SLA based Service Brokering in Intercloud Environments.

CONFERENCE PAPER · APRIL 2012

CITATIONS
13

READS
27

3 AUTHORS, INCLUDING:

Foued Jrad
Karlsruhe Institute of Technology
14 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Jie Tao
Karlsruhe Institute of Technology
126 PUBLICATIONS **1,124** CITATIONS
SEE PROFILE

SLA BASED SERVICE BROKERING IN INTERCLOUD ENVIRONMENTS

Foued Jrad, Jie Tao and Achim Streit
Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Karlsruhe, Germany
{foued.jrad, jie.tao, achim.streit}@kit.edu

Keywords: Cloud Broker, Intercloud Computing, Cloud Interoperability, Service Level Agreement (SLA), OCCI.

Abstract: The fast emerging Cloud computing market over the last years resulted in a variety of heterogeneous and less interoperable Cloud infrastructures. This leads to a challenging and urgent problem for Cloud users when selecting their best fitting Cloud provider and hence it ties them to a particular provider. A new growing research paradigm, which envisions a network of interconnected and interoperable Clouds through the use of open standards, is Intercloud computing. This allows users to easily migrate their application workloads across Clouds regardless of the underlying used Cloud provider platform. A very promising future use case of Intercloud computing is Cloud services brokerage. In this paper, we propose a generic architecture for a Cloud service broker operating in an Intercloud environment by using the latest Cloud standards. The broker aims to find the most suitable Cloud provider while satisfying the users' service requirements in terms of functional and non-functional Service Level Agreement parameters. After discussing the broker value-added services, we present in detail the broker design. We focus especially on how the expected SLA management and resource interoperability functionalities are included in the broker. Finally, we present a realistic simulation testbed to validate and evaluate the proposed architecture.

1 INTRODUCTION

Cloud computing has recently emerged as a new key technology for outsourcing organizations' IT infrastructures on an economical basis. It allows a dynamic provisioning of virtual hardware and scalable applications according to their needs using a transparent easy "pay as you go" pricing model. Over the last years the number of Cloud service providers has significantly increased. On the other hand "vendor lock in" issues and the lack of common Cloud standards hindered the interoperability across these providers. Thus, today the Cloud customer is facing a challenging problem of selecting the appropriate Cloud offers that fits his needs. Therefore, standardized interfaces and intermediate services are needed to prevent monopolies of single Cloud providers.

The introduced vision of global interconnected Clouds called Intercloud (Bernstein et al., 2009), much like the internet as network of networks, addresses the above interoperability issues with the large focus on open standardized interfaces. Hereby Cloud consumers should be able to freely choose and seamlessly switch between different Cloud platforms while satisfying the demands for the guaranteed quality of service (QoS).

The common future use cases and functional requirements for Intercloud computing are defined in (GICTF, 2010). One of these promising use cases is market transactions via brokers. In such a use case a broker entity acts as a mediator between the Cloud consumer and multiple interoperable Cloud providers. The broker supports the consumer by selecting the provider that better meets his requirements. L. Frank Kenney, former research director at Gartner, claimed (Gartner, 2009) the need for Cloud brokers: "*The future of cloud computing will be permeated with the notion of brokers negotiating relationships between providers of cloud services and the service customers*".

Motivated by the above considerations, we present in this paper a generic architecture for a Cloud service broker operating in an Intercloud environment. Our proposed approach brings the following three benefits: (1) It allows a seamless access to Cloud resources through a standardized abstraction layer between consumers and providers; (2) It enables through a uniform interface the monitoring and management of deployed Cloud

services hosted on several Cloud providers by hiding their underlying technical details; (3) It finds the most suitable Cloud resources taking into account of user requirements specified by SLA.

The rest of this paper is organized as follows: in the next section we discuss prior works related to Cloud service brokering. We also identify how our work differs from related work. Section 3 presents the generic architecture of the broker. In section 4 and section 5 we propose a simulation environment for the broker and present first evaluation results. Section 6 concludes the paper with a brief summary and describes our future research directions.

