2011 Third IEEE International Conference on Coud Computing Technology and Science

Combining Grid and Cloud Resources for Hybrid Scientific Computing Executions

Amanda Calatrava, Germán Moltó, Vicente Hernández

Instituto de Instrumentación para Imagen Molecular (I3M).
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT
Camino de Vera s/n, 46022 Valencia, España
Email: amcaar@ei.upv.es, {gmolto,vhernand}@dsic.upv.es

***Abstract*—The advent of Cloud computing has paved the way to envision hybrid computational infrastructures based on powerful Grid resources combined with dynamic and elastic on-demand virtual infrastructures on top of Cloud deployments. However, the combination of Grid and Cloud resources for executing computationally intensive scientific applications introduces new challenges and opportunities in areas such as resource provisioning and management, meta-scheduling and elasticity. This paper describes different approaches to integrate the usage of Grid and Cloud-based resources for the execution of High Throughput Computing scientific applications. A reference architecture is proposed and the the opportunities and challenges of such hybrid computational scenarios are addressed. Finally, a prototype implementation is described and a case study that involves a protein design application is employed to outsource job executions to the Cloud when Grid resources become exhausted.**

***Keywords*-Cloud computing; Grid computing;**

I. Introduction

Grid computing [1] has enabled the deployment of large scale distributed computational infrastructures among research institutions, such as the WLCG (Worldwide LHC Computing Grid). This has leveraged collaborations across institutions in the shape of Virtual Organizations (VOs), which have faced research challenges beyond what was possible before these infrastructures existed.

Grid infrastructures deliver the computational power required for resource-starved scientific applications. However, Grid technologies have also revealed some drawbacks in its current implementation approaches. Currently, the access to the large scale Grid infrastructures requires approval from scientific committees which evaluate the impact and scientific merit of the applications requesting access to the Grid. This procedure discourages scientists who might require immediate access to a large pool of computational resources on a specific time frame (because of a conference deadline, for example).

In addition, Grid resources are configured by the remote site administrators and, therefore, the scientific jobs must be previously adapted in order to satisfy the appropriate hardware and software environment (i.e, Operating System, required libraries, CPU architecture, etc.) to guarantee the successful execution of the application on the remote resource. This has been a major drawback of Grids, since scientists need to have substantial knowledge of the remote execution environments and Grid middleware to properly adapt the applications to these environments. Frequently, they are reluctant to adapt their applications to the underlying computational infrastructure.

The advances in virtualization technologies and hypervisors have opened new avenues to pre-package scientific applications into Virtual Machines (VM) which encapsulate the required hardware and software configuration for their execution (into the so called Virtual Appliances), thus decoupling the physical hardware from the application execution, a dependence that exists when using the Grid. The development of Virtual Machine Management (VMM) technologies has fostered the realization of elastic computational infrastructures in the shape of a pool of VMs that can grow and shrink as per-user demand and that execute on top of a physical computing infrastructure. This is the basis of Cloud computing which, according to the NIST definition, is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or provider interaction [2].

This paper advocates for the combined usage of Grid and Cloud resources to execute computationally intensive scientific applications. By leveraging the combined benefits of Grid platforms (high performance, large pool of resources, etc.) and Cloud platforms (elasticity, on-demand access, customizability, etc.) an increased computational performance can be obtained. For that, it is important to envision the different scenarios on which both technologies can coexist. This paper describes three different scenarios on which the Cloud can be used alongside the Grid to increase computational efficiency for High Throughput Computing (HTC) scientific applications. It focuses on the features to be considered when deploying a virtualized infrastructure and meta-scheduling the jobs on a hybrid Grid/Cloud infrastructure. It then introduces a modular architecture and describes the technologies employed.

The remainder of the paper is structured as follows.

978-0-7695-4622-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CloudCom.2011.73

First, section II provides an overview of related work in hybrid usage of Grid and Cloud resources. Next, section III describes the three proposed models of integrating Grid and Cloud resources for job executions. Then, section IV introduces a case study using a protein design application which uses one of the proposed models of integration to assess the benefits of the hybrid approach. Finally, section VI concludes the paper and points to future work.

II. Related work

There are previous works in the literature that aim at combining the usage of Grid and Cloud infrastructures. For example, in [3] the authors focus on the integration of High Performance Grids with Cloud environments, determining different usage models for applications (acceleration, conservation and resilience). For that, they use the CometCloud software, an autonomic framework to execute application on dynamically federated hybrid infrastructures based on Grids and Clouds. In [4], the authors study the usage of hybrid infrastructures for the execution of workflows, where the scheduling criteria are chosen through the SLA (Service Level Agreement) specified by the user from these ones: runtime, execution cost, reliability and network bandwidth. Other works in the literature also address the integration of Cloud concepts in Grid infrastructures, such as the virtualization of Grid resources, as described in [5] or the illusion of elastic Grid infrastructures using a Cloud provider as a backend when peak demands of computing resources are requested, as proposed in [6].

