Proceedings of the Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)
Krakow, Poland

Jesus Carretero, Javier Garcia Blas
Roman Wyrzykowski, Emmanuel Jeannot.
(Editors)

September 10-11, 2015

On Autonomic HPC Clouds

Dana Petcu

Institute e-Austria Timişoara and West University of Timişoara, Romania
petcu@info.uvt.ro

Abstract

The long tail of science using HPC facilities is looking nowadays to instant available HPC Clouds as a viable alternative to the long waiting queues of supercomputing centers. While the name of HPC Cloud is suggesting a Cloud service, the current HPC-as-a-Service is mainly an offer of bar metal, better named cluster-on-demand. The elasticity and virtualization benefits of the Clouds are not exploited by HPC-as-a-Service. In this paper we discuss how the HPC Cloud offer can be improved from a particular point of view, of automation. After a reminder of the characteristics of the Autonomic Cloud, we project the requirements and expectations to what we name Autonomic HPC Clouds. Finally, we point towards the expected results of the latest research and development activities related to the topics that were identified.

Keywords HPC Clouds, Autonomic Computing

I. Introduction

The main characteristics of Cloud computing that have make it a market success for distributed applications are the elasticity in resource usage and on-demand self-service. However, the availability of a large number of computing resources has attract also various parallel computing applications that are ready to can make a compromise in performance in favor of scalability opportunities and instant availability of resources. While the supercomputing center have long queues that are serving resource-greedy batch parallel applications already tuned to match the architecture of the supercomputer, the HPC resources offered as Cloud services are nowadays seen as alternative for just-in-time parallel applications, real-time or interactive parallel applications, or even to support special environmental settings that are hard to be set in a supercomputing center through the interaction with admins. However, most of the current parallel applications are expected to run in the Cloud environment in the same manner as they are running on supercomputers (especially for production phases of an application), despite the multiple warnings from the literature about the lost of performance relative to the case of a supercomputing usage. In such context, the Cloud environment is expected to behave as a Grid environment, with the main difference that the user has more control on the software stack that is supporting the application execution (a thin difference as the Grid environments are allowing to simulate Cloud environments on top of Grid services). While such perspective is useful for the users who do not have access to the Grid environments that are mainly targeting academic users, the Cloud computing paradigm can offer more to HPC applications than the grid-like perspective.

Despite the fact that even the Cloud computing community has not taken up yet the full benefit of the rich results that have been obtained in the field of autonomic computing, we discuss in this paper a particular approach, on autonomic computing in HPC Clouds.

The paper is organized as follows. Firstly, we discuss the concept of Autonomic Cloud and the techniques that can make it a reality. Then we introduce the concept of Autonomic HPC Cloud and we look how the Autonomic Cloud concepts and techniques are projected in this particular case. Finally, we investigate the prospects of new results in the field by pointing towards various initiatives to provide partic-

ular solutions for Autonomic Cloud, respectively Autonomic HPC Clouds.

II. Autonomic Clouds

II.1 Autonomic computing

The automation of resource management is referred nowadays under the name of autonomic computing [1]. Autonomic computing refers also to information systems capable to self-manage following the goals set by human administrators. It is an inter-disciplinary field involving knowledge from several well-known branches of computer science, like distributed systems, artificial intelligence, bio-inspired computing, software engineering and control systems.

An autonomic system is expected to take decisions on its own using certain policies. It should also check and optimize its status and automatically adapt itself to changing conditions. The concept of autonomic computing is inspired from biology, in an analogy with the autonomic nervous system which takes care of low-level functions of the human body such as temperature regulation without any conscience action.

We consider in this paper that that the Autonomic Computing refers to the self-managing characteristics of a computing system.

II.2 The concept of Autonomic Cloud

Autonomic techniques are applied in Cloud environments, as rapid scalability of the pool of resources is needed to support unpredictable number of demands without a human intervention, or fast adaptation is requested to avoid the failures of hardware resources.

In what follows, we consider that the main characteristics of an Autonomic Cloud are the followings, as argued in [2]:

- involves computing and software services which instance number varies by adapting to unpredictable changes;
- follows a contextual behavior through methods of self-management, self-tuning, self-configuration, self-diagnosis, and self-healing;
- easy to manage and operate the services and deployments by using techniques for the design, build, deployment and management of resources with minimal human involvement;
- presents itself as a robust and fault tolerant system.

II.3 Architectures

The authors of [3] proposed a framework for evaluating the degree of adaptability supported by an architectural style and classified the most known architectural styles for distributed applications (e.g., pipe and filter, publish/subscribe, SOA, peer-to-peer) according to this framework. The most adaptable architectural styles for Autonomic Clouds can follow such adaptability evaluation.

