Part I

Value-added services in cloud brokering

Chapter 1

Cloud performance and placement in cloud brokering

Introduction

As the number of cloud computing services increases, so does the interest of consumers to be able to compare these services in order to choose those best adapted to their needs. This chapter focuses on the performance issues related to cloud provider evaluation and on the role of cloud brokers in the automatic optimization of resource allocation across multiple cloud providers. This chapter is structured as follows. Section "cloud performance evaluation". presents a survey of the current studies related to cloud performance evaluation. The motivation for and challenges behind the evaluation of cloud provider performance is also detailed in this section. Section "Placement in cloud brokering" describes the current methods used to allocate resources in cloud brokering. The studies are classified into two categories: placement based on non-functional requirements and application-aware placement.

Cloud performance evaluation

Motivations and challenges

The current cloud computing landscape hinders a straightforward comparison of cloud provider service offerings. In the case of computing resources, this is mainly due to the heterogeneity of VM configurations and prices. On one hand, traditional cloud providers such as Amazon, Rackspace and WindowsAzure sell fixed-size VMs. These VM configurations often vary from one cloud provider to another, making it impossible to make a direct comparison. On the other hand, new cloud providers in an effort to attract consumers, look to differentiate their services through technology by allowing consumers to freely configure the size of the computing resources to be purchased.

VM performance evaluation adds another layer of complexity to the comparison of cloud providers. Firstly, consumers have little knowledge and control over the infrastructure hosting their applications. Due to the virtualization of hardware used in cloud computing, cloud providers may use resource sharing practices (*e.g.* processor sharing, memory overcommit, throttling or under-provisioned network [4]) that degrade the performance of a cloud application. Secondly, cloud provider's data centers are equipped with hundreds of thousands of servers with different qualities of hardware and software. The evaluation of performance cross all the data centers of multiple cloud providers, implies a trade-off between thoroughness, time and cost of the evaluation [5]. Thirdly, cloud providers may continually upgrade or extend their hardware and software infrastructures, and new commercial services and technologies may gradually enter the market [6]. Therefore, performance evaluations become quickly out of date and the tools for performance measurement must be continuously re-designed. Finally, there are no cloud-specific benchmarks to evaluate all VM features [7]. However, traditional benchmarks can partially satisfy the requirements for cloud performance evaluation.

Cloud performance evaluation would be beneficial for both consumers and cloud providers [5]. Consumers testing their applications across multiple cloud providers can choose the cloud provider that represents the best performance-cost trade-off. Also, performance evaluations can serve as a recommendation for the performance of a particular system [4] or can give technical arguments to consumers to put pressure on cloud providers to use better practices [7]. A provider may identify its market positioning in order to improve its services or to adjust its prices [5].

Studies related to cloud providers performance evaluation

An exhaustive study about the academic approach to commercial cloud services evaluation has been carried out by the Australian National University [6]. A Systematic Literature Review (SLR) was the methodology employed to collect the relevant data to investigate the evaluation of cloud services. As a result, 82 relevant cloud service evaluation studies were identified. The key findings of this study represent a state-of-practice when evaluating cloud services and are as follows:

- 50% of the relevant studies investigated applying cloud computing to scientific issues, while only 16% of the studies focused on the evaluation of business applications in the cloud.
- 21 cloud services over 9 cloud providers were identified. 70% of the relevant studies evaluated cloud services provided by Amazon Web Services (AWS).
- Three main aspects and their properties for cloud services evaluation have been investigated (performance, economics and security), performance being the most studied aspect (78 studies).
- There is no consensus regarding the definition and the usage context of metrics. Some metrics with the same name were used for different purposes, some metrics with different names were essentially the same. The study identified more than 500 metrics including duplications.
- There is a lack of effective metrics vis-à-vis elasticity and security aspects in cloud computing. Therefore, it is hard to quantify these aspects.
- There is not a single or a small set of benchmarks that provides a holistic evaluation of cloud services. The SLR identified around 90 different benchmarks in the selected studies of cloud services evaluation. These benchmarks can be grouped in three

main categories: application, synthetic and micro-benchmarks, as explained below.

- 25 basic setup scenarios for constructing complete cloud service evaluation experiments have been identified and classified.
- The cloud service evaluation is getting more and more attention from the research community. The number of relevant studies was 17 times larger in 2011 (34 studies) than in 2007 (2 studies).

Cloud performance evaluation is done by running application benchmarks, synthetic benchmarks or micro-benchmarks in single or multiple cloud providers. Application benchmarks correspond to real-world software that provides an overall view of the performance of a specific application. Synthetic benchmarks simulate application behavior by imposing a workload on the system. Similarly, micro-benchmarks impose a workload with the aim of measuring hardware-specific VM features. Since there are no cloud-specific benchmarks, cloud performance has been measured through widely used benchmarks such as TPC-W (a transactional web e-Commerce benchmark) [8], HPCC (a software suite consisting of 7 basic benchmarks) [4, 9, 10], NPB (set of parallel benchmarks to evaluate the performance of parallel supercomputers) [4, 11] or common measurement tools such as *ping* or *iperf* [12,13]. Also, specific benchmarks have been developed to measure cloud performance of CPU, memory, disk and network [14, 15] further the VM provisioning or deprovisioning time [10, 12, 16]. Details about the studies related to cloud providers' performance evaluation are presented in Table Appendix A.

Recent studies tend to clarify confusing concepts, inaccurate terms, as well as to unify the metrics used by previous cloud performance evaluation studies. Li *et al.* propose a taxonomy of performance for evaluating commercial cloud services [8] and potential approaches to bring about a holistic impression of cloud services performance through a single figure of merit [17].

Cloud Virtual Machine (VM) characterization

According to the studies of cloud provider performance evaluation presented in the previous section, a cloud VM can be represented by a set of criteria and a set of capacities (Figure 1.1). The criteria set is composed of the

VM physical properties (*i.e.* communication, computation, memory and storage) and of cloud service related features (*i.e.* availability, reliability, scalability and variability). The set of capacities corresponds to the metrics used to describe the performance of the criteria. Both criteria and capacities are described below.

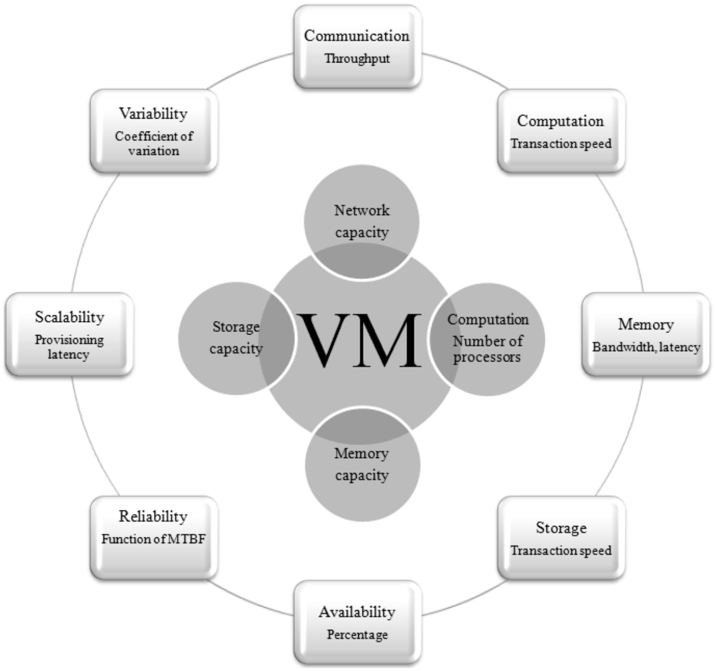

Figure 1.1: VM characterization. The inner circle represents the physical properties of a VM. The outer circle presents the performance criteria and examples of their capacities.

Criteria

- *Communication* is defined as the property of transferring data between two entities through a network. Three types of communications can be distinguished: intra- and inter-datacenter network, and wide-area network [5]. Intra-datacenter network refers to the communication of two VMs belonging to the same datacenter, while inter-datacenter network corresponds to the communication between two VMs located in different datacenters but belonging to the same cloud provider. Wide-area network refers to the communication between a VM located in a datacenter and an external host on the Internet.

- *Computation* refers to the physical property of processing data. In the case of a VM, computation corresponds to the evaluation of the virtual CPU.
- *Memory* corresponds to the physical property of storing data on a temporary basis. Both RAM memory and cache are considered in this category.
- *Storage* refers to the physical property of storing data on a permanent basis, until the data is removed or the service is suspended by the end-user.
- *Availability* is defined as the percentage of time an end-user can access the cloud service (Equation 1.1). For a given interval of time, it is calculated as a ratio of the uptime of the cloud service to the total time of the interval, usually on a yearly basis.

$$\text{Availability} = \frac{\text{total uptime}}{\text{total time of the interval}} \tag{1.1}$$

- *Reliability:* In the literature, the definition of reliability varies depending on different contexts or perspectives. Here, reliability refers to the ability of a cloud service to perform its function for a specified period of time (Equation 1.2). It is defined based on the previous failures experienced by users and the promised Mean Time Between Failures (MTBF) by the cloud provider [18].

$$\text{Reliability} = \left(1 - \frac{\text{number of users experiencing a failure}}{\text{number of users}}\right) \times \text{MTBF} \tag{1.2}$$

Thus, if a cloud provider promises a MTBF of 8760 hours (one failure per year) and 20% of his clients experienced a failure in an interval less than promised by the cloud provider. The reliability performed by the cloud provider, according to the definition presented in this study is 7008 hours (or 9 months and 22 days).

- *Scalability (also known as elasticity)* is the extent to which the application's capacity can be adapted to the demand of end-user [19]. Two types of scalability can be distinguished: *Horizontal* [20, 21] and *Vertical* [22, 23]. The former refers to the provisioning of multiple instances of the cloud service (*e.g.* deploy new VMs). The latter implies adding more resources to a current cloud service (*e.g.* adding more processors or storage to a VM).
- *Variability (also known as stability)* refers to the variation of performance of a cloud service. Unlike the availability and the reliability that are either provided by the cloud provider or that can be easily calculated, variability depends on the values of the capacities (as explained below). Therefore, variability can be considered as a derived capacity. Several metrics have been employed to evaluate variability [24]. Here, the Coefficient of Variation (CV) has been used, which is defined as the ratio of the standard deviation to the mean (Equation 1.3). The CV is useful for comparison between data sets with different units (as it is the case for most of the benchmarks), since it allows for the comparison of the degree of variation from one data set to another.

$$\mathrm{CV} = \frac{1}{\bar{x}} \cdot \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{1.3}$$

Here, N is the number of measurements; $x_1, ..., x_N$ are the measured results; and $\bar{x}$ is the mean of those measurements.

