Next Generation HPC Clouds: A View for Large-Scale Scientific and Data-Intensive Applications

Dana Petcu[1], Horacio González–Vélez[2], Bogdan Nicolae[3], Juan Miguel García–Gómez[4], Elies Fuster–Garcia[5], and Craig Sheridan[6]

[1] West University of Timişoara, Romania
[2] National College of Ireland, Ireland
[3] IBM Research, Ireland
[4] Universitat Politècnica de Vàlencia, Spain
[5] Veratech for Health SL, Spain
[6] Flexiant, UK

Abstract. In spite of the rapid growth of Infrastructure-as-a-Service offers, support to run data-intensive and scientific applications large-scale is still limited. On the user side, existing features and programming models are insufficiently developed to express an application in such way that it can benefit from an elastic infrastructure that dynamically adapts to the requirements, which often leads to unnecessary over-provisioning and extra costs. On the provider side, key performance and scalability issues arise when having to deal with large groups of tightly coupled virtualized resources needed by such applications, which is especially challenging considering the multi-tenant dimension where sharing of physical resources introduces interference both inside and across large virtual machine deployments. This paper contributes with a holistic vision that imagines a tight integration between programming models, runtime middlewares and the virtualization infrastructure in order to provide a framework that transparently handles allocation and utilization of heterogeneous resources while dealing with performance and elasticity issues.

Keywords: heterogeneous clouds, cloud storage, HPC, data analytics.

1 Introduction

Data analytics and data-intensive scientific applications are undoubtedly a major driving force behind scientific advancement and business insight. Over the years, the increasing complexity of such applications has led to a rapid evolution of the computational infrastructure, to the point where massive computational facilities and data-centers are necessary in order to satisfy the computing and data storage needs. In a ceaseless quest for performance, a large variety of hardware devices and software stacks were developed to catch up with the increasing requirements. Unsurprisingly, such infrastructures are prohibitively expensive to own and maintain for the vast majority of users, which makes infrastructure clouds particularly appealing as an alternative, thanks to their pay-as-you-go model.

However, current usage patterns of clouds are mostly concentrated on long-running scale-out deployments that exhibit little dependencies between the virtual machines

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part II, LNCS 8806, pp. 26–37, 2014.

(VMs) or data sharing constraints: most large commercial cloud providers, such Amazon AWS and Rackspace, follow a philosophy of "throwing" more VMs at an application in order to deal with scalability issues. While this scale-out solution works for workloads that are loosely coupled, in the context of scientific and enterprise applications, there is a need to introduce support for closer coordination between virtualized resources in order to enable a tighter coupling in an efficient fashion.

These patterns were long ago acknowledged as difficult to deal with at large scale in the HPC (high performance computing) community, with decades of efforts dedicated to overcome them. In this context, the introduction of IaaS clouds as an alternative to high-end, privately-owned infrastructure presents a new challenges that calls for disruptive solutions, because the whole viewpoint needs to be changed: instead of optimizing the application to make the best out of a fixed physical infrastructure, we need to adopt the opposite: adapt the infrastructure to the dynamic needs of the application in order to satisfy performance requirements while incurring minimal costs. Thus, we cannot simply "port" techniques developed in the HPC community for IaaS clouds: novel approaches need to be developed that are designed to adopt the viewpoint from scratch. This has consequences across the whole IaaS software stack, from low level virtualization technologies up to the programming models exposed directly at application level.

More specifically, there are several important dimensions to this problem, related both to functional and non-functional aspects. With respect to functional aspects, application developers need programming models and tools to *automate resource allocation*. This aspect is non-trivial due to increasing complexity of cloud offers, which overwhelm not only new but also existing users: for example, if they opt to use Amazon EBS [2], they have to choose from 36 services, 20 instance types, 6 instance families, 2 generations of instances, 3 types of billing models and 2 types of block storage options. This level of complexity is already present for a single provider, not to mention the scenario of leveraging multiple at the same time, which in addition to increased complexity, also introduces the need to address *interoperability*. Furthermore, once the application is up and running, *elasticity* is an issue: programming models need to expose the right abstractions to let applications express their needs to grow and shrink dynamically in an easy way, while hiding the complexity of resource allocation.