The authors of [4] classified the architectural contributions to Cloud computing; to match the classification requirements at platform-as-a-service layer, for example, the programming environment (i.e., the tools for the development of applications) need to be ignored in order to focus on the execution environment with the goal to find an optimum deployment of application components on Cloud resources. At the software-as-a-service level the main goal is to make sure that services respect their QoS (e.g., response time).

An autonomic computing framework is naturally implemented using multi-agent systems, like in [5]. However, also other artificial intelligence techniques, like genetic algorithms, neural networks, or multi-objective and combinational optimization heuristics, can be successfully applied.

II.4 Techniques

Auto-scaling and load balancing. In order to take advantage of the elasticity characteristic of the Cloud, the deployed applications need to be automatically scaled in a way that makes the most efficient use of resources. Several frameworks have been introduced in the last years to support the application development taking into account the need of scalability. For example, SmartScale [6] and AutoJuju [7] are automated

scaling frameworks that uses a combination of vertical (adding more resources to existing VM instances) and horizontal (adding more VM instances) scaling to ensure that the application is scaled in a manner that optimizes both resource usage and the reconfiguration cost incurred due to scaling. Such scaling strategies are encountered also in [8] where different scalability patterns are used in combination with performance monitoring to allow automatic scalability management.

The auto-scaling of application components is useless without techniques for load balancing the requests among the scaled components. Usually load balancing is considered among physical machines, like in [9]. As Cloud applications are built from components, a load balancing among software components or services need to be considered. An energy-aware load balancer is proposed in [10].

Scheduling. The scheduling problem is as a multi-objective problem where the transfer, deployment and energy consumption costs need to be simultaneously minimized. Several approaches used in heterogeneous environments as presented in [11, 12, 13, 14] can be applied.

The problem of finding the mapping which minimizes the cost is NP complete and as a result scheduling (meta-)heuristics should be used to find sub-optimal solutions. Meta-heuristics such as those based on neural networks or evolutionary algorithms or linear programming proved their efficiency in solving cost problems found in scheduling problems [15].

Adaptive resource provisioning. The problem of adaptive virtualized CPU provisioning has received a lot of attention (for example, in [16, 17, 18]). However, an automated adaptive resource provisioning system as proposed on [19], based on feedback controllers (customer add-on outside of the Cloud service itself), was not reported yet. Current adaptive systems are based on real-time monitoring, like in [20].

In [21] an automated framework for resource allocation is presented: it can adapt the parameters to meet specific accuracy goal, and then dynamically converge to near-optimal resource allocation (optimal in terms of minimum costs). The proposed solution can handle unexpected changes in the data distribution characteristics and/or rates of the streaming application. Resource allocation for streaming processing was considered also in [22] which proposed an elastic scaling of data parallel operators.

A current challenge for Cloud providers is to automate the management of virtual servers while taking into account both high-level quality of service requirements of hosted applications as well as the resource management costs. In this context, the paper [23] proposes an autonomic resource manager to control the virtualized environment which decouples the provisioning of resources from the dynamic placement of virtual machines and which aims to optimize a global utility function which integrates both the degree of SLA fulfillment and the operating costs (a constraint programming approach to formulate and solve the optimization problem).

Probabilistic models were extensively used to assess the reliability of software systems at the architectural level, like in [24, 25], and these should to applied also to Cloud systems. The idea of [26] to reason at runtime about the non-functional attributes of the system and to perform accordingly some adaptations is particularly interesting in the context of Autonomic Clouds.

Service selection. Due to the tremendous number of Cloud services and the lack of a standard for their specification, manual service selection is a time costly task. Automatic methods for matching the user needs with the offers are therefore needed. In [27] for example a method for finding semantically equal SLA elements from differing SLAs by utilizing several machine learning algorithms is presented, together with a framework for automatic SLA management. Themis [28] is an implementation of a proportional-share auction that maximizes resource utilization while considering virtual machine migration costs; it uses a set of feedback-based control policies to adapt the application bid and resource demand to fluctuations in price. A decision support system based on risk analysis was proposed more recently in [29].

Service composition. Cloud service description languages should allow the automatically composition of

Cloud service to achieve a certain goal. The paper [30] formalizes the issue of automatic combination of Cloud services. Moreover, a proof-of-the-concept implementation is revealed to leverage a batch process for automatically constructing possible combinations of Cloud services, followed by a search for the best fit solution.

In [31] is presented an approach, named Café (Composite Application Framework), to describe configurable composite service-oriented applications and to automatically provision them across different providers. Components can be internal or external to the application and can be deployed in any of the delivery models present in the Cloud. The components are annotated with requirements for the infrastructure they later need to be run on. A component graph is used to capture the dependencies between components and to match against the infrastructures offered by different providers.

Service discovery. Software agents have been successfully used in the recent years for service discovery, brokering or composition. Cloudle [32] is such an agent-based search engine for Cloud service discovery proving that agent-based cooperative problem-solving techniques can be effectively adopted for automating Cloud service composition. As reported in [33], agents can be used also in the mechanism for service migration between Clouds.

