Automatic Provisioning of Intercloud Resources driven by Nonfunctional Requirements of Applications

Jungmin Son, Diana Barreto, Rodrigo N. Calheiros, and Rajkumar Buyya

Cloud Computing and **D**istributed **S**ystems (CLOUDS) Laboratory
Department of Computing and Information Systems
The University of Melbourne, Australia
{jungmins,dianaba}@student.unimelb.edu.au, {rnc,rbuyya}@unimelb.edu.au

Summary

Cloud computing, especially the Infrastructure-as-a-Service (IaaS), allows system administrators to obtain computing and storage resources instantly and easily without up-front cost. As a result, their job to purchase new hardware and install them in server room is replaced by simply browsing the websites of cloud providers and choosing the right option. However, it is complex and challenging task for system administrators to decide the appropriate type of virtual machine (VM) offering sufficient resource for the application. Also, finding the best provider among explosively increasing cloud providers is demanding hard labor to compare vast options. In this work, we propose an automatic system that performs provisioning on public clouds based on the non-functional requirements of applications. It translates the high level non-functional requirements from administrators into VM resource parameters, selects the most adequate type of VM and the provider, and allocates actual VMs from the selected provider. The prototype shows that the system is effective in receiving non-functional requirements and provisioning resources on different cloud providers based on such requirements.

Keywords

cloud provisioning

cloud resource selection

non-functional requirements

cloud monitoring services

Inter-clouds

I. INTRODUCTION

Cloud computing has emerged as a new computing paradigm offering subscription-oriented services in place of traditional in-house computing infrastructure. Through its utility computing concept with pay-as-you-go model, enterprises can avoid upfront investment for establishing

infrastructures to provide computation power to uncertain or fluctuating demand. For system administrators in an enterprise, instead of installing new servers and network equipment, they can easily acquire computing resources from one or more cloud providers with a few clicks on a web page and pay only for the actual usage. They only need to select one or more cloud providers and services that fit for their applications from Inter-cloud environment (Buyya, Ranjan & Calheiros 2010).

Throughout the context, "cloud provider" or "provider" refers a company or an organization that provides public cloud computing services. "Customer" refers a company, an organization or a person who uses the cloud computing service offered by cloud providers to deploy their application and serve to "end-users". Furthermore, "System administrator" is a person who is in charge of managing computer resources and configuring infrastructures such as servers and network. In short, an enterprise will become a customer of the cloud provider when the system administrator in the enterprise decides to use the cloud service of the provider.

Cloud computing delivers its service to customers in three models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS provides a complete stack of application from the cloud provider, while PaaS provides software platforms that can be used by the customer's application. In both cases, the provider is in charge of managing and controlling the underlying environment of the services. In contrast, IaaS provides low-level computer resources, i.e. VMs, on which applications are deployed. Thus, the customer must configure and control computing resources of VMs, whereas only the underlying physical infrastructures are managed by the provider.

When deploying SaaS or PaaS services from multiple cloud providers, system administrators are less demanded regarding infrastructural decisions, because services are limited to specific applications or platforms that have fewer options to be chosen, and cloud providers offer automatic features for their services such as automatic scaling.

On IaaS, however, system administrators are expected to make more decisions, since VMs parameters can be selected, which incurs several challenges. First of all, there are a vast number of options of VM resource amounts from several possible providers. In most providers, the VM size is defined by the number and power of CPU cores, the amount of RAM, and the size of storage space, which comes with either a predefined amount set by the provider or flexibly configurable by the customer. As each provider has its own set of resource types with different pricing policies, there is no common rule for definition of VM types. For example, Amazon EC2[1] offers 17 predefined types of VMs depending on the sizes of resources, while Microsoft Azure[2] has 8 different VM types. Moreover, providers such as CloudSigma[3] provide more flexible configuration where a resource set is freely selectable by the customer without any fixed size. In addition to the size of VM, several factors affect the decision on how the price for VM usage is determined, such as choice of operating system, location of the data center, contract durations, etc. In summary, system administrators are in charge of deciding the best option among various resource types, pricing schemes, and cloud providers.

[1] Amazon. 2013. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/ (accessed November 25, 2013).
[2] Microsoft. 2013. Microsoft Windows Azure Infrastructure Services. http://www.windowsazure.com/en-us/solutions/infrastructure/ (accessed November 25, 2013).
[3] CloudSigma. 2013. CloudSigma Features. http://www.cloudsigma.com/#features (accessed November 25, 2013).

Secondly, estimating the right amount of resources is important in order to determine the optimal size of VMs. Although cloud computing provides elastic scaling that allows changes in the number and types of VMs after setup, determining the initial resource set is vital as it reduces the need to reconfiguration, which demands time to be completed as it requires booting new VMs with the new configuration. Meanwhile, non-functional requirements, such as the expected number of end-users, usage patterns and acceptable response time, are obtainable from accumulated statistics for existing application, or can be predicted from the application specification. Nonetheless, converting non-functional requirements to low-level VM resource requirements, such as number and computing power of CPU cores and amount of RAM, is difficult without comprehensive knowledge about the underlying infrastructure. By estimating proper resource sizes, an enterprise can avoid over-provision leading to spending funds with unnecessary resources or under-provision causing performance loss or failure of the application that results in end-user unsatisfaction and consequent loss of revenue.

Finally, VMs must be allocated from the chosen provider with the preferred resource type within the set amount of time. As customers do not have control over the cloud infrastructure, there are no guarantees that resources will be allocated to the customer unless some type of reservation is made beforehand. When no reservation is in place, resource unavailability in one provider forces system administrators to find another provider that meets all the determined requirements and that is able to fulfill the resources request. It requires extra effort and cost to the system administrator, and in worst case, it may cause the failure of service if it takes too long time to be accomplished.

In this work, we propose an architecture to address the challenges emerging from the system administrators' perspective. Using our architecture, administrators can acquire the desired number of VMs from the best provider with proper resource size that covers their non-functional requirements. This will help system administrators to migrate their applications to the cloud by setting up the required IT infrastructure without much concern of calculating amount of resources. Furthermore, enterprises can reduce the cost of cloud usage by selecting the optimal set of resources for their applications based on the supplied applications' non-functional requirements.

