Execution of Scientific Workflows on Federated Multi-cloud Infrastructures

Daniele Lezzi[1], Francesc Lordan[1], Roger Rafanell[1], and Rosa M. Badia[1,2]

[1] Barcelona Supercomputing Center - Centro Nacional de Supercomputación (BSC-CNS)
{daniele.lezzi,francesc.lordan,roger.rafanell,rosa.m.badia}@bsc.es
[2] Artificial Intelligence Research Institute (IIIA), Spanish Council for Scientific Research (CSIC)

Abstract. Recently cloud services have been evaluated by scientific communities as a viable solution to satisfy their computing needs, reducing the cost of ownership and operation to the minimum. The analysis on the adoption of the cloud computing model for eScience has identified several areas of improvement as federation management and interoperability between providers. Portability between cloud vendors is a widely recognized feature to avoid the risk of lock-in of users in proprietary systems, a stopper to the complete adoption of clouds.

In this paper we present a programming framework that allows the coordination of applications on federated clouds used to provide flexibility to traditional research infrastructures as clusters and grids. This approach allows researchers to program applications abstracting the underlying infrastructure and providing scaling and elasticity features through the dynamic provision of virtualized resources. The adoption of standard interfaces is a basic feature in the development of connectors for different middlewares ensuring the portability of the code between different providers.

Keywords: Programming models, Cloud interoperability, Cloud federation, Standards.

1 Introduction

The revolutionary advent of Cloud computing has changed the way IT services are developed and offered. Enterprises have switched their IT organization moving from on premises owned physical infrastructures to rented on demand services. This new model has implied a shift from capital expense avoiding up-front investments in the procurement of computational infrastructures in favor of a renting model where resources are acquired, by Amazon EC2 [13], Microsoft Azure [7] or Google Apps [6], on a pay per use basis. Similarly, Cloud computing has revealed useful and convenient in many scientific research fields allowing to have access to almost unlimited resources without the need of taking care of the management and deployment of the underlying infrastructure. In the last years, several research projects [12], [10], [4] have analyzed how to face these challenges

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 136–145, 2014.

exploring the adaptability of virtualization and Cloud computing to research. In particular the European Grid Infrastructure (EGI) [3] has started a specific Federated Cloud Task Force [5] in order to evaluate the adoption of clouds to make the existing infrastructure more flexible and scalable to expand its services to new user communities. An hybrid model for eScience is being gradually adopted with cloud infrastructures, both private and public, offered as complementary solutions to traditional environments as grids and clusters to provide flexibility and cost-effectiveness to the so called long-tail of science. Nevertheless cloud adoption is not so straightforward for non-technical users that have to deal with issues related to the adaptation of their applications to new environments and with the lack of guaranteed portability and interoperability between different providers.

In order to facilitate the execution of applications in the cloud, tools that simplify their development, deployment and transparent execution on federated infrastructures are required. In this paper we present the enhancements to the COMP Superscalar programming framework (COMPSs) [29] to make it interoperable with the EGI Federated Cloud testbed in order to optimize the execution of the related use cases. COMPSs allows the automatic parallelization of sequential applications on top of computational infrastructures as grids and clouds providing elasticity features. In particular, here we focus on the development of a connector for the management of virtual appliances using the Open Cloud Computing Interface (OCCI) [8] specification and on the evaluation of the framework executing a workflow for the biodiversity community in EGI.

The rest of the paper is organized as follows. Section 2 describes the related work and Section 3 introduces COMPSs and how it enables interoperability with several clouds offerings. In particular Section 3.2 presents the main contribution of the paper, the developments to make COMPSs interoperable with the EGI Federated Cloud. Section 4 evaluates the execution of a use case highlighting the benefits of using COMPSs in a federated multi-cloud environment. Section 5 concludes the paper and presents the future work.