With respect to non-functional aspects, *performance* and *optimal resource utilization* are key concerns. In particular, the problem of efficient resource sharing under concurrency to improve both performance and resource utilization is more difficult in IaaS clouds, considering they are multi-tenant and thus susceptible to more system noise and jitter compared to bare-bone architectures, for which this is already a problem [12]. On the other hand, the multi-tenant aspect also introduces new exploitable avenues, since common access patterns and synergies between different users can be detected and leveraged to improve both performance and resource utilization (e.g. data deduplication can be used to find common data between users and store only a single copy).

We argue in this paper for the need to provide a framework of tightly integrated layers that enables seamless access to high-performance heterogeneous resources by exposing the right programming models and abstractions at user level, while optimizing the performance and cost effectiveness of the infrastructure specifically for tightly coupled scientific and data-intensive applications running at large scale. As the target

users of such applications are mostly domain-experts and do not necessarily have a deep understanding of the technical details of the infrastructure, *transparency* is crucial: we propose to hide the functional and non-functional challenges as much as possible from the users, such that they can leverage simplified programming models that enable them to focus on the domain-specific aspects without giving up on performance, while at the same time enabling cloud providers to maximize infrastructure utilization and introduce competitive targeted offers with reduced costs.

2 High Priority Areas of Improvement

On the road to materialize our vision, we identified several high priority areas listed below.

2.1 Virtualization and Storage

Large scale scientific and data-intensive applications are often tightly coupled, which introduces the need for frequent synchronization and data sharing under a high degree of concurrency. Naturally, this leads to intense communications and exchanges of data, putting a high pressure on the networking infrastructure. Network virtualization overhead in this context was known to be a major barrier [9], however it is gradually improving thanks to wider adoption of single root I/O virtualization [6]. On the other hand, two major areas still remain open: scalable data storage and management, and access to accelerators.

Scalable Data Storage and Management. In an IaaS cloud, data storage is typically achieved by manually provisioning raw virtual disks that are then attached to running VM instances. All details related to the management of such raw disks, including what size or type to pick, how to use it (e.g., with what file system) and when to attach/detach a virtual disk are the responsibility of the user, which greatly increases complexity and leads to several issues. First, there is limited support to address *sharing under concurrency*: users have to manage sharing themselves or deploy a higher level abstraction (e.g. parallel or distributed file system) on top of their VMs, which is a solution that suffers from performance penalties in cloud environments. Second, there is limited potential to leverage *elasticity*: to avoid the complexity of manually managing disks, user often over-provision storage to cover the worst-case scenario, which leads to unnecessary tied-up resources that generate costs. Thus, it is important to introduce novel storage abstractions that can handle these aspects transparently to reduce application complexity while addressing the aforementioned issues.

Access to Accelerators. There are a few technologies to enable access to accelerators within a VM (e.g. pass-through [20]). However, these often force exclusive access to the accelerator, which negates any resource optimization opportunities due to multi-tenancy. Although there is progress in this direction especially in the context of GPUs [3], general purpose accelerators beyond GPUs were not explored so far from

this perspective. Furthermore, several advanced virtualization features that are crucial on IaaS clouds (such as live migration) are not yet supported. Besides multi-tenancy support, *elasticity* is also an important issue: users cannot dynamically turn on/off accelerators and easily move between general purpose CPUs and accelerators in order to optimize utilization and reach their goals with minimal costs. Thus, it is important to introduce novel abstractions that overcome these limitations and hide the complexity of accelerator management and sharing from the upper layers.

2.2 Provider Heterogeneity

Cloud Service Providers (CSPs) typically have their own Application Programming Interfaces (APIs) that allow customers to deploy and manage their cloud resources. However, users accessing multiple CSPs face the challenge of adapting their applications to a multiplicity of cloud environments with mostly incompatible APIs. Besides this drive, the multi-cloud migration is necessary for backup purposes when a CSP becomes unavailable at a certain point [24]. Moreover, the financial argument is another motivation for approaching the portability challenge as it enables seamless switching between CSPs when economic factors change [10]. Further reasons for using multiple Clouds have been identified in [21].

The Open Cloud Computing Interface (OCCI) [8] initiative aims to circumvent the vendor locks-in problem by engaging with a several initiatives such as SLA@SOI [7] and OpenNebula/OpenStack. The OCCI aims to standardize the RESTful APIs for task management. The standardization was started for the IaaS and now is extensible for SaaS and PaaS. However, OCCI fails to exhibit common platform for vendor APIs to define VM and their operations [14].