The rest of this chapter is organized as follows. We look into related works further in Section 2 and explain backgrounds and motivates of this work in Section 3. The architecture is described in Section 4, with details of each component and their implementation. In Section 5, we present the performance evaluation of a system prototype based on the proposed architecture. Finally, Section 6 concludes the chapter and proposes future works.

II. STATE OF THE ART

Several studies have been conducted by different groups in cloud provisioning, monitoring services and resource selection area. As our proposal integrates each of these features into one single framework, individual works are reviewed for each of these areas.

A. Cloud Provisioning

Resource provisioning in cloud computing refers to the decision about number, types, and location of resources to be deployed for a specific purpose. Definition of resources for provisioning may also include details on required processor, amount of storage, network bandwidth, and other relative resources from the cloud provider. The large number of variables

related to resources definition makes it a complex problem for being solved in an optimal way. Nevertheless, when simultaneous utilization of multiple cloud providers is sought, the problem becomes even more challenging.

Several approaches have been proposed for resource provisioning on multiple cloud providers. Grozev and Buyya reviewed and compared the architectures and brokering mechanisms of those Inter-cloud systems (Grozev, Buyya 2012). The authors proposed taxonomies for Inter-cloud architectures and presented detailed surveys of each project.

In centralized federation architectures for Inter-clouds, there exists a central component that aggregates status of cloud providers and finds available resources from participating data centers. For example, if one provider receives a request to provision resources from its client but cannot provide them, the request is redirected to another provider that can offer the desired resources.

The peer-to-peer federation architecture is similar to centralized federation approach except for the absence of central component. In this architecture, cloud providers communicate and negotiate with each other directly without a centralized server.

Independent Inter-cloud approaches enable resource provisioning from multiple clouds without direct exchange between providers as in the previous approach. This is achieved with an independent service or library that supports multiple cloud providers. For example, in the industry RightScale[4] gives a single Dashboard and APIs to manage multiple clouds. They provide a configuration framework with templates to set up the VMs easily. Also, they provide an easy management tool on multiple cloud providers, but do not perform the provider selection.

Independent approaches also include providing APIs for cloud application development and support deployment on multiple clouds, what allows developers to regard the heterogeneous clouds as a single platform with transparent access. Instead of developing an application using different APIs provided by each provider, these libraries include homogeneous controlling and provisioning functions supporting multiple providers. Apache Jclouds[5], for example, is a Java library for Java-based interaction with various providers, which provides a provider-independent API for execution of operations regarding provisioning of computing resources and storages. While these libraries are helpful to develop an application able to execute on various providers, they just provide an alternative to provider-specific APIs, and therefore they do not offer cloud provider selection or automatic resource provisioning, which is still a task of system administrators' using such libraries.

B. Cloud Monitoring Services

Cloud monitoring is an important service to check the health of each data center and compare different type of VMs in different cloud providers. IaaS cloud providers offer the computing resources in terms of VM, which is composed of the unit of CPU cores, the amount of memory and the size of disk spaces. These terms are defined by the provider itself, thus it is not straightforward to compare different types of VMs in different providers by just comparing the number of computing units they advertize. Also, periodical check is necessary to determine the reliability and the availability of the provider. These metrics can be obtained by measuring up-time percentage of the provider.

[4] RightScale. 2013. RightScale Cloud Portfolio Management. http://www.rightscale.com/products/cloud-portfolio-management.php (accessed November 25, 2013).
[5] Apache Software Foundation. 2013. Apache jclouds. http://jclouds.apache.org/ (accessed November 25, 2013).

The importance of knowing the performance of different public cloud providers has encouraged the development of monitoring services that report metrics to a better picture of real behavior of the different services.

CloudHarmony[6] reports results of benchmarks in regard to performance, network, and uptime for a wide set of public cloud providers. To collect these metrics, monitoring services are located inside and outside of the cloud provider and additionally some benchmark applications are executed on behalf of CloudHarmony.

CloudSleuth[7] provides a tool called Cloud Provider View, which displays the perceived response time and the percentage of availability of various providers in different locations. It operates by deploying a test application on each data center, and continuously monitoring its performance and availability. By analyzing the collected data from applications running in each data center, they provide the status of each data center, which can help others to compare different data centers and providers.

CloudStatus[8] collects, in real-time, observations of infrastructure metrics such as availability, response time, latency, and throughput from Amazon and Google cloud services. These metrics are aggregated by the server from sources inside and outside of the provider, and calculated to diagnose the health of the cloud. With the diagnosed result, they provide an overall status of the cloud in real-time that can affect the performance of the applications running in the cloud. Instead of monitoring specific instances of the cloud, the results cover overall availability and normalized metrics across the cloud.

C. Cloud Providers Ranking and Selection

Before starting the provisioning process, a provider and the type of resources which will satisfy all the requirements should be chosen by the administrator. The selection criteria can be various depending on the requirements. Constraints are the requirements that must be fulfilled by the cloud provider. For example, some governmental applications may be restricted to be running within their national territory due to the legislation. In such case, geographical constraints should be applied to choose the provider, so that providers who have no data center in such nation will be excluded from the choice. Preferences are the criteria to make ordering of the providers. Up-time percentage of the data center can be one of the preferences when an application needs higher reliability. Price of the service can be another preference if the system administrator concerns more about the cost, resulting in selection of the most economic provider.

In this area, a number of researches have been already conducted by several scholars. Some of them are introduced on the following paragraphs.

Li et al. developed CloudCmp (Li et al. 2010), which compares different cloud providers by using a tool to perform systematic benchmarking. It evaluates the performance of elastic computing, persistent storage and intra-cloud and wide-area networking in each provider, and compares providers using unified metrics in each service. Authors also proposed CloudProphet (Li et al. 2011), which estimates applications resources and performance in the cloud. This uses

[6] CloudHarmony. 2013. CloudHarmony. http://cloudharmony.com/ (accessed November 25, 2013).
[7] CloudSleuth. 2013. CloudSleuth – global provider preview. http://cloudsleuth.net/ (accessed November 25, 2013).
[8] Hyperic. 2013. CloudStatus. http://www.hyperic.com/products/cloud-status-monitoring/ (accessed November 25, 2013).

the trace-and-replay method that records the workload of an application from a traditional infrastructure, and measures the performance when the recorded workload is replayed in a cloud environment.

