Are Public Clouds Elastic Enough for Scientific Computing?

Guilherme Galante[1], Luis Carlos Erpen De Bona[1], Antonio Roberto Mury[2], and Bruno Schulze[2]

[1] Department of Informatics – Federal University of Paraná - UFPR
Curitiba, PR – Brazil
{ggalante,bona}@inf.ufpr
[2] National Laboratory for Scientific Computing - LNCC
Petrpolis, RJ – Brazil
{aroberto,schulze}@lncc.br

Abstract. Elasticity can be seen as the ability of a system to increase or decrease the computing resources allocated in a dynamic and on demand way. It is an important feature provided by cloud computing, that has been widely used in web applications and is also gaining attention in the scientific community. Considering the possibilities of using elasticity in this context, a question arises: "Are the available public cloud solutions suitable to support elastic scientific applications?" To answer the question, we present a review of some solutions proposed by public cloud providers and point the open issues and challenges in providing elasticity for scientific applications. We also present some initiatives that are being developed in order to overcome the current problems. In our opinion, current computational clouds have not yet reached the necessary maturity level to meet all scientific applications elasticity requirements.

Keywords: Cloud computing, elasticity, scientific applications.

1 Introduction

Recently, cloud computing has emerged as an alternative for solving scientific computing problems, with the promise of provisioning virtually infinite resources. According to Simmhan et al. [1], the use of cloud computing environment can be attractive to the scientific community in many ways, benefiting users that own small applications, but also those who perform their experiments in supercomputing centers. In fact, several authors in the technical literature share this opinion and present advantages and benefits of using cloud computing to perform scientific experiments [2].

Cloud computing offers to end users a variety of resources from the hardware to the application level, by charging them on a pay-per-use basis, allowing immediate access to required resources without the need to purchase additional infrastructure. In addition, an important characteristic, not available on traditional architectures (e. g., clusters and grids), emerged on cloud computing: *elasticity*. Elasticity can

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 294–307, 2014.

be seen as the ability of a system to increase or decrease the computing resources allocated in a dynamic and on demand way. Ideally, to the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time [3]. There are two elasticity types: vertical and horizontal. In vertical elasticity resources, such as processing, memory and storage resources can be added/removed from a running virtual instance. Horizontal elasticity is the ability of the cloud to vary the number of VM's allocated to a service according to demand.

Traditionally, cloud elasticity has been used for scaling traditional web applications in order to handle unpredictable workloads, and enabling companies to avoid the downfalls involved with the fixed provisioning (over and under-provisioning) [4]. In scientific scenario, the use of cloud computing is discussed in several studies [5][6], but the use of elasticity in scientific applications is a subject that is starting to receive attention from research groups [7].

This interest is related to the benefits it can provide, including, improvements in applications performance and cost reduction. Improvements in the performance of applications can be achieved through dynamic allocation of additional processing, memory, network and storage resources. Examples are the addition of nodes in a master-slave application in order to reduce the execution time, and the dynamic storage space allocation when data exceeds the capacity allocated for the hosted environment in the cloud.

The cost reduction is relevant when using resources from public clouds, since resources could be allocated on demand, instead allocating all of them at the beginning of execution, avoiding over-provisioning. It could be used, for example, in applications that use the MapReduce paradigm, where is possible to increase the number of working nodes during the mapping and to scale back resources during the reduction phase. Elastic applications can also increase computational capabilities when cheaper resources became available. An example is the allocation of Amazon Spot Instances, when the price becomes advantageous [8].

Thus, considering the possibilities of using elasticity in the scientific context, a question arises: *are the available public cloud solutions suitable to support elastic scientific applications?* To answer this question, this paper presents a survey covering the elasticity mechanisms offered by major public cloud providers and analyses the limitations of the solutions in providing elasticity for scientific applications. We also present some initiatives that are being developed in order to overcome the current challenges.

1.1 Public Cloud Elasticity Mechanisms

In this section we present nine elasticity solutions proposed by major IaaS and PaaS public providers. In general, most public cloud providers offer some elasticity feature, from the most basic, to more elaborate automatic solutions.

Amazon Web Services [9], offers a mechanism called Auto-Scaling, as part of the EC2 service. The solution is based on the concept of Auto Scaling Group (ASG), which consists of a set of instances that can be used for an application.