There are also different approaches, like in [7], where the ProActive middleware is employed to perform High Performance Computing (HPC) executions of applications on hybrid infrastructures based on Grid and Cloud resources. SAGA [8] is an API that enables applications to be submitted to Grid and Cloud resources without application modification. However, the decision of the infrastructure to be employed relies on the user. Swarm [9] is a job manager that enables application execution on three different infrastructures (Grid, Windows Server Cluster and Cloud). However, for simplicity the Virtual Machines are assumed to be already set up before the scheduling process, thus neglecting the typical overheads on Cloud deployments such as deployment of Virtual Machines and their contextualization and configuration.

This paper focuses on hybrid meta-scheduling for High Throughput Computing (HTC) applications rather than High Performance Computing (HPC).

III. Concurrent Usage of Grid and Cloud infrastructures

The usage of computational resources from mixed infrastructures can be achieved by means of pluggable components to access each infrastructure linked to an orchestrating central manager that coordinates both the resource provisioning and the meta-scheduling of jobs on these resources.

For that, Figure 1 depicts the reference architecture of a hybrid Grid/Cloud infrastructure, together with the required components for job meta-scheduling, as considered in this paper. On the left hand side, the user wants to execute an HTC application, such as a parameter sweep study that consists of many independent jobs. The user submits the set of jobs to the Enactor module together with the job's hardware (architecture, available RAM, etc.) and software requirements (OS, application dependencies, etc.). The Enactor decides the most appropriate infrastructure for each job and orchestrates the Grid and the Cloud connectors to perform the execution.

Executing the jobs on a Grid can be directly performed via a Grid metascheduler such as GridWay or GMarte [10] and has been extensively studied in the literature. However, scheduling jobs to be executed on a Cloud infrastructure is a three-stage process. First of all, the infrastructure of VMs must be deployed on top of the physical infrastructure. Next, the jobs must be allocated to be executed on the VMs. Finally, the infrastructure of VMs must be decommissioned (undeployed) if there are no pending jobs to be executed. Concerning the policies of deployment and undeployment of VMs, certain parameters should be considered such as the number of pending jobs, their estimated duration, the current state of the Grid infrastructure and the allocated budget for the executions, when requesting computational resources from a Cloud provider. This is responsibility of the Cloud Enactor, depicted in detail in Figure 2.

When performing job executions in a Cloud environment, the architecture on which this paper is based includes a catalog and repository service of Virtual Machine Images (VMIs) such as the Virtual Machine Image Repository & Catalog (VMRC) [11] that enables to search for the most appropriate VMI to execute a given job considering both the hardware and software requirements of the job. The architecture includes a contextualization service that is in charge of configuring the VMIs to deploy the appropriate software dependences for the execution of an application on a VM. For example, when a job that requires the Java Virtual Machine to be executed is submitted to the Cloud enactor, this component would try to find an appropriate VMI that already satisfies the job requirements. If no one is found, then the contextualization service would be employed to dynamically deploy Java onto the VM at runtime before the execution of the job. Notice that the combined usage of the contextualization service and the VMRC service reduces the average contextualization and configuration time of VMIs since this phase might only be performed once if the configured VMI is later stored in the VMRC and used for subsequent executions.

