J Grid Computing (2016) 14:193–216
DOI 10.1007/s10723-016-9361-3

An Analysis of Public Clouds Elasticity in the Execution of Scientific Applications: a Survey

Guilherme Galante · Luis Carlos Erpen De Bona · Antonio Roberto Mury · Bruno Schulze · Rodrigo da Rosa Righi

Received: 4 February 2015 / Accepted: 11 January 2016 / Published online: 28 January 2016

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 provide elasticity to scientific applications?" To answer the question, in a first moment we present a survey on the use of cloud computing in scientific scenarios, providing an overview of the subject. Next, we describe the elasticity mechanisms offered by major public cloud providers and analyzes the limitations of the solutions in providing elasticity for scientific applications. As the main contribution of the article, we also present an analysis over some initiatives that are being developed to overcome the current challenges. In our opinion, current computational clouds are developing rapidly but have not yet reached the necessary maturity level to meet all scientific applications elasticity requirements. We expect that in the coming years the efforts being taken by numerous researchers in this area identify and address these challenges and lead to better and more mature technologies that will improve cloud computing practices.

Keywords Cloud computing · Elasticity · Scientific applications

G. Galante (✉)
Department of Computer Science, Western Paraná State University, Cascavel, PR, Brazil
e-mail: guilherme.galante@unioeste.br

L. C. E. De Bona
Department of Informatics, Federal University of Paraná, Curitiba, PR, Brazil
e-mail: bona@inf.ufpr.br

A. R. Mury · B. Schulze
National Laboratory for Scientific Computing, Petrópolis, RJ, Brazil

A. R. Mury
e-mail: aroberto@lncc.br.br

B. Schulze
e-mail: schulze@lncc.br.br

R. R. Righi
Applied Computing Graduate Program, Unisinos, São Leopoldo, RS, Brazil
e-mail: rrrighi@unisinos.br

1 Introduction

Scientific Computing is the key to solving "grand challenges" in many domains and has provided advances and new knowledge in diverse fields of science. It is the combination of engineering, natural science, computer science and mathematics, making scientific computing a demanding field for all participating parties: engineers contribute with challenging

applications and technical knowledge; physicists and other natural scientists build the models; mathematicians provide numerical methods and algorithms for the simulation of complex processes; and computer scientists contribute with the construction of infrastructures, data structures and algorithms.

Running large and accurate simulations requires a large amount of computing resources, often demanding the use of supercomputers, computer clusters or grids. Thus, scientific computing has historically been dependent on the advances of High Performance Computing (HPC) and parallel processing. In last few years, in addition to supercomputers and grids, cloud computing has proved itself as a new way to acquire resources on demand [1].

Cloud computing emerged as an alternative for solving scientific computing problems, with the promise of provisioning virtually infinite resources. According to Simmhan et al. [2], the use of cloud computing environment can be attractive to scientific community in many ways, benefiting users that own small applications, but also those who run 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 execute scientific experiments [3, 4].

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 be seen as the ability to on-demand scale-up and down the number of cloud resources allocated for an application [5]. Ideally, to the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time. 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. On the other hand, horizontal elasticity is the ability of the cloud to vary the number of VM's allocated to a service according to demand.

Considering the importance of elasticity, several mechanisms to provide this feature are offered by public cloud providers, such as, Amazon EC2, GoGrid and Rackspace, and by many academic works [6, 7]. These mechanisms has been traditionally used for dynamic scaling server-based applications, such as web, e-mail and databases servers, to handle unpredictable workloads, and enabling companies to avoid the downfalls involved with the rigid provisioning (over and under-provisioning) [8, 9]. In scientific scenario, the use of cloud computing is discussed in several studies [1, 3, 10, 11], but the use of elasticity in scientific HPC applications is a subject that is starting to receive attention from research groups.

This interest is related to the benefits that cloud elasticity 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. The cost reduction is relevant when using resources from public clouds, since resources could be allocated on demand, instead of allocating everything at the beginning of execution, avoiding over-provisioning and allowing economy of scale. Elastic applications can also increase their computational capabilities when cheaper resources became available, as presented in the works of Taifi et al. [12] and Chohan et al. [13], in which Amazon Spot Instances are used.

Examples of using elasticity in scientific applications include the dynamic storage space allocation when data exceeds the capacity allocated for the hosted environment in the cloud [14, 15]; applications that use the MapReduce paradigm, where is possible to increase the number of working nodes during the Map and consequently to scale back resources during the Reduce phase [13, 16]; workflows execution, in which is possible to dynamically adjust the pool of nodes required to resolve a given workflow step [17, 18]; and many others [6]. Thus, considering the possibilities of using elasticity in the scientific context, a question arises: *are the available public cloud solutions suitable to provide elasticity to scientific applications?* To answer this question, in a first moment we present a survey on the use of cloud computing in scientific scenarios providing an overview of the subject. Next, we describe the elasticity mechanisms offered by major public cloud providers and analyzes the limitations of the solutions in providing elasticity for scientific applications. We also present some initiatives that are being developed to overcome the current challenges. This last part of the manuscript comprises our main contribution for the

state-of-the-art when combining cloud elasticity with HPC, parallel and distributed computing, where we detach the open issues in the aforementioned area.

2 Scientific Applications in the Cloud: An Overview

Scientific applications are characterized by large amounts of data or intensive processing requirements. Access to faster and powerful computational resources enables the resolution of larger, more complex problems in a more precise way. Currently, the use of HPC and parallel architectures is the solution *de facto* for running scientific applications with high computational demands. These systems are composed of large machines or formed by the aggregation of processing power, memory, network and storage resources from several independent machines, which are used in a coordinated manner to execute parallel and distributed applications. Examples of architectures are HPC clusters, grids and recently, the clouds.

The literature shows many studies focusing on the use of clouds in scientific computing. In one hand, some authors consider cloud computing as a good alternative to meet the demands for resources of the scientific community. On the other hand, some works point out the disadvantages of using cloud in the execution of scientific applications. In this section, we present an overview covering the benefits and drawbacks involved in the execution of high performance and parallel scientific applications in the cloud.

2.1 Science in the Cloud: Benefits

Cloud computing can provide a variety of resources (hardware and software) that can be allocated quickly, on demand and at relatively low cost, without the need to purchase additional infrastructure [3]. According to Oliveira et al. [19], the main benefit of cloud computing is that the average user is able to access a variety of resources without having to purchase or configure an entire infrastructure. This is a fundamental need for scientific applications, since scientists do not need to worry about the complexity of the environment, focusing only on their activities research. This characteristic is also interesting in the cases where research groups have temporary demands or have few financial resources to purchase equipment [11, 20].

In fact, many scientific cloud platforms have emerged recently with the proposal of providing processing capabilities, appropriate storage and network for scientific computations. Some examples are the scientific clouds Helix Nebula, ScienceClouds and CloudLab.

Helix Nebula [21] is a pioneering partnership between big science and big business in Europe that is charting a course towards the sustainable provision of cloud computing – the Science Cloud. This strategy targets to boost scientific innovation and to bring new discoveries through novel services and products. Currently, there are four flagship projects running on the cloud: CERN-LHC, EMBL Next Generation DNA Sequencing, ESA SSEP earthquake and volcano research, and PIC Medical research in neurodegenerative diseases.

The Science Clouds [22] project consists of an informal group of small cloud platforms available on a voluntary basis by a group of institutions and public clouds. Started in mid-2008, the project has the objectives of help the experimentation and the use of cloud computing in IaaS style for scientific projects. The group is formed by the universities of Chicago (USA), Cornell (USA), Purdue (USA), which voluntarily provide computing resources that are accessed through Nimbus Phantom middleware.

CloudLab [23] is a "meta-cloud" that is, it is not a cloud itself; rather, it is a facility for building clouds. It provides bare-metal access and control over a substantial set of computing, storage, and networking resources; on top of this platform, users can install standard cloud software stacks, modify them, or create entirely new ones. The initial CloudLab deployment will consist of approximately 15,000 cores distributed across three sites at the University of Wisconsin, Clemson University, and the University of Utah.

Some public clouds have also proposed some high-performance solutions, such as those offered by Amazon [24] and Sabalcore [25]. High Performance Computing on Amazon EC2 is enabled by the Cluster Compute-optimized and GPU instance types (virtual machines). Cluster Compute Instances combine high compute resources with a high performance networking for HPC applications and other demanding network-bound applications. Cluster GPU Instances provide GPUs with proportionally high CPU and increased network performance for applications benefiting from highly parallelized processing that can be

accelerated by GPUs using the CUDA and OpenCL programming models. Sabalcore offers HPC services over the Internet. They offer a full cloud HPC infrastructure. This company offers an on-demand cloud system, with preinstalled OS and applications. No virtualization is used, and computation is directly applied on the metal. Resources of this system are interconnected using an Ethernet or an Infiniband network. Intel or AMD CPU and NVidia graphics card compose the nodes of this system.

Other possibility of using clouds in science is the use of scientific PaaS. PaaS provides a higher-level of abstraction than IaaS as developers are provided with a platform containing services that can be used to build applications. The e-Science Central [26] is a cloud based platform that provides science application developers a high-level, cloud-based platform on which to build their applications. It supports secure storage and versioning of data, audit and provenance logs and processing of data using workflows. Other example of PaaS platform is Aneka [27], 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 employ the compute resources on either public or private clouds. Currently, major cloud providers (e.g., Amazon and Google) offer PaaS services, however not focusing exclusively scientific applications.

Some authors discuss the benefits of using of cloud resources to supplement local resources to satisfy peaks or fluctuating demands. In the work of Marshall et al. [28] Amazon EC2 instances are used to extend the capabilities of local virtual clusters created using Nimbus middleware. Bicer et al. [29] describe a framework for storing and processing data using local and cloud resources. Calatrava et al. [30] present a study about the use of a hybrid infrastructure, comprising cloud and grid resources, for the execution of scientific applications. A similar solution is proposed by Calheiros et al. [27], which integrates features of cloud and grid (Desktop Grids) for running Aneka platform applications. Mateescu et al. [31] propose Elastic Cluster a cloud-based platform for management of HPC and cloud resources. It is a self-resizable resource that adapts to the workload to meet the timing requirements of the applications. Elastic Cluster automatically provisions additional resources leased from a private or public cloud. The worker nodes of the Elastic Cluster can run either traditional physical machines or virtual machines.