Self-configuration. An autonomic computing application is expected to be composed of autonomic components interacting with each other. An autonomic component can be modeled in terms of a local and a global control loops with sensors for self-monitoring, effectors for self-adjustment, knowledge and planner or adapter for exploiting policies based on self- and environment awareness.

The authors of [34] proposed an automated line for deploying distributed application composed of a set of virtual appliances, which includes a decentralized protocol for self-configuring the virtual application machines. The solution named VAMP (Virtual Applications Management Platform) relies upon a formalism for describing an application as a set of interconnected virtual machines and on an engine for interpreting this formalism and automating the application deployment using infrastructure services. The formalism offers a global view of the application to be deployed in terms of components with the associated configuration- and interconnection constraints and with their distribution within virtual machines; it extends OVF language, dedicated to virtual machines description, with an description language for distributed application software architecture.

The above described approach is complemented by the one from [35] where is exposed a mechanism that requires zero manual intervention during the configurations on the IP addresses of the resources from a Cloud center.

An automated approach to deploy pre-configured and ready-to-run virtual appliances on the most suitable Cloud infrastructure is still missing. However, in [36] is proposed an architectural approach using ontology-based discovery to provide QoS aware deployment of appliances on Cloud service providers.

The paper [37], that exposes the design of an adaptive framework for optimizing the execution of scientific applications in the Cloud, uses a MAPE-K loop, well known in autonomic computing, and relies on the concept of utility for optimizing the configuration of the Cloud stack on a per-job basis.

Self-healing. The policy-based, goal-based, or utility-based approaches are reactive techniques that enable the Autonomic Cloud to respond to problems only when they occur [38]. For example, in [39] a policy based management is used to evaluate the state of the system against predefined rules and actuate self-healing to return the system to the desired state – focus is put on investigating the capability of the system to recognize a fault and react to it. Alternatively, proactive techniques like the ones proposed in [40, 41] predict a set of environmental conditions before they actually happen.

Reactive and proactive techniques are usually implemented using multi-agent systems. In the context of Autonomic Clouds we can distinguish between two types of solutions. The first type of solutions exploits the idea of building multi-agent controllers for autonomic systems, which are capable to manage the system and themselves, as in [42]. The second idea is

No. Characteristic	Target	Proposed approach
1. Resources/services variability	Auto-scaling	Combine vertical with horizontal scaling
		Use scaling patterns
		Performance monitoring
	Load-balancing	At the physical machines level
		At the software service level
		At the application components level
	Scheduling	Optimize transfer costs
		Optimize deployment costs
		Optimize energy consumption
	Adaptive resource provisioning	Control-policy to adapt to price fluctuations
		Use feedback controllers
3. Contextual behaviour	Self-healing	Policy-based approach to evaluate state against rules
		Multi-agent controllers
		VMs/services acting as agents
	Self-configuration	Use descriptors of the interconnected components
		Component dynamic identification
		Use pre-configurated appliances
	Automatic pricing	Self-adjust price according application requirements
3. Easy deployment & management	Optimal deployment Selection	Use patterns of architectural styles
		Find semantically equal SLA
		Auctions for mazimize utilization & reduce migration costs
	Composition	Description language to support composition
		Automatically construct combinations & search for best fit
		Dependency graphs for application components and their annotation with infrastructure requirements
	Discovery	Agent-based search engine
4. Robust and fault tolerant	Reliability	Decouple resource provisioning from dynamic placement
		Reason at run-time about the non-functional attributes
		Use probabilistic models
	Migration	Agent-based mechanism

Table 1: Characteristics, targets and approaches for Autonomic Clouds (after [2])

to allow virtual machines and services to behave like agents and to make decisions based on local policies and local knowledge as in [43, 44].

Autonomic pricing mechanism. Most of the Cloud service providers charge their clients for metered usage based on fixed prices. In [45] were exposed pros and cons of charging fixed prices as compared to variable prices. Deploying an autonomic pricing mechanism that self-adjusts pricing parameters to consider application and service requirements of users is shown to achieve higher revenue than various other common fixed and variable pricing mechanisms.

Summarize. Table 1 summarizes the main characteristics and targets in Autonomic Cloud as well as the existing approaches.

III. HPC Clouds

The Cloud paradigm is attractive for the HPC applications due to the promise of: (a) instant availability

instead the long waiting queues of supercomputers; (b) software stack that can be selected by the application owner instead the need of a match with the installed OS or libraries; (c) larger capacity than offered by the local or regional HPC centers; (d) abstractization of the hardware resources through virtualization and which provide a certain level of portability, instead a strong dependence on the hardware architecture; (e) service level agreements instead best effort.