Capacities

The presented capacities have been defined by Li *et al.* in [8].

- *Transaction speed* defines how fast transactions (*e.g* job execution, read/write operations) can be processed.
- *Data throughput (Bandwidth)* is considered as the amount of data processed by any physical property in a given period of time.
- *Latency* includes all the time-related capacities of a cloud service.
- *Other* consists of dimensionless metrics (*i.e.* availability, CV) or single metrics such as the reliability.

Placement in cloud brokering

The placement or resource allocation in cloud brokering refers to the mechanisms used to distribute infrastructure resources across multiple cloud providers based on end-user' needs and constraints. The optimization goal in placement is to select a single or a set of cloud providers to optimally deploy a service based on optimization criteria, for example, cost optimization or performance optimization. Placement mechanisms can be classified into *non-functional requirements-based placement* and *application-aware placement.* The non-functional requirements-based placement corresponds to the allocation of cloud infrastructure based on matching both cloud provider resources and end-user requirements. The application-aware placement is based on the constraints that guarantee a Quality of Service (QoS) of the application running on top of the infrastructure.

Non-functional requirements-based placement

Placement studies based on non-functional requirements consider performance of cloud providers and/or dynamic pricing scenarios[1]. In the literature, two cloud brokering placement scenarios have been identified: static and dynamic. Static placement assumes that changes within the

[1]Another non-functional requirement, out of the scope of this book, covers end-users limiting the set of placement solutions due to political and legislative considerations. For example, end-users could avoid placing data either outside or inside a given region (*e.g.* the EU Data Protection Directive which regulates the processing and free movement of personal data within the European Union).

cloud environment never happen. Dynamic placement addresses the issue of how to reconfigure cloud resources optimally, adapting them to new situations when conditions change (*e.g.* cloud provider outage, new VM prices, and *etc.*). The approaches described below are based on exact models (*e.g.* binary integer programming formulation).

Static placement

Tordsson *et al.* [25] propose an architecture for cloud brokering and a placement algorithm based on the performance of the GridNPB/ED benchmark and the price of resources. An end-user may constrain resource deployment by specifying the type and number of VMs to be deployed and the percentage of VMs located within each cloud provider. Chaisiri *et al.* [26] propose an optimal VM placement across multiple cloud providers that considers both reserved and on-demand provisioning plans. However, including the reservation plan implies not only a long-term commitment in exchange for lower prices regarding on-demand service provisioning but also raises new issues in case of underprovisioning or overprovisioning of IaaS resources. On one hand, in the underprovisioning scenario, the demand can be fully met through on-demand resources at a higher cost. On the other hand, in the overprovisioning scenario, questions arise such as: Who (the end-user or the cloud broker) is going to pay for the unutilized IaaS resources?

The usefulness of cloud brokering placement for fully-decoupled or loosely-coupled applications is studied by Van den Bossche *et al.* [27] and Moreno-Vozmediano *et al.* [11]. Both approaches improve the cost-effectiveness of the deployment and consider an on-demand provisioning plan and a hybrid IaaS cloud architecture. Van den Bossche *et al.* [27] propose a cost-optimal placement for preemptible but non-provider-migratable batch workloads with a strict completion deadline. The workloads are characterized by memory, CPU and data transmission requirements. The problem is tackled by Linear programming. Moreno-Vozmediano *et al.* [11] evaluate the scenario of deploying a computing cluster on top of a multi-cloud infrastructure for solving loosely-coupled Many-Task Computing (MTC) applications. The goal is to improve the cost-effectiveness of the deployment, or to implement high-availability strategies. This approach is evaluated through a low scale testbed including a local data-center and three different public cloud sites.

This testbed is complemented with simulations that include a larger number of resources.

Dynamic placement

Lucas-Simarro *et al.* [28] propose a VM placement algorithm with the goal of minimizing the costs for end-users in a dynamic pricing environment. The cloud broker transfers a client's infrastructure from one cloud provider to another based on price fluctuations. The algorithm calculates possible future prices based on the average cloud provider's price and its price trend. In order to guarantee the performance of the applications running on top of the IaaS resources, the placement decisions are constrained by: The maximum and minimum number of VMs to reallocate in each placement and a load balancing requirement that indicates the percentage of resources to maintain within each cloud provider. In this approach, the placement problem is limited to one VM configuration. Lucas-Simarro [29] extends this work to multiple VM configurations and addresses the problem of performance optimization. Performance optimization consists in maximizing the performance of the deployed resources by choosing the VMs with the best performance in terms of hardware resources (hard disk, memory, CPU). A drawback of this approach is that VM performance measurements can only be provided by end-users after testing all VM configurations within each cloud provider.

A more complex model that not only involves cost-optimization but also copes with changes in the cloud environment through VM migration is proposed by Tordsson *et al.* [30]. In this model, the time for VM migration is estimated as the time required to shut down a VM within one cloud provider and start a new VM with the same configurations within another.

Chaisiri *et al.* [31] propose an optimal cloud resource provisioning algorithm minimizing the cost of resource provisioning for a certain period given the uncertainty for demand and price. The optimal decision calculated by the cloud broker is based on the end-users' demands and cloud providers' prices. This allows the cloud broker to adjust the number of resources acquired in advance under reservation and the number of resources to be acquired under on-demand provisioning, taking into account that reserved VMs are generally cheaper than on-demand ones. This approach tackles the underprovisioning and overprovisioning problem.

Chaisiri addresses this problem through stochastic integer programming.

Application aware placement

The application-aware placement dynamically scales resources up or down across multiple cloud providers' infrastructures under QoS constraints specific to the application. In the case of tightly-coupled applications with low delay or strong communication requirements, the placement process should guarantee a single-cloud deployment [32]. However, in the case of fully-decoupled[2] or loosely-coupled applications, the placement process may take advantage of the heterogeneity of cloud providers' offers to deliver a cost-effective solution that guarantees the performance of the application [27, 33]. In the case of interactive applications (*e.g.* on-line gaming), user experience relies on network bandwidth and on the latency caused by geographical distances [32]. Therefore, these kinds of applications should be treated near the geographical location of their origin to achieve lower latency and higher throughput.

The importance of cloud brokering for telecommunication services is highlighted by Carella G. *et al.* [34]. In this approach, the cloud broker enhances his placement mechanisms based on: real-time data on network performance, QoS requirements and cloud providers' prices. The goal is to provide to telecommunication service operators a minimum QoS to satisfy customer requirements by monitoring the deployed services. This approach is evaluated in a testbed composed of a cloud broker and an IP Multimedia Subsystem (IMS) deployment. The cost-effective placement of Web 2.0 applications with high-availability and fault-tolerance requirements across multiple cloud providers is proposed by Frincu *et al.* [35]. In this approach, authors consider applications consisting of several components and connectors (C/Cs). C/Cs are reallocated by making a snapshot, stopping the execution of each C/C, moving the snapshot to a new VM and starting the C/C from the snapshot. A cloud broker architecture with the intelligence to react to changes in business processes by changing the cloud configuration across multiple cloud providers is described by Grivas *et al.* [36].

The placement of services with different QoS and service provisioning

[2]Applications are fully-decoupled when the jobs that form the application have no precedence constraints, and can be executed in parallel.

requirements for risk assessment services and e-learning education applications is tackled by Quarati *et al.* [37]. The goal is to maximize user satisfaction and broker revenues by reducing energy costs, through energy saving mechanisms. For this, the cloud broker allocates IaaS resources to the public or private cloud, based on end-user's QoS expectations and the workload of the private resources. This approach was evaluated through a discrete event simulator.

Conclusion

In this chapter, research work tackling the problem of cloud performance evaluation and placement in cloud brokering has been surveyed. A shortcoming in the current approaches to cloud performance evaluation is the absence of a single figure of merit that provides a straightforward comparison of cloud providers. Regarding the problem of placement in cloud brokering, the surveyed studies assume that cloud providers offer the same type of VM configurations. This assumption is not true for all the cases; VM configurations may vary from one cloud provider to another. For some cases, even a VM offered by a cloud provider in one location, may not exist in another location belonging to the same cloud provider. These issues are tackled in the next two chapters.

Chapter 2

Towards a figure of merit for cloud performance

Introduction

Let's imagine creating a figure of merit for automobiles and that the most expensive Mercedes-Benz has the highest figure of merit. Does this mean that everyone should buy that particular car? What if you want to tow a trailer? In cloud computing, although we may be able to find a single value to represent cloud performance, it does not mean that this value will be useful for all types of applications. According to the physical property that limits performance, applications can be classified as CPU-bound, memory bound or I/O bound (Figure 2.1). Therefore, the application profile must be taken into account in the calculation of a single figure of merit.

Currently, the information given by cloud providers allows for a simple but inaccurate comparison between providers. End-users can choose a cloud provider by comparing quantitatively different VM offers (*e.g.* number of cores per dollar, memory or storage capacity per dollar). Using this simple approach, end-users can select the cloud provider that offers the largest quantity of resources at the lowest price. However, this only makes sense in the unreal scenario in which cloud providers have qualitatively homogeneous resources.

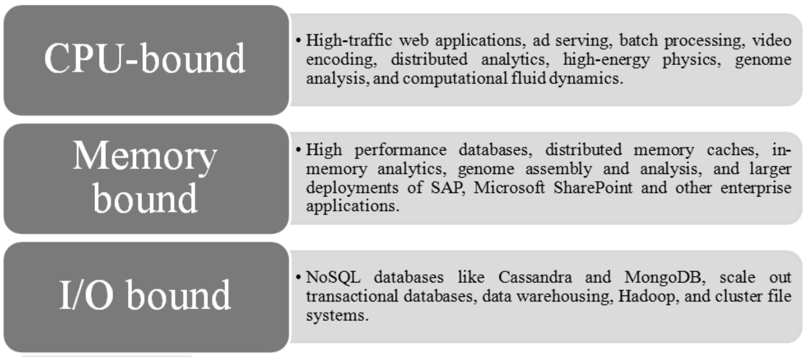
CPU-bound
• High-traffic web applications, ad serving, batch processing, video encoding, distributed analytics, high-energy physics, genome analysis, and computational fluid dynamics.
Memory bound
• High performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, and larger deployments of SAP, Microsoft SharePoint and other enterprise applications.
I/O bound
• NoSQL databases like Cassandra and MongoDB, scale out transactional databases, data warehousing, Hadoop, and cluster file systems.

