting, in average, only 15% of overhead for such inter-domain execution compared to the average of the best single domain ones. This overhead is mainly due to high-latency communication and message tunneling.

From a cost-performance tradeoff point of view, the usage of Small EC2 instances provides a better MFlops/s per dollar spent ratio, but the overall execution time is much bigger than the one obtained using Extra Large instances for instance which might not be acceptable for certain applications. Previous performance evaluations of Amazon EC2 [3] showed that MFlops/sec obtained per dollar spent decrease exponentially with increasing the number of computing cores and correspondingly, the cost for solving a linear system increases exponentially with the problem size. As summarized by Figure 8, our results point in the same direction. Figure 8 also shows that regarding our benchmarked application, when mixing resources, costs associated to Cloud usage lessen, compared to a situation using only Cloud nodes. This general remark is obvious at first glance because in-house resources are assumed to be available for free for an end-user[3], however this can root some finer analysis of the mixture configurations and so justify mixture decisions. For example, average node performance per dollar spent is increased due to the good performance level of the nodes subset acquired from Grid5000 despite of the associated overhead due to inter-domain communications. For such kind of reasons, we believe the mixture of resources to be relevant in practice, since a trade-off between performance and cost can be reached, considering budget, performance requirements and available amount of in-house resources.

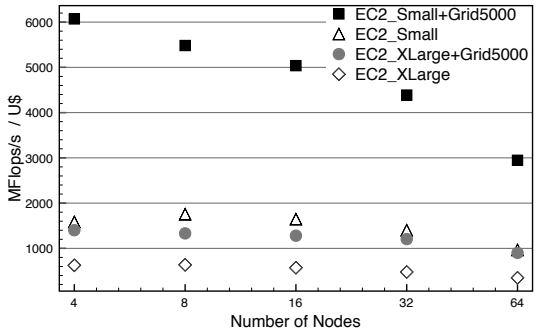

Fig. 8. MFlop/s/U$ over Grid5000, EC2 and Mix

V. Related work

A large number of related works in Grid/Cloud computing exists, depending on the issue that is addressed. In next paragraphs we present a summary of some of them, classified in three main categories: multi-domain communication solutions, middlewares for hybrid Grid/Cloud computing and Grid/Cloud programming frameworks.

a) Multi-domain communication solutions: The main existing solutions to handle multi-domain communications are either based on Virtual Private Networks (VPNs) or overlay networks. On one side, VPNs have been one of the main solutions to connect resources in multiple administrative domains as they are easy to use, giving the impression that resources pertain to a single local network and secure the communications (i.e. providing authentication and messages encryption), e.g. Amazon VPC [10] allows the extension of private resources with Amazon EC2 ones. On the other side, overlay networks provide a communication support that can be customized with resources usage policies and deployed at application level, e.g. SmartSockets [11] offers an efficient Java socket library, capable to automatically discover connectivity problems and solve them through different connection setup mechanisms. Our solution to handle multi-domain communications is an overlay network that is deployed at the application level and it can be easily adapted to complex heterogeneous resources configuration.

b) Middleware for Hybrid Grid/Cloud Computing: A large number of middlewares for hybrid Grid/Cloud Computing propose solutions to manage Grid/Cloud resources, e.g. OpenNebula Toolkit [12] is an open source virtual infrastructure engine that allows a dynamic deployment and reallocation of virtual machines by leveraging existing virtualization platforms. RightScale [13] offers a set of tools that ease the deployment, management and monitoring of Cloud instances despite of the underlying Cloud providers. Nimbus [14], originated from the Globus Grid middleware, includes a set of open source tools that together provide an *Infrastructure-as-a-Service* (IaaS) Cloud computing solution to turn private resources into a Cloud. CometCloud [15] implements autonomic computing engine based in workflow scheduling over Grid and Cloud environments. ProActive integrates an abstract deployment model which can be completely customized by users and integrated on applications. The ProActive Resource Manager also can be integrated on the applications to manage dynamically resources not only from Clouds but also from any private resources. By using tunneling and forwarding mechanisms, the Resource Manager can also be easily used in multi-domain environments. To our knowledge this integration of application development and resource management is not yet available in other Cloud resource managers.

c) Hybrid Grid/Cloud Programming Frameworks: Some middlewares for distributed computing also provide programming libraries with an associated programming model, e.g. Grid-aware implementations of the MPI standard: MPICH-

[3]machines buying, their housing costs plus system administrator(s) salaries can not be avoided but are not charged to end-users in general

G2 [16] and GridMPI [17] offer mechanisms to deploy MPI applications in multi-domain environments with optimizations in collective communications but they are exclusively dedicated to MPI applications, and require connectivity among resources. GridGain middleware [18] allows the deployment of applications on both public or private Clouds, applications developed along a map/reduce programming model, through deamons installed on Grid/Cloud nodes. ProActive approach for multi-domain computing consists in offering solutions to such computing platforms main issues, and allows the integration of these solution into applications, without the need of software installation or a platform running beforehand. The DiscoGrid framework goes further than basic solution of multi-domain issues, by proposing a new paradigm based in SPMD and supporting the coupling of legacy applications.