2 RELATED WORK

The problem of multi domain service brokering in Cloud has received a lot of attention in academia and industry in the recent years.

(Buyya et al., 2010) presented the architecture of a federated Cloud computing environment named InterCloud to support the scaling of applications across multiple vendor Clouds. The idea behind their introduced federation concept is to enhance Cloud providers provisioning capabilities in case of sudden spikes in workload by leasing available computational and storage capabilities from other Cloud service providers. The main components of the proposed architecture are a Cloud Broker, a Cloud Exchange and a Cloud Coordinator. A client initiates a Cloud broker in order to meet the specified QoS targets, whereas Cloud Coordinators, acting as gateway between their internal datacenters and external Clouds, publish their services to the federation. Cloud Exchange acts as a mediator bringing together service providers and customers. It aggregates infrastructure demands from the application brokers and matches them against the available resources published by the Cloud Coordinators. The proposed architectural framework is still a research vision and its development is planned in context of the CLOUDBUS[6] project. However, the simulation results showed that the federation approach brings significant benefits to user's application performance.

(Theilmann et al., 2010) presented a flexible framework for multi-level SLA management within Clouds developed in context of the SLA@SOI[2] EU project. The core framework consists of a Business Manager and an SLA Manager. The Business Manager controls all the relations between customers and providers, whereas the SLA Manager deals with all the SLA related issues including negotiation, provisioning and monitoring. Besides the core framework, a domain-specific Service Manager provides management functionalities for the SLA Manager by interfacing the native provisioning system. The main contribution of the SLA@SOI framework is that the service quality can be predicted and enforced at run-time through an automated SLA management.

(Metsch et al., 2010) implemented a prototype broker architecture based on a combination of the core SLA@SOI framework and the RESERVOIR[1] framework. This latter allows an easy and on-demand provisioning of virtualized infrastructure resources within a federated Cloud platform. In their presented architecture, the core SLA@SOI framework acts as an SLA-based broker, whereas the RESERVOIR sites act as SLA@SOI third party providers and candidates for SLA provisioning. The interoperability between the two Cloud frameworks is achieved by implementing a standardized Service Manager interface using the Open Cloud Computing Interface API (OCCI, 2011).

(Kertesz et al., 2011) investigated the use of autonomic computing principles for resource management and SLA enforcement in Cloud environments. They proposed an SLA-based Service Virtualization (SSV) architecture, which is built on three main components: a Meta-Negotiator responsible for agreement negotiations, a Meta-Broker for selecting the proper execution environment and an Automatic Service Deployer for service virtualization and on-demand deployment.

As the first industry driven project, the TM Forum[3] Cloud Service Broker Catalyst explored the role of a value-added service broker by demonstrating a proof of concept for a trusted and transparent Cloud management platform.

A market analysis of the current commercial Cloud broker solutions shows that most products concentrate on the aggregation and mediation of the services deployed on well-known public Cloud providers by providing integrated management and monitoring interfaces. From the few products offering additional brokering features, we refer to SensibleCloud[4] and CloudSwitch[5]. While the former offers an automated SLA based multi-Cloud management, the latter allows customer to select the

[1] http://www.reservoir-fp7.eu//
[2] http://www.sla-at-soi.eu/
[3] http://www.tmforum.org
[4] http://www.sensiblecloud.com/
[5] http://www.cloudswitch.com
[6] http://www.cloudbus.org

best suitable Cloud service for their needs.

In summary, the work in (Buyya et al., 2010), (Kertesz et al., 2011) and (Metsch et al., 2010) are the most related works to this paper. The desired broker architecture in this work acquired some ideas from the previous designs. However, we have designed a high-level generic architecture by integrating several state of the art technologies and standards. First, our solution combines all the brokering features included in the previous works like SLA management, service deployment and monitoring and provider selection. Second, we provide an abstraction layer to hide the technical details of Cloud providers by using current Intercloud standards. Moreover, our implemented simulation environment prepares a realistic testbed to validate and evaluate the proposed architecture.

3 CLOUD SERVICE BROKER DESIGN

In this section we discuss the design of our proposed Cloud service broker.

