Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 1167 – 1176

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

A Dynamic Data-Driven Simulation Approach for Preventing Service Level Agreement Violations in Cloud Federation

Funmilade Faniyi[a,*], Rami Bahsoon[a], Georgios Theodoropoulos[b,1]

[a] *School of Computer Science, The University of Birmingham, B15 2TT, United Kingdom*
[b] *IBM, Research Lab, Dublin, Ireland*

Abstract

The new possibility of accessing an infinite pool of computational resources at a drastically reduced price has made cloud computing popular. With the increase in its adoption and unpredictability of workload, cloud providers are faced with the problem of meeting their service level agreement (SLA) claims as demonstrated by large vendors such as Amazon and Google. Therefore, users of cloud resources are embracing the more promising cloud federation model to ensure service guarantees. Here, users have the option of selecting between multiple cloud providers and subsequently switching to a more reliable one in the event of a provider's inability to meet its SLA. In this paper, we propose a novel dynamic data-driven architecture capable of realising resource provision in a cloud federation with minimal SLA violations. We exemplify the approach with the aid of case studies to demonstrate its feasibility.

Keywords:
Cloud Computing, Cloud Federation, SLA Management, Market Mechanism, Distributed Simulation

1. Introduction

Cloud computing has gained popularity over the last few years, borrowing ideas from grid and utility computing. Cloud-based systems are characterised by provision of a large pool of computational resources accessible over the Internet on-demand, with the capacity to elastically scale as needed [1]. By leveraging cloud services, organisations have the potential of gaining access to previously unattainable resources, scaling/shrinking these resources based on their demand and paying for only actual resource usage. This mode of payment has the benefit of saving the cost incurred by users since most vendors offer their services on a pay-per-use or subscription basis [2].

Industrial, governmental and academic stakeholders are already making innovative use of the cloud. For example, researchers are devising ways of outsourcing scientific experiments to the cloud [3], while academics are already designing educational solutions around cloud technology to facilitate learning and laboratory work among students [4]. In this paper, we limit our scope to public cloud service providers (CSPs) or simply 'cloud providers'. Examples include Amazon Web Services (AWS) (http://aws.amazon.com/) and Rackspace (http://www.rackspace.com/).

*Corresponding author
Email addresses: fof861@cs.bham.ac.uk (Funmilade Faniyi), r.bahsoon@cs.bham.ac.uk (Rami Bahsoon), georgheo@ie.ibm.com (Georgios Theodoropoulos)

[1] This research was initiated while Georgios Theodoropoulos was with the School of Computer Science, University of Birmingham, UK.

doi:10.1016/j.procs.2012.04.126

The increase in the adoption of cloud computing has placed high demand on cloud resources. Consequently, cloud providers are faced with a challenging problem of being unable to fulfil the claims made in their service level agreements (SLAs). According to [5], within a six-month period in 2011, several top cloud providers (e.g. Amazon, Google, Microsoft) experienced service outages which sometimes lasted for periods ranging from few hours up to one week. Common sources of these SLA violations are unanticipated outages caused by software, hardware or network faults [6]. This problem is further exacerbated by the high volatility of the cloud environment [7] and the unforeseen workload on cloud data centres which results in unpredictable performance [8]. The uncertainty about the quality of service (QoS) of cloud services reduces user confidence in the technology. Hence, promoting the perception of cloud as "unreliable" for critical applications.

To reverse this trend, users that rely on the single cloud provision model are embracing the more flexible and resilient cloud federation model [9, 10]. A cloud federation consists of multiple cloud providers who are able to seamlessly interact among themselves. The hypothesis is that as cloud computing evolves, next generation clouds would have the capability to form a federation where it will be possible to leverage on each others' computational resources [11]. This provides an opportunity to mitigate risks of violating cloud users' SLAs by migrating jobs among providers in the federation.

In order for the cloud federation to actualise satisfactory SLA compliance; a middleware layer for coordinating the activities of cloud users and cloud providers in the federation is required [9]. Crucially, this middleware layer must be highly adaptable to changing conditions in the cloud environment and rapidly respond to events that may trigger SLA violation. Therefore, the middleware requires mechanisms for dynamic scaling to accommodate heavy workload, detect imminent failure of cloud providers, and migrate jobs among clouds in the federation at run-time.

In previous work [12], we motivated the need for engineering highly robust, adaptable and scalable cloud systems using the dual concepts of self-awareness and self-expression. Key among the requirements for these next generation clouds is the efficient and transparent management of jobs deployed on them to prevent SLA violation. In this paper, we take this work further by outlining the design of an architecture which realises the requirement of preventing SLA violations within the context of federated cloud. The proposed cloud federation architecture incorporates a novel middleware layer which is designed based on the principles of the Dynamic Data-Driven Application Systems (DDDAS) paradigm. The middleware layer selects cloud providers on behalf of users, and subsequently manages the execution of submitted jobs to ensure they are successfully provisioned within the bound of specified SLA constraints.