SMICloud (Garg, Versteeg & Buyya 2011) is a framework to rank cloud providers for a given application considering the Service Measure Indexes (SMI): accountability, agility, assurance of services, cost, performance, security and privacy, and usability. It operates by assigning different Key Performance Indicators (KPI) to evaluate these indexes in different cloud providers.

Zhang et al. proposed a declarative recommender system to select a cloud provider (Zhang et al. 2012). The system receives as input requirement parameters from system administrators and determines the best provider that satisfies the requirements. However, the types of input parameters are resource sizes that should be transformed from non-functional high level requirements.

Recently, Rak et al. presented a cost/performance evaluation tool (Rak, Cuomo & Villano 2013) on top of the mOSAIC [9] platform. Evaluation is performed by simulating and estimating resources, cost and response time of an application. The authors propose the use of non-functional requirements to create a system that suggests the best option among a set of cloud providers.

III. RESOURCE PROVISIONING DRIVEN BY NON FUNCTIONAL REQUIREMENTS

Resource provisioning inside of an organization is usually performed by system administrators. They are in charge of the IT infrastructure and make decisions about where to deploy the applications. The system administrators also fix problems related to failures on the hardware and software that support this infrastructure. Although the more to the cloud frees system administrators from managing underlying IT infrastructure and the resulting hardware and software issues, it also bring new challenges.

Before deploying an application in the cloud, system administrators need to consider how much resource this application will consume in terms of resource capacity as defined by different cloud providers. It is a challenging task because this estimation may differ from the one for an in house IT infrastructure, and from one cloud provider to the other. Additionally, the constraints for cloud selection have to be considered as the selected provider should fulfill every requirement.

After determining resources and other requirements that application needs to execute satisfactorily in the cloud, the next task to be performed by system administrators is to decide which cloud providers can supply the estimated resources and which of them are more appropriated to host the application. This selection of candidate cloud providers is a laborious task as there are a large number of available providers, each of which offering varieties of services, which cannot be directly compared with the services provided for others.

More than 65 public clouds providers are registered by the monitoring service CloudHarmony. Additionally, each of them offers diverse range of services. For example, GoGrid[10] offers just one type of machine x-Large, that is configured with 8 cores of CPU and 8GB of memory, while Amazon offer configurations m1.xlarge, m2.xlarge, m3.xlarge, and c1.xlarge, all of them with

[9] mOSAIC. 2013. mOSAIC Cloud. http://www.mosaic-cloud.eu/ (accessed November 25, 2013).
[10] GoGrid. 2013. GoGrid. http://www.gogrid.com/ (accessed November 25, 2013).

different amount of resources. Moreover, other clouds providers, such as CloudSigma, do not have default configuration but allow flexible configuration of resources.

After selecting the candidate providers that can supply the services to deploy the application, the next concern of system administrators is to obtain the required resources from cloud providers. This is a challenging task because APIs and interfaces to communicate with cloud providers are diverse and non-standardized. Furthermore, latency in communication, provider's outage, and eventually lack of resources can impede the administrator's demand to be fulfilled. In this case, another provider would need to be contacted for obtaining resources.

As was discussed previously, these activities of estimating resources, selecting cloud providers, and allocating resources shows that the work performed by system administrators is still complex. Therefore, solutions should be developed in order of support system administrators to reduce this complexity and to motivate organizations to move their applications to the cloud.

A. Non-functional requirements

Currently, when system administrators want to acquire resources from IaaS cloud providers, they need to know the details of the resources they need. For example, consider a situation where the local infrastructure has extra resources to execute applications successfully. In this environment, administrators may have knowledge about the expected behavior and performance of their applications and this knowledge can be represented using non-functional requirements.

Non-functional requirements, according to the definition provided by (Glinz 2007), are attributes and constraints of an application created to achieve some level of quality and performance. Therefore they are not related with functions that the system should do, but with properties how the systems should be, including availability, reliability, portability, cost, efficiency, usability, and testability. For example in a shopping application, non-functional requirements can include the number of on-line transactions it can support, however shipment tracking functionality is not included in the non-functional requirements.

One non-functional requirement can be described with the number of features associated with it. For example, to describe the cost of the local IT infrastructure, the related features are the maximum price to spend in the new hardware acquisition and the price for maintenance of the infrastructure. In the specific context of clouds, the related feature could be the maximum price for using the resources during a time period.

Some of non-functional requirements which can be used to evaluate cloud providers are described below;

- **Portability** is related with the easiness of an application to be executed in various platforms. It includes a type of operating system or image running on VMs.
- **Reliability** includes Mean Time To Repair (MMTR) and Mean Time To Fail (MMTF) of cloud providers which affect the probability of system failure. The type of environment, either on development, on test or in production, will decide the level of reliability. It also

relates whether an application is acceptable to use Spot Price model[11] of Amazon, since the Spot Model sacrifices the reliability while low cost could be achieved.

- **Availability** corresponds to the system ability to respond user requests, including uptime percentage of the cloud provider and locations of end-users. When multiple machines are required to balance the overload, load balancing is crucial feature for achieving high availability.
- **Efficiency** is a requirement related with the application performance such as expected throughput and response time of an application. Also, it is important to know the type and the amount of workload that applications should support, since optimal scheduling or provisioning techniques that maximize the performance can be chosen based on that information.
- **Cost** is related with how much the customer is willing to pay for a service that satisfies all the other non-functional requirements.

Associated features to each non-functional requirement are presented in the Table 1. These non-functional requirements are considered to design the architecture for resource provisioning described in the following section.

Table 1 Initial non-functional requirements to consider in the system.

Non-functional requirement	Relevant Features
Portability	VM's operating system or image 64 or 32 bits architecture of the H/W and O/S
Reliability	Mean Time To Repair (MMTF) Mean Time To Fail (MMTR) Requires application backup Type of environment(development, test, or production) Allows spot price model
Availability	Uptime percentage End-user locations Requires load balancing for high availability
Efficiency	Expected throughput Response time Type of workload, Amount of workload
Cost	Maximum price to pay for a window of time

IV. SYSTEM ARCHITECTURE

The architecture is composed of three independent modules: High Level NFR (Non-Functional Requirements) Translator, Cloud Service Selector and Resource Allocator. Figure 1 shows the general view of the system. Each of its components is detailed next.

[11] Using this model, users can bid for resources that will be delivered only if the bid exceeds the market price of the service, but once the bid becomes less than the market price, the computing service can be removed without any notice.