Computational clouds have also been used to reduce the response time of scientific applications [2, 32]. Commonly, the submission of a job in a high performance environment is made through a queue whose waiting time can vary from minutes to days, depending on the period allocated and on the available resources. Considering that clouds can provide resources instantaneously, it is possible to reduce the application response time, since waiting periods are eliminated [4, 33].

Another use for cloud computing is described by Edlung et al. [34]. The authors use clouds to execute tasks that do not require high performance features, but are allocated in supercomputer queues. According to the study, approximately 20 % of the applications that currently use the Nordic supercomputing centers (distributed among Sweden, Finland, Norway, Denmark and Iceland) can be dispatched to the cloud, reducing queues and waiting time of applications that really need high performance resources. The authors believe that similar scenarios may occur in other supercomputing centers around the world.

The customization of execution environments, in terms of hardware and software, is also an interesting feature to scientific computing [35]. Scientific applications have strong dependencies related to specific versions of operating system, tools and libraries. Due to the variability of the software layers supported in each location, to configure and play the exact execution environment (when possible) requires hours to days of work and coordination [4].

Virtualization allows to use highly customized environments for scientific jobs, ranging from libraries down to the operating system. All the software stack can be encapsulated in a single virtual machine image that can be easily transferred to other physical machines, copied or saved for later use, enabling the execution of replicas of the complete environment [36, 37]. Some research groups, such as CERN [38], are investigating the use of virtual machine images for the distribution of all software stack necessary for running applications. It helps the resources provisioning for an application, since the virtual machine can be run as easily locally as on remote community infrastructure or resources outsourced commercially.

Clouds may offer much more than computing power. A recent trend is that scientific contributions require the publication of large datasets. Some examples are the Allen Institute's Brain Atlas and the genome databases from National Center for Biotechnology Information, and Ensembl. These data repositories are widely used by researchers to do computation-intensive analysis of data collected by community. Hosting these datasets in public clouds is much easier than requiring individual scientists (or even universities) to build their own data hosting systems [39].

In fact, current public cloud providers offer a variety of storage solutions, such as, Amazon S3 and EBS, and Microsoft Azure storage services. Hence, users of cloud services should not need to worry about scalability because the storage provided is virtually infinite and the network links are virtually capable to quickly transfer the data between the available servers. Therefore, from the data management point of view, cloud providers must provide full availability. If components fail, the service provider must replace them and make the data available using replicas in the meantime. Another important reason to build new services based on cloud computing is that no investments are needed upfront and the cost grows linearly and predictably with the increase of usage [40].

Elasticity is another important feature that can be explored by scientific applications, and that is not available in traditional architectures. Traditionally, HPC applications are executed on clusters or even in grid architectures. Overall, both have a fixed number of resources that must be maintained in terms of infrastructure configuration, scheduling and energy consumption. In addition, tuning the number of processes to execute a HPC application can be a difficult procedure: (i) both short and large values will not efficiently explore the distributed system; and (ii) a fixed value cannot fit irregular applications, in which the workload varies during execution or occasionally is not predictable in advance. Conversely, cloud elasticity abstracts the infrastructure configuration and technical details about resource scheduling from users, who pay for resources, and consequently energy, in accordance with applications demands.

Applications can make use of virtualization and high availability of resources offered by clouds to dynamically acquire new resources according to demands [6]. This feature is specially interesting for dynamic applications whose resource requirements cannot be determined exactly in advance, either due to changes in runtime requirements or due to interesting changes in application structure [41]. The ability to exploit these attributes could lead to applications with new and interesting usage modes and dynamic execution on clouds and therefore new application capabilities. Besides, some initiatives use elasticity to improve the resources management in traditional platforms, such as clusters [42] and grids [43], hence many researchers are proposing grid-of-clouds architectures and others have positioned themselves as offering HPC-oriented clouds.

Considering that the elasticity is the main focus of this work, the Sections 3, 4 and 5 have the objective of presenting the state of art, the challenges and initiatives towards elastic scientific applications. Table 1 summarizes the benefits of using cloud computing in scientific scenarios and related references.

2.2 Science in the Cloud: Drawbacks

Although the use of clouds presents many benefits, as discussed so far, there are some issues without consensus and preventing widespread adoption of cloud computing in the scientific community, among which we can mention the performance, cost, applications portability and data management in the cloud.

Several authors analyze the performance of scientific applications in public clouds. Bientinesi et al. [44] present a study on the performance of mathematical applications on Amazon EC2 (large instances). The authors point out that processing nodes, if

Table 1 Science in the Cloud: Benefits

Benefit	Related work
Quick access to a variety of resources	[3, 11, 19, 20]
Local resources supplementation	[27–30]
Response time reduction	[2, 4, 32, 33]
Workload adequation	[34]
Hardware and software customization	[4, 35]
Experiments replication	[36–39]
Elasticity	[6, 41]

considered individually, are as good as those found in high performance systems. However, the high latency of interconnection network and the physical machines sharing (affecting cache memory) influence the overall platform performance. Similar conclusions are also presented by Hill and Evangelinos [45], He et al. [32] and Rehr et al. [46]. In the study by Gupta et al. [47], the authors analyze the network performance issue and conclude that public clouds are suitable for applications with low communication patterns, such as embarrassingly parallel and tree-structured computations, and HPC-optimized platforms are better for strongly coupled tasks.

Church and Goscinski [48] observed the performance of high performance features offered by public clouds. They evaluate high-speed and low latency networks including the Amazon cluster instances and two private clouds with VMware vSphere technology [49]. Results show that the performance obtained in these clouds is similar to those achieved in a cluster with InfiniBand network. Performance of private clouds is also analyzed in some works [36, 50]. It was found that the performance of private clouds is determined by the underlying infrastructure. This means that even considering the virtualization layer, the performance of the applications is close to achieved in bare metal.

The virtual machine performance variability is other relevant issue. As the cloud computing environment is commonly highly heterogeneous, there may be machines with different processors types and models. This difference between CPUs can directly influence the performance of VMs and, consequently, the applications encapsulated within, even in instances of the same type. An example is presented by Schad et al. [51], where the performance results of a MapReduce execution on Amazon EC2 show fluctuations up to 24 %. Another issue related to performance disparities due to heterogeneity is that the slowest instance will be the bottleneck for an entire execution of a parallel application (MPI, for example), thus, under-utilizing faster machines.

Phillips et al. [52] also studied the performance variation. Analysing a set of codes with different patterns (7 dwarfs), they point out that on Amazon EC2, the description of the virtual machines using just EC2 Compute Units (ECUs) and RAM size is not sufficient for predicting performance. Machines described as “small” may have different clock speeds and cache size and even knowledge of this additional detail does not help in performance prediction as some applications run slower on the machines with a faster clock. The performance variation in private clouds is discussed in the work of Rego et al. [53], in which the authors evaluate the influence of the diversity of different physical machines present in a private cloud. The authors also propose a tool to standardize the performance of virtual machines in private clouds, regardless of the physical hardware.

The cost of doing science in the cloud is an issue that must be examined. Considering that most public clouds adopt the pay-per-use model, it is important to estimate the final cost to be paid and to determine how financial resources available to a scientific experiment will be used [19]. In general, the price to be paid for cloud usage can be classified into three types (to be examined by users): insource (scientists using their own cloud, implying in providing the necessary infrastructure and maintenance), pay-per-use (pays a certain amount related to the use of cloud resources, usually per hour) and component (payment for using each type of component, independent of time). However, this review is far from simple, since the costs reduced by the adoption of the cloud model, such as purchasing equipment and hiring of support personnel, are difficult to calculate. Thus, each situation must be examined individually to see which is the most viable solution. The works of Nanath and Pillai [54] and Negru and Cristea [55] present cloud cost models, that can help to evaluate the costs. Different considerations about the cost of using the cloud for scientific purposes appear in the works of Berriman et al. [56], Fox and Gannon [57], Li et al. [58] and Vecchiola et al. [3].

In addition, moving science to the cloud may also present hidden costs, for example those associated to recoding or porting applications to take advantage of cloud functionalities [39]. When looking to port a specific HPC application to the cloud there is no generic approach we can follow. The challenge is to think differently and rewrite the application to support the new computational and programming models [57]. It is necessary to analyze the structure and nature of the application and determine how to exploit what the cloud has to offer. For example, if the application is tightly coupled, then obviously it will restrict the freedom of where you port the application to, on the other hand if it is relatively loosely coupled then this will

enable the greatest freedom when porting the application. Fox and Gannon [57] suggest the use of frameworks (e.g, MapReduce or Hadoop), PaaS platforms (e.g, Microsoft Azure and Aneka), and workflows in the construction or adaptation of applications that will run in the cloud.

There are also issues related to data management in the cloud: security, data placement and data lock-in. In general, moving data off premises increases the number of potential security risks and appropriate precautions must be made. According to Xiaotao et al. [59], in the context of scientific computing, data is highly sensitive, involving intellectual property, planning information or even national security, and you cannot trust the security mechanisms implemented in clouds, no matter how well intentioned they may be.

In order to prevent confidentiality violations, cloud services customers might resort to encryption. However, encryption is effective in securing data before it is stored at the provider, it cannot be applied in services where data is to be computed, since the unencrypted data must reside in the memory of the host running the computation [60]. Zissis and Lekkas [61] argue that securing data traveling over the network is a hard and highly complex issue, while the threat of data modification and data interruption is continuously rising. A cloud environment increases this complexity as it does not only require protection of traffic towards the cloud but additionally between cloud hosts, as they lack a traditional physical connection. Considering the security issues, organizations that have sensitive data should avoid using public clouds to store their data and applications, and dedicated systems remains most appropriate for these situations.

Data placement is also an importance issue. Running scientific workflow applications usually need not only high performance computing resources but also massive storage. For example, reported real-life use of the Montage workflow has a workflow of 10,422 tasks that exchange a total of 57 GB of data, whereas the Cybershake workflow has 80 partitions, with each running a total of 24,132 tasks exchanging 58 GB of data [62].

Thakar et al. [63] argue that it is impossible to migrate a large database (TB of data). The limitations of cloud services make it very difficult to migrate data to the cloud without making changes to the schema and settings, degrading performance or make the data unusable. These difficulties suggest that much work and coordination needs to occur between cloud service providers and their potential clients before science databases could successfully and effectively be deployed in the cloud.