The application requirements are currently only partially taken into account in the application deployment and execution phases. By combining the benefits of SLA@SOI [7] and RESERVOIR [23], Metsch et. al. [13] have delivered a framework working with OCCI and providing a proof-of-concept of inter-operation between clouds. For scalable provisioning of resources and services, Buyya et. al. [5] have proposed an architecture for federated cloud environments. Similarly, the Contrail project used a Virtual Execution Platform (VEP) [11] to provide the virtual distribution of the resources and deploy the users applications independently. Enhanced with a proprietary Cloud broker, the mOSAIC project [22] provided an open-source deployable platform as a service that allows code portability between major IaaS providers.

However, the above mentioned approaches do not necessarily take into account the structure of a given application when elastically expanding and contracting resources in a widely distributed environment with heterogeneous resources. Thus, there is a need to *automate elasticity across cloud providers given specific application requirements and fine resource granularity that goes beyond VMs*.

2.3 Automation

Automation is a current challenge which requires to be adopted across cloud computing services, in order to increase efficiency and to facilitate the interaction of users with cloud services. An automation tool should arguably provide full automation with

little or no human interaction. Automation solutions such as Chef, Puppet Labs, and CFEngine are currently tested. The Chef configuration management system can be used in order to automate the installation and configuration of the applications. If the OVF format is used, the flexibility is therefore increased and the creation of a specific Chef recipes is not required for each cloud middleware [4].

There are a number of proprietary and open-source commercial tools such as RightScale, Amazon's CloudFormation and AutoScaling, SlipStream, Dell Cloud Manager, and Scalr which automate the access, deployment, and/or management of resources. Several research initiatives have been also recently initiated. PANACEA (www.panacea-cloud.eu) is currently developing solutions for a proactive autonomic management of cloud resources, based on a set of machine learning techniques and virtualization. CELAR (www.celarcloud.eu) intends to provides automatic resource allocation for applications that activates a right amount of resources based on application demands.

However, the work is still incomplete. Additional features should be made available to the current tools such as: *automated configuration and deployment of applications, automated user management, auto-scaling, automated recovery, automated backup, or automated governance*.

2.4 Ubiquitous Access

The design of a well-structured and optimized access layer is mandatory for a cloud computing platform focused on high demanding data analytics applications. An access layer should be responsible for making the applications available from a wide variety of devices including smart-phones, tablets or workstations, as well as available from the main operative systems. It must not only collect the data needed for the computation (even if the data is stored externally), but also prepare the acquired data and to send it to the computational platform in order to be processed. The way the access layer has to communicate with external data sources depends largely on the data analytics application. In case of the application described in Section 3, the images are collected from healthcare information systems; to do so, clinical information guidelines developed by Integrating the Healthcare Enterprise need to be followed.

Interoperability becomes an important issue when mobile users need to interact and communicate with the cloud. The current interface between mobile users and cloud are mostly based on the web interfaces. The rapid advances in HTML5 have resulted in a much more mobile friendly version of the best-known Web language, which has paved the way for web applications to work on any HTML5-compliant web browser.

Finally, in order to define and implement all the required functionalities of the access layer, a standard language could be used to describe a topology of cloud based web services, their components, relationships, and the processes that manage them. Nowadays one of the most used solutions is the Amazon AWS CloudFormation Template [1], a JSON data standard that allow cloud application administrators to define a collection of related AWS resources. However, platforms such OpenStack that includes Heat orchestrator are migrating from the AWS proprietary solution to a neutral, native Heat Template format/DSL. The Topology and Orchestration Specification for Cloud Applications (TOSCA [19]) is an emerging standard language that includes specifications to describe processes that create or modify web services. TOSCA is expected to

interoperate with the upcoming Heat Native DSL. An access layer can use a generic definition of the services architecture based on standard languages such as the TOSCA. This allows the adaptability of the access layer capabilities to fit the requirements of future applications. Moreover, such an approach will allow the reusability of common web services facilitating the easy deployment of new applications.

3 Case Study: Multi-biomarker Profile Imaging

In what follows we describe an case study and in the next section we reveal how an advance of the state-of-the-art is possible.

