UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base

Corso di Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XXVIII Ciclo
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione (DIETI)

NETWORK MONITORING IN PUBLIC CLOUDS: ISSUES, METHODOLOGIES, AND APPLICATIONS

Valerio Persico

Ph.D. Thesis

Tutor
Prof. Antonio Pescapé

Coordinator
Prof. Francesco Garofalo

March 2016

To my family.

Abstract

Cloud computing adoption is rapidly growing thanks to the carried large technical and economical advantages. Its effects can be observed also looking at the fast increase of cloud traffic: in accordance with recent forecasts, more than 75% of the overall datacenter traffic will be cloud traffic by 2018. Accordingly, huge investments have been made by providers in network infrastructures. Networks of geographically distributed datacenters have been built, which require efficient and accurate monitoring activities to be operated. However, providers rarely expose information about the state of cloud networks or their design, and seldom make promises about their performance. In this scenario, cloud customers therefore have to cope with performance unpredictability in spite of the primary role played by the network. Indeed, according to the deployment practices adopted and the functional separation of the application layers often implemented, the network heavily influences the performance of the cloud services, also impacting costs and revenues.

In this thesis cloud networks are investigated enforcing *non-cooperative* approaches, i.e. that do not require access to any information restricted to entities involved in the cloud service provision. A platform to monitor cloud networks from the point of view of the customer is presented. Such a platform enables general customers—even those with limited expertise in the configuration and the management of cloud resources—to obtain valuable information about the state of the cloud network, according to a set of factors under their control. A detailed characterization of the cloud network and of its performance is provided, thanks to extensive experimentations performed during the last years on the infrastructures of the two leading cloud providers (Amazon Web Services and Microsoft Azure).

The information base gathered by enforcing the proposed approaches allows customers to better understand the characteristics of these complex network infrastructures. Moreover, experimental results are also useful to the provider for understanding the quality of service perceived by customers. By properly interpreting the obtained results, usage guidelines can be devised which allow to enhance the achievable performance and reduce costs. As a particular case study, the thesis also shows how monitoring information can be leveraged by the customer to implement convenient mechanisms to scale cloud resources without any *a priori* knowledge.

More in general, we believe that this thesis provides a better-defined picture of the characteristics of the complex cloud network infrastructures, also providing the scientific community with useful tools for characterizing them in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we present the scenario in which this study is conducted, also introducing the basic definitions and concepts adopted in the thesis. Afterwards, we outline the motivations stimulating our research work. The ending part of the chapter outlines the contribution and the organization of the thesis.

1.1 The emergence of the cloud model

1.1.1 Definitions and concepts

The idea behind cloud computing is not a new one, as computing facilities were envisioned to be provided as a utility to the general public already in 1960s [1]. However, the term *cloud* started gaining popularity from 2006, when Google's former CEO Eric Schmidt used it to describe services delivered across the Internet [2]. The definition of cloud computing commonly accepted today has been published in 2011 by the American National Institute of Standards and Technologies (NIST) [3]. According to this definition

> *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".*

This *shared pool of configurable resources* is commonly referred to as the *cloud*.

Cloud computing is envisioned as the new frontier of the Internet era. Indeed, thanks to the rapid development of processing and storage technologies and to the success of

the Internet, computing resources have become more and more cheaper and ubiquitously available than before [2]. This technological trend has enabled the transformation of such computing model—only hypothesized until few years before—into a commercial reality. The evolution of cloud computing over the past few years is potentially one of the major advances in the history of computing [4].

The emergence of cloud computing has tremendously impacted the Information and Communication Technology (ICT) industry, where large companies—such as Amazon, Microsoft, and Google—strive to provide more powerful, reliable, and cost-efficient cloud platforms, and business enterprises seek to reshape their business models to gain benefit from this new paradigm. Cloud computing is a general purpose technology that can provide a fundamental contribution to promote growth and competition, also helping the economy to recover from a severe downturn as the current one [5]. Indeed, the adoption of the cloud paradigm is able to drastically reduce the fixed costs of entry and production. It turns part of them into variable costs related to the production necessities, thus positively impacting competition in all sectors where fixed ICT spending is crucial.

Several important classes of existing applications are becoming even more compelling with cloud computing and contribute further to its momentum [6, 7, 8]. Interesting examples encompass:

- *mobile interactive applications*, (e.g., services accessible from energy-constrained devices with limited computational capabilities that respond in real time to information provided by both human and non-human data sources); common practices enabled by the cloud paradigm—such as offloading consuming tasks to the cloud—allow to save energy and enhance service performance;

- *parallel batch processing* (e.g., analytics aimed at decision support); thanks to cloud's cost associativity (that allows customers to use hundreds of computers for a short time) and programming abstractions such as MapReduce [9] or Hadoop [10], cloud computing presents a unique opportunity for batch-processing and analytics jobs that analyze terabytes of data; without the cloud paradigm, these jobs either would have taken hours to finish or would have needed substantial expenditures to acquire and maintain dedicated infrastructures to perform processing in acceptable time;

- *on-demand storage* (e.g., unstructured data buckets or database services); cloud storage systems provide customers with the ability to store seemingly limitless

amounts of data for any duration of time; customers have access to their data from anywhere at any time and only pay for what they use and store; moreover, data is durably stored using both local and geographic replication to facilitate disaster recovery.

Essential characteristics. According to its standard definition [3], the cloud computing model is supposed to have essential characteristics, such as: (i) *on-demand self service*, (ii) *broad network access*, (iii) *resource pooling*, (iv) *rapid elasticity*, and (v) *measured service*.

These characteristics guarantee that a consumer can unilaterally provision computing capabilities (e.g., server time, network storage, broadband access) as needed, automatically, and without requiring human interaction with service providers. These capabilities are available over the network and can be accessed through mechanisms that promote heterogeneous client platforms. The pooled resources composing the cloud (e.g., storage, processing, memory, and bandwidth) allow to serve multiple consumers (multi-tenant model), with resource assignment that follows consumers' demand. Furthermore, dynamic resource assignment gives to the customers the illusion of infinite resources, able to scale rapidly outward or inward with demand. It is worth noting how this characteristic also generates a sense of location independence: the customer has no control or knowledge over the precise location of the provided resources, but is allowed to access to them only at a higher level of abstraction (e.g., geographic region, or even country). Finally, cloud systems leverage metering capabilities at different levels of abstraction, in order to both automatically control resources and implement pay-per-use billing models.

Deployment models. Cloud systems can be deployed according three main different models: (i) *private cloud*, (ii) *community cloud*, and (iii) *public cloud*.

While for private clouds, the infrastructure is provisioned for the exclusive use by a single organization, the community cloud is provisioned for the use of a community of users with shared concerns. Both may be owned, managed, and operated by the organization (one of the community in the case of community clouds) or by a third party, and may exist on or off premises. Finally, the infrastructure of a public cloud exists on the premises of cloud providers and is previsioned for the use of the generic public. It may be owned, managed, and operated by a business, academic, or government organization. The standard [3] also considers the existence of a fourth deployment model—i.e., the *hybrid*

cloud—that is the composition of two or more cloud infrastructures (private, community, or public) bound together by data and application portability, but still remaining unique entities.

Involved entities. According to the standards roadmap provided by the NIST [11] five major entities that perform tasks related to cloud computing can be introduced: (i) *the cloud provider*, (ii) *the cloud consumer*, (iii) *the cloud carrier*, (iv) *the cloud auditor*, and (v) *the cloud broker*.

The cloud provider is the entity responsible for making a service available to cloud consumers. Cloud providers build the requested services, manage the technical infrastructure required for providing the services, provision the services at agreed-upon service levels, and protect their security and privacy. They are in charge of deploying, orchestrating, managing the provided services, also guaranteeing privacy and security.

The cloud consumer represents a person (or an organization) that uses the service from a cloud provider and maintains a business relationship with it. He/she requests appropriate cloud services, sets up service contracts, uses the services, and is billed for the services provisioned. The cloud consumer may also act as a service provider, as he/she can utilize leased resources in order to setup new services accessible to *final users* that may have no business relationship with the cloud provider. In the following, when we want to emphasize the business relationship existing between the consumer and the provider, we will refer to the former as *cloud customer*.

A cloud carrier acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers. Cloud carriers provide access to consumers (and final users) through network, telecommunication, and other access devices. The distribution of cloud services is normally provided by network and telecommunication carriers. According to the standard, carriers should also include transport agents, i.e. business organizations that provide physical transport of storage media such as high-capacity hard drives.

A cloud auditor is a party that can conduct independent assessments of cloud services. The auditor can evaluate the services provided by a cloud provider in terms of performance, security controls, or privacy impact.

A cloud broker is an entity that manages use, performance, and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers. Indeed,

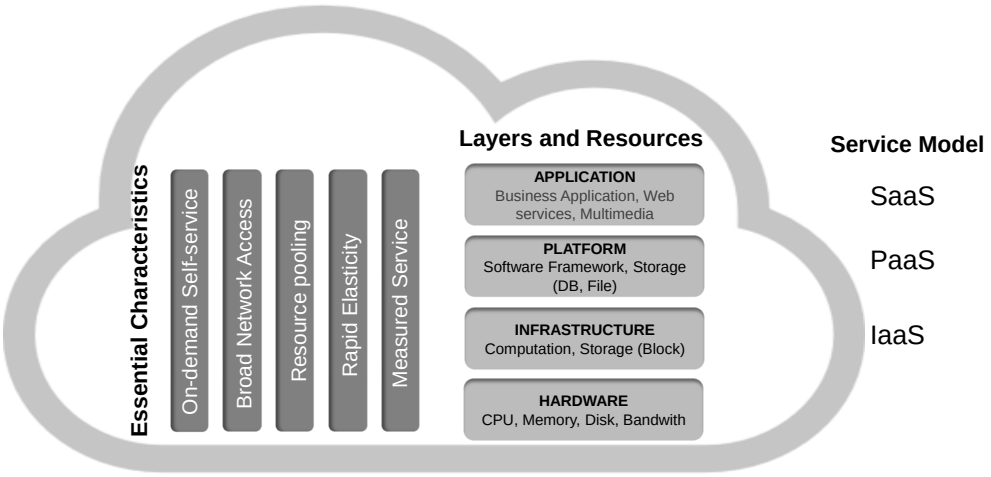


Figure 1.1: Layered cloud architecture and service models.

as cloud computing evolves, the integration of cloud services can be too complex for cloud consumers to manage: in this case, a cloud consumer may request cloud services from a cloud broker, instead of directly contacting a cloud provider. In more details, a cloud broker can provide services in three categories: (i) *service intermediation* (i.e., enhancing a given service by improving some specific capability and providing value-added services to cloud consumers, e.g., managing access to cloud services, identity management, or performance reporting, enhanced security); (ii) *service aggregation* (i.e., combining and integrating multiple services into one or more new services e.g., by providing data integration and ensuring the secure data movement between the cloud consumer and multiple cloud providers); (iii) *service arbitrage* (similar to service aggregation except that the services being aggregated are not fixed).

A number of different interactions may exist among these entities. For instance, a cloud consumer may request cloud services from a cloud provider directly or via a cloud broker. A cloud auditor conducts independent audits and may contact the others (e.g., the carrier) to collect necessary information.

Cloud architecture and service models. Independently of the deployment model adopted, the architecture of the cloud environment can be seen as separated into four different layers: (i) *the hardware layer*, (ii) *the infrastructure layer*, (iii) *the platform layer*, and (iv) *the software layer*.

The hardware layer, typically implemented in datacenters, is responsible of managing cloud physical resources (i.e., power and cooling systems, physical servers, switches and routers). As a datacenter is typically composed of thousands of servers organized in racks (interconnected through switches, routers, and other fabrics) typical issues at this layer include the configuration of the hardware, or power and traffic management.

The infrastructure layer, also known as the virtualization layer, is in charge of pooling storage and computing resources, by partitioning the physical resources. To this aim virtualization technologies—such as Xen [12], KVM [13], and VMware [14]—are often adopted, allowing to run multiple virtual machines (VMs) over the same physical server. Thanks to the adopted technologies this layer is able to implement essential features, such as the ability of dynamically assigning resources.

The platform layer consists of operating system and application frameworks (e.g., programming language execution environments, databases, web servers). Its purpose is to minimize the burden of deploying cloud applications to the consumers.

The application layer consists of the actual cloud application, which can take advantage of typical cloud features, such as automatic scaling to achieve better performance, increase availability and reduce costs.

Each layer is loosely coupled with the layers above and below. This allows each layer to evolve independently from the others. This property also guarantees to increase the modularity of the architecture, thus enabling to support a wide range of applications without sacrificing ease of management or ease of maintenance. Generally speaking, cloud computing employs a service-driven business model [2]. Indeed, every layer of the architecture presented can be implemented as a service to the layer above and acts as a consumer of the layer below.

According to a commonly accepted partition, cloud services can be grouped in three different categories [3]: (i) *Infrastructure as a Service (IaaS)*, (ii) *Platform as a Service (PaaS)*, (iii) *Software as a Service (SaaS)*.

IaaS allows the consumer to be provisioned with processing, storage, networks, and other fundamental computing resources. The consumer is able to deploy and run arbitrary operating systems and applications. However, according to the layered model, the consumer does not manage or control the underlying cloud infrastructure and sometimes has possibly limited control over networking components (e.g., host firewalls).

PaaS gives to the consumer the capability to deploy onto the cloud infrastructure applications acquired or directly created by the consumer using programming languages, libraries, services, and tools supported by the provider. When accessing the cloud infrastructure at this layer, the consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Finally SaaS offers the ability to leverage provider's applications running on the cloud infrastructure managed by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific configuration settings.

Figure 1.1 summarizes the key concepts about cloud architecture and service models.

1.1.2 Public-cloud market

Public cloud adoption is growing faster than private cloud one. Indeed, it is possible to register a greater adoption of public cloud resources, especially with strengthening of security aspects, as the business sensitivity to costs associated with dedicated ICT resources grows along with demand for agility [15]. Worldwide spending on public-cloud services will grow at a 19.4% Compounded Average Growth Rate (CAGR)—almost six times the rate of overall ICT spending growth—from nearly $70 billion in 2015 to more than $141 billion in 2019 [16]. This growth is primary driven by large and very large companies. Small and medium business will also significantly contribute however, as 40% of the worldwide total will come from companies with less than 500 employees [16]. An increasing number of organizations now also run mission-critical business applications on cloud, indeed: a significant portion of them is migrating most or all of their infrastructure to cloud IaaS, to avoid major capital expenditure, such as a hardware refresh or the construction of a datacenter [17].

In more details, according to latest reports about public IaaS cloud computing, the market is dominated by only a few global providers among the huge number of offers. While there is a broad (and still increasing) number of cloud suppliers, most customers are settling on just four providers: Amazon Web Services, Microsoft, IBM, and Google. As of February 2016, together those four represent 51% of the total cloud market [18]. Studies

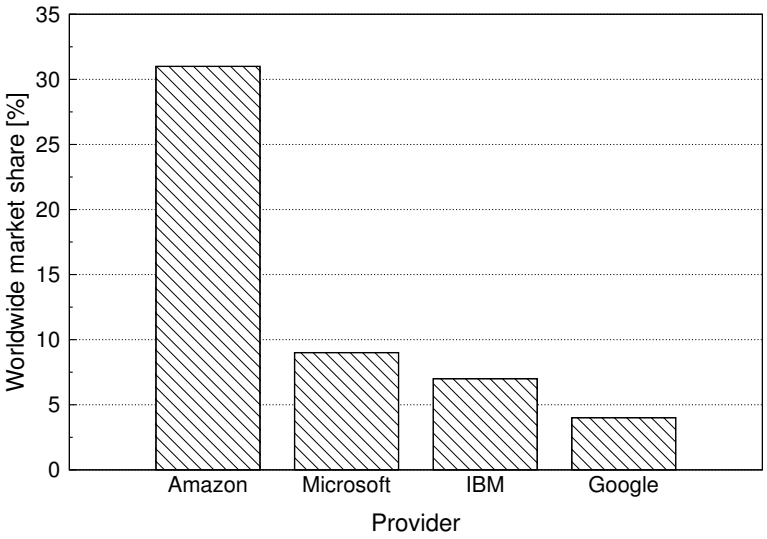


Figure 1.2: Cloud services market share—Q4 2015. Data source: [18].

show that the cloud market is quite clearly bifurcating with a widening gap between the big four cloud providers and the rest of the service provider community: developing and maintaining the necessary global scale datacenter infrastructure, along with the required marketing and operations support, is simply beyond the reach of all but a very small number of players. Of the 51% share of the market that they hold, Amazon, Microsoft, IBM, and Google represent the 31%, 9%, 7%, and 4%, respectively. So Microsoft, IBM, and Google combine for 20% of the market, compared to Amazon's 31% (see Figure 1.2).

Market share has therefore continued to become more heavily concentrated, although the market has dramatically grown. In spite of the number of providers offering cloud services the market is dominated by only a few global providers—most notably Amazon, but increasingly also Microsoft: these two providers comprise the majority of workloads running in public cloud IaaS in 2015 [17]. Amazon Web Services is the clear market leader—with over a million active customers in more than 190 countries [19]—while Microsoft is the only clear challenger, also due to the continual investments in the latest infrastructure technologies. Both providers are steadily expanding their global infrastructure whose growth is backed by billion investments: infrastructural expansion is claimed to be a priority because of the direct benefits generated for the customers.

1.1.3 Main advantages and obstacles to adoption

Cloud computing has a great impact on business thinking. It facilitates a change in the way companies operate, as it enables them to react faster to business needs while driving greater operational efficiencies. The cloud model however, introduces a non-negligible set of issues, which proved to potentially limit it widespread adoption. In this section, both the advantages carried by the cloud model and the issues raised by its adoption will be briefly discussed.

The emergence of the cloud computing model and its popularity is motivated by a number of both technical and economical peculiarities that carry advantages for both the provider and the consumer. One of the most notable benefit is the improvement of efficiency and the optimization of hardware and software resources utilization; for instance, in cloud datacenters workloads can be distributed at a higher density on VMs thanks to virtualization properties; active VMs can be also clustered, and migrated onto a limited set of running physical servers; this allows to reduce the energy consumption for hardware and network infrastructure [20].

Cloud computing can enable more energy-efficient use of computing power also positively impacting the environment [21, 22]. The average amount of energy needed for a computational action carried out in the cloud is far less than the average amount for an on-site deployment. This is because different organizations can utilize the same physical resources, leading to a more efficient use of the shared resources. Providers claim they are designing for energy-efficient performance with platforms that can support usages and applications with dramatically decreased energy consumption, as their goal is to reduce the environmental impact of their operations while continuing to meet high-performance requirements for computing [23, 24].

Virtualization gives the tenants the illusion of a dedicated infrastructure with unlimited resources, also guaranteeing security and fault isolation. The on-demand service schema together with resource elasticity allows consumers to lease resources at runtime, with provisioning time of minutes rather than weeks, adapting them to their actual needs. Common practices also enhance robustness against disasters—also to consumers with reduced expertise and cash—as data can be easily duplicated across multiple geographic sites.

Resources are accessible through the public Internet, thus enabling ubiquitous access to them. This allows resources to be utilized from anywhere and any device (e.g., resource-

constrained mobile devices), enhancing the flexibility degree of the applications, also enabling collaboration scenarios harder to implement before.

These technical advantages reflect economical benefits to the cloud consumer. The economic appeal of the cloud model is often described as "*converting capital expenses into operating expenses*". Indeed the absence of up-front capital expenses allows capital to be redirected to core business investments. Thanks to resource elasticity, customers are able to increase resource utilization, thus increasing efficiency. Moreover, the customers are usually relieved from the complexity and the cost associated to maintenance issues, such as updating software on the servers or reconfiguring the network.

Common use cases in which advantages carried by the cloud model are evident, encompass the ones reported in the following. The cloud model helps the customer design and deploy services whose demand is unknown in advance: (e.g., a startup may need to support a spike in demand when it becomes popular, followed by a potential reduction in demand). Another common case is related to the deployment of services whose demand varies with time. Indeed, provisioning a system to sustain the peak load exhibited few days per month, leads to underutilization other times. The cloud model allows an organization to pay by the hour for resources, and may lead to cost savings also when the hourly cost for renting a resource is higher than the cost to own one. Finally, cost associativity, leads to perform faster batch analytics, by using hundreds of machines for one hour, instead a single machine for hundreds of hours.

However these on-demand easily-usable features provided by cloud providers come at a cost. Indeed, the advantages carried by the high level management interface leveraged by consumers hide newly introduced challenges, that have proved to be non-trivial obstacles to the adoption of the cloud paradigm in several application fields. From the one hand, managing services from a higher level of abstraction (e.g., by deploying VMs, application frameworks, and software containers) relieves the consumer from a number of management issues. On the other hand, implementation details are often kept hidden, causing troubleshooting, system assessment, and anomaly management to be more complex than in other contexts, due to the limited visibility over the system and its characteritics.

In more details, a number of issues are commonly identified as the main obstacles to cloud adoption, and define stimulating research tracks [25]. Despite the attention paid by cloud providers, performance unpredictability of cloud system is a major issue in cloud computing. This is particularly true for applications that support critical services,

and ones that have to provide service-level agreements to final users. Virtualization represents a flexible and cost-effective way to share physical resources (such as processors and I/O interfaces among multiples VMs) and proved to impact the computation and communication performance of cloud services [26]. Nevertheless, commercial providers typically base their Service Level Agreements (SLAs) on the availability they offer only. In addition, very few studies have been performed to understand the performance of large scale complex cloud systems. Cloud providers usually keep system design information such as network topology confidential (especially in the case of public clouds), and rarely unveil it for security and commercial reasons [27, 28]. This exacerbates the problem, making the investigation of performance figures harder. As a consequence, application deployment and its optimizations are forced to ignore detailed performance information.

Also service availability is considered a common concern, indeed. The only solution to very high availability supposed to be plausible is relying on multiple cloud computing providers [25]. But this usually requires to turn up to brokering services or additional management overhead. Unfortunately, consumers are not able to natively control and manage heterogeneous cloud resources (i.e., provided by different providers) in an easy way, due to the absence of standard management APIs.

The fact that applications continue to become more and more data intensive (needing huge amount of data to run, or producing it) generates a set of concerns both related to the privacy of sensitive data, and to their management. Although software stacks have improved interoperability across platforms, APIs for cloud computing are still essentially proprietary. Consequently, consumers cannot easily extract data from their applications. This limitation in moving data from site to another is preventing organizations from adopting cloud computing: data lock-in makes consumers vulnerable to both price increase and providers going out of business.

Another problem strictly associated to the former is related to the transfer of these data to and from the cloud, and among geographically distributed cloud sites. Huge investments are made by providers in network infrastructure, in order to support the dramatically changing demand produced by the on-demand resource adoption. Because of the huge volume of data involved, networks represent the bottleneck to data transfer, and due to the non-negligible cloud transfer costs, cloud users and cloud providers have to think about the implications of placement and traffic at every level of the system if they want to minimize costs. This kind of reasoning can be seen in the continuous development

Advantages
• cost efficiency
• energy efficiency
• illusion of dedicated infrastrutures
• ease of infrastructure management
• ubiquity of access to resources
• flexibility
• scalability

(a)

Obstacles to adoption
• lack of visibility into implementation details
• performance unpredictability
• data-transfer cost/performance
• absence of standardized APIs
• potential provider lock-in
• difficulties in troubleshooting
• data privacy concerns

(b)

Table 1.1: Cloud Computing: main advantages (Table 1.1a) and obstacles to adoption (Table 1.1b).

of new cloud sites by cloud providers, and their distribution all over the globe.

Final remarks. The emergence of cloud computing constitutes a fundamental change in the way ICT services are designed, developed, deployed, scaled, maintained, and paid for. Since its practical emergence, cloud computing has rapidly become a widely adopted model and it is more and more used to deliver services over the Internet thanks to both the technical and the economical advantages it carries. Accordingly, the number of cloud-based services has increased rapidly and strongly in the last few years, and so has increased the complexity of the infrastructures behind them.

Due to the peculiarities of cloud systems and the commonly accepted major obstacles to cloud adoption, effective and efficient monitoring activities are constantly needed to properly operate and manage such complex infrastructures. In the following we first discuss the need of monitoring cloud systems, its goals, and the common issues raised (see § 1.2). We then consider the role of the network in these complex systems, also providing a taxonomy for the public cloud networks (see § 1.3).

1.2 Cloud monitoring

In the last years, in line with the rapid emergence of the cloud paradigm and with the increase of the number of cloud-based services designed and deployed, the complexity of the cloud infrastructures behind these services has strongly increased. Cloud services are on-demand, scalable, and elastic, and serve multiple mutually untrusted customers. Accordingly, the cloud infrastructure is required to expose a set of features, such as availability, reliability, guaranteed QoS, scalability, flexibility, dynamic load balancing, security, and privacy [29, 30]. In order to reach these non-trivial goals, cloud systems have become more complex from both the qualitative and the quantitative point of view. For instance, to provide these desirable features, advanced virtualization techniques, robust and dynamic scheduling approaches, advanced security measures, and disaster recovery mechanisms are commonly implemented and operated in cloud systems. Moreover, datacenters for cloud computing continue to grow in terms of both hardware resources and traffic volume, thus making cloud operation and management more and more complex. In this scenario, effective, efficient, and accurate monitoring activities are required to efficiently operate these infrastructures and to manage their increasing complexity [31].

In the following we first introduce the possible abstraction levels at which monitoring activities can be performed and discuss their goals with respect to the different entities involved; then we present the available approaches that these activities may take advantage of.

1.2.1 Abstraction levels and goals

In cloud computing, both *high-* and *low-level* monitoring is required [32]. Low-level monitoring is primarily concerned with the status of the physical infrastructure of the whole cloud (e.g., physical servers, storage areas, etc.). It is related to information collected by the cloud provider and usually not exposed to the consumers. On the other hand, high-level monitoring is related to information on the status of the components of the virtual platform. This information is collected at the infrastructure, platform, or application layers by providers or consumers through platforms and services operated by themselves or by third parties.

Monitoring activities may be beneficial to many of the different entities involved as a number of heterogeneous activities directly depends on cloud monitoring tasks. Cloud

monitoring is essential to the provider in order to deal with system management activities at large scale, is clearly instrumental for the cloud auditor, and can also be helpful to the customer to manage the leased services or to gather unadvertised information and better understand the characteristics of the service he/she pays for. In the following the most relevant objectives and contexts of these activities are briefly described, relating them to the different entities.

From the provider point of view, monitoring activities are fundamental to properly operate the overall cloud infrastructure. Indeed, having a monitoring system able to capture the state of a complex system like a cloud, is essential [33]. From the one hand, it is crucial to properly control and manage both the hardware layer and the ones above. On the other hand, it is able to provide useful information about performance indicators, for both applications and platforms. These indicators also help the provider to properly plan and design of cloud system.

The need for monitoring activities is also stimulated by volatility of resources and fast-changing conditions which may lead to faults. Virtualization technologies—that allow virtualized resources to migrate at any time from a physical machine to another—introduce an additional complexity level for the provider which is in charge of managing both the physical and the virtual layers. Virtualized servers normally share physical processors and I/O interfaces with others, thus potentially impacting both the computation and communication performance of cloud services and generating security concerns. Accordingly, monitoring activities are often adopted to dynamically evaluate the impact of the virtualization and to properly manage security aspects.

It is worth noting that from the provider point of view, timely, reliable, and comprehensive monitoring is needed to perform troubleshooting, locate problems withing the datacenter, and enforcing control actions. To properly manage huge and complex systems, monitoring activities are required to be very efficient therefore, since they must support real-time operation, also when scaled up to tens of thousands heterogeneous nodes, dealing also with complex network topologies, and I/O structures. As a consequence, proper and efficient data analysis is crucial to generate new beneficial knowledge from the huge amount of data gathered.