For application data that are possible to be moved (some may have fixed locations), it is not possible to move them whenever and wherever we want, since cloud data centers may belong to different service providers so that data movement would result in costs. In addition, data movement between different providers may spend many time, since sites are spread around the Internet with limited bandwidths. To effectively store these data, a data manager must intelligently select where these data will reside [64].

In addition, the data lock-in is an important issue in data movement/placement. Considering that cloud APIs have not been subject of active standardization, it is difficult to extract data and programs from one site to run on another. Customer lock-in may be attractive to cloud computing providers but cloud computing users are vulnerable to price increases, to reliability problems, or even to providers going out of business. The lack of cloud standards and the cloud interoperability problem is addressed in Sections 4.3 and 5.

Table 2 summarizes current drawbacks of using cloud computing in scientific scenarios and the related references. The aforementioned points directly impact in cloud elasticity and we highlight the network contention caused by the interference between virtualized environments in public clouds [65], as well the fluctuation in performance caused by the same reason. In both cases, new scheduling algorithms must try to avoid the overhead caused by virtualization and to improve the applications performance [66, 67].

After presenting this overview of the benefits and drawbacks of using cloud computing for scientific purposes, in the next sections we focus on the central subject of this paper: **cloud elasticity**. In Section 3, we

Table 2 Science in the Cloud: Drawbacks

Drawback	Related work
Network performance	[36, 44, 47, 48, 50]
Performance fluctuations	[51–53]
Cost	[3, 19, 54–58]
Applications portability	[39, 57]
Data management	[59–61, 63, 64]

present the elasticity mechanisms offered by public cloud providers. Considering the presented solutions, in Section 4 we point out the current challenges in providing elasticity to scientific applications. Section 5 describes some solutions that are being developed to overcome these challenges.

3 Elasticity Mechanisms in Public Cloud Environments

In this section, we present eleven elasticity solutions proposed by important IaaS and PaaS public providers. We evaluated these solutions to analyze how the mechanisms of elasticity have been implemented and what features are provided by them, and then, to define what are the challenges and opportunities for research in the area (Section 4). We focused our study in public clouds, since they are available to wide public, and, due to their large infrastructures, can provide access to scalable (and elastic) computing resources.

Amazon Web Services [24], 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 a reactive approach, in which, for each ASG there is a set of rules that defines the number of VMs 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.

Amazon also offers the Amazon Elastic MapReduce (EMR), a service that provides fully managed hosted Hadoop framework on top of Amazon Elastic Compute Cloud (EC2). The service ramps up or reduces resource usage depending on the demand at any given time. EMR could be used for data analysis in log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, bioinformatics and etc.

Rackspace [68] 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 [69] 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 VMs replication, GoGrid also support vertical elasticity for memory. The solution provided by Joyent [70] is also based VM replication accessed via API. However, the provider includes an automatic feature called CPU bursting, which temporarily increase the CPU capability of up to 400 % to handle workload spikes.

A more comprehensive elasticity solution is provided by Profitbriks [71]. According to its documentation, horizontal and vertical elasticity are available, allowing changes in virtual environments manually (via interface) or using an API. It allows a user to build a virtual server with the exact number of cores it decides is right for the job (up to 62). Similarly, CloudSigma [72] and ElasticHosts [73] also allow their cloud consumers to customize any combination of cloud server instances resources capacity according to their needs. The approach adopted by Profitbricks, CloudSigma and ElasticHosts is different from the adopted by other providers, such as, Rackspace, GoGrid and Amazon, that offer prepackaged machines configurations.

To overcome the lack of automated mechanisms of some cloud providers, tools such as RightScale [74] have 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 a 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.

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 themselves 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 [75].

An example of PaaS platform with elasticity support is Aneka [27] (already described in Section 2.1). 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 [76], 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 [77] 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 sometimes, 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 [78].

Although there are dozens of other public cloud providers, the offered elasticity mechanisms are not significantly different from those presented here. Table 3 summarizes the solutions presented in this section.

4 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.

4.1 Inappropriate Elasticity Mechanisms

Most of the elasticity solutions implemented by public providers was originally developed for dynamic scaling server-based applications, such as http, e-mail and databases, and to try overcome the problems caused by fluctuating demands. In fact, cloud computing can provide different resources on-demand for many of the

Table 3 Elasticity Solutions Characteristics

System	Service	Mode	Elasticity
Amazon [24]	IaaS	Automatic/API/MapReduce	Horizontal
Rackspace [68]	IaaS	Manual/API	Horizontal
GoGrid [69]	IaaS	Manual/API	Horizontal Vertical (memory)
Joyent [70]	IaaS	Automatic/ Manual/API	Horizontal
Profitbricks [71]	IaaS	Manual/API	Horizontal/Vertical
CloudSigma [72]	IaaS	Automatic/API	Horizontal/Vertical
ElasticHosts [73]	IaaS	Automatic/API	Horizontal/Vertical
RightScale [74]	IaaS (service)	Automatic	Horizontal
Aneka [27]	PaaS	Automatic	Horizontal (container)
AppEngine [76]	PaaS	Automatic	Horizontal
Azure [77]	PaaS	Automatic/ Manual/API	Horizontal

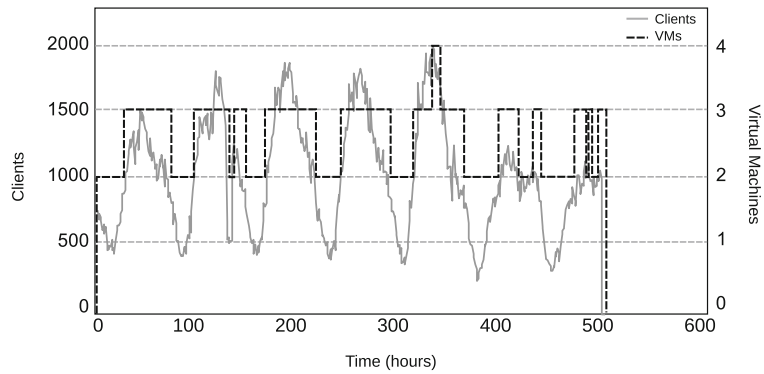


Fig. 1 Use of elasticity in a web application. Adapted from Roy et al. [79]

typical scaling points that applications need including servers, storage and networking [8].

Most of the current elasticity 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 [6, 78]. The control is carried out by an elasticity controller that uses 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 Fig. 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 VMs, enabling application to handle the load variations [79].

Although the described solutions are successfully used 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 [80]. 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. In addition, the fact of either a premature death of a process or a consolidation of a VM that hosts one or more processes from a tightly coupled parallel code can imply the termination of the application execution. Particularly, the crash of a single process causing the application termination is commonly observed in MPI applications.

Most of traditional scientific applications (not constructed or adapted for cloud computing) are executed in batch mode and their workloads are defined by input files containing the data to be processed [10]. Besides, scientific jobs tend to be resource-greedy, using intensively all provided resources. Figure 2

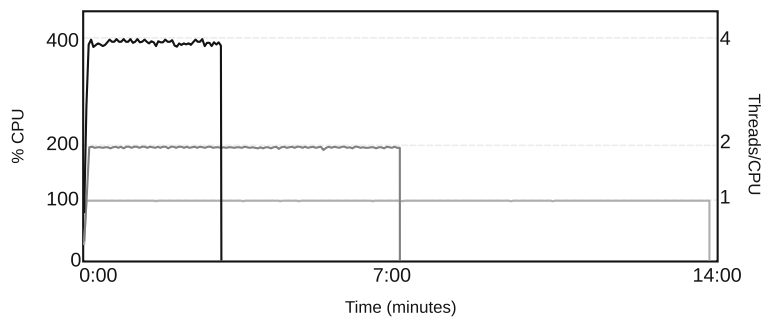


Fig. 2 Scientific application CPU usage with different number of threads

illustrates this behavior in the execution of a scientific experiment (multithreaded 2D heat transfer application). Note that all processing capabilities are constantly used, independently from the number of threads/CPUs employed. The absence of external requests and constant and intense use of resources make ineffective the use of traditional elasticity mechanisms based in monitoring data.

Using an elasticity mechanism such as offered by public cloud providers, the high CPU usage could, for example, indicate the need for additional resources, causing the allocation of new virtual machines or new vCPUs. However, the allocation of new resources has no effect in the CPU usage, since application is not designed to use the extra VM or vCPU, and thus, more and more resources would be allocated indefinitely without performance improvements. Likewise, the use of manual approach neither is applicable, since is not possible to estimate the application state and if more resources are really needed.

4.2 Database Scalability

In scientific experiments, the scientists usually need a database to store provenance data, because a database provides features such as indexing and concurrency control, that a simple storage does not provide. However, scaling relational database in the cloud is one of the critical factors in the migration of applications to the cloud, since they cannot be scaled very easily, since they are in general intended for an infrastructure that is statically provisioned [81].

Scalability of a system only provides us a guarantee that a system can be scaled up from a few machines to a large number of machines. In cloud computing environments, we need to support additional property that such scalability can be provisioned dynamically without causing any interruption in the service [82]. In fact, current technology fails to provide adequate tools and guidance if an existing relational database deployment needs to scale-out from a few machines to a large number of machines [83].

4.3 Resources Availability

Even though clouds offer the promise of elasticity of computing resources that would appear to users as endless supply, there are scenarios in the scientific world for which the resources offered by a single cloud would not be enough. Once the demand for processing power reaches the maximum capacity for a provider, there are no additional means to acquire new resources for the users [84]. Considering the available cloud platforms, none of them can accept the instantiation of a system with thousands of virtual machines for the period of time required to run large scale scientific applications [85]. It happens because the elasticity of a cloud computing provider is limited by its capacity and policies, imposing limits on the amount of resources that a single user can acquire in each instant of time. If a service provider had not enough local resources to fulfill its customers' requirements, it should start denying the acceptance of new customers or canceling low priority services that were already running on the system. This has further implications than just losing the revenue from some services, because it also implies a loss of reputation and therefore a loss of future customers.

Table 4 shows the resources quotas in some of major cloud providers. Most compute services impose limits on the number of instances you can have active at a given time. It may affect the elastic allocation of resources in applications whose demands grow fast and continuously, or that require a large amount of resources during workload peaks.

In fact, for most users, the quota allowed is enough for their applications (generally, web applications). But, considering the applications characteristics, 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.

