Effectively and Efficiently Supporting Grid and Cloud Integration via a DBMS-based Framework

Alfredo Cuzzocrea[1], Osvaldo Gervasi[2], Mirko Mariotti[3], Flavio Vella[4], and Alessandro Costantini[5]

[1] DIA Dept., University of Trieste and ICAR-CNR, Italy
alfredo.cuzzocrea@dia.units.it
[2] DMI Dept., University of Perugia, Italy
osvaldo.gervasi@unipg.it
[3] FISGEO Dept., University of Perugia, Italy
mirko.mariotti@unipg.it
[4] DI Dept., University of Rome La Sapienza, Italy
vella@di.uniroma1.it
[5] CNAF-INFN, Italy
alessandro.costantini@cnaf.infn.it

Abstract. This paper provides anatomy, models and functionalities of a DBMS-based systems for integrating Grids and Clouds. Our study starts from recognizing the similarity of some axioms of Grid and Cloud computing, still being these computational paradigms very different for what regards both computing and economic models. Our proposed system is centered along a well-designed DBMS schema that allows to obtain a seamless integration between Grids and IaaS Cloud providers. The paper details how images from a Cloud environment are deployed in reply to a specific task execution invoked from the (integrated) Grid environment, as well as other essential components of the proposed architecture (e.g., resource access and grant, user authorization, resource discovery and sharing, job and task management and distribution, integration with other computational platforms, and so forth).

1 Introduction

Nowadays two trends are evident in the way the computational resources are organized and managed to provide users the computing infrastructures adequate for the emerging computing needs. On the one hand, virtualization technologies are massively adopted, based on more and more powerful Cloud systems (Openstack, Opennebula, Eucaliptus, etc.), along with systems for deploying virtual machines and all technologies related to the Cloud scenario. On the other hand, it is clear that in order to increase the performances of the computing systems the best way is to adopt heterogeneous architectures, specializing them on the basis of the requested type of computation from the users. Examples of this type are the usage of GPUs for the fast solution of different problems in computer science like graph analysis [9], cryptography [19, 29, 30] or computational logic[16], the

development of innovative architectures, like the Parallella board[24], and the adoption of Field-Programmable Gate Array (FPGA) device as a computing resource[20, 12]. Cloud and Grid computing share some essential driving ideas which led to the construction of both large scale federated Grid infrastructures which can be summarized as follows: (*i*) bring the promise of encapsulating the complexity of hardware resources and make them easily accessible by means of high-level user interfaces; (*ii*) address some form of the intrinsic scalability issues of large scale computational challenges; (*iii*) cope with the need of resources that cannot be hosted on premises.

However, the key differences between Grids and Clouds concern abstractions and compute models adopted by both paradigms [18]. It can be said that Grids are built "bottom up" and are concerned more with federation of static existing resources that typically are legacy clusters built around a Local Resources Management System (LRMS) that exploits the Batch computing model.

The development of applications for Grid environments requires the knowledge of the Grid infrastructure abstractions. This process, aimed at enabling the application to run in such environments, is in fact called "Grid-enabling" and can be rather complex [17]. On the other hand, Cloud users can choose their own compute model, leveraging more general (without the needs of a fine tuning of the environment) interfaces that often lead to simpler interaction and application development [27].

As sketched before, there are several problems that do not play nicely with the heterogeneous nature of aggregated resources in Grids. For example, many scientific applications need different environments (operating systems, libraries) and hardware (i.e., Multicore Processors, FPGA, GPUs, etc.). From a Batch point of view this represents a set of requirements influencing scheduling decisions for both top and local level resource managers. So the Grid sites heterogeneity plays a central role in job distribution (workload) among sites that match the aforementioned requirements.

Furthermore the Grid workload can be often unpredictable and subject to burst increase, that lead to unbalanced distribution in resource usage, and even deterioration of QoS. In this context the Grid workflow represents a weak point for the Batch model in which resources are often statically managed and partitioned, and cannot be adapted in advance to meet possible requirements. Moreover the use of Clouds could allow the extension of private resource pools in number and typology with positive effects on Quality of Service (QoS).