VI. Conclusion

In this paper, we have advocated two main ideas: that hybrid cluster/Grid/Cloud platforms may provide a better cost-performance trade-off than pure Cloud platforms and that the Grid middlewares improved with adequate extensions can provide basic elements to HPC applications willing to use these multi-domain hybrid platforms.

HPC benchmarks performed in the Amazon EC2 Cloud showed that, effectively, performance of Cloud resources is generally lower than the performance private resources can offer. This fact, added to the fact that HPC users already have private resources, motivated the development of extensions in the ProActive middleware to support the deployment and efficient execution of applications in Cloud platforms and in hybrid environments composed by private clusters, Grid nodes and public Clouds. Main mechanisms integrated into ProActive include extensions to the ProActive abstract deployment model to support the deployment of EC2 instances, the support to multi-protocol communication in multi-domain environments, including tunneling and forwarding techniques to handle firewalls and NAT traversal.

While we believe these mechanisms should be enough to develop basic multi-domain ProActive applications, we also believe that these simple solutions to multi-domain issues may not be enough to run more complex tightly-coupled applications, like legacy SPMD applications. Therefore, we introduced a framework, developed in the context of the DiscoGrid project which offers support to automated deployment, point-to-point and collective communication in multi-domain platforms

The implementation of such versatile and complex framework has shown that the mechanisms introduced in ProActive are powerful enough to build a modular Grid/Cloud framework to support complex scientific domain-decomposition applications. Since no modifications at application codes are necessary to port applications between different platforms, these mechanisms can also be considered to smoothly allow to migrate, even partly, applications from clusters and Grids to Clouds.

Performance results obtained with the DiscoGrid framework (and, by extension, of ProActive for mixed Grid/Cloud computing) showed that, although the Cloud performance is slightly lower than the dedicated cluster for computational intensive codes, this performance can be improved by the addition of private resources. The empirical overhead of using a hybrid platform for the execution of a specific non-trivial HPC application was around 15%, which is a non-negligible value, but lower enough to consider using such environments in production and obtain more processing power, yet reducing costs compared to the pure usage of Cloud platforms.

Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed (see https://www.grid5000.fr). The authors would also thank Amazon Web Services and the PacaGrid CPER for providing computing resources.

References

[1] S. Jha, A. Merzky, and G. Fox, "Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 8, pp. 1087–1108, 2009.

[2] I. Foster, "What's faster–a supercomputer or ec2?" http://ianfoster.typepad.com/blog/2009/08/whats-fastera-supercomputer-or-ec2.html, August 2009.

[3] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *UCHPC-MAW '09: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. New York, NY, USA: ACM, 2009, pp. 17–20.

[4] "ProActive Parallel Suite," http://proactive.inria.fr.

[5] A. AWS, "Amazon ec2 instance types," http://aws.amazon.com/ec2/instance-types/, January 2010.

[6] F. Baude, D. Caromel, L. Mestre, F. Huet, and J. Vayssière, "Interactive and descriptor-based deployment of object-oriented grid applications," *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, 2002.

[7] B. Amedro, D. Caromel, F. Huet, V. Bodnartchouk, C. Delbé, and G. L. Taboada, "HPC in Java: Experiences in Implementing the NAS Parallel Benchmarks," *APPLIED INFORMATICS AND COMMUNICATIONS*, 08 2010. [Online]. Available: http://hal.inria.fr/inria-00504630/en/

[8] E. Mathias, V. Cavé, S. Lanteri, and F. Baude, "Grid-enabling spmd applications through hierarchical partitioning and a component-based runtime," in *Euro-Par '09: Proc. of the 15th Int. Euro-Par Conference on Parallel Processing*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 691–703.

[9] M. Bernacki, S. Lanteri, and S. Piperno, "Time-domain parallel simulation of heterogeneous wave propagation on unstructured grids using explicit non-diffusive, discontinuous Galerkin methods," *J. Comp. Acoustics*, vol. 14, no. 1, pp. 57–81, 2006.

[10] A. VPC, "Amazon virtual private cloud," http://aws.amazon.com/vpc/, January 2010.

[11] J. Maassen and H. E. Bal, "Smartsockets: solving the connectivity problems in grid computing," in *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*. ACM, 2007, pp. 1–10.

[12] "Opennebula project." [Online]. Available: http://www.opennebula.org/

[13] "Rightscale." [Online]. Available: http://www.rightscale.com

[14] "Nimbus toolkit." [Online]. Available: http://workspace.globus.org/

[15] H. Kim, Y. el Khamra, S. Jha, and M. Parashar, ""an autonomic approach to integrated hpc grid and cloud usage"," in *e-Science*, 2009.

[16] N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 63, no. 5, 2003.

[17] H. Saito, K. Hironaka, and K. Taura, "A scalable high-performance communication library for wide-area environments," in *GRID '08: Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 310–315.

[18] GridGain, "Grid grain: The open cloud platform." [Online]. Available: http://www.gridgain.com/