However, there are several HPC requirements that are not matching the Cloud concept and are hindering its adoption in the HPC communities: (a) the need to be close to the bar metal to obtain the highest possible performance from the underlying machines instead the use of virtualized environments; (b) the need to use optimized HPC libraries that are hardware dependent instead general purpose software; (c) the need of tuned hardware for particular computations, including combination of CPUs, GPUs and FPGAs, instead commodity hardware; (d) the need for a large number of resources for well-defined time instead a small number of resources for an indefinite time; (e) the need for high throughput interconnections based on userspace communication pathways requiring wired data transfer instead virtualized networks transfers; (f) the need for parallel file-systems instead distributed file-systems.

The HPC applications of which performance is not strongly affected by the differences between the Cloud infrastructure services and the supercomputing center services are the ones which do not have special requirements in high-throughput interconnection and storage. For example, parametric studies in which a certain program is executed multiple times with different parameters (or bags of independent tasks) can run in Cloud environment with a relative low lost in performance compared with supercomputing environments. Moreover, new opportunities are offered by the instant availability for parallel streaming applications or real time parallel applications.

The HPC-as-a-service (HPCaaS) concept makes a compromise between the common understanding of distributed Cloud services and the HPC requirements which where enumerated above. It refers to a bare-metal compute model similar to an in-house cluster (therefore, named here cluster-on-demand). The HPC cluster of Amazon EC2[1], Gompute[2], Penguin-on-Demand (POD)[3], R-HPC[4], Cyclone[5], Salbacore[6] are few example of HPCaaSs. Dozens of popular open source applications frequently used by the HPC community, including schedulers and load balancers, are ready to be selected and executed in the acquired environments. GPGPUs are available at request. Data transfer inside the clusters are not charged. High throughput networks are available.

However, the HPCaaS offers are hardware dependent and a HPC application owner still need to evaluate the services before the deployment for production purposes. The scale-up and scale-out options are rarely encountered (a counterexample is in the offer of Cyclone). As the offers are quite different, the migration of parallel applications between various HPCaaS provider sites remains an issue. Despite the adoption of concepts of service level agreement adoption, the fault tolerance remains an application provider problem. Self-configuration is far to be achieved and the application provider that intends to consume a HPCaaS needs to have parallel programming knowledge. A holistic vision of the evolution of HPC Clouds towards a tight integration betweeb programming models, runtime middleware and virtualization infrastructure was presented recently in [46].

IV. Towards Autonomic HPC Clouds

In this section we discuss the requirements and expectations of future HPCaaSs compliant with the concepts and techniques of autonomic computing.

IV.1 The concept of Autonomic HPC Clouds

In most cases, the scalability of an HPC application is predictable. This fact allows the supercomputing centers to request as input parameter for the job submission descriptor the number of resources that will

[1] https://aws.amazon.com/hpc/
[2] http://www.gompute.com/
[3] https://pod.penguincomputing.com/
[4] http://www.r-hpc.com/
[5] http://www.sgi.com/products/hpc_cloud/cyclone/
[6] http://www.sabalcore.com/

be allocated, as well as a time interval. While this hypothesis is maintained by the current HPCaaSs, it is not the case for most of other Cloud services that are designed to support scale-ups and -outs and not defined time intervals for resource allocations. The design of Autonomic HPC Clouds should foreseen that a variable number of resources are possible, especially to enable the Big Data or streaming data parallel processing.

The hardware failures were considered until now a minor problem in the HPC computing world. However, with the advent of the race for achieving the petascale levels in HPC, the hardware and software failure tolerance increased in importance. The changes in the programming paradigms and languages that are foreseen for petascale systems to coupe with failure tolerance are of high interest for the HPC Cloud providers in conjunction with the pressure to fulfill high quality service level agreements. Self-* procedures are welcomed in this context to be combined with the classical tools for HPC support.

The deployment and the control of the HPC applications and its resources is still far from being hardware-independent. Moreover, few of the HPCaaS providers are measuring the costs based on job submission rather on being based on the allocated hardware.

Consequently, we consider that the four main characteristics of Autonomic Clouds are relevant in the definition of an Autonomic HPC Cloud. An HPC Cloud is autonomic if: (1) involves computing and software services which instance number varies; (2) follows a contextual behavior through methods of self-management, self-tuning, self-configuration, self-diagnosis, and self-healing; (3) easy to manage and operate the services and deployments; (4) presents itself as a robust and fault tolerant system.

IV.2 Requirements related to the architecture

A similar framework with the one proposed in [3] to evaluate the degree of adaptability supported by an architectural style of distributed systems should be adopted for the architectural styles of parallel applications (e.g. master-slave, bag of tasks, parametric studies).