Figure 2.1: Examples of types of applications[a]

[a]Source: http://aws.amazon.com/ec2/instance-types/

Another more precise alternative for comparing cloud providers, is to evaluate the performance of an application across multiple cloud providers and to choose the set of cloud providers with the best performance-cost ratio [29]. This approach is technically feasible through a cloud broker but requires a time-consuming and highly expensive task: The application must be evaluated in each VM configuration offered by each cloud provider. Moreover, the performance estimation becomes increasingly inaccurate as the cloud providers upgrade their infrastructures (Section "Motivations and challenges"). In contrast to the methods described above, a figure of merit for cloud performance based on the application profile, can serve as a general guide for the performance of a particular system configuration and provide a straightforward comparison of cloud providers for a given type of application. In this chapter, a figure of merit for cloud performance is calculated, by running benchmarks in advance across multiple cloud providers and by obtaining a composed metric based on the benchmarks' results.

The motivation behind a figure of merit in cloud brokering is twofold. Firstly, the performance of the cloud providers can be measured in advance with consistent results for a period of time. This is due to the fact that the time to run benchmarks is much shorter than the upgrade cycle of cloud infrastructures. Secondly, a cloud broker, granting access to multiple cloud infrastructures as a trusted third-party, may provide up-to-date evaluations of cloud provider performance. Cloud brokers may automatically deploy benchmarks and process the results. Thereby, a cloud broker can easily automate the process for calculating a figure of merit. This chapter is organized as follows. Section "Performance evaluation" describes, the methodology and the experimental setup used in this cloud evaluation. This is followed by the evaluation of the provisioning time and the evaluation of criteria such as computation, memory, storage and variability for different types of VMs and cloud providers. In Section "Figure of merit of VM cloud performance", two approaches to calculate a single figure of merit for cloud performance are presented. Finally, Section "Case study: CPU-Intensive application" presents a case study for a CPU-intensive application. In this case study, three different methods have been used to calculate a figure of merit. Real performance result have been used in the study.

Performance evaluation

Evaluation methodology

There is a lack of standardized methodology for cloud performance evaluation through benchmarks (*cf.* Section "Studies related to cloud providers performance evaluation"). The methodology employed in this work to measure cloud performance. Is composed of five main steps (Figure 2.2):

1. *Define scenarios:* the stakeholders (*i.e.* cloud providers to be evaluated) are identified, as well as the features related to the cloud services such as VM configurations and datacenter locations.

2. *Identify Benchmarks:* selection of suitable benchmarks according to the scenario formulated in the previous step. If the evaluation of a specific application is going to be performed, this step is omitted. Evaluation-related issues such as the number of benchmark repetitions and the type of workload are defined in this step [8].

3. *Run tests:* the resources are acquired on the selected cloud providers' locations. Then, benchmarks are deployed into the chosen VM configurations. At the end of this step, the results are collected and the resources are released.

4. *Process results:* the results are treated and synthesized. For example, by calculating a figure of merit for cloud performance or by generating a graphical representation that summarizes the main results, aspects and trends.

5. *Analyze results:* In this final step, comments or recommendations are formulated based on the results.

Define scenarios
Identify benchmarks
Run tests
Process results
Analyze results

Figure 2.2: Evaluation methodology

Experimental setup

The initial idea was to create an Operative System (OS) image consisting of all the scripts and benchmarks necessary to measure cloud performance. This image would be uploaded and deployed in every cloud. Thus, the same workload conditions for every cloud provider is guaranteed. However, the cloud providers, covered in this study, present issues that hinder or completely prevent VM import. In some cases, cloud providers only support the import of VMs generated via licensed software (*e.g.* Amazon only support the import of images generated with VMware vSphere Client). In other cases, the import of VM images is only supported for some of the image formats (*e.g.* Cloudsigma only supports import of VMs in *RAW* format). Finally, particularly in recently emerged cloud providers, the import of VM images is not supported at all. For these reasons, the choice was made to build a VM image from the images already offered by cloud providers. The experimental setup presented here consists of three phases: image setup, running benchmarks and processing benchmark results. These phases occur once the accounts have been created in every cloud provider to be evaluated and the payment details have been registered. More issues related with this evaluation of performance are presented in Appendix B.

The image setup is as follows. First, a VM via web interface or command line is created. During the VM creation, the OS system is chosen. In this setup, Linux CentOS 6.X for a 64 bits processor architecture, an OS supported by the majority of cloud providers, has been chosen. Once the VM has been created, the OS is updated and a *ssh* server is installed to enable a secure remote control of the VM. Then, the scripts, benchmarks and tools necessary to evaluate cloud performance are installed and configured. The scripts have been developed in Python. The execution permissions of the */etc/rc.local* file have been modified and changed, in order to automatically trigger the benchmarks once the VM is turned-on. The *phoronix-test-suite* [38] has been selected as the framework to deploy benchmarks due to its wide set of supported benchmarks (more than 350). For the transmission of benchmark results, *s3cmd*, a command line tool for using the Amazon S3 service, has been used. The image containing all the scripts, benchmarks and tools necessary to evaluate cloud performance has been called *ceilo* (Figure 2.3). As explained above, the benchmarks are triggered automatically and sequentially once the VM is turned-on. Once all the benchmarks have been executed, the results are sent to an Amazon S3 bucket and the VMs are automatically turned-off. Benchmark results correspond to XML files. Thus, result files are parsed with *xsltproc* [39], a command line tool for applying XSLT stylesheets to XML documents; and values such as the average, the variance, the standard deviation and the coefficient of variation are calculated. Finally, for some of the results, a script to automatically generate graphical representations of the results with *google charts* is run.

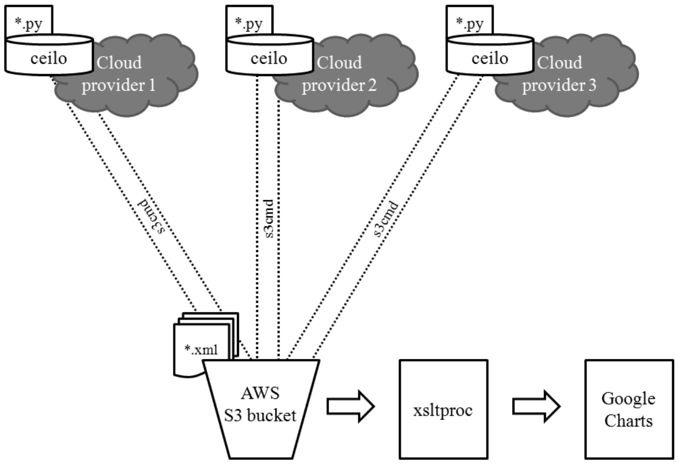

Figure 2.3: Experimental setup

In this study, the performance of 37 different VM types across 8 cloud providers with data centers in Europe has been measured (Table 2.1). According to the pre-configured VMs offered by some cloud providers and in order to compare VMs with similar capacities, five different sizes have been defined: VM sizes (*xs*, *s*, *l*, *m* and *xl*) (Table B.1). This classification has been based on the number of virtual CPUs (vCPUs) and the RAM memory size.

Id	Cloud provider
ARU	ArubaCloud
AWS	Amazon Web Services
CLO	Cloudsigma
JOY	Joyent
LUN	Lunacloud
PRO	Profitbricks
RAC	Rackspace
WIN	WindowsAzure

Table 2.1: Evaluated cloud providers

Criteria	**Capacity**	**Benchmark**	**Metric**	**Type**
Computation	Transaction speed	7-zip [40]	MIPS	HB
		C-Ray [41]	seconds	LB
Memory	Data throughput	Stream [42]	MB/s	HB
		CacheBench [43]	MB/s	HB
Storage	Transaction speed	Threaded I/O Tester [44]	MB/s	HB
		Iozone [45]	MB/s	HB

Table 2.2: Benchmarks

The performance evaluation presented here is based on 6 benchmarks. These benchmarks measure the computation, memory and storage capacities (Table 2.2). Depending on the operation implemented by the benchmark, the magnitude of the results can be classified as: Lower is Better (LB) or Higher is Better (HB). LB means that the lower the value, the better the system to execute a given benchmark. Inversely, HB means that the higher the value, the better the system to execute a given benchmark.The provisioning time has been measured with the scripts developed for this research. In this section, the mean values and the standard deviation of the obtained results have been plotted. An analysis related to the benchmark duration is presented in Appendix B.

Provisioning time

The provisioning time (or scaling latency) is defined as the time taken by a cloud provider to allocate a new VM once the end-user requests it [5]. The provisioning time corresponds to the amount of time it takes for a cloud provider to power-on a VM (VM provisioning time) and the boot time of the OS, defined as the time between when the VM has been powered-on and the VM is ready to be used. The provisioning time has a direct impact in the scalability of a cloud application, particularly in peak load scenarios, where the deployment of cloud infrastructure must follow the workload variations and the VMs must be ready to be used as soon as possible.

The VM provisioning time for every VM size of WindowsAzure and Amazon has been measured (Figure 2.4). In general, WindowsAzure has a higher provisioning time and a larger standard deviation than Amazon. The VM provisioning time for Amazon stays under 25s while for WindowsAzure it is consistently over 25s.