The rest of the paper is structured as follows: Section 2 describes foundational work in cloud federation and existing SLA management techniques. Section 3 presents an illustrative example to highlight the unique properties of the cloud federation model. The design of the novel cloud federation middleware is discussed in section 4. A case study is presented in section 5 to demonstrate the feasibility of the approach. The paper concludes in section 6.

2. Related Work

This section presents relevant research in cloud federation SLA management. Due to the newness of the research area, we also consider research efforts for SLA management in single cloud provision model.

The inability of cloud providers to meet satisfactory QoS levels as specified in SLAs led to the vision of cloud federation. Buyya et al. [10] envisioned the federated cloud computing (or InterCloud) model as an environment that could flexibly respond to variations in workload, network and resource conditions by dynamically coordinating multiple clouds in the federation. Since it is infeasible for a cloud provider to have data centres in every country, the federated cloud environment offers the additional benefit of rapidly scaling to meet the needs of geographically distributed cloud users than any single cloud provider [10]. The RESERVOIR project [9] also sets out a vision similar to [10] for an open federated cloud computing model to address the limited scalability of single cloud providers and lack of interoperability among them.

These works [10, 9] identified the importance of the middleware coordination layer in the cloud federation. According to [9], this middleware layer (referred to as Service Manager in their work) is the highest level of abstraction responsible for coordination of cloud providers and cloud users in the federation. Importantly, it ensures that cloud users' jobs are forwarded to cloud provider(s) who are capable of executing those jobs without violating SLA constraints. Our work builds on this vision in the design of the middleware layer using the DDDAS approach [13].

The work of [14] employed an autonomic resource provision technique to manage SLA in federated cloud. While their work considers all phases of the SLA life cycle, there is no explicit provision for post-negotiation causes of SLA

violations such as variation in workload. Brandic I. et al. [15] presented a proposal for SLA management in a single cloud infrastructure. Their work provides a method for mapping low-level resource metrics to high-level cloud user SLA specifications, and deducing the likelihood of SLA violations from this mapping. Another interesting approach is the autonomic resource allocator proposed by D. Ardagna et al. [16] for managing SLAs of multiple applications running on a single cloud. The authors considered SLA violation from the dimension of workload variation with the objective of maximising cloud providers' revenue. The problem of SLA violations in our work is addressed from the context of a cloud federation. In addition, we assume a broader set of events that may cause these violations, namely, heterogeneous user requests, workload variations and unavailability of cloud providers in the federation.

To tackle the problem at hand, we leverage on advances in the use of DDDAS for large-scale systems (e.g. grid computing [13]). The DDDAS approach provides a sound basis to reason about changes in user and cloud provider behaviour in the cloud federation. It also informs the design of simulations to efficiently match users' requests with providers' offerings, monitor job execution and rapidly adapt to risks to prevent SLA violations from occurring.

3. Motivation and Background

In this section, the single cloud provision model and its limitations at meeting cloud users' SLA are presented. Subsequently, we motivate the vision of the cloud federation paradigm and its potential to resolve the issues raised.

3.1. Illustrative Example

Consider Amadas cloud provider, an owner of three data centres distributed across the US and Europe. Amadas is specialised at providing cloud infrastructure-as-a-service solutions to companies who depend on it for resources required to host their applications. We consider four users of Amadas cloud: a science laboratory, a social networking portal with a global user-base, an e-Commerce portal and a global news website. Each of these cloud users have different application requirements. Specifically, the science lab uses the cloud to run data and computational intensive scientific experiments; the social networking portal performs read/write intensive operations; the news website performs mostly read-only operations while the e-Commerce website performs highly consistent transactional operations. Figure 1 depicts the distribution of the cloud users across Amadas cloud's data centres.

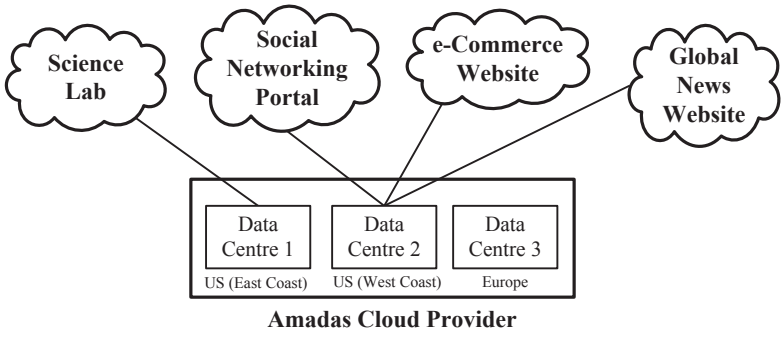

Figure 1: Single Cloud Provider Model

Suppose breaking a major news event raises high traffic from the social networking portal due to activities of its large user-base. This causes a *slashdot effect* that results in many of the social network portal's users visiting the global news website. As expected, the heavy traffic from the social networking and news services will cause a spike in the workload on Amadas' data centre 2. The resources in this data centre are therefore dedicated to the two highly demanding services, while other services depending on it (e.g. the e-Commerce website) are left with limited resources. This resource starvation is capable of making transactions performed on the e-Commerce website inconsistent, thus resulting in poor quality of service. Amadas' policy of not allowing coordination between its data centres makes it unlikely for system administrators to take mitigation measures (e.g. moving some of the services to the less demanded data centre 1) beforehand.