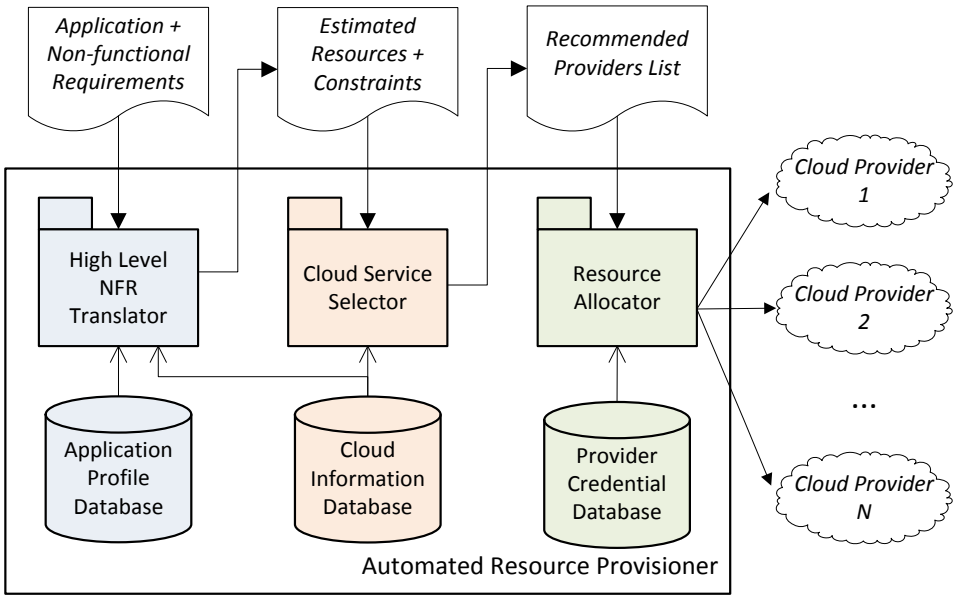

Figure 1 System architecture.

The High Level NFR Translator translates the non-functional requirements provided by the system administrator into the technical specification for a cloud infrastructure. It receives non-functional requirements, such as efficiency, availability, and reliability of a specific application and estimates the amount of resources, such as number of CPU cores, memory, and storage requirements. The output also includes specific constraints that cloud providers should fulfill, such as locations or contract periods, which is evaluated from the given non-functional requirements. When it calculates the estimated resources, it interacts with the Application Profile Database to store and retrieve the profile for the application. Furthermore, this module also interacts with the Cloud Information Database to generate the selectable input options, such as available data center locations and contract periods.

The Cloud Service Selector is responsible for recommending the cloud providers that are more adequate to host an application. For this purpose, it receives the estimated size of resources and other parameters such as the constraints and the prioritization of the non-functional requirements, and then builds a list of suggested cloud configurations, including the recommended provider. It interacts with the Cloud Information Database, which contains all information about various resource types in each provider and their pricing information, to apply the constraints and calculate the total cost incurred by the recommended configuration.

The Resource Allocator searches for available resources based on the recommended providers and acquire resources directly from cloud providers. For this task, it uses the output list given by the Cloud Service Selector and tries to get VMs from the most suitable provider. If the requested resources are not available from the top-listed provider, it tries to allocate the resource from the second best provider. The process continues until either all resources are allocated or no more providers are left in the list. It interacts with the Provider Credential Database in order to obtain login credential information for each provider.

As each component is designed to work independently, one component can be substituted by another program or module with better performance. For example, the Declarative Recommender System (Zhang et al. 2012) can be used to produce the recommended providers

list that is fed into Resource Allocator. Since their system includes blob storage and network usage costs, it might produce better results for those services, while our Cloud Service Selector focuses more on computing services. In such case, the Resource Allocator could read the output of the Declarative Recommender System and try to allocate resources.

A. System Components

1) High Level NFR Translator

The High Level NFR Translator evaluates the non-functional requirements and translates them into the technical parameters for cloud providers. The architecture proposed for the Cloud Resources Estimator follows a client-server model of three tiers composed of database, business, and presentation tier.

The presentation tier corresponds to the graphic interface in order to access the application functionality. It allows system administrators to select requirements, change their orders and enter the detailed parameters of each requirement.

The business tier, containing the main logic, retrieves data from the database, creates a workflow to evaluate non-functional requirements and performs estimating the amount of resources and determining the constraints. Figure2 presents the class diagram of the business tier.