Monitoring also allows cloud providers to formulate more realistic and dynamic SLAs and better pricing models by exploiting the knowledge of user-perceived performance. For instance, continuous monitoring the system in terms of availability or delay supplies

the provider with both information such as the workload generated by the consumer and the performance and Quality of Service (QoS) he/she perceives; this information may be used to implement recover actions and may be also partially exposed to the consumer (usually at coarse granularity), as it is helpful to analyze the state of the system, also for identifying SLA violations.

Although monitoring activities are naturally associated to the cloud provider they are a source of invaluable information also for those entities not directly involved in the management of the cloud infrastructure.

Cloud monitoring is undoubtedly functional to the cloud auditor, in order to perform the independent examination of cloud services with the intent to express an opinion thereon. Audits can evaluate security controls, privacy impact, and performance of cloud the service, and are performed to verify conformance to standards, through a review of objective evidence. Monitoring is mandatory and instrumental in certifying SLA compliance when auditing activities are performed to respect regulation (e.g., when government data or services are involved).

Monitoring is also helpful to the customer, in order to gather information about the state of the service he/she pays for, which is hosted remotely, and is accessible only through a high-level interface. Monitoring is essential to predict and keep track of the evolution of all the parameters involved in the process of QoS assurance and can be also used to implement mechanisms to identify SLA violation. Monitoring activities offer also a way to quantify the residual capacity of running resources in front of the actual workload. Hence they can be leveraged to gather the information-base needed to adapt resource deployment in real time.

Monitoring needs become critical in the context of public-cloud services, where a resilient and trustworthy monitoring action helps to reach a proper level of visibility over the level of availability and QoS obtained.

Indeed, monitoring, is necessary to the consumer for performing billing tasks and for verifying usage, and therefore to validate the pay-as-you-go billing model implemented by the provider.

Consumers can leverage monitoring activities in order to perform root-cause analysis, for instance to understand if any occurring failure or performance issue is caused by the provider, by the network infrastructure, or by the application itself. Indeed, the datacenter and the overall cloud system infrastructure represent a big challenge for troubleshooting

as the root cause of each problem can be searched is a number of different components (e.g., the network, or the hosts) and each of them is made of several layers (physical hardware, virtualization layers, operating system, etc.) [31].

From the consumer's angle, monitoring activities are also instrumental to compare different providers. Many public cloud providers offer pay-as-you-go services, with a variety of options in pricing and feature set, and adopting varying approaches to infrastructure, virtualization, and software implementation. This leads to a problem of plenty. Monitoring activities offer a way to evaluate key requirements of cloud offerings, thus allowing the customer to perform better choices [34].

In conclusion, the examples reported above clearly show how the different entities involved in cloud-related activities can benefit from monitoring activities. Each of them however has to face different challenges depending on role played, also according to the specific goal pursued as being intrinsically interested in information at different levels of abstraction.

1.2.2 Available approaches

The specific role assumed clearly defines the information base that each entity can access. For instance, the cloud provider natively has a privileged point of view on the overall cloud system, as being able to observe (and control) it at every layer, from the hardware to the application. Accordingly, the provider can set up dedicated monitoring infrastructure—which can be based on both hardware and software probes—deploying it at any of these layers. In addition, it exactly knows all the design and implementation details of the cloud infrastructure, usually considered as high confidential and therefore not advertised for both security or commercial reasons by commercial providers. This set of information may encompass the detailed design of the datacenter (e.g., the topology of the network and the specific technologies leveraged) or the algorithms implemented for placing servers to customers asking for them.

Similarly, the cloud carrier can access information about the design of the communication infrastructure that delivers cloud services to the consumers. Therefore it is aware of both the traffic engineering strategies enforced—which may impact performance—and the commercial agreements defining how the cloud traffic is transferred.

Conversely, external entities not directly involved into the management of the cloud system or in the delivery of cloud traffic, such as the customer, the broker, or the auditor

usually have no access to any data related to specific design choices which the infrastructure is based on.

It is therefore possible to identify two antipodal families of approaches that can be adopted to perform cloud monitoring activities. These reflect the privileged role that the entity performing the monitoring activities may play, independently from the entity itself and are briefly discussed in the following.

- *Cooperative approaches* require the entity that manage the system under investigation to explicitly cooperate. In other words, these approaches rely on restricted information, originally available only to entities in charge of managing the cloud system. This information may come in different forms, such as server and device logs, traffic traces, system design, etc. Therefore, monitoring activities according to these approaches can be conducted by entities playing privileged roles. The major advantage of adopting such approaches is the possible awareness of the design choices adopted and of the management strategies enforced for the system under investigation. This leads to possibly access to non obfuscated information. However, in practice these approaches can be adopted only by the entities that directly manage the cloud system or may take advantage of particular relationships with the provider (i.e., business partners, collaborating research institutions, etc.).

- *Non-cooperative approaches* do not require access to any restricted information. These approaches adopt the point of view of the general consumer of the cloud service, who can only access it through the interface designed by the provider. These approaches have to necessarily deal with the complexity of the cloud system in its entirety, as the customer is forced to interact with the cloud by accessing it at one of the upper layers of the cloud architecture. Thus who implements these approaches is forced to deal with the complexity introduced by the multiple existing layers laying between the one directly accessed and the one targeted for monitoring. This is a non-trivial limitation to face, as the monitoring results may be also heavily affected by these intermediate layers (e.g., the impact virtualization has to be taken into account [26]). On the other hand, it does not require any privileged role to be enforced, as it is based on data extracted and collected through the adoption of ad-hoc tools.

1.3 Cloud networking

The network is a key component for cloud systems as the characteristics and the performance figures of the communication infrastructure they rely on potentially impact the performance of the overall system [35]. The recent literature strongly remarked its importance, even reporting that *"without high performance networks there would be no such a thing as cloud computing"* [36].

On the one hand, cloud network resources (e.g., within the datacenter) are provided in the same way as computational or storage resources. VMs beside CPU and disk are provided with high-performance network connectivity, in order to make them suitable to support a wide range of applications. In this sense, the desired properties for the cloud network are similar to those of the other resources (on-demand availability, illusion of potentially infinite scaling, fast scalability, etc.). For instance, it is important to stably have enough network bandwidth between VMs, notwithstanding the changing load imposed by multiple consumers because cloud applications usually process large amounts of data and exchange them through the network, both when VMs are deployed into the same datacenter [37, 28, 34] and over multiple geographically distributed sites [38, 39, 34]. Among the different apparently infinite resources managed and provided by cloud systems, network ones have a critical role [36].

On the other hand, cloud services are ubiquitous by definition [3], i.e. they are accessible from everywhere through a network infrastructure. In the case of public clouds, these services are necessarily off-premise, and consequently, both the cloud consumer and the final user can only access them from remote. As such, the cloud network is the only means for the consumer to access any kind of cloud resource. Accordingly, its properties in terms of performance, reliability, and resilience are of the utmost importance, as they impact the QoS perceived by end users, the ability to control other resources, and the availability of the services [40, 41].

1.3.1 Impact on Internet traffic

As a direct consequence of the rapid adoption of and migration to cloud architectures, the cloud traffic (i.e., the traffic generated by and/or directed to cloud systems) is dramatically growing. This is also supported by the ability of cloud datacenters to handle significantly higher traffic loads, thus leading to better application performance.

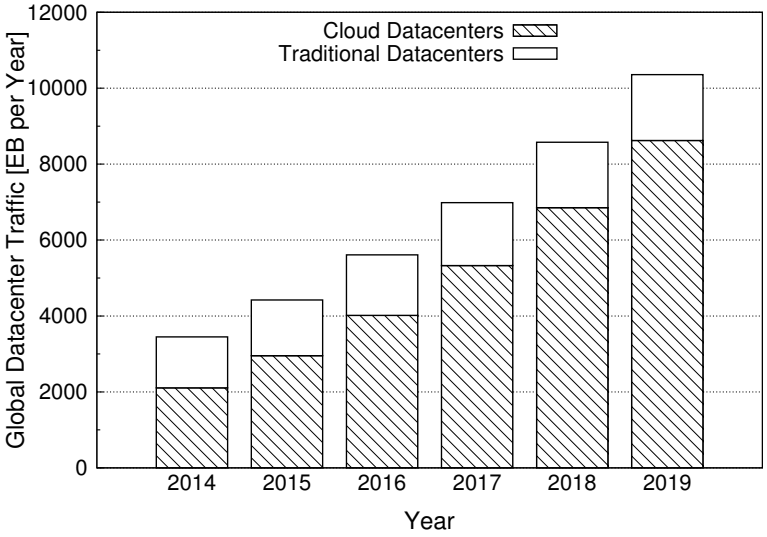


Figure 1.3: Traditional-datacenter vs. cloud-datacenter traffic growth (global traffic, EB per year). Data source: Cisco Global Cloud Index [15].

The relevance of the global cloud-traffic evolution is also highlighted by recent reports and forecasts [15]. From a quantitative point of view, the impact of cloud computing on Internet traffic is clear. Most of the Internet traffic has originated or terminated in a datacenter since 2008 indeed—when peer-to-peer traffic ceased to dominate the Internet application mix. Datacenter traffic is expected to continue to dominate Internet traffic for the foreseeable future, although its nature is undergoing a fundamental transformation due to changes in cloud applications, services, and infrastructures. As a result more than three-quarters of datacenter traffic will be cloud traffic by 2018.

Figure 1.3 shows how datacenter traffic is expected to grow in the next years. The amount of annual global datacenter traffic in 2014 was estimated to be around 3.4 ZB, and by 2019 will triple to reach 10.3 ZB per year (25-percent CAGR). Indeed, cloud datacenter traffic will grow at a faster rate (32-percent CAGR) that leads to a 4-fold growth from 2014 to 2019. As a results of this trend, by 2018 more than three-fourths (78 percent) of all workloads will be processed in cloud datacenters.

1.3.2 Cloud network infrastructures

In accordance with the conspicuous growth in volume of the cloud traffic, cloud networking and its performance are attracting more and more the interest of both the scientific and industrial communities [42, 43]. Datacenter and cloud architectures continue to evolve to

address the needs of large-scale multi-tenant datacenters and clouds and geo-distributed networks of datacenters are being built. These usually consist in a dozen mega datacenters (so termed because of the number of servers they host and the high power they draw), and a larger number of micro datacenters (used primarily as nodes in content distribution networks) [44]. Accordingly, huge investments have been made for cloud networking, in terms of both research and cutting-edge infrastructures.

Networking is a non-negligible source of cost for datacenter infrastructures, being estimated to amount to around 15 percent of the datacenter's total worth and being comparable to power costs [44]. The capital cost of networking gear for datacenters (primarily switches, routers, and load balancers) is a significant fraction of the cost of networking. The remaining networking costs are concentrated in wide-area networking: peering points, where traffic is handed off to the Internet Service Providers (ISPs) in charge of delivering packets to end users, inter-datacenter links carrying traffic between geographically distributed datacenters, and regional facilities needed to reach wide area network interconnection sites. The costs to deploy and manage this distributed networking infrastructure have decreased dramatically over the past few years, but still remain significant, and still exceed the cost of networking inside the datacenter.

Datacenters are interconnected across the wide-area network via routing and transport technologies in order to provide the pool of resources, known as the cloud. A typical cloud infrastructure is shown in Figure 1.4. It should be understood that variations of this architecture do exist (e.g., in smaller datacenters, it is likely that some network layers are collapsed). It is used in this thesis as a reference architecture to discuss cloud networking, its organization, and common adopted technologies.

Large cloud datacenters target to support tens of thousands of servers and tenants, exabytes of storage, and terabits per second of traffic [42]. The virtualization of computing and storage resources inside a datacenter provides the foundation for offering resources and application services to multiple tenants on the same infrastructure: via computing virtualization, multiple VMs are created on the same server, possibly for different tenants. In order to support a large number of tenants, networks connecting both resources within the same datacenter and geographically distributed datacenters have been evolving from the basic virtual LAN (VLAN) and IP routing architecture to an architecture that provides network virtualization on a larger scale. These network infrastructures are required to support a large number of tenants, bandwidth growth, VM mobility, network elasticity

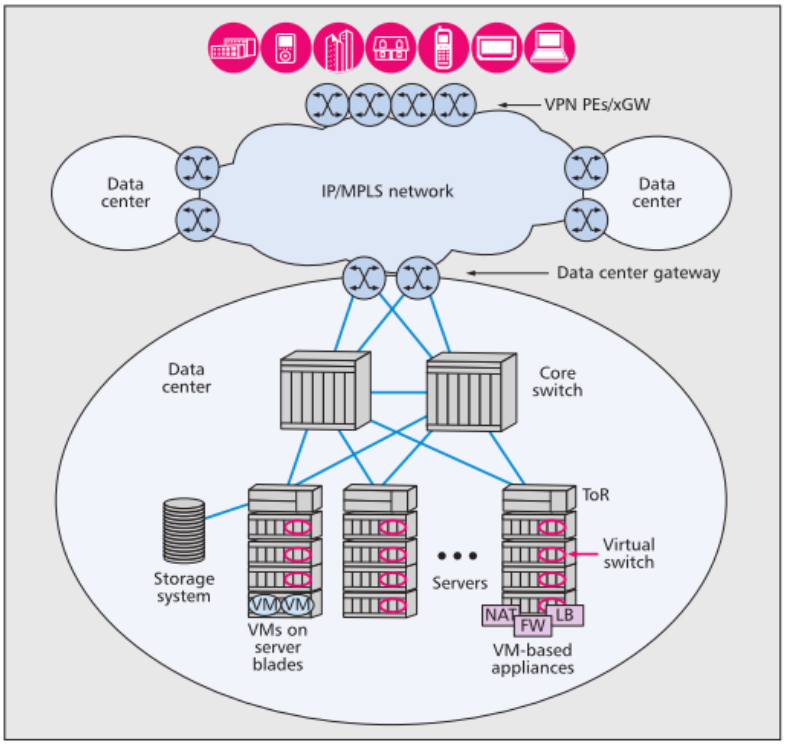


Figure 1.4: Generic architecture for cloud networking. Source: [42].

and provisioning agility, efficient resource utilization, and efficient traffic engineering with performance constraints [42].

The datcenter depicted in Figure 1.4 includes *Virtual Switches* (vSWs), *Top-of-Rack switches* (ToRs), and *core switches* at different tiers of a network hierarchy within a datacenter. In addition, a gateway and IP/MPLS networks provide inter-DC connectivity, and connectivity to the Internet and end users. The vSW is generally a software-based Ethernet switch function running inside a server host. It can support Ethernet and/or IP services, while providing for switching and routing context separation among tenants sharing the same server. A vSW may be single or dual-homed to the ToRs via Ethernet links. A ToR is a hardware-based network element that typically supports Ethernet VLAN services and/or simple IP routing for the datacenter infrastructure. Other deployment scenarios may use *End-of-Row switches* (EoRs) to provide a similar function to a ToRs, often with a larger number of physical ports and higher switching capacity. ToRs or EoRs aggregate Ethernet links from the servers and are usually dual-homed to core switches via Ethernet links. A core switch aggregates multiple ToRs or EoRs, and can support large-scale virtual LAN services and/or simple IP routing for the datacenter. Depending

on the size of the datacenter, there could be two or more core switches, and these switches can form a hierarchy of more than one layer. The datacenter gateway to the wide-area network provides datacenter interconnection and connectivity to Internet and Virtual Private Network (VPN) customers.

Common server-ToR or server-EoR links are 1GbE and 10GbE, whereas Driven by cost, 10GbE is a popular choice across the remaining datacenter core [42]. These higher rates will also find their way to servers, ToRs, and EoRs incrementally.

Datacenters may connect to one or more network service providers to provide connectivity to users accessing cloud services from private sites or the Internet, or to gain connectivity among datacenters. Connectivity among datacenters can be obtained from a service provider as a leased fiber or private line service. It can also be a layer 2 Ethernet or IP VPN service. In addition, datacenter inter-connection may utilize optical transport to provide for large bandwidth demands that arise from moving or mirroring large blocks of data and video among datacenters. Datacenter geo-diversity—although requiring *ad-hoc* designed services and/or applications to be properly leveraged—lowers latency to users, enhances their experience, and increases reliability in the presence of outages taking out an entire site. Economies of scale available at the time the datacenter is designed, play a major role in the choice of the placement and the size of the datacenter itself, which are determined by server cost, power availability, and local factors such as power concessions or tax regimes. However, although one would like to place datacenters as close to the users, also transfer cost and latency among datacenter are crucial. Finding an optimal balance between these aspects is challenging, and the available solutions are heavily impacted by network costs. More in general, cloud infrastructures deployed are the results of a challenging optimization problem, involving several factors such as the benefits achieved through geo-diversity, economies of scale, and the network cost.

1.3.3 Cloud network taxonomy

As the number of services and applications backed by the cloud paradigm increased during the years and their requirements became more and more urgent, so the complexity of cloud systems and of their network raised to both support advanced features and respond to the rapid adoption by consumers. Indeed, the cloud network interconnects a high number of cloud resources of different kind (VMs, storage buckets, software containers, etc.). These cloud resources are connected to each other (both when they are placed in

the same datacenter and in different geographically distinct sites), and to geographically spread consumers accessing cloud services from the public Internet. As the resources are characterized by a high level of dynamism, also the network is demanded to have proper requirements in terms of flexibility and scalability.

Interactions among the cloud and the consumers may be of different nature [45], and they often involve the cloud network in its entirety, because of the functional separation of layers often implemented. Cloud applications perform a number of different network interactions in a transparent manner to the final user. The characteristics and the performance of the network infrastructures that these applications rely on for each interaction is important, as impacting the Quality of Experience (QoE) perceived by users in accessing cloud services.

For ease of description, the complex cloud infrastructure leveraged by cloud services can be seen as the composition of three major network areas:

- the intra-datacenter network;
- the inter-datacenter network;
- the cloud-to-user network.

Each of these network areas has different characteristics, as it is designed for different purposes and is subjected to different constraints (in terms of resources, available technologies, control of the infrastructure, and ownership). For instance, in some cases we deal with a network infrastructure deployed over the datacenter limited area and that is completely owned, controlled, and managed by the cloud provider itself. In others, the providers leverage resources owned by telco operators, under specific agreements. In the following these three areas will be briefly described, in order to report their peculiar characteristics and highlight their main differences.

The *intra-datacenter* network, connects cloud resources (e.g., VMs leased by consumers) among themselves and with the shared services placed within the datacenter. The performance of this network is vital to the performance of applications distributed across multiple nodes in a cloud datacenter. Indeed, the functional separation of layers (application servers, storage, and databases) leads more and more to the design and the implementation of distributed solutions, generating replication, backup, and read and write traffic traversing the datacenter. Furthermore, parallel processing divides tasks and sends them to multiple servers, contributing to intra-datacenter traffic [9, 10].

The *inter-datacenter* network is a wide-area network that connects datacenters placed in geographically distributed regions. It often has quite different properties compared to the intra-datacenter network, due to the different characteristics of the services that rely on it, the different technologies adopted, and the need to rely on external network service providers. Lots of applications and services are designed to benefit from the geographically distributed architecture. In accordance to the evolution of the inter-datacenter network, cloud applications more and more exchange traffic among datacenters placed in two different sites on the globe [46, 47, 48]. Common tasks that rely on the inter-datacenter network are related to the proliferation of services that need to shuttle data between clouds, and to the growing volume of data that needs to be replicated across datacenters (for instance to place contents near to final users).

The *cloud-to-user wide-area network* is defined as the collection of network paths between a cloud's datacenters and external hosts on the Internet. It may play a role in many kinds of applications (e.g., from simple web-based solutions to collaborative applications such as documents sharing, voice and video telepresence, or distributed games). More in general, high-performance networks are a requirement for practices and designs inherent to the cloud computing infrastructure, such as replication, task distribution, sharing, synchronization, offload or rapid scaling. Speed and latency matter, indeed: substantial empirical evidences suggest that performance directly impacts revenue. For example, Google reported 20 percent revenue loss due to a specific experiment that increased the time to display search results by as little as 500 ms. Amazon reported a 1 percent sales decrease for an additional delay of as little as 100 ms [49, 50]. The performance of the cloud-to-user network is clearly impacted by the location of the user. The latency, in particular is affected by the (geographic) distance from the user to the cloud service. This creates a strong motivation for geographically distributing datacenters around the world to reduce speed-of-light delays, Accordingly, all the major cloud providers host the offered services at differing locations.

Cloud traffic by network. Recent forecasts report that the portion of traffic residing within datacenters will slightly decline over the forecast period, accounting for nearly 75 percent of datacenter traffic in 2014 and about 73 percent by 2019. Despite this decline, the majority of traffic remains within the datacenter because of factors such as the functional separation of application servers, storage, databases, or parallel processing

Table 1.2: Global datacenter traffic by network(EB per year). Source: Cisco Global Cloud Index [15].

	2014	2015	2016	2017	2018	2019	CAGR 2014–2019
Intradatacenter	2,602	3,342	4,233	5,235	6,358	7,566	24%
Interdatacenter	234	321	432	564	723	905	31%
Datacenter-to-user	613	760	946	1,185	1,495	1,886	25%
Total	3,449	4,423	5,611	6,984	8,576	10,357	25%

which generates replication, backup, and read and write traffic traversing the datacenter.

Traffic between datacenters is growing faster than either traffic to end-users or traffic within the datacenter. Up from nearly 7 percent at the end of 2014, this type of traffic will account for almost 9 percent of total datacenter traffic by 2019. The high growth of this segment is due to the proliferation of cloud services and the consequent need to shuttle data between clouds, and the growing volume of data that needs to be replicated across datacenters, and the increasing prevalence of content distribution networks.

Final remarks. The network is a key component of a cloud system. As previously discussed, the access to network resources is one of the service provided on-demand to cloud consumers, and it is through the network indeed that cloud resources are connected together, and that the wide range of cloud services is made ubiquitously accessible to geographically distributed consumers. In order to support the increasing demand and guarantee adequate performance and services, cloud networks are evolving, thus becoming more and more complex over time.

However, notwithstanding the widespread adoption of public-cloud services and the central role of cloud networks, due to both security and commercial issues, public-cloud providers rarely provide customers with the desired information about their state and expected performance.

1.4 Contribution and organization of the thesis

Cloud computing is having a dramatic impact on the ICT industry, significantly contributing to the promotion of growth and competition. Therefore, a large adoption of this paradigm by enterprises has been registered over the last few years, generating billion-dollar worldwide spending. According to the current trend, where organizations more

and more outsource their ICT infrastructure, large companies are striving to provide customers with more appealing cloud services also making huge investments in increasingly efficient and complex public-cloud infrastructures. Among the components of these complex cloud infrastructures, the network plays a primary role as its properties directly impact the performance of the overall cloud system. Indeed its performance has a huge impact on applications, costs, and QoS perceived by consumers.

Considering the large and increasing adoption of public clouds, the huge investments being made to connect public cloud nodes (both among themselves and to the customers), and the impact on the Internet of the traffic they generate, a consistent information base about the properties of the public-cloud network is essential. This is true for the entities involved in the management of the system, but also for customers as they could derive from it guidelines to take better advantage of cloud services (hints on service deployment and scaling, service migration, expected performance and variability prediction, etc.).

In spite of the urgency in having detailed information about the cloud networking environment, its design, and its performance figures, the general customer naturally suffers from the lack of accurate and trustworthy information about it. The critical dependence of the industry from public-cloud infrastructures has grown faster than the customer's understanding of their dynamics, their performance, and their limitations. This scarce comprehension is the consequence of the fact that detailed information about cloud performance, characteristics, settings, and load are considered confidential by cloud providers for security and commercial reasons [27, 36]; in addition, SLAs only vaguely describe network performance guarantees, and customers can only refer to incomplete and rough information advertised by the provider [51].

Although public-cloud datacenters intrinsically depend on high-performance networks to connect servers within the datacenter and to the rest of the world, providers seldom make any promises about network performance, even though many cloud customers do want to be able to rely on network performance guarantees. Consequently, public-cloud consumers suffer from unpredictable network performance which leads to application performance and cost issues. In this context, activities aimed at monitoring public-cloud networks through non-cooperative approaches are essential.

In this thesis we aim at investigating the techniques for monitoring the performance of public-cloud networks through non-cooperative approaches, focusing on the possible methodologies that can be enforced, their related issues, and some applications of the

gathered information.

With respect to the two leading public-cloud providers, i.e., Amazon Web Services and Microsoft Azure (see § 1.1.2), our study involves all the three network areas introduced in § 1.3.3. Focusing on network monitoring activities, possible techniques, and related issues we propose a detailed analysis of the performance of the intra-datacenter, inter-datacenter, and cloud-to-user networks, also deriving some helpful usage guidelines for customers. In addition we propose a monitoring platforms that helps consumers to go beyond the scarce information exposed by cloud providers about the design performance of their networking infrastructures. Finally, we focus our attention on one of the possible applications of the information gathered through monitoring activities, i.e. the automatic scaling of resources in public clouds.

The remainder of the thesis is organized as follows. Chapter 2 proposes an extensive study of the literature about the approaches for monitoring cloud systems and their performance, focusing on the techniques for monitoring public cloud networks through non-cooperative approaches. Chapter 3 presents CloudSurf, a platform that enable the general customer to monitor public-cloud networking infrastructures. We detail its architecture and implementation, and show how its features are helpful to characterize the performance of the public-cloud networks. In Chapter 4, Chapter 5, and Chapter 6 we present the main results of our analyses for the intra-datacenter, inter-datacenter, and cloud-to-user networks, respectively. In each chapter the aspects of major interest have been deepened. Chapter 7 proposes a solution that leverages monitoring data to implement strategies to automatically scale cloud resources in public clouds, in front of the possibly fluctuating workload and the unknown environment. Finally, we draw our conclusion in Chapter 8.

Chapter 2

Cloud network monitoring: state of the art

In this chapter we propose an extensive study of the scientific literature about the approaches and the solutions for monitoring cloud systems, focusing on the aspects related to the public-cloud network and its performance.

We first introduce the common issues related to cloud monitoring systems implemented by providers, showing the potentialities of the studies that leverage information collected from privileged points of view. Then we focus our attention on the available techniques for monitoring cloud networks through non-cooperative approaches, and on their main outcomes and limitations.

2.1 Cloud monitoring systems and related issues

From the provider point of view, monitoring is crucial to manage both the hardware and software cloud infrastructure, as it is functional to a set of activities, such as datacenter management, SLA management, billing, troubleshooting, and security management.

Because of some peculiarities, such as the high level of dynamism, the virtualization techniques employed at several layers, or the presence of a huge number of consumers possibly generating huge volumes of traffic, cloud systems introduce a number of new challenges to be tackled with respect to other slowly-changing infrastructures. Indeed, cloud platforms have different requirements than computing grids or clusters, as consisting of multiple layers and service paradigms that enable to provide on demand users with virtually infinite resources. Therefore monitoring systems for clouds expose more strict requirements than the homologous for the other platforms mentioned above. As such,

general purpose monitoring tools (e.g., Ganglia [52], Nagios [53], Zabbix [54], MonALISA [55], and GridICE [56]) traditionally used by system administrators to monitor fixed or slowly-changing distributed infrastructures are unsuitable to be directly utilized for rapidly elastic and dynamic cloud deployments.

Cloud monitoring systems require a number of specific properties that have been deeply investigated in the scientific literature, as discussed in the following. A monitoring system is required to be scalable, as it has to cope with a potentially huge number of different resources and consumers. In the literature such issue has been mainly addressed by proposing architectures in which monitoring data and events are propagated to the control application after their aggregation and filtering. As such, most of the proposed approaches—regardless of the specific low- or high-level monitored parameters—propose aggregation and filtering before propagating data and events to the control application. For instance, some solutions adopt a subsystem to propagate event announcements [57, 58, 59, 60] and/or rely on agents, which are responsible for performing data collection, filtering and aggregation [57, 58, 61].