Problem Description. The next generation of medical imaging will provide patient-specific images relative to the biological-processes underlying the tissues as a non-invasive tool for the diagnosis, prognosis, treatment and follow-up of complex diseases, such as cancer and neurological disorders. The current medical imaging techniques enhance anatomical and functional aspects of the human body in three-dimensional spaces. Nevertheless, the characterization of the underlying biological processes at voxel level requires the combination of complementary biomarker images that enhance different characteristics from the tissues. The typical input of these studies is a 5-dimensional structure for each patient where each voxel has a set of biomarkers' values positioned in a 3D space at a specific moment of a time series. The calculation of multi-biomarker profile images is therefore computational and space-demanding, as it involves several phases over a stack of images from different acquisitions and protocols of the same patient or even of several patients. Each case typically takes more than 900 minutes on a state-of-the-art workstation.

The computation of the multi-biomarker profile images can be divided in five sequential phases: pre-processing, quantification, feature extraction, feature reduction and classification. The pre-processing phase is composed by seven steps that may be applied to each independent image of the study: denoising, inhomogeneity correction, super-resolution, registration, skull strepping and intensity range normalization. This phase can arguably be done concurrently for each image. Next, the quantification step calculated derived biomarkers' images relative to the underlined biological processes of the tissues from the functional images. Those functional images are usually sequences of images, hence, this step can be done concurrently at the acquisition level. After the pre-processing and quantification steps, additional features are extracted from the anatomical images and biomarkers' images. Those features, such as first-order central moments (mean, variance, skewness, and kurtosis), increase the information extracted from the patient. This phase can be parallelized at the image level. After the feature extraction, the number of features for each voxel may be high (e.g. from 20 to 100). Hence, linear and non-linear reduction techniques are applied to obtain a non-redundant representation of the data. This phase can be parallelized at patient level. Afterwards, the supervised or non-supervised classification using structured or unstructured algorithms receive the reduced 5D structure of data to classify each voxel into a biological-relative label. When unsupervised methods are applied, a post-processing step may be required to construct the final biological relative labels. This phase can be done concurrently at patient level.

Expected Benefits and Challenges of Moving to IaaS Clouds. As described above, the computation involves a high degree of parallelism not only per-patient but also from the perspective of handling multiple patients independently in parallel. Thus, moving to a cloud infrastructure has the potential to speed up not only the number of cases that can be handled at the same time, but also each individual case itself. However, such a move is difficult for a domain expert that has limited knowledge about IaaS cloud infrastructures. Specifically, there are several difficulties: (1) how to parallelize the various computational phases and orchestrate the I/O between them for the purpose of speeding up individual cases; (2) how to interleave and pipeline the individual cases together in order to process as many cases in parallel with as little computational and storage resources as possible; (3) how to elastically grow and shrink the computational and storage capability to match the needs of the hospital; (4) how to enforce quality-of-service constraints and priorities (e.g. some cases have tight deadlines). Thus, it is important to come up with a programming model and runtime that focuses on the requirements and the description of the workflow, while hiding all details about parallelization, resource provisioning and elasticity from the domain expert.

4 Towards an Integrated Framework

Figure 1 explains our vision in a nutshell: users write their applications using specialized programming models that enable simple ways to determine and attain goals on parallelism, elasticity, task dependencies, functional requirements, and performance-cost optimization.