So why do not dismiss Grids and adopt Cloud solutions? There are several reasons: it is not yet clear how some critical issues (data management, security, etc.) are to be dealt with in the Cloud era, while in Grid are well-established. Furthermore, the costs of an eventual shift in technology must be thoroughly investigated. A more reasonable approach is an integration process that combines the features of both.

2 Integration Opportunities

Before the Cloud era, even if these issues were addressed in various works[10, 28, 21], the proposed solutions were often heavy customized and too tightly dependent on particular technological choices.

With the success of Cloud computing through the spread of IaaS providers [7, 2, 3], the development of interfaces for the simplification of virtual management [4, 6] and related libraries (i.e.,[5, 1], etc.) paved the way to several possibilities for Grid and Cloud integration. As a matter of fact, even if Cloud solutions have been, since their first definition[13], primarily driven by business motivations, the IaaS service model seems to respond to some of the Batch model issues and can overcome them with both on-demand and adaptive characteristics.

To the best of our knowledge there are three approaches of site-level integration between the Batch oriented Grid compute model and the service oriented nature of Clouds.

Grid over Clouds. According to this alternative, a whole Grid site is built on top of a public/private Cloud. Through this schema, the Grid infrastructure can be built by instantiating resources according to the real needs of the users. In [8], the authors provided a "Grid as a service" tool in order to create new Grid sites, or to add computational resources to existing Grid sites by exploiting a Platform as a service (PaaS) approach. Similar approach is also adopted in [23].

Hybrid with Batch-Dependent Cloud-Enabled LRMS. In this model a single local Batch system is used to schedule the jobs on a pool of dynamically provisioned resources either on premises or public/private Clouds. For example, in [26], the authors described a solution which enables building dynamical environments through Grid jobs or local Cloud jobs. The solution proposed is built around the LRMS which handles each request. This approach presents manifold limitations. The main drawbacks are the following: *i*) the solution is strongly dependent on a particular technology adopted (i.e. LRMS requires customizations); *ii*) the approach is not *elastic*[11] since it enables the spawn of a virtual environment on local resources only.

Hybrid no Batch-Dependent. In this model the local Grid site spawns resources (even whole clusters) on public/private Clouds on the basis of the jobs requests. The integration is done at the Computing Element level. In this way, several computational resources (i.e. resources available on other computational centers) can be exploited by a fine grained control over virtual instances. In [29], the first solution based on Cloud-over-Grid approach was presented. The authors also validated their solution providing to Grid users special virtual computational resources as GPUs.

The last two approaches can be also identified as two types of "Cloud-over-Grid". In the present work, we describe our solution, that may be used to implement any of the described hybrid approaches with the special attention to the no Batch-dependent model.

3 System Anatomy

The proposed system is based on the adoption of Cloud systems to enable the Grid sites to provide to the users a set of non-traditional resources, like the aforementioned ones and dynamical environments. As an example a Grid user may request to run in a server equipped with a given GPU, or with a particular software library installed or operating system.

Our system has been designed according to the Unix principles: each component is autonomous, independent from the others, specialized in carrying out a named task in a simple way. According to this approach we have chosen to use tags for cataloging the virtual machines that have a certain type of hardware features (such as a named architecture, hybrid systems, GPU, etc.) or software (operating systems, special libraries installed). These tags are published using the standard techniques of the Grid environment, as features implemented and published by a particular site, and which can be specified as requirements by the users when submitting a job. In this way, the Grid information system enable the users to submit jobs requiring special environments, provided only by some Grid sites.

In Figure 1 the project logical schema is sketched. Our solution in built around a DBMS that plays a central role since it contains the configuration of the system, in terms of the connected clusters, the Cloud systems, and their environments and status. The architectural workflow of our solution is implemented by different agents connected to the DBMS each performing a specific action; they will be described in detail later.