The scenario presented above describes only one cause of SLA violation (i.e. unpredictable workload) when using the single cloud provider model. Other causes could be resource failure in the data centre [5, 6] and security attack on cloud resources [17]. The occurrence of these scenarios typically results in the violation of cloud users' SLAs. Thereafter, Amadas suffers bad reputation and may be penalised in the form of monetary or service credit payment. The cloud federation paradigm offers a framework that could be leveraged upon to prevent this problem.

3.2. The Cloud Federation Model

The emergence of many cloud providers offering various services has propelled the vision of cloud federation [9, 10]. The proponents of the cloud federation model advocate that next generation cloud providers will have the capacity to seamlessly interact among themselves thereby taking advantage of economies of scale [11]. This would afford providers the possibility of outsourcing resources at run-time in the event of failure of any cloud provider in the federation. Opencirrus (https://opencirrus.org/) is an example of a testbed that is designed for cloud federation research. The cloud federation model is shown in figure 2.

Cloud users interact with the federation via a middleware layer. Therefore, the internal operation of cloud providers in the federation is transparent to the cloud users. The middleware layer coordinates interaction with cloud users and interaction among cloud providers in the federation. Each cloud provider is equipped with a cloud manager component which interfaces with the middleware layer and coordinates the resources of its cloud. All interaction with each cloud provider is via its cloud manager component.

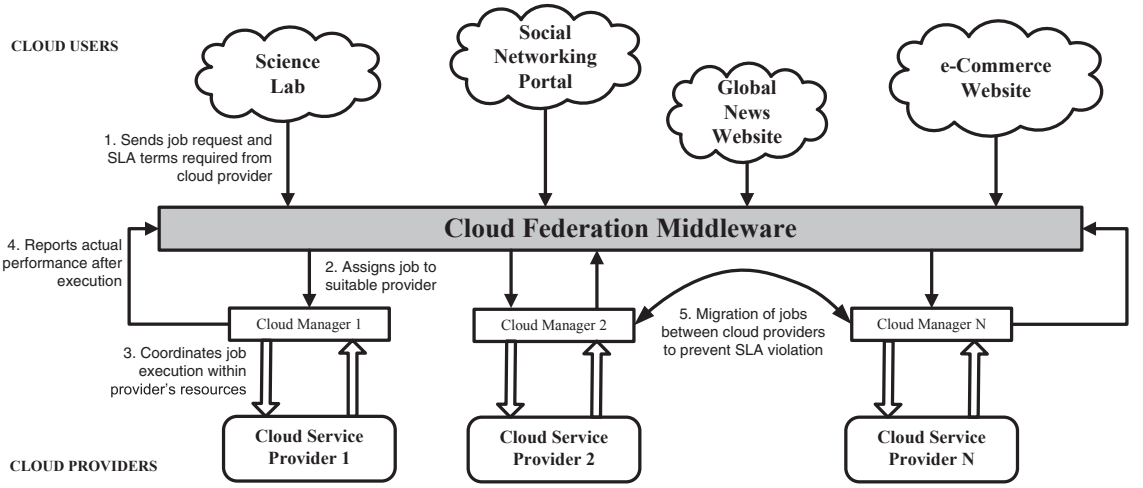

Figure 2: Cloud Federation Model

To fully realise the cloud federation model, there are a number of open research problems, such as: formalism of a language to inform negotiation among cloud providers at run-time [18], interoperability of data formats and interfaces (APIs) to facilitate inter-cloud communication [19], and middleware layer design for coordinating cloud federation resources [14].

The objective of our work is the design of a cloud federation middleware that is capable of matching cloud users' requests with cloud providers' offerings without violating cloud users' SLA. Due to the high volatility of the cloud environment, the middleware should continuously steer the federation based on the current status of cloud providers' offerings and requests demanded by users. We exploit the DDDAS approach [20] to achieve this.

4. The Approach

In this section, the design of the cloud federation architecture is presented. The middleware layer of the architecture incorporates the DDDAS approach to mitigate risks of violating SLAs.