2 Related Work

Several frameworks allow the graphical composition of an application workflow and its execution on distributed infrastructures. Taverna [23] is a workflow language and computational model designed to support the automation of complex, service-based and data-intensive processes. A prototype version has been tested on a private cloud testbed deploying Taverna Servers on a set of virtual machines but neither plugins are offered to execute tasks on public clouds nor the possibility exists of elastically modify the pool of resources. In [15] authors evaluate the usage of Pegasus workflow manager on multiple clouds. The setup of the execution is not straightforward because forces the user to manually predeploy several nodes configured as Condor workers, while COMPSs automatically deploys the applications on dynamically created virtual machines. The gUSE framework [18] includes a workflow manager and a distributed computing

infrastructure (DCI) bridge that provides standard access, through a Basic Execution Service (BES) [17] compliant interface, to various DCIs including clouds. COMPSs implements a similar approach [20] through a deployment and execution service accessible by compliant clients. In gUSE the support to clouds is delegated to an external service and is still in development phase in the context of the SCI-BUS project [25]. The Contrail project is developing ConPaaS [24], a runtime environment for elastically hosting and deploying applications on federated and multi-clouds following SLA constraints. Applications in ConPaaS are composed of any number of services programmed through a common generic Python interface for the service management and a Javascript part to extend the front-end GUI with service-specific information and control. The main difference with COMPSs is the existence of predefined patterns for the implementation of a service, thus forcing the users to write new code or to adapt the existing one to use the currently provided services. Support to OCCI specification is not available in ConPaaS at writing time.

3 COMPSs Interoperability in the EGI Federated Cloud

3.1 The EGI Federation Model

The federation model of the EGI Cloud Infrastructure Platform provides an abstract cloud management stack to operate an infrastructure integrated with the components of EGI Core Infrastructure Platform. This model defines the required external interfaces and corresponding interaction ports that each resource provider has to support. The former requires the deployment of interoperable standard interfaces for VM management (OCCI), data management (CDMI [22]) and information discovery (GLUE 2). The interaction with the core EGI components includes the integration with the AAI VOMS (Virtual Organization Membership Service) [14] system and with the monitoring and accounting services. An Appliance Repository and a VM Marketplace support the sharing and deployment of appliances across the providers. The EGI Cloud Testbed includes different management middlewares (OpenStack [27], OpenNebula [28], StratusLab, WNoDeS [26] and Synnefo [11]) but the federation model does not impose any specific technology leaving to the providers the responsibility to identify and deploy the solution that fits best their individual needs with the only constraint of providing the appropriate interfaces in order to be interoperable with other implementations.

3.2 COMPSs Integration in the EGI Cloud

COMPSs is a framework, composed of a programming model and a runtime, that aims to ease the development and deployment of distributed applications and web services. The core of the framework is its programming model which allows the programmer to write applications in a sequential way and to execute them on top of heterogeneous infrastructures exploiting the inherent parallelism

of the application. The COMPSs programming model is task-based allowing the programmer to select the methods of the sequential application to be executed remotely. This selection is done by means of an annotated interface where all the methods that have to be considered as a task are defined with some annotations describing their data accesses and constraints on the execution resources. At execution time this information is used by the runtime to build a dependency graph and orchestrate the tasks on the available resources. One important feature of the COMPSs runtime is the ability to exploit the cloud elasticity by adjusting the amount of resources to the current workload. When the number of tasks is higher than the available cores, the runtime turns to the cloud looking for the provider that offers the resources that better fit the requirements of the application with a lower economical cost. Symmetrically, when the runtime detects an excess of resources it powers off unused instances in a cost-efficient way. Such decisions are based on information provided by the user who describes the details of the software images and instance templates available for every cloud provider. Since each cloud provider offers its own API, COMPSs defines a generic interface to manage resources and to query details about the execution cost of multiple cloud providers during the same execution. The implementations of this interface, called *connectors*, are in charge of translating the generic requests to the actual provider's API.

In order to support the usage of EGI Federated Cloud IaaS resources, a new connector has been developed to operate with the interfaces and protocols described in subsection 3.1. The support for the VM management is implemented using the rOCCI client [9] that transparently interoperates with the rOCCI servers deployed in the testbed on top of different middlewares as OpenNebula and OpenStack. The connector supports different authentication methods, including X509 type through VOMS proxy certificates. To setup each connector instance, the user indicates which virtual images and instance types are offered by the specific provider; thus, when the runtime asks for the creation of a VM, the connector selects the appropriate image and resource template according to the requirements (in terms of cpu, memory, disk, etc) and invokes the rOCCI client through *Mixins*, extensions of the OCCI Model that allow the instantiation of a VM through templates with additional capabilities (network, storage, etc). For the data management, the support to CDMI protocol is offered through the VENUS-C middleware to store and retrieve input data and final results, as detailed in [17]. The interaction with the deployed resources is done via SSH. To allow the job submission and data transfers between worker VMs, public key exchange is used as authentication method. At boot time, the VM contacts the Perun [19] server to obtain the public key of the user. Once the VM is accessible, the connector contextualizes it configuring the SSH keys and deploying the software packages such as the binaries used by the tasks, the application code and the COMPSs worker application. The connector also implements methods for the computation of the economical and temporal costs on each provider such as the accumulated execution cost, the current cost per hour, the cost per hour

