GaaS: Customized Grids in the Clouds

G.B. Barone[1], R. Bifulco[1], V. Boccia[2], D. Bottalico[1], R. Canonico[1], L. Carracciuolo[3]

[1] Università degli Studi di Napoli Federico II
gbbarone@unina.it, roberto.bifulco2@unina.it, davide.bottalico@unina.it, roberto.canonico@unina.it
[2] Italian National Institute of Nuclear Physics, Italy
vania.boccia@na.infn.it
[3] Italian National Research Council, Italy
luisa.carracciuolo@cnr.it

Abstract. Cloud Computing has been widely adopted as a new paradigm for providing resources because of the advantages it brings to both users and providers. Even if it was firstly targeted at enterprises wishing to reduce their equipment management costs, it has been rapidly recognized as both an enabler for new applications and as a mean to allow enterprises of all sizes at running high demanding applications. Recently, Cloud Providers are trying to attract new applications, such as scientific ones, that today already benefit from distributed environment like Grids. This work presents a way to remove the paradigm mismatch between Cloud and Grid Computing, enabling the use of Cloud-provided resources with well-established Grid-like interfaces, avoiding the need for users to learn new resources access and use models. The proposed approach is validated through the development of a prototype implementation and its integration in a working Grid environment.

Keywords: Virtualization, Cloud Computing, Grid Computing

1 Introduction

On-demand computing is a model in which computing resources are made available to users as needed. It could be considered a valid solution for people who need a huge amount of resources, to reduce the *Total Time to Solution*, and cannot bear the costs of systems. In particular, these costs grow up when the needed resources are provided by specialized systems, e.g., HPC ones.

The scientific community developed the Grid Computing paradigm to enable the sharing of huge amount of resources through a well-defined distributed infrastructure model, in order to solve large scale problems in a collaborative manner. The Grid Computing resources aggregation model is rather "static": a group of organizations set up several Grid management services and computing resources in a layered structure that separates the management responsibilities (and corresponding management services) among the organizations involved in the Grid.

Users belonging to the organizations forming the Grid can retrieve information on resources (e.g., their number, status, configuration, etc.) and access them, but can neither change the topology of the grid (e.g., by increasing the number of resources) or manage resources configuration and composition. It would be desirable to have a more *"elastic"* infrastructure in which users can ask for resources on-demand, to suit their needs in terms of resources type and configuration (i.e compilers, scientific libraries, problem solving environments, etc.).

With the advent of new applications and the pervasiveness of IT into everyday activities, also the industrial and private sectors have developed a need for fast access to high demanding IT infrastructures at low costs. The industry answer to these needs has been the Cloud Computing model.

According to the official NIST definition, *"Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources ... that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [10].

Cloud Computing assumes different names depending on the provided resources. When provided resources are computing nodes and storage, Cloud Computing is called Infrastructure as a Service (IaaS). Other resources types include application developing environments (Platform as a Service, PaaS in short) and specific software applications (Software as a Service).

Given the flexibility in resources management through the Cloud Computing paradigm, it seems a promising approach to provide flexible Grid Computing infrastructures through the combination of the Grid and Cloud paradigms.

Some interest has been already shown in this direction [11]. So far, the proposed approaches may be labeled as either "Grid over Cloud" or "Cloud over Grid", since the composition of the two paradigms may be performed through either the exploitation of IaaS-provided resources to build Grid infrastructures or through the use of Grid-provided computing resources to create IaaS clouds.

In this paper we describe our experience in designing and implementing a solution that creates more flexible Grid infrastructures, by exploiting IaaS-provided resources, in a novel way resembling the PaaS paradigm. We call our solution Grid as a Service (GaaS).

The paper is organized as follows: in the next section is presented the Grid reference architecture, in section 3 we provide an overview on the *state of art* about the integration of Cloud and Grid service models, in section 4 we describe the GaaS model and in section 5 is reported a case study related to the deployment of a prototype on the SCoPE Grid Computing Infrastructure[12]. Finally, in section 6 we present future works and conclude.