In addition, monitoring systems have to be elastic in order to address the dynamic changes of the monitored entities. This is required to correctly monitor virtual resources continuously created and destroyed. This property also implies scalability, and adds the requirement of supporting real-time upsizing and downsizing of the pool of the monitored resources [62]. As many different monitoring systems proposed for large distributed systems have been designed for a relatively slowly changing physical infrastructure they do not directly support a rapidly changing dynamic infrastructure and they are not suitable for as-is adoption in cloud scenarios.

The adaptability, i.e. the ability of adapting the computational and the network loads imposed to the system itself is a also a requirement. Monitoring activities have to be not invasive, as the workload generated by monitoring activities (e.g., associated to monitoring data collection, processing, storage, and transmission) potentially impacts computing and communication resources and represents a cost for the cloud infrastructure. Several studies address this issue [58, 59, 61, 63, 64] by tuning the amount of monitored resources and the monitoring frequency.

Since monitoring is instrumental to activities connected to core goals, related tasks need to be timely and accurate, such that detected events are available on time for their intended use [59] and correct information can be leveraged to effectively perform manage-

ment activities. However, it is worth noting that granting timeliness implies challenges or trade-offs between opposing requirements. For instance, the shorter the sampling interval, the smaller is the delay between the time a monitored condition happens and is captured. Thus, a trade-off between accuracy and sampling frequency is necessary, considering also the resource constraints (e.g. CPU, network bandwidth, or memory) [63, 59].

While on the one hand the above issues are strictly related to the management activities, that are by definition a responsibility of the provider, on the other hand, part of the information collected could be also beneficial to the consumer (e.g., current load of the datacenter, state of the hosting machines, etc.).

With respect to largely adopted public clouds however, only a subset of the information collected by the provider is exposed to the consumers through the purposely-designed monitoring interfaces. Cloud specific monitoring tools—such as Amazon CloudWatch [65], AzureWatch [66], CloudMonix [67], Monitis [68]—provide Monitoring-as-a-Service to cloud consumers in public clouds and are directly made available by the cloud provider itself, or by third parties. Although these tools are easy to use and well integrated with underlying platforms, (thus hiding all the complexity related to monitoring tasks discussed above) their adoption is limited to specific providers, and to the scarce set of high-level observations exposed. Moreover, the information provided is typically coarse-grained (with sampling time of several minutes) and is mostly related to the residual capacity of the virtual resources (e.g., CPU or disk utilization) rather than to their performance. For instance, in the case of Amazon CloudWatch the customer is provided with information related to the volume of the traffic that the leased VMs have injected into the network during the last five minutes, but no detail about its state (overhead of the device along a path, available bandwidth, latency, etc.) is provided.

More in general, public cloud providers usually do not provide consumers with detailed information about the cloud network, its design, or its performance [36, 27, 19, 69], although cloud datacenters and the services they host intrinsically depend on high-performance networks to connect servers within the datacenter and to the rest of the world. Whereas cloud providers typically offer different service levels at varying prices they only advertise CPU, memory, and disk storage. While all cloud providers grant network connectivity to tenant VMs, they seldom make any promises about network performance, and so cloud tenants suffer from highly-variable, unpredictable network performance.

This situation motivates the recent research related to non-cooperative approaches.

Before discussing it and the related issues in § 2.2, we will discuss in the following the potentiality of the cloud network monitoring approaches that can leverage information directly provided by entities playing privileged roles (§ 2.1.1).

2.1.1 Studies leveraging privileged points of view

As little is known about the characteristics of the both intra- and inter-datacenter traffic, a set of scientific contributions leverage provider-side privileged information to devise its characteristics and properties. More specifically, some works took advantage of system logs, socket-level, SNMP at both the datacenter hosts and the devices [70, 71, 72] network traces [73, 74]. It is worth noting that not all the researchers can access this kind of information and therefore these studies are hard to replicate and validate.

Kandula et al. [70] contribute to the investigation the characteristics of datacenter traffic, leveraging server log collection in a 1500-server operational cluster and tracking events at socket layer. Thanks to their privileged point of view, the authors provide a first attempt to the characterization of inter-datacenter traffic, inferring both the macroscopic patterns (e.g., which servers talk to which others, when, and for what reasons) as well as the microscopic characteristics (e.g., flow durations, inter-arrival times, etc.) and devise useful guidelines for datacenter network designers.

Benson et al. [71] analyze SNMP logs collected at 19 datacenters to examine temporal and spatial variations in link loads and losses. They found that the links closer to the edge observe a greater degree of loss, whereas the ones in the core expose higher degree of utilization. Studying packet traces collected at a small number of switches in one datacenter they also found evidence of ON-OFF traffic behavior. As such, thanks to the rich information collected, the authors develop a framework that derives ON-OFF traffic parameters for datacenter traffic sources and models their sending behaviors.

Potharaju et al. [72] present a preliminary analysis of cloud intra-datacenter and inter-datacenter network failures from a service perspective, taking advantage of a wide range of network data sources including *syslog*, SNMP alerts, network trouble tickets, maintenance tracking, revision control system, and traffic carried by links. Authors provide insights to improve service reliability, and present two aspects of failure characteristics, i.e., the problem root causes and the server downtime. For what concerns the intra-datacenter network, considering three classes of devices, i.e., *top-of-rack switches*, *aggregation routers*, and *access routers*, they reveal that interface-level errors, network card

problems, and unexpected reloads were prominent among all the three types of devices. Aggregation switches exhibit the highest downtime, while access router the lowest. Top-of-rack switches failures tend to be relatively infrequent compared to access routers and aggregation switches and hence expose the lower downtime. A relevant contribution to top-of-rack switches downtime is related to a set of older generation devices. About the inter-datacenter network instead, authors found that link flapping (e.g., due to BGP or OSPF protocol issues and convergence, and possibly generated in response to a fiber cut) dominates failure root causes, while high utilization represents the second one.

Yingying et al. [73] present a first study of inter-datacenter traffic characteristics using anonymized *NetFlow* datasets collected at the border routers of five major *Yahoo!* datacenters. Applying some heuristics based on the IP addresses and on router interfaces the authors identify two types of inter-datacenter traffic (i.e.,the one triggered by traffic generated by the users and the background traffic). Moreover, their results capture some insights into the way the provider has designed its infrastructure, identifying a hierarchical organization for the datacenter deployment. They also show that several *Yahoo!* services have correlated traffic. These correlations have important implications for distributing different services at multiple datacenters.

Bermudez et al. [74] perform a large scale observation and analysis of Amazon AWS traffic. In their work the authors rely on a 1-week portion of a 3-month passive measurements dataset collected at one large-city PoP of an italian ISP. They explore the EC2, S3, and CloudFront Amazon services to unveil their infrastructures, the pervasiveness of content they host, and their traffic allocation policies. They found that among the datacenters available, the one placed in the US is the most used one as it handles alone 85% of total traffic generated by EC2 and more than 64% for S3. In addition, results also show that this popular datacenter is also the worst performing one. Authors confirm that companies offering contents from EC2 and S3 tend to rely upon one datacenter only, making the network to pay the large cost of carrying information to far-away end-users and thus enhancing the risk of service disruption in case of failures involving the network or the datacenter.

Enforcing different and complementary approaches, these works show the potential of analyses conducted leveraging privileged datasets. Although these results cannot be always generalized (as depending on the specific dataset [70]), this kind of studies is able to expose very interesting outcomes (e.g., datacenter organization, traffic correlation, device

failure rate, network congestion, incast), as they can deal with the problem of interest in depth (e.g., authors can perform root-cause analyses or can rely on accurate ground truth).

In turn the implementation of these approaches, has to face problems related to scalability and efficiency (because of the size of the dataset they have to deal with and the impact that its collection may have on the infrastructure itself [70, 71]), and—above all—privacy and security, because of the nature of data that may impact the privacy of users or may contain details that the provider does not wish to disclose [73]. As a results, few researchers can benefit from these datasets and a very limited number of works in the literature can effectively pursue similar approaches, since this kind of information is usually considered as highly confidential and thus is not publicly exposed for both commercial and security reasons. As such, these studies are hard to validate and replicate.

2.2 Non-cooperative approaches

Implementing non-cooperative approaches is the natural solution to the lack of monitoring information exposed by the provider. In this section we discuss how non-cooperative monitoring approaches have been adopted in the literature to investigate public clouds and in particular their networks infrastructures.

The works in the literature face the non-trivial task of monitoring and evaluating cloud networks with different goals in mind. A number of pioneering works, for instance, in front of environments with unknown characteristics try to shed light on the suitability of public cloud networks to host specific applications or classes of applications [75, 76, 77, 37]. Some studies propose an evaluation of specific cloud services [78, 79], also focusing on how network performance is subjected to variation in front of the virtualization techiniques adopted [37, 26]. Others, due to the large amount of cloud offerings available, aim at providing cloud customers with a fair comparison among a number of different providers or cloud solutions [34, 77]. A set of works, propose and evaluate approaches that through network measurements investigate the reliability, the availability, or the performance of cloud services [80, 81, 41, 40]. Some proposals are targeted instead to the optimization of cloud performance or to the reduction of costs for the customer [28, 38, 82]. Finally, in some cases, network monitoring activities are functional gather data to feed models for simulation [83, 48, 82].

It is worth noting that beyond their final goals all the studies adopting non-cooperative approaches have to face some common issues. A first set of issues is related to the collection of data. As no privileged entity makes available the information of interest, in order to obtain significant data researchers have to cope with experimental campaigns involving public clouds. As such, they have to play the role of the consumer. Because of the pay-per-use model, this implies additional costs, that often impose a trade-off with the number of experiments and therefore the accuracy of the observation. In order to investigate the characteristics of the cloud network, the proposed approaches mainly leverage active techniques i.e. that measure the properties of the network by injecting purposely created traffic into it, in some cases from a number of distinct vantage points, not always available to the experimenters [76]. Note that this may be the source of non-negligible additional costs, as providers usually impose charges based on the volume of traffic carried by cloud networks.

Secondly, researchers have to cope with the interpretation of data coming from a highly dynamic and uncontrolled environment. Indeed, the factors that may influence monitoring results are not under the complete control of the customer, and therefore of the researcher performing the experiments. Additionally, the presence of virtualization, differently than in traditional environments, may heavily compromise the interpretation and even the validity of experimental results [26]. The lack of a ground truth further exacerbates this issue, as outcomes without the help of privileged entities cannot be easily validated. As such, sometimes studies [28, 27] implement additional monitoring activities to obtain insights into the design of network infrastructure implemented by the provider providing help in the interpretation of other results (i.e., related to the performance).

In the following we will discern between the studies that perform an evaluation of the cloud network with respect to a specific application by directly deploying it onto the cloud and monitoring its behavior (*application-specific approaches*—§ 2.2.1), and those which aim at evaluating the cloud infrastructure, thus achieving more general results (*application-agnostic approaches*—§ 2.2.2).

2.2.1 Application-specific approaches

A number of works in the literature aims at investigating the suitability of public clouds to run a given class of applications (e.g., web-application deployment, scientific workflows, or science grids). Those works usually take into account the performance of the applications

according either to the Quality of Experience perceived by users or the Quality of Service indices limited to those of interest for the application under test, and often consider the performance of the network. Often, these studies propose a comparison with in-house testbeds in order to understand the performance degradation generated by cloud environments.

Juve et al. [75] propose an evaluation of the cost and performance of scientific workflows on Amazon EC2 and S3. In order to evaluate the cost and performance of these workflows in the cloud, they conducted experiments using three applications: an astronomy application, a seismology application, and a bioinformatics application. These have been chosen to cover a wide range of domains and resource requirements, as they need resources in terms of I/O, CPU, and memory in different ways. On the basis of their experimental results, authors claim that the performance of EC2 is good enough for many applications, and the cost was within reason for the applications and scenarios studied.

Liu and Wee [76] present a benchmark performance study on various cloud components to show their performance results and revealing their limitations. Adopting a web-server benchmark that emulates different web applications (e.g., banking, support, and e-commerce) they studied the performance of Amazon EC2, S3, and Elastic Load Balancing services, and Google App Engine. Their results show how the performance of a single web server is an order of magnitude lower than the state-of-art hardware solutions. Moreover they found that the performance of a software load balancer based approach is limited by both a single network interface and traffic relaying, which halves its effective throughput.

Tudoran et al. [77] compare the Azure public cloud to Nimbus private cloud [84], considering the needs of scientific applications (computation power, storage, data transfers, and costs). They observed dramatic performance and cost variations across platforms in their VMs, storage services, and network transfers. They found that the public cloud delivers almost double performance although with a higher variability.

Palankar et al. [79], *Hill et al.* [85], *Schad et al.* [37], and *Saljooghinejad et al.* [81] also propose some evaluations based on specific applications (considering the data generated by a high-energy physics project, a web-based application to integrate data from ground-based sensors with satellite data, a specific MapReduce job, and web-based key-value store application, respectively) though not limiting their contribution to those. Therefore will be discussed in the next session.

All these work are well motivated by the need of performance details of applications in cloud environments and provide pioneering results in this sense. However, their results cannot be easily generalized as they directly address the performance evaluation of specific applications in specific scenarios, although they could be of great interest for the readership specifically interested in that. Indeed, the deployment scenarios and the configuration factors considered are usually chosen according to the sensibility of the experimenter. Therefore these works are not able to provide an effective contribution to the characterization of public cloud infrastructures.

2.2.2 Application-agnostic approaches

A number of works in the scientific literature performs activities to monitor public-cloud infrastructures as a whole, also focusing on their networking aspects. Differently than the studies presented above, these works aim at characterizing the cloud infrastructure in its generality, i.e., try to deliver general results about its current state and its expected performance, which do not depend on a specific application. These works are motivated by different purposes. A part of them aims at identifying anomalous behaviors or at testing the suitability of public-clouds to host some class of applications (they evaluate cloud infrastructures instead of directly testing the application itself). Others evaluate the performance of public-cloud systems in order to propose a comparison of the offers made by different providers. Finally, a part of these works, performs monitoring activities to feed models or to support theoretical approaches and sustain their practical feasibility in the wild.

Palankar et al. [79] specifically focus on S3 providing as a main contribution the evaluation of S3 ability to provide storage support to large-scale science projects from a cost, availability, and performance perspective. In their experimentations authors evaluate download time for objects of different sizes, concurrent performance, and performance from different remote locations. Authors claim that S3 is not designed to properly accomplish the needs of the science community and identify some application requirements that are not satisfied by S3. In particular, they state that, while S3 successfully supports relatively simple scenarios (e.g., personal data backup) and can be easily integrated in the storage tier of a multi-tiered web application, its security functionality is inadequate to support complex, collaborative environments like the ones in scientific collaborations.

Iosup et al. [86] propose a performance analysis of long-term traces from Amazon and Google cloud providers. They take into account a number of performance indicators (deployment time, cloud-storage download and upload throughput, query response time, update latency, etc.). For what concerns the network, they found daily, weekly, and yearly patterns for the download throughput. In addition their results suggest that when designing an application that uses Amazon S3, application architects should consider several design decisions including which datacenter to store the buckets.

Hill et al. [85] focus on Azure compute, storage, and network performance. For what concerns the network resources, they evaluate the VM instantiation time and the TCP performance between a number of VM pairs in terms of latency and throughput. In the first case, they consider the round-trip time obtained transferring 1 byte of information, while for measuring the throughput they instruct a VM to transfer 2GB of information (the duration of each throughput experiments is around 30 seconds). Based on these experiments, they also provide their performance-related recommendations for the customers. Although the authors claim to have performed $10,000$ experiments, it is worth noting that they only consider ten distinct pairs of VMs (five for the latency experiments and five for the throughput experiments, respectively). Consequently, their results can be hardly generalized. Moreover, no information about the datacenter in which the analysis is carried or about the deployment factors considered (e.g., the type and the size of the VM) is exposed.

Schad et al. [37], focus on performance unpredictability in cloud computing and carry out a study on performance variability in Amazon EC2. Their study takes into account performance in terms of instance startup time, CPU, memory, disk I/O, and S3 access. They show that, differently from physical clusters, cloud VMs of different sizes suffer from a large variance in performance, that also generates issues to applications.

For what concerns the performance of the network, they used *iperf* to evaluate the maximum TCP and UDP throughput. Their results reported slightly more oscillation in performance in US location than in EU location (probably due to the differing demands associated to the regions). More in general, they found values ranging from from 200 to 800 KB/s. On the basis of their experience, the authors also suggest that when performing measurements in cloud environments it is important to consider the specific deployment scenarios, as they could be somehow related to performance variability possibly observed. It is worth noting that the performance values reported by the authors are two orders of

magnitude lower that the ones reported by *Li et al.* [34] obtained with the same monitoring tool.

Hu et al. [41] focus on cloud availability, evaluating the ability of network- and application-layer probes to monitor this aspect. They test both storage and VM services, for Amazon, Microsoft, and Google, and found that the widely adopted ICMP probes potentially lead to both underestimate and overestimate cloud service availability. They found that ICMP is not robust as application-layer measurements indeed, as it may miss internal failures in the cloud back-ends. Therefore they conclude that it is important to adopt end-to-end application-layer measurements (e.g., HTTP) to best characterize cloud service availability.

Feng et al. [83, 48] propose a protocol and a set of algorithms to support video conferencing provided as a cloud service and taking advantage of the inter-datacenter network. The evaluation of the proposed approach over the Amazon EC2 cloud, proves to deliver a substantial performance advantage over state-of-the-art peer-to-peer solutions. In their study, the authors performed an experimental evaluation of Amazon network paths interconnecting seven different datacenters, considering medium and small VMs. They monitored inter-datacenter paths for 3 minutes and revealed very different throughput and latency values.

Garcia-Dorado and Rao [38] propose *CloudMPcast* to construct overlay distribution trees for bulk-data transfer that both optimizes costs of distribution, and ensures end-to-end data transfer times are not affected. Extensive evaluations of CloudMPcast leveraging an extensive set of inter-data-center bandwidth and latency measurements from both Azure and EC2 have shown significant benefits. Cost savings range from 10% to 60% across a wide variety of scenarios. Authors' proposal has been supported by experimental analyses, which investigated the throughput and the latency leveraging two minute long experiments during one day.

Tomanek and Kencl [40] introduce *CLAudit*, a prototype cloud-latency monitoring platform that utilizes the PlanetLab network [87] to place globally distributed probes that periodically measure cloud-service latency. Measurement are performed at various layers of the communication stack in order to also detect anomalous behaviors.

Mulinka and Kencl [80] present methods for automated detection and interpretation of suspicious events within the multi-dimensional latency time series obtained by *CLAudit*. They introduce three different metrics (namely, threshold, standard deviation, and

histogram) to detect occurrences of suspicious events. They validate these methods of unsupervised learning and analyze the most frequent cloud-service performance degradations. It is worth noting that the absence of ground truth data, hinders to perform a complete validation of the outcome of their study.

Saljooghinejad et al. [81] present a black-box technique to evaluate the performance of cloud applications based on latency observation perceived by the end-users. In more details, the methodology they propose aims at estimating the maximum load an application can sustain (in terms of user requests), monitoring the end-to-end latency and identifying at which load the latency increases beyond acceptable levels. Although not directly interested in cloud-to-user latency, they monitor its trend to infer the state of the server in order to characterize the cloud system performance. To this end, they wisely mitigate the impact of network delay on end-to-end latency by comparing the latter to measurement results obtained with *tcpping* that do not depend on the server-side performance.

Venkataraman et al. [82] address the challenge of predicting the performance of applications deployed in public clouds under various resource configurations in order to automatically choose the optimal configuration (i.e., improving performance at reduced cost). They adopt a black-box approach which also measures the performance of the intra-datacenter network in terms of network bandwidth to collect model data. They evaluated their proposal in Amazon EC2 infrastructure and found that the available network bandwidth per core may change with the size of the VM. In this study the network measurement activities support the design of the model, but very coarse information is provided about the methodology enforced and its results. Detailed network performance collected in Amazon EC2 are omitted.

Li et al. [34] propose *CloudCmp*, a platform to compare cost and performance of different cloud providers. The comparative study proposed includes four popular and representative cloud providers: Amazon, Azure, Google, and Rackspace. The authors identify a set of common functionalities, and therefore propose a comparison in terms of elastic computing (i.e., start-up time), persistent storage, and networking services. For what concerns the networking services, they adopt standard tools such as *iperf* and *ping*, and consider the intra-datacenter, the inter-datacenter, and the cloud-to-user network performance.

They found that the intra-datacenter network performance may heavily vary among providers. In more details, Amazon and Azure reports intra-datacenter TCP throughput

around 800Mbps, on average, while lower performance (200 Mbps) have been observed for Rackspace, possible due to throttling or under provisioned networks. Larger variability was observed for Amazon. In terms of latency, all datacenters achieved low round-trip time ($<$ 2ms) for all pairs of VMs tested.

The inter-datacenter analysis is limited to pairs of datacenters within the United States. Rackspace reported the worst performance (around 100 Mbps), while for both Amazon and Azure median TCP inter-datacenter throughput higher than 200 Mbps was observed. Authors found that the latencies between datacenters largely correspond to the geographical distance between them.

Finally, they also consider the optimal latency measured considering a set of distributed vantage points. The results reported that Amazon and Azure have similar latency distributions, worse than Google's, on average, but much better than the one measured for Rackspace, possibly due to the smaller number of datacenters composing the infrastructure of the latter. Moreover, Amazon reported the higher latencies for vantage points placed in South America and Asia, because of the absence of datacenters in these regions at the time of experimentations were performed.

The monitoring activities performed in the works discussed above are able to provide interesting insights into the infrastructure of the cloud providers analyzed. However, all of them share two main shortcomings: (i) they are limited to pure black-box analyses, as they do not consider any information about the design of the cloud infrastructure; (ii) they do not explicitly consider the impact that the controlled virtualized environment may have on the network performance and on the monitoring tools adopted, and therefore the reported results are subjected to potential misinterpretation. In the following, we detailed analyze the works that try to go beyond these limitations. We first consider the studies that try to unveil the design of public-cloud services and infrastructures. We then discuss the works that explicitly consider the impact of the virtualized cloud environment.

Unveiling the design of public-cloud services and infrastructures.

A very limited set of works, try to go beyond the pure black-box analyses of cloud systems. As the implementation of cloud-based systems and services usually hides implementation details, a number of studies is devoted to methodologies for obtaining useful information from public-cloud deployments, thus extending the information base available to general customers or to final users. This kind of information can be leveraged to

provide security assessments, improve application performance, and enhance cloud usage patterns.

Raiciu et al. [27] conducted a study aimed at discovering the network topology of the Amazon datacenters, and show—through simulation—how this kind of information potentially has a big impact in the optimization of the applications. Although intra-datacenter network topologies are kept hidden by providers for security and commercial reasons, they found that a great deal of information can still be inferred by leasing many VMs and running measurements from cloud VMs. In more details, they adopt well known network diagnosis tools such as *ping*, *traceroute*, and *iperf* to reconstruct the topology of one of the Amazon EC2 datacenters. Given a pair of VMs, they identify a number of possible situations: the VMs are on the same physical machine (2-hop distance, 4 Gbps throughput); the VMs are in the same rack (3-hop distance, same /24 address space, 1 Gbps throughput); the VMs are in the same subnet (4-hop distance, 1 Gbps throughput); the VMs are in different subnets (6-hop distance, 1 Gbps throughput). Based on the information-base gathered, they propose *CloudTalk*, a language designed to allow users to describe network tasks such that the network can accurately estimate their completion time, in order to enable application optimizations.

Similarly, *LaCurts et al.* [28] in their study analyze the path lengths obtained with *traceroute* measurements within Amazon and Rackspace datacenters in order to identify communication bottlenecks. For Amazon environment, they found the observations consistent with a multi-rooted tree topology. Moreover, observing a number of paths longer than three hops, they infer that a significant number of VMs is not located on the same physical rack. Experimental evidences obtained by applying the same approach to Rackspace infrastructure, suggest that the provider enforces specific techniques to hide topology informations. Authors also perform a number of throughput measurement campaigns, involving the two providers. For Amazon, these campaigns (performed leveraging *netperf* and 10-second long measurements and involving medium VMs) reported that most paths (roughly 80%) have throughputs between 900 Mbps and 1100 Mbps. On the other hand, little performance variation was observed in Rackspace, as every path has a throughput of almost exactly 300 Mbps. The authors propose then *Choreo*, a system to perform network-aware application placement. Choreo measures the network path between each pair of VMs to infer the TCP throughput between them and places the applications according to their requirements.

He et al. [88] provide an empirical measurement study of popular IaaS cloud providers (Amazon and Azure) that examines cloud usage patterns and identifies ways in which customers could better leverage IaaS clouds. This measurement study, combining data from a university packet captures[1], interrogation of DNS records for websites listed on the Alexa top 1-million list, and lightweight probing, profiles the deployment patterns observed for popular web services and uncovers that the analyzed cloud deployments are somewhat precarious, as the vast majority of websites—even the more popular ones—use only one cloud region. In addition to resiliency benefits, they found that multi-region deployments could increase performance in terms of latency and throughput.

Ristenpart et al. [89] study the vulnerabilities introduced by design approaches usually introduced by cloud providers. using the Amazon EC2 service as a case study. The authors perform an empirical measurement study based on *nmap*, *hping*, and *wget* to perform network probes to determine liveness of EC2 VMs. They show that it is possible to map the internal cloud infrastructure, identify where a particular target VM is likely to reside, and then instantiate new VMs until one is placed co-resident with the target. They also explore how such placement can then be used to mount cross-VMside-channel attacks to extract information from a target VM on the same machine.

It is worth noting that most of the proposed approaches gather additional information with respect to the ones available from interface exposed by the providers, (e.g., to infer the topology of public-cloud datacenters) have become harder to be enforced over the course of time, because of the countermeasures put in practice by providers that strive to obfuscate them.

Investigating the impact of virtualization on network performance and measurements.

Virtualization techniques enforced by providers in order to provide flexible and cost-effective resource sharing is known to possibly introduce penalties in terms of performance. A limited number of studies has investigated the impact of machine virtualization in networking performance, also focusing on public-cloud scenarios.

Whiteaker et al. performs experiments with two popular virtualization techniques, to examine the effects of virtualization on packet sending and receiving delays. Using a

[1]For the sake of accuracy, this work is not based on pure non-cooperative approaches, as packet captures at the border of a university site require collaboration from entities with privileged roles with respect to the general customers.

controlled environments the authors investigate the influence on delay measurements when competing VMs perform tasks that consume CPU, memory, I/O resources, hard disk, and network bandwidth. Their results indicate that heavy network usage from competing VMs can introduce delays as high as 100 ms to round-trip times. Furthermore, they found that virtualization adds most of this delay when sending packets, whereas packet reception introduces little extra delay.

Although not directly related to public-cloud environments, this work is worth to be considered as one of the first investigating the impact of virtualization on delay and delay measurements. Moreover, as the virtualization techniques investigated in controlled environment are (with some variations) popular tools in public cloud implementations, the issues raised by this work are general and also hold also public clouds.