Table 4 Quota policies in some public cloud providers

Cloud	Quota
Amazon EC2	20 instances per region
DigitalOcean	10 instances
Google Compute Engine	24 CPU cores per region
Microsoft Azure	20 CPU cores
Rackspace	128 GB memory
HP Cloud Compute	40 instances or 15 GB memory
Verizon	20 instances

A possible solution to the resources availability problem is the use of multiple clouds to ensure the required amount of resources [86]. The technical aspects of using multiple clouds is nowadays still in an infancy stage, mainly due to the diversity of the approaches of the concept implementation. Attempting to use different providers as a backup would mean different protocols, security schema and new APIs to be employed. Differences are likely in network addressing, directory services, firewalls, routers, switches, identity services, naming services, security policies, and other resources, preventing them from agreeing upon a widely accepted, standardized way to support cloud applications [87]. Such vendor lock-in has seriously limited the flexibility that cloud end users would like to process, when it comes to deploy applications over different infrastructures in different geographic locations, or to migrate a service from one provider to another [88].

4.4 Limited Resources Granularity

Different classes of applications often have different workload patterns and characteristics and, therefore, their elasticity requirements may vary accordingly. Ideally, highest levels of economic elasticity may be achieved by enabling cloud consumers to customize any combination of resources capacity and as much as their application workloads require [89, 90].

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), and consequently, cloud consumers are likely to pay for unused resources capacity. Renting this fixed combination of cloud resources could not reflect the exact application demand [91].

There are other point to be observed. Most of the cloud providers does not support vertical elasticity, i.e., it is impossible to add a single CPU, memory, or I/O devices to a running VM. Using VMs as basic units for dynamic resource provisioning for cloud applications incurs considerable overhead and costs. Allocation of new VMs incurs considerable waste of computing resources as each VM consumes resources which could not directly used by the applications. More subtle changes, such as modifying VMs capacity, can be conducted to achieve the desired effects with less running cost and, usually, in less time [92].

These limitations restrict the use of elasticity by diverse scientific applications, e.g., the ones that use multithreaded parallelism or have phases with distinct demands of memory and I/O. For example, Moltó et al. [93] present a mechanism for dynamic allocation of memory for scientific applications in a private cloud. The replication of the presented experiments are not possible in most current public clouds, since is not possible to change the memory amount of a VM. In the work of Galante and Bona [94], a mechanism for providing elasticity to OpenMP (multithreaded) applications is presented. Similarly, the experiments, executed in a private cloud, are not supported in most public clouds, considering that dynamic allocation of VCPUs is not supported.

4.5 Spin-up and Spin-down Times

The great advantage of the elasticity is the ability to dynamically provide resources in response to a demand. Scaling speed is also an important factor that could smooth/restrain the ability for achieving the elasticity requirements [90]. 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, the resource allocation is instantaneous, i.e., there is no time delay between detecting load changes and changing resourcing levels [95]. However, in real world clouds, the startup time can vary (ranging from 1 to 10 minutes), depending on many 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 addressing highly dynamic workloads.

Li et al. [58] measured allocation latency and booting latency of the smallest instances, both Windows and Linux OS, for three providers including Amazon server instances. They found that all providers have average allocation latency below the 10 minutes and Windows instances take longer than Linux instances to be created or booted. A similar experiment was presented by Mao and Humphrey [96], where they evaluated the average VM spin-up time on Amazon EC2 (m1.small), Azure (Small) and Rackspace

(Type IV) instances. The results of this experiment are presented in Table 5.

This large spin-up time makes prohibitive the use of elasticity by high performance applications. For example, MPI-2 provides an interface that allows the creation of processes during the execution of a MPI program, and the communication by message passing. This feature could be combined to elasticity to enable the dynamic creation of VMs according to new process instantiation. However, if the spin-up time is high, the performance of application may be affected, since the application blocks waiting for the new VM.

In turn, *spin-down time* is the interval between no longer requiring a resource and no longer paying for it [89], 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 [77].

5 Towards Scientific Elastic Applications

After evaluating the challenges exposed in Section 4, we point some possibilities of using the elasticity in scientific applications, and describe some solutions that are being developed to overcome the challenges. Thus, below we reanalyze each mentioned problem in detail, emphasizing here the possible solutions. At a glance, the challenges and perspectives of elasticity for scientific applications are summarized in Table 6.

5.1 Inappropriate Elasticity Mechanisms

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.

In the PaaS context, we highlight the functioning of the AutoElastic middleware which self-organizes the number of virtual machines without any user intervention [97]. Using reactive and horizontal techniques to enable elasticity, AutoElastic employs a source-to-source translation to transfer a non-elastic application into an elastic one. To accomplish this, the user must not insert any elasticity code in his/her application, but it must be modeled carefully as a master-slave iterative application. Figure 3 illustrates the pseudo code of the master and the slave in the parts (a) and (b), while part (c) presents the elasticity code to be inserted between lines 2 and 3 of the master code. The use of a loop-based application is pertinent to avoid an inconsistent global state during the application's execution: process and resource reorganization is taken either in the beginning or in the ending of the loop. An AutoElastic application can be developed using MPI 2.0, which offers a Sockets-based point-

Table 5 Average VM spin-up time. Adapted from Mao et al. [96]

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.

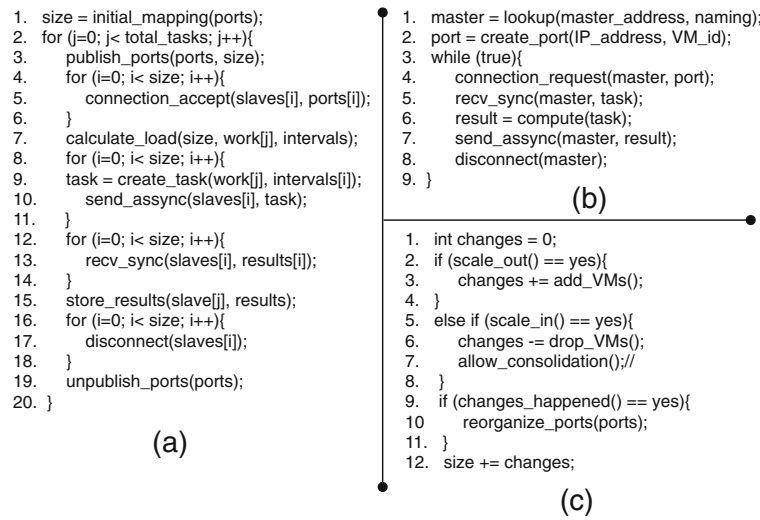
Table 6 Elasticity: challenges and possibilities

Challenge	Possibilities	Related Works
Inappropriate elasticity mechanisms	1.1– Use of PaaS and MapReduce for new applications; 1.2– Use of PaaS source-to-source translators transforming a master-slave iterative non-elastic application into an elastic one; 1.3– Workflows can be ported to clouds and adapted to use the cloud elasticity; 1.4– Development of new tools and frameworks focusing different application models;	[12, 13, 16, 24, 36, 80, 97–106]
Database Scalability	2.1– Use of Non-SQL databases; 2.2– Development of cloud-friendly relational databases;	[82, 83, 107–110]
Resources availability and Cloud interoperability	3.1– Creation standards for cloud interoperability; 3.2– Cloud Federation;	[21, 86, 88, 111–118]
Limited resources granularity	4.1– Offering of replication and resizing of cloud resources for processing, memory, storage and networking 4.2– Resources-as-a-Service (RaaS) model 4.3– Use of container virtualization	[71–73, 91]
Spin-up and spin-down time	5.1– Use of new virtualization techniques to speed up the virtual resources provisioning process 5.2– Changing providers billing policy to allow the use a fine-grained cost model which only charges for consumed resources 5.3– Asynchronous elasticity mechanism, in which the application is not blocked during scale out operations	[72, 95, 97, 119–122]

to-point communication directives to provide dynamic process creation paradigm. Finally, AutoElastic can be deployed over any public cloud infrastructure since both inserting the AutoElastic translator/compiler in the PaaS level and compiling the user application against it.

Another interesting approach is the use of the MapReduce paradigm [98], 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. A user not only can launch many servers at the beginning, but can also launch additional servers in the middle of computation [13, 16]. The new servers can automatically figure out the current job progress and poll the queues for work to process. Previous works 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 [99, 100]. Some public cloud providers supports MapReduce applications. An example is the service called Amazon Elastic MapReduce that enables running MapReduce applications directly on the cloud [24]. It uses a hosted Hadoop framework running on the web-scale infrastructure of Amazon EC2 and Amazon S3.

In the case of legacy applications, scientific workflows is an example of approach that can benefit with



```
1.  size = initial_mapping(ports);
2.  for (j=0; j< total_tasks; j++){
3.      publish_ports(ports, size);
4.      for (i=0; i< size; i++){
5.          connection_accept(slaves[i], ports[i]);
6.      }
7.      calculate_load(size, work[j], intervals);
8.      for (i=0; i< size; i++){
9.      task = create_task(work[j], intervals[i]);
10.         send_assync(slaves[i], task);
11.     }
12.     for (i=0; i< size; i++){
13.         recv_sync(slaves[i], results[i]);
14.     }
15.     store_results(slave[j], results);
16.     for (i=0; i< size; i++){
17.         disconnect(slaves[i]);
18.     }
19.     unpublish_ports(ports);
20. }
```

(a)

```
1.  master = lookup(master_address, naming);
2.  port = create_port(IP_address, VM_id);
3.  while (true){
4.      connection_request(master, port);
5.      recv_sync(master, task);
6.      result = compute(task);
7.      send_assync(master, result);
8.      disconnect(master);
9.  }
```

(b)

```
1.  int changes = 0;
2.  if (scale_out() == yes){
3.      changes += add_VMs();
4.  }
5.  else if (scale_in() == yes){
6.      changes -= drop_VMs();
7.      allow_consolidation();//
8.  }
9.  if (changes_happened() == yes){
10      reorganize_ports(ports);
11. }
12.  size += changes;
```

(c)

Fig. 3 A master-slave iterative programming model of the AutoElastic [97] initiative: (**a**) Master process; (**b**) Slave process; (**c**) elasticity code to be inserted in the Master process at PaaS level by using either method overriding, source-to-source translation or wrapper technique

cloud elasticity [101]. 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 [102]. In this context, cloud providers have recognized the importance of workflow applications to science and provide their own native solutions, such as the Amazon Simple Workflow Service (SWF) [24]. Platforms and frameworks for elastic execution of workflows are being developed in academy as well. Lin and Lu [17] propose SCPOR, a scientific workflow scheduling algorithm that is able to schedule workflows in need of elastically changing compute resources.Yu et al. [123] developed MWPC, an architecture for execution of elastic workflows on heterogeneous distributed computing resources. More recently, Leslie et al. [18] present DEWE, a framework for the distributed, elastic execution of scientific workflows. DEWE enables scientists to design algorithms with the ability to shape the underlying execution environment during runtime.

