Future Generation Computer Systems 32 (2014) 54–68

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous Cloud environments

A. Kertesz [a,*], G. Kecskemeti [a], I. Brandic [b]

[a] MTA SZTAKI, H-1518 Budapest, P.O. Box 63, Hungary

[b] Vienna University of Technology, A-1040 Vienna, Argentinierstr. 8/181-1, Austria

ARTICLE INFO

Article history:
Received 30 October 2011
Received in revised form
6 April 2012
Accepted 17 May 2012
Available online 28 May 2012

Keywords:
Cloud computing
Service virtualization
SLA negotiation
Service brokering
On-demand deployment
Self-management

ABSTRACT

Cloud computing is a newly emerged computing infrastructure that builds on the latest achievements of diverse research areas, such as Grid computing, Service-oriented computing, business process management and virtualization. An important characteristic of Cloud-based services is the provision of non-functional guarantees in the form of Service Level Agreements (SLAs), such as guarantees on execution time or price. However, due to system malfunctions, changing workload conditions, hard- and software failures, established SLAs can be violated. In order to avoid costly SLA violations, flexible and adaptive SLA attainment strategies are needed. In this paper we present a self-manageable architecture for SLA-based service virtualization that provides a way to ease interoperable service executions in a diverse, heterogeneous, distributed and virtualized world of services. We demonstrate in this paper that the combination of negotiation, brokering and deployment using SLA-aware extensions and autonomic computing principles are required for achieving reliable and efficient service operation in distributed environments.

1. Introduction

The newly emerging demands of users and researchers call for expanding service models with business-oriented utilization (agreement handling) and support for human-provided and computation-intensive services [1]. Though Grid Computing [2] has succeeded in establishing production Grids serving various user communities, and both Grids and Service Based Applications (SBAs) already provide solutions for executing complex user tasks, they are still lacking non-functional guarantees. Providing guarantees in the form of Service Level Agreements (SLAs) is also highly studied in Grid Computing [3–5], but they have failed to be commercialized and adapted for the business world.

Cloud Computing [1] is a novel infrastructure that focuses on commercial resource provision and virtualization. These infrastructures are also represented by services that are not only used but also installed, deployed or replicated with the help of virtualization. These services can appear in complex business processes, which further complicates the fulfillment of SLAs. For example, due to changing components, workload and external conditions, hardware, and software failures, already established SLAs may be violated. Frequent user interactions with the system during SLA negotiation and service executions (which are usually necessary in case of failures), might turn out to be an obstacle for the success of Cloud Computing. Thus, there is demand for the development of SLA-aware Cloud middleware, and application of appropriate strategies for autonomic SLA attainment. Despite business-orientation, the applicability of Service-level agreements in the Cloud field is rarely studied yet [6]. Most of the existing works address provision of SLA guarantees to the consumer and not necessarily the SLA-based management of loosely coupled Cloud infrastructure. In such systems, it is hard to react to unpredictable changes and localize, where the failures have happened exactly, what is the reason for the failure and which reaction should be taken to solve the problem. Such systems are implemented in a proprietary way, making it almost impossible to exchange the components (e.g. use another version of the broker).

Autonomic Computing is one of the candidate technologies for the implementation of SLA attainment strategies. Autonomic systems require high-level guidance from humans and decide which steps need to be done to keep the system stable [7]. Such systems constantly adapt themselves to changing environmental conditions. Similar to biological systems (e.g. human body) autonomic systems maintain their state and adjust operations considering their changing environment. Usually, autonomic systems comprise one or more managed elements e.g. QoS elements.

* Corresponding author.
E-mail addresses: keratt@inf.u-szeged.hu, keratt@sztaki.hu (A. Kertesz), kecskemeti@sztaki.hu (G. Kecskemeti), ivona@infosys.tuwien.ac.at (I. Brandic).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.05.016

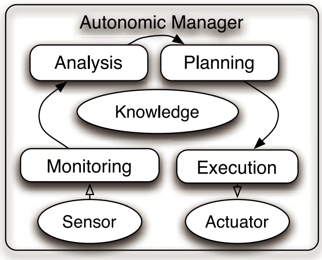

Fig. 1. General architecture of an autonomic system.

An important characteristic of an autonomic system is an intelligent closed loop of control. As shown in Fig. 1, the Autonomic Manager (AM) manages the element's state and behavior. It is able to sense state changes of the managed resources and to invoke an appropriate set of actions to maintain some desired system state. Typically control loops are implemented as MAPE (monitoring, analysis, planning, and execution) functions [7]. The *monitor* collects state information and prepares it for *analysis*. If deviations to the desired state are discovered during the analysis, the *planner* elaborates change plans, which are passed to the *executor*. For the successful implementation of the autonomic principles to loosely coupled SLA-based distributed system management, the failure source should be identified based on violated SLAs, and firmly located considering different components of the heterogeneous middleware (virtualization, brokering, negotiation, etc. components). Thus, once the failure is identified, Service Level Objectives (SLOs) can be used as a guideline for the autonomic reactions.