Wang and Ng [26] present a measurement study to characterize the impact of virtualization on the networking performance of the Amazon EC2. They measure the processor sharing, packet delay, TCP/UDP throughput and packet loss among Amazon EC2 virtual machines. They found that even though the datacenter network is lightly utilized, virtualization can still cause significant throughput instability and abnormal delay variations. In more details, they report that small VMs always share processors with other instances, while medium instances get 100% CPU sharing for most of the cases. They correlate this finding to unstable network performance. They performed hundreds of experiments adopting synthetic traffic generation tools, and found that medium VMs can achieve similar TCP and UDP throughput, while the TCP throughput of small VMs is much lower than their UDP throughput. Thanks to further analyses, they identify the impact of CPU scheduling synchronization on TCP control dynamics as a possible root cause of the observed performance. End-to-end delay measurements reported that delays among VMs are highly variable, and possibly impacted by internal security management systems. For what concerns the packet loss estimation, experimental evidences reports that in virtualized datacenter environments, there are additional difficulties to infer network properties using statistics. Some valid assumption in traditional network environments may not hold in virtualized datacenters. The reported results have a number of implications for network measurement and experiments performed in the cloud as they may be heavily biased by enforced management mechanisms. Moreover unstable network throughput and large delay variations can also have negative impact on the performance of scientific computing applications.

Barker and Shenoi [90] conduct a study to investigate whether dynamically varying background load from such applications can interfere with the performance seen by latency-sensitive tasks, in terms of CPU, disk, and network jitter and throughput fluctuations seen over a period of several days. In their empirical study they evaluate the efficacy of Amazon EC2 (considering small VMs), for hosting latency-sensitive applications, aiming at quantitatively determining whether such background load can impact application performance on a cloud server and by how much. In their network benchmark, they used *traceroute* to measure round-trip times within Amazon EC2 infrastructure. They considered both the latency from their server to the next immediate hop as well as the sum of the first three hops. Their network-based tests suggest that a certain amount of latency variation is to be expected when running in a cloud-based environment, but do not implicate resource sharing as the cause of this variation.

As virtualization in public clouds prove to impact performance to different extents we believe that its effects have to be properly taken into account when adopting non-cooperative approaches.

Final remarks. Table 2.1 summarize the works that implement non-cooperative approaches to monitor cloud networks.

As playing a crucial role, the characteristics of public-cloud networks have been investigated by the recent literature in different ways in order to go beyond the poor information exposed by cloud providers. As shown in the table, unsurprisingly the distribution of the providers investigated by the existing works reflects their current adoption. Therefore, most of them deals with Amazon, Azure, or both. A limited number of works investigate the networks of Rackspace and Google clouds, possibly because of the lower adoption or the later public release, respectively. A very small number of studies takes into account more than one provider, however.

Most of the works analyze the intra-datacenter network—often limiting their scope to it—also driven by the need of specific classes of applications to know the performance of the network within the datacenters. In spite of the new architectural solutions [46] that leverage inter-datacenter networks to support applications having strict requirements in terms of bandwidth or latency, only few works investigate its performance for public cloud providers [34, 83, 48, 38]. To the best of our knowledge, in one only case [34] a study takes into consideration more than one network portion.

More in general, rarely these works are focused on the network itself. Therefore, often the network has been marginally investigated and only coarse-grained results are provided, thus ignoring the management solutions implemented by the providers, the impact of the virtualization on the measurement results, or the impact of the configuration choices available to the customer.

As a consequence, the picture of the cloud network stemming out from the literature appears definitely blurred, also presenting strongly conflicting results [37, 34, 28].

Table 2.1: Application-agnostic cloud monitoring studies. *Intra-DC*, *Inter-DC*, and *C2U* stand for *intra-datacenter*, *inter-datacenter*, and *cloud-to-user* networks, respectively. A:Amazon, Z:aZure, G:Google, R:Rackspace. + reports if the study proposes an Investigation of the Design of the provider infrastructure and/or of the impact of virtualization.

Reference	**Year**	**Provider**				**Network**			
		A	**Z**	**G**	**R**	**intra-dc**	**inter-dc**	**c2u**	+
Palankar et al. [79]	2008	✓						✓	
Wang and Ng [26]	2010		✓			✓			✓
Barker and Shenoi [90]	2010	✓						✓	✓
Li et al. [34]	2010	✓	✓	✓	✓	✓	✓	✓	
Schad et al. [37]	2010	✓				✓			
Iosup et al. [86]	2011	✓						✓	
Hill et al. [85]	2011		✓			✓			
Raiciu et al. [27]	2012	✓				✓			✓
Tudoran et al. [77]	2012		✓			✓			
Feng et al. [83]	2012	✓					✓		
Feng et al. [48]	2012	✓					✓		
Juve et al. [75]	2012	✓				✓			
Tomanek and Kencl [40]	2013		✓					✓	
LaCurts et al. [28]	2013	✓			✓	✓			✓
Garcia-Dorado and Rao [38]	2015	✓	✓				✓		
Mulinka and Kencl [80]	2015		✓					✓	
Saljooghinejad et al. [81]	2016	✓	✓	✓				✓	
Venkataraman et al. [82]	2016	✓				✓			

Chapter 3

The CloudSurf platform

In this chapter we introduce CloudSurf, a platform we have designed, implemented, and recently publicly released [91]. CloudSurf allows to monitor public-cloud networking infrastructures from the customer viewpoint through non-cooperative approaches, i.e. without relying on information restricted to the cloud provider or to entities playing a privileged role with respect to the provision of cloud services.

3.1 Motivations

A growing number of companies are riding the wave to provide public-cloud computing services, such as Amazon, Microsoft, Google, or IBM. These companies offer a wide range of services of different nature that continuously evolve over time, in accordance with the needs of the market and the technological progress. This situation leads to a problem of plenty, in terms of both available providers and range of services that each of them makes available at different cost. Since no detail about the strategies implemented to manage the network or its expected performance is usually exposed by providers, the customer has to face a number of practical limitations: (i) because of the lack of information about network performance or network management strategies, the customer is not aware of how network resources are allocated to the cloud services he/she is going to rent; for instance the customer is not aware of the variability of the performance of the network over time; consequently, the customer cannot properly evaluate the suitability of the cloud system for the application to deploy; (ii) the customer cannot perform informed choices (for instance, based on expected network performance) in selecting the cloud offerings from various providers; indeed, providers cannot be properly compared; (iii) for a given

provider, the customer is not able to understand how selecting more expensive services would lead to have better network performance (i.e. the customer cannot choose the service that better suites his/her needs among the wide range of options available for a given provider).

Adopting non-cooperative approaches is the natural solution to the described situation. The scientific literature provides a number of examples in this regard (see § 2.2). However, the outcome of these pioneering works is limited in scope for what concerns the performance of the network. In addition, the obtained results prove to conflict in some cases. The lack of knowledge about the specific conditions in which the analyses proposed in the literature have been performed makes these kind of analyses hardly repeatable. The different points in time in which these analyses were performed as well as the different methodologies and tools adopted do not ease the comparison, as providers' infrastructure rapidly evolves over time and virtualization proved to differently impact tools [26], respectively. Additional issues raise from the expertise needed in managing cloud services and configuring the tools to perform the experimentations.

We propose CloudSurf in order to overcome the limits of the state of the art. CloudSurf is designed to perform cloud network monitoring activities from the general customer's angle and aims at providing the customer with a means to easily investigate the performance of cloud networks on demand and according its needs.

3.1.1 Desirable properties

Considering also the limitations of the approaches already implemented for monitoring cloud networks, we can find the following desirable features for a platform with such aim.

- *No need for information restricted to privileged entities.* As already discussed, our platforms is thought to be fully oriented to the general customer. Therefore the approaches implemented do not require the cooperation of any privileged entity. As the monitoring activities do not rely on this advanced information, they have to primarily leverage active monitoring approaches, i.e. they have to inject measurement traffic into the network in order to monitor its performance.

- *Ease of use.* Cloud services are leveraged by customers with different backgrounds, not necessarily having an advanced expertise in network monitoring activities and its related issues. A desirable property of the platform is to be easy to use, such

that its features can be leveraged by the wide community of people interested in cloud network performance.

- *Comprehensiveness.* As the providers offer a rapidly evolving wide range of service with different characteristics and properties, it is desirable for the platform to possibly deal with the most popular providers and all the options made available by each of them. In addition customers may be interested in different properties of the networks (e.g., minimum bandwidth guaranteed, maximum throughput achievable, latency). Accordingly the platform has to allow customers to perform the experimentation suited for gathering the aspects of interest.

- *Ability to predict the cost of the experimentations.* Non-cooperative approaches require to interact with the provider as general customers do. This implies that the user of the platform is subjected to the pay-as-you-go paradigm, in accordance to the terms of contract of each provider. Cloud network experimentations could be very costly (especially when repeated analyses are needed or when high-rate traffic has to be generated) as providers usually charge customers not only for the computation or memory capabilities of the VMs, but also for the traffic generated by them. It is desirable that the platform would be able to estimate the cost that the customers would be subjected to.

- *Easy sharing of the results of the analyses.* According also to the cost of the experimentations, it would be desirable that the platform would provide an easy way to share the outcome of the analyses with the community of people interested in them.

3.2 Architecture and implementation

With the above desirable features in mind, CloudSurf is designed to transparently perform cloud network measurement activities. Thanks to CloudSurf a cloud customer can measure the performance of the cloud network in any scenario of interest and/or analyze the outcome of measurement activities with no specific expertise neither in managing cloud resources nor in utilizing monitoring tools. Hereafter we will refer to the customer utilizing the CloudSurf platform as *the CloudSurf user* or simply *the user*.

The architecture designed for CloudSurf includes all the basic components required to perform cloud network monitoring activities. These components are: (i) the cloud-

probes, (ii) the master, and (iii) the results repository. Figure 3.1 shows an overview of the architecture and its main components. The master is the entity in charge of orchestrating the overall monitoring process. The cloud-probes are the components in charge of performing the strict monitoring activities. At the end of the monitoring process, its outcome flows into the results repository in order to be shared with the community of users.

The CloudSurf platform has been implemented in Python and has been released under Affero GPL (AGPL) license [92]. It is designed to support cloud monitoring activities for the leading public cloud providers: Amazon and Microsoft (see § 1.1.2). In the following the components will be briefly described together with the functionalities they implement. When needed, practical code examples related to the implementations will be shown.

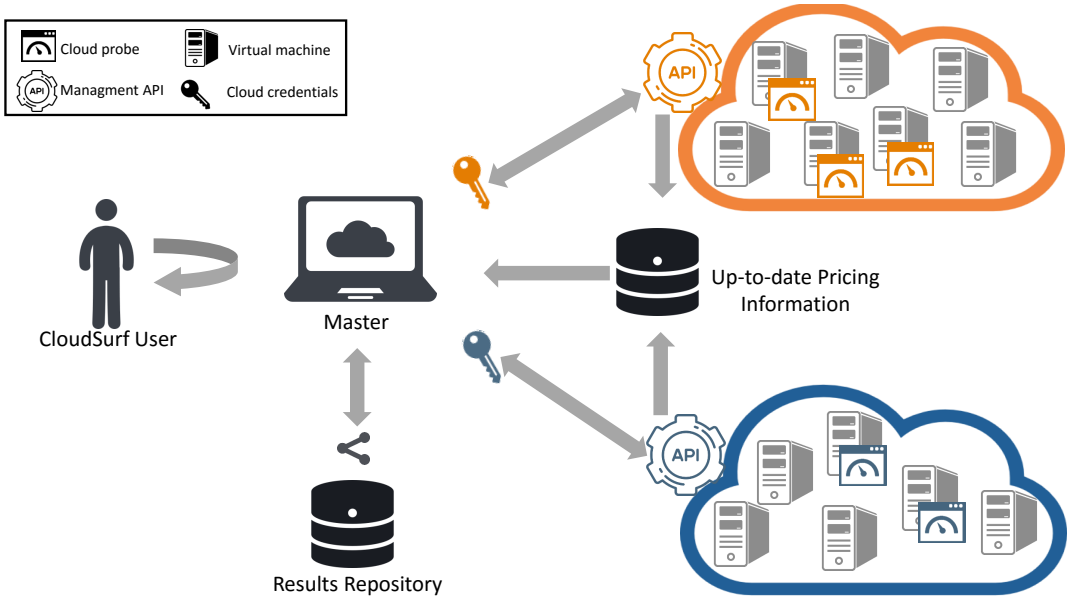


Figure 3.1: Architecture of CloudSurf.

3.2.1 Cloud probes

The cloud probes are remote measurement servers deployed on demand. They are deployed onto the cloud by the master when the user of the platform requires some experiments to be performed. The probes are thought to be passive entities, as they wait for instructions to be performed once deployed.

As the CloudSurf platform implements non-cooperative approaches through active

measurements, the cloud probes act as the endpoints of the experiments, and may play the role of both the sender and the receiver, as required by the master. In more details, the probes enable to utilize a number of tools to perform different kinds of experiments. Some examples of the tools integrated by the platform are *nuttcp*, *ping*, or *paris traceroute*.

In addition each probe implements a number of utility functions, allowing the master to cope with management issues (e.g., verify if the probe is alive, check the state of the probe, get the state of a certain experiment).

The remote measurement service exposed by each probe is implemented through the XML-RPC protocol. XML-RPC is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism. The parameter types allow nesting of parameters into maps and lists, thus larger structures can be transported. Therefore, XML-RPC can be used to transport objects or structures both as input and as output parameters. Identification of clients for authorization purposes can be achieved using popular HTTP security methods.

According to the proposed approach, the probes generate and receive two types of traffic: (i) *the control traffic*, exchanged between a probe and the entity that controls it, that is transported over HTTP and encoded according to the XML-RPC protocol; (ii) *the measurement traffic* generated by the probes and exchanged among them.

Experiments implemented by the probes can be classified as *one-sided* or *two-sided*. In the former case, the experiment does not require control over the destination, i.e. the sender does not require an active process listening at the destination side. In the second case, the experiment does require that at the receiver side, a process has been activated. Accordingly, in the case of two-sided experiments the entity that orchestrates the experiments has to coordinate the two probes involved, while in the case of one-sided experiments there is not such a need.

3.2.2 Master

The master is the entity in charge of orchestrating the experiments. According to the design of CloudSurf, it is the only entity with which the cloud customer has to directly interact. In more details, the master is responsible for the following tasks: (i) cloud environment setup and deployment of the monitoring probes; (ii) management of measurement experiments; (iii) experiment-cost estimation; (iv) results collection. The master allows the user to take advantage of the features implemented by the platform through a

Command Line Interface (CLI).

Cloud environment setup and deployment of the monitoring probes. The master is required to properly setup the cloud environment and allocate cloud resources in order to perform the monitoring activities required by the user. The CloudSurf platform leverages the services made available by providers to deploy monitoring probes within public-cloud datacenters taking advantage of the IaaS model. This layer guarantees the level of flexibility needed for the analyses based on non-cooperative approaches, as also demonstrated by previous works [26, 34, 37]. In particular, IaaS allows to deploy the monitoring probes on demand onto the cloud, taking advantage of VMs instantiated for the purpose.

In more details, in the setup phase, the master is in charge of managing user's credentials, configuring firewalling rules, launching VMs, and deploying the monitoring probes onto them. At the end of the monitoring activities the master decommissions the VMs and restores the state of the environment to carry it to its initial conditions.

In order to setup the cloud environment the master interacts with the provider through the cloud management API exposed. CloudSurf leverages the Python implementation of the interfaces made available by the providers. To interact with the provider through the interface, the master needs the proper level of authorization from the user. Also recalling that the ease of use is a design principle, CloudSurf needs the user to configure valid access credentials for the provider by simply storing them in a configuration file. This aspect is directly borrowed from providers' friendly interface, that allows to easily manage users, assigning them to groups, privileges, etc. For instance, in the case of Amazon, credentials have to be organized in a configuration file as reported in the following.

```
[Credentials]
aws_access_key_id = <ACCESS_ID>
aws_secret_access_key = <SECRET_ID>
```

Once the credentials have been arranged, the platform is able to autonomously configure the cloud environment. It is worth noting that some of these activities depend on the specific experiment that the user wants to perform, as they concur in the identification of the specific scenario in which network performance is investigated (see § 3.3).

Before deploying the measurement probes, VMs have to be lunched, properly configuring their firewall rules and the credential to access them.

The firewall must be configured in order to let both the data and control traffic pass.

To this aim with respect to the Amazon cloud environment, the platform defines a set of rules through a `security_group` object. An example is reported in the following.

```
security_group = \
    self.ec2_connection.create_security_group(
        "probe firewall",                    # name
        "data and control traffic allowed") # description

# ALLOW SSH incoming connections (TCP:22)
security_group.authorize(
    ip_protocol="tcp",      # protocol
    from_port=22,           # src port
    to_port=22,             # dst port
    cidr_ip="0.0.0.0/0")    # src address

# ALLOW XML-RPC control Traffic (TCP:8022)
security_group.authorize(
    ip_protocol="tcp",      # protocol
    from_port=8022,         # src port
    to_port=8022,           # dst port
    cidr_ip="0.0.0.0/0")    # src address

# ALLOW incoming measurement traffic (TCP:18100 and TCP:18101)
security_group.authorize(
    ip_protocol="tcp",      # protocol
    from_port=18100,        # src port
    to_port=18101,          # dst port
    cidr_ip="0.0.0.0/0")    # src address
```

Outgoing traffic is allowed by default. The examples shows how two rules are defined to let both the control traffic (TCP, port 8022) and the measurement traffic (TCP ports 18100 and 18101) pass. In addition the TCP 22 port also open in order to allow the master to also leverage the Secure Shell (SSH) protocol. The protocol and the ports associated to the measurement traffic change with the experiment and the tool leveraged.

In order to access the VMs via SSH indeed, for each experiment a couple of PEM-encoded RSA keys is created: the public one is deployed onto the VMs through the management API, while the private one is locally stored and adopted for interacting with the VMs for the following tasks. It is worth noting that thanks this step, afterwards the CloudSurf platform can interact with the VMs with no need to pass through the provider's API. This also means that the steps after the VM creation are the same whichever the involved provider is.

Once these tasks have been completed, the VMs can be launched. For each instantiated VM, the platform stores a data structure as the one reported in the following.

```
id                  # The unique ID of the VM
group               # The security group associated with the VM
public_dns_name     # The public dns name of the VM
private_dns_name    # The private dns name of the VM
state               # The instance's current state
key_name            # The name of the SSH key associated with the VM
instance_type       # The type/size of the VM
launch_time         # The time the VM was launched
private_ip_address  # The private IP address of the VM
ip_address          # The public IP address of the VM
platform            # Platform of the VM (e.g. Linux)
```

No other activity can be performed before the VMs are in the running state. As the startup time may vary with the provider or the resources leased, the CloudSurf platform takes advantage of the *status-check* functionality made available by the providers to be sure that the VMs launched are actually running. When the status check returns a positive outcome, the probes can be deployed on the newly instantiated VMs, i.e. the tools needed are installed and configured on the cloud VMs, and the measurement servers are actually started.

Experiment management. Once the probes are available, i.e. the master can leverage the measurement services they expose, the experimentations can be actually launched. Each measurement campaign is identified by the following data structure.

```
campaign = {"sender": sender_VM,
            "receiver": receiver_VM,
            "experiment_list": [exp1, exp2, ..., expN]}
```

This structure contains all the information needed to define an experiment. The `sender` and the `receiver` keys identify the VMs involved in the campaign. The `experiment_list` parameter represents a list of experiments. Each experiment object contains all the information to execute a measurement activity. An example for an experiment aimed at investigating the TCP max throughput achievable and the network latency with the *nuttcp* tool is reported in the following.

```
tcp_throughput_latency = {
    "tool": "NUTTCP",          # tool
    "port": 18100,             # measurement traffic dst port
    "proto": "tcp",            # measurement traffic protocol
    "duration": 300,           # experiment duration (s)
    "bandwidth": 1000000000,   # measurement traffic bitrate (bps)
    "sleep_exp": 0}            # time to wait after the experiment (s)
```

Given a campaign, the master is in charge of orchestrating it, by scheduling the experiments, also according `sleep_exp` parameter, that defines the time to wait after each of the experiments. The master drives the sender and possibly the receiver VM, according to the fact that the experiment is two-sided or one-sided.

Experiment-cost estimation. Each experiment comes at a cost which the user is subjected to, according to the pay-as-you-go model. The platform offers an estimation for this cost before running the experiment, in order to allow the user to evaluate its economical feasibility.

This cost depends on the charges imposed by the specific provider. Two quotas can be identified: (i) the expense associated to the VMs leased; (ii) the expense related to the traffic transferred over the network. These two quotas are heavily impacted by the specific scenario of the experimentation, as the cost of the VMs leased varies with the type of the VM its characteristics. Because of the pay-as-you-go model, it is also impacted by the duration of the experiment. The cost of the traffic generated also varies with the characteristics of the experiment, as traffic transfer is usually charged based on traffic volume. For instance experiments aiming at investigating the maximum throughput achievable with TCP (e.g., based on *nuttcp*) may require more traffic to be generated with respect to experiments aimed at monitoring the latency of the network (e.g., based on *ping*), as the former need to inject full-size TCP packets at full rate, while the latter usually injects ICMP packets of few bytes at a lower rate. In addition providers charge differently the measurement traffic based on its destination. For instance, the traffic between two VMs in the same datacenter comes at a lower fee with respect to traffic between two distinct datacenters. Finally, in some cases, the actual volume of traffic generated in a certain time interval is hard to be accurately estimated, as it depends on the characteristic of the network going to be investigated, that is not known *a priori*. However its upper bound can be easily computed, as it depends on the duration of the experiment and on the traffic the probes are required to generate.

The general formulation implemented by CloudSurf for estimating the cost associated to an experiment is provided in the following:

$$Exp\_cost = \left\lceil \frac{D}{3600} \right\rceil * (C_{VM}^{sender} + C_{VM}^{receiver}) + R * C_{Traffic}$$

where:

- D is the duration of the experiment (in seconds),
- C_{VM}^{sender} is the (hourly) cost of the sender VM,
- $C_{VM}^{receiver}$ is the (hourly) cost of the receiver VM,
- R is the rate of measurement traffic (in GB/s),
- $C_{Traffic}$ is the charge that the provider imposes to the data transfer (€/GB).

The ceiling operator is justified by the fact that the VM usage is charged per hour or fraction. Control traffic is not taken into account by the formula as negligible when compared to measurement traffic.

In order to gather information about the costs imposed by the provider (i.e., the C_{VM}^{sender}, $C_{VM}^{receiver}$, and $C_{Traffic}$ parameters in the above formula), the master takes advantage of a cost-estimation module. This module looks up to the pricing web pages of the providers and retrieve information from there. The CloudSurf platform extended a pre-existing open-source module (namely, the *awspricingfull* project [93] that retrieves some of the information needed for Amazon).

Results collection. When a campaign is terminated, the master gathers all the results temporarily stored on the probes. As the CloudSurf platform allows to take advantage of a number of different tools—each with its own output format—the gathered results are forcedly heterogeneous. In order to solve this heterogeneity issue, the master relies on a parsing module, in charge of translating the raw output into an homogeneous format. In more details, for each tool supported, the master has a parsing routine that translates the unstructured heterogeneous source into a JSON-encoded homogeneous format. An example of the output of a campaign obtained—after the parsing routine has been run—is reported in the following.

```
[{
"start": 1452774556,
"proto": "tcp",
"exp_name": "nuttcp_1",
```

```
"duration": 300,
"camp": {
    "region_rcv": "us-east-1d",
    "size_snd": "t2.medium",
    "provider_rcv": "aws",
    "region_snd": "us-west-2a",
    "provider_snd": "aws",
    "size_rcv": "t2.nano"
    },

"info_snd": sender_info,
"info_rcv": reveiver_info,

"results": {
            "delay": {
                    "detailed": null,
                    "synt": {
                            "std": null,
                            "pctl5": null,
                            "min": null,
                            "max": null,
                            "pctl25": null,
                            "pctl95": null,
                            "median": null,
                            "pctl75": null,
                            "mean": 72.88
                            }
                    },
            },
            "tput": {
                    "detailed": {
                            "1452774556": 70.54241818181818,
                            "1452774557": 109.51876666666666,
                            "1452774558": 104.85843,
                            "1452774559": 108.00174,
                            ...
                            "1452774570": 108.52799
                            },
                    "synt": {
                            "std": 24.666100557004235,
                            "pctl5": 30.828042,
                            "min": 17.30131,
                            "max": 114.29306000000001,
                            "pctl25": 61.342079999999996,
                            "pctl95": 111.673483,
                            "median": 80.21698,
                            "pctl75": 94.89526500000001,
                            "mean": 78.4539640360123
                            }
                    },
```

```
            }]
```

For each campaign, the parser module generates an output file that consists of a list of experiments. For each experiment key details are saved, such as: the start time, the duration, the tool adopted. Finally for each parameter investigated with the experiment (throughput, latency, loss, etc.), both the instantaneous values (when available) and the synthetic information are stored.

After having parsed the outcome of the analysis, the master upload it to a community repository that stores all the information

3.3 Identifying scenarios of interest

Characterizing the cloud network performance through non-cooperative approaches is a very challenging task for the extremely high number of possible scenarios in which a cloud customer may operate according to choices done when setting up the cloud environment, i.e. in the deployment phase. In this section we introduce the factors of interest to identify the scenarios when characterizing the network performance with the CloudSurf platform. It is worth noting that all the factors considered are under the direct control of the customer, and therefore can be driven by the CloudSurf platform. No expertise in managing public-cloud services is needed from the platform user: the deployment choices are transparently enforced by the platform, according to the available provider interface and user's will.

This approach eases as much as possible the understanding of the precise conditions in which the analysis is performed and allows to identify the major factors impacting the performance perceived by the users, beyond guaranteeing the ability to replicate the analysis in exactly the same conditions. This also helps to understand the blurred image about the public-cloud network performance provided by the scientific literature, as the few available pioneering works marginally focusing on this aspect, operated in few limited scenarios, and reported conflicting results.

We believe that identifying these factors and their impact on cloud network performance represents an important contribution for the analysis of public-cloud networks. In the following, we discuss the deployment factors potentially having a major impact on the intra-datacenter network performance, first considering those common to the providers considered (§ 3.3.1), and then the provider-specific ones (§ 3.3.2). Finally we discuss also

the experiment-configuration factors (§ 3.3.3).

3.3.1 Common deployment factors

In this section we introduce the *common factors*, i.e. those factors that can be leveraged whichever provider is subjected to our analyses, as reflecting common practices enforced by any public cloud provider.

VM type and size. When deploying VMs onto the cloud, a customer can choose their type and size among the ones made available by the provider. In this way, users can take advantage of different preconfigured settings in terms of storage size, computation capabilities, and network performance.

VM type (also referred to as *family* or *series* in the Amazon and Azure documentation, respectively [19, 69]) indicates a family of VMs optimized for a given task (storage, computation, etc.) and has been introduced with the intent of easing the configuration of the cloud environment.

Once the type is selected, the customer can decide the size of the VM to further specify storage and computation capabilities. Each VM type includes one or more sizes, thus enabling the customers to adapt leased resources based on their workload requirements. Machine hourly cost changes according to both type and size.

While the documentation by both Amazon and Azure clearly describes the available resources in terms of memory and CPU, it omits details about network performance associated to the leased resources, as no quantitative information about the performance of the network associated to each type and size is provided. Seldom works in the literature explicitly consider the type and the size of the VM as having substantial impact on network performance.

Geographical region. When instantiating a new VM onto the cloud, the customer can choose among a number of different geographically distributed regions. Both the cloud providers considered have deployed a complex infrastructure distributed world-wide. Customers can select different regions to meet their own technical and legal requirements. Each region is associated to different datacenters claimed to be completely independent from the others.

As of March 2015, Amazon deployed datacenters in 12 different regions spread world-wide. Throughout the next year, the Amazon global infrastructure will expand with new

datacenters in 5 new geographic regions [19]. On the other hand, Azure infrastructure consists in datacenters spread in 22 geographically distributed regions, while 5 new regions have been announced [69].