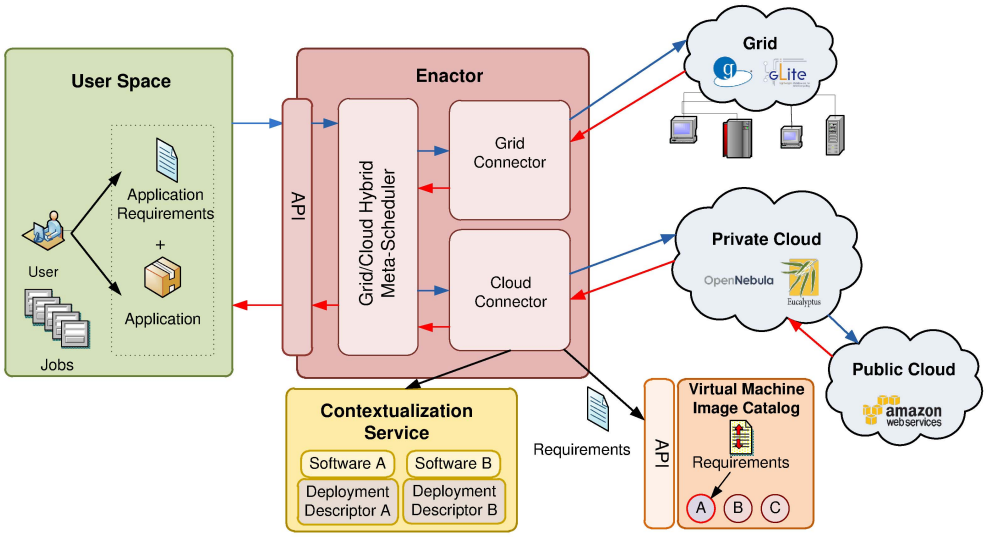


Figure 1. Combined management of Grid and Cloud resources for scientific application execution.

A. *Parameters involved in the meta-scheduling process*

The concurrent usage of Grid and Cloud resources requires analyzing the parameters involved in the meta-scheduling process in order to decide the most convenient infrastructure and the computational resources that are able to support the execution of the job. There are common parameters to be considered for both infrastructures to perform hybrid meta-scheduling on Grid and Cloud infrastructures and specific items, specially for Cloud infrastructures. Firstly, we analyze the most important common parameters:

- **Estimated execution time of the job.** This should be an estimator of the makespan of the job in a concrete platform in order to extrapolate the required time on other platforms. If appropriate, the execution of multi-parametric batch jobs can be instrumented to have a historical background of the average execution time of the jobs on a particular infrastructure. A fine grain detail of the execution time of the jobs allows, for example, to schedule shorter jobs to slow computational resources when scheduling groups of inhomogeneous jobs.
- **Data transfer time.** The time requires for file staging from user space into the computational infrastructure (and vice versa) needs to be calculated in order to compute the overhead of performing a remote execution. A data-intensive application might not benefit from a remote execution if a 1 GByte database should be transferred for a 5 minute processing. Both the required input files to the job and the bandwidth between user space and the remote computational resources is employed to estimate this value. Notice that the bandwidth can be dynamically recomputed after each transfer, so that the meta-scheduler can adapt to network changes.
- **Application dependences.** Scientific applications rely on software dependencies such as numerical libraries, software packages and specific operating systems. Therefore, application dependences should be considered when scheduling jobs since this information is employed to find computational resources that (partly) match these dependences. The hardware dependences, such as CPU architecture or available RAM are mandatory for a successful and efficient execution.
- **Type of job.** The type of application determines how appropriate a computational resource is. An HPC parallel application would certainly benefit a dedicated cluster on a Grid with low latency communications among the nodes. However, independent serial jobs can seamlessly run on virtualized resources from a Cloud.

These are the most important parameters that should be considered when provisioning virtualized resources from a Cloud infrastructure and scheduling jobs to be executed on them. Notice that depending on the Cloud provider, some of these could be neglected. For example, the budget only plays a role for a public Cloud provider.

- **Access time to the Catalog of VMIs.** This component enables to choose the most appropriate VMI considering the application requirements. The time required to interact with this service involves mainly the VMI transfer from the catalog to the deployment site in the Cloud, which merely depends on the state of the underlying network and the VMI size.
- **Deployment time of the VMs.** This is the time required by the VMM to deploy a new VM on top of the physical

infrastructure. With the advent of Green Computing techniques, which enable to dynamically turn on and off physical resources where the VMs are executed, this time might get increased to include the time required to boot up the physical node and the VMI.

- **Contextualization and configuration time.** This phase enables to deploy the appropriate software dependencies of the application and the application itself onto the VM in order to satisfy the job software requirements.
- **Availability of already-deployed VMs.** A user might already have deployed and contextualized VMs in a Cloud (due to previous executions, for example). Therefore, under these circumstances the overhead introduced by the deployment of new VMs would be alleviated.
- **User budget.** Accessing a public cloud on a pay-per-use basis requires defining a maximum budget to be spent on provisioning resources. This imposes restrictions on the amount of VMs to be deployed and the time these VMs should be active.
- **Billing policies.** If the provider bills for complete hours, the allocation of jobs could try to refine scheduling decisions to avoid a 61-minutes execution, since two hours would be charged. Alternatively, there is no point in shutting down the VMs if a full hour has already been paid. These could be used as spare resources to execute jobs.
- **Time slots.** If the Cloud provider defines certain time slots with different prices, these should be considered when deciding to provision and release virtual resources, trying to minimize the budget consumption of the user while maintaining an appropriate QoS.
- **Trust and reputation.** In a multi-cloud scenario, where virtual computational resources can be provisioned from different Cloud providers, trust and reputation play a fundamental role. The trust might be estimated by a historical log of failures obtained when using a certain provider, the adherence to the Service Level Agreement (SLA) and the QoS obtained.