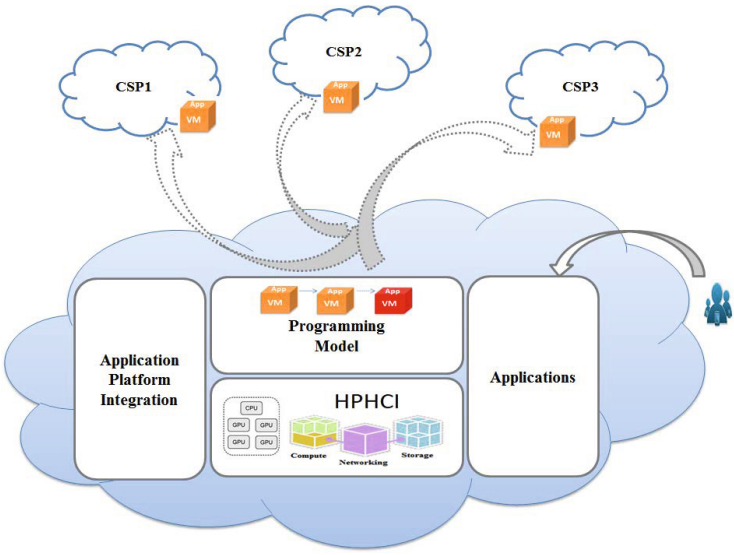

Fig. 1. General schema of the proposed framework

A specialized runtime environment is expected to interpret the user requirements (expressed through the programming models) and automates the provisioning and orchestration of lower-level resources. The key here is to enable autonomic elasticity and data sharing, potentially over multiple clouds, in order to minimize resource usage and cost, whilst preserving functional and non-functional constraints (e.g. performance requirements, SLAs, security etc.). Lower-level resources are exposed by the high performance heterogeneous cloud infrastructure (HPHCI) layer. They include data storage and management services and accelerator-enabled computational capabilities that are specifically designed to deliver high performance at large scale while enabling efficient sharing and transparent elasticity. The programming model runtime and the HPHCI layer integrate into the overall cloud ecosystem through a dedicated application platform integration layer that is responsible to ensure inter-operability through standardized APIs, effectively enabling our approach not only to co-exist with other cloud building blocks and tools, but also to span multiple cloud providers. Furthermore, an additional ubiquitous access layer is responsible to provide easy access and control to jobs to users of the software stack from a variety of devices, both static and mobile.

4.1 High Performance Heterogeneous Cloud Infrastructure

The framework intends to provide cloud infrastructure building blocks. These blocks are specifically targeted at data analytics applications which use advanced programming models in order to take advantage of both the cloud elasticity and the heterogeneous hardware. In this context, we propose several approaches to handle the two directions identified in Section 2.1.

Data Storage and Management. Build a lightweight storage layer that is centred around the idea of "storage neighbourhoods." Such neighbourhoods encompass groups of VMs that share storage resources based on interest, access patterns, resilience, and high availability requirements. The key in this context is that the VMs can help each other out to meet these sharing requirements for tightly coupled application without the need for a heavyweight repository that enables sharing at global level (e.g. parallel file system) while at the same time increasing the scalability and elasticity potential thanks to the focus on locality. Preliminary work [15,16,18,17] undertaken so far shows interesting potential for this avenue.

Access to Accelerators. Introduce virtualization techniques that enable sharing of accelerators between VMs at scale and consolidation in a multi-tenant environment. Similar to the storage neighbourhood concept, VM can cooperate to achieve elasticity and improve the utilization of the CPUs and accelerators by offloading work remotely or migrate on-the-fly as a whole to different physical nodes altogether.

Starting from these two high-priority areas, building higher level data abstractions on top of them specifically designed to enhance the potential of data-intensive applications is essential. In this context, the idea is to enrich raw data with metadata and active storage aspects that optimize the application and the programming model runtime with respect to high level data processing requirements, such as graph analytics or NoSQL.

4.2 Programming Model Runtime

A programming model is needed to enable the user to express functional and non-functional requirements (e.g. performance requirements, SLAs, security) and the provisioning and orchestration of low-level resources.

One important aspect to consider in this context is heterogeneity of multiple clouds and how to avoid vendor lock-in. We propose to create generic deployment templates based on different IaaS providers by deconstructing the complete description regarding an instance running in a standardized form. Based on the OVF, such deployment templates will be based on architectural patterns and forwarded to instantiate the pre-configured virtual machines on the specific IaaS platform. Moreover, the automated installation of applications onto the deployment template will be executed through open-source tools such as Chef cookbooks. Application programmers will not need to know the specifics of different APIs of the underlying IaaS from different vendors. New optimization mechanisms are needed to support transparent elasticity of resources between heterogeneous cloud platforms through the programming model.

Elasticity is intended to provide targeted performance constraints and on-demand provisioning and de-provisioning of cloud resources between heterogeneous cloud platforms, driven by usage policies, availability, and costs. Consequently, the efficiency of deployment and configuration of computationally-demanding and/or data demanding applications across multiple cloud providers will be arguably improved by the automation framework. At current state-of-the-art, the application developers not only need to have expertise in both areas (i.e. application domain and cloud services), but also need to manually implement the complex workflows, a time consuming error prone process. This will allow the realization of a seamless inter-operability in a multi-target environment, which is currently perceived as a drawback of adopting the cloud services.