It is worth noting that different regions (i) have been designed and launched at differing points in time and thus may leverage different technologies; (ii) may be subjected to different management strategies; (iii) are interested by different workloads; (iv) may be subjected to different fees.

Works in the literature either omit the region under test [28, 27] or report performance variations across regions [34, 37].

Table 3.1 summarizes the choices available to the customer according to the factors introduced.

Table 3.1: Common factors (VM type, VM size, and geographically distinct regions) and available choices for Amazon and Azure.

Factor	**Provider**	**Available choices**
VM types	Amazon (9)	General Purpose (T2, M4, M3); Compute Optimized (C4, C3); Memory Optimized (R3); GPU (G2); Storage Optimized (I2, D2);
	Azure (5)	General Purpose (A-series); Compute Optimized (D-series, Dv2-series); Memory Optimized (G-series); Storage Optimized (DS-series)
VM sizes	Amazon (8)	micro, small, medium, large, xlarge, 2xlarge, 4xlarge, 8xlarge (depending on the family);
	Azure (12)	1–8, 11–14 (depending on the series);
Region	Amazon (12)	Northern Virginia, Northern California, Oregon, GovCloud, Sao Paulo, Ireland, Frankfurt, Singapore, Tokyo, Sydney, Seoul, Beijing;
	Azure (22)	Iowa (2), Virginia (3), Illinois, Texas, California, Ireland, Netherlands, Honk Kong, Singapore, Tokyo, Osaka, Sao Paulo, New South Wales, Victoria, Pune, Chennai, Mumbai, Shanghai, Beijing;

3.3.2 Provider-specific deployment factors

As the services offered by the different providers are backed by different implementations and technologies, the service interfaces exposed may slightly differ. Accordingly, some of the factors we have identified are provider-specific.

Availability zone (Amazon). Inside each Amazon region, the customer can choose among multiple *availability zones* (i.e. different locations advertised to be interconnected each other through low-latency links), opening the possibility of designing robust applications able to overcome potential zone fails [19]. Azure's customers cannot leverage a similar option. Indeed, differing sites in the same geographic location may exist for Azure, but they are considered as part of differing regions (see Table 3.1).

Results in the literature are conflicting about the impact of availability zones on the network performance [37, 34].

Configuration (Azure). During the deployment process, the Azure customers can select specific configurations for their VMs. In more details, they can deploy their VMs in the same *Affinity Group* or *Virtual Network* (VNET). According to Azure documentation [69], the affinity group is a way to "group cloud services by proximity to each other in the Azure datacenter in order to achieve optimal performance". VNETs, instead, are a way to allow services and VMs to "communicate securely with each other". Placing a VM in a given affinity group or VNET is optional and comes with no additional cost for the customer. A similar option is not available to Amazon customers.

3.3.3 Experimental configuration factors

In this section we discuss those factors under the control of the user performing active measurements during the strict experimental phase. These factors are related to how traffic is generated.

Transport protocol. CloudSurf allows to measure the network performance with both UDP and TCP traffic. On the one hand, UDP is typically used to analyze the performance of the raw IP traffic. UDP adds no closed loop-control, leaving the complete control on the generated traffic to the probe, no matter what the state of the network is. On the other hand, TCP traffic, which is governed by flow and congestion control, forces measurement traffic to be subjected to the status of the network path and pro-

vides information on the performance of the numerous TCP-based applications. Because of the highly virtualized environment, traffic leveraging different transport protocols may receive different treatments.

Packet size. Traffic generated with differing packet sizes may incur in different performance, as (i) larger packets may be subjected to fragmentation, thus generating additional overhead for the intermediate devices; (ii) different size may have a different impact on the generation capability of the probes when traffic is required to be generated at high rates. The impact cannot be easily estimated *a priori*, because of the lack of visibility into the specific system implementation. CloudSurf allows to generate traffic with different packet sizes, so to investigate the impact of this factor.

Communication channel. According to common practices, cloud VMs can be reached through a public or a private IP address [94]. Private addresses can be used only for communications between VMs deployed in the same datacenter. Public addresses, instead, allow the VMs to be reachable from the public Internet. Communicating through public addresses, however, could come at an additional cost, depending on the traffic volume.

In our analyses we measured network performance when the receiver VM is reached both through its private and public address. We refer to these logical communication channels as *private* and *public channel* respectively. It is worth noting that private and public channels may not correspond to the same physical path.

Probe placement and VM relocation. A cloud customer does not have neither visibility nor influence on which specific physical machine in the datacenter hosts her VMs. In our analyses, we also obtained empirical evidences suggesting the presence of mechanisms nullifying traditional topology-discovery strategies such as ICMP filtering. These mechanisms prevented us to gain knowledge about the relative position of the VMs through the adoption of tools like traceroute and ping. The only way we have to indirectly evaluate the impact of VM placement is to instantiate the VMs multiple times, i.e., we ask the cloud provider to terminate the VMs and recreate them from scratch. We refer to this process as *relocation*.

CloudSurf allows to implement relocation considering experiments as part of the same campaign or not.

3.4 Practical usage examples

To further clarify the working mechanisms of CloudSurf, we provide here a detailed description of how an experimentation is performed through practical usage examples and taking advantage of sequence diagrams for each of the phases—namely, the *initialization*, *setup*, *experimental*, and *termination* phases.

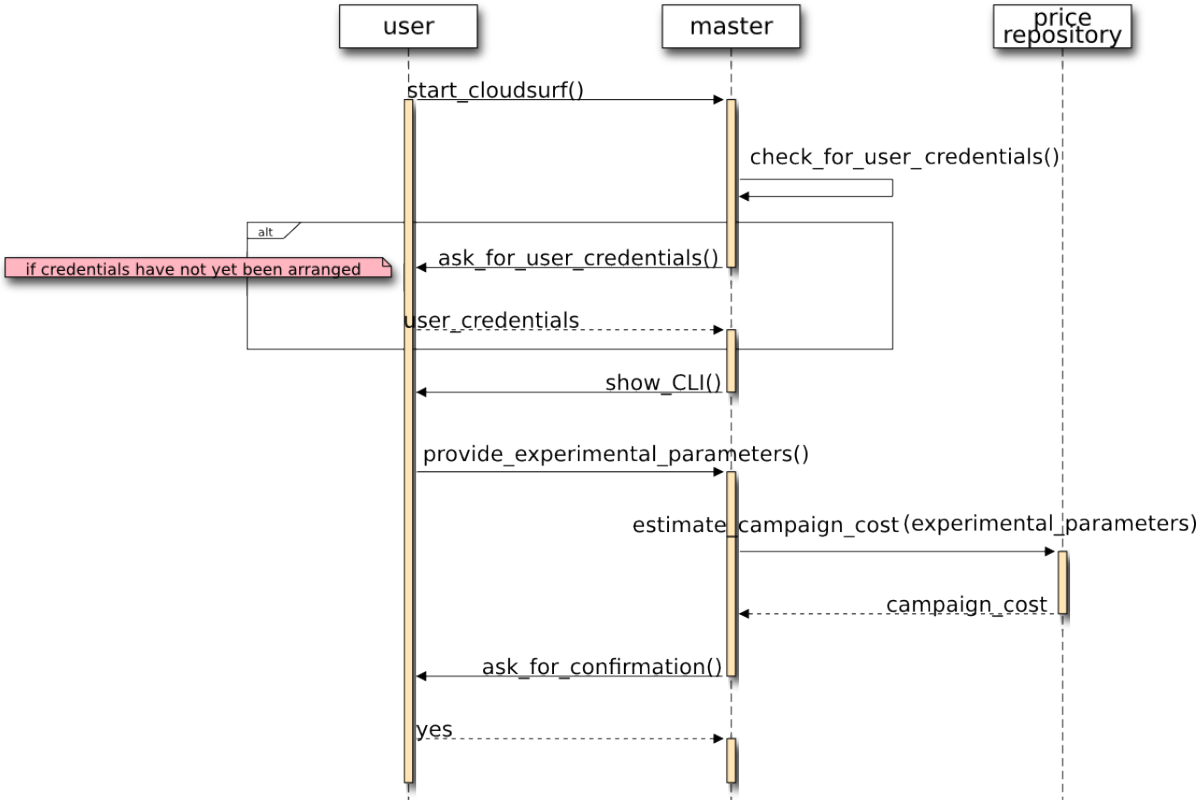


Figure 3.2: CloudSurf initialization phase.

The usage session starts with the initialization phase, when the user launches the CloudSurf platform by activating the master (see Figure 3.2). Right after the startup, the master asks for credentials (in order to gain access to public cloud services). Whether the credentials are correct, the user can access the features of the platform. Trough the CLI (see Figure 3.3) the user is asked to set all the factors for the experimental campaign, thus to unambiguously identify the scenario of interest (see §3.3).

On the basis of the parameters for the experimental campaign (its duration, the amount of traffic to be generated, the type and the size of the VM, the geographical region, the communication channel), the master takes advantage of the cost-estimation

(a) Home screen. (b) Experiment-selection screen.

Figure 3.3: CloudSurf command line interface.

module, which evaluates the cost which the user would be subjected to if the campaign would be run. The user is then asked for confirmation on the basis of the cost estimation, and if he/she, accepts to continue the setup phase starts.

During the setup phase the master interacts with cloud management API in order to prepare the cloud environment (see Figure 3.4). First, the master defines firewall rules, in order to let both control and measurement traffic pass through. Note that the rules depend on the experiment to be performed. Then the master ask for the creation of a couple of PEM-encoded key pairs, and locally stores the private one. At this point, the VMs needed for the campaign can be started, asking the management interface to deploy the previously created private key onto them. To reduce the time to wait and thus enhance the user experience, the sender and the receiver VMs are launched in parallel. After the VM instantiation has been asked to the provider, their actual availability is checked, as the subsequent steps require the VMs to be running. Once the VMs are actually running, all the code constituting the measurement server is moved onto them through the *ssh* protocol. An automatic configuration procedure is then run on each VM. Finally the

measurement servers are actually started via the *ssh* protocol.

After the setup phase, the experimental phase starts, according to the specifications provided by the user (see Figure 3.5). The master schedules the experiments on the basis of the request of the user, and when an experiment has to be launched, the master starts it via the XML-RPC protocol. Note that at these steps the interaction with the receiver is optional. When the experiment has been scheduled by the probe, the master is provided with a unique identifier which is used in the following steps. Once the experiment has been launched, measurement traffic between the probes is generated according to its type and characteristics. While the experiment is running, the master checks for its status (via the XML-RPC protocol and leveraging the identifier) until its termination.

When each experiment of the campaign is terminated, the termination phase starts. Here the master retrieves the experimental results from both the sender and the receiver. Right after that, it asks through the cloud management interface to terminate the VMs, as they are no more needed. The master then translates the heterogeneous results into a homogeneous JSON format through the parsing module. Finally, the master uploads the parsed results onto the community repository and shows their summary to the user.

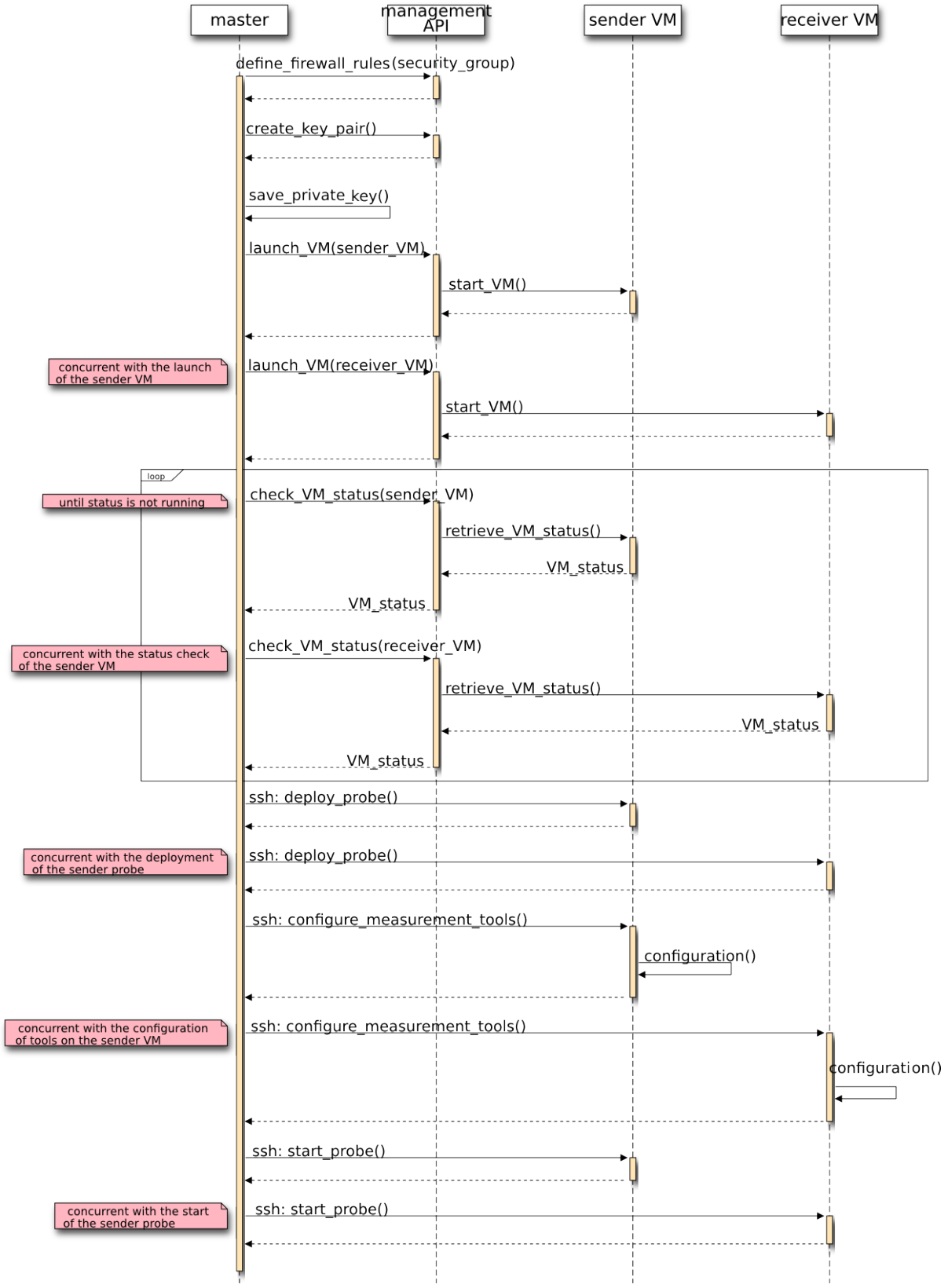


Figure 3.4: CloudSurf setup phase.

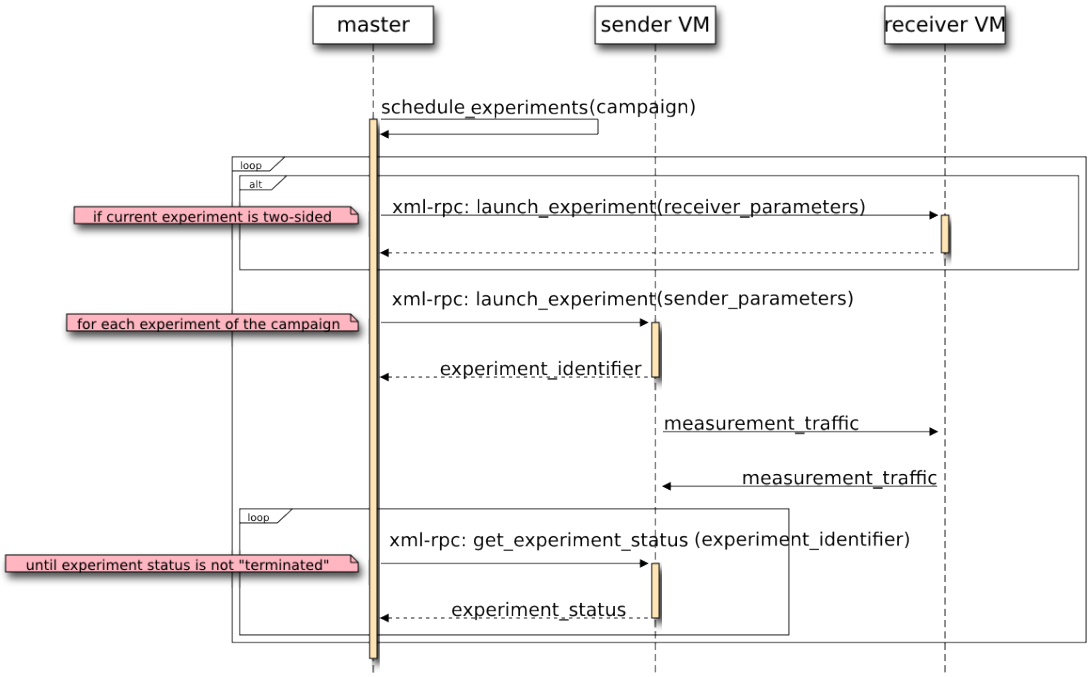


Figure 3.5: CloudSurf experimental phase.

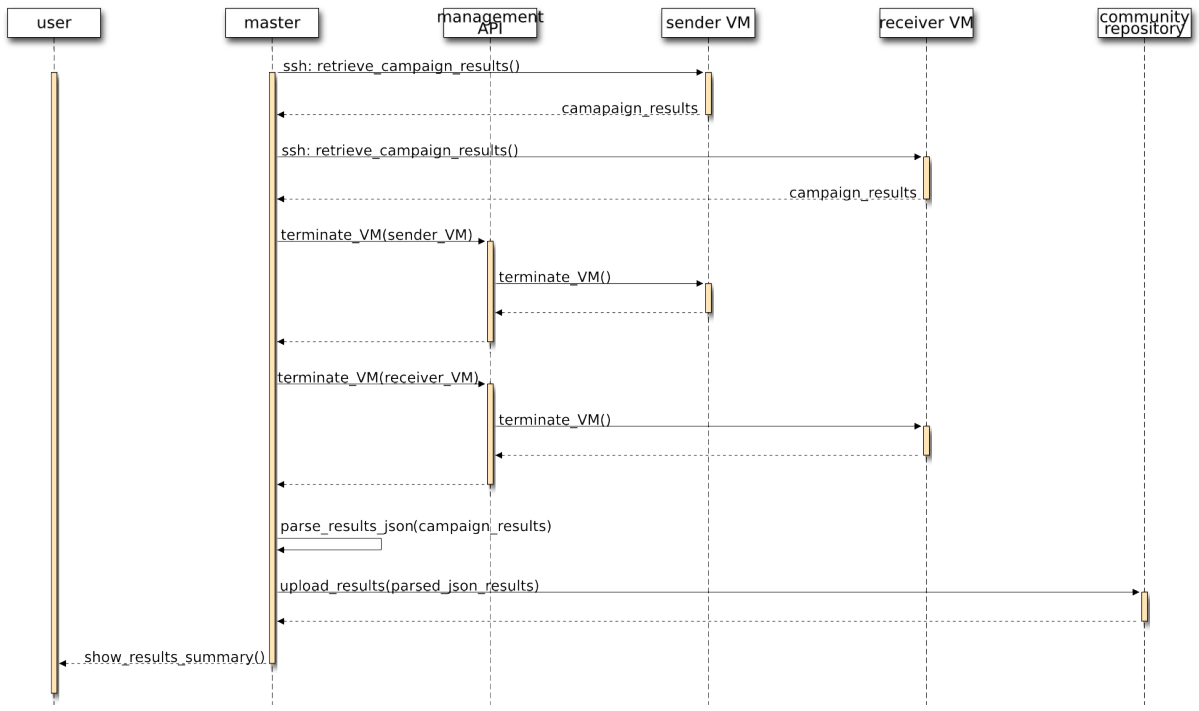


Figure 3.6: CloudSurf termination phase.

Chapter 4

Intra-datacenter network performance

The characteristics of the cloud systems, the strategies typically enforced by providers to manage resources in public-cloud datacenters, and the general trend of not advertising detailed information about expected network performance lead cloud customers to deal with network performance unpredictability. This is also testified by the outcomes of the analyses in the recent literature, which in addition reported significant variability in the perceived intra-datacenter network performance [51, 34, 37, 95]. The generated uncertainty hurts application performance and makes customer costs unpredictable to the customers, also causing revenue loss to the provider.

The high-level management perspective fostered by the cloud paradigm, led to an elementary interface between the customer and the providers, that has significantly contributed to the popularity of the paradigm thanks to its simplicity. According to it, customers simply ask for the amount of compute and storage resources they require (with a few additional details depending on the specific provider), and are charged on a pay-as-you-go basis. While attractive, this interface misses details about the cloud network however [51]. While all cloud providers provide high performance network connectivity to cloud systems, they seldom make any promises about network performance. Consequently, customers may suffer the absence of guarantees about the network.

The resources leased by customers inside the datacenter (e.g., the VMs) are interconnected by network infrastructures shared among all tenants. The expected performance and the management strategies of these networks are usually kept hidden by providers, which—due to security and commercial reasons—rarely provide detailed in-

formation about the organization of the network, its performance figures, and the mechanisms they employ to regulate it. The shared nature of the network and the highly dynamic and rapidly changing number of customers and users, may cause the perceived network performance to significantly vary. Indeed, the performance achieved by traffic between VMs may depend on a variety of factors outside the tenant's control, such as the network load or the placement of the tenant's VMs, and is further exacerbated by the oversubscribed nature of datacenter network topologies [35].

The problem of network sharing within these high performance environments has been largely investigated, as the shared networking environment dictates the need for mechanisms to partition network resources among VMs. Therefore today cloud providers commonly employ sophisticated virtualization techniques and strategies for sharing network resources among a high number of largely uncoordinated and mutually untrusted customers. However, datacenter management issues are not limited to ones inherent to networking, and the presence of constraints over multiple dimensions may also raise conflicts. For instance, proposed energy-management policies aimed at saving power costs, suggest to consolidate or migrate VMs into fewer physical hosts, in order to reduce the number of active physical hosts and save energy [96, 97]. However, these kind of approaches encourage resource sharing and lead to overload limited portions of the datacenter, thus possibly generating permormance issues that may involve also the network as a side effect. Hence, datacenter management policies are the results of strategies focused on the optimization of heterogeneous issues, which may also drive to conflicting solutions.

More in general, among the large body of proposals about the intra-datacenter network management, a clear agreement about what should be the desirable properties cannot be found [36]. On the one hand there is an agreement in broad terms that an approach should be *scalable* (i.e. it must work for very large numbers of VMs, tenants, server machines, and—potentially—switches, and should scale to higher bandwidth), *efficient*, (i.e. it should not require high hardware expenses or waste resource), and should lead to a *predictable* resource allocation. On the other hand, there is an open debate about other properties, often generated by the tendency to transfer desirable properties from the public Internet to the multi-tenant datacenter networks. For instance different opinions exist about the fact that cloud datacenter network should be *work conserving* or *fair*. In addition to the high number of proposed approaches to manage the intra-datacenter networks [98, 99, 100, 101, 51, 102], each provider may implement the management strate-

gies in its own way, also according to the needs dictated by the operating context i.e., network gear available, virtualization environment adopted, etc. [34].

As such, these factors lead to high unpredictability in the performance offered by the cloud network to a tenant, which in turn has several negative consequences, also for the provider. Variable network performance is one of the leading causes for unpredictable application performance in the cloud [37], which is a key hindrance to cloud adoption [25]. This affects a wide range of applications: from transaction processing applications [103] and MapReduce-like data intensive applications [104], to web applications that face users [86, 37]. Unpredictable network performance generates limited cloud applicability. The ability of supporting several class of applications is severely impeded by the lack of network performance predictability. The poor and variable performance of high-performance and scientific computing applications in the cloud is well documented [105]. The same applies to data-parallel applications that heavily rely on the network to ship large amounts of data at high rates [104]. Network also impacts cost predictability, thus discouraging cloud adoption. Customers pay based on the time they occupy their VMs, and this time is influenced by the network performance, as they implicitly end up paying when network driven tasks have been completed. It is worth noting how network performance unpredictability leads also to cost unpredictability, even when network communications are supposed to be free of charge.

In this context, monitoring activities allow to deepen the understanding of cloud systems. Indeed, they help to cope with the uncertainties generated by this unknown environment. As providers rarely expose detailed information about *how* they realize the services they offer, i.e., about their design and implementation, non-cooperative approaches are the only viable opportunity customers can leverage in order to perform informed decisions.

Non-cooperative monitoring activities allow to shed light on the performance unpredictability the customers have to usually face. Monitoring activities can be planned to to know in advance the performance of a specific cloud service, also allowing to estimate how it is subjected to changes in the future. Moreover, they can also help to identify the conditions that generate performance variability, thus identifying the most convenient scenarios to setup, the factors that have a major impact on the performance, and those under the direct control of the customer.

However non-cooperative approaches are challenging to be enforced, as they have to face all the issues generated by the unknown and someway hostile cloud environment

itself. Indeed, network virtualization and dynamic allocation strategies transparently employed by the cloud provider may strongly impact network performance measurements, whose results may appear misleading or incorrect. As we detail along this chapter, in our experimentations we have observed both traffic shaping policies causing transient fluctuation of the network throughput measured and limitations on the maximum rate at which the traffic can be delivered to a VM depending on its size. In addition, each provider reported its own peculiarities. In the following we provide evidences of such policies and limitations, also showing the impact caused on performance measurements.

In this chapter, we aim at improving the knowledge about both the performance of public-cloud intra-datacenter networks and the non-cooperative approaches to be possibly enforced, by focusing on the two leading public-cloud providers: *Amazon Web Services* and *Microsoft Azure* [17], hereafter simply referred to as Amazon and Azure. In more details we focused on the IaaS layer offered by these providers, namely *Elastic Compute Cloud* (EC2) for Amazon and *Virtual Machines* for Azure. Previous works only marginally analyzed the performance of the network offered by these services, by considering very few limited scenarios and providing preliminary results. To fill this gap, we provide a detailed characterization of the intra-datacenter network performance for these providers in terms of maximum achievable throughput, as it represents one of the most interesting parameters for the customers, impacting both QoS and costs [36, 28, 51]. We propose here the results obtained from more than 5000 hours of experimentation to characterize the network throughput offered by the two providers. We did not rely on the limited and coarse-grained information advertised and provide, for the first time in literature, the main contributions reported in the following.

- We identify and select a set of scenarios of interest for both providers, to perform measurements in cloud environments in order to obtain a significant characterization.

- Through our analyses, we improve the understanding of the complex policies and limitations of the intra-datacenter network and also characterize and quantify their impact on measurement experiments. We show how to properly configure and tune non-cooperative approaches to monitor public-cloud environments, in order to obtain significant characterization also with limited observation periods.

- We show how non-cooperative approaches are helpful to improve the knowledge of

the performance of the cloud intra-datacenter networks: we carefully characterize the network throughput in a large set of different scenarios (obtained varying parameters including the VM size, the geographical region, the transport protocol, the communication channel, etc.) and show the impact of the choices of the customers.