2 Grid Computing reference architecture

We assume as Grid reference architecture the one implemented by the middleware gLite-EMI, developed in the context of EGI (European Grid Infrastructure). The gLite-EMI middleware provides a Grid infrastructure that is accessible to community members organized into Virtual Organizations (VO). A VO

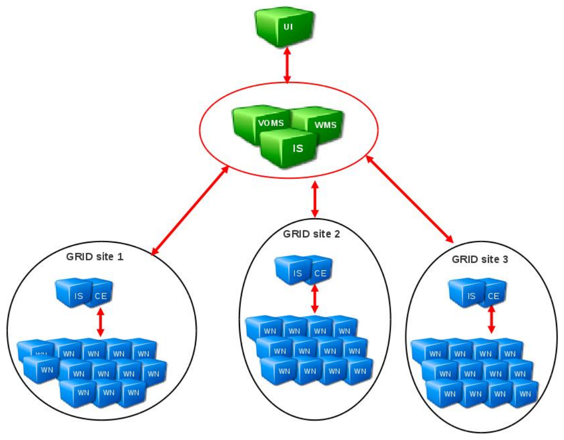


Fig. 1. A gLite-based infrastructure with some central services (UI, VOMS, WMS, IS) and three grid sites (with a CE and several WN).

is defined in [7], as *"a set of individuals and/or institutions defined by such sharing rules..."*. *"VOs vary tremendously in their purpose, scope, size, duration, structure, community and sociology"*. In particular, people from a scientific community, sharing the same "experiment"/applications, can constitute a VO.

VO "managers" make available all the software needed to run the applications of interest of the community on computing resources (the applicative level of the middleware).

The Grid infrastructure is a distributed infrastructure whose management is centralized, while the computation functions are distributed among several sites. The infrastructure provides users with high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the infrastructure itself. Services are embedded into a consistent security framework [8]. Those provided are services for authentication/authorization (e.g. VOMS - Virtual Organization Management System), resources allocation and discovery (e.g. LB/WMS - Logging & Bookiping and Wokload Management System), infrastructure Information System (IS). Computing resources (WNs - Worker Nodes) are provided by means of CE (Computing Element) that is an endpoint with a set of queues handled by an LRMS (Local Resource Management System). User can access these services from a User Interface (UI).

Management services (UI, VOMS, WMS) are instantiated only once and shared among all the sites, while computing-related services (IS, CE) are replicated in each site. A graphical representation of a minimal gLite-based Grid infrastructure is presented in figure 1.

To use the Grid infrastructure, the user has to (1) authenticate himself on the infrastructure; (2) define a *job* in terms of resources requirements and tasks to be performed; (3) submit the job to the infrastructure by selecting the appropriate resource queue; (4) monitor the job status; (5) retrieve the job execution results.

The *resource queue* is an abstraction provided by the Grid architecture to either group resources based on their properties and to share such resources among several users.

The typical Grid usage model described so far does not allow the user in customizing the grid environment. Users cannot change the Grid infrastructure that runs their experiment, in particular, a user cannot create a new Grid site nor add an existing site for his VO. Also the Grid sites are static and cannot be customized by users, hence, it is not possible to add new worker nodes to a site to extend its capabilities, and it is also not allowed to organize resources into customized queues to shape them in accordance to the computation needs. Finally, even the configuration of the worker nodes cannot be changed.

3 Related works

In this section we describe related work about the Cloud and Grid integration, presenting examples of "Cloud-over-Grid" and "Grid-over-Cloud" approaches.

An example implementation of the *"Cloud-over-Grid"* is presented in [15], where CLEVER, a cloud management system, is used to provide an IaaS system over Grid. The solution requires the installation of both a specialized CLEVER's management software and a virtual machine monitor (e.g. VirtualBox) into Grid worker nodes. When the CLEVER cloud requires more resources, new worker nodes can be assigned to it, to dynamically extend the resources available to the cloud.

WNoDeS [14] applies a *"Cloud-over-Grid"* approach as well. WNoDeS (Worker Nodes on Demand Service), developed by the Italian National Institute for Nuclear Physics (INFN), is a solution to virtualize computing resources and to make them available through local, Grid or Cloud interfaces. The Grid infrastructure is exploited through the use of a *"special"* gLite job: the *"power on"*. Users define "Power on" jobs selecting tailored virtual machine images to be lunched on computational resources managed by the CE.

In [5] is described an example of "Grid-over-Cloud", that transparently provides dynamically-instantiated VM-based worker nodes, in an EGEE production grid.