4.1. Data-Driven Middleware for Cloud Federation

It is important to justify the choice of DDDAS as a promising approach for implementing this layer. Two drivers for pursuing the DDDAS approach are the characteristics of the cloud domain, and the requirement of the problem at hand. As illustrated in section 3.1, the cloud environment is highly volatile and exhibits dynamics primarily originating from two actors: cloud users and cloud providers. Cloud users have different application needs (e.g. computation, storage or read/write-intensive) and their demand for cloud resources varies based on the workload placed on their services. Similarly, cloud service providers' (CSPs) offerings are heterogeneous and vary from time to time e.g. due to resource failure or workload imposed on cloud infrastructure.

These properties necessitates an approach such as DDDAS, that provides a basis for reasoning about these dynamics and efficiently self-adapting to them at run-time. In particular, the adaptation should adhere to SLA constraints.

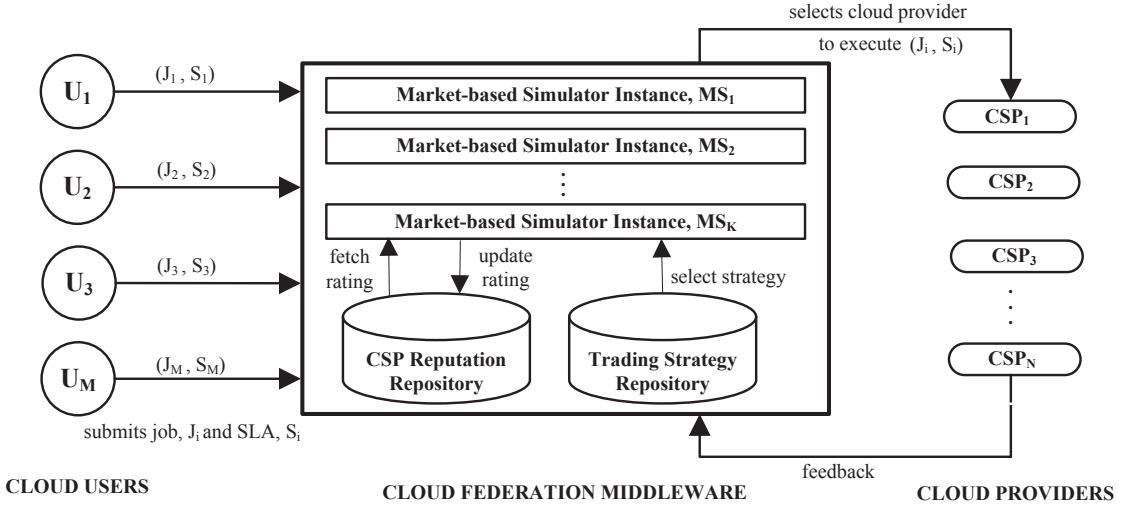

Figure 3: Cloud Federation Architecture incorporating Data-Driven Middleware Layer

The cloud federation architecture is shown in figure 3. The middleware layer in the architecture is composed of distributed simulator instances; this is similar to previous usage of the DDDAS approach in distributed domains (e.g. trust prediction in mobile networks [21]). These simulators receive requests from cloud users and select cloud providers capable of meeting these requests. The simulators receive control feedbacks from cloud providers signifying successful execution of submitted job requests or *risk alerts* signifying their inability to do so. Risk alerts could be triggered by reasons such as unanticipated resource failure and unforeseen spike in workload. The simulators act on this feedback by taking risk mitigation actions such as selection of substitute cloud provider(s) to avoid violating the SLA terms of submitted requests. Feedbacks received by the simulator can also be discriminated as: high priority (requires immediate intervention), medium priority (react within a time bound) or low priority (trivial, non-threatening risk). The knowledge acquired from the continuous interaction between simulators and CSPs improves the accuracy of the simulators at selecting reliable CSPs. Next, we discuss each component of the architecture in more detail.

4.2. Cloud Environment

The two major actors in the federated cloud environment are cloud users and cloud providers.

- Cloud Users: refers to individuals or organisations who interface with the cloud federation. Each cloud user typically owns some service(s) which it provisions to its clients. As illustrated in section 3.1, services may differ in their requirements. We model cloud users as a set $U = \{U_1, U_2, ..., U_M\}$, where M is the number of users requesting resources at a time instance. The requirements of services owned by cloud user, U_i, are bundled as a series of job request, J_i, with an associated SLA, S_i, which specifies the constraints for the requested job. The SLA, S_i, encompasses terms such as response time, availability and scalability constraints. To request cloud resources, users submit the 2-tuple (J_i, S_i) to the cloud federation via its middleware layer.