3.1 Overall Architecture

Figure 1 shows a generic architecture of the service broker. The main components and their roles are gathered in Table 1.

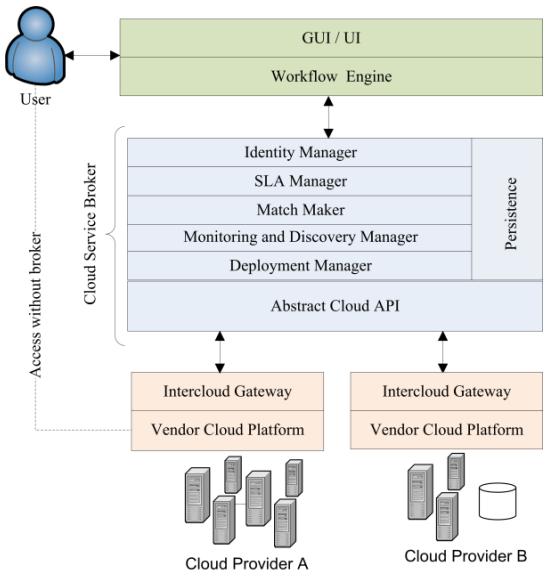

Figure 1: Cloud service broker architecture.

As depicted in Figure 1, the service broker acts as a mediator between the Cloud user and multiple Cloud providers by providing attractive value-added services to users on top of heterogeneous Cloud vendors. The main functions of the broker are assisting the user in finding the best provider for his service needs with respect to specified SLA and providing him with a uniform interface to manage and monitor the deployed services.

The needed interactions between the different broker components during a service lifecycle and therefor required interfaces to Cloud providers are described in detail in the following sections.

Table 1: Components of Cloud service broker architecture.

Component	Role
SLA Manager	It negotiates the SLA creation and handles the SLA provisioning.
Monitoring and Discovery Manager	It queries resource information and monitors the SLA metrics.
Match Maker	It selects the best Cloud providers for user requests using different matching algorithms.
Deployment Manager	It deploys the service on the selected provider.
Identity Manager	It handles the user authentication and ensures IDs and roles enforcements.
Persistence	It stores broker specific data (e.g. monitoring, SLA templates and resources data).
Abstract Cloud API	A standard abstract API used to manage Cloud resources on different Cloud providers.
Intercloud Gateway	It acts as service frontend for the Cloud provider and interacts through the standard Cloud API with the broker.
Vendor Cloud Platform	The native Cloud platform hosted by the Cloud provider.

3.2 SLA Management

In (Linlin et al., 2010) an SLA is defined as a formal contract between service providers and consumers to guarantee that consumers' service quality expectation can be achieved. According to them, the SLA lifecycle in utility computing systems has six steps, which are 'discover service providers', 'define SLA', 'establish agreement', 'monitor SLA violation', 'terminate SLA' and 'enforce penalties for violation'.

In this work the broker acts on behalf of the consumer to process all the SLA management tasks. In the following subsections we discuss how SLA management is handled in our proposed architecture throughout the entire SLA lifecycle.