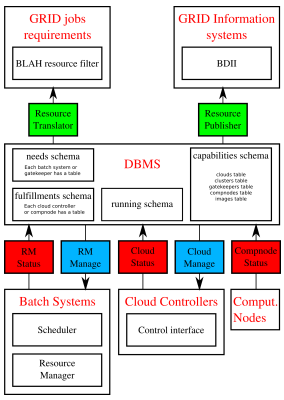

Fig. 1. Description of the Proposed System

In the remaining part of this paper we will use the following terms, and corresponding meaning:

Computing Element (CE): is the set of resources made by the *Gatekeeper* and the *Cluster*.

Gatekeeper: is the system that provides the gateway through which the Grid jobs are submitted to the Batch system running on the local farm nodes;
Cluster: it is a Grid enabled Cluster, i.e. a bunch of *Worker Nodes* (WNs), connected to a Computing Element (CE) and connected to the Grid system. When referring to Clusters we will mean the Cluster Resource Manager.
Cloud: it is a Cloud infrastructure with a Cloud controller like OpenNebula, OpenStack or Eucalyptus.
Computational node: a single server used as target of the incoming Grid job without the use of a Batch system or a Cloud System.
Cloudtag: Tag used to mark the images and to organize the infrastructure resources.

4 DBMS Structure

The information about Clusters and Clouds is collected on a DBMS system. From implementation point of view we have chosen PostgreSQL for this purpose. The informations have been divided into four logic blocks, each one mapped to a DBMS schema:

- The *capabilities* schema contains informations about the Clusters, Cloud Controllers and Virtual Machine images known to the system. It also contains the information about tagging the Virtual Machine images to publish this information through to Grid information system.
- The *needs* schema contains a live view of the *cloudtag* needed by clusters.
- The *fulfillments* schema contains a live view of the *cloudtag* offered by Cloud systems.
- The *running* schema contains the list of the running jobs, with related details.

4.1 The Capabilities Schema

The *capabilities* schema contains the information related to the composition of the various systems. Each software component reads from the DB the necessary information, since the capabilities schema contains all the information related to the structure of the system. The information contained in this schema concern the Cloud, Clusters, Gatekeepers, the Computational nodes connected to the system, and, more important, the Virtual Machine images.

Information on the Active Cloud Systems The table *clouds* of the capabilities schema traces the following information related to the active Cloud systems: (*i*) type of Cloud system (i.e.: OpenStack, OpenNebula or Eucaliptus); (*ii*) the description of the Cloud system; (*iii*) the information on how to interact with the system, which may, or may not, contain authentication information.

Information on the Active Clusters The table *clusters* of the capabilities schema traces the following information related to the Batch system of the active Clusters: (*i*) type of Cluster's Resource Manager (Torque/MAUI, LFS etc); (*ii*) the description of the Cluster; (*iii*) the optional information on how to interact with the Batch system of the Cluster.

Information on Gatekeepers The table *gatekeepers* of the capabilities schema traces the information related to the Gatekeepers of the active Grid nodes. In particular, the more important information are: (*i*) gatekeeper information and the Information System of the Grid site; (*ii*) description of the Grid site; (*iii*) the optional information on how to access the Gatekeeper and/or the Information System.

It is relevant to notice the reason why we implemented two separate tables, one for Gatekeepers and one for the Batch Systems, even if the Grid site is the same, then the CE is listed in the *gatekeepers* table and the Batch System in *clusters* table. We kept separated the two tables because we want to stress the fact that the job path, and the related sequence of events and actions, are different if they are under the control of a Batch System or not.

Information on Standalone Computational nodes The table *compnodes* of the capabilities schema traces the information related to the access to Computational nodes capable of executing jobs. They represent the real or virtual machines not connected to a Batch System, we want to include in our System. The most relevant information are: (*i*) node type; (*ii*) operating system; (*iii*) information related to the access to the node.