B. Models of Grid/Cloud Integration

There are different scenarios in which a hybrid Grid/Cloud infrastructure introduces potential benefits for scientific computing. We have identified three different scenarios that will be analyzed in this subsection.

1) Outsourcing to the Cloud when Grid resources become exhausted: A Cloud infrastructure would be used when the Grid infrastructure cannot cope with the workload (in terms of number of jobs) to be executed by the user. Despite the large scale of Grid infrastructures, the users are typically restricted to the resources from one or several VOs. Therefore, depending on the current workload of the accessible resources, the user's jobs might suffer from resource starvation. This is where the Cloud infrastructure can temporarily alleviate this problem, with its ability to

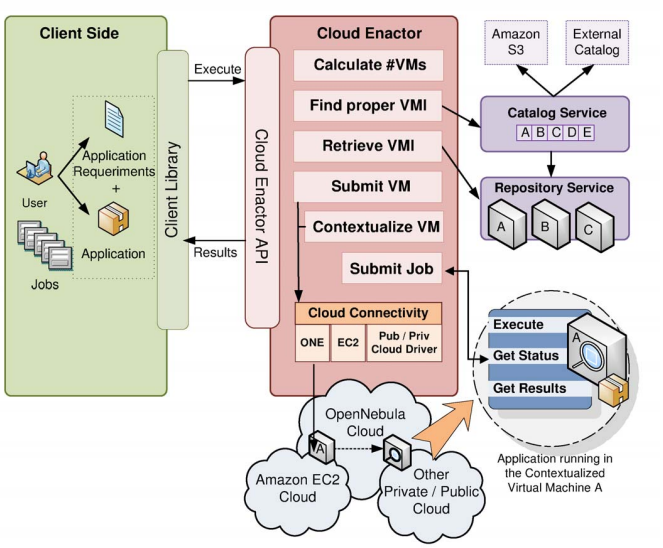


Figure 2. Detailed view of the Cloud Enactor.

elastically deploy a virtual computational infrastructure on which to delegate the execution of additional jobs until the Grid infrastructure is ready to accept the execution of new jobs. Outsourcing computations to a Cloud enables to complement the computational power of Grid with the on-demand elasticity provided by the Cloud. This allows to maintain an appropriate quality of service when facing an unexpected peak workload or an outage.

This scenario proceeds as follows. The meta-scheduling process decides that a job should be executed on the Grid. For that, the job is delegated to the appropriate enactor in our architecture. When the job is scheduled to be submitted to the Grid, the enactor detects that the infrastructure has no available execution slots, by querying the information system (MDS, BDII, etc.) Since the job cannot be executed the Grid metascheduler notifies the enactor. If there is no available Cloud infrastructure, the execution of the job would be postponed until a free execution slot could be found. However, if a Cloud infrastructure can be accessed, the enactor delegates the job into the Cloud metascheduler which requires the appropriate virtual infrastructure for the execution of the job(s) onto the virtual machines.

2) Using the Cloud when job requirements cannot be met: Scientific applications might involve a large number of software and hardware dependencies. If no computational resource is found that satisfies the requirements of the job, then it cannot be executed. In a Grid infrastructure, this problem arises when complex software must be executed on a machine that is not under the control of the user. It is not feasible to try to adapt the computational resource to the requirements of the jobs. For the hardware requirements, this is definitely impossible. For the software requirements,

the Grid user might try to deploy the software dependences at runtime before the execution of the job although the installation might be tricky and since no admin access to Grid resources can be achieved. This is where the Cloud infrastructure helps to solve this problem.

This scenario proceeds as follows. After analyzing the hardware and software dependencies of the job, the information system of the Grid is queried in order to find the most appropriate computational resource that matches the specified requirements. If there is no available Grid resource to execute the job, the enactor delegates the execution of the job to the Cloud scheduler, which would deploy the appropriate virtual infrastructure to support the execution of the job.