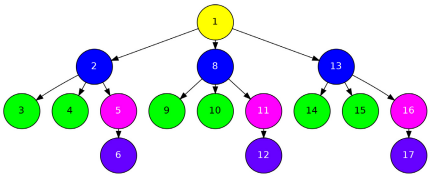
for a VM that fulfills certain requirements or the time required to start a new instance.

4 Evaluation

A series of tests has been performed in order to evaluate the performance of COMPSs and the developed connector to run a use case provided by an EGI user community. In the following section we describe the implementation of the application, the deployed testbed, the benchmarks and the results obtained.

4.1 Modeller Service

The *Modeller* Web Service provides a COMPSs implementation of the Extended Open Modeler Web Service (OMWS+) [21] which performs Ecological Niche Modelling (ENM) operations automatically converting multi-stage & multi-parameter experiment requests into a set of single invocations of the openModeller (OM) [16] suite. The Serivce provides operations that execute a COMPSs workflow, whose example dependency graph is depicted in Figure 1, composed of OM operations that *model* a species distribution, *test* it agains an input dataset, *project* the distribution on a geographical map and print it on an image file (*translate*). In the figure the numbers represent the generation order of the tasks by the COMPSs runtime and each type of task is depicted with a color.



Type	Tasks
Convert	1
Model	2 8 13
Test	3 4 9 10 14 15
Project	5 11 16
Translate	6 12 17

Fig. 1. Dependency graph for an execution of COMPSs workflow with 1 species, 3 modelling algorithms and multiple tests. COMPSs generates three model tasks, two tests operations and three project tasks, one per model.

4.2 Testbed

The Web Service has been deployed on the federated cloud scenario depicted in Figure 2 and composed of the following elements:

- ***Client***: Java application that invokes the Modeller Service.
- ***WS server***: machine running an Apache Tomcat 7.0 [2] WS container hosting the service. It is a Intel Core 2 Quad Q9300 at 2.5 GHz, 4 GB of RAM and 320 GB of disk space. Both, the service main program and the COMPSs runtime execute in this machine located at BSC premises in Barcelona, Spain.

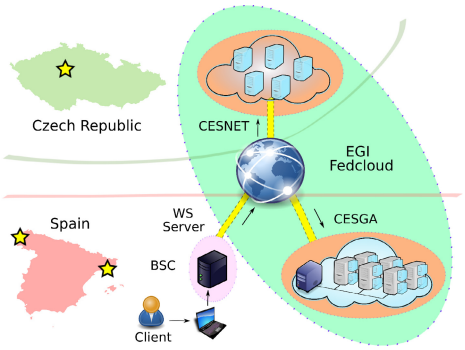


Fig. 2. The testbed comprises two EGI FedCloud providers, CESNET and CESGA, providing VMs to the Modeller Service running at BSC

– ***EGI Federated Cloud***: composed of two providers, CESGA (Spain) and CESNET (Czech Republic) both providing resources through the rOCCI server for OpenNebula cloud middleware. CESGA contributed 33 octo-core servers (256 CPUs) and 14.5 TB of shared NFS/SSH filesystem, CESNET contributed ten 24-core servers, 96 GB of memory and 44 TB of shared filesystem. The providers offered different VM plans; CESGA provided *Large* VMs template consisting of 2 cores and 8 GB of memory. CESNET provided an *XLarge* template which consists of 4 cores and 15 GB of memory. The VMs images, stored in the appliance repository, take up 10 GB and were configured with a Linux distribution, the COMPSs runtime, the Modeller service packages and the OM suite.