Amazon Auto-Scaling uses an reactive approach, in which, for each ASG there is a set of rules that defines the number of VM's to be added or released. The metric values are provided by CloudWatch monitoring service, and include CPU usage, network traffic, disk reads and writes. The solution also includes an API and a command-line interface for manually access the scaling features.

Rackspace [10] also implements horizontal elasticity, but unlike Amazon, it does not have a native automatic elasticity service. The provider offers an interface and an API to control the amount of resources allocated, leaving to the user the implementation of more elaborate automated mechanisms.

Similarly, GoGrid [11] has not built in elasticity capabilities, although it provides an API for remote control of the hosted virtual machines (VM's). Thus, the user is responsible for monitoring the service and taking the scaling decisions. The creation and removal of resources is done through calls to the API. Besides VM's replication, GoGrid also support vertical elasticity for memory.

The solution provided by Joyent [12] is also based VM replication accessed via API. However, the provider include an automatic feature called CPU bursting, which temporarily increase the CPU capability of up to 400% in order to handle workload spikes.

A more comprehensive elasticity solution is provided by Profitbriks [13]. According to its documentation, it is possible to use horizontal and vertical elasticity, allowing changes in virtual environments manually (via interface) or using an API. It allows an user to build a virtual server with the exact number of cores it decides is right for the job (up to 62). This approach is different from the adopted by other providers, such as, Rackspace, GoGrid and Amazon, that offer pre-packaged machines configurations.

To overcome the lack of automated mechanisms of some cloud providers, tools such as RightScale [14] has been developed. RightScale is a management platform that provides control and elasticity capabilities for different public cloud providers (Amazon, Rackspace, GoGrid, and others) and also for private cloud solutions (CloudStack, Eucalyptus and OpenStack). The solution provides an reactive mechanisms based on an Elasticity Daemon whose function is to monitor the input queues, and to launch worker instances to process jobs in the queue. Different scaling metrics (from hardware and applications) can be used to determine the number of worker instances to launch and when to launch these instances.

In order to take full advantage of the elasticity provided by clouds, it is necessary more than just an elastic infrastructure. It is also necessary that the applications have the ability to dynamically adapt itself according to changes in its requirements. In general, applications developed in Platform-as-a-Service (PaaS) clouds have implicit elasticity. These PaaS clouds provide execution environments, called containers, in which users can execute their applications without having to worry about which resources will be used. In this case, the cloud manages automatically the resource allocation, so developers do not have to constantly monitor the service status or interact to request more resources [15].

An example of PaaS platform with elasticity support is Manjrasoft Aneka [16]. Aneka is a .NET-based application development platform, which offers a runtime environment and a set of API's that enable developers to build applications by using multiple programming models such as Task Programming, Thread Programming and MapReduce Programming, which can leverage the compute resources on either public or private Clouds. In Aneka, when an application needs more resources, new container instances are executed to handle the demand, using local or public cloud resources.

Other example is the Google AppEngine [17], a platform for developing scalable web applications (Java, Python, and JRuby) that run on top of server infrastructure of Google. These applications are executed within a sandbox and AppEngine take care of automatically scaling when needed.

Azure [18] is the solution provided by Microsoft for developing scalable applications for the Cloud using .NET framework. Despite offering platform services, Azure does not provide an transparent elasticity control. The scaling of resources (VM's) is based on rules that the user defines specifically for an application.

Other cloud providers also provide elasticity mechanisms but the features offered are not substantially distinct from presented above. Basically, the current elasticity solutions offer a VM replication mechanism, accessed using an API or via interfaces, and in some cases the resources allocation is managed automatically by a reactive controller, based in a set of rules. Vertical elasticity is not fully addressed by most cloud providers. Other feature implemented in IaaS and PaaS clouds is the load balancing. Load balancers are used to distribute the workload among all available VM instances [19].

The solutions presented in this section and their characteristics are summarized in Table 1.