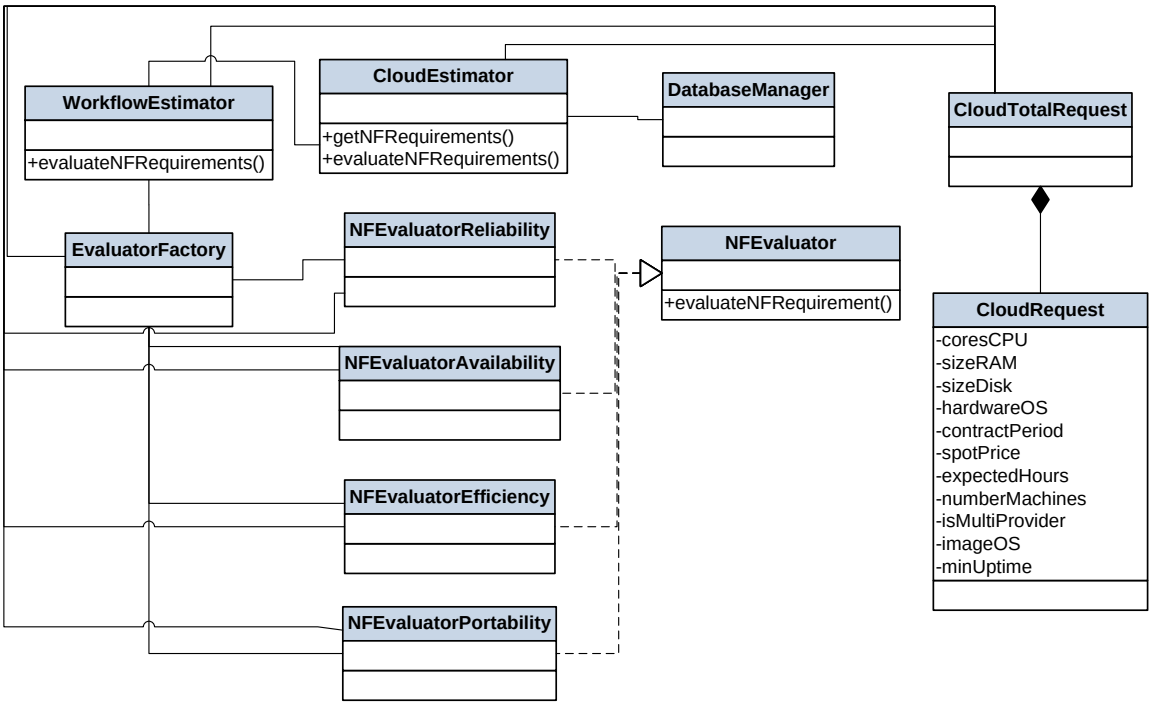

Figure 2 Class diagram of High Level NFR Translator.

When the CloudEstimator receives the application and non-functional requirements from the presentation tier, it begins the non-functional requirements evaluation process using the WorkflowEvaluator and builds up the CloudTotalRequest object that holds information of required resources. The order of evaluation for each requirement is determined according to its priority given by system administrators. Also, for each non-functional requirement, the associated evaluator class is used to update CloudTotalRequest. For example,

NFEvaluatorAvailability is used to evaluate availability requirement and change value in CloudTotalRequest that adds a constraint to meet the availability requirement.

Among various evaluators, the efficiency of the evaluator is a key to estimate the size of resources. In order to evaluate the efficiency and estimate resources, we propose the use of an application profile technique that runs the application components with a given workload. This technique has been widely adopted in several literatures (Li et al. 2010, Li et al. 2011, Shimizu et al. 2009). The results of the execution, containing information about resources consumed and performance achieved, are stored in the application profile database. When the component can be profiled in different infrastructures, better estimations can be achieved (Shimizu et al. 2009).

Once the profile is obtained, the information is fed into an estimator model. The new data obtained using the model should also be stored in the database and possibly updated with the values obtained after the application is in production in the cloud infrastructure. Also, the data can be used to update the model itself in order to increase its accuracy. After profiling and modeling the application and estimating the required resources, the output is generated by updating the existent CloudTotalRequest with the result of the estimation. Figure 3 summarizes the process.

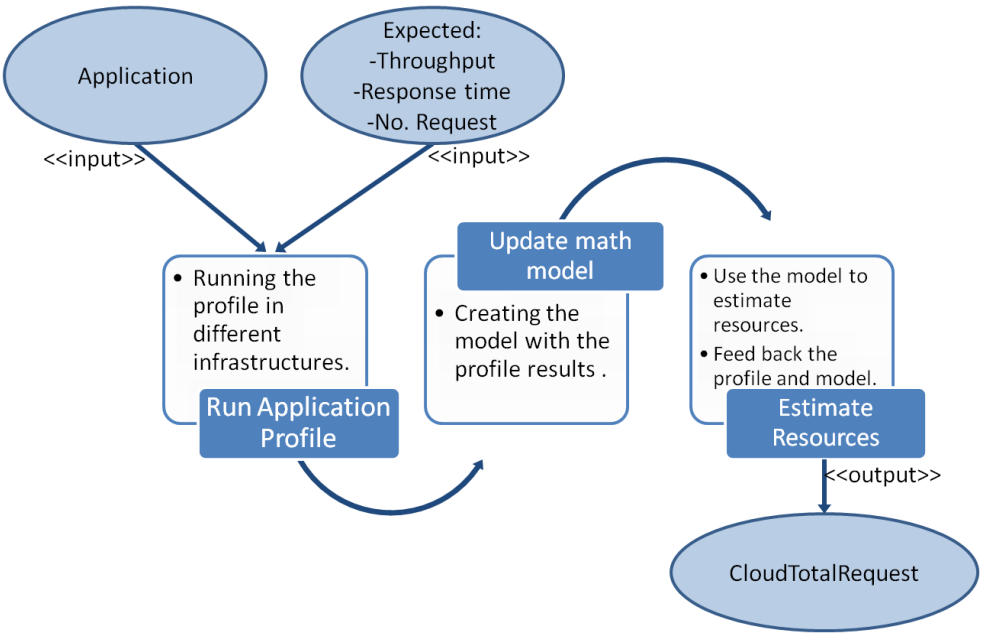

Figure 3 Steps executed by the efficiency evaluator in High Level NFR Translator.

In the database tier, details of the components that compose the applications are stored along with their profiles when executing on different infrastructures. An application is constituted of several components, and the throughput of the entire application can be predicted based on the performance of individual components and the cost of communication among them, as discussed by (Stewart, Shen 2005). This approach has the advantage of allowing the estimation of the application performance with components deployed in different infrastructures and possibly in different cloud providers. Also, it retrieves entries from the Cloud Information Database to show a list of requirement options for input parameters when the initial view is available.

2) Cloud Service Selector

The Cloud Service Selector lists the recommended providers and their services by analyzing the estimated resources and constraints. The components are designed following the Model-View-Controller model, as it can easily separate each component by reducing dependencies to other components. Figure 4 shows the architecture of the Cloud Service Selector.

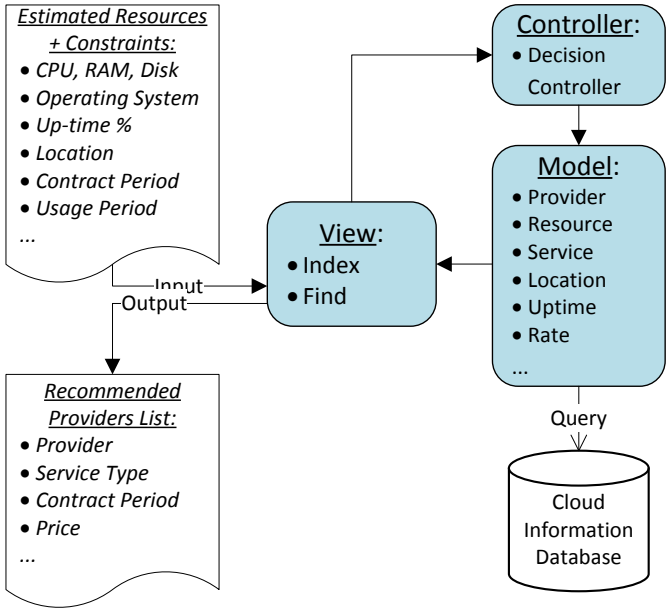

Figure 4 Component architecture of Cloud Service Selector.