4.3 Results

Two different usage modes of the implemented use case are discussed below, representing two ways of usage of the ENM service. In the *Single* test a complex workflow is submitted in a unique request, while in the *Multiple* test, simpler workflows are submitted to the service. When the ENM service starts, it automatically instances one *Large* VM at CESGA and two *XLarge* VMs at CESNET that will be always available throughout different executions. This strategy reduces the initial response time because the first requests can be immediately submitted without waiting for the instantiation of new VMs. The number of pre-deployed instances can be configured in the connector.

Single Request. In order to evaluate the performance of the runtime and how it adapts the VMs pool size to the workload, a single invocation request has been issued. This request includes 6 species and 2 algorithms, thus producing 12 distribution models; each model is then tested and projected generating a total of 36 dependent tasks. This test exploits different VM templates offered by the EGI FedCloud providers by using method constraints defined in the COMPSs

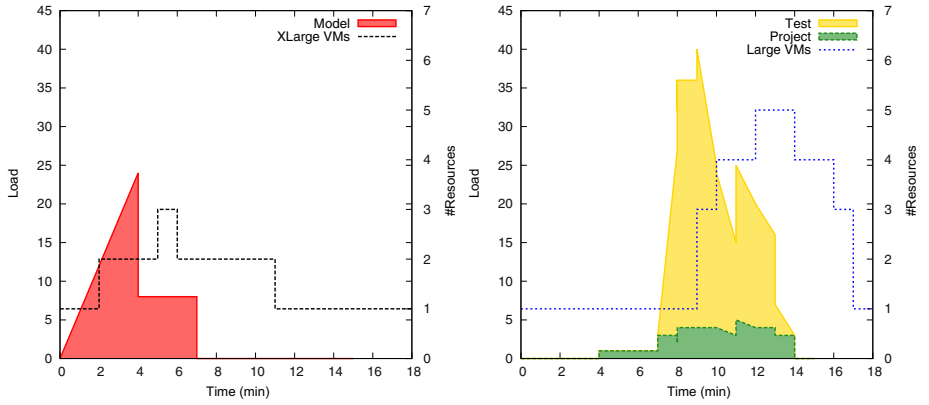


Fig. 3. System load *vs.* available virtual resources during a single request execution. Figure 3(a) and Figure 3(b) represent the *model* and the *test, project* tasks respectively.

application interface. This forces the *model* and *test, project* tasks to be executed on *XLarge* and *Large* VMs respectively. Figure 3 describes the runtime behavior during the execution of a single request; a growth of the number of tasks forces the runtime to acquire new resources while, as the load decreases, virtual resources are progressively released adapting the system needs to the current workload. In the figures, VMs are available after a creation time specific of each provider and released at the end of the tasks currently executed there.

Multiple Requests. This test aims to evaluate the global service performance on a multi-request scenario. In this case, a set of invocations has been issued following a Gaussian random workload pattern, generated by Apache JMeter [1] benchmark suite. The JMeter test plan has been tuned to issue 30 simultaneous

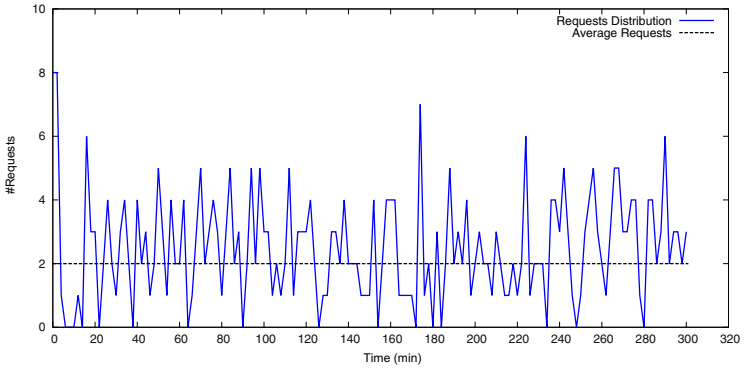


Fig. 4. Requests per minute. The average cadence is set on 2.41 requests/min.

requests in a random time frame of 0/2 minutes during 5 hours as depicted in Figure 4. This suite also permits to obtain and analyze some service Key Performance Indicators (KPI) such as: response time, dynamic resource consumption and throughput. In this experiment, each request requires low computation time, generating a dependency graph composed of three OM tasks (*model*, *test* and *project*) where the last two ones executed in parallel.