Other legacy scientific applications (e.g., MPI, multithreaded) rely on IaaS cloud services and solely use static execution modes, in which an instance of VM is perceived as a cluster node [41]. 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.

ElasticMPI offers elasticity for MPI applications by stop-and-relaunching the application with a newer resource configuration [80]. The system assumes that the user knows in advance the expected conclusion time for each phase of the program. The monitoring system can detect that the current configuration cannot fulfill the given deadline and adds more virtual instances. Vectors and data structures are redistributed and the execution continues from the last iteration. Applications that do not have an iterative loop cannot be adapted by the framework, since it uses the iteration index as execution restarting point. Furthermore, the approach of ElasticMPI imposes changes in the application source code by inserting monitoring directives. Yet programming with MPI, SpotMPI is a toolkit to facilitate the execution of real MPI applications on volatile auction-based cloud platforms (spot instances) [12]. The toolkit provides optimal checkpointing intervals and restarting of applications after out-of-bid situations through calculations of the density of out-of-bid failures from price history.

Rajan et al. [105] presented Work Queue, a framework for the development of master-slave elastic applications. Applications developed using Work Queue allow adding slave replicas at runtime. The slaves are implemented as executable files that can be instantiated by the user on different machines on demand. When executed, the slaves communicate with the master, that coordinates task execution and the data exchange.

Moltó et al. [93] developed an architecture for dynamic memory allocation for scientific applications. The authors focused on dynamic memory

management to automatically fit at runtime the underlying computing infrastructure to the application, thus adapting the memory size of the VM to the memory consumption pattern of the application. The architecture uses the VM memory usage information to decide when to scale up or scale down.

Wottrich et al. [124] propose OpenMR, a execution model based on OpenMP and MapReduce that enables the usage of highly-parallel, distributed machine clusters while automatically providing fault tolerance and workload balancing. Since OpenMR is built upon MapReduce, the elasticity solutions developed for MapReduce are also available to OpenMR.

Galante and Bona [94] also propose a solution to provide elasticity for OpenMP. In this solution, the OpenMP directives are extended to support the automatic adjustment of the number of VCPUs according to the amount of threads in execution. These elasticity-aware directives can automatically control elasticity, hiding the complexity of writing and executing elasticity strategies of the user. In addition, some routines were added to user-level library, targeting to provide a more precise control over the elastic execution. The solution also includes support for elastic memory allocation, taking advantage of the ballooning technique available in most of modern hypervisors.

A more generic platform is proposed by Caballer et al. [125]. The CodeCloud platform supports the execution of scientific applications in different programming models (such as master-slave, MPI, MapReduce and workflows) on cloud infrastructures. The elasticity is automatic-reactive and is enabled by a set of rules that define the elasticity modes of the infrastructure during the execution of the application.

In a previous work, we presented Cloudine, a framework for development and execution of elastic scientific applications in clouds [106]. The framework focuses on parallel and distributed applications that runs directly over the VM operating system of IaaS clouds. In Cloudine, the elasticity control is based on the concept of elasticity primitives, which are basic functions inserted into source code that allow applications to perform requests for allocation or deallocation of resources. Cloudine supports C/C++ languages and provides a set of primitives for dynamic allocation of VCPUs, memory and virtual machines. The main benefit of this approach is the possibility of constructing tailor made elastic applications controllers based in the particularities of each application. It enables to include elasticity features to existing (or even legacy) applications, to extend functionalities of other frameworks and to add elasticity support to well-known parallel programming libraries.

Table 7 Elasticity mechanisms and features

Reference	Application type	Supported elasticity
Righi et al. [97]	MPI	automatic-reactive, horizontal
Chohan et al. [13]	MapReduce	manual, horizontal
Iordache et al. [16]	MapReduce	manual, horizontal
Lin and Liu [17]	workflow	automatic, horizontal
Leslie et al. [18]	workflow	automatic, horizontal
Raveendran et al. [80]	MPI	programmable, horizontal
Taifi et al. [12]	MPI	programmable, horizontal (spot instances)
Rajan et al. [105]	master-slave	manual, horizontal
Moltó et al. [93]	generic (memory)	automatic-reactive, vertical
Wottrich et al. [124]	OpenMP/MapReduce	manual, horizontal
Galante and Bona [94]	OpenMP	programmable, vertical
Caballer et al. [125]	master-slave / MPI / MapReduce / workflow	automatic-reactive, horizontal and vertical
Galante and Bona [106]	generic (memory, VCPU and VM)	programmable, horizontal and vertical

Table 7 summarizes the characteristics of the frameworks and platforms developed to provide elasticity to scientific applications.

5.2 Database Scalability

Considering that existing database management systems are not well suited for deployment in the cloud, an alternative is the use of NoSQL databases [83]. These databases have been designed to take advantage of large resource pools and provide high availability and high performance. Moreover, these databases were designed to cope with resource availability changes (elasticity) [107]. Google's Bigtable, HBase, Amazon's Dynamo, Facebook's Cassandra, and LinkedIn's Voldemort are a representative set of such systems [108].

Typically, these systems only support read and write operations on individual rows. Because of these limitations of NoSQL cloud data storage systems, applications that are built on top of these systems need to be much more complex. A particular disadvantage of such systems is that the task of maintaining consistency of the database is delegated to application developers, and is therefore highly error prone. Considering these issues, some works, such as [82] and [109], address the design of cloud friendly SQL DBMSes. In commercial field, the ScaleBase [110] provides scalability via elastic scale out/in, allowing continuously increase total MySQL database size and throughput to follow application workload requirements.

5.3 Resources Availability and Cloud Federation

The resources availability problem is closely related to the providers policies, but we believe that as demand grows, these limitations will be handled gradually. The potential of cloud resources is enormous and it became evident when a cluster composed by EC2 cluster instances (26,496 cores) was able to achieve 484.2 TeraFLOPS for the High Performance Linpack benchmark and totalizing 105,984 GB RAM, placing the cluster at 101^{st} position in the November 2014 Top500 list (64^{th} in November 2013).

As we said in Section 4, a possible solution to resources availability problem is the use of multiple clouds, but there is a lack of standards to enable *interoperability*. This fact has drawn attention of both the industry and the academia. The industry tries to address the cloud interoperability issues via standardization and many cloud interoperability standards have been proposed and even put into use in recent years.

For operations such as transferring a virtual machine image and data between providers, standardized formats for the data being transferred, billing, and identity management are needed. Some standards, such the Open Virtualization Format (OVF) [111], the Open Cloud Computing Interface (OCCI) [113] and the Cloud Data Management Interface (CDMI) [112], were developed. The OVF is a packaging standard initiated by distributed Task Force (DMTF) for deployment of virtual appliances. By abstracting a virtual appliance as a single atomic unit, it enables simplified deployment and migration of virtual appliances across multiple virtualization platforms, including IBM, Microsoft, VirtualBox, XenServer, SUSE Studio, OpenStack, and others. The OCCI community is an open community under the umbrella of the Open Grid Forum (OGF), with contributing members from industry and academia. The current OCCI specifications define a set of common standard interfaces of management for IaaS cloud interoperability. The CDMI, proposed by the Storage Networking Industry Association (SNIA), defines the functional interface that applications can use to create, retrieve, update and delete data elements from a cloud. It is the storage backbone in cloud interoperability.

Even considering the several efforts, it may take a couple years for the standards to be fully agreed upon and adopted, if ever. As practical, short-term solutions, researchers have been developing technologies to enable interoperation among clouds, from both the cloud provider's and user's perspectives [88]. Similarly, grids have devoted huge efforts to reach standardization both in the user interface and in the inner interfaces (for accessing resources), and so reach seamless interoperability. So, enhancing existing grid standards is a possibility to ensure the required interoperability [126].

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 demand needs. In this scenario, the exceeding demands of a cloud are ful-

filled by leasing available computational and storage capabilities from other cloud service providers [86, 127]. The development of fundamental techniques and software systems that integrate distributed clouds is critical to enabling composition and deployment of large scale elastic applications. In this sense, some architectures and platforms for cloud federation has been proposed, as presented in the works of Buyya et al. [114] and Yangui et al. [116].

Buyya et al. present InterCloud, a federated cloud computing environment with the objective of supporting the dynamic expansion or contraction of capabilities (VMs, services, storage, and database) for handling sudden variations in service demands. In the paper of Yangui et al., the authors present CompatibleOne, an open source broker, which provide interoperable middleware for the description and federation of heterogeneous clouds and resources provisioned by different cloud providers, allowing users and developers to combine different services available from different suppliers.

Related to the use of multiple clouds in the scientific context, we highlight two initiatives: the EU Brazil Cloud Connect [117] and the EGI Federated Cloud [118]. EU Brazil Cloud Connect is an international cooperation project aimed at accelerating scientific discovery to advance knowledge on several challenges of high social impact. The collaboration between European and Brazilian research centers enables the project leverages a set of components for the use of supercomputing, private cloud and cloud opportunistic resources. These resources are exposed through programming frameworks and scientific gateways, easing the adaptation and deployment of the applications that use data and computing resources.

The EGI Federated Cloud is a grid of academic private clouds and visualized resources, built around open standards and focusing on the requirements of the scientific community. Federated operations brings together the operational tools, processes and people necessary to guarantee standard operation of heterogeneous infrastructures from multiple independent providers, with a lightweight central coordination. This includes, monitoring, accounting, configuration and other services required to federate service provision for access by multiple research communities.

5.4 Limited Resources Granularity

The resources granularity issue is starting to be solved with the emergence of providers like Profitbricks [71], ElasticHosts [73] and CloudSigma [72] that allow its cloud consumers to customize their cloud servers by varying CPU, RAM, disk and data transfer/bandwidth capacity at very fine-grained levels. This feature is very valuable for real elasticity, since resources can be allocated more efficiently.

Ben-Yehuda et al. [91] proposes the Resource-as-a-Service (RaaS) model, 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 by frames; I/O is sold on the basis of I/O devices with bandwidth and latency guarantees.