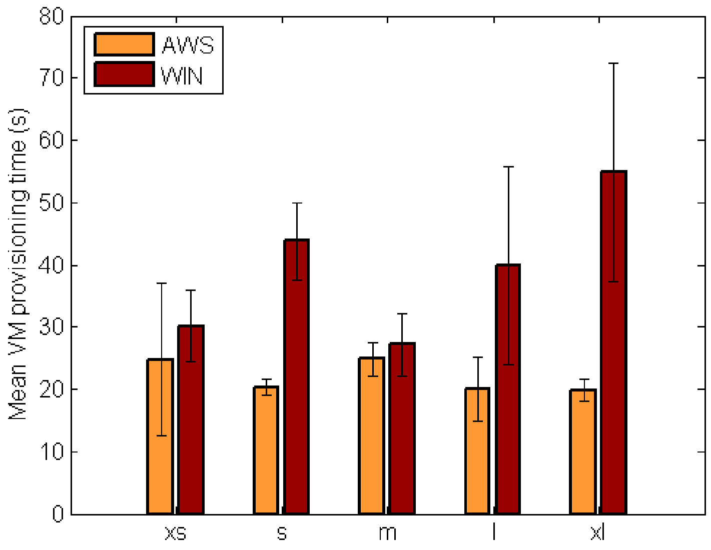

Figure 2.4: Average VM provisioning time for Windowsazure and Amazon

The boot time for every VM size of every cloud provider has also been measured (Figure 2.5). In 5 out of 8 evaluated cloud providers, the VM boot time varies from 10 to 25 seconds and is independent of the VM size. In the case of Joyent, the boot time is inversely proportional to the VM size. Arubacloud presents a VM boot time that varies from 40 to 50 seconds*s* and it is independent of the VM size. Lunacloud presents the lowest boot time values for *xs*-, *s*-, *m*- and *xl*-VM sizes. Lunacloud's *l*-size VMs present the highest boot time among all the evaluated cloud providers. There is not a logical explanation to this fact, from the collected data. It has been inferred: Lunacloud's *l*-size VMs shared the same processors family (Intel Xeon E5-2620) with the other Lunacloud's VM sizes. Thus, the processor brand is unlikely to be the reason for these boot time differences. Unfortunately, there are no additional results or information about the Lunacloud's underlying infrastructure to determine the reasons behind this high boot time.

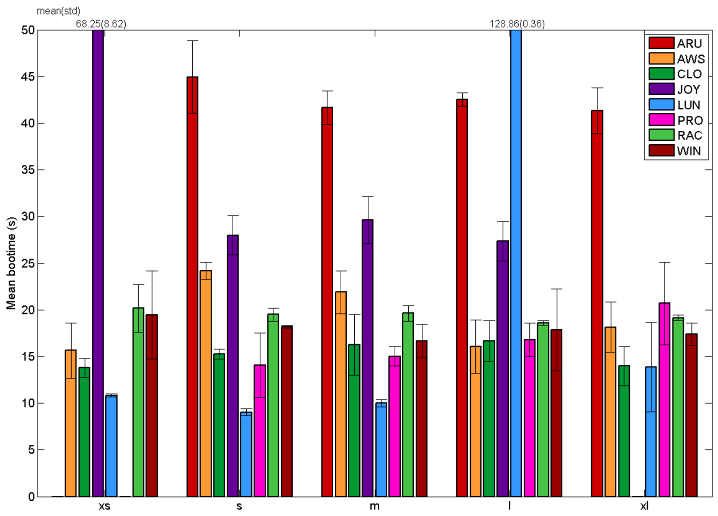

Figure 2.5: Average boot time

Computation benchmarks performance

The transaction speed has been measured with *7-Zip* and *C-Ray* benchmarks (Figure 2.6). 7-Zip is an application to compress files. The 7-Zip benchmark consists of compressing a file with random data and measuring the number of CPU instructions executed during the compression. C-Ray measures floating point CPU performance. By default, the benchmark uses only a small amount of data, such that on most systems the CPU does not have to access the RAM to run the benchmark. In this performance evaluation, C-Ray was set up to measure the time to render an image with a resolution of 800x600 pixels. Therefore unlike for 7-Zip in C-Ray, lower results are better.

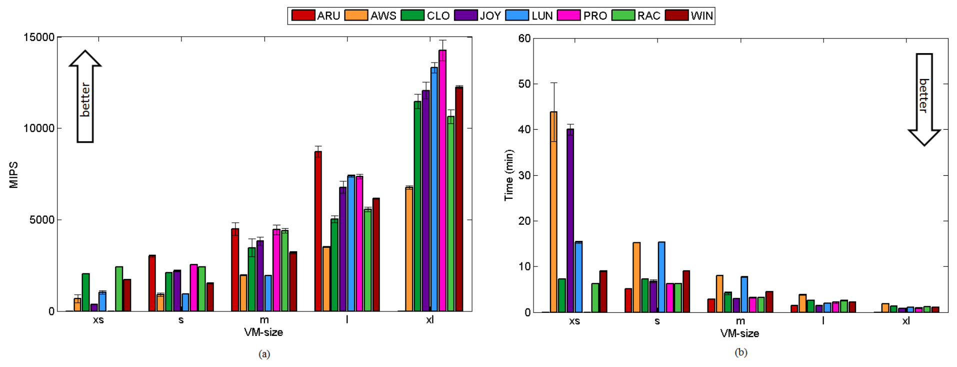

Figure 2.6: Performance of computation benchmarks (a) 7-Zip results (HB) (b) C-Ray results (LB)

Memory benchmarks performance

RAM and cache memory bandwidth have been measured with the *Stream* and *Cachebench* benchmarks, respectively (Figure 2.7). Stream is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernels. In this performance evaluation, Stream was set up to measure the memory bandwidth through the *copy* and *add* operations. The copy operation consists of fetching two values from memory and updating the value of one of these fetched values with the other. The add operation fetches three values from memory and updates one of the fetched values with the sum of the other two fetched values. Cachebench is a benchmark designed to evaluate the performance of the cache memory present on a system. In this performance evaluation, CacheBench was set up to measure the cache memory bandwidth through *read* and *write* operations. In general, CacheBench results show the writing speed is around 60%-80% faster than the reading speed (Figure 2.7e).

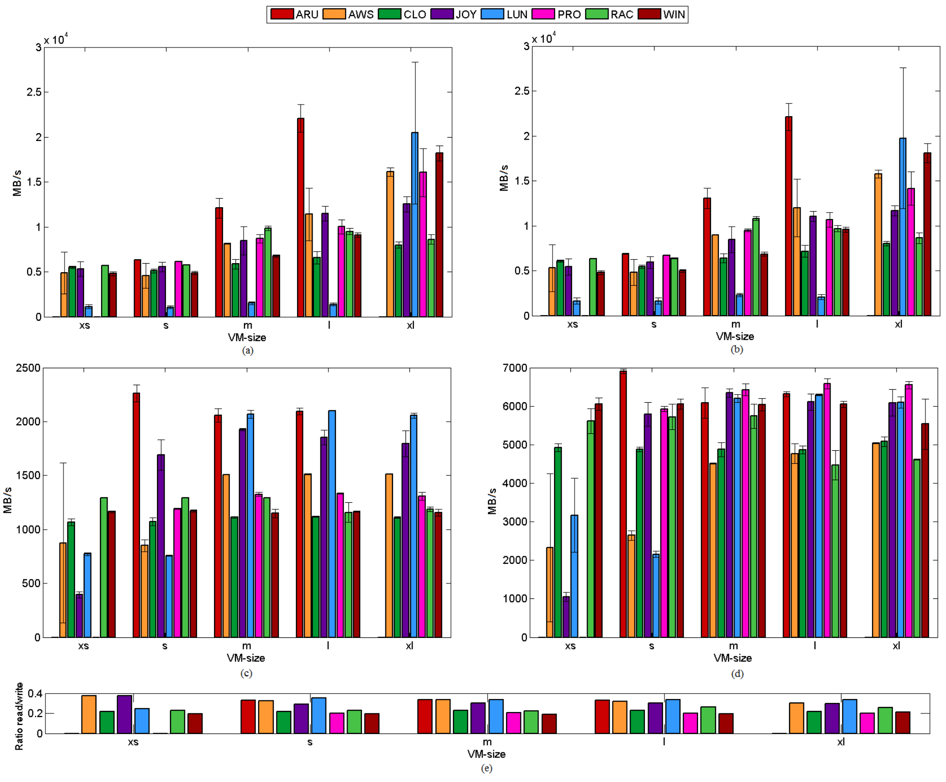

Figure 2.7: Performance of memory benchmarks. (a) Stream results for copy operation (b) Stream results for add operation (c) CacheBench results for read operation (d) CacheBench results for write operation (e) Ratio CacheBench read/write speed.

Storage benchmarks performance

The storage bandwidth has been measured with the *Iozone* and *Threaded I/O Tester (TIO)* benchmarks (Figure 2.8). Iozone is a filesystem benchmark tool. The benchmark generates and measures a variety of file operations. In this performance evaluation, Iozone was used to measure the transaction speed for reading and writing a file of 2GB. Similarly, the read and write speeds have been measured with TIO for a 64MB file by using 16 threads. For the small VM sizes (*xs* and *s*), the read and write speeds are comparable for both benchmarks. For the *m*-, *l*- and *xl*-VM sizes, the read speed is at least ten times faster than the write speed (Figure 2.8c). Iozone's read/write ratio is bigger than 150 for Amazon (Figure 2.8f).

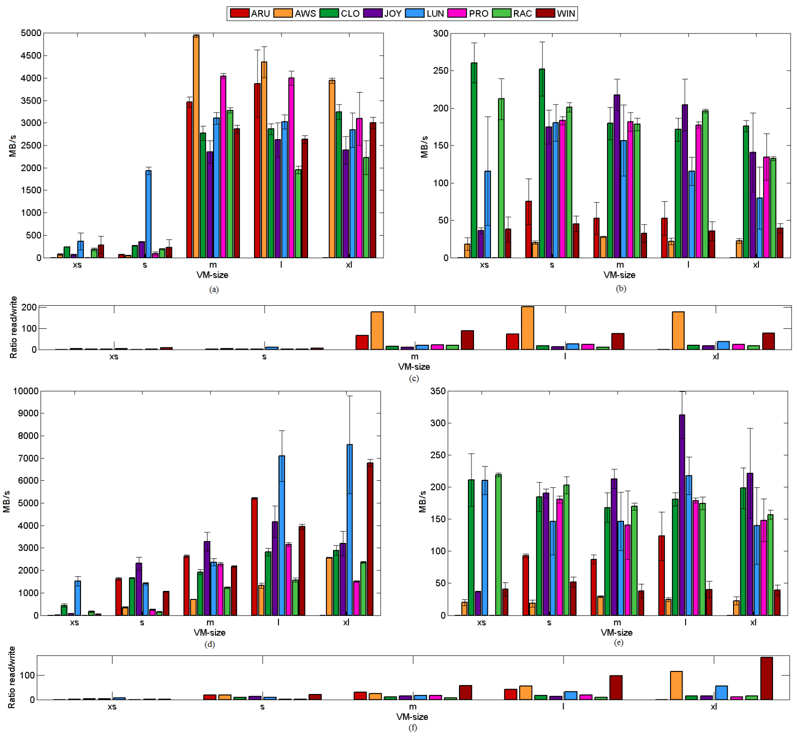

Figure 2.8: Performance of storage benchmarks. (a) Iozone read speed (b) Iozone write speed (c) Ratio Iozone read/write speed (d) TIO read speed (e) TIO write speed (f) Ratio TIO read/write speed.