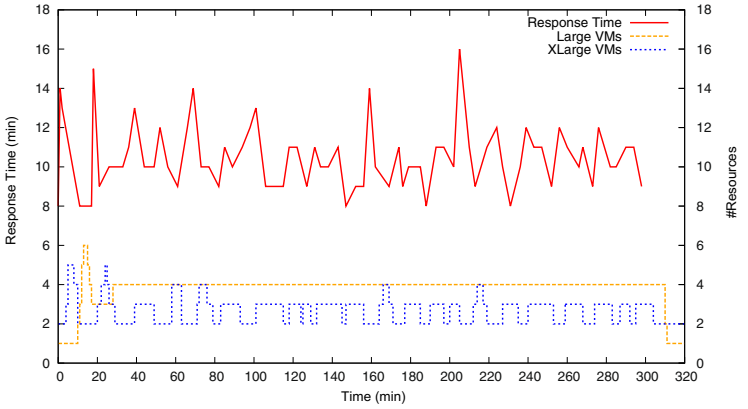


Fig. 5. Response time *vs.* resource consumption

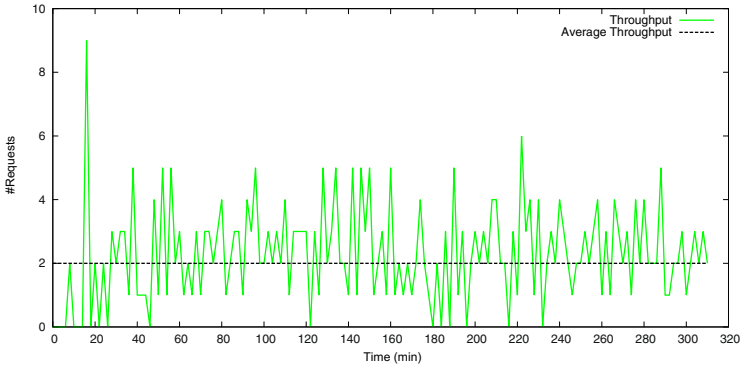


Fig. 6. Responses per minute (throughput). The sustained value is 2.34 responses/min.

The response time is depicted in Figure 5 together with the evolution of the number of virtual resources used in the execution. When multiple requests are issued at the same time (Figure 4) the response time increases and the COMPSs runtime reacts to the rise of the load produced, adapting the number of resources needed to execute every kind of new task, according to the task constraint. As represented in *#Resources* curve, the connector manages *XLarge* type machines at CESNET and *Large* type ones at CESGA.

Another important performance indicator to consider is the global throughput of the service. Figure 6 shows a slightly lower sustained value (2.9%) than the

average requests per minute, thanks to the accuracy on the resources management system. This evidences a fast responsiveness of COMPSs framework facing unexpected workload peaks.

5 Conclusions and Future Work

This paper presented the extension of the COMPSs framework to execute applications in the EGI Federated Cloud. The developed OCCI compliant connector allows the orchestration of COMPSs applications on hybrid environments elastically adapting the number of resources to the actual load requirements. The application used in the evaluation, published as virtual appliance in the EGI Marketplace, has been executed in order to evaluate two possible scenarios. The results of the tests demonstrate that the runtime is able to deal with variations of loads of complex workflows by provisioning virtual instances on demand and also to serve multiple independent execution requests properly managing the pool of resources.

The ongoing work includes the improvement of the connector to support contextualization features that will be introduced in the EGI FedCloud in the production phase of the testbed and the enhancement of resource management in order to support the assignment of tasks to multi-core instances.

Acknowledgements. This work has been supported through the grants: SEV-2011-00067 of Severo Ochoa Program, awarded by the Spanish Government and the Spanish Ministry of Science and Innovation under contract TIN2012-34557, the Generalitat de Catalunya (contract 2009-SGR-980) and the European Commission (EUBrazilOpenbio: RI-288754 and EGI-InSPIRE: RI- 261323).