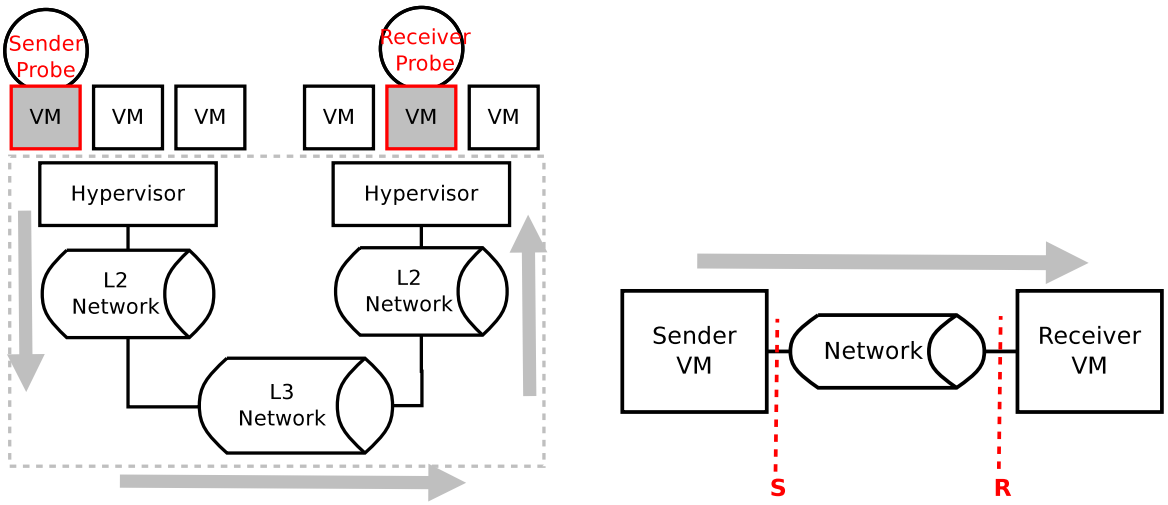
- From the results of the monitoring activities performed we derive guidelines that can help customers in performing advantageous deployment choices.

The chapter is organized as follows: § 4.1 introduces the cloud network architecture and its abstraction we refer to in this chapter; § 4.2 presents how the scenarios of interest have been selected for the experimentations; § 4.3 summarizes the results about the intra-datacenter network in the literature; § 4.4 presents experimental results about the impact of the virtualized cloud environment onto the measurement activities; § 4.5 offers an overall view of the performance of the intra-datacenter networks for the considered providers, which is extended by the more detailed results presented in § 4.6; § 4.7 presents the usage guidelines possibly derived from the analyses performed.

4.1 Reference architecture

In this section we provide the reference architecture adopted to enforce intra-datacenter network monitoring activities in public clouds through non-cooperative approaches. We aim at measuring the network performance between a pair of VMs under the control of the same cloud customer in terms of one of the most interesting metric, i.e. the maximum throughput achievable. To this end, we can leverage the CloudSurf platform (see Chapter 3) to deploy monitoring probes within public-cloud datacenters, taking advantage of the IaaS model. Being driven by CloudSurf, these probes consists in VMs instrumented with a standard operating system and all the necessary network measurement and diagnostic tools needed for estimating the network performance.

We measure the performance of the network generating synthetic traffic between the deployed probes. In the following, we use the term sender and receiver probe to identify the one in charge of sending and receiving the network traffic, respectively. Figure 4.1 reports the conceptual scheme we refer to in our experimental campaigns. On the basis of what discussed in § 1.3.2, the traffic generated by the sender probe normally first traverses the hypervisor layer at the sender side that enforces virtualization, as reported



(a) Cloud intra-datacenter networking architecture.

(b) Adopted abstraction.

Figure 4.1: Cloud network architecture and its abstraction. Two different monitoring points can be identified for each experiment, able to catch the dynamics of outgoing traffic at the sender (S) and of incoming traffic at the receiver (R).

in Figure 4.1a. Then it flows through L2/L3 devices and middleboxes part of the intra-datacenter high performance network. Finally, the traffic reaches the hypervisor at the receiver side before being delivered to the receiver probe.

It is worth noting that acting as cloud customers, we are not aware of the specific location of the sender and receiver probes, which may also be hosted and managed by the same hypervisor. In addition we have no specific information about the design of both the hypervisor and the L2/L3 network possibly deployed between the two probes. Furthermore, by adopting a black box approach, we consider the L2/L3 devices as well as the hypervisors as part of the network connecting sender and receiver probe (Figure 4.1b).

Basically, we consider as network all the logical and physical components interposed between the virtual network cards connecting sender and receiver VMs which host the monitoring probes. Note that this choice entirely fits the point of view of the general cloud customer who has no visibility on the cloud physical infrastructure, network internal dynamics, and cloud provider policies, despite their potentially heavy impact on the performance he/she perceives.

4.2 Scenario selection strategy

Cloud providers allow customers to highly characterize their environments (see § 3.3). The high number of available options translates into a large number of scenarios in which a cloud customer may operate. Sampling the space of possible scenarios is necessary when the goal is to provide meaningful and representative results of the intra-datacenter performance while keeping the complexity and cost of the analysis acceptable. Indeed, as shown in the following, a large number of experiments is needed to obtain representative results, in order to deal with both the complexity of the environment that is not known *a priori*. Moreover, the cost researchers have to face for these analyses—as they have to cope with the cost of both the VMs leased and the synthetic traffic generated—is also a limiting factor, often hard to estimate in advance. In the following the choices we made to deal with this issues are reported.

Our goal is to measure the network throughput between VMs placed inside the same region. Since this operation is costly, we are forced to select a subset of all the possible regions for our experiments. We have therefore selected four regions placed in different continents, to obtain a representative picture of the network performance of each cloud provider. These regions were activated at different times from 2006 to 2011 for Amazon, and from 2010 to 2014 for Azure. Hence, they also potentially leverage different technologies [37]. Namely, the selected regions are: *North Virginia (United States, US)*, *Ireland (Europe, EU)*, *Singapore (Asia-Pacific, AP)*, and *Sao Paulo (South America, SA)* for Amazon and *North California (US)*, *Ireland (EU)*, *Singapore (AP)*, and *Sao Paulo (SA)* for Azure. Table 4.1 summarizes the choices made.

In our experiments, we focused on *general purpose* VMs, which provide a balance of CPU, memory, and network resources, making them the best choice for many applications (small and medium-sized databases, memory-hungry data processing tasks, and back-end servers for SAP, Microsoft SharePoint, and other enterprise applications [19]). Although excluding from our analysis more expensive optimized VMs, we characterized the performance perceived by customers relying on general purpose VMs. Indeed, we aim at characterizing the performance according to the most popular choices done by the customers [106]. Therefore, we considered general purpose VMs, namely *m3* and *A-series* for Amazon and Azure, respectively. For both providers we considered three VM sizes: *m3.medium*, *m3.large*, and *m3.xlarge* for Amazon, and *A2*,*A3*, and *A4* for Azure. Here-

Table 4.1: Selected regions and notation adopted

Adopted Notation	Provider	Location	Year Launched
US	Amazon	North Virginia	2006
	Azure	California	2010
EU	Amazon	Ireland	2007
	Azure	Ireland	2010
AP	Amazon	Singapore	2010
	Azure	Singapore	2014
SA	Amazon	Sao Paulo	2011
	Azure	Sao Paulo	2014

Table 4.2: Selected VM types and sizes and notation adopted. Prices may vary with region

Adopted Notation	Provider	Type and Size	vCPU	RAM (GB)	Advertized Networking Performance	Cost €/h
M	Amazon	m3.medium	1	3.75	Moderate	0.0700–0.0980
	Azure	A2	2	3.5	N/A	0.0790–0.0980
L	Amazon	m3.large	2	7.5	Moderate	0.1400–0.1960
	Azure	A3	4	7	N/A	0.1590–0.1960
XL	Amazon	m3.xlarge	4	15	High	0.2800–0.3920
	Azure	A4	8	14	N/A	0.3170–0.3910

after, we respectively refer to them simply as *medium* (M), *large* (L), and *xlarge* (XL). As reported on the website [19, 69], this type of VM has fixed performance (in terms of CPU), which guarantees the absence of CPU resource-sharing and performance-variability phenomena that could impact the measurement process [26]. Table 4.2 contains more details on the VMs type and sizes adopted for both providers in our analyses.

For experimentations inherent to Amazon cloud, we have deployed VMs in different availability zones part of the same region, to evaluate the impact of this choice. Results regarding this aspect are reported when relevant.

For Azure, we considered also the impact of the configuration. In our analysis, we considered the following three configurations: (i) the two communicating VMs are deployed in the same (cloud-only) VNET (hereafter simply CFG_1); (ii) the two VMs are deployed

in the same affinity group (CFG_2); (iii) the customer provides no preferences (CFG_3).

Experimental configuration factors introduced in § 3.3.3 have been also considered, as detailed in the following sections.

4.3 Summary of the results in the literature

In this section we discuss the overall picture of the cloud intra-datacenter performance provided by the recent literature. As monitoring cloud performance has recently attracted great interest, some researchers analyzed network performance trying to go beyond the scarce information advertised by the providers. Unfortunately, these pioneering works adopted different methodologies and tools, reporting conflicting results that are hard to compare. Seldom the adopted methodology is described in enough details to allow the replication of the analysis. Very few of the possible scenarios have been tested, and the results are not supported by a proper analysis of the impact of the virtualization environment. This strongly limits the representativeness of the provided results.

In the following we provide a detailed review of the works that analyzed the intra-datacenter performance inside cloud environments, focusing on Amazon (§ 4.3.1) and Azure (§ 4.3.2).

4.3.1 Amazon

In the following, we review related works focusing on Amazon EC2.

Li *et al.* [34] proposed a non-cooperative approach to benchmark different clouds in terms of cost, VM deployment time, computation, storage, and networking. Regarding the network performance, they focused on both the intra-datacenter and the wide-area network, and measured throughput and latency using *iperf* and *ping*. For the Amazon EC2 intra-cloud network, the authors measured a TCP throughput in the range 600–900 Mbps. Due to the cost of the measurements, however, the authors also admitted that their results are achieved in few specific scenarios and cannot be considered general. Wang *et al.* [26] focused on the impact of virtualization on networking performance in public clouds and characterized it for EC2. They took advantage of *ping* and *ad-hoc* tools to characterize intra-cloud network performance using small and medium VM sizes. The authors measured significant delay variation and throughput instability. According to them, this variability seems not to be related to any explicit rate shaping enforced

Table 4.3: The overall picture of the intra-datacenter network performance of Amazon EC2 from the literature. NA stands for not available information.

Paper	**Year**	**VM type/size**	**Region**	**Measured throughput [Mbps]**	**Notes**
Li *et al.* [34]	2010	NA/NA	US (North California, North Virginia), EU (Ireland)	600–900	• Different throughput variability in different regions • No impact of availability zone
Wang *et al.* [26]	2010	NA/small NA/medium	US (North California), EU (Ireland)	400–800 (small) 700–900 (medium)	• 10-second long measurements
Shad *et al.* [37]	2010	NA/small	US (North California), EU (Ireland)	1.6–6.4(US) 3.2–7.2 (EU)	• Higher variability when VMs are placed in different availability zones • Higher variability in US region
Raiciu *et al.* [27]	2012	NA/medium	NA	1000–4000	• Available bandwidth related to mutual position
LaCurts *et al.* [28]	2013	NA/medium	NA	296–4405	• 10-second long measurements

by the provider. The paper reports maximum network throughput of 700–900 Mbps for medium-sized VMs with both TCP and UDP. The authors performed experiments over space (large number of VMs, short time interval) and time (reduced number of VMs, long time interval). Shad *et al.* [37] carried out a study on the performance unpredictability of AWS. Regarding network performance, the authors used *iperf* to evaluate maximum TCP and UDP throughput. They found that networking performance (available bandwidth intra- and inter- availability zone) ranges from 200 to 800 KB/s (1.6–6.4 Mbps) in US datacenter and from 400 to 900 KB/s (3.2–7.2 Mbps) in Europe datacenter. The authors reported that the network performance is 9% higher for instances placed inside the same availability zone. It is worth noting that these values are strongly conflicting with those reported in previous studies. Raiciu *et al.* [27] used different tools (*traceroute*, *ping*, and *iperf*) to obtain a blueprint of the EC2 network performance and took advantage of it to properly deploy applications and optimize their performance. They reported evidences of paths between VMs of different lengths and with available bandwidth between 1 and 4 Gbps, depending on VM mutual position. Finally, LaCurts *et al.* [28] described an approach to improve application performance by deploying the applications on the nodes with adequate network performance. The measurement study performed by the authors to motivate their system is based on *netperf*. This study showed a large variability of network throughput measured with medium-sized VMs from Amazon EC2. Such parameter

varied between 300 to 4400 Mbps, and most of the measurments (80%) reported values between 900 Mbps and 1100 Mbps.

Table 4.3 provides the overall picture on the Amazon EC2 intra-cloud network performance provided by the literature.

4.3.2 Azure

Only few works analyzed the Azure public cloud and investigated the performance of the intra-datacenter network. We discuss them in the following. Hill et al. [78] measured the TCP network throughput among 10 VMs of small size reporting throughput values ranging between 80 and 800 Mbps. They also reported great throughput variability over time between fixed VMs. Tudoran et al. [107] considered 20 small- and 20 extra-large-sized VMs reporting that extra-large VMs did not exhibit the high TCP throughput variability exposed by small VMs. Finally, Li et al. [34] investigated also the intra-datacenter network performance for Azure, and reported that the TCP throughput between two VMs placed in the same datacenter is always close to 800 Mbps.

Table 4.4: The overall picture of the intra-datacenter network performance of Azure from the literature. NA stands for not available information.

Paper	**Year**	**VM type/size**	**Region**	**Measured throughput [Mbps]**	**Notes**
Li *et al.* [34]	2010	NA/NA	US (North California, North Virginia)	800	
Hill et al. [78]	2010	NA/small	NA	80–800	• variability over time between fixed VMs
Tudoran et al. [107]	2010	NA/small, NA/extra-large	NA	664(S) 812(XL)	

More in general, all the works discussed are affected by a set of severe limitations, as report in the following.

- None of these works aimed at characterizing the intra-cloud network performance as a primary goal.
- All of them only marginally or preliminarily analyzed the network performance considering a limited number of scenarios in which a cloud customer may operate.

- These works do not take into account the large set of customization that the provider offers. Therefore, the impact on the network performance of the decisions operated by the customers during the VM-deployment phase is today largely unknown.
- Moreover, the methodology adopted in these works is not thoroughly described making the results hard to compare. In addition, with the only remarkable exception of the authors of [26], the authors to these works did not consider the impact of the virtualization on the measurement results.

4.4 Tuning non-cooperative approaches

Virtualization techniques enforced proved to introduce significant performance penalties to applications [77], invalidate measurement outcomes [26], and compromise the interpretation of typical measurement metrics [108]. Synthetic traffic has been widely adopted in previous works [34, 26, 37, 27, 28] but the potential impact of the VM-generation capabilities has been neglected in such literature.

In the following we show through specific examples how the complex management mechanisms enforced by cloud providers may impact the performance perceived by the cloud customers and potentially tamper with monitoring activities performed in public clouds. We also show how these mechanisms if not properly took into account may compromise results previously proposed in literature. Accordingly, we can define how to properly tune non-cooperative approaches to properly perform network monitoring activities in cloud environments.

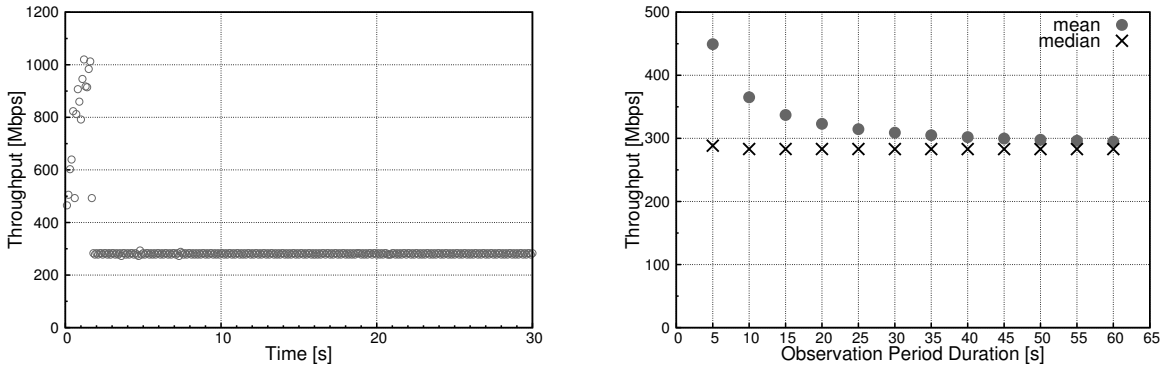
4.4.1 Identifying a proper metric for measuring network throughput

Measuring network throughput in the cloud can be very costly since this operation consumes computation and network resources that are charged by the cloud provider according to the pay-as-you-go paradigm. Furthermore, fast and accurate measurements are highly appreciated to guarantee high responsiveness to those frameworks exploiting network measurements [28, 27]. As a consequence, finding a good trade-off between accuracy and cost is of the utmost importance.

Properly knowing the impact of the resource allocation enforced by the provider proved to be a key requirement in our experimentations [26]. Our analysis showed also how the

metric chosen to evaluate the network throughput may determine misleading results. Indeed, tuning the duration of the observation period as well as selecting the right metric for the network throughput is further complicated by the effects of the mechanisms employed by the provider to reach the desired network resource allocation.

In the light of this knowledge acquired thanks to the outcome of non-cooperative analyses, in this section we show how properly selecting the metric can help measure network performance as it improves measure accuracy and allows to reduce the observation period, thus leading to also reduce the measurement costs. In the following we report how the mechanisms enforced by cloud providers to manage the intra-datacenter network may interfere with commonly adopted monitoring practices. As a clarifying example, we show how the bandwidth management mechanisms enforced by Amazon may impact monitoring results, whether not properly considered.



(a) First 30 seconds of a TCP intra-datacenter communication highlighting the presence of an initial transient spike. Similar spikes were observed in all the TCP and UDP communications monitored.

(b) Capturing the stable maximum network throughput achievable with different metrics.

Figure 4.2: Measuring network throughput in Amazon EC2. Typically, the network throughput reaches a stable value only after an initial transient period, likely due to the network resource allocation strategy employed by the cloud provider (4.2a). The initial spike impacts the accuracy of the network throughput measurements. The median value captures the stable value of network throughput much sooner than the mean value, requiring a shorter and cheaper observation period (4.2b).

An example of the evolution over time of the maximum throughput achievable between two Amazon VMs—observed with TCP traffic generated with the *nuttcp* tool—as well as its impact, is reported in Figure 4.2. Figure 4.2a shows how an intra-datacenter communication in Amazon cloud environment typically reaches a higher network throughput

during a first transient period, and then settles to a lower yet stable value. Applications using short-lived communications may obtain higher, although unstable network throughput. Note that this trend cannot be explained with TCP internal dynamics such as slow start or congestion control mechanisms. Indeed, a similar atypical behavior was *always* observed also in all the UDP-based communications we monitored. Hence, we consider this as a clear evidence of traffic shaping policies (e.g. token bucket), employed by the cloud provider as network resource allocation strategy.

The presence of this initial throughput spike cannot be ignored by the researchers willing to provide an accurate view of the network performance through non-cooperative approaches. Measurements performed during the initial interval are not representative of the expected performance over longer periods. The initial spike at the beginning of the communication may also explain the different throughput ranges of values reported in literature (see § 4.3.1).

Figure 4.2b shows how well mean and median values calculated over observation periods of increasing durations properly capture the maximum network throughput in the stable period. We have monitored the network throughput between medium-sized VMs over intervals of different durations. We have then computed the different metrics (i.e. mean and median) by only considering the throughput samples obtained during the first 5 seconds, first 10 seconds, and so on. The figure shows that the mean throughput value converges to the stable value much slower than the median one. This finding is consistent across all the experimental campaigns we performed, i.e., for different combinations of VM sizes, in different regions, for different types of traffic, over different channels.

According to these results, when aiming at measuring the stable throughput achievable between a couple of Amazon VMs, we have decided to report the median value of the samples over observation periods. This metric represents the *stable throughput* achievable in a communication between VMs deployed in the same Amazon region, filtering the noise caused by this initial transitory. In the following we refer to this value simply as the maximum throughput. Note that choosing the median is not *universally* the right choice, but it represents a valid option for Amazon. A similar trend has not been observed when testing Azure's infrastructure.

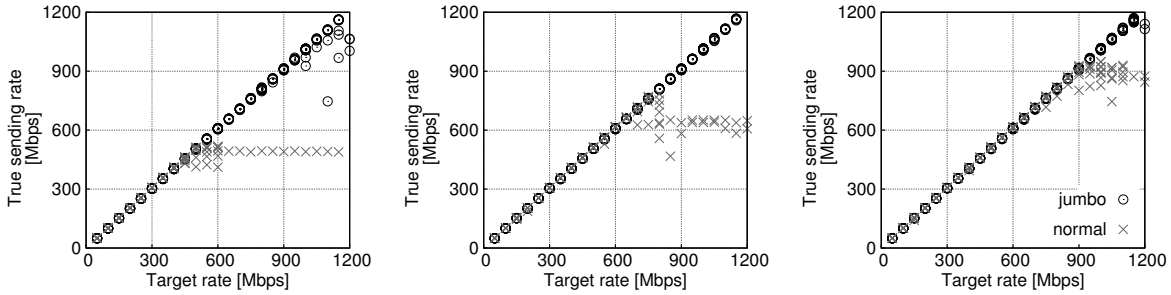
4.4.2 Investigating and understanding the impact of the virtualization

In a traditional environment, it is commonly assumed that the maximum network throughput achievable over a path can be accurately measured through the generation of synthetic traffic only when the computation capabilities of the involved end hosts are not a bottleneck for the communication. For instance, if the measurement tool adopted on the sender host is not capable to fulfill the remaining capacity of the network path because of the limited computation capabilities, the maximum available network throughput is incorrectly underestimated.

In the cloud the above assumption still holds. However it cannot be considered separated from the operational context, i.e. the impact of the virtualization layer over the communication channel under investigation and the measurement technique adopted must be taken into account. Therefore, when performing monitoring activities into the cloud, the effect of the strategy enforced by the specific provider and the direct impact of the vitualization layer have to be properly evaluated. It is worth noting how the specific mechanisms enforced are characteristics for the specific cloud provider.

Impact of VM size and packet size on traffic generation capabilities at sender side. We focus here on the impact of virtualization on VM traffic-generation capabilities at sender side. As an example of the above mentioned limitation, we have investigated it within the Amazon cloud environment.

We have instructed *nuttcp* to generate traffic at a given target rate and monitored the true sending rate (i.e. the rate of the traffic actually flowing into the network) to check whether the tool is able to sustain the target rate on a given VM. We have performed experiments with two different application-level packet sizes: 1024 bytes and 8192 bytes (hereafter simply *normal* and *jumbo* UDP packets). It is worth noting that TCP protocol does not suit this kind of analysis due to the congestion control mechanism that would force the sending rate to be limited by the bottleneck along the whole end-to-end path. We have performed 350 experiments 8-minutes long for each VM size in different regions, with target rates ranging from 50 to 1200 Mbps. Target rate and true sending rate are compared in Figure 4.3: VMs of any size are not able to inject traffic into the network at a rate higher than a given threshold (hereafter referred to as *cap*) when using normal packets. This cap proved to mainly depend on the sender VM and its size. In more



(a) M. (b) L. (c) XL.

Figure 4.3: Target rate vs True sending rate for different sending VM-sizes. When using normal packets (1024 B), the true sending rate does not overcome a cap value. This limitation is not observed in case of jumbo packets (8192 B).

Table 4.5: Cap in Mbps on true sending rate observed when using normal packets [$\mu \pm \sigma$].

Region	**M**	**L**	**XL**
US	489.1(±17)	747.3(±9.0)	944.1(±19.1)
EU	495.5(±20.0)	731.8(±10.3)	948.0(±15.3)
AP	485.5(±3.8)	730.2(±9.7)	925.1(±22.8)
SA	492.6(±5.3)	748.1(±24.5)	1018.3(±43.8)

details, the cap has proved to be very stable over time for a fixed VM: experimental results have shown that the Coefficient of Variation[1] (CoV) of the cap is always smaller than 2% for any observation period up to 72 hours. However, relocating (i.e. destroying and re-creating) the VM, also in the same region and with the same size, may reveal different cap values. Table 4.5 reports aggregated statistics on the cap values for all the considered regions: larger VMs can achieve a higher value of the maximum true sending rate with normal packets, which can be explained with the resource partition enforced by the provider (e.g. higher computation capabilities to larger VMs). On the other hand, we have observed no cap on the true sending rate when using jumbo packets: in this case, the target rate is achieved by imposing a much lower load on the virtual CPU.

Figure 4.4 reports the distribution of the cap values for the EU region (Ireland) with VMs relocated several times. In this region, medium, large, and xlarge instances are subjected to a cap imposing a maximum throughput of 495.5, 731.8 and 948.0 Mbps on average, respectively. Interestingly, although Amazon advertizes that both medium and

[1]CoV $= \frac{\sigma}{|\mu|}$, is the ratio of the standard deviation over the mean. It gives indication about how much values fluctuate around their mean value.

large VMs receive *Moderate* networking performance [19], our results clearly show that large instances are allowed to inject traffic into the network at a much higher rate. We have also noticed a higher variability of the cap imposed to xlarge VMs, with 63% of large instances receiving a cap higher than 5% of xlarge VMs.

In summary, synthetic traffic generation capability of the adopted VMs should be carefully taken into account when the final goal is measuring the intra-datacenter network performance through non-cooperative approaches. We have observed that the adopted measurement tool on EC2 VMs is not able to generate traffic at the requested target rate when relying on 1024-byte packets. This is not true when using jumbo packets. The obvious conclusion might be the adoption of jumbo packets. However, we will see in the following that this choice can have a detrimental impact on the network throughput at the receiver side.