Table 1. Elasticity Solutions Characteristics

System	**Service**	**Mode**	**Elasticity**
Amazon [9]	IaaS	Automatic/API	Horizontal
Rackspace [10]	IaaS	Manual/API	Horizontal
GoGrid [11]	IaaS	Manual/API	Horizontal Vertical (memory)
Joyent [12]	IaaS	Automatic/ Manual/API	Horizontal
Profitbricks [13]	IaaS	Manual/API	Horizontal Vertical
RightScale [14]	IaaS (service)	Automatic	Horizontal
Aneka [16]	PaaS	Automatic	Horizontal (container)
AppEngine [17]	PaaS	Automatic	Horizontal
Azure [18]	PaaS	Automatic/ Manual/API	Horizontal

2 Challenges and Open Issues

Although many elasticity solutions has been developed by cloud providers, there are some issues that must be addressed to enable the wide use of elasticity in scientific applications.

2.1 Inappropriate Elasticity Mechanisms

Most of the elasticity solutions implemented by public providers were originally developed for dynamic scaling server-based applications, such as http, e-mail and databases. Most of these mechanisms are based on controlling the number of virtual machines that host the applications server components and in the use of load balancers to divide the workload among the many VM instances. The control is carried out by an elasticity controller that employs data from a monitoring system to decide when instances must be added or removed. The decisions are based on a set of rules that specify the conditions to trigger some actions over the underlying cloud. Every condition is composed of a series of metrics which are compared against a threshold to trigger actions over the underlying cloud. These metrics include the number of connections, number of requests and resources usage such as CPU, memory and I/O.

An example is presented in Figure 1, where it is possible to observe the allocation of VM's in function of the connected clients. The elasticity controller uses the number of clients to dynamically allocate or deallocate VM's, enabling application to be ready to handle the load variations [20].

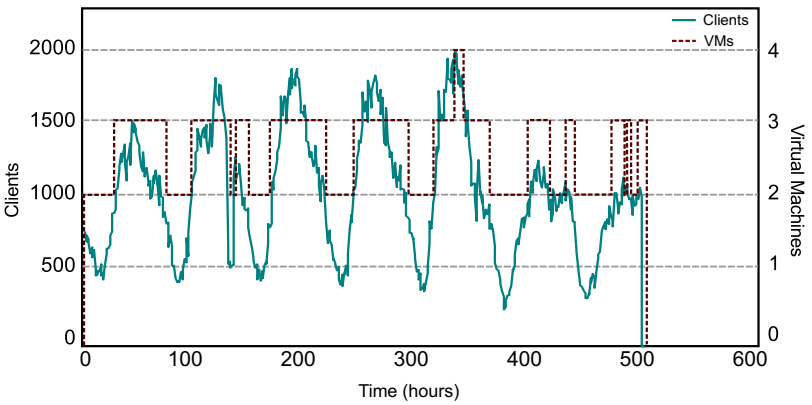

Fig. 1. Use of elasticity in a web application. Adapted from [20].

Although these solutions are successfully employed in server-based applications, scientific applications cannot benefit from the use of these mechanisms. Scientific applications have almost always been designed to use a fixed number of resources, and cannot explore elasticity without appropriate support [21]. The

simple addition of instances and the use of load balancers has no effect in these applications since they are not able to detect and use these resources.

Most of scientific applications are executed in batch mode and their workloads are defined by input files containing the data to be processed [22]. Besides, scientific jobs tend to be resource-greedy, using intensively all provided resources. Figure 2 illustrates this behavior in the execution of a scientific experiment (multithreaded 2D heat transfer). Note that all processing capabilities are constantly used, independently from the number of threads/CPUs employed. The absence of external requests and the constant and intense use of resources make ineffective the use of traditional elasticity mechanisms based in monitoring data.

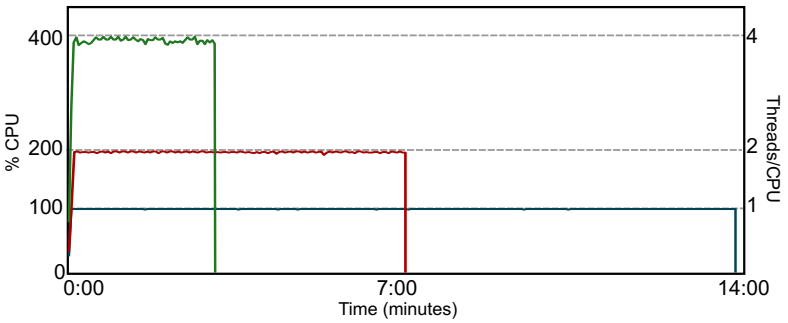

Fig. 2. Scientific application CPU usage with different number of threads

Using a elasticity mechanism such as offered by cloud providers, the high CPU usage could indicate the need for additional resources, causing the allocation of new virtual machines or new CPUs. However, the allocation of new resources has no effect in the CPU usage, since application is not designed to use the extra VM or CPU, and thus, more and more resources would be allocated indefinitely. Likewise, the use of manual approach neither is applicable, since is not possible to estimate the application state and if more resources are needed.