4.3 Ubiquitous Access

The development and deployment of an access and visualization platform is necessary to allow seamless access to the computational resources of the high-performance heterogeneous cloud infrastructure from a multiplicity of devices. We propose to focus on three main features: (1) the cross-platform compatibility and rich interaction environment, (2) the capability of collecting and preparing the data needed for the computation, even if the data is stored in external data sources, and (3) the capability of interchanging heterogeneous data with the computational resources. In the development and deployment of the seamless access and visualization platform an effort will be done to enhance the reusability of the modules and services developed.

Methodology-wise, an approach as described above is experimental in nature. It systematically leverages a set of standardized, and representative applications, in order to study their access patterns and identify the functional and non-functional limitations and weaknesses to formulate best practices and software products that can be confidently used by industry practitioners and academics. Once we have obtained the desired performance levels, we can build the higher level data abstractions closely aligned to the infrastructure building blocks. Their design will be based on the results and lessons learned from the prototyping, and will include again an iterative improvement phase to

make sure desired performance and resource utilization levels are satisfied. Once the higher level data abstractions are complete, they will be integrated into the programming model runtime via standardized APIs and open data formats, integrated vertically to the application which will then be ubiquitously accessible.

4.4 How the Proposed Framework Enhances the Multi-biomarker Use Case

A cloud-based service can be developed to generate multi-biomarker profile images from the combination of a set of anatomical and functional images that can be seamlessly accessed and that agrees with the security constrains of medical information. Specifically, this approach can be applied to perfusion-weighted images to segment the tumoral, peritumoral, and edema regions of primary glioblastoma tumours in biological signatures relative to the aggressiveness of the tumour, such as the neoangiogenesis and microvascular proliferation. The development of this cloud computing technology can provide support in the decision-making to clinical centres, medical image analysis SMEs, and expert radiologists, with independence of their computational capabilities, infrastructures, location and devices. The cloud-based system should provide access to these services through mobile devices and low performance computers optimizing available resources by the institutions involved in this process. The services can be used in a flexible and scalable manner, establishing the necessary resources based on demand. Several alternatives will be implemented for the computer-intensive modules to take advantages of heterogeneous computing infrastructures, such as those composed by CPUs and GPUs. Finally, the use of a cloud-based service allows the easy update of the analysis algorithms and fast inclusion of new features.

5 Conclusions

In this paper we insisted on several challenges that result from the change in the way we reason about large scale scientific and data intensive computations on IaaS clouds, which involves adapting the infrastructure dynamically to the needs of the application in order to satisfy performance requirements while incurring minimal costs. In this context, *automated resource allocation*, *interoperability*, *elasticity*, *performance* and *optimal resource utilization* are key goals that are difficult to achieve with existing approaches, both because of the complexity introduced by specific IaaS aspects (such as multi-tenancy), as well as the fact that previous work was mostly designed to leverage a fixed infrastructure.

To this end, we advocated for a new generation of data storage and management services, accelerator-enabled computational capabilities, parallel pattern-oriented and heterogeneity-agnostic programming models in order to achieve the aforementioned goals in a *transparent* fashion, ultimately enabling users to easily adopt the advantages of IaaS clouds without sacrificing performance, while at the same time enabling cloud providers to maximize infrastructure utilization and introduce competitive targeted offers with reduced costs. We suggested several high priority areas that range from low-level virtualization and storage capabilities to high level abstractions that facilitate ubiquitous access. We believe that the key to enable such high priority areas

to best contribute to a new disruptive approach that specifically addresses the needs of large scale scientific and data-intensive applications on IaaS clouds in an efficient fashion is an integrated co-design that guides all software layers according to the unified goals and the feedback from domain-experts that come with real-life use cases.

Acknowledgment. The research of the first author is partially supported by the grant FP7-REGPOT-CT-2011-284595 (HOST).