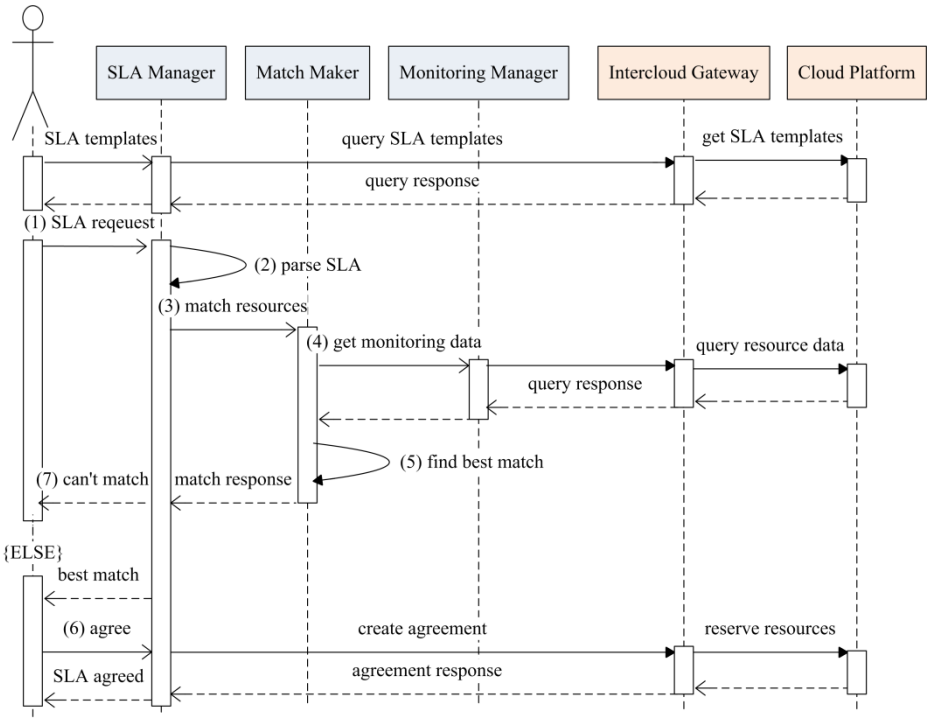

Figure 2: The SLA negotiation flow.

3.2.1 SLA Discovery and Definition

In order to discover the SLA features supported by the different Cloud providers the user asks, with the assistance of a user interface, the SLA Manager to retrieve the provider SLA templates published through the Intercloud Gateways. An SLA template represents amongst others the QoS parameters and the margin values that the provider can accept (e.g. supported OS, gold or silver SLA level). In a next step, the user can populate a suitable template with required values or even define a new SLA from scratch to describe the functional and non-functional service requirements. Some typical SLA parameters used in context with IaaS Cloud provisioning are provided in Table 2. These parameters are important for a provider matching by the broker and therefore the user needs to give reasonable values for them.

Table 2: Sample SLA parameters for IaaS.

functional	**non-functional**
CPU cores	response time
memory size	budget
CPU speed	completion time
in/out bandwidth	data transfer time
OS type	availability
storage size	persistence (Yes/No)
image URL	reservation (Yes/No)

3.2.2 SLA Negotiation

In this step a negotiation process managed by the broker is started in order to reach an SLA agreement between the user and the appropriate Cloud providers. The ideal SLA negotiation flow of an incoming SLA user request to the service broker is illustrated by the sequence diagram in Figure 2.

The SLA negotiation is done as follows: The user submits a service request (1) with the previous defined SLA to the SLA Manager. Then, the SLA Manager, after parsing the SLA definition (2), asks the Match Maker, if it could execute the service with the specified requirements (3). In order to respond to this request, the Match Maker starts a match making process to find the best suitable provider (5) by matching the gathered resource properties (e.g. datacenter configuration and supported SLA metrics) from the Monitoring Manager (4) with the service requirements by applying predefined matching algorithms. At the end, the user gets a response to his request from the SLA Manager with the respective matching results. Upon user acceptance, an agreement can be established (6) with the matched provider and the required resources could be reserved. If none of the providers can be matched, the aforementioned steps may be repeated for renegotiation (7) until reaching an agreement.