The component receives the estimated resources and constraints as input from the High Level NFR Translator through its View subsystem. Once the information is provided to the Controller, it retrieves records satisfying the requirements from the model through a query to Cloud Information Database. The database stores information about providers and their services including resource configurations, geographical locations, operating systems, uptime percentages, contract periods, prices, and other parameters to evaluate the constraints and the cost.

The requirements are used as constraints for the model, e.g. the retrieved records should have more resources than the requirements, matched location and operating system, and shorter contract period. When three months contract is specified, for example, a service with one month contract can be retrieved but a service with six months contract cannot.

Once it gets all candidate providers and their service types, the model calculates the expected price based on the usage period. For some providers, such as GoGrid, only the contract period affects the total price. However, in other cases, such as Amazon's Reserved instances, the actual running time of the instance influences to charges, in addition to the upfront fee for the contract establishment. Hence, the actual usage period should be included to calculate the correct price. Once the contract and the actual usage period are input by the system administrator, total cost can be simply calculated using pricing information stored in the Cloud Information Database.

After applying constraints and calculating actual price for each provider, a list of providers is generated and prioritized according to the order. If the system administrator selected price as the most concerning aspect, the cheapest provider will be the first entry of the output list. In other

case, the provider with higher up-time percentage will be the first if the administrator weights more on availability rather than the cost.

3) Resource Allocator

The Resource Allocator interacts with providers and requests allocation of the resources defined in the list of specifications. The component requires a Cloud Configuration list as its input, which is the output from the Selector. Each row in the list contains information about the provider, resource set, location, image information, and the number of machines to be allocated.

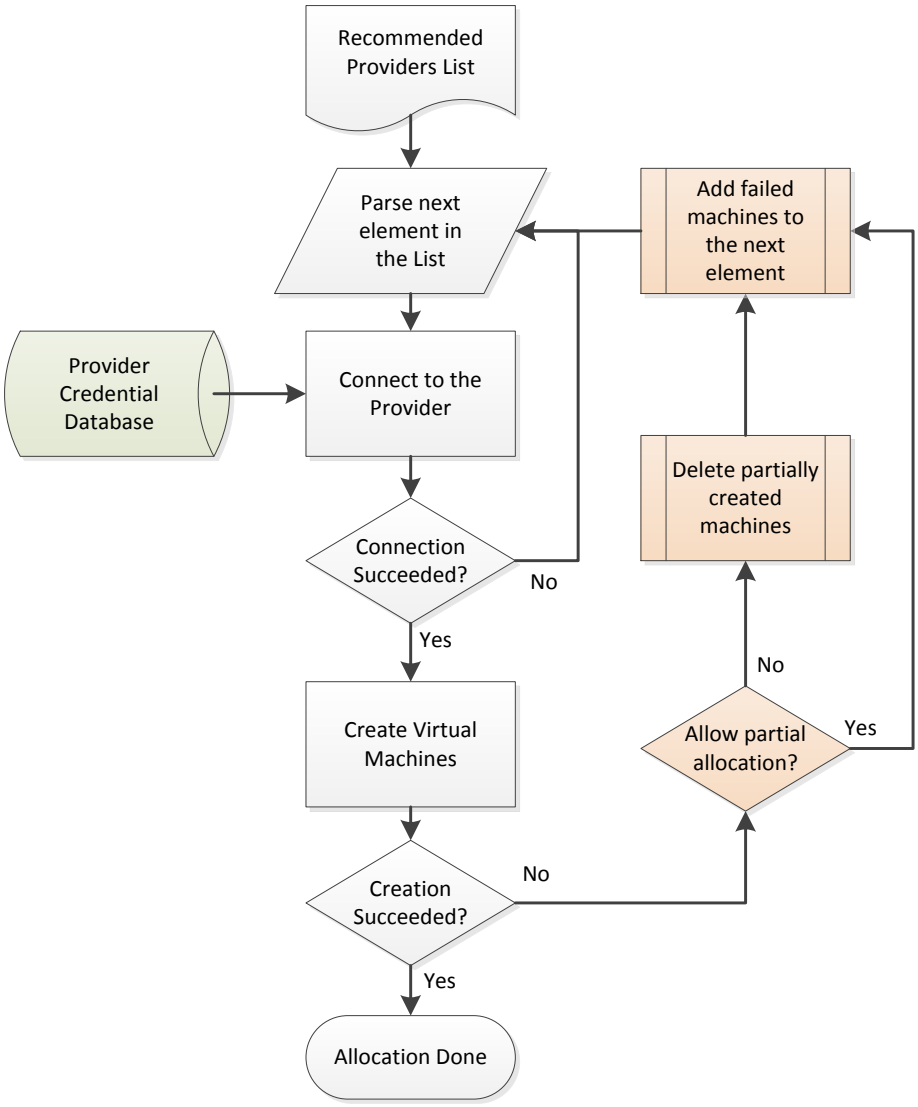

Figure 5 Flowchart of Resource Allocator.

Figure 5 describes the flow of the Resource Allocator. It starts parsing the input configuration list and building a collection of Cloud Configuration objects storing the parsed information. Once all elements are parsed, it tries to create VMs to the provider using that object. When the system connects to the provider, it authenticates the identity using credentials stored in Provider Credential Database. If the connection is successfully established, it requests the number of VMs with the specific information of resource, location, and image to the cloud provider with a

timeout. Once either the allocation succeeds or the timeout exceeds, the Allocator analyses the results and proceeds to the next option.

Additionally, partial allocation on different providers is supported by the Resource Allocator. Depending on the selection of whether the allocation needs to occur on a single provider or can be split among multiple providers, there is an additional process to be carried out. If the partial allocation to several providers is allowed, the remaining number of VMs is added to the next option in the list. If the input specifies that only one provider can be used, it does not proceed with the allocation and repeats the process for the next entry.

B. Implementation

In order to illustrate and evaluate the functionality of the proposed architecture, we implemented a prototype for High Level NFR Translator, Cloud Service Selector, and Resource Allocator. Each component is developed separately in order to ensure the independency of the component.