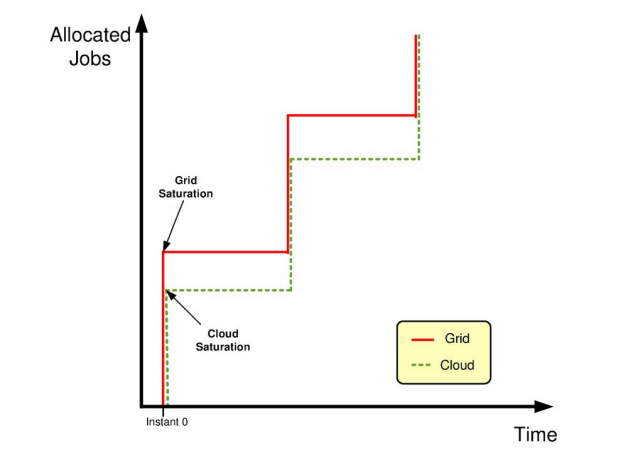
3) Using the Grid and the Cloud for workload distribution: This scenario involves using both the Cloud and the Grid infrastructure as shoulder-to-shoulder providers of computational power for the concurrent execution of jobs. This involves a workload distribution among both infrastructures in order to access a larger pool of computational resources to accelerate the execution of jobs. A workload distribution among separate infrastructures allows executing the jobs with a higher throughput, since a larger number of resources is concurrently used (depending on the availability and budget of the user). This is of special importance for multi-parametric batch jobs, where independent jobs are executed.

In this scenario, the enactor is in charge of dispatching jobs to the Grid and the Cloud schedulers, trying the balance the load among both infrastructures. The desirable outcome in this model consists in maximizing the throughput of both infrastructures in terms of resource usage, that is, trying to increase the rate of finished jobs per time unit. This might involve greedy approaches that try to take advantage of the peak computational capabilities obtaining as much resources as possible from both infrastructures.

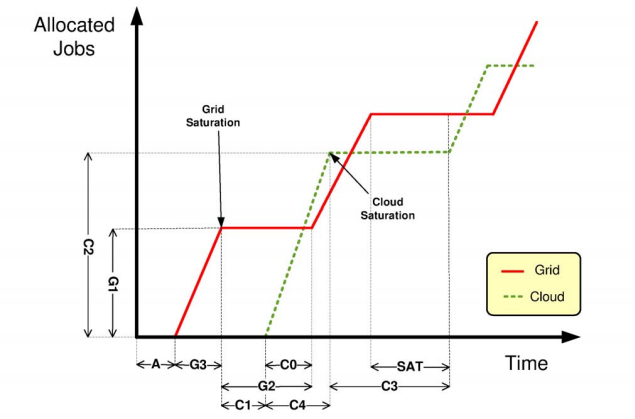
C. Assessing the mixing of Grid and Cloud infrastructures

For the sake of a better explanation of the benefits of a hybrid Grid/Cloud infrastructure, this section details the conditions concerning the first execution model described in the previous section. This one will be employed for the execution of the case study in section IV.

Figure 3 depicts the workload conditions, based on allocated jobs versus time, of a combined usage of a Grid infrastructure and a Cloud infrastructure where the latter is only used when the former has no free available resources left. A High Throughput Computing scheme is assumed, where a certain pool of jobs is available to be executed by the enactor or a flow of jobs arrives to the meta-scheduler. The jobs are assumed to have a similar execution time. This is precisely the case of parameter sweep applications and case studies that arise in many scientific and engineering applications.



(a) Ideal situation



(b) Real situation

Figure 3. Workload conditions of Grid and Cloud infrastructures under ideal and real situations.

On the one hand, Figure 3.a assumes ideal conditions in which no overheads exists when dealing with the computational infrastructures and the meta-scheduling process is perfect, distributing the workload evenly among the infrastructures so that it has the highest rate of job executions. The job allocation begins at instant 0 (depicted with an offset from the y-axis for the sake of clarity) and it is assumed to involve no extra time. Under these ideal conditions, the only action that consumes time is the execution of the jobs. A group of jobs are initially allocated to the Grid until all the execution slots are filled. Immediately, the Grid saturation is detected and the virtual infrastructure is deployed in negligible time (ideal conditions) and jobs are allocated to the VMs. When the jobs finish the execution, other pending jobs can be scheduled on the spare computational resources.

On the other hand, Figure 3.b includes all the overheads discarded in the previous figure, thus depicting a realistic

situation. The meta-scheduling process starts at time 0, where a certain time (A) is involved in the initial setup of the scheduling of the jobs. This depends on the Grid infrastructure and the meta-scheduler itself. Scheduling the jobs into the Grid infrastructure requires querying the Grid information systems (MDS, BDII, etc.) to get the most up-to-date information about the current state of the Grid infrastructure to better decide how to distribute the jobs among the available computational resources. Then, jobs have to be submitted for execution. This requires a certain time ($G3$). The number of resources available to the user might impose an upper limit to the degree of concurrency in the execution of jobs, thus defining the maximum number of jobs concurrently executed on the Grid ($G1$). The amount of time involved in executing the jobs in the Grid is depicted in the figure by $G2$ and it depends on the computational load of the jobs and the capabilities of the Grid resources. Notice that the differences of the computational capabilities of the Grid resources might introduce a variability in the execution time of the jobs. This would produce a staircase effect in the slope of the job allocation to the Grid, since jobs finished would create free execution slots for new jobs to be allocated.