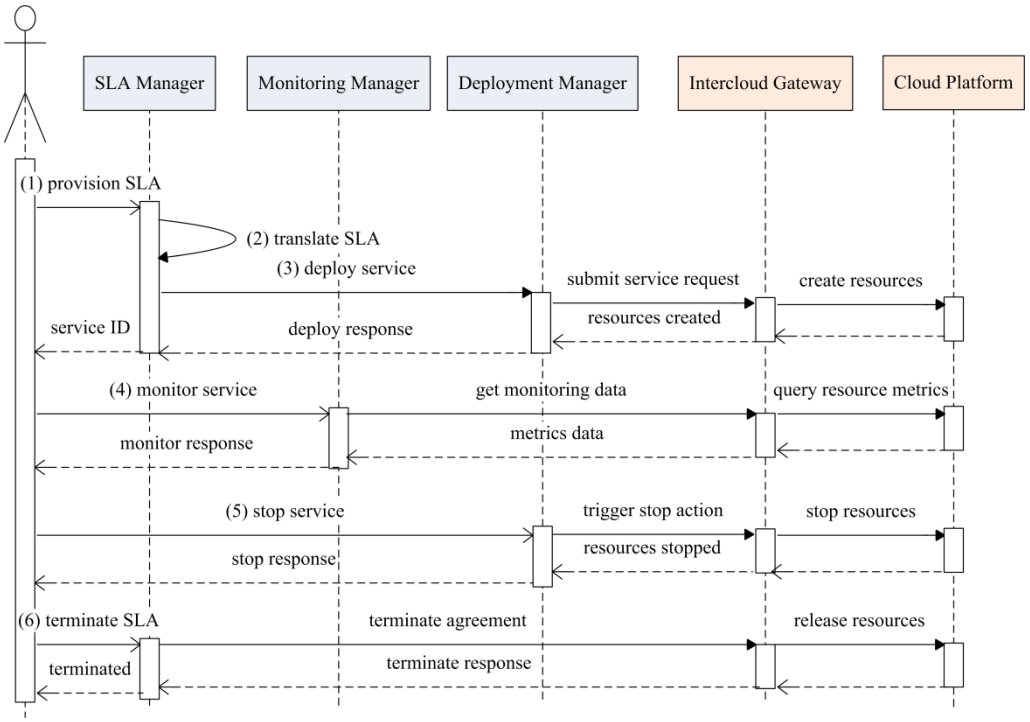

Figure 3: The SLA provisioning and monitoring flow.

3.2.3 SLA Provisioning and Monitoring

After establishing an agreement, a provision request (1) is submitted by the user to the SLA Manager. The interactions needed by the broker components to handle the provision request are shown by the sequence diagram in Figure 3.

The SLA provisioning is done as follows: After receiving the provision request, the SLA Manager translates (2) the associated SLA to a service request and asks the Deployment Manager to deploy the service (3). The Deployment Manager forwards the request to the appropriate Intercloud Gateway to create and start the requested resources. As response a unique service ID, used to address the deployed service, is returned. The user is then able to monitor the service (4) by requesting the metric values of the agreed QoS parameters from the Monitoring Manager. The user is also able to perform actions on the deployed resources (e.g. start and stop) (5). Once an agreement is terminated (6), all the created resources should be released by the Cloud platform.

3.3 Provider Intercloud Gateways

The entire communication between the broker and Cloud providers is realized through Intercloud Gateways. This key component runs on the provider side by interfacing the vendor Cloud platform. It provides management and monitoring interfaces while hiding the internal provider policies. The concept of Intercloud Gateways is similar to the Cloud Coordinator in (Buyya et al., 2010), with the difference in that, the former simply interacts only with the broker, whereas the latter needs interaction with both the Cloud Exchange and the Cloud broker.

While looking for an abstract Cloud API compatible with our architecture, we found that the Open Cloud Computing Interface specification (OCCI, 2011) is the most applicable among existing APIs. OCCI is an extensible specification for remote management of Cloud infrastructures, allowing the development of interoperable tasks. The current OCCI specification focuses on IaaS offerings by defining three abstract resource types, which are compute, storage and network. All the operations for creating, deleting, starting and stopping these resources can be requested on a REST manner over HTTP methods (GET, POST, PUT and DELETE).

By integrating OCCI in our architecture, the Intercloud Gateway becomes an OCCI server, whereas the abstract Cloud API is the OCCI Client. In this way, the entire communication between the broker and providers runs through standard OCCI interfaces and no adapters are needed in the broker.

4 SIMULATION TESTBED

We developed based on the CloudSim simulation toolkit (CoudSim, 2011) a simulation environment for the broker. This allows us to validate and evaluate the proposed broker architecture without the setup of a real testbed, which is extremely time and cost consuming. CloudSim is a scalable open source simulation tool offering features like support for modeling and simulation of large scale Cloud computing infrastructures including datacenters, brokers, hosts and VMs on a single host.