Impact of packet size and communication channel. We have discovered that packet size and communication channel (public or private) have an impact on the network throughput measured. Figure 4.5 reports how the network throughput measured at the receiver side changes when the true sending rate increases for all the nine possible combinations of sender and receiver sizes. In this analysis, we have instructed the sender probe to perform 8-minute long generations of UDP traffic for each target rate. We have

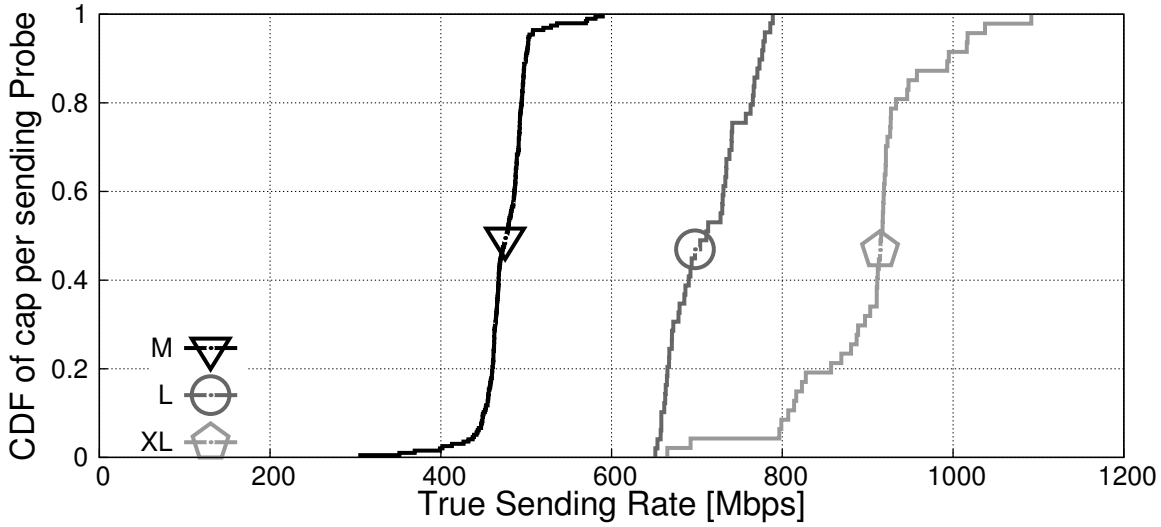
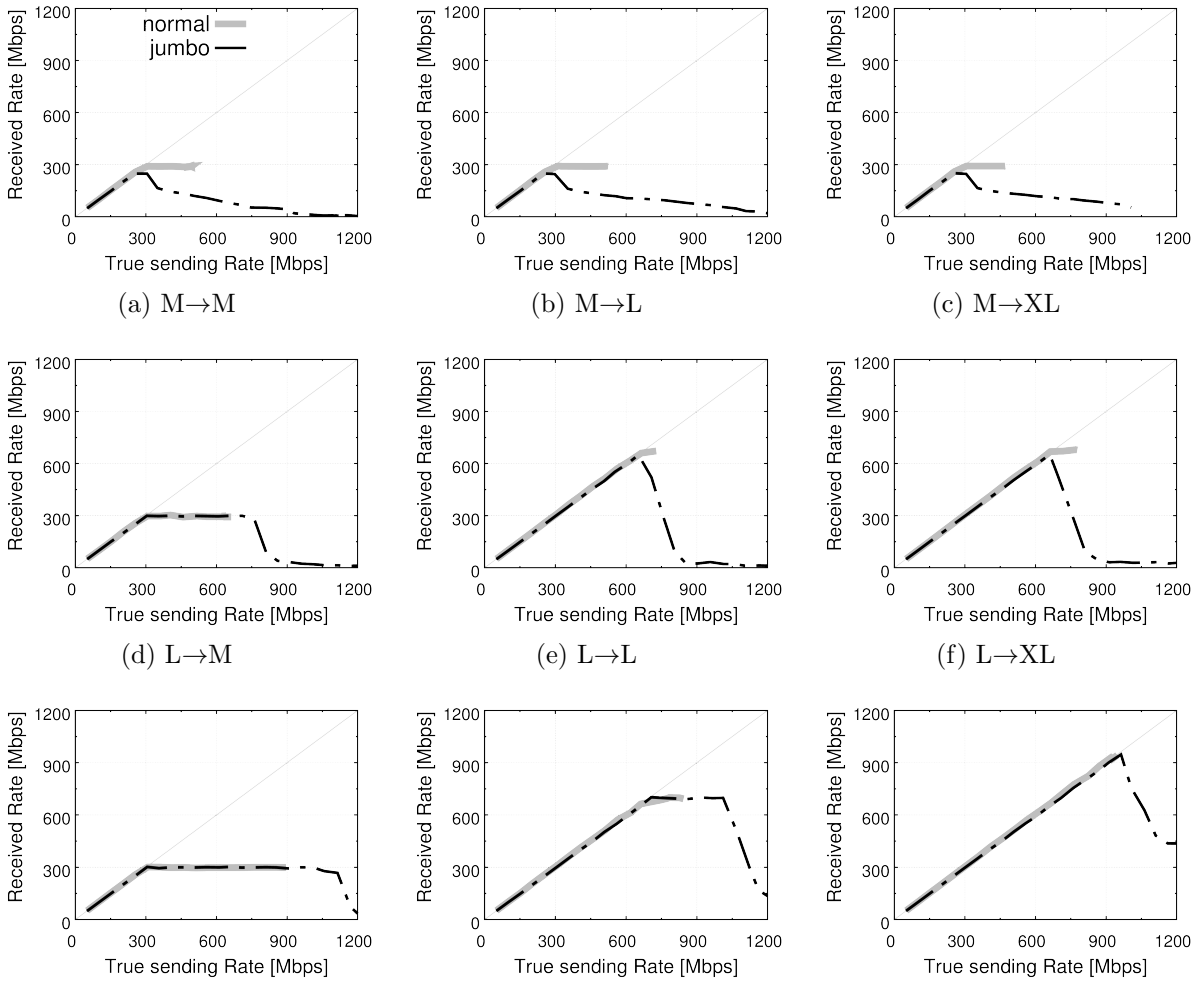


Figure 4.4: Cap value distributions for EU Region (Ireland). The values of the cap observed when using normal packets vary with the size of the virtual machine: the larger the VM size, the higher the true sending rate allowed.



(a) M→M (b) M→L (c) M→XL

(d) L→M (e) L→L (f) L→XL

(g) XL→M (h) XL→L (i) XL→XL

Figure 4.5: Maximum UDP throughput towards the VM public address. The network throughput at receiver side dramatically decreases at high true sending rates when using jumbo packets (dashed black lines). This behavior is not observed with normal packets (solid gray lines). *M:medium, L:large, X:xlarge.*

considered target rates ranging from 50 to 1200 Mbps. In each experiment, we have extracted the median value of true sending rate at the sender side and the median value of the network throughput at the receiver side. Overall, we have performed 350 experiments for each region by also relocating the VMs. The results reported in Figure 4.5 are related to the EU region (Ireland) but they are quantitatively and qualitatively representative also for the other regions.

The figure shows that the network throughput saturates to a maximum value independently from the packet size. Such value represents the maximum throughput an application deployed on a VM can achieve towards another VM in the same datacenter, through the network slice granted by the cloud provider. Interestingly, the figure also shows that jumbo packets allow generating traffic at higher rate but such higher rate dictates a drastic decrease of the network throughput at the receiver. The figure highlights a common pattern in the network throughput as a function of the true sending rate. The shape of the curves can be modelled through the following equations, describing the network throughput $R(x)$ as a function of the true sending rate x for each packet size:

$$\text{Jumbo packets: } R(x) = \begin{cases} x & x \leq \alpha \\ \alpha & \alpha < x \leq \beta \\ \Psi(x) & x > \beta \end{cases}$$

$$\text{Normal packets: } R(x) = \begin{cases} x & x \leq \alpha \\ \alpha & x < cap \end{cases}$$

Basically, the network throughput increases with the true sending rate up to a first value, which depends on the packet size. We have named this value *flattening edge* (α) because after this point, the throughput trend is typically flat or at least it does not increase, saturating to a constant value. After this point, the two packet sizes show significantly different behaviors. With jumbo packets the network throughput at the receiver side starts to strongly decrease after a certain value of the true sending rate (the phase represented by $\Psi(x)$ in the previous equations). We have named the value of the true sending rate after which the throughput starts decreasing as *penalty edge* (β) because after this value, the network throughput significantly drops. On the other

(a) Flattening edge – The network throughput saturates after this value of true sending rate. *N: normal packets, J: jumbo packets.*

		receiver		
		M	**L**	**XL**
sender	**M**	$251.9^{(J)}$–$307.6^{(N)}$	$251.5^{(J)}$–$307.7^{(N)}$	$252.0^{(J)}$–$307.4^{(N)}$
	L	$302.2^{(J)}$–$306.2^{(N)}$	$656.9^{(J)}$–$678.8^{(N)}$	$656.8^{(J)}$–$663.6^{(N)}$
	XL	$304.8^{(J)}$–$308.8^{(N)}$	$710.6^{(J)}$–$664.2^{(N)}$	$961.5^{(J)}$–$924.1^{(N)}$

(b) Penalty edge – The network throughput rapidly decreases for true sending rate higher than this value. This happens only with jumbo packets.

		receiver		
		M	**L**	**XL**
sender	**M**	302.9	302.2	303.1
	L	708.1	656.9	656.8
	XL	1111.4	1013.9	961.5

Table 4.6: Estimated values for the (a) flattening and (b) penalty edge. The tables show the average values computed over the experiments performed in different regions. Standard deviation omitted being negligible.

hand, this specific trend is not spotted when adopting normal packets. Indeed, we notice that the network throughput does not decrease after the penalty edge when using normal packets (see Fig. 4.5a for instance), even if high true sending rates are not achieved when relying on this kind of synthetic traffic. In the following we also provide the results of further analyses in order to explain this trend.

For normal packet traffic, we have experimentally observed that the evolution of the network throughput with the true sending rate is highly predictable. For jumbo packets, this property is experimentally verified up to the penalty edge: with higher true sending rate, we have always noticed a strong decrease of the network throughput but we could not derive an exact trend. The figure also clearly shows how the flattening and penalty edges depend on the size of sender and receiver probes: detailed values are reported in Table 4.6. The penalty edge values significantly increase for larger sender sizes. Considering the values of the medium-sized sender as a baseline, we have observed a growth for this threshold of more than 2× (3×) for large (xlarge) sender VMs. The flattening edge values, instead, seem to be determined by the smallest between sender and receiver VM size, i.e. the threshold increases only in case of larger size for both sender and receiver. The values of this threshold also depend on the packet size: especially for medium-sized sender probes (Figure 4.5, first row), we have noticed higher values for the flattening edge when comparing normal to jumbo packets, while differences are also noticed for xlarge

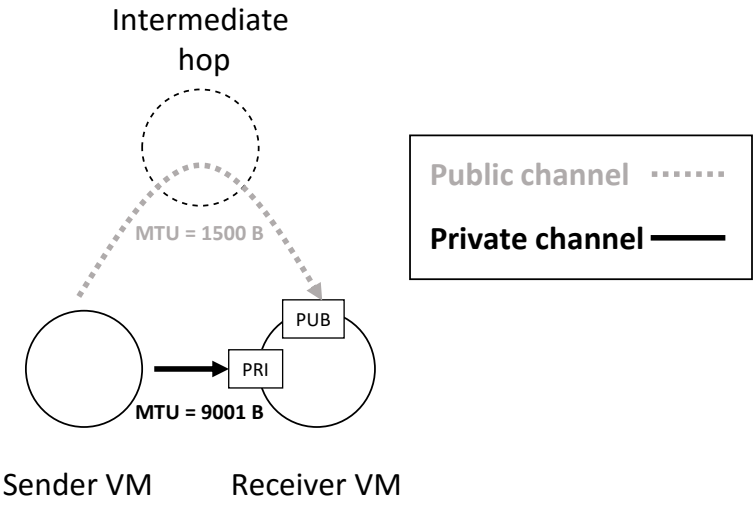


Figure 4.6: EC2 intra-datacenter paths. Traffic directed to the receiver VM crosses two different paths when directed to the private (PRI) and the public (PUB) IP address. In the latter case an intermediate hop is traversed.

sender probes communicating with either large or xlarge instances. Finally, Table 4.6 also reveals those sender-receiver combinations for which the flattening edge is equal to the penalty edge: these combinations correspond to the curves in Figure 4.5 where the network throughput starts decreasing immediately after the growing trend (Figure 4.5e, 4.5f, 4.5i). Note that this throughput decreasing trend has not been observed on the private channel. We dig into this phenomenon in the next section.

The impact of these results is twofold. On the one hand, researchers aiming at characterizing the performance of the cloud network may strongly underestimate the maximum throughput. This happens if they rely on the injection of UDP traffic at high target rate, as often suggested by classic methodologies. Indeed, injecting traffic at high rate always determines very low network throughput at destination, when issuing jumbo packets on the public channel. On the other hand, users and applications seeking the highest network throughput at destination have to carefully limit the sending rate.

Deepening throughput detrimental effect. We experimentally observed the maximum network throughput achieved on the public channel *always* strongly decreases when (i) using jumbo packets, and (ii) the true sending rate overcomes a threshold we named penalty edge. This phenomenon, however, was never observed when the communication was established through the private channel. We performed further analyses looking for

differences between private and public channels potentially explaining the causes of the observed phenomenon.

In this analysis, we employed the network diagnostic tool named *tracepath* [109] to infer the characteristics of private and public channels. Tracepath represents an evolution of the classic traceroute tool, providing additional path-related information. We took advantage of tracepath and performed multiple experiments (from 5 to 10) for each of the considered scenarios. Interestingly the outcome was the same across them. The results of this analysis are outlined in Figure 4.6. We noticed three main differences between public and private channels. First, we discovered that the network traffic flowing through the private channel *always* directly reaches the receiver probe whereas one intermediate network-layer device is *always* traversed on the public channel. This device is likely the middlebox in charge of translating public into private addresses. Differently from [27], we never observed paths connecting sender and receiver probes involving more than one intermediate hop. Several possibilities may explain this discrepancy including operational changes in the datacenter such a more efficient VM allocation strategy posing the VMs in the proximity of each other, as well as a change in the internal network infrastructure in terms of devices or configurations. Second, the Maximum Transmission Unit (MTU) is 9 KB on the path of the private channel—thus supporting jumbo frames—while it is only 1.5 KB on the public one. Third, and consequently, injecting jumbo packets on the public channel induces IP packet fragmentation. We experimentally observed that packet fragmentation occurs directly at the sender VM. These results were verified across all the tested regions.

Based on these findings, we can provide a possible explanation of the observed phenomenon. Using jumbo packets allows the sender to easily inject synthetic traffic into the network at the desired rate. However, the injected packets are fragmented on the public channel determining a potentially disruptive impact on the throughput measured at the receiver side. Indeed, each jumbo packet is fragmented in 6 smaller packets: losing even one of these fragments causes an entire jumbo packet to be discarded.

4.5 An overall picture of the achievable throughput

Thanks to the acquired knowledge, in this section we provide a picture of the network performance perceived by customers within public-cloud datacenters. As we aim at pro-

viding an overall view of the intra-datacenter performance in terms of the achievable throughput, we present in this section the aggregate results. Therefore in the following we show the average performance figures, and their variability across the differing datacenters available.

The experimentations behind these results, have considered the different scenarios stemming our from the combination of the factors previously introduced, in order to obtain results not biased by the choices performed. In more details in the following we consider the maximum throughput achievable in all the combination obtained by varying region and VM size (same size for sender and receiver VMs), for the differing transport protocols and considering the private communication channel (that is a reasonable choice for communications between resources within the same datacenter). For Amazon we also deployed VMs in all the different availability zones available in each region considered, while for Azure we considered all the configurations (same VNET, same Affinity Group, no specific configuration, see § 3.3.2) that the provider allows to enforce.

In the following, we report first the outcomes of the analysis that involved the Amazon infrastructure. Then we discuss the experimental results stemming out from Azure.

4.5.1 Amazon

In order to provide a general assessment for the Amazon intra-datacenter network performance, we have performed 10 10-minute lasting experiments for each combination considered, for a total of 480 experiments. According to § 4.4.1, we extracted the median as a significant synthetic metric for observing the stable throughput achievable. A combination is identified by VM size, transport protocol, and region. Across different experimentations we also randomly changed the availability zone in which the VMs is deployed, considering both the cases in which the sender and the receiver are placed in the same availability zone and the cases in which they reside in differing availability zones. At the end of each experiment, both the sender and the receiver VMs are terminated, thus each experiment refers to different pairs of VMs.

A summary of the results is reported in Table 4.7. Experimental results support the findings that follow.

- The VM size proved to be the factor having the major impact on the maximum throughput achievable. Interestingly, although Amazon documentation reports as

Table 4.7: Overall picture of the maximum stable throughput within Amazon datacenters. Mean and standard deviation refers to the average values for each region.

	TCP [Mbps] $\mu \pm \sigma$	**UDP [Mbps]** $\mu \pm \sigma$
M	298 ± 1	299 ± 0
L	696 ± 3	699 ± 1
XL	993 ± 8	996 ± 1

Moderate the network performance for both M and L VMs [19], our results show that L-sized VMs definitely receive more network resources than medium instances.

- Comparing Table 4.5 and Table 4.7, we can see that the traffic generation cap on the sender side does not significantly impact the network throughput measured, on average. Hence, the network throughput measured through synthetic traffic generation can be considered reliable since the sender machine proved not to be a bottleneck for the communication.

- For almost all the explored combinations of VM sizes, we have observed TCP performance to be equal to (or slightly higher than) the one obtained with UDP. Our analyses indicate that TCP (UDP) traffic between two M, L, and XL VMs can be delivered at 298 (299), 696 (699), and 993 (996) Mbps, respectively.

- The intra-datacenter network throughput is very similar across the different regions as reported by the small standard deviation values: the cloud provider seems to adopt a strategy to guarantee similar network performance to its customers in different regions. This is interesting considering that the hourly cost for a VM varies with the regions (there is a gap of +40% between the least and most expensive regions).

We can compare Table 4.9 and Table 4.3 to highlight a few important differences carefully considering that (i) previous works rarely provided details about which specific type of VMs they employed for the experimentation, and (ii) the features offered by the provider to the customer may have changed over time. We have experimentally observed that the size of the VM has a huge impact on the perceived network performance, an aspect underrated in [34, 37, 28] and only partially considered in [26]. We have measured

a much lower network throughput for medium instances (300Mbps) than the one reported in [28] (700–900Mbps), [27] (1000–4000Mbps), and [26] (296–4405Mbps).

A first possible explanation for this discrepancy is a change in the operational status of these datacenters, potentially caused by the deployment of higher-performance networking gear or by a variation of the resource allocation strategy. Another possible explanation may be spotted looking at the measurement methodology adopted. These works monitored the network throughput with experiments during only 10 seconds. As we already described in § 4.4.1, network throughput in Amazon EC2 is typically much higher and unstable during a first transient period of time. This throughput burst over short observation periods may heavily impact the accuracy of the measurements.

4.5.2 Azure

In order to provide a general assessment for the Azure intra-datacenter network performance, we have performed 10 10-minute lasting experiments for each combination considered, for a total of 720 experiments. A combination is identified by VM size, transport protocol, region, and configuration. At the end of each experiment, both the sender and the receiver VMs are terminated, thus each experiment refers to different pairs of VMs.

Table 4.8: Overall picture of the maximum stable throughput within Azure datacenters. Mean and standard deviation refers to the average values for each region.

	TCP [**Mbps**] $\mu \pm \sigma$	**UDP** [**Mbps**] $\mu \pm \sigma$
M	505 ± 151	494 ± 147
L	733 ± 37	708 ± 34
XL	961 ± 46	842 ± 24

A summary of the results is reported in Table 4.8. Experimental results support the following findings.

- Also for Azure the size appears to be the factors with the highest impact on the perceived performance. In general, deploying VMs of larger size leads to an improvement also from the network performance point of view, although this result is not clearly acknowledged in the Azure documentation.

- A high variability across regions exists (see the standard deviation in Table 4.8). This can be explained considering that datacenters, being launched at different point in time, may also leverage different techonologies having differing performance.
- TCP performs better than UDP, being able to deliver traffic at a rate that is 11, 25, and 19 Mbps higher for M, L, and XL VMs, respectively, on average.

4.6 A closer look at the achievable throughput

In this section, after the coarse-grained view we provided above, we show how the network performance perceived by the customer can vary by acting on the customization options that the customer has, i.e. leveraging the factors previously introduced.

4.6.1 Amazon

In the light of the previous findings, in this section we detail how the factors that can be leveraged by Amazon customers can impact the performance of the network. In more details, we show in the following how the measured performance may change when the communication takes place between two VMs of differing sizes, and passing from the private to the public communication channel. Table 4.9 shows a breakdown of the throughput performance.

Table 4.9: Maximum stable throughput for Amazon EC2 across different regions when the receiver VM is reached through the public or private IP address.

Sender to Receiver	**UDP**		**TCP**	
	Public $\mu \pm \sigma$	**Private** $\mu \pm \sigma$	**Public** $\mu \pm \sigma$	**Private** $\mu \pm \sigma$
M→M	291 ± 0	298 ± 1	293 ± 1	299 ± 0
M→L	291 ± 0	300 ± 2	293 ± 1	300 ± 0
M→XL	291 ± 0	298 ± 2	293 ± 1	298 ± 2
L→M	300 ± 1	300 ± 1	299 ± 0	300 ± 1
L→L	665 ± 13	696 ± 3	684 ± 1	699 ± 1
L→XL	670 ± 6	694 ± 4	685 ± 2	700 ± 1
XL→M	299 ± 1	300 ± 1	299 ± 2	301 ± 1
XL→L	708 ± 22	698 ± 5	699 ± 2	702 ± 0
XL→XL	897 ± 16	993 ± 8	939 ± 4	996 ± 1

This deeper analysis allows to identify several interesting aspects, reported in the following.

Impact of the communication channel and of the transport protocol. Traffic is exchanged at a slightly higher rate along the private channel compared to the public one. Also considering the extra-fee paid to use public channels, cloud customers should always prefer the private channels over the public ones, when possible. Also, for almost all the explored combinations of VM size, we have observed equal or higher network throughput for TCP compared to UDP.

Impact of the VM size. Introducing results of measurements performed between VMs of different sizes, we found that the maximum throughput is always limited by the minimum size between the sender and receiver. Therefore, when the network throughput is the most important aspect of interest, the best performance can be obtained with VMs of the same size.

Our analyses indicate that the cloud provider allows M, L, and XL VMs to deliver UDP (TCP) traffic at maximum 300 (300), 696 (700), 993 (996) Mbps respectively. Similarly, M, L, and XL VMs are allowed to receive UDP (TCP) traffic at maximum 300 (301), 708 (702), 993 (996) Mbps.

Network throughput variability over time. Additional long-range measurements, showed that the network performance variability is very stable over time. As an example, Figure 4.7 shows the results of a 24-hour long campaign, in which the maximum achievable TCP throughput has been measured through 4 5-minute long experiments per hour (EU region, M-sized VMs). As shown in the figure, the performance of two fixed VMs are very stable over time all over the 24-hour observation period. Note that the reported maxima reflect the presence of the initial spikes already discussed in § 4.4.1. A number of similar experiments have been performed varying the size of the VMs involved and the region in which the VMs have been deployed. Similar considerations about the performance variability can be drawn.

4.6.2 Azure

We deepen in this section the overall results about Azure intra-datacenter performance presented above. We discussed here the results obtained with 800 hours of experimenta-

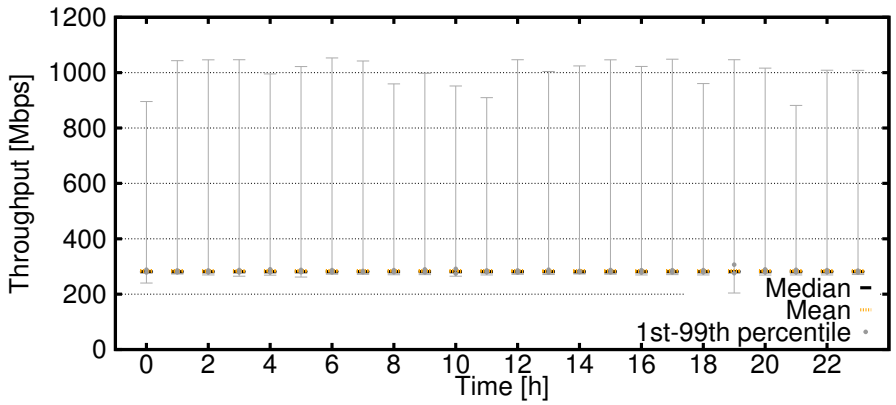


Figure 4.7: Variability over time of the throughput for two M-sized fixed Amazon VMs deployed into the EU region.

tion, aimed at investigating the higher variability observed for the achievable throughput within the Azure infrastructure.

In more details in the following we discuss the variability in terms of (i) throughput variability over time, (ii) throughput variability across different scenarios, and (iii) throughput variability in the same scenario.

Network throughput variability over time. The first aspect we investigated is whether and how the network throughput varies over time once the two communicating VMs are deployed on the cloud. With this analysis, we aim at verifying whether the performance of the network slice assigned by the provider to a customer dynamically changes over time. We discovered that the maximum intra-datacenter network throughput is very stable on average, while a slightly higher variability is observed when the instantaneous values are taken into account.

To carry out this analysis over large periods of time, we performed a series of 24-hour-long campaigns. In each campaign, (i) we created and deployed two VMs of the same size in a region and monitored them for 24 hours; (ii) every hour the sender VM transferred synthetic traffic to the receiver VM at the highest possible rate with nuttcp for 5 minutes; (iii) for each 5-minute-long experiment, we registered the network throughput achieved. We launched multiple campaigns by considering all the possible combinations of VM sizes, regions, configurations, and transport protocols.

To quantify how much the network throughput varied, we rely on the CoV. The CoV quantifies how much the samples are spread around the mean value: a very low CoV

implies that the standard deviation is negligible compared to the mean, thus underling a low throughput variability.

In *all* the long-lasting campaigns we performed, the estimated CoV was always below 0.01, i.e. the average network throughput between two fixed VMs computed over 5-minute-long experiments in 24 hours of experimentation proved to be very stable. Hence, no daily pattern was observed. These observations hold independently from the VM size, the cloud region, the configuration, or the transport protocol. An instance of 24-hour campaign for two medium-sized VMs deployed in the AP region is reported in Figure 4.8: note how the neither the average nor th median value significantly varies over time.

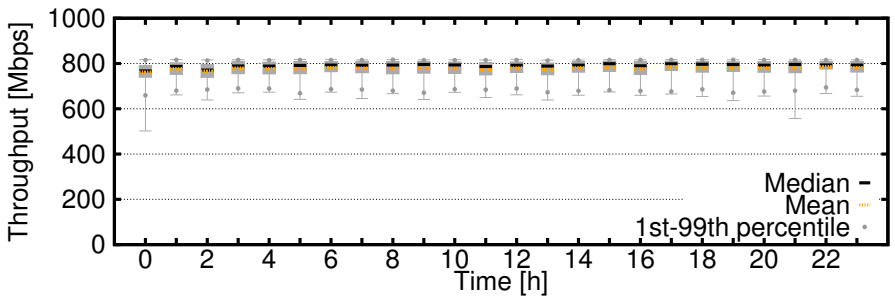


Figure 4.8: An instance of long-term campaign. The maximum intra-cloud network throughput between two medium VMs deployed onto AP region shows a slight variability in each 5-minute-long experiments over 24 hours. However, the average/median value per-experiment is strongly stable over time.

We also investigated the variability of throughput instantaneous values (1-second granularity), finding that in some well-defined cases, the variability of the throughput is slightly impacted by some of the factors considered. For each experiment we performed, the CoV computed over the instantaneous values was always lower than 0.1, i.e. the instantaneous values only slightly fluctuate around the average. In case of TCP traffic, the factors seem having a non-negligible impact on the network throughput variability over time as shown in Figure 4.9. In this case, we noticed how (i) the variability over time was slightly higher for larger VM sizes—see Figure 4.9a; (ii) deploying medium-sized VMs in regions such as AP and US exposed to higher throughput variability compared to SA and EU regions—see Figure 4.9b; (iii) a higher variability affected medium-sized VMs placed in the same VNET (CFG_1) or affinity group (CFG_2) compared to the case in which we let the cloud provider taking a decision about it (CFG_3)—see Figure 4.9c. The similar factor impact was not observed in case of UDP traffic.

According to these results, when a customer deploys two VMs on Azure, the communication between them can achieve a maximum throughput which on average is not subjected to significant changes over time, no matter the size, the region, the configuration, or the transport protocol. At the same time, instantaneous throughput oscillations exist and are larger for larger VM sizes, for specific regions, and in specific configurations. Since the average network throughput does not significantly change over time, measuring the throughput between two VMs even over a short period of time provides a good indication of the throughput that will be achieved in the future between the same VMs. We leveraged this finding to design the experimental campaigns discussed in the following.

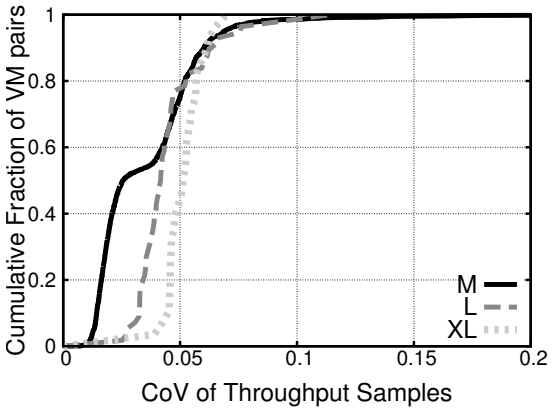
Network throughput variability across scenarios. The second aspect we investigated is whether and how the network throughput varies in different scenarios, i.e. when the customer operates different decisions during the deployment phase. To this end, we performed 5.5K short campaigns. In each campaign, (i) we created from scratch and deployed two same-sized VMs; (ii) we measured the network throughput between them for 5 minutes and registered the average throughput achieved with both the transport protocols; finally (iii) we terminated the VMs. We considered again all the possible combinations of sizes, regions, configurations, and transport protocols. We performed multiple campaigns for each combination, collecting multiple network throughput samples. Note that, as already described above, the network throughput measured over 5 minutes represents a good indication of the performance that will be achieved in the next 24 hours.