When the enactor module detects that the Grid is overloaded it must decide to provision and use Cloud resources. This takes a $C1$ time. This involves provisioning the computational resources from a Cloud provider (or a Virtual Machine Manager). $C2$ represents the capacity of the allocated Cloud infrastructure. This depends on the budget of the user, which might define the maximum number of concurrent Virtual Machines (allocated computational resources) to be deployed. Once the virtual infrastructure is set up, it is time to contextualize the VMs in order to support the execution of the scientific application and to allocate the $C2$ jobs on them. This step is performed by the Cloud scheduler, which requires a $C4$ time.

The $C3$ interval corresponds to the time invested in the execution of the jobs in the Cloud. This might be larger than the time invested in executing the same number of jobs in a Grid due to the overhead of virtualization and the fact that Grid resources tend to be more computationally powerful than VMs in a Cloud. $C0$ stands for the time since jobs have been delegated to the Cloud infrastructure until new execution slots are available in the Grid infrastructure. A short $C0$ means that it does not pay off to outsource to the Cloud since by the time the virtual infrastructure is ready, the Grid resources would be free. The time interval SAT indicates that both infrastructures are saturated and, therefore, cannot execute new jobs. This is a desirable situation from the point of view of resource usage.

Notice that Figure 3.a assumes that the Grid infrastructure has a larger number of job execution slots than the Cloud, whereas Figure 3.b assumes a Cloud able to provision a larger number of VMs than execution slots available in the Grid. This distinction enables to analyze the behavior from two different point of views, depending on the budget of the user to provision Cloud resources and the number of Grid resources available to execute the jobs. The latter typically depends on the availability to access the resources at different VOs within a Grid. Considering that the default policy of Amazon EC2 is to allow a maximum of 20 concurrent instances (VMs) per EC2 region, a large Grid deployment might offer larger computational capacity to increase the number of simultaneous jobs in execution.

IV. Case Study

In order to assess the behavior theoretically studied in the previous section, this section analyses the execution of a case study on a hybrid Grid/Cloud environment. A computationally intensive scientific application which performs the optimization of protein with target properties on both sequential and parallel platforms [12] has been employed. The case study is composed of multi-parametric batch jobs where, in this study, different random seeds are employed to perform the optimization of the same protein. The case study has been specially downsized to get job executions in the order of minutes (instead of hours) to better focus on job scheduling.

The application uses Monte Carlo Simulated Annealing (MCSA) to explore the conformational space of rotamers in a bi-objective optimization process that (1) attempts to obtain a stable protein by minimizing its energy and (2) pursues the functionality of adhering to a certain ligand. A synthetic protein of 1000 positions is employed. The optimization is an iterative procedure where each iteration consists of evaluating the suitability of a certain rotamer (the three-dimensional structure of an amino acid) in a position, from the set of candidate rotamers that could be in that same position. This stochastic procedure for protein design typically converges to the most stable protein after thousands of iterations. Therefore, its computational cost can exceed the resources of a single organization. The usage of hybrid Grid/Cloud infrastructures can greatly benefit this application in order to access a larger pool of computing resources. The application has been developed in the C programming language and it requires a C compiler and the MPICH Message Passing Interface (MPI) libraries.

Regarding the hybrid infrastructure, we have relied on computational resources available within our research group for the sake of experimentation with hybrid deployments. The Grid infrastructure is composed of an HPC cluster of PCs with the Globus Toolkit 4.2.1 installed with a total 50 nodes dual-processors Intel Xeon 2.80 Ghz with 2 GBytes of RAM per node. The GMarte meta-scheduler [10] is employed to execute on the Grid using a greedy approach that selects the most appropriate resource for the execution of each job. In scenarios on which jobs from multiple users are randomly submitted to a single meta-scheduler,

a global scheduling of jobs is unfeasible. GMarte considers the current load of computational resources together with the dynamic state of the network to choose the most suitable resource (trying to minimize the job makespan). The Cloud infrastructure is based on a deployment of OpenNebula 2.2, using the VMWare hypervisor, on a cluster of PCs with 20 dual-processor Pentium Xeon 2 GHz with 1 GByte of RAM.