StratusLab [9] is applying a "Grid-over-Cloud" approach as well. The StratusLab project aims at developing a complete, open-source cloud distribution that can be deployed in production in both academic and industrial environments. StratusLab provides Grid services using StratusLabs IaaS system resources. The provided Grid infrastructure can exploit the dynamic nature of the cloud, provisioning resources as needed and running user-level (and community level) services using pre-packaged appliances, selected by users and made available by a *"marketplace"*.

In [4] we presented the design and implementation of an on demand computing service, which is able to obtain a right trade-off among management cost reduction, environmental sustainability and user satisfaction. In particular, the work described an experience in designing and implementing a flexible infrastructure, built on the basis of local or remote cloud resources, with the aim of saving energy to reduce the overall operational cost and to improve environmental sustainability.

Recently, also commercial Cloud Providers are trying to explore the HPC market, by providing resources for, e.g., scientific computations. Most notably, Amazon, is offering "cluster compute" instances, through its Elastic Compute Cloud service, whose resources are tailored for HPC, i.e., they are provided with huge amounts of RAM, processing power, and are deployed on a 10 Gigabit Ethernet network with low delay[1]. To ease the execution of specific workloads, some tools provide automatic configuration of Amazon resources. An example in such sense is CloudFlu[6], that allows the easy execution of OpenFOAM[13] jobs on an automatically configured cluster of Amazon EC2 HPC resources.

4 GaaS: Grid as a Service

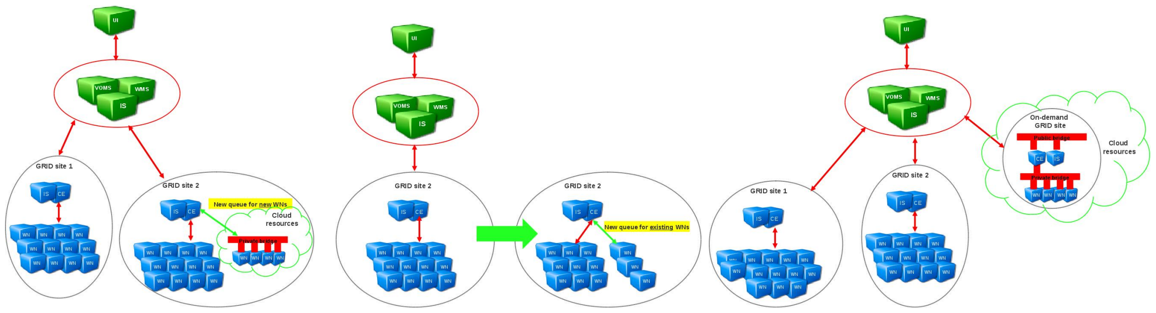
In 2012, the European Middleware Initiative (EMI, http://www.eu-emi.eu) has published a report in which they describe four possible integration scenarios of virtualized infrastructures in the Grid computing architecture [11]. In this paper, we present *Grid-as-a-Service* (GaaS), a service model designed according to the *Dynamic Grid Services* scenario described in that report:

> *[Dynamic Grid Services] utilizes the cloud infrastructure to provision grid services using IaaS/PaaS/SaaS models. The grid services, or suitable subsets of the current grid services, can therefore be instantiated on demand ... by deploying and configuring the services on base virtual machines according to specific user community requirements and then disposed of when not needed anymore.*

The GaaS model combines the advantage of providing users with an usage model that is familiar to the traditional Grid, with the possibility of flexible management of computational resources in a IaaS-like manner. Hence, our model can be classified as a Platform-as-a-Service for extending Grid environments with elastic (e.g., virtual) resources. By using GaaS, "privileged" grid users, e.g. the VO administrator, can define new Grid Sites, add computational resources to existing Grid Sites and modify the resources aggregation scheme, e.g., site queues. In particular, GaaS provides privileged users with the following functions:

1. WNs management (fig. 2.a): definition, addition and deletion of WN to be used by the Grid infrastructure;
2. Queues management (fig. 2.b): dynamic management of resources in queues, and queues policies configuration;
3. Sites management (fig. 2.c): creation and management of new Grid Sites;

GaaS flexibility provides several advantages to traditional Grid infrastructures, e.g., WNs can be customized with software tailored to a given set of users,



(a) *WNs management* (b) *Queues management* (c) *Site management*

Fig. 2. The GaaS solution.

as well as queues can be configured to fulfill a specific computation needs. Moreover, GaaS support the creation of complete Grid sites in order to, e.g., enable a community that has to share resources for the life time of a project, to avoid the burden of configuring from scratch all the required services and resources. Computational resources can be both virtual and physical. The use of virtual resources is not denied to HPC users but, if virtual resources are chosen, users are notified by IS, about the possible performance limitations.