- Cloud Service Providers (CSPs): offer different specialised cloud services (X-as-a-service), where X could be storage, infrastructure, platform or software. The cloud manager component of each CSP (see figure 2) notifies the middleware about its resource capacity, pricing and service terms. The middleware utilises this information for selecting the CSP to execute cloud users' job requests. Importantly, the cloud manager sends notification about job completion or risk alerts to the middleware which acts on them to mitigate SLA violation. CSPs are modelled as a set $\{CSP_1, CSP_2, ..., CSP_N\}$, where N is the number of CSPs in the federation.

4.3. Cloud Federation Middleware

The middleware layer consists of many simulator instances which coordinate the cloud federation. The distributed simulation approach is preferred due to the scale of the cloud federation. A single simulator instance may constitute a bottleneck under heavy workload and thus make the middleware less scalable. Distribution of simulator instances improves the scalability of the middleware and affords dedication of simulators to multiple concerns. For example, some simulators may be more efficient at selecting CSP for specific types of incoming job requests than others. In accordance with the DDDAS paradigm, these simulators collect data based on current state of the cloud, perform measurements to detect probable violations and effect control changes to mitigate them.

More precisely, the simulators on this layer perform the following functions:

- receive job requests and associated SLAs as input from cloud users,
- inspect offerings of cloud service providers at specified intervals,
- select cloud service provider(s) to execute job requests based on job type and SLA terms,
- monitor job execution to detect risk alerts from cloud providers, and
- perform control actions to prevent the violation of users' SLAs whenever a risk alert is received.

In this work, we elaborate on the CSP selection, risk detection and SLA violation prevention steps. The simulators reaches the CSP selection decision by utilising a market control algorithm. In general, market-based control techniques (many of which are discussed in [22]) are particularly suited for managing large-scale systems because of their decentralised, robust and highly scalable properties. Simulator instances are modelled as a set $\{MS_1, MS_2, ..., MS_K\}$, where K is the number of active simulator instances in the middleware. The relevance of the trading strategy and reputation repository (in figure 2) will be discussed in the next section.

4.4. Market-Based Simulator

A distributed/decentralised market mechanism is required to coordinate the interaction of the distributed simulators. A good candidate for realising this goal is the retail-inspired posted-offer market mechanism [23]. This is because the mechanism assumes full decentralisation amongst market entities (i.e. buyers and sellers). There is no notion of a centralised auctioneer or market-maker as it exists in other computational market analogies (e.g. continuous-double auction [24]). The posted-offer mechanism has also been shown to reach an efficient allocation of resources among market entities by following computationally inexpensive negotiation steps [23]. For example, an adaptation of the mechanism has been used to manage resources in peer-to-peer grid computing [25].

Buyers in the market are modelled as simulator instances and sellers as CSPs. Buyers seek to maximise their success throughput (i.e. the number jobs allocated to sellers which are completed within SLA constraints), while sellers are interested in maximising their profit by continuously executing jobs according to specified SLA constraints.

4.4.1. Assumptions

We make the following assumptions in our adaptation of the posted-offer market mechanism:

1. Each cloud user, U_i, submits its job request to one simulator instance, MS_i at any time instance,
2. Every CSP has the capacity to execute each job requested by a cloud user, and
3. Market entities (buyers and sellers) are self-interested. This means they seek to maximise their objectives without cooperating with each other.

4.4.2. Computing Market Prices

One important element of any market mechanism is the pricing strategy adopted by both buyer and seller agents. Within the context of our market mechanism, items (i.e. jobs) are priced using 'artificial money' not real money. Essentially, we adhere to the principle of most market-based control systems [22], where 'artificial money' is used primarily as a control tool and not as a financial transaction instrument as it exists in real world markets.

Buying Price Computation: Buyers (simulator instances) determine their private valuation for a job based on the following SLA parameters: (i) job priority and (ii) expected job completion time. Their private valuation is determined by their utility function, which for buyer MS when allocating job J is defined by:

$$U_{MS}(J) = -\alpha U(x_1) + \beta U(x_2) + U(J) \tag{1}$$

where x_1 and x_2 represent price and reliability respectively; $U(x_1)$ and $U(x_2)$ are price and reliability distribution functions respectively; and $U(J)$ is the payoff derived from the execution of job J. The co-efficients α and β represent the priority and expected job completion time as defined in the SLA. The values of α and β are specified by the cloud user in the SLA such that the buyer's objective reflects the actual weight of the job. Consequently, the buyer seeks to maximise its utility function by meeting the job's SLA constraint.

Selling Price Computation: Sellers (CSPs) compete in the cloud federation market to increase their profit. This means that they make trading decisions based on the utility they expect to derive from the execution of the job. The utility of a seller, when executing a job J is defined by:

$$U_{CSP}(J) = S(J) - C(J) \tag{2}$$

where $S(J)$ is the selling price decision functions and $C(J)$ is the cost incurred for executing the job. Specific details about how these functions are defined depends on individual CSPs. Ultimately, the seller's objective is to maximise its utility function at any given time.