We extended CloudSim with an Intercloud Gateway package to simulate an OCCI-based service frontend. For this, we used the open source (OCCI4JAVA, 2011) OCCI implementation to mediate the CloudSim DatacenterBroker class. In contrast to the OCCI specification, we do not use a REST based communication between the Cloud service broker and the Intercloud Gateway, as the CloudSim simulation runs on one host. In such way, only minimal changes are needed in the broker to use it with real OCCI enabled production Clouds. Moreover, we implemented an OCCI monitoring mixin to allow the Intercloud Gateway to query resource properties and metrics from CloudSim.

5 EVALUATION RESULTS

In order to assess the scalability of our proposed approach, an experiment has been conducted: We continuously generate VM requests (at a rate of 1 req/min) and let the broker select a random provider from 6 datacenters (each made up of 50 hosts with 2 CPUs) and then deploy the requested VM on it. The VM can be one of the 4 Amazon EC2 instance types (micro, small, large and high CPU). We measured the percentage of successfully deployed VMs by varying the request number from 50 to 2000. We repeated the experiment with only one and then with three datacenters (DC). The results presented in Figure 4 show that our approach scales well with increasing number of service requests and providers.

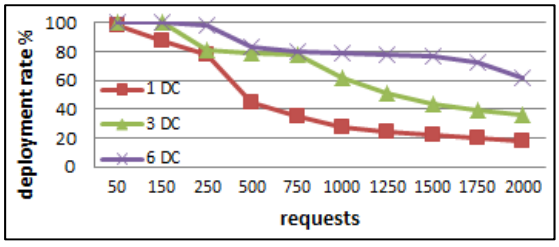

Figure 4: Cloud service broker deployment efficiency.

6 CONCLUSIONS

The lack of interoperability and the heterogeneity in current public Cloud platforms justify the need for intermediate broker services to assist the user in finding the appropriate provider for his needs.

In this paper we described the architecture design of a Cloud service broker and proposed a simulation environment to test and evaluate the broker. Especially, we focused on the SLA management and interoperability features included in the broker.

In our future work, we will use simulation to investigate and evaluate the performance and efficiency of different SLA-aware match making algorithms by supporting multiple SLA parameters. Furthermore, we will investigate the execution of workflows with the help of the Cloud service broker and improve the automated SLA negotiation in the broker by implementing a new OCCI SLA mixin.

REFERENCES

Bernstein. D. et al. (2009). Blueprint for the Intercloud: Protocols and Formats for Cloud Computing Interoperability. In *ICIW2009*, *4th International Conference on Internet and Web Applications and Services,* pages 328-336.

Buyya R., Ranjan R. and Calheiro R. N. (2010). InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In *ICA3PP2010, 10th International Conference on Algorithms and Architectures for Parallel Processing,* pages 13–31.

CloudSim 2.1.1 (2011). *CLOUDS Lab, Unversity of Melbourne*, http://www.cloudbus.org/cloudsim/.

Gartner (2009). Cloud Consumers Need Brokerages to Unlock the Potential of Cloud Services. *Press Release.*

GICTF (2010). Use Cases and Functional Requirements for Inter-Cloud Computing. *White Paper 9-8-2010.*Kertesz A., Kecskemeti G. and Brandic I. (2011). Autonomic SLA-aware Service Virtualization for Distributed Systems. In *PDP2011*, *19th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 503-510.

Linlin Wu and Buyya R. (2010). Service Level Agreement (SLA) in Utility Computing Systems. *Tech. Rep. 2010.*

Metsch T., Edmonds A. and Bayon V. (2010). Using Cloud Standards for Interoperability of Cloud Frameworks. *Tech. Rep. 2010.*

OCCI v1.1 (2011). Open Cloud Computing Interface specification. *OCCI-WG*, http://www.occi-wg.org.

OCCI4JAVA v0.2 (2011). https://github.com/occi4java.

Theilmann W., Happe J., Kotsokalis C., Edmonds A. and Kearney K. (2010). A Reference Architecture for Multi-Level SLA Management. In *Journal of Internet Engineering, Vol 4, No 1, December 2010.*