Other approach is the use of containers. This kind of virtualization partitions the physical machines resources, creating multiple isolated user-space instances. ElasticHosts [73] uses this approach in the Elastic Containers service. This service is based in Linux Containers and enable the dynamic allocation of CPU, memory, storage and bandwidth resources. The resources management in container-based virtualization systems is normally done by Control Groups (cgroup), which restricts a resource usage for process groups. For example, using cgroups it is possible to limit/prioritize CPU, memory and I/O usage for different containers [128].

5.5 Spin-up and Spin-down Times

A possible solution for both spin-up and spin-down times consists of using new virtualization techniques and changing providers billing policy. Some works [119–121] 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 [95], 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 consumed: the byte transmitted, the byte stored, and the millisecond of processing time. Currently, the closest to the ideal charging scheme is offered by Google Compute Engine, in which all machine types are charged a minimum of 10 minutes and after this period, instances are charged in 1 minute increments, rounded up to the nearest minute [122]. Other interesting billing model is proposed by CloudSigma [72], in which customers are charged every 5 minutes for storage, memory and CPU separately, and with a degree of granularity not always offered by competitors.

Besides working at virtualization technique or billing level, another approach attacks the construction of the application and how new resources are delivered to it. In this way, AutoElastic [97] provides a framework to control the elastic execution of iterative master-slave applications, enabling to them a feature named Asynchronous Elasticity. The main idea is not blocking the application at any scaling out operation. To accomplish this, AutoElastic works with an elasticity manager and shared data area. Particularly, this data area is useful to control elasticity by enabling communications between the master and the manager. Figure 4 depicts the functioning of the AutoElastic components to enable the Asynchronous Elasticity. At implementation level, the Autoelastic Manager is implemented outside the cloud and uses the binary SCP from the SSH package to obtain or place files from/to the cloud front-end. These files are in a shared folder that is enabled with NFS and represents elasticity actions and process data. The authors follow this implementation because the AutoElastic Manager cannot access the NFS directly, unless it is placed inside the cloud. Finally, Asynchronous Elasticity is not dependent of a cloud provider technology, being useful on any public cloud provider.

6 Final Remarks

This article contributes as starting point for those who are porting their applications to the cloud, and expect to explore the most important feature found in this environment type: elasticity. We presented an overview of current elasticity mechanisms

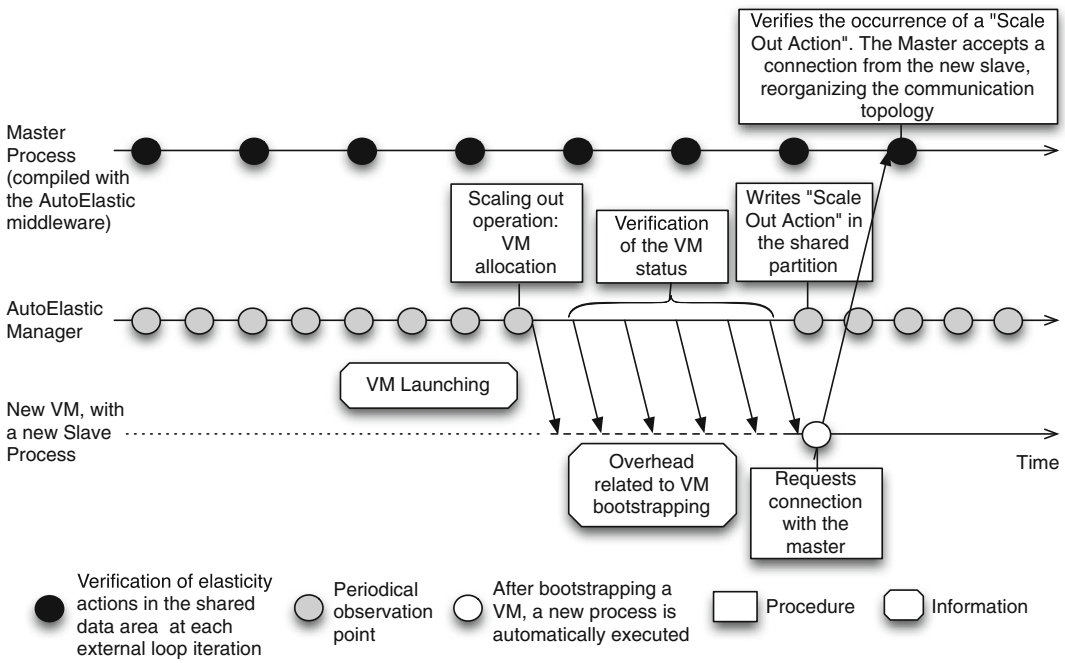


Fig. 4 Functioning of the master, the new slave and the AutoElastic Manager to enable the Asynchronous Elasticity [97]

and discussed about the current limitations and the possibilities towards using elasticity in scientific applications.

According to the problem statement presented in Section 1, we argue that the current computational clouds have not yet reached the necessary maturity level to meet all scientific applications requirements. *So, what do we need to achieve this?*

First, we need real public elastic clouds. In these elastic clouds, it would be possible to allocate customized virtual environments, where the user can configure each component independently. Similarly, the billing scheme will be defined by an extremely fine-grained cost model, which only charges for resources that are consumed. The use of vertical and horizontal elasticity is also essential, making possible to allocate from a single vCPU to a cluster with hundreds or thousands of VMs. The supply of large-scale resources will be possible by using multiple cloud infrastructures through unified interfaces and standards.

In practice, the issues related to customized environments, billing model and use of vertical and horizontal elasticity are already addressed by few cloud providers, such as CloudSigma and Profitbricks. However, in most major players the resources are allocated in terms of VMs, only horizontal elasticity is provided, and not-so-fair billing models are employed. In addition, the federation and use of multiple clouds to provide large-scale resources is also a reality, as described in the EU Brazil Cloud Connect and EGI Federated Cloud initiatives (Section 5.3). Although there are several task forces and researches addressing this issue, cloud interoperability is not currently implemented commercially.

To take full advantage of the elasticity provided by clouds, it is necessary more than just an elastic infrastructure, we need elastic applications as well. Our study concludes that there are already scientific application models (including MapReduce, workflows and iterative master-slave applications), that can either immediately benefit or easily adapted to take profit from the elasticity on clouds. Particularly, these models are characterized by having data locality, loosely coupled, high throughput or fault tolerant, fitting better the current cloud model. On the other hand, tightly-coupled applications will need to be re-engineered to realize the full benefits of the new computing capacity that is now available on demand.

Note that simply moving applications to the cloud is usually not sufficient to take advantage of elasticity. We need frameworks and programming interfaces to transform into reality the HPC scientific applications and cloud elasticity duet, and, although many cloud providers offer elasticity mechanisms, few focus on scientific applications. Solutions as those presented in Section 5.1 could be offered by public clouds to provide elasticity to applications with different characteristics and demands.

To conclude, we expect that in the coming years the efforts being taken by numerous researchers in this area identify and address these challenges and lead to better and more mature technologies that will improve the current cloud computing practices. In these efforts, we believe that a good knowledge of existing technologies, techniques and architectures such as those developed in the fields of HPC and grid computing will greatly help accelerating the pace of research and development of the cloud.

References

1. Villamizar, M., Castro, H., Mendez, D.: E-Clouds: a Saas Marketplace for Scientific Computing. In: Proceedings IEEE/ACM 5th International Conference on Utility and Cloud Computing, pp. 13–20. IEEE (2012)
2. Simmhan, Y., Van Ingen, C., Subramanian, G., Li, J.: Bridging the Gap between Desktop and the Cloud for Escience Applications. In: Proceedings IEEE 3rd International Conference on Cloud Computing, pp. 474–481. IEEE (2010)
3. Vecchiola, C., Pandey, S., Buyya, R.: High-Performance Cloud Computing: a View of Scientific Applications. In: Proceedings 10th International Symposium on Pervasive Systems Algorithms, and Networks, pp. 4–16. IEEE (2009)
4. Ramakrishnan, L., Jackson, K.R., Canon, S., Cholia, S., Shalf, J.: Defining Future Platform Requirements for e-Science Clouds. In: Proceedings 1st ACM Symposium on Cloud Computing, pp. 101–106. ACM (2010)
5. Herbst, N.R., Kounev, S., Reussner, R.: Elasticity in Cloud Computing: What It Is, and What It is Not. In: Proceedings 10th International Conference on Autonomic Computing, pp. 23–27. USENIX (2013)
6. Galante, G., Bona, L.C.E.: A Survey on Cloud Computing Elasticity. In: Proceedings IEEE/ACM 5th International Conference on Utility and Cloud Computing, pp. 263–270. IEEE (2012)
7. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.: A review of auto-scaling techniques for elastic applications in cloud environments. J. Grid Comput. **12**(4), 559–592 (2014)