Virtual Machine Images A job can be received by a Gatekeeper or by a Batch System and sent to a virtual resource (Cloud) or on a standalone Computational node. The aforementioned resources can be of two types: those that require the fulfillment of a need (Gatekeepers and Clusters) and those that satisfy the need (Clouds and Computational nodes). The association between requests to meet and who can satisfy them is performed inside the table *images*.

The possible job flows originated by this schema are four and they are described in Table 1.

Table 1. Possible Job Flows

Component	**Target**
Gatekeeper	Cloud
Gatekeeper	Computational node
Cluster	Cloud
Cluster	Computational node

The single flows will be discussed in the next sections. The table *images* contains the couple of values *Component* from which the job is coming and *Target*, indicating the job flow in the system, the tag that will be published by the Grid Information System in order to notify the presence of the resource, and a series of information related to the possibility of creating multiple instances. In particular, are advised the following information: (*i*) how many instances may be generated (for a single computational node this value is 1); (*ii*) number of jobs per instance; (*iii*) magnitude and boundaries of the instances; (*iv*) waiting time before destroying the images.

4.2 The Needs Schema

In the *needs* schema the software agents running on Clusters and/or Gatekeepers connected to the system, maintain the status of requests to be satisfied. Each agent has associated a table containing the list of jobs with the related *cloudtags*. The job listed in such tables are all waiting jobs. Running jobs are listed in the *running* schema.

4.3 The Fulfillments Schema

In the *fulfillments* schema are instead listed the resources available to satisfy the requests, so that the systems may know for each *cloudtag* where is located the Cloud or the Computational node.

4.4 The Running Schema

We included in the system also the *running* schema having the purpose of storing the state of running resources.

5 Conclusions and Future Work

In the present work, we have figured out different integration strategies which allow a simple interoperability between Batch-oriented and Service-oriented computing models, namely Computational Grids and Cloud Computing. We provided a straightforward implementation of one of the proposed strategies.

The work may be extended in several ways. The DBMS may be removed from the architecture and the system may be re-engineered to be fully distributed adopting for example the protocol 9P described in [25]. Furthermore we can explore innovative approaches for data and big-data management. In this respect, some interesting directions to be taken into consideration are: (*i*) *fragmentation issues* (e.g., [14]); (*ii*) *uncertain data management issues* (e.g., [22]); (*iii*) *general big data management issues* (e.g., [15]).

References

1. Web site of boto library, a python interface to amazon web services. `https://github.com/boto/boto`. Accessed on 2015-09-09.
2. Web site of eucalyptus. `http://www.eucalyptus.com`. Accessed on 2015-09-09.
3. Web site of nimbus project. `http://www.nimbusproject.org/`. Accessed on 2015-09-09.
4. Web site of the amazon elastic compute cloud (ec2):. `http://aws.amazon.com/ec2/`. Accessed on 2015-09-09.
5. Web site of the fog library sdk, an interface to openstack cloud environment:. `https://github.com/fog/fog/blob/master/lib/fog/openstack/docs/getting_started.md`. Accessed on 2015-09-09.
6. Web site of the open cloud computing interface (occi). `http://occi-wg.org/`. Accessed on 2015-09-09.
7. Web site of the open nebula project. `http://opennebula.org/`. Accessed on 2015-09-09.
8. G. B. Barone, R. Bifulco, V. Boccia, D. Bottalico, R. Canonico, and L. Carracciuolo. Gaas: Customized grids in the clouds. In *Euro-Par 2012: Parallel Processing Workshops*, pages 577–586. Springer, 2013.
9. M. Bernaschi, G. Carbone, E. Mastrostefano, and F. Vella. Solutions to the st-connectivity problem using a gpu-based distributed {BFS}. *Journal of Parallel and Distributed Computing*, 76:145 – 153, 2015. Special Issue on Architecture and Algorithms for Irregular Applications.
10. R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1507–1542, 2002.
11. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.
12. J. Castillo, . Bosque, C. Pedraza, E. Castillo, P. Huerta, and J. I. Martinez. Low cost high performance reconfigurable computing. In W. Vanderbauwhede and K. Benkrid, editors, *High-Performance Computing Using FPGAs*, pages 453–479. Springer New York, 2013.
13. R. K. Chellappa. "intermediaries in cloud–computing: A new computing paradigm". *INFORMS Meeting*, 1997.
14. A. Cuzzocrea, J. Darmont, and H. Mahboubi. Fragmenting very large XML data warehouses via k-means clustering algorithm. *IJBIDM*, 4(3/4):301–328, 2009.
15. A. Cuzzocrea, D. Saccà, and J. D. Ullman. Big data: a research agenda. In *17th International Database Engineering & Applications Symposium, IDEAS '13, Barcelona, Spain - October 09 - 11, 2013*, pages 198–203, 2013.
16. A. Dovier, E. P. A. Formisano, and F. Vella. Parallel execution of the ASP computation - an investigation on gpus. In *Proceedings of the Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015), Cork, Ireland, August 31 - September 4, 2015.*, 2015.
17. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid. *Berman et al.[2]*, pages 171–197, 2003.
18. I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