2.2 Resources Availability

Considering the available cloud platforms, none of them are able to accept the instantiation of a system with thousands of virtual machines for the period of time required to run a large scale scientific applications [23]. It happens because the elasticity of a cloud computing provider is limited by its capacity, and consequently, have to impose strict limits on the amount of resources that a single user can acquire in each instant of time, neglecting the infinite resources promise [24]. For instance, each Amazon EC2 regular customer has a limit of 20 reserved instances and 100 spot instances per availability zone that they can purchase each month; in Rackspace, all accounts have a preconfigured limit of 65 GB of total memory or approximately 130 individual 512 MB servers per region.

In fact, for the vast majority of users, the quota allowed is sufficient for their applications (generally, web applications). But, considering the applications characteristics, most of science-related users may want to receive from the cloud a high number of machines that could resemble a high-performance computing cluster. As resource-intensive applications begin effectively to use cloud computing, they will easily reach the scaling limits imposed by resources availability.

A possible solution to the resources availability problem is the use of multiple clouds to ensure the required amount of resources. Some academic works [25][26] have addressed this issue combining local and public clouds resources, however, the combined use of different public clouds remains challenging.

The reason for the current poor portability and limited interoperability between clouds is the lack of standardized API's, and consequently, each cloud provider has its own way on how cloud clients/applications/users interact with the cloud. As a consequence the interaction and migration of virtual machines and applications between clouds is a hard, if not impossible, task. This lack impacts on the development of mechanisms to provide large scale elastic computing models, able to scale resources among different cloud providers.

2.3 Limited Resources Granularity

Ideally, resources should be available at any granularity, allowing users to dynamically allocate from a single CPU to a complete virtual cluster, enabling different levels of elasticity [27]. However, in most IaaS clouds, clients acquire resources as a fixed set of compute, memory, and I/O resources (*instance types* in Amazon and *server sizes* in GoGrid and Rackspace). Renting a fixed combination of cloud resources does not reflect the applications demands [28].

There are a second point to be observed: Most of the cloud providers does not support vertical elasticity, i. e., it is impossible add a single CPU, memory, or I/O devices to a running VM. Changing the VM (or instance) type without rebooting is not also addressed. This limitations restrict the use of elasticity by diverse scientific applications, e. g., the ones that employ multithreaded parallelism or have phases with distinct demands of memory and I/O.

2.4 Spin-Up and Spin-Down Time

The great advantage of the elasticity is the ability to dynamically provide resources in response to a demand. However, one important fact in this dynamic process is that though cloud users can make their acquisition requests at any time, it may take some time for the acquired resources to be ready to use. This time period is called *spin-up time*.

In a perfectly elastic cloud, resourcing is instantaneous, i. e., there is no time delay between detecting load changes and changing resourcing levels [29]. However, in real world clouds, the startup time can vary (ranging from 1 to 10 minutes), depending on a number of factors including: type of cloud platform;

operating system type; number, size, or speed of resources requested; the availability of spare resources in the requested region and the demand on the cloud platform from other users. Thus, the resources provisioning could be slower than expected, affecting the efficacy and efficiency of actual elasticity mechanisms in handling highly dynamic workloads. Table 2 show the average VM spin-up time on Amazon EC2 (m1.small), Azure (Small) and Rackspace (Type IV) instances [30].

Table 2. Average VM spin-up time. Adapted from [30].

Cloud	**OS Image**	**Avg. Spin-up Time**
EC2	Linux(Fedora) ami-48aa4921	96.9 secs.
EC2	Windows (Win Server 2008) ami-fbf93092	810.2 secs.
Azure	WebRole default	374.8 secs.
Azure	WorkerRole default	406.2 secs.
Azure	VMRole - Win Server 2008R2	356.6 secs.
Rackspace	Linux (Fedora) flavor 71	44.2 secs.
Rackspace	Windows (Win Server 2008R2) flavor 28	429.2 secs.