Since the clusters are also employed for the execution support of other research projects, this case study will employ a dedicated subset of nodes. 10 nodes for the Grid ($G1$) and 4 nodes for the Cloud ($C2$). Each job will run on a single node. Therefore, the maximum number of concurrent jobs corresponds to the number of available nodes in the hybrid Grid/Cloud infrastructure (14). Other jobs will only be processed as soon as the running jobs finish. In addition, single-CPU identical VMs will be deployed. A round-robin strategy will be employed to allocate the jobs on the VMs, since the will be equally capable. The deployment of the VMs that host the execution of the application will use a VMI that already contains a pre-installation of MPICH2 and the application itself. The VMI will be cataloged in the VMRC catalog. The number of VMs to deploy will be the minimum between the capacity of the physical infrastructure and the number of pending jobs. The VMs will be deployed until the end of the executions in order to avoid repeating the process of VM deployment and contextualization.

Starting and monitoring the jobs is made through GRAM (Grid Resource Allocation Manager) in the case of the Grid. For the Cloud, we use the Opal 2 Toolkit [13] a web services wrapper that enables to start and monitor applications inside the VM. This component will be deployed at runtime through in-house developed contextualization software. The case study involves 30 protein design jobs with different random seeds. The jobs have a similar duration.

V. Results & Discussion

Figure 4 shows the distributions of the tasks between the two infrastructures (Grid and Cloud) using the approach of outsourcing the job executions to the Cloud when Grid resources become exhausted. In order to assess the behavior of the execution model, the results analysis has been performed considering the time intervals mentioned in section III-C.

According to the results obtained, a single node of the Grid delivers better performance than a VM deployed on the Cloud infrastructure since a job in that Grid takes an average 6 minutes ($G2$) while in a VM in that Cloud takes an average 8 minutes ($C3$). Notice that $G2$ and $C3$ is the execution time of the set of jobs submitted to a particular infrastructure. However, since the jobs are concurrently executed, these values also represent the execution time of a single job on each infrastructure. This time difference together with the fact that the Grid has more job execution slots, made the Grid the preferred decision by the hybrid meta-scheduler. A total of 22 jobs were executed on the Grid.

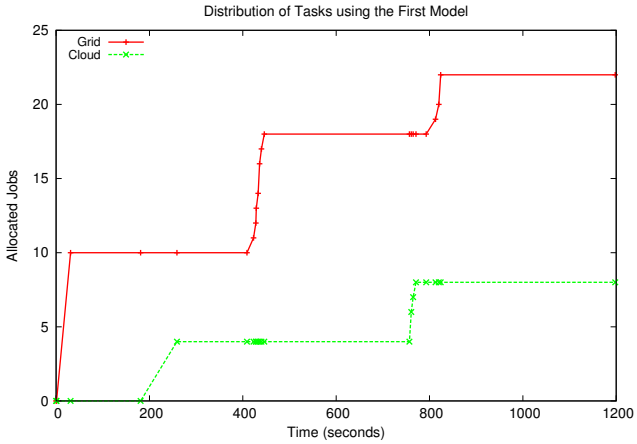


Figure 4. Distribution of tasks when outsourcing the job executions to the Cloud when the Grid resources are exhausted.

Comparing the ideal and the real situations, the initialization time of the hybrid meta-scheduling process A is negligible. Focusing on the time intervals concerning the Grid it can be seen that the real executions shows a very similar behavior to the theoretical approach. The $G3$ interval, devoted to the allocation of jobs to the Grid and transferring the input files to the computational resources, is reduced compared to the time dedicated to the execution of the jobs, which is considerably larger ($G2$).

Concerning the intervals related to the execution of jobs in the Cloud, the interval $C1$ involves the following steps:

- **Detecting the exhaustion of the Grid infrastructure.** It involves querying the Grid information system (MDS) to find out that there are no spare computing resources left, thus involving a negligible time.
- **Deploying the VMs.** This is the time since a request to the VMM is sent until the VMs are up and running and the system services (like SSH) ready. This involves an average 150 seconds. The OpenNebula configuration in the cluster must perform an SSH-based copy of the VMI to the internal node before booting it up.

The time interval concerning the allocation of jobs to the cloud $C4$ involves the following phases:

- **File staging.** This involves uploading the files required to contextualize the VMI. Since the VMI has the application pre-installed, minimal configuration components must be uploaded. This involves a negligible time.
- **Contextualization.** This phase will be in charge of deploying Apache Tomcat and Ant, together with Opal, and later deploy the scientific application to be used by Opal. This takes an average 80 seconds.

It is important to point out that the VMs are concurrently deployed in order to simultaneously perform the contextualization time of each VM. Besides, they are not shut down until the end of the execution of the case study. This is

why when using the Cloud for the second time (at time instant 780) the interval $C1$ no longer appears since the VMs are already deployed. Also, the interval $C4$ is halved since the VMs are already contextualized and ready to accept job execution. The $C0$ value is large enough for the hybrid meta-scheduler to consider scheduling jobs to the Cloud, since the contextualization time of the VMs is relatively small compared to the time invested in the execution of the jobs.