8. Chieu, T.C., Mohindra, A., Karve, A.A., Segal, A.: Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment. In: Proceedings IEEE International Conference on e-Business Engineering, pp. 281–286. IEEE (2009)
9. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, a., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.onaharia., M.: A view of cloud computing. Commun. ACM **53**(4) (2010)
10. Wang, L., Zhan, J., Shi, W., Liang, Y.: Cloud, can scientific communities benefit from the economies of scale? IEEE Trans. Parallel Distrib. Syst. **23**(2), 296–303 (2012)
11. Oliveira, D., Ogasawara, E.: Is cloud computing the solution for brazilian researchers? Intl. J. Comput. Appl. **6**(8), 19–23 (2010)
12. Taifi, M., Shi, J.Y., Khreishah, A.: SpotMPI: a Framework for Auction-Based HPC Computing Using Amazon Spot Instances. Springer-Verlag (2011)
13. Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., Krintz, C.: See Spot Run: Using Spot Instances for Mapreduce Workflows. In: Proceedings 2nd USENIX Conference on Hot Topics in Cloud Computing. USENIX (2010)
14. Vo, H.T., Chen, C., Ooi, B.C.: Towards elastic transactional cloud storage with range query support. Proc. VLDB Endowment **3**(1-2), 506–514 (2010)
15. Nicolae, B., Riteau, P., Keahey, K.: Bursting the Cloud Data Bubble: Towards Transparent Storage Elasticity in Iaas Clouds. In: 2014 IEEE 28th Intl Parallel and Distributed Processing Symposium, pp. 135–144. IEEE (2014)
16. Iordache, A., Morin, C., Parlavantzas, N., Riteau, P.: Resilin: Elastic MapReduce over Multiple Clouds. In: Technical Report RR-8081, INRIA, Rennes, France (2012)
17. Lin, C., Lu, S.: SCPOR: an Elastic Workflow Scheduling Algorithm for Services Computing. In: Proceedings the 5th IEEE International Conference on Service-Oriented Computing and Applications, pp. 1–8. SOCA, IEEE (2011)
18. Leslie, L., Sato, C., Lee, Y., Jiang, Q., Zomaya, A.: DEWE: a Framework for Distributed Elastic Scientific Workflow Execution. In: 13th Australasian Symp. on Parallel and Distributed Computing, pp. 3–10. AusPDC, ACS (2015)
19. Oliveira, D., Baio, F.A., Mattoso, M.: Migrating Scientific Experiments to the Cloud. http://www.hpcinthecloud.com/hpccloud/2011-03-04/migrating_scientific_experiments_to_the_cloud.html (2 july 2015, last accessed)
20. Truong, H., Dustdar, S.: Cloud computing for small research groups in computational science and engineering: Current status and outlook. Computing **91**(1), 75–91 (2011)
21. Helix Nebula - The Science Cloud. http://www.helix-nebula.eu/ (02 july 2015, last accessed)
22. Science Clouds. http://scienceclouds.org (02 july 2015, last accessed)
23. CloudLab. http://www.cloudlab.us (02 july 2015, last accessed)
24. Amazon Web Services. http://aws.amazon.com (02 july 2015, last accessed)
25. Sabalcore. http://www.sabalcore.com (20 july 2014, last accessed)
26. e-Science Central. http://www.esciencecentral.co.uk/ (07 july 2015, last accessed)
27. Calheiros, R.N., Vecchiola, C., Karunamoorthy, D., Buyya, R.: The aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds. Future Gener. Comput. Syst. **28**(6), 861–870 (2011)
28. Marshall, P., Keahey, K., Freeman, T.: Elastic Site: Using Clouds to Elastically Extend Site Resources. In: Proceedings 10th IEEE/ACM International Conference on Cluster Cloud and Grid Computing, pp. 43–52. IEEE (2010)
29. Bicer, T., Chiu, D., Agrawal, G.: A Framework for Data-Intensive Computing with Cloud Bursting. In: Proceedings Conference on High Performance Computing Networking, Storage and Analysis Companion, pp. 5–6. ACM (2011)
30. Calatrava, A., Moltó, G., Hernandez, V.: Combining Grid and Cloud Resources for Hybrid Scientific Computing Executions. In: Proceedings 3rd International Conference on Cloud Computing Technology and Science, pp. 494–501. IEEE (2011)
31. Mateescu, G., Gentzsch, W., Ribbens, C.J.: Hybrid Computing-Where HPC meets grid and cloud computing. Future Gener. Comput. Syst. **27**(5), 440–453 (2011)
32. He, Q., Zhou, S., Kobler, B., Duffy, D., Mcglynn, T.: Case Study for Running HPC Applications in Public Clouds. In: Proceedings 19th ACM International Symposium on High Performance Distributed Computing, pp. 395–401. ACM (2010)
33. Li, J., Humphrey, M., Cheah, Y., Ryu, Y., Agarwal, D., Jackson, K., Ingen, C.: Fault Tolerance and Scaling in E-Science Cloud Applications: Observations from the Continuing Development of MODISAzure. In: Proceedings 6Th International Conference on e-Science, pp. 246–253. IEEE (2010)
34. Edlund, A., Koopmans, M., Shah, Z.A., Livenson, I., Orellana, F., Kommeri, J., Tuisku, M., Lehtovuori, P., Hansen, K.M., Neukirchen, H., Hvannberg, E.: Practical Cloud Evaluation from a Nordic Escience User Perspective. In: Proceedings 5th International Workshop on Virtualization Technologies in Distributed Computing, pp. 29-38. ACM (2011)
35. Mauch, V., Kunze, M., Hillenbrand, M.: High performance cloud computing. Future Gener. Comput. Syst. **29**(6), 1408–1416 (2013)
36. Vöckler, J.S., Juve, G., Deelman, E., Rynge, M., Berriman, B.: Experiences Using Cloud Computing for a Scientific Workflow Application. In: Proceedings 2nd International Workshop on Scientific Cloud Computing, pp. 15-24. ACM (2011)
37. Keller, M., Meister, D., Brinkmann, A., Terboven, C., Bischof, C.: eScience Cloud Infrastructure. In: Proceedings 37th Euromicro Conference on Software Engineering and Advanced Applications, pp. 188–195. IEEE (2011)
38. CERN: CERNVM. http://cernvm.cern.ch/portal (15 january 2015, last accessed)

39. Hellerstein, J.L., Kohlhoff, K.J., Konerding, D.E.: Science in the Cloud: Accelerating Discovery in the 21st Century. IEEE Internet Comput. **16**(4), 64–68 (2012)
40. Sakr, S., Liu, A., Batista, D.M., Alomari, M.: A survey of large scale data management approaches in cloud environments. IEEE Commun. Surv. Tutorials **13**(3), 311–336 (2011)
41. 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. Wiley (2011)
42. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Elastic Management of Cluster-Based Services in the Cloud. In: Proceedings the 1st Workshop on Automated Control for Datacenters and Clouds, pp. 19–24. ACDC, ACM (2009)
43. Nie, L., Xu, Z.: An Adaptive Scheduling Mechanism for Elastic Grid Computing. In: 5th International Conference on Semantics, Knowledge and Grid, pp. 184-191. SKG (2009)
44. Bientinesi, P., Iakymchuk, R., Napper, J.: HPC On competitive cloud resources. In: Furht, b., Escalante, A. (eds.) Handbook of Cloud Computing. Springer (2010)
45. Evangelinos, C., Hill, C.N.: Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. ACM (2008)
46. Rehr, J.J., Vila, F.D., Gardner, J.P., Svec, L., Prange, M.: Scientific computing in the cloud. Comput. Sci. Eng. **12**(3), 34–43 (2010)
47. Gupta, A., Milojicic, D.: Evaluation of HPC applications on cloud. Technical report HPL-2011-132, HP laboratories, palo alto USA (2011)
48. Church, P., Goscinski, A.: Iaas Clouds Vs. Clusters for HPC: a Performance Study. In: Proceedings 2nd International Conference on Cloud Computing, GRIDs, and Virtualization, pp. 39–45. IARIA (2011)
49. VMWare vSphere. http://www.vmware.com/products/vsphere (02 july 2015, last accessed)
50. Simons, J.E., Buell, J.: Virtualizing high performance computing. ACM Oper. Syst. Rev. **44**(4), 136–145 (2010)
51. Schad, J., Dittrich, J., Quiané-Ruiz, J.: Runtime measurements in the cloud: observing, Analyzing, and Reducing Variance. Proc. Very Large Database Endowment **3**(1-2), 460–471 (2010)
52. Phillips, S.C., Engen, V., Papay, J.: Snow White Clouds and the Seven Dwarfs. In: Proceedings 3rd IEEE International Conference on Cloud Computing Technology and Science, pp. 738–745. IEEE (2011)
53. Rego, P.A.L., Coutinho, E.F., Gomes, D.G., Souza, J.N.: FairCPU: Architecture for Allocation of Virtual Machines Using Processing Features. In: Proceedings 4th International Conference on Utility and Cloud Computing, pp. 371–376. IEEE (2011)
54. Nanath, K., Pillai, R.: A model for Cost-Benefit analysis of cloud computing. J. Intl. Technol. Inf. Manag. **22**(3), 93–117 (2013)
55. Negru, C., Cristea, V.: Cost models - pillars for efficient cloud computing: Position paper. Intl. J. Intell. Syst. Technol. Appl. **12**(1), 28–38 (2013)
56. Berriman, G.B., Juve, G., Vckler, J.S., Deelman, E., Rynge, M.: The application of cloud computing to scientific workflows: a study of cost and performance. Proc. Royal Soc. Assoc. **371**(1983), 1–14 (2012)
57. Fox, G., Gannon, D.: Using Clouds for Technical Computing. In: Catlett, C., Gentzsch, W., Grandinetti, L., Joubert, G., Vazquez-Poletti, J. (eds.) Cloud Computing and Big Data, pp. 81–102. IOS Press (2013)
58. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: Comparing Public Cloud Providers. In: Proceedings 10th Annual Conference on Internet Measurement, pp. 1–14. ACM (2010)
59. Xiaotao, Y., Aili, L., Lin, Z.: Research of High Performance Computing with Clouds. In: Proceedings 3rd International Symposium on Computer Science and Computational Technology, pp. 289–293. Academy Publisher (2010)
60. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards Trusted Cloud Computing. In: Proceedings Conference on Hot Topics in Cloud Computing. USENIX (2009)
61. Zissis, D., Lekkas, D.: Addressing cloud computing security issues. Future Gener. Comput. Syst. **28**(3), 583–592 (2012)
62. Chen, W., Deelman, E.: Integration of Workflow Partitioning and Resource Provisioning. In: Proceedings 12Th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 764–768. IEEE (2012)
63. Thakar, A., Szalay, A., Church, K., Terzis, A.: Large Science Databases Are Cloud Services Ready for Them? Sci. Program. **19**(2-3), 147–159 (2011)
64. Yuan, D., Yang, Y., Liu, X., Chen, J.: A data placement strategy in scientific cloud workflows. Future Gener. Comput. Syst. **26**(8), 1200–1214 (2010)
65. Lin, J.W., Chen, C.H.: Interference-aware virtual machine placement in cloud computing systems. In: International Conference on Computer Information Science. Volume 2 of ICCIS, pp. 598–603 (2012)
66. Gupta, A., Milojicic, D., Kalé, L.V.: Optimizing VM Placement for HPC in the Cloud. In: Proceedings the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, pp. 1–6. FederatedClouds, ACM (2012)
67. Canali, C., Lancellotti, R.: Automatic Virtual Machine Clustering Based on Bhattacharyya Distance for Multi-cloud Systems. In: Proceedings the 2013 International Workshop on Multi-cloud Applications and Federated Clouds, pp. 45–52. MultiCloud, ACM (2013)
68. Rackspace. http://www.rackspace.com (15 january 2015, last accessed)
69. GoGrid. http://www.gogrid.com (15 january 2015, last accessed)
70. Joyent. http://joyent.com (02 july 2015, last accessed)
71. Profitbricks. https://www.profitbricks.com (02 july 2015, last accessed)
72. CloudSigma. https://www.cloudsigma.com/ (02 july 2015, last accessed)
73. CloudSigma. http://www.elastichosts.com (02 july 2015, last accessed)
74. RightScale. http://www.rightscale.com (15 january 2015, last accessed)