Even if our approach is based on a Grid-over-Cloud model similar to the StratusLab one, in GaaS resources are made available through their configuration in Grid abstractions, e.g., queues. Hence, provided resources can be reconfigured or differently aggregated on the basis of users needs (in a way resembling the PaaS paradigm). Moreover, GaaS enables the provisioning of several high level functions: from queues creation/reconfiguration to the instantiation and configuration of whole grid sites.

5 The SCoPE case study

For the validation of the proposed model we implemented a prototype and integrated it into the context of the S.Co.P.E. Datacenter at University of Naples Federico II, a self contained grid infrastructure that offers storage and computational resources and all the high level core services for infrastructure management (VOMS, WMS, IS, etc.). Moreover S.Co.P.E. resources are integrated also into national IGI and international EGI relevant distributed computational infrastructures and used from people belonging to different scientific reserach fields and to VOs from very rilevant international experiments (e.g. LHC, ATLAS, Super-B, etc.). Thus S.Co.P.E. is a suitable context to validate effectively our approach by means of a prototype.

Our prototype is based on the gLite-EMI [8] Grid middleware and on the OpenNebula [2] cloud management system. The modularity of OpenNebula allows for fast introduction of new features to the management system, hence, it allows the easy integration of the Cloud-provided resources into the Grid infrastructure. A subset of the S.Co.P.E. resources are assigned to the OpenNebula managed resources pool. Such resources host an hypervisor, currently Xen [3], to create virtual machines (VM), that are then used as dynamically provided resources for the Grid infrastructure. VMs are used to both create Grid's WNs and management services such as CE, IS, etc.

The main efforts in the prototype development were (i) the definition of templates for gLite-EMI services configuration, and (ii) the enabling of their fast provisioning.

In many IaaS management system (and in OpenNebula as well), VM templates are usually stored in a "template repository". A new VM is created copying the selected template to the running location of the VM. The duration of this process is the main factor in the resources deployment time. Since the copy process involves the storage infrastructures that are hosting the *template repository* and the newly created VMs disks, assuming that a minimal VM template is several hundreds of megabytes big, the process, for each VM creation, takes a time in the order of dozens of seconds.

To optimize the infrastructural resources used during the provisioning process, and to reduce the overall provisioning time, we took into account the peculiarities of the GaaS system. In particular, we made the following observations:

1. all the VMs are prescribed to host the same operating system, which is imposed by the gLite-EMI middleware;
2. all the VMs hosting the Grid services (e.g., WN, CE, etc.) can be produced by customizing the configuration of a single VM template;

We designed our VM disk provisioning system in order to provide fast VM creation and avoid as much data copy as possible. Our solution is based on the GNU/Linux's Logical Volume Manager (LVM). LVM allows the creation of logical volumes (LV) and the creation of snapshots starting from a reference LV. Snapshots can be read and written, since their creation is performed through the use of a "*delta meta-data*", that contains all the differences with the original LV. This approach makes the creation of a snapshot really fast (a few milliseconds) since it involves no copy of data. Once the snapshot is created, following observation 2, a configuration script is executed to customize the virtual resource according to its functional destination. Since both read and write actions involve an a read/update of the *delta meta-data*, the operating performance of the snapshot could be compromised in particular conditions. In our case, we are mainly interested into reading performance, and, moreover, we assume that the majority of reads happen in the bootstrapping process of a VM (e.g., for the loading of the required applications).