4.4.3. Trading Mechanism

The steps of the posted-offer mechanism for each trading round is described as follows:
Step 1: sellers (CSPs) publish the prices and service terms of their resource offerings.
Step 2: buyers search for any seller whose service term meets the SLA constraint, S_i of the job at hand.
Step 3: if buyer finds a matching seller, the job is allocate to it, then step 4, otherwise step 2.
Step 4: buyer (simulator instance) monitors the selected seller (CSP) at intervals.
Step 5: if seller (CSP) executes job successfully then it sends completion notification else a risk alert is sent.
Step 6: if buyer (simulator instance) detects risk alert or seller (CSP) is not responsive then step 2 else step 7.
Step 7: buyer (simulator instance) compares actual CSP job performance with SLA constraint.
Step 8: cloud user is notified of completed job.

We make the following important refinements to the posted-offer mechanism to capture the dynamics of the cloud federation environment. Since CSPs vary unpredictably in their ability to successfully execute allocated jobs, each CSP is assigned an initial reputation rating which is updated dynamically based on its actual performance. The CSP reputation repository (see figure 3) is used to capture these reputation ratings. Therefore, the following additional operations are performed in steps 2, 6 and 7.
Step 2:* buyers search for any seller in the reputation repository within an acceptable[2] reputation rating.
Step 6:* if buyer detects that seller (CSP) is not responsive then buyer updates reputation repository with a negative rating for the CSP.
Step 7:* if actual CSP performance violates SLA constraint, buyer updates reputation repository with a negative rating for the CSP otherwise updates with positive rating.

These refinements increase the probability of selecting a highly reliable CSP for jobs which are specified as high priority by cloud users. The trading strategy adopted by buyers (in step 2) also contributes to the selection of suitable

[2]The range of acceptable rating is defined by the job priority specified by the cloud user for the job at hand.

CSP for each job. The *substitution strategy* is implicitly specified in the mechanism (step 7 and 2). This is defined by the replacement of a CSP with another one when a risk alert is triggered or when it stops responding. Since buyers are self-interested and able to adopt diverse strategies; the trading strategy repository provides buyers with other predefined strategies from which they can select to maximise their utility. An example is *risk aversion strategy*, where a risk averse buyer splits up a job among multiple CSPs, to mitigate the risk of violation by any one CSP. Essentially, the priority and SLA constraints of a job are the deciding factors when selecting a trading strategy for it.

The global objective of the cloud federation market is to maximise the number of jobs successfully completed by sellers (CSPs) within SLA constraints. For jobs, J, ($J = 1, .., M$) allocated to seller, CSP_i, ($i = 1, .., N$), this objective is defined by:

$$G(i, j) = Max. \sum_{i=1}^{N} \sum_{J=1}^{M} (E_{ij} - A_{ij}) \tag{3}$$

where N and M are respectively the total number of sellers and jobs currently in the cloud federation. E_{ij} and A_{ij} are the expected and actual completion times of job J executed by seller entity, CSP_i respectively.

5. Applicability

In this section, we present an example of a service selection problem to demonstrate a candidate application of the architecture presented in figure 3. For simplicity, two SLA terms are considered: cost of service and response time.

5.1. Hypothetical Case Study

The work of M. Jaeger et al. [26] presented a variety of workflow patterns for service composition. In practice, software-as-a-service (SaaS) cloud providers publish concrete instances of the abstract services in these workflows. Here, we consider an example of a workflow compositional model of the online shopping cart of a company which relies on dynamically composed services to meet its customers' orders. The compositional model is shown figure 4.

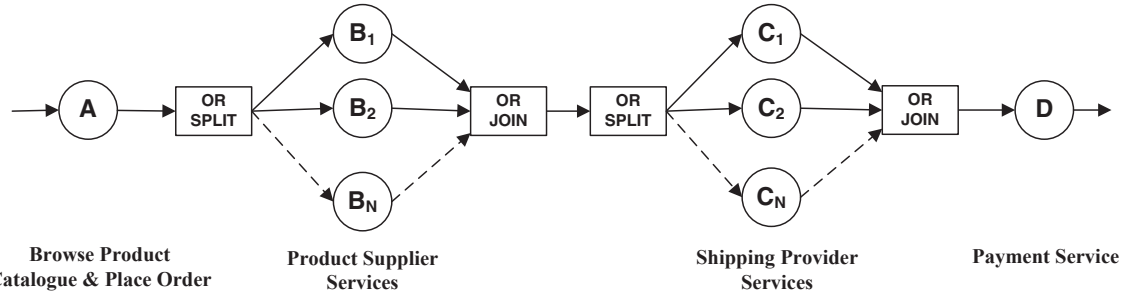

Figure 4: Online Shopping Cart Service Composition

Four services are required to ensure orders placed by customers of the company are met.