In turn, *spin-down time* is the interval between no longer requiring a resource and no longer paying for it [27], and is directly related to the costs of using the cloud services. In Amazon, each partial instance-hour consumed will be billed as a full hour, i. e. the spin-down time is up to 1 hour. In Azure, instance hours are billed as full hours for each clock hour an instance is deployed. For example, if you deploy an instance at 10:50 AM and delete the deployment at 11:10 AM, you will be billed for two hours [18].

3 Towards Scientific Elastic Applications

Evaluating the challenges previously exposed, in this section we point some possibilities of using the elasticity in scientific applications, and describe some solutions that are being developed to overcome the challenges.

To address the problems related to *inappropriate mechanisms* we must consider two situations: (1) the development of new applications for the cloud, and (2) the execution of legacy applications in this environment type.

In new projects of scientific applications for the cloud, the applications must be reduced to frameworks that can successfully exploit the cloud resources. One possible approach is the use of building-blocks provided by PaaS clouds. In this case, the elasticity should be included in the modules and components provided, being managed transparently to the user. Generally, PaaS-based applications use execution environments called containers, which could automatically adapt their capabilities to satisfy the demands of the applications.

Another interesting approach, is the use of the MapReduce paradigm [31], that has gained popularity as a cloud computing framework on which to perform automatically scalable distributed applications. This application model can scale incrementally in the number of computing nodes. An user not only can launch a number of servers at the beginning, but can also launch additional servers in the middle of computation [8] [32]. The new servers can automatically figure out the current job progress and poll the queues for work to process. Previous work [33] has shown that MapReduce is well suited for simple, often embarrassingly parallel problems, but shown significant problems with iterative algorithms, like conjugate gradient, fast Fourier transform and block tridiagonal linear system solvers [34].

In case of legacy applications, scientific workflows is an example of approach that can benefit with cloud elasticity [35]. They can use the cloud capability to increase or reduce the pool of resources according to the needs of the workflow at any given time of processing [36]. Platforms and frameworks for executing scientific workflows in the cloud are being developed in academy. Examples of workflow system include Polyphony [37], Pegasus [38] and ClowdFlows [39].

Other legacy scientific applications (e. g. MPI, multithreaded) rely on IaaS cloud services and solely utilize static execution modes, in which an instance of VM is perceived as a cluster node [40]. To efficiently support elastic execution across cloud infrastructures, tools and frameworks, with support to scientific languages (C/C++, Fortran) and libraries are still required. Trying address this issue, a couple of academic researches have developed solutions to enable the development of elastic scientific applications. Some examples are the works of Raveendran et al. [21], addressing MPI applications, Rajan et al. [41], focusing on master-slave applications, and Galante and Bona [42] that present a platform for development of elastic applications based on the use of elasticity primitives.

The second problem addressed is the *resources availability*. It is closely related to the providers policies, but we believe that as demand grows, these limitations will be overcomed gradually. The potential of cloud resources are enormous and it became evident when a cluster composed by 1064 cc2.8xlarge instances (17024 cores) cluster was able to achieve 240.09 TeraFLOPS for the High Performance Linpack benchmark, placing the cluster at 127^{th} position in the June 2013 Top500 list.

As we said before, a possible solution to resources availability problem is the use of multiple clouds, but there is a lack of standards that enable *interoperability*. In this sense, some initiatives are attempting to create cloud standards. The Cloud Computing Interoperability Forum [43], are working on the creation an open and standardized cloud interface for the unification of various cloud API's. The IEEE [44] also has a project (P2301) on cloud portability and interoperability.

Other (future) perspective is based on the cloud federation. A federated cloud is the deployment and management of multiple external and internal cloud computing services to match business needs [45]. In this scenario, the exceeding demands of a cloud are fulfilled by leasing available computational and storage

capabilities from other cloud service providers. Some architectures for cloud federation has been proposed [46] [47], but practical results are still preliminary. Development of fundamental techniques and software systems that integrate distributed clouds in a federated fashion is critical to enabling composition and deployment of elastic application services.