In figure 3 is presented a performance comparison of read and write operations on 64 KBs data blocks. These tests were executed on an HP DL380 Proliant server equipped with two Intel Pentium IV Xeon 2.8 GHz CPUs, 5 GB of PC-2100 RAM. The server was running a Debian Linux with the OS kernel configured to use only 1 GB of RAM. We compare the results obtained by operating on both a raw partition (labeled as "normal") and on a LVM snapshot (labeled as "snapshot"). For various amounts of read/written data (ranging from 512 bytes up to 8 GBytes), we compare the throughput achieved for write (left graphs) and read (right graphs) operations. Figure 3 shows the results of six series of experiments. On the left hand of the figure, graphs a), c) and e) present the throughput obtained by write operations performed in the following cases: graph a) refers to operations performed on freshly mounted disks, with no OS caching effects, graph c) refers to sequential write operations performed several times on the same data, in order to maximize the OS caching effects, graph e) refers to random write operations. On the right hand of the figure, graphs b), d) and f) present the results for similar experiments involving read operations.

From the graphs it can be easily observed the effect of caches on performance, when data size is bigger than the available filesystem cache (our system had about 512 MBs of cache space). What is of particular interest for us, is that the performance drop when snapshots are used is marginal, since the *delta meta-data* is likely stored in the system cache anyway, hence, we can fast provision resources to the grid, without paying any sensible penalty on disk performance. Moreover, performance drops are visible when the written/read data size is bigger than the filesystem cache, in particular for write operations. Since snapshots are just used to create VMs' OS bootable disks, i.e., disks that contains the OS and the applications code but that are not meant to be used as data storage, they are mainly involved in read operations, and the dimension of read data is likely to be smaller or comparable with the dimension of the filesystem cache. Hence, we expect little or no performance drop in the VM operations.

6 Conclusion and Future Work

In this paper we presented GaaS, a PaaS model for Grid Computing systems, that lets VO administrartors to dynamically customize the grid environment they are offering to VO's unprivileged members. VO administrators can define new Grid Sites, add computational resources to Grid Sites and modify the resources aggregation scheme (queues). We implemented a prototype of our model and deployed it in a real-world Grid Datacenter. Moreover, our prototype implements a virtual resources fast provisioning scheme, that exploits some properties of the Grid environment. Our implementation uses standard GNU/Linux tools, i.e., LVM, and a careful definition of easily customizable virtual resource templates.

Even if the presented work is a successful proof-of-concept, many issues still have to be solved. In particular we have to assess the applicability of virtualized resources in HPC contexts, the payed overhead, and the possibility to extend the model to a mix of virtualized and physical resources according to the users needs.

We are working on solutions able to allow new communities wishing to use the grid to instantiate new grid infrastructure also for the non existing VOs. Moreover, we are also planning an evaluation of the impact on management operations and costs of our approach, in order to integrate a smart management