Variability

The variability of VMs has also been studied. For this, a single value of variability has been calculated by averaging the Coefficient of Variation (CV) of all the benchmark results. The distribution of variability (Figure 2.9) shows that 70.3% of the evaluated VMs have a variability less or equal to 10%. Since physical servers host many VMs at the same time, one should expect that the bigger the VM size, the lower the variability, and vice versa. However, results show that even small VM sizes present low variability values. The percentage of VMs with a variability between 40% and 45% corresponds to the *xs*-VM size of AWS. One possible explanation to this fact is that the number of processor of the AWS's *xs*-VMs is not constant, providing spiky CPU resources [46].

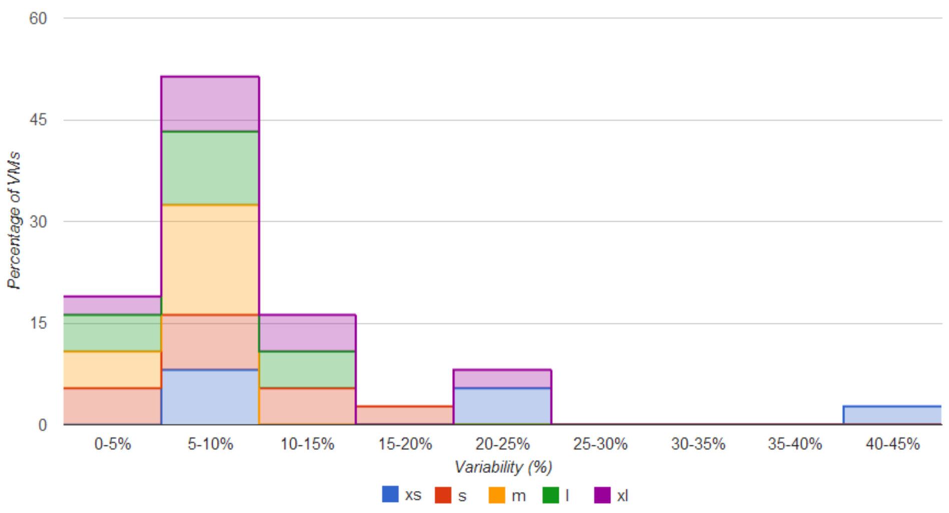

Figure 2.9: Distribution of variability for the measured VMs

Figure of merit of VM cloud performance

No benchmark offers a holistic view through a single score of cloud performance. Instead, benchmarks have their own specific metrics and magnitudes to express results. This heterogeneity in results prevents a straightforward, simple calculation of an absolute figure of merit for VM cloud performance. Moreover, even in the case of benchmarks sharing the same units to express results, it is incorrect to directly add values from different benchmarks. The reason is that the magnitudes of values can differ significantly, for example, read and write cache results differ by a magnitude of three. Therefore, it is not an easy task to choose a cloud provider based on individual benchmark results. In this section, some methods to calculate a figure of merit for VM cloud performance to allow a simple cloud provider selection are presented.

Mean and radar plot as figures of merit

Most of the performance evaluation studies report individual benchmarking results (Table A.1). In an attempt to express the holistic performance of a cloud service through a single score, Li *et al.* [17] propose *Boosting*[1] *Metrics* such as the *mean* (*eg.* arithmetic, geometric, harmonic) and the *radar plot*. The *geometric mean*, by definition, is the n^{th} root of the product of the n units in a data set (Equation 2.1). There is a defect when employing means as the method of boosting metrics: The results from different benchmarks must use the same units. This shortcoming is overcome by using a radar plot.

$$M = \sqrt[n]{\prod_{i=1}^{n} \text{Benchmark}_i} \tag{2.1}$$

$$\text{HB Standardized}_i = \frac{\text{Benchmark}_i}{\text{MAX}(\text{Benchmark}_{1,\ldots,n})} \tag{2.2}$$

[1]The boosting concept comes from the machine learning field. In cloud service evaluation, boosting refers to the creation of a measurement based on primary metrics that measure individual cloud service features.

A radar plot is a simple graphical tool that can depict three or more quantitative values relative to a central point (Figure 2.10). When benchmark results are expressed in different metrics, Li *et al.* propose two standardization methods to express results over a predefined baseline: Higher is Better (HB) (Equation 2.2) and Lower is Better (LB) (Equation 2.3). HB (LB) means the higher (the lower) the benchmark result, the better.

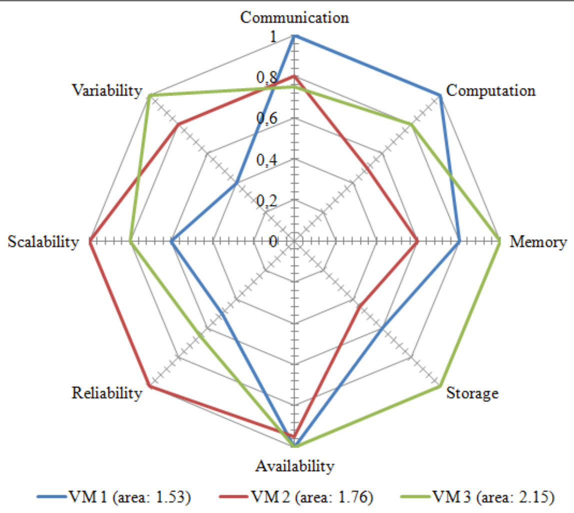

Figure 2.10: Radar plot as a figure of merit

$$\text{LB Standardized}_i = \frac{\frac{1}{\text{Benchmark}_i}}{\text{MAX}(\frac{1}{\text{Benchmark}_{1,\ldots,n}})} \tag{2.3}$$

Here HB Standardized$_i$ and LB Standardized$_i$ refer to the standardized i^{th} benchmark result. Thus, the area of the polygon representing n standardized benchmarking results can be considered as a figure of merit for cloud performance (Equation 2.4) [17].

$$\text{Figure of merit}_{(\text{radar plot})} = \sum_{i=1}^{n} \frac{\sin\left(\frac{2\pi}{n}\right) \times \text{Standardized}_i \times \text{Standardized}_{mod(i+1,n)}}{2} \tag{2.4}$$

Although these metrics result in a figure of merit, they present huge flaws such as lack of weighting and categorical scores[2].

Simple figure of merit

The simple figure of merit for cloud performance is a method similar to the one employed by companies reporting cloud performance such as CloudSpectator or Cloudharmony. It is called simple since it does not take into account the trade-off among the different criteria. In this method, each benchmark result is scaled between two fixed values, A and B. Where A is the lower bound corresponding to the worst performance result (wpr) and B is the upper bound corresponding to the best performance result (bpr). The intermediate values (x_i) are calculated with Equation 2.5. Then, all the scaled values are averaged and a single figure of merit is obtained for each VM configuration. This method has been applied to the data previously reported (*c.f.* Section 2) to obtain a figure of merit for cloud performance (Figure 2.11). More results based on the simple figure of merit method can be found in Appendix B.

$$\text{Performance score} = \begin{cases} A + \frac{B-A}{\text{bpr}-\text{wpr}}(x_i - \text{wpr}) & \text{if HB benchmark} \\ B - \frac{B-A}{\text{bpr}-\text{wpr}}(x_i - \text{bpr}) & \text{if LB benchmark} \end{cases} \tag{2.5}$$

[2]Score with a limited and usually fixed, number of possible values.

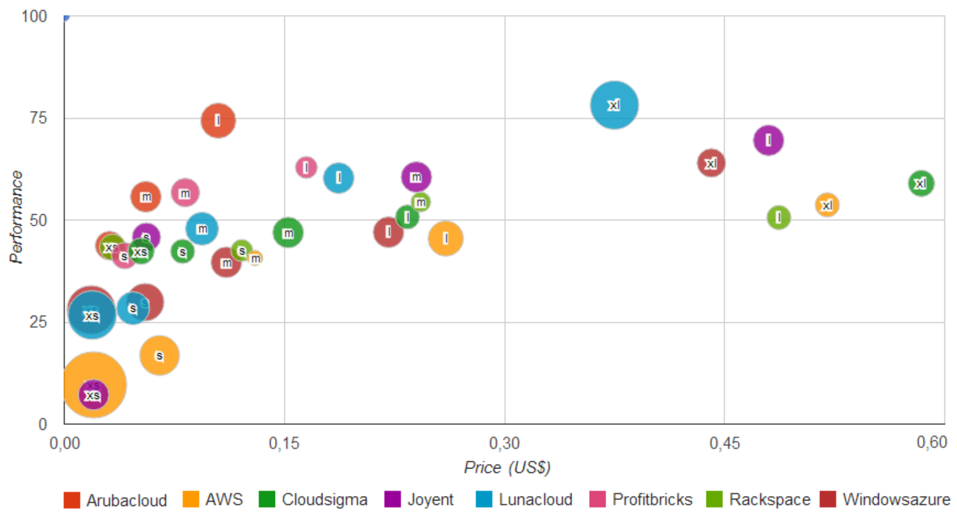

Figure 2.11: Correlation between performance and price for different VM sizes. The variability is represented by the size of the spot. A=1 and B=100.

Figure of merit based on Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) is a structured technique for analyzing, organizing and solving problems related to Multiple Criteria Decision Making (MCDM) [47]. In AHP, complex problems are simplified and structured by arranging the decision factors in a hierarchical structure. The trade-offs among criteria are determined by a pairwise comparison. Unlike the traditional weighted sum-based methods, AHP is based on both subjective and objective evaluation measures. In cloud computing, AHP has been used to rank cloud services [18, 48]. In this section, AHP is used to determine the relative merit of members of a set of alternatives. This process consists of three phases: hierarchy structure modeling, judgement of priorities and hierarchical synthesis.

Phase 1: Hierarchical structure modeling

In this phase, the problem is defined and the goal is determined. Also, all the criteria that have an influence in resolving the issue are identified, as well as the alternatives that offer an answer to the problem. Both criteria and alternatives are organized in a hierarchical structure. The hierarchical structure used here (Figure 2.12) is based on the performance criteria described previously (*c.f.* Section 1). The alternatives correspond to the different cloud providers supported by a cloud broker. Each alternative represents a set of benchmark results that contains a figure of merit for each criterion. In this case, the goal is to find a figure of merit for a cloud infrastructure based on performance.