The *resources granularity* issue is starting to be solved with the emergence of providers like Profitbricks (see Section 1.1) that enable users to combine different amounts of compute, memory, and I/O resources, i. e., offering vertical and horizontal scaling. This feature is very valuable for real elasticity, since resources can be allocated more efficiently.

Ben-Yehuda et al. [28] describe a perfect scenario, where compute, memory, and I/O resources could be rented and charged for dynamic amounts and not in fixed bundles. Clients rent VM's with some minimal amount of resources, and other resources needed are continuously rented in a fine-grained fashion. The resources available for rent include processing, memory, and I/O resources, as well as emerging resources such as accelerators, such as, FPGAs and GPUs. Processing capacity is sold on a hardware-thread basis, or as number of cycles per unit of time; memory is sold on the basis of frames; I/O is sold on the basis of I/O devices with bandwidth and latency guarantees.

The last issue, *spin-up and spin-down times*, will be overcomed with the use of new virtualization techniques and changing providers billing policy, respectively. Some works [48] [49] [50] present techniques to speed up the virtual provisioning process, but so far, these techniques have not yet been implemented by mainstream providers. In turn, the spin-down problem could be solved by changing the way providers charge by the use of resources. According to Brebner [29], even though it is unlikely that any cloud platform are perfectly elastic, it is possible to model it by assuming an extremely fine-grained cost model which only charges for resources that are actually consumed: the byte transmitted, the byte stored, and the millisecond of processing time.

To summarize, the challenges and perspectives of elasticity for scientific applications are presented in Table 3.

4 Final Remarks

Based on the analysis and studies made so far, from the point of view of providing elasticity, we argue that the use of cloud computing in supporting scientific applications may be an advantageous tool. Nevertheless, some care must be taken when using legacy applications, most of them will no fit to the current cloud model, and specific developments must be made when designed new scientific applications for this environment, to be able to use it in all its capability.

However, there are already scientific applications models (e. g. MapReduce and workflows) that can immediately benefit and with appropriate adjustments even more. Applications characterized by having data locality, loosely coupled, high throughput and fault tolerant, are more appropriate for the current cloud model.

Table 3. Elasticity: challenges and possibilities

Challenge	Possibilities	Related Works
Inappropriate elasticity mechanisms	– Use of PaaS and MapReduce for new applications – Workflows can be ported to clouds and adapted to use the cloud elasticity; – Development of new tools and frameworks	[40] [31] [33] [34] [35] [37] [38] [39] [21] [41]
Resources availability and Cloud interoperability	– Creation standards for cloud interoperability – Cloud Federation	[43] [44] [45] [46] [47]
Limited resources granularity	– Offering of replication and resizing of cloud resources for processing, memory, storage and networking	[13]
Spin-up and spin-down time	– Use of new virtualization techniques to speed up the virtual resources provisioning process – Changing providers billing policy in order to use a fine-grained cost model which only charges for resources actually consumed	[29] [48] [49] [50]

According to the presented in this paper, the answer to the question "*are the available cloud solutions suitable to support elastic scientific applications?*" is that the current computational clouds have not yet reached the necessary maturity level to meet all scientific applications requirements. We expect that in the coming years, significant advances in virtualization and in cloud management, allow the improvement of the elasticity solutions in scientific context.

Acknowledgment. This work is partially supported by CAPES and INCT-MACC (CNPq grant nr. 573710/2008-2).

References

1. Simmhan, Y., van Ingen, C., Subramanian, G., Li, J.: Bridging the gap between desktop and the cloud for escience applications. In: Proceedings of the 3rd Intl. Conference on Cloud Computing, CLOUD 2010, pp. 474–481. IEEE (2010)
2. Srirama, S.N., Willmore, C., Ivanitev, V., Jakovits, P.: Desktop to Cloud Migration of Scientific Experiments. In: 2nd International Workshop on Cloud Computing and Scientific Applications, CCSA 2012. IEEE/ACM (2012)
3. Badger, L., Patt-Corner, R., Voas, J.: Draft cloud computing synopsis and recommendations recommendations of the national institute of standards and technology. Nist Special Publication 146, 84 (2011)

4. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, A., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, L., Ionaharia, M.: A View of Cloud Computing. Commun. ACM 53(4) (April 2010)
5. Wang, L., Zhan, J., Shi, W., Liang, Y.: In cloud, can scientific communities benefit from the economies of scale? IEEE Transactions on Parallel and Distributed Systems 23(2), 296–303 (2012)
6. Oliveira, D., Baiao, F.A., Mattoso, M.: Migrating Scientific Experiments to the Cloud. HPC in the Cloud
7. Galante, G., Bona, L.C.E.: A survey on cloud computing elasticity. In: Proceedings of the Intl. Workshop on Clouds and eScience Applications Management, CloudAM 2012. IEEE/ACM (2012)
8. Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., Krintz, C.: See Spot Run: Using Spot Instances for Mapreduce Workflows. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010. USENIX Association (2010)
9. Amazon Web Services, `http://aws.amazon.com/`
10. Rackspace, `http://www.rackspace.com/`
11. GoGrid, `http://www.gogrid.com/`
12. Joyent, `http://joyent.com/`
13. Profitbricks, `https://www.profitbricks.com/`
14. RightScale, `http://www.rightscale.com/`
15. Caron, E., Rodero-Merino, L.: F. Desprez, A.M.: Auto-scaling, load balancing and monitoring in commercial and open-source clouds. Technical Report 7857. INRIA (2012)
16. Calheiros, R.N., Vecchiola, C., Karunamoorthy, D., Buyya, R.: The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. Future Generation Computer Systems 28(6), 861–870 (2011)
17. Google App. Engine, `http://code.google.com/appengine`
18. Microsoft Azure, `http://www.windowsazure.com/`
19. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. SIGCOMM Comput. Commun. Rev. 41, 45–52 (2011)
20. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: Proceedings of the 4th Intl. Conference on Cloud Computing, CLOUD 2011, pp. 500–507. IEEE (2011)
21. Raveendran, A., Bicer, T., Agrawal, G.: A framework for elastic execution of existing mpi programs. In: Proceedings of the Intl. Symposium on Parallel and Distributed Processing Workshops and PhD Forum, IPDPSW 2011, pp. 940–947. IEEE (2011)
22. Wang, L., Zhan, J., Shi, W., Liang, Y.: In cloud, can scientific communities benefit from the economies of scale? IEEE Trans. Parallel Distrib. Syst. 23(2), 296–303 (2012)
23. Costa, R., Brasileiro, F., de Souza Filho, G.L., Sousa, D.M.: Just in Time Clouds: Enabling Highly-Elastic Public Clouds over Low Scale Amortized Resources. Technical report, Federal University of Campina Grande (2010)
24. Costa, R.,, F.B.: On the amplitude of the elasticity offered by public cloud computing providers. Technical report, Federal University of Campina Grande (2011)
25. Fitó, J.O., Presa, I.G., Fernández, J.G.: Sla-driven elastic cloud hosting provider. In: Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, PDP 2010, pp. 111–118. IEEE (2010)