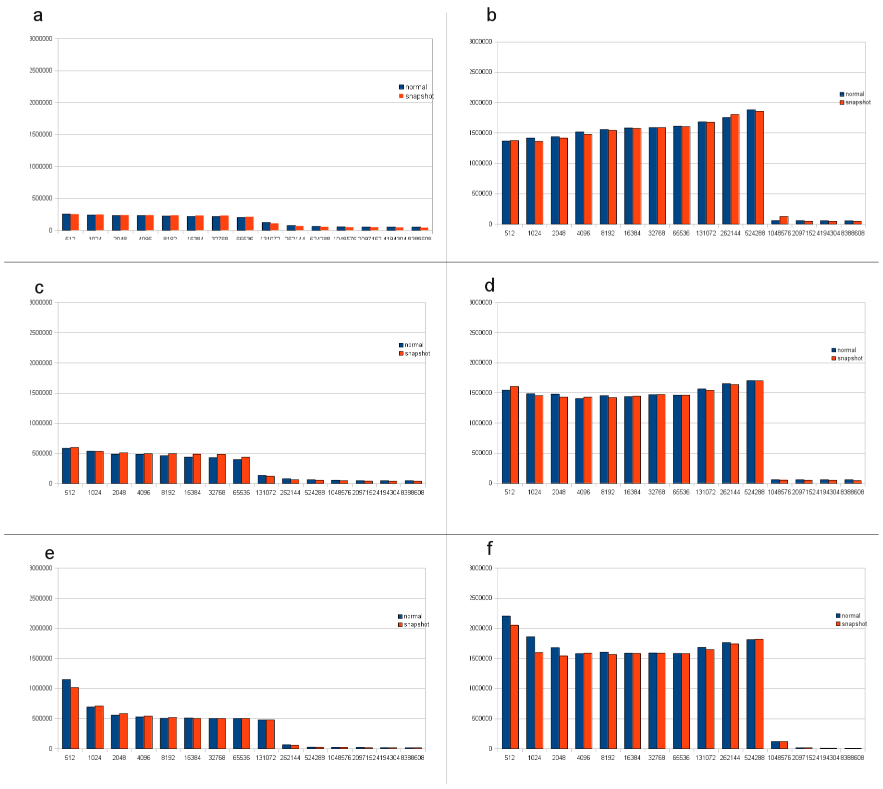


Fig. 3. Storage write (left column) and read (right column) performance. The y axis shows the read and write throughput (bytes/sec), while the x axis shows the amount of data read/written. Graphs a, b refer to sequential operations with no OS caching effects. Graphs c, d refer to sequential repeated operations to maximize OS caching effects. Graphs e, f refer to random operations.

of resources with the aim of providing energy savings, in a Green Computing perspective.

Acknowledgments. This work is part of the activities of a multidisciplinary group (GTT) that is responsible for the S.Co.P.E. infrastructure management. The work is also part of the activities carried out by GTT in the context of the Italian Grid Infrastructure (IGI).

References

1. Amazon: High Performance Computinig (HPC) on AWS, `http://aws.amazon.com/hpc-applications/`

2. Andic, M., Dejan, Llorente, I.M., Montero, R.S.: Opennebula: A cloud management tool. Internet Computing, IEEE 15(2), 11 –14 (march-april 2011)
3. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: ACM SIGOPS Operating Systems Review. vol. 37, pp. 164–177. ACM New York, NY, USA (2003)
4. Barone, G., Bifulco, R., Boccia, V., Bottalico, D., Carracciuolo, L.: Toward a flexible, environmentally conscious, on demand high performance computing service. In: Data Compression, Communications and Processing (CCP), 2011 First International Conference on. pp. 136 –138 (june 2011)
5. Childs, S., Coghlan, B., McCandless, J.: Dynamic virtual worker nodes in a production grid. In: Min, G., Di Martino, B., Yang, L., Guo, M., Runger, G. (eds.) Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops, Lecture Notes in Computer Science, vol. 4331, pp. 417–426. Springer Berlin/Heidelberg (2006), http://dx.doi.org/10.1007/11942634_44
6. CloudFlu: CloudFlu - HPC cloud computing for OpenFOAM (R) users, http://sourceforge.net/projects/cloudflu/
7. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15(3), 200–222 (Aug 2001), http://dx.doi.org/10.1177/109434200101500302
8. Laure, E., et al.: Programming the Grid with gLite. Computational Methods in Science and Technology 12(1), 33–45 (2006)
9. Loomis, C., Airaj, M., Bégin, M., Floros, E., Kenny, S., O'Callaghan, D.: StratusLab Cloud Distribution. In: Petcu, D., Vázquez-Poletti, J. (eds.) European Research Activities in Cloud Computing, pp. 260–282. Cambridge Scholars Publishing (2012)
10. Mell, P., Grance, T.: The NIST definition of Cloud Computing
11. Memon, M., Nagy, Z., Yen, E., Koeroo, O.: Virtualization and Cloud Computing Task Force Report V.0.7 (2012), http://cdsweb.cern.ch/record/1359910/files/EMIVirtCloudReport-v0.7.doc
12. Merola, L.: The S.Co.P.E. Project. In: Proceedings of the Final Workshop of Grid Projects of the Italian National Operational Programme 2000-2006 Call 1575. pp. 18–35. Consorzio COMETA (2009)
13. OpenFOAM-Foundation: OpenFOAM, http://www.openfoam.com/
14. Salomoni, D., Italiano, A., Ronchieri, A.: WNoDeS, a tool for integrated Grid and Cloud access and computing farm virtualization. In: Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP 2010). pp. 18–35 (2011)
15. Tusa, F., Paone, M., Villari, M., Puliafito, A.: Clever: A cloud cross-computing platform leveraging grid resources. In: UCC. pp. 390–396. IEEE Computer Society (2011), http://dblp.uni-trier.de/db/conf/ucc/ucc2011.html#TusaPVP11