The High Level NFR Translator is implemented using JavaEE, JavaEE GlassFish and MySQL, and is developed on three tiers basis including Application Profile Database. The component's prototype is configured with four non-functional requirements: portability, availability, reliability, and efficiency. The requirements are selected with the sub-features described in Table 1, which are necessary to create a resource estimation request.

The Cloud Service Selector is implemented with conventional Model-View-Controller architecture using Ruby on Rails. The Cloud Information Database is also included to store different resource types, pricing policies, and data center locations of each provider. By enabling the input of resource requirements, it displays the prioritized providers' list as an output.

The Resource Allocator is implemented with Java on console interface. In order to cover various providers in a single program, we use Jclouds, multi-cloud supporting library written in Java. Jclouds allows developers to use homogeneous APIs to connect to different clouds, thus, developers do not need to use different APIs for different providers. Instead, Jclouds provides a single interface to connect, create, and destroy VMs. Also in the prototype, the Provider Credential Database is implemented as a list of provider name, user name, and password and parsed by the Resource Allocator.

V. PERFORMANCE EVALUATION

Evaluation is performed independently for each module of the system, as they are designed and developed independently. We evaluate the High Level NFR Translator in order to have its performance measure. The Cloud Service Selector is assessed in order to evaluate the functionality of selecting the cloud provider candidates that meet requirements, and to evaluate the benefits of prioritizing the candidate providers according to non-functional requirements. Finally, the performance of the Resource Allocator is also evaluated.

A. High Level NFR Translator

The goal of this experiment is to evaluate if the proposed architecture is able to scale dynamically when the number of requirements and requirement's features increase.

The platform used to test the application is the Australia Research Cloud NeCTAR[12]. This experiment utilizes an instance of type m1.small. Instances of this type have 4GB of RAM, 1 CPU core and 10 GB of Disk. The Operating System used in the VM is Ubuntu 13.04.

Requests with varying number of requirements and number of features per requirement are generated. Number of non-functional requirements varies from 10 to 300, in steps at 10. The same has been performed for the number of features. For each increment of requirements and their features, we measure the time taken to evaluate non-functional requirements and to transform them into technical parameters. The result shown in Figure 6 demonstrates that the prototype scales satisfactorily when the number of requirements increases. It takes less than four seconds with 300 requirements and 300 features input, which is an acceptable time to process such amount of requirements.

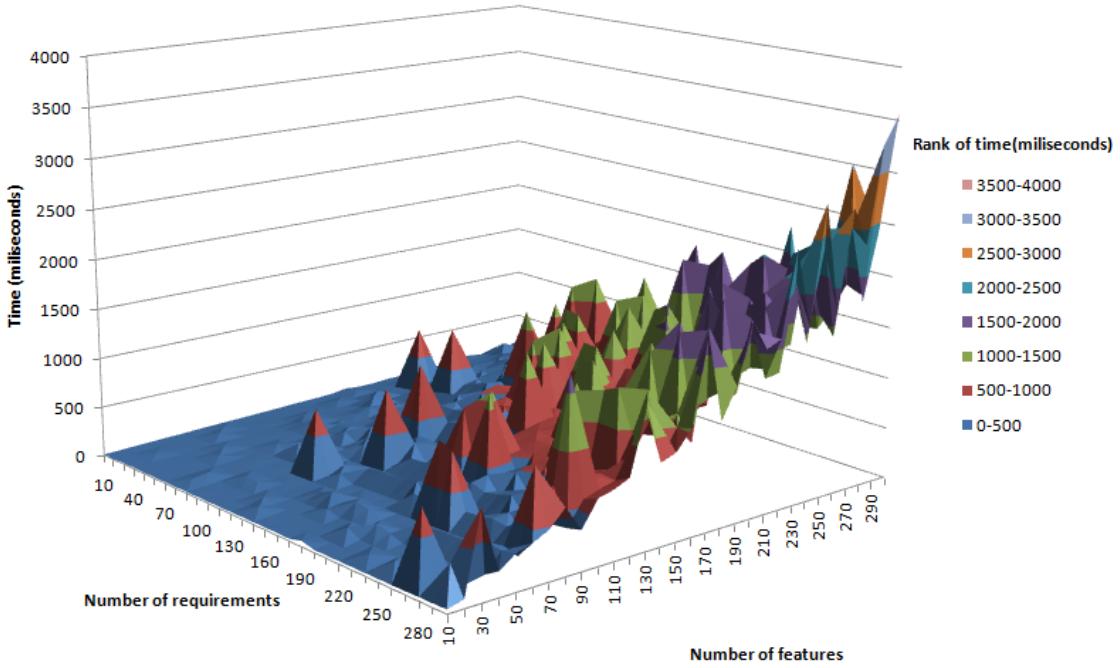

Figure 6 High Level NFR Translator Performance.

B. Cloud Service Selector

The Cloud Service Selector should be able to select a valid service and calculate the precise price based on the given requirements. It also should prioritize the provider candidates according to the priority of the non-functional requirements as defined by the system administrator.

For the evaluation of the Selector, the contents in Cloud Information Database are crucial to the result, as the quality of the result depends on the database information. For the purpose of this this experiment, we create a static database with synthetic data that allows us to test the functionality. Table 2 shows a part of the database elements used for the experiment, with various instance types and pricing schemes of different providers. In order to validate the system,

[12] NeCTAR. 2013. NeCTAR Research Cloud. http://www.nectar.org.au/research-cloud (accessed November 25, 2013).

we input the parameters of the user case described in Table 3 into the Cloud Service Selector, and obtain the result of the ordered list.

Table 2 Database information used for Cloud Service Selector experiment.

Provider	Resource type	CPU (cores)	RAM (MB)	Disk (GB)	OS	Datacenter Location	Contract period	Resp. Time (msec)
Provider1	m1.small	1	1,700	160	Windows	Oceania	None	10.39
Provider1	m1.small	1	1,700	160	Windows	Oceania	12 month	10.39
…								
Provider2	small	1	1,000	50	Linux	Europe	1 month	25.40
…								
Provider3	custom	(Max) 20	(Max) 32,000	(Max) 1,024	Linux	Europe	1 month	31.60
Provider3	custom	(Max) 20	(Max) 32,000	(Max) 1,024	Linux	N. America	1 month	18.02
Provider3	custom	(Max) 20	(Max) 32,000	(Max) 1,024	Linux	N. America	3 month	18.02
…								

Table 3 User case for Cloud Service Selector validation.