Currently, the targeted architecture HPCaaS is of IaaS type. No offers exists yet for HPC oriented PaaS or HPC based SaaS. Moreover, the variety of the HPCaaS services in terms of hardware support opens the problem of portability of the HPC Cloud applications and the interoperability in case of building HPC Clouds Federations or Multi-HPC-Clouds.

An abstract representation of the HPCaaS which allow an automatic deployment is needed to be defined, standardized and adopted by HPCaaS providers is urgently needed.

IV.3 Autonomic computing techniques compliance with HPC Cloud

Auto-scaling and load balancing In order to take advantage of the elasticity characteristic of the Cloud, the deployed applications need to be scale up and scale down during their execution. However, the current deployment style of most HPC applications is supposing the allocation of a fixed number of hardware and software resources. The introduction of automatic scaling mechanisms in the parallel programming paradigms and languages is necessary to be pursued.

The classic load balancers used in HPC environments are also assuming by default an estimation of the number of requested resources and execution time. In order to coupe with a variable number of resources during execution or the undefined execution time, or even with resource usage optimizations or energy-aware resource consumption, the current HPC load balancers should be revised and a good starting point are the new proposals for Cloud specific load balancers.

While horizontal and vertical scaling based on the number of virtual machines (more identical VMs, respectively more identical resources to existing VMs) are a custom in classical Cloud services, the scaling of HPC applications in heterogeneous environments composed of various computing, storage and network devices is a challenge problem not yet investigated. Different scalability patterns used in combination with performance monitoring can also help in HPC case the automatic scalability management.

Scheduling The scheduling problem was intensively studied by the HPC provider communities and beyond, and has been escalated with the Grid computing advent. In the context of the current trend to allow the portability of HPC applications from CPUs to GPGPUs and viceversa via abstractions at the programming level, or to take into consideration the energy consumption costs of interest for both the HPC Cloud provider and the HPC application provider, the existing scheduling techniques should be redefined.

A proof-of-concept framework was proposed in [47] for self-adaptive resource scheduling ensuring a trade off between performance and energy at real time on multi-core or many-core based heterogeneous Clouds.

Adaptive resource provisioning Finding the an optimum quantity of HPC resources can be critical for the automation in HPC Clouds. Being unaware of the power of the available resources, the HPC application provider is usually not able to estimate the application needs of resource. Automatic frameworks for resource provisioning and sizing for HPC applications in HPC Clouds are therefore needed. Such framework should use performance models of the application, performance estimators according to the available resources as well as performance monitoring date of previous executed applications to build some performance estimators. Feedback controllers (customer add-on outside of the Cloud service itself) are also welcomed if the application is not running for the first time in the HPC Cloud environment. Beyond the performance estimators, the automated framework for resource allocation should follow policies and rules (cost minimization, energy consumption minimization, communication minimization volume, specific SLAs, or a combination of these criteria).

A proposal for an autonomous resource management system with self-properties for HPC Cloud was discussed in [48].

Service selection HPC applications are often using special libraries. Few HPCaaS offers are including nowadays the access to pre-installed HPC specific software that enable the execution of particular applications (e.g. OpenFOAM provided by Cyclone). A market place of such appliances can improve the quality of the HPCaaSs.

The diversity of the HPCaaS due to the variation of the underlying hardware poses the problem of the right service for a certain HPC application. Moreover, there is a lack of a standard for their specification. Autonomic methods for matching the HPC application needs with the offers are therefore needed.

Switching between computational resources from different providers or inside a Heterogenous Cloud in order to select the most suitable computational environment for an HPC application execution is a desirable feature to avoid hardware or vendor lock-in. In the case of HPC applications, the migration is hindered by hardware incompatibility and the need to tune the software for specific hardware to achieve high performance. A proof-of-concept tool named ADAPT [49] is trying to provide a solution by an autonomic execution monitor that exploits feedback from the runtime and resolves missing dependencies.

Service composition Often the HPC applications are build from multiple programs depended on specific libraries and which are executed in a specific order according to a particular workflow. This split in various components offers the opportunities to play with various configuration settings and to find an 'optimal' combination (from a certain point of view, e.g. response time, costs, energy etc). The approaches that were proposed for the Autonomic Cloud are well suited to be applied also in the case of the Autonomic HPC Cloud.

Service discovery The number of the available HPCaaS is limited and their offer is still easy to be followed by a human agent. The management tools behind the HPCaaS are currently proprietary. If open-source tools will be developed, it is possible that the number of the HPCaaS will increase exponentially, providing incomes to small computing centers. Then the discovery of the new services will become a problem also for the Autonomic HPC Cloud.

Self-configuration An automated approach to deploy pre-configured and ready-to-run virtual appliances on the most suitable Cloud infrastructure is still

missing. This fact is valid also for HPC Clouds. Moreover, the lack of virtualization layer (bar metal offers in HPCaaS) is complicating the application deployer task. The efforts to provide virtualization layers also for GPUs (available nowadays only for a particular hardware architecture) and to reduce the performance impact of the virtualization software are further steps towards an easy to manage configuration process.