In this paper we propose a novel holistic architecture considering resource provision using a virtualization approach combined with business-oriented utilization used for SLA agreement. Thus, we provide an SLA-coupled infrastructure for on-demand service provision based on SLAs. First we gather the requirements of a unified service architecture, then present our solution called SLA-based Service Virtualization (SSV) built on agreement negotiation, brokering and service deployment combined with business-oriented utilization. We further examine this architecture and investigate how previously introduced principles of Autonomic Computing appear in the basic components of the architecture in order to cope with changing user requirements and on-demand failure handling. After presenting our proposed solution, we evaluate the performance gains of the architecture through a more computationally-intensive biochemical use case.

The main contributions of this paper include: (i) the *presentation* of the novel loosely coupled architecture for the SLA-based Service Virtualization and on-demand resource provision, (ii) the description of the architecture including *meta-negotiation, meta-brokering, brokering and automatic service deployment* with respect to the principles of autonomic computing, and (iii) the *evaluation* of the SSV architecture with a biochemical case study using simulations.

In the following section we summarize related works. Then, in Section 3, we provide the requirement analysis for autonomous behavior in the SSV architecture through two scenarios. Afterwards, in Section 4, we introduce the SSV architecture, while in Section 5 the autonomous operations of the components are detailed. In Section 6 we present the evaluation of the SSV architecture with a biochemical case study in a heterogeneous simulation environment. Finally, Section 7 concludes the paper.

2. Related work

Though Cloud Computing is highly studied, and a large body of work has been done trying to define and envision the boundaries of this new area, the applicability of Service-level agreements in the Cloud and in a unified distributed middleware is rarely studied. The envisioned framework in [8] proposes a solution to extend the web service model by introducing and using semantic web services. The need for SLA handling, brokering and deployment also appears in this vision, but they focus on using ontology and knowledge-based approaches. Most of the related works consider virtualization approaches [9–11] without taking care of agreements or concentrate on SLA management neglecting the appropriate resource virtualizations [12,5]. Works presented in [13,14] discuss incorporation of SLA-based resource brokering into existing Grid systems, but they do not deal with virtualization. Venugopal et al. propose a negotiation mechanism for advanced resource reservation using the alternate offers protocol [15]; however, it is assumed that both partners understand the alternate offers protocol. Lee et al. discusses application of autonomic computing to the adaptive management of Grid workflows [16] with MAPE (Monitoring, Analysis, Planning, Execution) decision making [7], but they also neglect deployment and virtualization. The work by Van et al. [17] studied the applicability of autonomic computing to Cloud-like systems, but they almost exclusively focus on virtualization issues like VM packing and placement.

In [18], Buyya et al. suggest a Cloud federation oriented, just-in-time, opportunistic and scalable application services provisioning environment called InterCloud. They envision utility oriented federated IaaS systems that are able to predict application service behavior for intelligent down- and up-scaling infrastructures. Then, they list the research issues of flexible service to resource mapping, user- and resource-centric QoS optimization, integration with in-house systems of enterprises, and scalable monitoring of system components. Though they address self-management and SLA handling, the unified utilization of other distributed systems is not studied. Recent Cloud Computing projects, e.g. Reservoir [19] and OPTIMIS [20], address specific research topics like Cloud interoperability, but they do not consider autonomous SLA management across diverse distributed environments. Comparing the currently available solutions, autonomic principles are not implemented in a adequate way because they are lacking an SLA-coupled Cloud infrastructure, where failures and malfunctions can be identified using well defined SLA contracts.

Regarding high-level service brokering, LA Grid [21] developers aim at supporting grid applications with resources located and managed in different domains. They define broker instances, each of them collecting resource information from its neighbors and saving the information in its resource repository. The Koala grid scheduler [22] was redesigned to inter-connect different grid domains. They use a so-called delegated matchmaking (DMM), where Koala instances delegate resource information in a peer-to-peer manner. Gridway introduced a Scheduling Architectures Taxonomy to form a grid federation [23,24], where Gridway instances can communicate and interact through grid gateways. These instances can access resources belonging to different Grid domains. Comparing the previous approaches, we can see that all of them use high level brokering that delegates resource information among different domains, broker instances or gateways. These solutions are almost exclusively used in Grids; they cannot co-operate with different brokers operating in pure service-based or Cloud infrastructures.

Current service deployment solutions do not leverage their benefits on higher level. For example the Workspace Service (WS) [9] as a Globus incubator project supports a wide range of scenarios involving virtual workspaces, virtual clusters and service deployment from installing a large service stack to deploying a single WSRF service if the Virtual Machine (VM) image of the service is available. It is designed to support several virtual machines. The XenoServer open platform [10] is an open