References

1. Apache jmeter, `http://jmeter.apache.org`
2. Apache Tomcat, `http://tomcat.apache.org/`
3. Egi-inspire white paper, `http://go.egi.eu/pdnon` (last visited on April 16, 2013)
4. European Middleware Initiative (EMI), `http://www.eu-emi.eu/`
5. Federated Clouds Task Force, `https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce` (last visited on June 4, 2013)
6. Google App Engine, `http://code.google.com/appengine`
7. Microsoft Azure, `http://www.microsoft.com/azure`
8. Open Cloud Computing Interface, `http://occi-wg.org` (last visited on April 16, 2013)
9. The rocci framework, `http://dev.opennebula.org/projects/ogf-occi/wiki` (last visited on July 17, 2013)
10. Stratuslab, `http://stratuslab.eu/`
11. Synnefo cloud, `http://www.synnefo.org/` (last visited on July 17, 2013)
12. Virtual multidisciplinary ENvironments USing Cloud infrastructures Project, `http://www.venus-c.eu`

13. Amazon elastic compute cloud, Amazon EC2 (2008)
14. Alfieri, R., Cecchini, R.L., Ciaschini, V., dell'Agnello, L., Frohner, A., Gianoli, A., Lõrentey, K., Spataro, F.: VOMS, an authorization system for virtual organizations. In: Fernández Rivera, F. (ed.) Across Grids 2003. LNCS, vol. 2970, pp. 33–40. Springer, Heidelberg (2004)
15. Berriman, G.B., Deelman, E., Juve, G., Rynge, M., Vöckler, J.-S.: The application of cloud computing to scientific workflows: a study of cost and performance. Physical and Engineering Sciences 371 (1983) (January 1983)
16. de Souza Muñoz, M.E., De Giovanni, R., de Siqueira, M.F., Sutton, T., Brewer, P., Pereira, R.S., Canhos, D.A.L., Canhos, V.P.: openmodeller: a generic approach to species potential distribution modelling. GeoInformatica 15(1), 111–135 (2011)
17. Foster, I.: OGSA Basic Execution Service Version 1.0. Grid Forum Document GFD-RP. 108. (November 13, 2008), http://www.ogf.org/documents/GFD.108.pdf
18. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., Marton, I.: Ws-pgrade/guse generic dci gateway framework for a large variety of user communities. Journal of Grid Computing 10, 601–630 (2012)
19. Krenek, A., Sebestianov, Z.: Perun – fault-tolerant management of grid resources. In: Krakow Grid Workshop 2004, Krakow, pp. 133–140, Academic Computer Centre CYFRONET AGH (2004)
20. Lezzi, D., Memon, S., Rafanell, R., Soncu, H., Riedel, M., Badia, R.M.: Interoperable execution of escience applications on grids & clouds through open standards. In: Proceedings of the Unicore Summit 2012, IAS, Forschungszentrum Jülich GmbH (2012)
21. Lezzi, D., Rafanell, R., Torres, E., De Giovanni, R., Blanquer, I., Badia, R.M.: Programming ecological niche modeling workflows in the cloud. In: The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA 2013), Barcelona, Spain (March 2013)
22. Livenson, I., Laure, E.: Towards transparent integration of heterogeneous cloud storage platforms. In: Proceedings of the Fourth International Workshop on Data-intensive Distributed Computing, DIDC 2011, pp. 27–34. ACM, New York (2011)
23. Missier, P., et al.: Taverna, reloaded. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 471–481. Springer, Heidelberg (2010), http://www.taverna.org.uk/pages/wp-content/uploads/2010/04/T2Architecture.pdf
24. Pierre, G., Stratan, C.: ConPaaS: a platform for hosting elastic cloud applications. IEEE Internet Computing 16(5), 88–92 (2012)
25. SCI-BUS project: The sci-bus project (2011), http://www.sci-bus.eu/ (last visited on May 31, 2013)
26. Salomoni, D., Italiano, A., Ronchieri, E.: Wnodes, a tool for integrated grid and cloud access and computing farm virtualization. Journal of Physics: Conference Series 331(5), 052017 (2011)
27. Sefraoui, O., Aissaoui, M., Eleuldj, M.: Article: Openstack: Toward an open-source solution for cloud computing. International Journal of Computer Applications 55(3), 38–42 (2012)
28. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing 13, 14–22 (2009)
29. Tejedor, E., Badia, R.: Comp superscalar: Bringing grid superscalar and gcm together. In: 8th IEEE International Symposium on Cluster Computing and the Grid, CCGRID 2008, pp. 185–193. IEEE (2008)