An automatic I/O configurator for HPC applications was proposed in [50]; it utilizes machine learning models to perform black-box performance/cost predictions.

Self-healing The HPC Cloud offer should provide a certain quality stated in a service level agreement. E.q. hardware and software services should be available 99.99% of the time. Faulty services should be detected just in time and automatic reactive mechanisms should be in place. The performance monitoring of the services is therefore essential and current techniques should be improved both in terms of the number of sensors, but also in what concerns non-intrusiveness and policies. Proactive techniques were not approached yet in HPC Cloud context.

Autonomic pricing mechanism HPCaaS billings are usually based on hardware allocation time, rather than on job as expected in a supercomputing center (with one counterexample: R-HPC is able to provide job-based bills). Therefore, most of the HPCaaS providers charge their clients based on fixed prices. Variable prices can be conceived based on the urgency of the HPC tasks versus the current load of the provider resources and the supplementary sofware stack that is requested beyond the bar metal.

V. Expectations from on-going research activities

Panacea, HARNESS, MIKELANGELO and CloudLightning are four European research and development on-going initiatives that are working with Autonomic Clouds, respectively Autonomic HPC Clouds. Their progress will influence the future of these fields.

PANACEA[7] proposes innovative solutions for a proactive autonomic management of cloud resources, based on a set of advanced machine learning techniques and virtualization. PANACEA supports the following properties of the Autonomic Cloud: (a) self-healing against anomalies by recovering from multiple node and link failures and using proactive rejuvenation of applications and servers for preventing crashes and increasing the availability, predicting the threshold violation of response time of servers; (b) self-configuring by efficiently mapping user's requirements onto distributed clouds and configuring on-the-fly in the presence of anomalies, self-optimizing using proactive migration of virtual machines from one cloud resource to another, maintaining the quality of service of end-to-end flows; (c) self-protecting using proactive reconfiguration of overlay networks to protect against security attacks.

HARNESS[8] integrates heterogeneous hardware and network technologies into data centre platforms, to increase performance, reduce energy consumption, and lower cost profiles for Cloud applications. It develops an enhanced PaaS software stack that brings new degrees of freedom to cloud resource allocation and optimisation. Technologies such as FPGAs, GPGPUs, programmable network routers, and solid-state disks promise increased performance, reduced energy consumption, and lower cost profiles. Specialised technologies are virtualised into resources that can be managed and accessed at the platform level. The Cloud platform has access to a variety of resources to which it can map the components. A flexible application may potentially be deployed in many different ways over these resources, each option having its own cost, performance, and usage characteristics.

MIKELANGELO[9] is addressing the issues of HPC Clouds and targets a core bottleneck, the virtual I/O. The work is concentrated on improvement of virtual I/O in KVM.

CloudLightning[10] proposes a new way of provisioning heterogeneous cloud resources to deliver services, specified by the user, using a particular service de-

[7] http://projects.laas.fr/panacea-cloud/
[8] http://www.harness-project.eu/
[9] http://www.mikelangelo-project.eu/
[10] http://cloudlightning.eu/

scription language. Due to the evolving complexity of modern heterogeneous clouds, it proposes to build our system based on principles of self-management and self-organisation. The goal is to address energy inefficiencies particularly in the use of resources and consequently to deliver savings to the cloud provider and the cloud consumer in terms of reduced power consumption and improved service delivery.

VI. Conclusions

We defined the main characteristics of the Autonomic HPC Clouds and we identified the key elements of a roadmap to evolve HPC-as-a-Service offers to take advantage of the Cloud and Autonomic Computing concepts. The problems to be solved are complex and the solutions are not straightforward. Partial solutions are foreseen to be provided by ongoing research and development initiatives.

Acknowledgment

The work related to Autonomic HPC Clouds is supported by the European Commission under grant agreement H2020-6643946 (CloudLightning). The CLoudLightning project proposal was prepared by eight partner institutions, three of them as earlier partners in the COST Action IC1305 NESUS, benefiting from its inputs for the proposal. The section related to Autonomic Clouds is supported by the Romanian UEFISCDI under grant agreement PN-II-ID-PCE-2011-3-0260 (AMICAS).

References

[1] J.O. Kephart, D.M. Chess, "The vision of autonomic computing", *Computer* 36, pp. 41-50, 2003.

[2] D. Petcu, "Building automatic clouds with an open-source and deployable platform-as-a-service", in *Advances in Parallel Computing* 23, IOS Press, pp. 3-19, 2014.

[3] R.N. Taylor, N. Medvidovic, P. Oreizy, "Architectural styles for runtime software adaptation", in *Proceedings of WICSA/ECSA*, 2009, pp. 171-180.