75. Caron, E., Desprez, F., Rodero-Merino, L., Muresan, A.: Auto-scaling, Load Balancing and Monitoring in Commercial and Open-Source Clouds. In: Wang, L., Ranjan, R., Chen, J., Benatallah, B. (eds.) Cloud Computing: Methodology, Systems, and Applications. Taylor and Francis Group (2011)
76. Google App Engine. http://code.google.com/appengine (26 june 2015, last accessed)
77. Microsoft Azure. http://www.windowsazure.com (15 january 2015, last accessed)
78. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. Comput. Commun. Rev. **41**(1), 45–52 (2011)
79. Roy, N., Dubey, A., Gokhale, A.: Efficient Autoscaling in the Cloud Using Predictive Models forWorkload Forecasting. In: Proceedings IEEE 4th Interntional Conference on Cloud Computing, pp. 500–507. IEEE (2011)
80. Raveendran, A., Bicer, T., Agrawal, G.: A Framework for Elastic Execution of Existing MPI Programs. In: Proceedings IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, pp. 940–947. IEEE (2011)
81. Das, S., Agrawal, D., El Abbadi, A.: ElasTraS: An Elastic Transactional Data Store in the Cloud. In: Proceedings Conference on Hot Topics in Cloud Computing, pp. 1–5. USENIX (2009)
82. Agrawal, D., El Abbadi, A., Das, S., Elmore, A.J.: Database Scalability, Elasticity, and Autonomy in the Cloud. In: Proceedings the 16th International Conference on Database Systems for Advanced Applications, pp. 2–15. DASFAA, Springer (2011)
83. Pokorny, J.: NoSQL Databases: A Step to Database Scalability in Web Environment. In: Proceedings 13th International Conference on Information Integration and Web-based Applications and Services, pp. 278–283. ACM (2011)
84. Villegas, D., Rodero, I., Fong, L., Bobroff, N., Liu, Y., Parashar, M., Sadjadi, S.: The Role of Grid Computing Technologies in Cloud Computing. In: Furht, B., Escalante, A. (eds.) Handbook of Cloud Computing. Springer (2010)
85. 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 TR-3, Federal University of Campina Grande, Campina Grande, Brazil (2010)
86. Petcu, D.: Consuming resources and services from multiple clouds. J. Grid Comput. **12**(2), 321–345 (2014)
87. Papazoglou, M.P., Vaquero, L.M.: Knowledge-Intensive Cloud Services: Transforming the Cloud Delivery Stack. In: Kantola, J., Karwowski, W. (eds.) Knowledge Service Engineering Handbook. CRC Press (2012)
88. Zhang, Z., Wu, C., Cheung, D.W.: A survey on cloud interoperability: taxonomies, Standards, and Practice. SIGMETRICS Perform. Eval. Rev. **40**(4), 13–22 (2013)
89. 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, Sydney, Australia (2011)
90. Suleiman, B., Sakr, S., Jeffery, R., Liu, A.: On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. J. Internet Serv. Appl. **3**(2), 173–193 (2012)
91. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The Resource-as-a-service (RaaS) Cloud. In: Proceedings 4th USENIX Conference on Hot Topics in Cloud Computing. USENIX (2012)
92. Han, R., Guo, L., Ghanem, M.M., Guo, Y.: Lightweight Resource Scaling for Cloud Applications. In: Proceedings the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 644–651. CCGRID, IEEE (2012)
93. Moltó, G., Caballer, M., Romero, E., De Alfonso, C.: Elastic Memory Management of Virtualized Infrastructures for Applications with Dynamic Memory Requirements. Proced. Comput. Sci. **18**(0), 159–168 (2013)
94. Galante, G., Bona, L.C.E.: Supporting Elasticity in OpenMP Applications. In: Proceedings 22th Euromicro Conference on Parallel, Distributed and Network-Based Processing. IEEE (2014)
95. Brebner, P.C.: Is Your Cloud Elastic Enough?: Performance Modelling the Elasticity of Infrastructure As a Service (IaaS) Cloud Applications. In: Proceedings 3rd ACM/SPEC International Conference on Performance Engineering, pp. 263?-266. ACM (2012)
96. Mao, M., Humphrey, M.: A Performance Study on the VM Startup Time in the Cloud. In: Proceedings 5th IEEE International Conference on Cloud Computing, pp. 423–430. IEEE (2012)
97. Righi, R., Rodrigues, V., Andre da Costa, C., Galante, G., Bona, L., Ferreto, T.: Autoelastic: automatic resource elasticity for high performance applications in the cloud. IEEE Trans. Cloud Comput. **PP**(99), 1–1 (2015)
98. Srirama, S.N., Jakovits, P., Vainikko, E.: Adapting scientific computing problems to clouds using MapReduce. Future Gener. Comput. Syst. **28**(1), 184–192 (2012)
99. Bunch, C., Drawert, B., Norman, M.: MapScale: A Cloud Environment for Scientific Computing. Technical report. University of California, Santa Barbara, USA (2009)
100. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
101. Pandey, S., Karunamoorthy, D., Buyya, R.: Workflow Engine for Clouds. In: Buyya, R., Broberg, J., Goscinski, A.M. (eds.) Cloud Computing: Principles and Paradigms. Wiley (2011)
102. 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)
103. 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 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 606–611. IEEE (2010)
104. Kranjc, J., Podpečan, V., Lavrač, N.: ClowdFlows: A Cloud Based Scientific Workflow Platform. In: Proceedings European Conference on Machine Learning and Knowledge Discovery in Databases-Part II, pp. 816–819. Springer (2012)

105. Rajan, D., Canino, A., Izaguirre, J.A., Thain, D.: Converting a High Performance Application to an Elastic Cloud Application. In: Proceedings 3rd IEEE International Conference on Cloud Computing Technology and Science, pp. 383–390. IEEE (2011)
106. 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.) Proceedings 13th International Conference on Computational Science and Its Applications Volume 5, Lecture Notes in Computer Science. Springer (2013)
107. Cruz, F., Maia, F., Matos, M., Oliveira, R., Paulo, J.a., Pereira, J., Vilaça, R.: MeT: Workload Aware Elasticity for NoSQL. In: Proceedings 8th ACM European Conference on Computer Systems, pp. 183–196. ACM (2013)
108. Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., Koziris, N.: On the Elasticity of NoSQL Databases over Cloud Management Platforms. In: Proceedings 20th ACM International Conference on Information and Knowledge Management, pp. 2385–2388. ACM (2011)
109. Minhas, U.F.: Scalable and Highly Available Database Systems in the Cloud. PhD thesis, University of Waterloo, Ontario, Canada (2013)
110. ScaleBase. http://www.scalebase.com/ (02 july 2015, last accessed)
111. DMTF: Open Virtualization Format. http://www.dmtf.org/standards/ovf (02 july 2015, last accessed)
112. SNIA: Cloud Data Management Interface. http://www.snia.org/cdmi (02 july 2015, last accessed)
113. OGF: Open Cloud Computing Interface. http://occi-wg.org/ (02 july 2015, last accessed)
114. Buyya, R., Ranjan, R., Calheiros, R.N.: Inter-Cloud: Utility-oriented Federation of Cloud Computing Environments for Scaling of Application Services. In: Proceedings 10th International Conference on Algorithms and Architectures for Parallel Processing, pp. 13–31. Springer (2010)
115. Villegas, D., Bobroff, N., Rodero, I., Delgado, J., Liu, Y., Devarakonda, A., Fong, L., Masoud Sadjadi, S., Parashar, M.: Cloud federation in a layered service model. J. Comput. Syst. Sci. **78**(5), 1330–1344 (2012)
116. Yangui, S., Marshall, I.J., Laisne, J.P., Tata, S.: Compatibleone: The open source cloud broker. J. Grid Comput. **12**(1), 93–109 (2014)
117. EU Brazil Cloud Connect . http://www.eubrazilcloudconnect.eu/ (28 june 2015, last accessed)
118. European Grid Infrastructure. https://www.egi.eu/infrastructure/cloud/ (28 june 2015, last accessed)
119. Zhu, J., Jiang, Z., Xiao, Z.: Twinkle: A Fast Resource Provisioning Mechanism for Internet Services. In: Proceedings 30th IEEE International Conference on Computer Communications, pp. 802–810. IEEE (2011)
120. Tang, C.: A High-Performance Virtual Machine Image Format for Cloud. In: Proceedings USENIX Technical Conference. USENIX (2011)
121. De, P., Gupta, M., Soni, M., Thatte, A.: Caching VM Instances for Fast VM Provisioning: A Comparative Evaluation. In: Proceedings 18th International Conference on Parallel Processing, pp. 325–336. Springer (2012)
122. Google Compute Engine. https://cloud.google.com/compute/ (26 june 2015, last accessed)
123. Yu, L., Thain, D.: Resource Management for Elastic Cloud Workflows. In: Proceedings the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 775–780. CCGRID, IEEE (2012)
124. Wottrich, R., Azevedo, R., Araujo, G.: Cloudbased OpenMP Parallelization Using a MapReduce Runtime. In: 26th IEEE International Symposium on Computer Architecture and High Performance Computing, pp. 334–341. SBAC-PAD, IEEE (2014)
125. Caballer, M., De Alfonso, C., Molt, G., Romero, E., Blanquer, I., Garca, A.: CodeCloud: A platform to enable execution of programming models on the Clouds. J. Syst. Softw. **93**(0), 187–198 (2014)
126. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: Towards a cloud definition. SIGCOMM Comput. Commun. Rev. **39**(1), 50–55 (2008)
127. Goiri, I., Guitart, J., Torres, J.: Characterizing Cloud Federation for Enhancing Providers' Profit. In: Proceedings the 2010 IEEE 3rd International Confernce on Cloud Computing. CLOUD, IEEE (2010)
128. Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T., De Rose, C.A.F.: Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. In: Proceedings the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233–240. PDP, IEEE (2013)