19. O. Gervasi, D. Russo, and F. Vella. The aes implantation based on opencl for multi-/many core architecture. In *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pages 129–134, 2010.
20. B. Klauer. The convey hybrid-core architecture. In W. Vanderbauwhede and K. Benkrid, editors, *High-Performance Computing Using FPGAs*, pages 431–451. Springer New York, 2013.
21. M. M. L. Servoli and R. M. Cefalà. A proposal to dynamically manage virtual environments in heterogeneous batch systems. In *IEEE Nuclear Science Symposium Conference Record, 2008. NSS08*, pages 823–826, 2008.
22. C. K. Leung, A. Cuzzocrea, and F. Jiang. Discovering frequent patterns from uncertain data streams with time-fading and landmark models. *T. Large-Scale Data- and Knowledge-Centered Systems*, 8:174–196, 2013.
23. C. Loomis, M. Airaj, M. E. Bégin, E. Floros, S. Kenny, and D. O'Callaghan. Stratuslab cloud distribution. *European Research Activities in Cloud Computing*, page 271, 2012.
24. Parallella. Parallella 1-x reference manual, 2014. accessed on July 11, 2015.
25. R. Pike, D. Presotto, K. Thompson, H. Trickey, et al. Plan 9 from bell labs. In *Proceedings of the summer 1990 UKUUG Conference*, pages 1–9. London, UK, 1990.
26. E. Ronchieri, D. Cesini, D. D'Agostino, V. Ciaschini, G. D. Torre, P. Cozzi, D. Salomoni, A. Clematis, L. Milanesi, and I. Merelli. The wnodes cloud virtualization framework: a macromolecular surface analysis application case study. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 218–222. IEEE, 2014.
27. A. M. S. Jha and G. Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurr. Comput.: Pract. Exper.*, 21(8):1087–1108, June 2009.
28. B. Sotomayor, K. Keahey, and I. Foster. Combining batch execution and leasing using virtual machines. In *Proceedings of the 17th international symposium on High performance distributed computing*, pages 87–96. ACM, 2008.
29. F. Vella, R. M. Cefala, A. Costantini, O. Gervasi, and C. Tanci. Gpu computing in egi environment using a cloud approach. In *Computational Science and Its Applications (ICCSA), 2011 International Conference on*, pages 150–155. IEEE, 2011.
30. F. Vella, I. Neri, O. Gervasi, and S. Tasso. A simulation framework for scheduling performance evaluation on cpu-gpu heterogeneous system. In *Proceedings of the 12th International Conference on Computational Science and Its Applications - Volume Part IV*, ICCSA 2012, pages 457–469, Berlin, Heidelberg, 2012. Springer-Verlag.