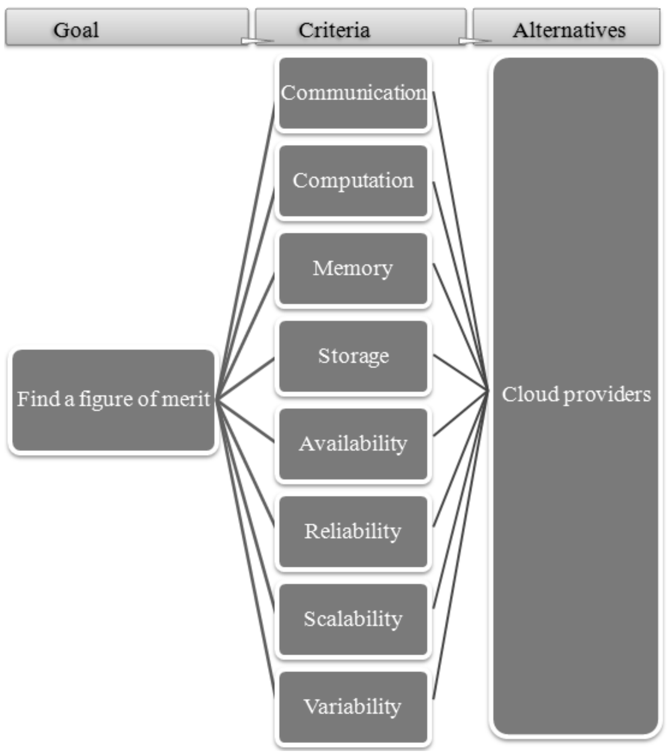

Figure 2.12: Hierarchy of the problem

Phase 2: Judgement of priorities

Pairwise comparisons are used to determine the relative importance of each alternative and each criterion. Saaty [47] proposes a relative rating scale (Table 2.3) by which the decision-maker expresses his opinion about the relative importance of one criterion over another. This scale allows for the quantification of the pairwise comparisons. This phase leads to the construction of P pairwise comparison matrices of size N-*by*-N, where N is the number of alternatives and P is the total number of criteria. One additional matrix C, the pairwise comparison criteria matrix, is constructed to express the relative weights between each one of the criteria to be evaluated.

Intensity of importance	**Definition**	**Explanation**
1	Equal importance	Two activities contribute equally to the objective
3	Weak importance of one over another	Experience and judgement slightly favor one activity over another
5	Essential or strong importance	Experience and judgement strongly favor one activity over another
7	Demonstrated importance	An activity is strongly favored and its dominance demonstrated in practice
9	Absolute importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between the two adjacent judgements	When compromise is needed
Reciprocals of above nonzero	If activity i has one of the above nonzero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i	

Table 2.3: Relative rating scale [47]

Phase 3: Hierarchical synthesis

Once all comparisons have been made in Phase 2, the numerical probability of each alternative is calculated. This probability determines the likelihood that the alternative has to fulfill the expected goal. This process is also applied to the matrix C that expresses the relative weights between each one of the criteria. The hierarchical synthesis phase is applied to the pairwise comparison matrices as follows:

1. Synthesize the pairwise comparison matrix. Given that the pairwise matrix S is of size N-by-N, the synthesized pairwise comparison matrix (A) is obtained by dividing each value of A by the total of its column, as follows:

$$\forall n \in [1...N], \forall i \in [1...N] : a_{ij} = \frac{s_{ij}}{\sum_{i=1}^{N} s_{in}} \tag{2.6}$$

   where a_{ij} is an element of matrix A in row i and column j.

2. Calculate the priority vector (V). The priority vector corresponds to the eigenvector of matrix A. The priority vector can be approximated to the average value of each row of matrix A (Equation 2.7), in order to avoid the mathematical effort required to calculate an eigenvector [49].

$$\forall v \in [1...N] : v_i = \frac{\sum_{i=1}^{N} a_i}{N} \tag{2.7}$$

3. Calculate the maximum eigenvalue (λ_{max}). λ_{max} is calculated by adding the product of each element of vector V by the sum of its corresponding column of matrix S (Equation 2.8).

$$\lambda_{max} = \sum_{j=1}^{N} v_j \cdot \sum_{i=1}^{N} s_{ij} \tag{2.8}$$

4. Calculate the Consistency Index (CI):

$$\text{CI} = \frac{\lambda_{max} - N}{N - 1} \tag{2.9}$$

5. Check the consistency of the pairwise comparison matrix (S). Saaty [50] suggests the Consistency Ratio (CR) in order to determine if the pairwise comparisons made by the decision maker are consistent. For example, consider three criteria x, y and z. If the decision maker has considered $x>y$ and $y>z$, then it would be inconsistent to consider that $x<z$. The CR is calculated as follows:

$$\text{CR} = \frac{\text{CI}}{\text{RI}} \tag{2.10}$$

where the Random Consistency Index (RI) is a fixed value provided by Saaty [50] (Table 2.4). The decisions are considered as consistent when $CR < 0.1$

N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

Table 2.4: Random Consistency Index (RI) [50]

Finally, the overall performance for each alternative is calculated. For this, a matrix L of size P-*by*-H is formed. In matrix L, each row corresponds to one of the H priority vectors found for each one of the P criteria. The overall performance (OP) corresponds to the product of the priority vector (G) of matrix C and the matrix L (Equations 2.11 and 2.12). Each element of vector OP corresponds to the performance of one of the H alternatives.

$$\text{OP} = GL \tag{2.11}$$

$$\mathrm{OP} = \begin{pmatrix} g_1 & g_2 & \cdots & g_P \end{pmatrix} \begin{pmatrix} l_{1,1} & l_{1,2} & \cdots & l_{1,H} \\ l_{2,1} & l_{2,2} & \cdots & l_{2,H} \\ \vdots & \vdots & \ddots & \vdots \\ l_{P,1} & l_{P,2} & \cdots & l_{P,H} \end{pmatrix} \tag{2.12}$$

Case study: CPU-intensive application

The objective of this case study is to find a single figure of merit for cloud performance for a CPU-intensive application (*e.g.* file encryption, encoding, scientific computing). In order to compare the different approaches previously presented, a figure of merit with the radar plot has been computed (*c.f.* Section "Figure of merit of VM cloud performance"), using the simple and the AHP techniques. The criteria considered were: computation, memory, storage, availability, scalability and variability. In order to calculate a figure of merit based on real data, the performance values previously obtained have been used and the availability values have been obtained from cloud providers' websites. The availability used is the one for which the cloud provider will not reimburse the end-user in case of service unavailability. This study has been limited to *s*-size VMs. The implementation details per technique are the following:

1. *Radar plot*: Benchmark results are standardized with Equations 2.2 and 2.3. A single figure of merit is calculated with Equation 2.4 (Figure 2.13a).

2. *Simple figure of merit*: Benchmark results are scored with Equation 2.5. Then, the single figure of merit corresponds to the mean value of the scored values for performance (Figure 2.13b).

3. *Figure of merit based on AHP*: The benchmark results are standardized with Equations 2.2 and 2.3. Mean values of benchmarks evaluating cloud performance for the same criterion have been found (*e.g.* in the case of the computation criterion, the mean value of standardized results of 7-Zip and C-Ray was calculated). The pairwise comparison criteria matrix (Table 2.5) considers computation and variability criteria with an equal importance. The

computation criterion is also far more important than the storage and scalability criterion. The memory criterion is slightly more important than the storage, availability and scalability criterion. The availability criterion is considered more important than the storage and the scalability criterion. The priority vector represents the relative importance of each criterion in the single figure of merit.

Criteria	**Computation**	**Memory**	**Storage**	**Availability**	**Scalability**	**Variability**	**Priority vector**
Computation	1	6	9	3	9	1	0.3753
Memory	1/6	1	3	3	3	1/6	0.1250
Storage	1/9	1/3	1	1/3	1	1/6	0.0408
Availability	1/3	1/3	3	1	3	1/3	0.1053
Scalability	1/9	1/3	1	1/3	1	1/3	0.0501
Variability	1	6	6	3	3	1	0.3036

Table 2.5: Pairwise comparison criteria. CR = 0.0702 and RI = 1.24.

The priority vectors are calculated for each criterion based on the these mean standardized values. The priority matrix for assessment of the overall cloud performance (Table 2.6) presents the overall performance for each cloud provider (Figure 2.13c).

Rank	**Criteria/ Provider**	**Computation**	**Memory**	**Storage**	**Availability**	**Scalability**	**Variability**	**Priority vector**
1	**ARU**	0.2507	0.2304	0.3016	0.1788	0.2726	0.2564	0.2455
6	**AWS**	0.0978	0.1152	0.0912	0.1128	0.1151	0.0995	0.1027
2	**CLO**	0.1319	0.1230	0.1334	0.1233	0.1649	0.1501	0.1371
3	**JOY**	0.1319	0.1334	0.1334	0.1233	0.1457	0.1323	0.1320
7	**LUN**	0.0765	0.0720	0.1061	0.1128	0.0641	0.0853	0.0830
4	**PRO**	0.1284	0.1310	0.1042	0.1128	0.0832	0.1040	0.1164
5	**RAC**	0.1074	0.1100	0.0882	0.1233	0.0890	0.1294	0.1144
8	**WIN**	0.0753	0.0850	0.0419	0.1128	0.0653	0.0431	0.0688

Table 2.6: Overall cloud performance matrix

Besides finding a figure of merit for cloud performance, the cost plays an important role in cloud service selection. For this reason, the performance-price ratio has been considered (Figure 2.13d). The results show that for the three methods to compute a figure of merit, ArubaCloud *s*-size VMs present the best performance among all the evaluated cloud providers. In the case, of the radar plot and the simple figure of merit approaches, the performance results for Joyent are close to those of ArubaCloud (Figure 2.13a-b). However, with the AHP approach, it can be clearly seen that in the case of a CPU-intensive application, ArubaCloud doubles in performance when compared to Joyent (Figure 2.13c).

Summary

In this chapter, cloud performance has been evaluated by using micro-benchmarks. The results obtained from benchmarks have been used to calculate a single figure of merit for cloud performance with the radar plot, the simple and the Analytic Hierarchy Process (AHP) techniques. The advantage of the AHP technique, in comparison to other techniques when calculating figures of merit, is that it is based on pairwise comparisons in which users can express the importance of one feature over another. Thus end-users can take into account the requirements an application has in terms of performance criteria.