[4] M. Litoiu, M. Woodside, J. Wong, J. Ng, G. Iszlai, "A business driven cloud optimization architecture", in *Proceedings of ACM SAC*, 2010, pp. 380-385.

[5] S. Venticinque, R. Aversa, B. Di Martino, D. Petcu, "Agent-based cloud provisioning and management. Design and prototypal implementation", in *Proceedings CLOSER*, 2011, 184-191

[6] S. Dutta, S. Gera, A. Verma, B. Viswanathan, "Smartscale: automatic application scaling in enterprise clouds", in *Proceedings of IEEE CLOUD*, 2012, pp.221-228.

[7] B. Karakostas, "Towards autonomic cloud configuration and deployment environments", in *Proceedings of ICCAC*, 2014, pp. 93-96.

[8] M. Ughetti, "Scalability patterns for platform-as-a-service", in *Proceedings of IEEE CLOUD*, 2012, pp. 718-725.

[9] C. Adam, R. Stadler, "A middleware design for large-scale clusters offering multiple services", *IEEE Transactions on Network and Service Management* 3:1, 2006, pp. 1-12.

[10] A. Paya, D. Marinescu, "Energy-aware load balancing and application scaling for the cloud ecosystem", *IEEE Transactions on Cloud Computing*, 2015, doi: 10.1109/TCC.2015.2396059.

[11] M.E. Frincu, N. M. Villegas, D. Petcu, H. Muller, R. Rouvoy, "Self-healing distributed scheduling platform", in *Proceedings of CCGRID*, 2011, pp. 225-234.

[12] M.E. Frincu, C. Craciun, "Dynamic and adaptive rule-based workflow engine for scientific problems in distributed environments", in *Cloud Computing and Software Services: Theory and Techniques*, CRC Press, 2010, pp. 227-251.

[13] M.E. Frincu, "Dynamic scheduling algorithm for heterogeneous environments with regular task input from multiple requests", in *Lecture Notes in Computer Science* 5529, 2009, pp. 199-210.

[14] F. Micota, M. Frincu, D. Zaharie, "Population-based metaheuristics for tasks scheduling in heterogeneous distributed systems", in *Lecture Notes in Computer Science* 6046, 2011, pp. 321-328.

[15] A. Benoit, L. Marchal, J.F. Pineau, Y. Robert, F. Vivien, "Scheduling concurrent bag-of-tasks applications on heterogeneous platforms", in *IEEE Transactions on Computers* 59:2, 2010, pp. 202-217

[16] P. Padala, K.G. Shin, X. Zhu, M .Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, "Adaptive control of virtualized resources in utility computing environments", in *Proceedings of Eurosys*, 2007, pp. 289-302.

[17] E. Kalyvianaki, T. Charalambous, S. Hand, "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters", in *Proceedings of ICAC*, 2009, pp. 117-126.

[18] S.M.Park, M.Humphrey, "Feedback-controlled resource sharing for predictable escience", in *Proceedings of SC*, 2008, article 13.

[19] H.C. Lim, S. Babu, J.S. Chase, S.S. Parekh, "Automated control in cloud computing: challenges and opportunities", in *Proceedings of ACDC*, 2009, pp. 13-18.

[20] F. Fargo, C. Tunc,Y. Al-Nashif, A. Akoglu, S. Hariri, "Autonomic Workload and Resources Management of Cloud Computing Services," in *Proceedings of ICCAC*, 2014, pp.101-110.

[21] S. Vijayakumar, Q. Zhu, G. Agrawal, "Automated and dynamic application accuracy management and resource provisioning in a cloud environment", in *Proceedings of GRID*, 2010, pp. 33-40.

[22] S. Schneider, H. Andrade, B. Gedik, A. Biem, K.L. Wu, "Elastic scaling of data parallel operators in stream processing", in *Proceedings of IPDPS*, 2009, pp. 1-12.

[23] H.N. Van, F. D. Tran, J.M. Menaud, "SLA-aware virtual resource management for cloud infrastructures", in *Proceedings of CIT*, 2009, pp. 357-362

[24] A. Filieri,C. Ghezzi, V. Grassi, R. Mirandola, "Reliability analysis of component-based systems with multiple failure modes", in *Lecture Notes in Computer Science* 6092, 2010, pp. 1-20.

[25] I. Krka, L. Golubchik, N. Medvidovic, "Probabilistic automata for architecture-based reliability assessment", in *Proceedings of QUOVADIS*, 2010, pp. 17-24.

[26] I. Epifani, C. Ghezzi, R. Mirandola, G. Tamburrelli, "Model evolution by run-time parameter adaptation", in *Proceedings of ICSE*, 2009, pp. 111-121.

[27] C. Redl, I. Breskovic, I. Brandic, S. Dustdar, "Automatic sla matching and provider selection in grid and cloud computing markets", in *Proceedings of GRID*, 2012, pp. 85-94.