26. Marshall, P., Keahey, K., Freeman, T.: Elastic site: Using clouds to elastically extend site resources. In: Proceedings of the 10th Intl. Conference on Cluster, Cloud and Grid Computing, pp. 43–52. IEEE (2010)
27. Islam, S., Lee, K., Fekete, A., Liu, A.: How a consumer can measure elasticity for cloud platforms. Technical Report 680, School of Information Technologies, University of Sydney (2011)
28. Ben-Yehuda, O.A., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The Resource-as-a-Service (RaaS) cloud. In: 4th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2011 (2012)
29. Brebner, P.: Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (iaas) cloud applications. In: Proceedings of the Third Joint WOSP/SIPEW Intl. Conference on Performance Engineering, ICPE 2012, pp. 263–266. ACM (2012)
30. Mao, M., Humphrey, M.: A performance study on the vm startup time in the cloud. In: Proceedings of the IEEE Fifth Intl. Conference on Cloud Computing, CLOUD 2012, pp. 423–430. IEEE (2012)
31. Srirama, S.N., Jakovits, P., Vainikko, E.: Adapting scientific computing problems to clouds using mapreduce. Future Generation Computer Systems 28(1), 184–192 (2012)
32. Iordache, A., Morin, C., Parlavantzas, N., Riteau, P.: Resilin: Elastic MapReduce over Multiple Clouds. Rapport de recherche RR-8081, INRIA (October 2012)
33. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
34. Bunch, C., Drawert, B., Norman, M.: MapScale: A Cloud Environment for Scientific Computing. Technical report, University of California (June 2009)
35. Pandey, S., Karunamoorthy, D., Buyya, R.: Workflow Engine for Clouds. In: Buyya, R., Broberg, J., Goscinski, A.M. (eds.) Cloud Computing: Principles and Paradigms, pp. 321–342. John Wiley & Sons, Inc. (March 2011)
36. Byun, E.K., Kee, Y.S., Kim, J.S., Maeng, S.: Cost Optimized Provisioning of Elastic Resources for Application Workflows. Future Gener. Comput. Syst. 27(8), 1011–1026 (2011)
37. Shams, K.S., Powell, M.W., Crockett, T.M., Norris, J.S., Rossi, R., Soderstrom, T.: Polyphony: A workflow orchestration framework for cloud computing. In: Proceedings of the 10th IEEE/ACM Intl. Conference on Cluster, Cloud and Grid Computing, CCGRID 2010, pp. 606–611. IEEE (2010)
38. Vöckler, J., Juve, G., Deelman, E., Rynge, M., Berriman, B.: Experiences using cloud computing for a scientific workflow application. In: Proceedings of the 2nd Intl. Workshop on Scientific Cloud Computing, ScienceCloud 2011, pp. 15–24. ACM (2011)
39. Kranjc, J., Podpečan, V., Lavrač, N.: ClowdFlows: A cloud based scientific workflow platform. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 816–819. Springer, Heidelberg (2012)
40. Jha, S., Katz, D.S., Luckow, A., Merzky, A., Stamou, K.: Understanding Scientific Applications for Cloud Environments. In: Buyya, R., Broberg, J., Goscinski, A.M. (eds.) Cloud Computing: Principles and Paradigms, pp. 345–371. John Wiley & Sons, Inc. (March 2011)
41. Rajan, D., Canino, A., Izaguirre, J.A., Thain, D.: Converting a high performance application to an elastic cloud application. In: Proceedings of the 3rd Intl. Conference on Cloud Computing Technology and Science, CLOUDCOM 2011, pp. 383–390. IEEE (2011)

42. Galante, G., Bona, L.C.E.: Constructing elastic scientific applications using elasticity primitives. In: Murgante, B., Misra, S., Carlini, M., Torre, C.M., Nguyen, H.-Q., Taniar, D., Apduhan, B.O., Gervasi, O. (eds.) ICCSA 2013, Part V. LNCS, vol. 7975, pp. 281–294. Springer, Heidelberg (2013)
43. CCIF: The Cloud Computing Interoperability Forum, http://www.cloudforum.org/
44. IEEE: Cloud Profiles Working Group, http://standards.ieee.org/develop/project/2301.html
45. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: Three-phase cross-cloud federation model: The cloud sso authentication. In: Proceedings of Second Intl. Conference on Advances in Future Internet, pp. 94–101 (2010)
46. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu, C.-H., Yang, L.T., Park, J.H., Yeo, S.-S. (eds.) ICA3PP 2010, Part I. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010)
47. Villegas, D., Bobroff, N., Rodero, I., Delgado, J., Liu, Y., Devarakonda, A., Fong, L., Sadjadi, S.M., Parashar, M.: Cloud federation in a layered service model. J. Comput. Syst. Sci. 78(5), 1330–1344 (2012)
48. Zhu, J., Jiang, Z., Xiao, Z.: Twinkle: A fast resource provisioning mechanism for internet services. In: Proceedings of the 30th IEEE Intl. Conference on Computer Communications, INFOCOM 2011, pp. 802–810. IEEE (2011)
49. Tang, C.: Fvd: a high-performance virtual machine image format for cloud. In: Proceedings of the 2011 USENIX technical conference, USENIX 2011, p. 18. USENIX Association (2011)
50. De, P., Gupta, M., Soni, M., Thatte, A.: Caching VM instances for fast VM provisioning: A comparative evaluation. In: Kaklamanis, C., Papatheodorou, T., Spirakis, P.G. (eds.) Euro-Par 2012. LNCS, vol. 7484, pp. 325–336. Springer, Heidelberg (2012)