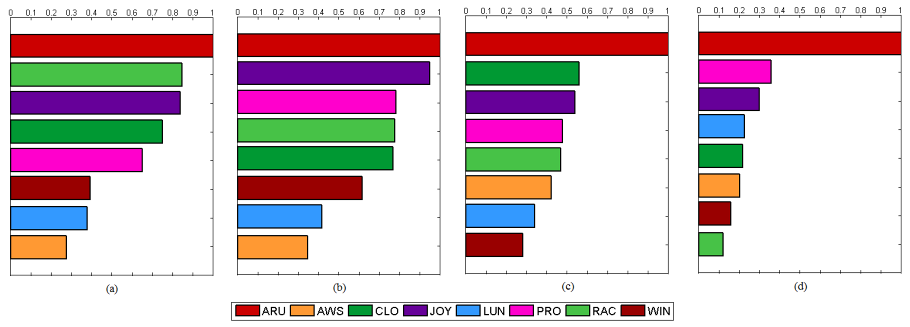

Figure 2.13: Comparison of figures of merit techniques for *s*-VM size. (a) Radar plot (b) Simple figure of merit, A = 0 and B = 1 (c) Figure of merit based on AHP (d) Performance-price ratio based on AHP performance values. The values of the four figures have been normalized for ease of comparison.

Chapter 3

An exact approach for optimizing placement in cloud brokering

Introduction

In the near future, cloud brokers may become the online travel planning companies for cloud computing. Several years ago, the travel industry was in the same situation as the current cloud computing industry. Travelers left planning in the hands of travel agencies who had agreements with airline companies and hotels in order to obtain reduced overall prices. This situation changed with the advent of travel planning websites that provide instant online comparisons for millions of flights on thousands of airlines. Similarly, the current service typically provided by consulting companies in helping end-users with the placement of cloud infrastructure may, in the near future, be provided by cloud brokers. Cloud brokers could provide the service of discovery and comparison of cloud provider offers and even the automatic and optimal placement of cloud resources.

Placement in cloud brokering refers to the techniques used to efficiently distribute infrastructure resources across multiple cloud providers (*c.f.*

Section "Placement in cloud brokering"). Cloud brokers may react to new situations when conditions change, dynamically repositioning or deploying new cloud infrastructure in order to maintain the performance of end-users' applications. Examples of scenarios in which a cloud broker may trigger placement algorithms in order to compute a new infrastructure topology include:

- *Changes in cloud market conditions*: For example, introduction of new VM configurations, change in prices, apparition of a new cloud provider or implementation of a new pricing model. In this scenario, a cloud broker could determine the impact of the changes of market conditions on the economies or performance of end-users applications. In the case of a positive impact, the end-user can be advised to migrate its cloud infrastructure.
- *Unexpected changes in cloud infrastructure*: Outages may strongly impact economies of end-users' running cloud applications. Although cloud providers offer economic compensation to end-users having experienced an outage, in most cases, this compensation is negligible in comparison to the impact of having a service unavailable (*e.g.* an e-commerce website down). Thus, cloud brokers may not only automatically redeploy infrastructure in recovery scenarios but also minimize the time an application is inaccessible.

This chapter is organized as follows. Goal programming, a technique to solve Multiple Criteria Decision Making (MCDM) problems is briefly described in Section "Goal programming". An exact approach for optimizing placement in cloud brokering is presented in Section "An exact approach for the placement problem". A case study considering an online trading platform is presented in Section "Case study: Online trading plataform".

Goal programming

The optimization goal in MCDM problems is to find an efficient (but not necessarily an optimum) solution by considering multiple objectives (or goals) that can possibly conflict with each other. Thus, MCDM problems contrast with Linear Programming (LP) problems which optimizes a single linear objective. Here, goal programming as a technique for solving MCDM problems has been considered. Goal programming is usually carried out using either the *weighted* or the *preemptive* method.

The weighted method transforms a MCDM problem into a standard LP. A weighted objective function corresponds to a weighted sum of functions representing the multiple objectives of the problem. The weight determines the priority of each objective. Although the computation of the weighted method is easy, there are drawbacks:

- Weighting is subjective and may result in under- or over-rating the contribution of objectives.
- The objectives may be expressed in different metrics or different orders of magnitude that prevent a straightforward calculation of the objective's function.

The preemptive method considers a MCDM problem as a set of multiple LPs with different priorities assigned by the end-user. Thus, each LP is optimized one at a time from the highest to the lowest priority (Figure 3.1). Between LP executions, the optimum value is added as a constraint to the successive LP model. This guarantees that the optimum value of a higher priority objective is not degraded by a lower priority objective. This process continues until the lowest priority is optimized. In cases where a limited amount of degradation is acceptable, a constraint is added as an inequality that allows higher priority solutions to be in the near region for the optimal solution.

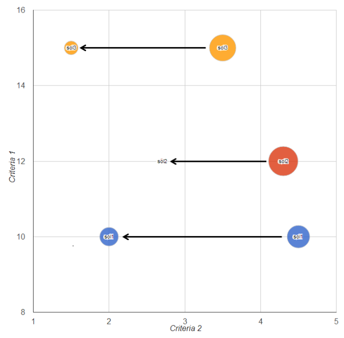

Figure 3.1: Preemptive method. Optimization priority: Criterion 1, Criterion 2, Criterion 3 (represented by the size of the spot).

An exact approach for the Placement problem

The cloud VM placement problem in cloud brokering can be represented as a constrained Knapsack problem: Given a set of VMs, each with a configuration, price and performance, determine the number of each VM configuration to provide so that the provisioning infrastructure is more than or equal to the end-user request (*i.e.* request is satisfied) whilst providing the most cost effective solution. The cloud placement problem is formulated as a Mixed-Integer Linear Programming (MILP) problem and the preemptive goal programming analysis is performed to solve this problem.

Parameters

- *End-user request parameters:*
  - $reqCPU$: number of vCPUs.
  - $reqMEM$: memory capacity.
  - $reqSTO$: storage capacity.
  - $reqNET$: network capacity.
  - $reqRTT$: average latency between the cloud providers and the application customers.
  - $reqAVA$: average availability.
  - $reqREL$: average reliability.
  - $reqSCA$: average VM provisioning time.
  - $reqVAR$: average variability.
  - $reqPER$: performance required by the end-user.
  - $LOCmax_k$: maximum percentage of resources that can be allocated to cloud provider k.
  - $VMmax$: maximum number of VMs.
  - *Pricing model*: on-demand, reserved or spot.

- *Cloud provider parameters:*
  - V: j-by-k matrix composed of v_{jk} elements. Where $v_{jk} = 1$ if and only if the VM configuration j exists at the cloud provider k, otherwise $v_{jk} = 0$.
  - CPU_{jk}: the number of vCPUs of the VM configuration v_{jk}.
  - MEM_{jk}: the memory capacity of the VM configuration v_{jk}.
  - STO_{jk}: the storage capacity of the VM configuration v_{jk}.
  - NET_{jk}: the bandwidth capacity of the VM configuration v_{jk}.
  - $Price_{jk}$: the price per unity of time for running a VM configuration of type v_{jk}.
- *Cloud broker parameters:* Parameters measured or calculated by the cloud broker.
  - p: index of the smallest VM configuration (in terms of computing, memory and storage).
  - η_k: the number of VMs of type p that guarantees the fulfillment of the request for the cloud provider k.

$$\forall k \in [1, K] : \\ \eta_k = max\left(\left\lceil \frac{reqCPU}{CPU_{pk}} \right\rceil, \left\lceil \frac{reqMEM}{MEM_{pk}} \right\rceil, \left\lceil \frac{reqSTO}{STO_{pk}} \right\rceil\right) \tag{3.1}$$

  - N: Taking into consideration the possibility of cloud providers having unlimited resources, N is an upper bound that limits the set of solutions. Thus, N represents the maximum number of any kind of VM configuration used to fulfill the request. This parameter is set in case of the $VMmax$ is not specified by the end-user.

$$N = max(\eta_k) \tag{3.2}$$

- RTT_k: the latency of the cloud provider k.
- α_k : average availability of a cloud provider k.
- β_k : average reliability of a cloud provider k.
- γ_{jk}: average time to provision a VM of type j at cloud provider k.
- cv_{jk}: average variability of a VM of type j at cloud provider k.
- $Performance_{jk}$: the performance of a VM configuration of type v_{jk}.

Variables

- *Binary variables:*
  - $x_{jk}^n = 1$: If and only if the VM n of type j is used and belongs to the cloud provider k, otherwise $x_{jk}^n = 0$.
- *Real variables:*
  - TCC: Total Computing Capacity. Amount of computing capacity for a particular solution.

$$TCC = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} CPU_{jk} x_{jk}^n \qquad (3.3)$$

– *TMC*: Total Memory Capacity. Amount of memory capacity for a particular solution.

$$TMC = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} MEM_{jk}x_{jk}^{n} \tag{3.4}$$

– *TSC*: Total Storage Capacity. Amount of storage capacity for a particular solution.

$$TSC = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} STO_{jk}x_{jk}^{n} \tag{3.5}$$

– *TVM*: Total VMs with a particular solution.

$$TVM = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} x_{jk}^{n} \tag{3.6}$$

– *TDC*: Total Deployment Cost. Total cost for deploying an infrastructure across multiple cloud providers.

$$TDC = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} Price_{jk}x_{jk}^{n} \tag{3.7}$$

– *TP*: Total performance of a particular solution.

$$TP = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} Performance_{jk}x_{jk}^{n} \tag{3.8}$$

– *TDT*: Total Deployment Time. Total time for deploying an infrastructure across multiple cloud providers.

$$TDT = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} \gamma_{jk} x_{jk}^{n} \tag{3.9}$$

– *TV*: Total Variability. Total variability for a particular solution.

$$TV = \sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N} cv_{jk} x_{jk}^{n} \tag{3.10}$$

Goal

Preemptive goal programming analysis is performed to solve the MILP depending on the scenario. For instance, in a disaster recovery scenario the criteria are prioritized as follows:

1. Minimize the real variable *TDT*.
2. Minimize the real variable *TDC* constrained by the minimal deployment time previously obtained.
3. Maximize the real variable *TP* constrained by the minimal deployment time and the total deployment cost previously obtained.

This method guarantees a minimal deployment time and suboptimal cost and performance values for the infrastructure to be provisioned.