Field	Value
Resource	CPU= 1 RAM= 512 MB Disk= 20 GB
Operating System	Any
Location	Any
Contract Period	3 months
Allow Unreliable Services?	No
Usage Period	3 months
Order By	Cost

Table 3 Results from Cloud Service Selector.

Rank	Provider	Resource type	OS	Datacenter Location	Contract period	Resp. Time	Price
1	Provider3	custom	Linux	N. America	3 month	18.02	81.11
2	Provider3	custom	Linux	N. America	1 month	18.02	83.57
3	Provider3	custom	Linux	Europe	1 month	31.60	91.80
4	Provider2	small	Linux	Europe	1 month	25.40	108.75
…							
7	Provider1	m1.small	Linux	Oceania	None	10.39	172.80
…							

Results (Table 4) show that the system suggests only services whose resources are larger than the requirements, with any operating systems, in any locations and for any contract periods less than three months. For example, the service from Provider1 with 12 months contract is excluded, and the lowest cost service from Provider3 is placed in the first rank. In addition, we evaluate the benefits of prioritizing by another non-functional requirement, response time. When an

enterprise can make more profit with faster response time by providing better user-experience, they will be willing to pay more for it. The system works precisely as it prioritizes the provider with fastest response time to the top of the list although its price is higher than others. As a result, the service of the Provider1, previously ranked number 7 in order of price, become the most suitable service with the fastest response time.

C. Resource Allocator

The Resource Allocator is tested using the list of cloud configurations that is obtained from the Selector. Although the system supports every provider supported by Jclouds, we evaluate it with 't1.micro' instances, cost-free VMs provided by Amazon EC2 service.

We build a cloud configurations list with two candidate configurations with Amazon-EC2, two t1.micro instances, Ubuntu 12.04 and Australia (ap-southeast-2) location, and another with USA East (us-east-1) location. The Allocator firstly tries to acquire VMs described on the first element, the Australia one. If the first request fails due to lack of resources in Australian data center, the Allocator attempts to the next candidate. When the system was executed, it succeeded to allocate two machines in Sydney, and we can find the successfully created machines on the control panel of Amazon's website.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

Cloud computing enables a major paradigm shift in the way that computing resources are acquired. Without any hardware acquisition, system administrators are able to obtain computing power to deploy their services within minutes using cloud computing. It also gives a possibility of paying only for consumed resources with no minimum contract and upfront cost.

However, deploying applications in the cloud is still a complex task for system administrators. They are expected to estimate resources required by their applications, which may be difficult because they frequently do not exactly know how much resources are actually necessary. Furthermore, administrators have to select the best cloud service amongst various providers and different types of services, and acquire them for applications to deliver expected performance.

In this chapter, we propose an architecture supporting system administrators in the arduous task of deploying applications on the clouds in three ways. Firstly, it translates non-functional requirements from administrators into actual Cloud resource parameters. Secondly, it selects the most convenient provider among different candidates that satisfies every requirement. Finally, the actual VMs are allocated automatically from the selected provider.

The proposed architecture is also verified through evaluation and validation. Each component is validated in performance and scalability for various sets of non-functional requirements. Also, we show that the number of VMs with adequate resources is actually allocated from the selected cloud provider at the end of the process.

For future directions, the proposed architecture can be applied to measure the performance of various techniques in each module. Several resource estimation techniques can be applied to the estimator in the architecture, which can lead to find the most accurate methodology to estimate resources in the clouds. Similarly, different approaches to select the best provider can be used for the selector module. In addition, Cloud Information Database can be improved by applying dynamic updates, thus it will keep the consistency between the system and the providers, and will provide more accurate selection by including real-time metrics measured by monitoring

services. Finally, dynamic resource provisioning can be applied to the system, which can dynamically perform the whole provisioning process depending on the real-time workload measured from the running application.

References

Buyya, R., Ranjan, R. & Calheiros, R.N. 2010, "InterCloud: utility-oriented federation of cloud computing environments for scaling of application services", *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing*, Springer-Verlag, Berlin, Heidelberg, pp. 13.

Garg, S.K., Versteeg, S. & Buyya, R. 2011, "SMICloud: A Framework for Comparing and Ranking Cloud Services", *Proceedings of the 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'11)*, pp. 210-218.

Glinz, M. 2007, "On Non-Functional Requirements", *Proceedings of 15th IEEE International Requirements Engineering Conference (RE '07)*, pp. 21-26.

Grozev, N. & Buyya, R. 2014, "Inter-Cloud architectures and application brokering: taxonomy and survey", *Software: Practice and Experience*, vol. 44, no. 3, pp. 369–390

Li, A., Yang, X., Kandula, S. & Zhang, M. 2010, "CloudCmp: comparing public cloud providers", *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10)*, ACM, New York, NY, USA, pp. 1-14.

Li, A., Zong, X., Kandula, S., Yang, X. & Zhang, M. 2011, "CloudProphet: towards application performance prediction in cloud", *Proceedings of the ACM SIGCOMM 2011 conference (SIGCOMM '11)*, ACM, New York, NY, USA, pp. 426.

Rak, M., Cuomo, A. & Villano, U. 2013, "Cost/Performance Evaluation for Cloud Applications Using Simulation", *Proceedings of 2013 IEEE 22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2013)*, pp. 152-157.

Shimizu, S., Rangaswami, R., Duran-Limon, H.A. & Corona-Perez, M. 2009, "Platform-independent modeling and prediction of application resource usage characteristics", *Journal of Systems and Software*, vol. 82, no. 12, pp. 2117-2127.

Stewart, C. & Shen, K. 2005, "Performance Modeling and System Management for Multi-component Online Services", *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation*, Berkeley, CA, USA, pp. 71.

Zhang, M., Ranjan, R., Nepal, S., Menzel, M. & Haller, A. 2012, "A Declarative Recommender System for Cloud Infrastructure Services Selection", *Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2012)*, eds. K. Vanmechelen, J. Altmann & O. Rana, Springer-Verlag, Berlin, Heidelberg, pp. 102-113.