An important aspect is the time that both infrastructures are at a peak load (SAT), since this is an evidence that resources are being fully utilized and proper scheduling with reduced overhead is being performed; in this case, a 78.3% of the time. Outsourcing job executions to a Cloud platform whenever the Grid infrastructure resources become exhausted enables to deliver additional power to the execution of computationally intensive HTC applications. This reduces the total execution time of the case study. The execution approach involved in this case study can be seamlessly extrapolated to larger studies and infrastructures.

VI. Conclusions and Future Work

This paper has studied the advantages of using a hybrid infrastructure composed by Grid and Cloud resources. These two technologies can work together providing the scientific community with an environment in which the researchers can execute computationally intensive scientific applications.

The proposed prototype is able to efficiently execute HTC scientific applications on a hybrid infrastructure. A mixed infrastructure composed of Globus Toolkit resources, for the Grid, and Virtual Machines deployed through OpenNebula, for the Cloud, has been evaluated. The scheduling approach enables to outsource job executions to the Cloud when no spare Grid resources are available. In addition, other models of hybrid Grid/Cloud execution models have been covered, pointing out the benefits of the Cloud in terms of elasticity and configurability. The usage of hybrid infrastructures enables to access a larger pool of computational resources which reduces the execution time of HTC application when compared to single infrastructures.

The future work involves improving the resource selection in Cloud platforms, together with studying new usage models on hybrid infrastructures, specially with the advent of sky computing where multiple Cloud providers can be chosen. The studies will be extended to public Clouds such as Amazon EC2 where network latencies and budget plays a major role when scheduling jobs to the Cloud.

Acknowledgments

The authors would like to thank the financial support received from the Vicerrectorado de Investigación de la Universidad Politècnica de València for the project PAID-06-09-2810 and to the Ministerio de Ciencia e Innovación for the project CodeCloud (TIN2010-17804).

References

[1] I. Foster and C. Kesselman, *The GRID 2: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 2004.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," Tech. Rep., 2009. [Online]. Available: http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf

[3] H. Kim and Others, "An autonomic approach to integrated hpc grid and cloud usage," in *2009 Fifth IEEE International Conference on e-Science*. IEEE, 2009, pp. 366–373.

[4] H. Kloh, B. Schulze, A. Mury, and R. Pinto, "A scheduling model for workflows on grids and clouds," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, no. December. ACM, 2010, p. 3.

[5] E. Huedo, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Architectures for Enhancing Grid Infrastructures with Cloud Computing," in *GRIDS, CLOUDS AND VIRTUALIZATION. Computer Communications and Networks*, ser. Computer Communications and Networks, M. Cafaro and G. Aloisio, Eds. London: Springer London, 2011, pp. 55–69.

[6] C. V. Blanco, E. Huedo, R. S. Montero, and I. M. Llorente, "Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers," in *2009 Workshops at the Grid and Pervasive Computing Conference*. IEEE, May 2009, pp. 113–120.

[7] B. Amedro, F. Baude, and F. Huet, "Combining Grid and Cloud Resources by Use of Middleware for SPMD Applications," *Conference on Cloud*, pp. 177–184, 2010.

[8] A. Merzky, K. Stamou, and S. Jha, "Application Level Interoperability between Clouds and Grids," *2009 Workshops at the Grid and Pervasive Computing Conference*, pp. 143–150, May 2009.

[9] S. Pallickara, M. Pierce, Q. Dong, and C. Kong, "Enabling Large Scale Scientific Computations for Expressed Sequence Tag Sequencing over Grid and Cloud Computing Clusters," in *Eigth International Conference on Parallel Processing and Applied Mathematics (PPAM 2009)*. Citeseer, 2009.

[10] J. M. Alonso, V. Hernández, and G. Moltó, "GMarte: Grid middleware to abstract remote task execution," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 15, pp. 2021–2036, Dec. 2006.

[11] J. V. Carrión, G. Moltó, C. De Alfonso, M. Caballer, and V. Hernández, "A Generic Catalog and Repository Service for Virtual Machine Images," in *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.

[12] G. Moltó, M. Suárez, P. Tortosa, J. M. Alonso, V. Hernández, and A. Jaramillo, "Protein design based on parallel dimensional reduction." *Journal of chemical information and modeling*, vol. 49, no. 5, pp. 1261–71, May 2009.

[13] S. Krishnan, L. Clementi, J. Ren, P. Papadopoulos, and W. Li, "Design and Evaluation of Opal2: A Toolkit for Scientific Software as a Service," in *2009 IEEE Congress on Services*, 2009.