Constraints

The constraints associated are the following:

- *Physical constraint*: This constraint guarantees VMs are allocated to an existing VM configuration.

$$x_{jk}^{n} \leq v_{jk} \tag{3.11}$$

- *VM configuration constraints*:

$$TCC \geq reqCPU \tag{3.12}$$

$$TMC \geq reqMEM \tag{3.13}$$

$$TSC \geq reqHD \tag{3.14}$$

$$VMmax \geq TVM \tag{3.15}$$

- *Load Balancing constraints*:

$$\begin{gathered} \forall k \in [1, K] : \\ \sum_{j=1}^{J} \sum_{n=1}^{N} CPU_{jk} x_{jk}^{n} \leq LOCmax_{k} \cdot TCC \end{gathered} \tag{3.16}$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N} MEM_{jk}x_{jk}^{n} \leq LOCmax_k \cdot TMC \tag{3.17}$$

$$\sum_{j=1}^{J}\sum_{n=1}^{N} HD_{jk}x_{jk}^{n} \leq LOCmax_k \cdot TSC \tag{3.18}$$

- *Availability and reliability constraint*:

$$\begin{gathered} \forall k \in [1, K] : y_k = \sum_{j=1}^{J}\sum_{n=1}^{N} x_{jk}^{n} \Rightarrow \\ \sum_{k=1}^{K} \alpha_k \cdot y_k \leq reqAVA \cdot \sum_{k=1}^{K} y_k \\ \sum_{k=1}^{K} \beta_k \cdot y_k \leq reqREL \cdot \sum_{k=1}^{K} y_k \end{gathered} \tag{3.19}$$

- *Latency constraint*:

$$\sum_{k=1}^{K} RTT_k \cdot \sum_{j=1}^{J}\sum_{n=1}^{N} x_{jk}^{n} \leq reqRTT \cdot TVM \tag{3.20}$$

- *Scalability constraint*:

$$\sum_{j=1}^{J}\sum_{k=1}^{K}\gamma_{jk}\cdot\sum_{n=1}^{N}x_{jk}^{n}\leq reqSCA\cdot TVM \tag{3.21}$$

- *Variability constraint*:

$$\sum_{j=1}^{J}\sum_{k=1}^{K}cv_{jk}\cdot\sum_{n=1}^{N}x_{jk}^{n}\leq reqVAR\cdot TVM \tag{3.22}$$

Case study: Online trading platform

Bezimie is a London based company that provides an online trading platform. Using Bezimie's application, traders can purchase and sell stocks and currencies easily through its web interface. The main competitive advantage of Bezimie regarding other online trading platforms is its low latency[1] when placing market orders or reporting quote prices for stocks and currencies to traders. Bezimie manages its cloud infrastructure with the help of the CompatibleOne cloud broker. The current cloud infrastructure topology of Bezimie consists of multiple VMs deployed across two providers (Amazon and Rackspace) with datacenters in Ireland and England. The cloud providers have been chosen due to their proximity to Bezimie's clients, resulting in a low average latency ideal for online trading (Table 3.1).

[1]The network latency is particularly important in financial instruments with a high price variation (volatility). Even a price variation of just a few cents may represent large amounts of money when trading in high volume. Moreover, higher latency connections are more prone to packet delivery delays and loss.

Bezimie is planning to expand its portfolio of clients to France. After some tests, Bezimie's IT department notes that the latency of French traders to its UK-based cloud infrastructure is acceptable ($\leq$ 125 ms) for online trading but greater than the latency of Zimie ($\leq$ 110 ms) its French counterpart and direct competitor. Therefore, in order to be competitive in the French market, Bezimie's IT department compares different solutions to serve its French traders with help of the CompatibleOne broker. The first optimization priority is to minimize the latency between cloud providers and traders; the second priority is to minimize the cost of the requested infrastructure; the third is to maximize the performance of the VMs acquired in the future. Bezimie's IT department chooses the best performing solution (19.6) with the lowest latency in France ($\leq$ 82 ms) at the lowest cost (2.5 US$ per hour) for serving its French traders (Figure 3.2). However, this solution places all the infrastructure serving French traders into one cloud provider (ArubaCloud). This represents a serious risk in terms of of cloud service outages.

Cloud provider	RTT to England (ms)	RTT to France (ms)
ARU	127	82
AWS	95	112
CLO	135	105
JOY	105	101
LUN	110	91
PRO	130	110
RAC	85	97
WIN	122	123

Table 3.1: RTT from cloud providers to current and future Bezimie's client portfolio

Bezimie's IT department also simulates disaster recovery scenarios through the CompatibleOne cloud broker. In disaster scenarios, the main priority for Bezimie is to minimize the time the service is offline while keeping an acceptable latency. The second priority is to minimize the cost of the required infrastructure. The solutions for provisioning time optimization in case of an ArubaCloud outage are presented in Figure 3.3. The figure shows the two optimization stages for different LOCmax values. Solutions resulting from the first optimization stage (minimization of the provisioning time) are shown on the right and solutions resulting from the cost optimization stage are shown on the left. Note that cost optimization brings cheaper but higher latency and more variable solutions.

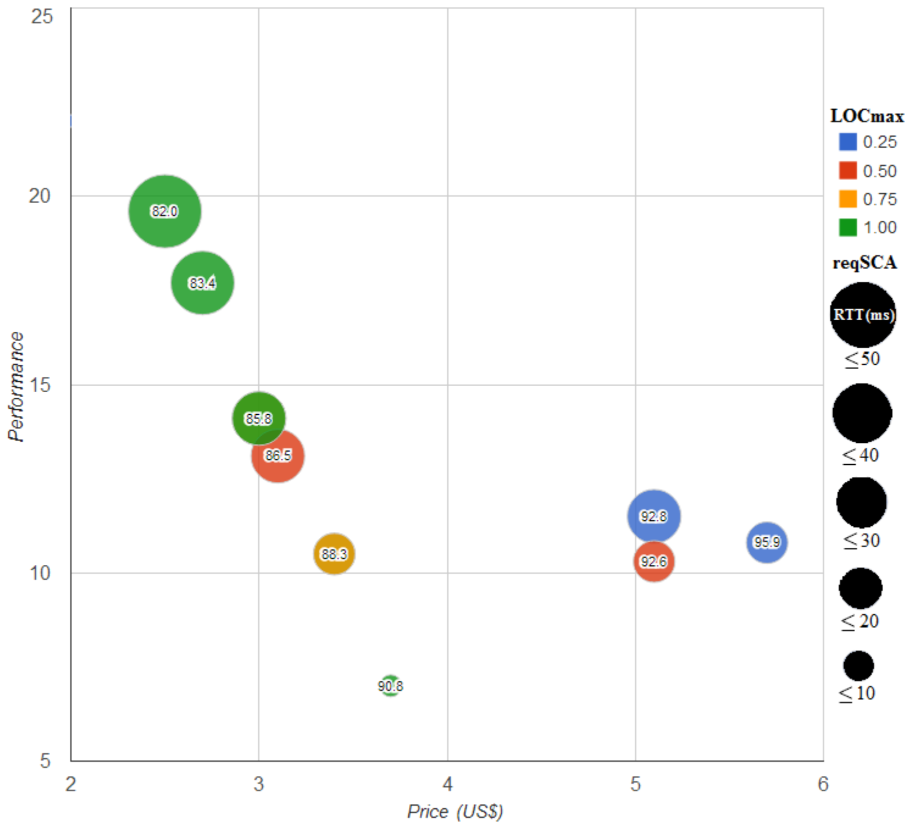

Figure 3.2: Preemptive optimization solutions for latency. The size of the spot represents the average provisioning time in seconds (reqSCA). The figure of merit used here is the same as the one obtained in the previous case study (*c.f.* Section 2). Parameters: reqCPU = 80, reqMEM = 60, reqSTO = 300, reqRTT $\leq$ 110ms.

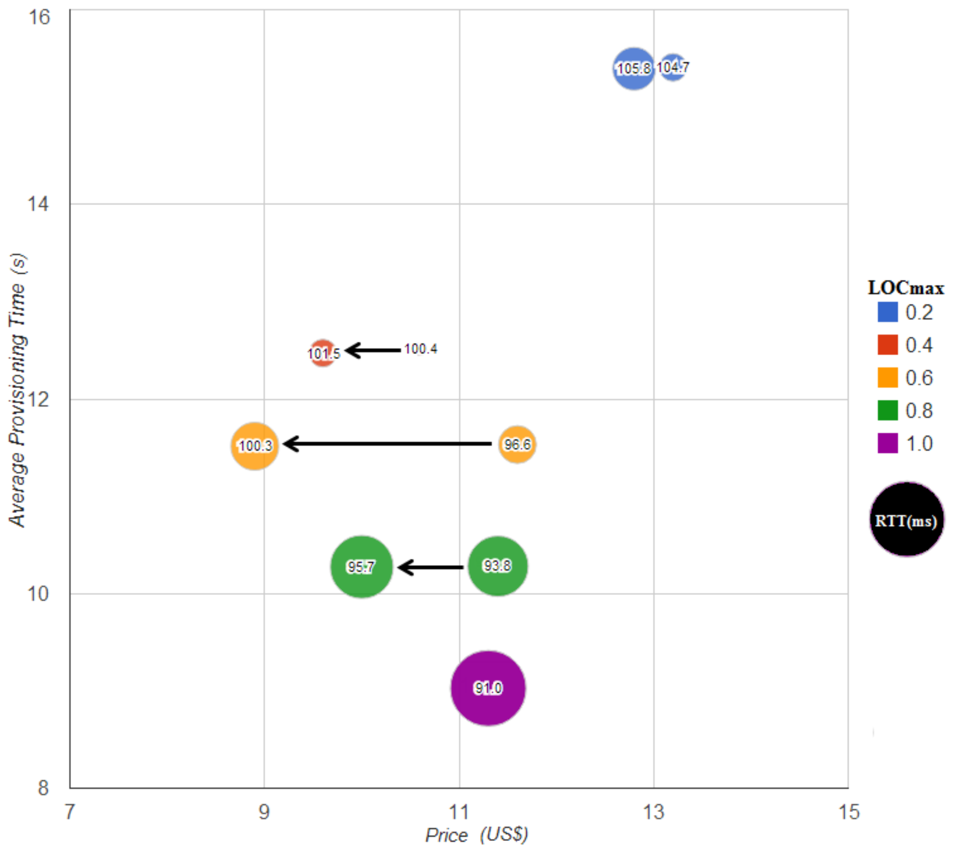

Figure 3.3: Preemptive optimization solutions for provisioning time. The size of the spots represents the solution variability. Parameters: reqPER = 20, reqRTT $\leq$ 110ms.