References

1. Amazon Web Services (AWS), http://aws.amazon.com/
2. Amazon Elastic Block Storage (EBS), http://aws.amazon.com/ebs/
3. Becchi, M., Sajjapongse, K., Graves, I., Procter, A., Ravi, V., Chakradhar, S.: A virtual memory based runtime to support multi-tenancy in clusters with gpus. In: HPDC 2012: Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing, pp. 97–108. ACM, Delft (2012)
4. Boob, S., González-Vélez, H., Popescu, A.: Automated instantiation of heterogeneous Fast Flow CPU/GPU parallel pattern applications in clouds. In: PDP 2014, pp. 162–169. IEEE, Torino (2014)
5. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu, C.-H., Yang, L.T., Park, J.H., Yeo, S.-S. (eds.) ICA3PP 2010, Part I. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010)
6. Dong, Y., Yang, X., Li, J., Liao, G., Tian, K., Guan, H.: High performance network virtualization with sr-iov. J. Parallel Distrib. Comput. 72(11), 1471–1480 (2012)
7. Edmonds, A., Metsch, T., Papaspyrou, A., Richardson, A.: Open cloud computing interface: Open community leading cloud standards. ERCIM News 2010(83), 23–24 (2010)
8. Edmonds, A., Metsch, T., Papaspyrou, A., Richardson, A.: Toward an open cloud standard. IEEE Internet Computing 16(4), 15–25 (2012)
9. Expósito, R.R., Taboada, G.L., Ramos, S., Touriño, J., Doallo, R.: Performance analysis of HPC applications in the cloud. Future Generation Computer Systems 29(1), 218–229 (2013)
10. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. Software: Practice and Experience 44(3), 369–390 (2014)
11. Harsh, P., Jegou, Y., Cascella, R.G., Morin, C.: Contrail virtual execution platform challenges in being part of a cloud federation. In: Abramowicz, W., Llorente, I.M., Surridge, M., Zisman, A., Vayssière, J. (eds.) ServiceWave 2011. LNCS, vol. 6994, pp. 50–61. Springer, Heidelberg (2011)
12. Hoefler, T., Schneider, T., Lumsdaine, A.: Characterizing the influence of system noise on large-scale applications by simulation. In: SC 2010, pp. 1–11. ACM/IEEE (2010)
13. Metsch, T., Edmonds, A., Bayon, V.: Using cloud standards for interoperability of cloud frameworks. A technical RESERVOIR report (2010)
14. Metsch, T., Edmonds, A., Nyrén, R., Papaspyrou, A.: Open cloud computing interface–core. In: Open Grid Forum, OCCI-WG, Specification Document (2010), http://www.ogf.org/documents/GFD.183.pdf
15. Nicolae, B.: Understanding Vertical Scalability of I/O Virtualization for MapReduce Workloads: Challenges and Opportunities. In: BigDataCloud 2013: 2nd Workshop on Big Data Management in Clouds, Aachen, Germany (2013)

16. Nicolae, B., Rafique, M.M.: Leveraging Collaborative Content Exchange for On-Demand VM Multi-Deployments in IaaS Clouds. In: Wolf, F., Mohr, B., an Mey, D. (eds.) Euro-Par 2013. LNCS, vol. 8097, pp. 305–316. Springer, Heidelberg (2013)
17. Nicolae, B., Riteau, P., Keahey, K.: Bursting the Cloud Data Bubble: Towards Transparent Storage Elasticity in IaaS Clouds. In: IPDPS 2014: Proc. 28th IEEE International Parallel and Distributed Processing Symposium, Phoenix, USA (2014)
18. Nicolae, B., Riteau, P., Keahey, K.: Transparent Throughput Elasticity for IaaS Cloud Storage Using Guest-Side Block-Level Caching. In: UCC 2014: 7th IEEE/ACM International Conference on Utility and Cloud Computing, London, UK (2014)
19. OASIS Open: Topology and orchestration specification for cloud applications version 1.0. OASIS Standard. Version 1.0 (November 2013), http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
20. Ou, W.S., et al.: On implementation of GPU virtualization using PCI pass-through. In: CloudCom 2012, pp. 711–716. IEEE, Taipei (2012)
21. Petcu, D.: Consuming resources and services from multiple clouds. Journal of Grid Computing, 1–25 (in press, 2014), doi:10.1007/s10723-013-9290-3
22. Petcu, D., Martino, B., Venticinque, S., Rak, M., Mahr, T., Lopez, G., Brito, F., Cossu, R., Stopar, M., Sperka, S., Stankovski, V.: Experiences in building a mosaic of clouds. Journal of Cloud Computing: Advances, Systems and Applications 2(1), 12 (2013)
23. Rochwerger, B., et al.: The Reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development 53(4), 4:1–4:11 (2009)
24. Rodero-Merino, L., Vaquero, L.M., Gil, V., Galán, F., Fontán, J., Montero, R.S., Llorente, I.M.: From infrastructure delivery to service management in clouds. Future Generation Computer Systems 26(8), 1226–1240 (2010)