- Service A: renders the company's product catalogue in a browser and provides a means for customers to place orders.
- Service B: provides selected product(s) in the customer order at a specified cost. Suppose N product supplier services are available, possible options are B_1, B_2,...,B_N.
- Service C: offers shipping services for product(s) in the customer order within specified delivery time and at a cost. For N shipping service providers, possible options are C_1, C_2,...,C_N.
- Service D: provides payment service to collect funds from customers on behalf of the company.

As illustrated in figure 4, product supplier services (B_1, ...,B_N) are substitutable depending on whichever offers the selected product(s) at a lower cost. Similarly , supplier services (C_1, ...,C_N) are substitutable depending on whichever meets the desired delivery time at a lower cost. The SLA constraint of interest to the company is to *minimise the cost of meeting orders (i.e. product and shipping cost) without exceeding the promised delivery time*.

5.2. Cloud Federation Solution

To meet the SLA constraint specified in customer orders, the following simplifying assumptions are made:

- The product supplier and shipping services are provided by software-as-a-service (SaaS) cloud providers.
- The order of each customer, i, contains only one product, O_i.
- The SLA term of the product supplier service, SLA_B, for order, O_i, specifies the minimum price the company is willing to pay for the product i.e. $SLA_B = (P^B{}_{O_i})$
- The SLA term of the shipping service, SLA_C, for order, O_i, specifies the minimum shipment price and maximum acceptable delivery time i.e. $SLA_C = (P^C{}_{O_i}, T_{O_i})$

The setup of the cloud federation is shown in figure 5. The cloud user (i.e. online shopping cart company) interfaces with the cloud federation to provide concrete instances of the product supplier and shipping services at run-time. For each requested service, the order and associated SLA terms are submitted to the cloud federation.

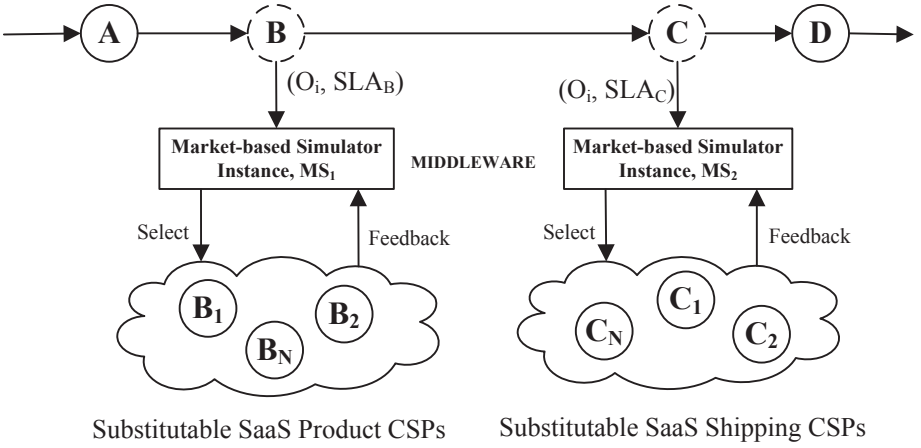

Figure 5: Coordination of Cloud Service Providers for Online Shopping Cart Service in Federated Cloud

The simulator instances in the middleware layer are able to select from two sets of CSPs, one for product supplier services and the other for shipping services. Each CSP publishes its cost and delivery time offerings via its cloud manager interface. The selected CSPs are made available for subscription in the workflow. Simulator instances continuously monitor CSPs, measure their performance against SLA terms and adapt future selection (via trading strategies and reputation ratings) to mitigate risks of violating SLAs. By leveraging on multiple CSPs from the open cloud federation environment, a robust and reliable composition that satisfies the company's SLA is achievable.

6. Conclusion and Future Work

The success of the cloud computing model depends hugely on the ability of cloud providers to keep promises made to users in their SLAs. The repeated inability of cloud providers to achieve this has given rise to the cloud federation model. Despite the current inception stage of cloud federation research, it holds promises of providing higher SLA guarantees for cloud users than the single cloud provision model. The cloud federation model achieves this by dynamically utilising the services of multiple cloud providers to meet users' requests. In this paper, we have presented the design of a cloud federation architecture aimed at coordinating the cloud federation entities to meet cloud users' SLA terms. The coordination (middleware) layer of the architecture uses a market-based DDDAS distributed simulation approach to achieve this in a scalable and robust manner.

We are currently working on the validation of the proposed approach via simulation studies. In the future, we shall report results about the architecture's ability to scale under varying workload conditions and its adaptability to various risk alert scenarios. To achieve improved measure about cloud providers' reliability, we shall extend the market mechanism to incorporate on-line learning capabilities. This is to facilitate early prediction about the likelihood of a

cloud provider's inability to meet SLA terms of jobs allocated to it in the federation. At a later stage, we shall carry out further experimental studies of the approach in a real cloud federation testbed.