[28] S. Costache, N. Parlavantzas, C. Morin, S. Kortas, "Themis: economy-based automatic resource scaling for cloud systems", in *Proceedings of HPCC*, 2012, pp. 367-374.

[29] S. Gupta, V. Muntes-Mulero, P. Matthews, J. Dominiak, A. Omerovic, J. Aranda, S. Seycek, "Risk-driven framework for decision support in cloud service selection", in *Proceedings of CCGRID*, 2015, 545-554.

[30] D.K. Nguyen, F. Lelli, M.P. Papazoglou, W.J. Van den Heuvel, "Issue in automatic combination of cloud services", in *Proceedings of ISPA*, 2012, pp. 487-493.

[31] R. Mietzner, T. Unger, F. Leymann, "Cafe: a generic configurable customizable composite cloud application framework", in *Lecture Notes in Computer Science* 5870, 2009, pp. 357-364.

[32] K. M. Sim, "Agent-based cloud computing", *IEEE Transactions on Services Computing* 99, 2011, pp. 1.

[33] C.T. Fan, W.J. Wang, Y.S. Chang, "Agent-based service migration framework in hybrid cloud", in *Proceedings of HPCC*, 2011, pp. 887-882.

[34] X. Etchevers, T. Coupaye, F. Boyer, N. De Palma, "Self-configuration of distributed applications in

the cloud", in *Proceedings of CLOUD*, 2011, pp. 668-675.

[35] C. Hu, M. Yang, K. Zheng, K. Chen, X. Zhang, B. Liu, "Automatically configuring the network layer of data centers for cloud computing", *IBM Journal of Research and Development* 55:6, 2011, 3:1–3:10.

[36] A.V. Dastjerdi, S.G.H. Tabatabaei, R. Buyya, "An effective architecture for automated appliance management system applying ontology-based cloud discovery", in *Proceedings of CCGrid*, 2010, pp. 104-112.

[37] M. Koehler, S. Benkner, "Design of an adaptive framework for utility-based optimization of scientific applications in the cloud", in *Proceedings of UCC*, 2012, pp. 303-308.

[38] J.O. Kephart, W.E. Walsh, "An artificial intelligence perspective on autonomic computing policies", in *Proceedings of POLICY*, 2004, pp. 3-10.

[39] T. Lorimer, R. Sterritt, "Autonomic management of cloud neighborhoods through pulse monitoring", in *Proceedings of UCC*, 2012, pp. 295-302.

[40] V. Casola, E.P. Mancini, N. Mazzocca, M. Rak, U. Villano. "Self-optimization of secure web services", *Computer Communications* 31, 2008, pp. 4312-4323.

[41] M. Litoiu, "A performance analysis method for autonomic computing systems", *ACM Transactions on Autonomous and Adaptive Systems* 2:1, 2007, article 3.

[42] B.A. Caprarescu, D. Petcu, "A self-organizing feedback loop for autonomic computing", in *Proceedings of COMPUTATIONWORLD*, 2009, pp. 126-131.

[43] K. Begnum, N.A. Lartey, L. Xing, "Cloud-oriented virtual machine management with MLN", in *Lecture Notes in Computer Science* 5931, 2009, pp. 266-277.

[44] B.A. Caprarescu, N.M. Calcavecchia, E. Di Nitto, D.J. Dubois, "SOS cloud: self-organizing services in the cloud", in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* 87, 2012, pp. 48-55.

[45] C.S. Yeo, S. Venugopal, X. Chu, R. Buyya, "Autonomic metered pricing for a utility computing service", *Future Generation Computer Systems* 26 (8), 2010, pp. 1368-1380.

[46] D. Petcu, H. Gonzalez-Velez, B. Nicolae, J.M. Garia-Gomez, E. Fuster-Garcia, C. Sheridan, "Next generation HPC clouds: a view for large-scale scientific and data-intensive applications", *Lecture Notes in Computer Science* 8806, 2014, pp. 26-37.

[47] J. Wood, B. Romoser, I. Zecena, Z. Zong, "B-MAPS: a self-adaptive resource scheduling framework for heterogeneous cloud systems", in *Proceedings of CAC*, 2013, Article 19.

[48] M.E. Frincu, D. Petcu, "Resource management for HPC on the cloud", in *High-Performance Computing on Complex Environments*, 2014, pp. 303-323.

[49] J. Slawinski, V. Sunderam, "Autonomic multi-target deployment of science and engineering HPC applications," in *Proceedings of ICCAC*, 2014, pp.180-186.

[50] L. Mingliang, J. Ye, Z. Jidong, Z. Yan, S. Qianqian, M. Xiaosong, C. Wenguang, "ACIC: Automatic cloud I/O configurator for HPC applications," in *Proceedings of SC*, 2013, pp. 1-12.