References

[1] P. Mell, T. Grance, The NIST Definition of Cloud Computing, Tech. rep., NIST, Information Technology Laboratory (2009).

[2] B. Suleiman, S. Sakr, R. Jeffery, A. Liu, On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure, Journal of Internet Services and Applications.

[3] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, On the use of cloud computing for scientific workflows, in: eScience, 2008. eScience '08. IEEE Fourth International Conference on, 2008, pp. 640 –645.

[4] L. Vaquero, Educloud: Paas versus iaas cloud usage for an advanced computer science course, Education, IEEE Transactions on 54 (4) (2011) 590 –598.

[5] A. R. Hickey, The 10 biggest cloud outages of 2011 (so far), http://www.crn.com/slide-shows/cloud/231000954/the-10-biggest-cloud-outages-of-2011-so-far.htm. Last accessed: 09-March-2012.

[6] H. S. Gunawi, T. Do, J. M. Hellerstein, I. Stoica, D. Borthakur, J. Robbins, Failure as a service (faas): A cloud service for large-scale, online failure drills, Tech. Rep. UCB/EECS-2011-87, Electrical Engineering and Computer Sciences, University of California, Berkeley (July 2011).

[7] F. Faniyi, R. Bahsoon, A. Evans, R. Kazman, Evaluating security properties of architectures in unpredictable environments: A case for cloud, in: Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on, 2011, pp. 127 –136.

[8] J. Schad, J. Dittrich, J.-A. Quiané-Ruiz, Runtime measurements in the cloud: observing, analyzing, and reducing variance, Proc. VLDB Endow. 3 (2010) 460–471.

[9] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F. Galan, The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 4:1 –4:11.

[10] R. Buyya, R. Ranjan, R. Calheiros, Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, in: C.-H. Hsu, L. Yang, J. Park, S.-S. Yeo (Eds.), Algorithms and Architectures for Parallel Processing, Vol. 6081 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 13–31.

[11] A. Celesti, F. Tusa, M. Villari, A. Puliafito, How to enhance cloud architectures to enable cross-federation, in: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 337 –345.

[12] F. Faniyi, R. Bahsoon, Engineering proprioception in sla management for cloud architectures, in: Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on, 2011, pp. 336 –340.

[13] F. Darema, Grid computing and beyond: The context of dynamic data driven applications systems, Proceedings of the IEEE 93 (3) (2005) 692 –697.

[14] P. Rubach, M. Sobolewski, Autonomic sla management in federated computing environments, in: Proceedings of the 2009 International Conference on Parallel Processing Workshops, ICPPW '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 314–321.

[15] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz, G. Kecskemeti, Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures, in: Proceedings of the 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, COMPSACW '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 365–370.

[16] D. Ardagna, M. Trubian, L. Zhang, Sla based resource allocation policies in autonomic environments, J. Parallel Distrib. Comput. 67 (2007) 259–270.

[17] S. Paquette, P. T. Jaeger, S. C. Wilson, Identifying the security risks associated with governmental use of cloud computing, Government Information Quaterly 27 (3) (2010) 245–253.

[18] I. Brandic, D. Music, S. Dustdar, Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services, in: Proceedings of the 6th international conference industry session on Grids meets autonomic computing, GMAC '09, ACM, New York, NY, USA, 2009, pp. 1–8.

[19] N. Loutas, E. Kamateri, F. Bosi, K. Tarabanis, Cloud computing interoperability: The state of play, in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, 2011, pp. 752 –757.

[20] F. Darema, Dynamic data driven applications systems: New capabilities for application simulations and measurements, in: V. Sunderam, G. van Albada, P. Sloot, J. Dongarra (Eds.), Computational Science ICCS 2005, Vol. 3515 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, pp. 661–712.

[21] O. Onolaja, G. Theodoropoulos, R. Bahsoon, A data-driven framework for dynamic trust management, Procedia Computer Science 4 (2011) 1751 – 1760, proceedings of the International Conference on Computational Science, ICCS 2011.

[22] S. H. Clearwater (Ed.), Market-based control: a paradigm for distributed resource allocation, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[23] P. Lewis, P. Marrow, X. Yao, Resource allocation in decentralised computational systems: an evolutionary market-based approach, Autonomous Agents and Multi-Agent Systems 21 (2) (2010) 143–171.

[24] V. Nallur, R. Bahsoon, Design of a market-based mechanism for quality attribute tradeoff of services in the cloud, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, New York, NY, USA, 2010, pp. 367–371.

[25] L. Xiao, Y. Zhu, L. Ni, Z. Xu, Gridis: An incentive-based grid scheduling, in: Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, 2005, p. 65b.

[26] M. Jaeger, G. Rojec-Goldmann, G. Muhl, Qos aggregation for web service composition using workflow patterns, in: Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International, 2004, pp. 149 – 159.