Table 4.11a and 4.11b extend the information contained in Table 4.8, by providing the average values computed over all the network throughput samples obtained in each scenario for TCP and UDP traffic, respectively.

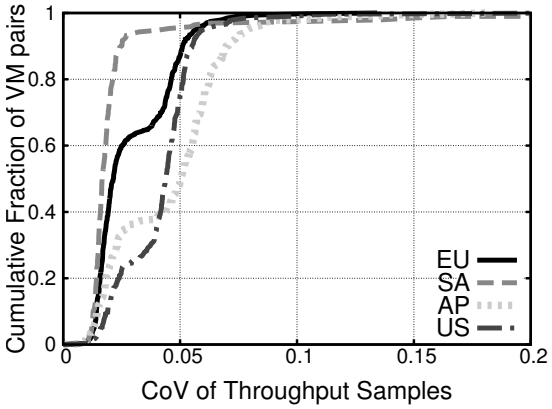
A closer look at these results suggests the following findings:

- All the factors have a non-negligible impact on the average network throughput—different values can be observed by reading these tables by rows (i.e. same region, but different sizes and configurations) and by columns (i.e. same VM size and configuration, but different regions).

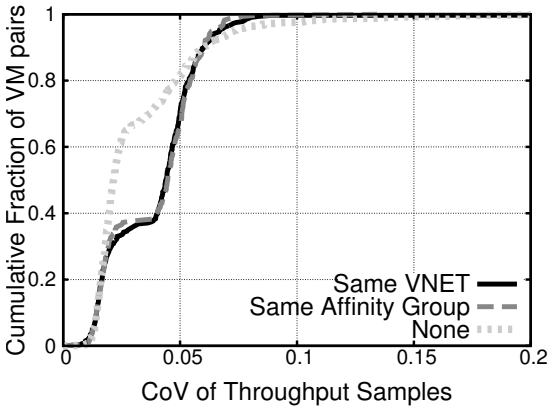
- Differently from what happens in traditional environments, UDP traffic does not clearly achieve better performance than TCP, although these experiments were performed in the same conditions and closely in time.



(a) VM-size impact.



(b) Region impact for medium-sized VMs.



(c) Configuration impact for medium-sized VMs.

Figure 4.9: TCP instantaneous throughput variability over time. Deploying larger VMs led to higher variability (4.9a). SA and EU regions (4.9b) as well as CFG_3 (4.9c) exposed lower variability.

Table 4.10: Average network throughput achievable in different scenarios [Mbps]. Underlined values refer to scenarios exposing high variability. We considered as high variable the scenarios associated to a standard deviation larger than 100 Mbps. For the remaining scenarios the standard deviation was found to be always smaller than 20 Mbps.

	M			**L**			**XL**		
	CFG$_1$	**CFG$_2$**	**CFG$_3$**	**CFG$_1$**	**CFG$_2$**	**CFG$_3$**	**CFG$_1$**	**CFG$_2$**	**CFG$_3$**
US	761.5	760.1	380.4	757.8	772.1	741.0	943.0	950.8	1121.6
EU	733.6	727.9	331.2	749.3	749.0	512.3	946.5	926.5	810.3
SA	255.6	256.7	237.0	734.0	758.9	736.3	946.6	944.5	931.3
AP	578.4	587.3	450.2	731.9	751.5	802.0	942.3	952.2	1112.9

(a) TCP traffic.

	M			**L**			**XL**		
	CFG$_1$	**CFG$_2$**	**CFG$_3$**	**CFG$_1$**	**CFG$_2$**	**CFG$_3$**	**CFG$_1$**	**CFG$_2$**	**CFG$_3$**
US	750.0	746.4	354.6	754.9	758.6	610.9	936.0	925.8	738.9
EU	734.4	723.6	315.5	768.0	760.0	435.8	882.3	845.9	686.4
SA	250.2	256.4	231.6	743.9	753.2	737.8	858.3	863.9	859.7
AP	565.2	568.2	437.1	740.1	747.8	687.0	900.6	903.7	703.2

(b) UDP traffic.

- CFG_1 (same VNET) and CFG_2 (same Affinity Group) typically guarantee higher network throughput than CFG_3 (no configuration), with only few exceptions such as the case of xlarge-sized VMs deployed in AP and US regions.

- In general, deploying VMs of larger size leads to an improvement also from the network performance point of view, although this result is not clearly acknowledged in the Azure documentation; also interestingly, this does not happen always: in both US and EU regions, when CFG_1 or CFG_2 is in place, medium VMs receive network throughput very similar to large VMs.

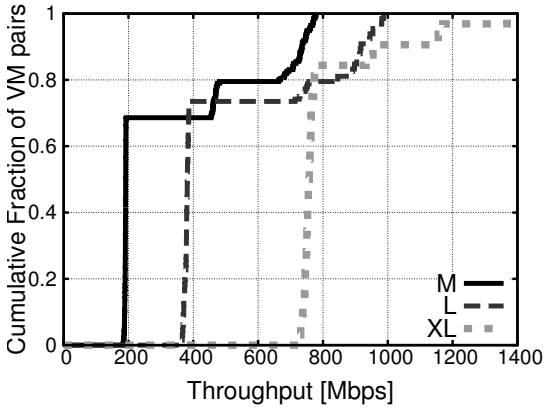
- the network throughput significantly varies across the regions, i.e. VMs of a given size and in a particular configuration may receive higher throughput in specific regions. This is true especially for medium VMs for which the throughput in SA proved to be much lower than what can be achieved in other regions.

When similar analyses were conducted in the past, only few factors were under the control of the Azure customers [34, 78, 107]. Nowadays, instead, the customer can highly

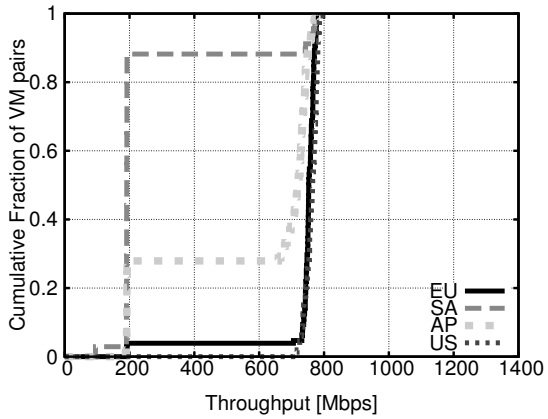
customize his/her cloud environment by acting on a wide set of factors. In this new context, we demonstrated that considering one or few combinations of factor values may lead to an incomplete image of the network performance a customer may experience. Since considering all the possible combinations is extremely costly both in terms of money and time, researchers willing to conduct similar analyses must very carefully design and describe the adopted methodology. This is essential not only to cross-validate and replicate the reported findings but also to understand their scope and validity. The methodology we proposed can be used to face the complexity of a similar task. For the Azure customers, these results clearly demonstrate that the decisions operated during the VM-deployment phase are very important at least from the network performance point of view. In this regard, Table 4.11a and Table 4.11b may support those customers willing to optimize their network performance by selecting the best combinations of VM sizes, regions, and configurations. For instance, we clearly demonstrated that it is convenient for the customers to deploy always the VMs inside the same VNET (CFG_1) or the same Affinity Group (CFG_2) since these configurations provided higher network throughput at no additional cost: this gain is more remarkable in specific regions such as US and EU.

Network throughput variability inside each scenario. In some of the considered scenarios, repeated measurements provided very different values of throughput. These scenarios are the ones for which Table 4.10 reports underlined values. In these cases, even if the customer operates exactly the same decisions during the deployment phase, the provider may grant very different levels of network throughput for her. This result is particularly interesting since according to the first outcome of this analysis (i.e. no performance variability over time for a fixed pair of VMs), an *unlucky* customer receiving low network performance should not expect any significant improvement over time. Some extreme cases where multiple experiments in the same scenario exposed completely different values of network throughput are reported in Figure 4.10. Note how the network throughput is quantized around few different values: although less marked, this behavior was qualitatively observed also for the other affected scenarios not detailed here due to space constraints. Figure 4.10a reports the network throughput measured within the EU datacenter across different campaigns where we adopted CFG_3 and two VMs of the same size: medium VMs received 200Mbps, 450Mbps, and more than 650Mbps in 70%, 10% and 20% of the campaigns, respectively. Hence, different customers deploying medium

VMs on EU region with CFG_3 may benefit from very different throughput values. A similar behavior has been qualitatively observed also for larger VMs. Interestingly, medium VMs received a network throughput higher than 75% of the campaigns involving large VMs for about 30% of the campaigns. Similarly, in 20% of the cases, large VMs achieved higher network throughput than 85% of the campaigns involving xlarge VMs: adopting larger VMs not always leads to higher network performance. Other interesting cases are reported in Figure 4.10b showing how a customer deploying medium VMs with CFG_1 in a given region may benefit from one of two very different network throughput values. In EU and US, the customer receives almost always high network throughput (760 Mbps). In SA, instead, the customer often receives low network throughput (200 Mbps) but in 10% of the cases, lucky customers can take advantage of 760 Mbps.



(a) Region: EU; Configuration: CFG_3; L4: TCP.



(b) Size: medium; Configuration: CFG_1; L4: TCP.

Figure 4.10: Network throughput variability inside the scenarios. A customer operating the same decisions during the deployment phase may receive very different network throughput.

Minimum network throughput guaraneteed. In the light of the observed variability, we also investigated the minimum achievable throughput a customer should expect from Azure when he/she adopts VMs of a given size. Table 4.11 reports the $1st$ percentile of the network throughput samples we registered across all the experiments for each VM size and for both transport protocols: empirical evidences suggest that independently from the cloud region or the configuration, a customer adopting larger VMs at higher fee can definitely achieve higher network performance in the worst case. Our experimentations reported that the providers guarantees, on average, *at least* 170 Mbps between two VMs

deployed in the same datacenter, independently from the region the customer relies on. Moreover, our results show how this value increases for larger VM sizes, reaching 327 Mbps and 468 Mbps for large and xlarge VMs, respectively. When restricting our results to specific regions (e.g., SA and AP) or trasport protocol (i.e. TCP) these values further increase.

Table 4.11: Minimum throughput guaranteed by Azure [Mbps].

	TCP			**UDP**		
Region	**M**	**L**	**XL**	**M**	**L**	**XL**
US	186.6	374.6	929.5	171.2	350.8	658.0
EU	185.4	364.1	728.5	170.5	327.5	468.8
SA	185.1	707.5	907.1	175.0	708.1	842.6
AP	186.0	718.1	935.0	174.9	519.1	614.9

4.7 Deriving usage guidelines

Intra-datacenter network performance details, although not being the only aspect of interest when leasing cloud resources, are extremely beneficial to the customers. Notwithstanding the interest customers may have in this kind of information, they are forced to make uninformed choices when leveraging cloud services because of the unknown management strategies enforced by the providers and the cloud interface they expose.

Non-cooperative approaches provide additional insights with respect to the scarce and coarse information about the performance figures publicly advertised. In this section we present a list of guidelines that can be devised from the information base gathered thanks to non-cooperative approaches, i.e. we show how the lesson learned through non-cooperative monitoring activities can be beneficial to customers.

4.7.1 Performing informed deployment choices

A first achievement reached thanks to non-cooperative approaches is the ability to shed light on the unpredictability of the intra-datacenter network performance. The outcome of the analyses presented above allows customers to perform their cloud deployment choices knowing in advance the expected performance. Indeed, the customer may leverage the set of factors we have identified to properly impact the performance provided by the network and its variability. Therefore customers may choose the size, the region, the provider, and

the configuration for the VMs to be allocated, according to the need of the application they are going to deploy.

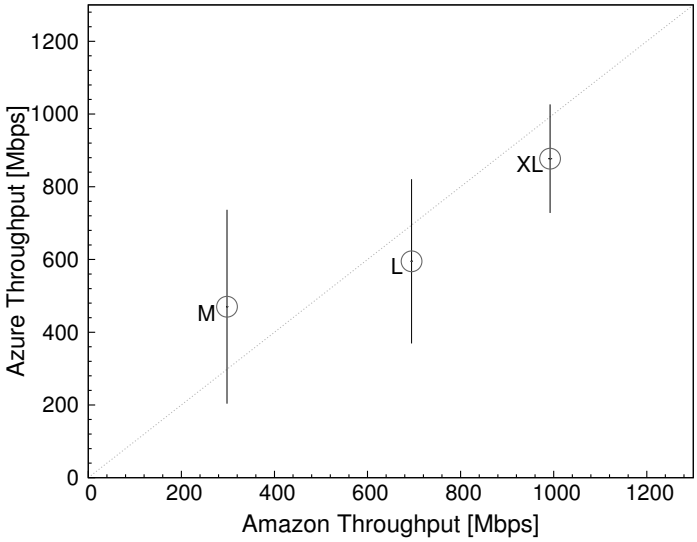


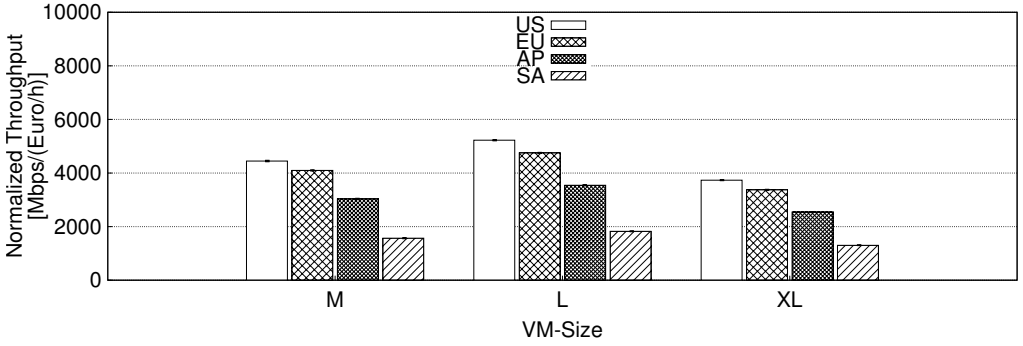
Figure 4.11: Expected Amazon and Azure intra-datacenter network performance in terms of achievable TCP throughput. Circles are centered on average values, while horizontal and vertical error bars report the standard deviation for Amazon and Azure, respectively. Amazon variability is negligible.

Figure 4.11 shows the expected performance for both the providers considered. The figure compares the average values for the throughput achievable with M, L, and XL Amazon and Azure VMs. As discussed above while Amazon intra-datacenter network performance is primarily determined by the VM size, the performance of the network within Azure datacenters is heavily impacted also by other factors such as the region, or the configuration. Therefore a markedly higher variability (larger error bars) can be spotted in the graph when considering the values averaged over all the scenarios considered. According to our results, medium-sized Azure VMs perform better than Amazon's (470 Mbps and 298 Mbps, respectively). Conversely, the same assumption does not hold for large- and xlarge-sized VMs, whose performance in terms of throughput is better in Amazon infrastructure (695 vs. 595 and 992 vs. 877 Mbps, for L and XL VMs, respectively).

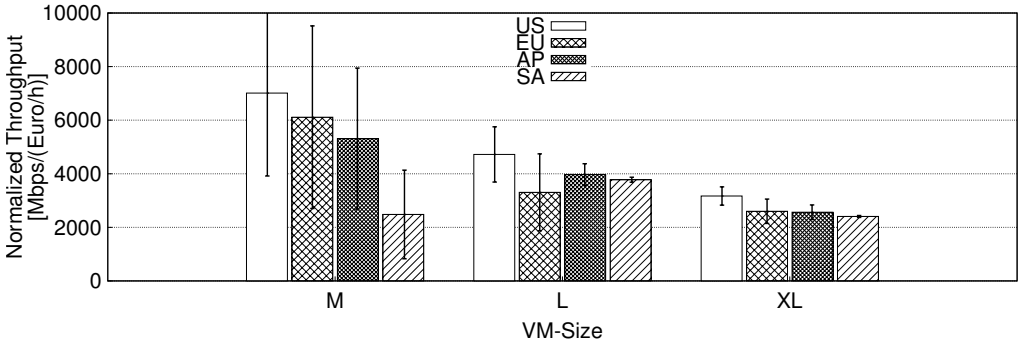
In the light of the generated knowledge, customers can choose among the possible offerings of the providers, based on their needs. It is worth noting that those discussed above are average values, i.e. have been obtained without considering the impact of the factors under the direct control of the customer, except the size of the VM.

Providers therefore offer services at different performance. As the fees associated to

these services are also of interest to the customers, because of the pay-as-you-go paradigm, in the following we extend our analysis in order to encompass also the economic aspect. To capture also the impact of the network performance on the costs—and thus compare differing performance figures coming at differing costs—we introduce the *normalized throughput*, defined as the ratio of the achievable throughput in a certain scenario to the hourly cost of a VM associated. The intent of this metric is to investigate the relationship between the network performance and the costs, i.e. to evaluate how network performance varies when the cost associated to the VM increases. It is worth noting that network performance comes in bundle with CPU and memory capabilities. Therefore the higher cost for larger VMs is due also to higher computational or memory resources. Here we focus on the network capabilities available to the customers. In other words, we assume the point of view of a customer primarily interested in network performance.



(a) Amazon.



(b) Azure.

Figure 4.12: Performance in terms of normalized throughput varies with region and VM size, for both providers.

Figure 4.12 shows how the normalized throughput varies with the VM size and the

region for Amazon (see Figure 4.12a) and Azure (see 4.12b). The figures show that the normalized throughput is higher for Azure in the case of M-sized VMs, on average, notwithstanding the presence of a higher variability. Compared to Azure, Amazon exposes better performance for XL VMs, with the only exception of the SA region. When considering L VMs, we have that EU and US region have better performance for Amazon, while AP and SA expose better performance for Azure, on average. Finally, for fixed size and provider, the US region has always better performance.

These results can be explained by (i) the different fees which customers are subjected to in different scenarios (impacted also by the different tax regimes which the providers are subjected to in the different regions) and potentially by (ii) unbalanced resource deployments, different technologies leveraged, or unbalanced catchment areas across different regions and scenarios, that may lead to different performance.

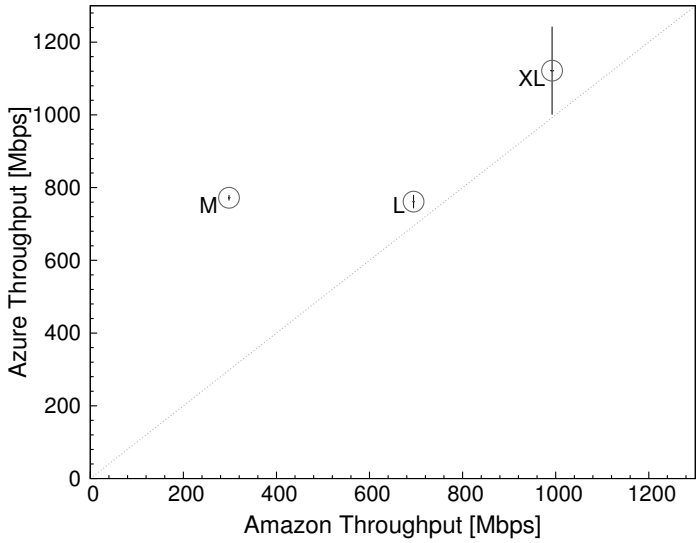


Figure 4.13: Properly quantifying the impact of the factors under his/her control allows the customer to perform advantageous deployment choices, and thus improve the perceived network performance. Circles are centered on average values, while horizontal and vertical error bars report the standard deviation for Amazon and Azure, respectively. Azure values refer to medium, large and xlarge VMs placed in the US region, leveraging CFG_1, CFG_2, and CFG_3, respectively.

When customers are not subjected to other constraints that impose limitations to their choices, they can take advantage of these results to identify the proper trade-off between costs and performance. Customers could further benefit from the information base gathered through non-cooperative monitoring activities also exploiting the impact associated

to deployment factors. For instance, considering the case of a distributed application running on multiple VMs that exchange data among them and whose performance depends on the network throughput but not on the location of the datacenter (e.g. a scientific application), a customer may opt for deploying the VMs onto the best-performing region setting up the most convenient configuration.

Figure 4.13 shows how leveraging Table 4.11a to identify the best deployment choice for Azure (thus choosing the US region and a certain configuration depending on the specific VM size) can lead to a significant improvement of the expected performance with respect to the one obtained through a blind choice. Indeed, the specific set of choices performed allows to obtain results always better in terms of the respective Amazon counterpart. Secondly, the variability of the expected performance appears heavily reduced—according also to the outcome of the analysis presented in § 4.6.2, thus guaranteeing also more predictable performance. With respect to the case in which a customer performs a blind choice (see Figure 4.11), the advantageous deployment choices guarantee to achieve, on average, a performance increment of 64%, 28%, and 28% for M, L, and XL VMs, respectively.

For what concerns Amazon, deploying all the VMs inside the cheapest region seems the best option to obtain the maximum performance at the lowest cost, as network performance is not subjected to changes when changing region.

Finally, for both providers we found that the network performance is very stable over time. This suggests that the providers do not change the network slice assigned to a VM in a dynamic way. In other words, our results reported that cloud applications deployed onto systems of both providers, are not subjected to performance instability due to the variation of the network environment, in spite of the high dynamics of the cloud.

4.7.2 Optimizing network performance by leveraging real-time monitoring

According to the results shown in § 4.6.2, an Azure customer must be aware that even when the same decisions are operated during the deployment process, the network throughput received may significantly vary. On the basis of the results obtained, we define here an approach to obtain the higher network performance level within the Azure cloud.

Differently than the approach proposed in the previous section, the performance optimization strategy proposed here is based also on real-time monitoring activities, instead

of leveraging only historical information. This is required to cope with the variability obtained also when all the factors under the control of the user are fixed. Therefore this approach can be used together with the previous one.

The approach we devise is composed of the steps described in the following.

1. Selecting the desired combination of factor values during the deployment phase according to needs, costs, and expected network performance.

2. Deploying the VMs.

3. Measuring the maximum network throughput, leveraging non-cooperative approaches.

4. Comparing the obtained throughput with the reference values we reported in Table 4.10 which provide the expected performance for the deployment performed.

5. A) If the measured throughput is satisfactory, as the performance will be stable over time, the customer can leverage the VM deployed.
   B) Otherwise when the measured throughput is much lower than the reference value, there is no possibility for the customer to receive a higher throughput over time. In this case, a good option for the customer is to terminate the VMs, create and deploy the VMs from scratch by operating the same decisions on the factor values, and measure the throughput again to check whether it is satisfactory or not (i.e. starting again from step 2).

For instance, it is convenient for a customer deploying two medium-sized VMs in CFG_1 in AP region and to terminate and recreate them when he/she measures only 200 Mbps of achievable throughput: by relocating the VMs, he/she has high chance to get 750 Mbps of throughput (see Figure 4.10). Note that all these operations should be performed during the deployment phase, i.e. before the customer applications or services are made active.

Chapter 5

Inter-datacenter network performance

Cloud service providers and on-line service companies (e.g., Amazon, Facebook, Google, Microsoft, and Yahoo!) have made huge investments in *networks of datacenters* that host their on-line services and cloud platforms to cope with the increasing demand. While the complexity of these network infrastructures is completely transparent to cloud customers, the performance available to them is deeply affected by them. Unfortunately, cloud providers often provide no information about the performance a customer should expect from the cloud network, although customers could significantly benefit from details about the Quality of the Service (QoS) guaranteed. In fact, all major providers grant high-performance network connectivity to their customers, but they provide no more than qualitative information about its performance, mainly due to security and commercial reasons [51, 36, 27].

Top players have made huge investments in specific technologies and cutting-edge solutions to connect distributed cloud resources and guarantee proper performance in presence of dramatically dynamic demand. Datacenter operators may purchase transit bandwidth from Telcos (usually paying based on flat or 95th percentile pricing schemes), or own dedicated lines [39]. For instance, the backbone that carries traffic between datacenters is the largest production network at Google and runs on an SDN- and OpenFlow-enabled infrastructure, in order to improve manageability, performance, utilization, and cost-efficiency of such proprietary WAN [110].

More in general, wide-area transit bandwidth costs more than building and maintaining the internal network of a datacenter [44], a topic that has recently received much more

attention [35]. Networking costs are estimated to amount to around 15% of a datacenter's total worth, and are more or less equal to its power costs (see § 1.3.2).

Expensive investments in this regard are further justified by traffic trends recently estimated by sector reports [15]: (i) cloud IP traffic is going to account for a more and more significant part of the overall IP traffic, being estimated to grow at a compound annual growth rate (CAGR) of 31% from 2014 to 2019; (ii) in more details, public-cloud usage is growing faster than private one, and 56% of the cloud workloads will be in public-cloud datacenters by 2019, up from 30% in 2014; (iii) finally, traffic between datacenters is growing faster than either traffic to end-users or traffic within the datacenter, and will account for almost 9% of total datacenter traffic by 2019. The rapid growth of this traffic is due to the proliferation of cloud services, the need to shuttle data between clouds, and the growing volume of data that needs to be replicated across datacenters. Netflix [111]—the leading provider of on-demand Internet video streaming in the US and Canada—is an interesting case study. It accounts for 29.7% of the peak downstream traffic in US [46], and is able to support seamless global service by partnering with Amazon Web Services for delivery of content and services, as AWS enables Netflix to quickly deploy thousands of servers and terabytes of storage within minutes.

The effects of this interesting trend can also be spotted in the scientific literature: novel solutions leverage the high network performance offered by public-cloud inter-datacenter WANs to develop high performance applications aimed at transferring contents (e.g., multimedia) among datacenters spread world-wide [48, 83, 47]. This recent literature further extends the range of typical usages of public-cloud inter-datacenter networks, that include bulk-data transfer or on-line content transfer (e.g., from and to storage buckets).

In this situation, however, very little information is available about performance figures offered by public-cloud networks connecting datacenters placed in different geographic regions. Few results are publicly available: public-cloud providers advertise qualitative performance indicators at most or do not disclose them at all; information provided by state-of-the-art, public-cloud monitoring services [112] currently does not include inter-datacenter performance; finally, to the best of our knowledge, the scientific community did not focus on the problem yet, and the poor preliminary results cannot be considered exhaustive. The monetary cost of the experimentations necessary for obtaining this kind of information, not directly unveiled by providers, additionally exacerbates this issue. Hence, it is hard to draw significant conclusion upon the available information. As a

consequence, a customer willing to set up a multi-datacenter application in public clouds is not able to cleverly determine the provider or the regions most indicated to host it, based on the inter-datacenter network performance offered for a certain cost.

To fill the existing gap, we have performed a deep experimental evaluation of the inter-datacenter network of the two leading cloud providers: *Amazon Web Services* (hereafter Amazon) and *Microsoft Azure* (hereafter Azure) [17]. All our experimentations did not rely on providers' support, i.e., have been fulfilled with *non-cooperative* methodologies, by adopting the point of view of a generic cloud customer: we have collected performance data about network paths interconnecting public-cloud datacenters leveraging active approaches for approximatively 800 hours, taking into account a set of geographic regions hosting datacenters for both the providers.

To the best of our knowledge, our analysis extends the literature in a number of aspects, as reported in the following.

- Our work is able to depict a clear picture of the inter-datacenter network performance in terms of network throughput and latency, for the two leading public-cloud providers.

- Experimental results investigate the impact on network performance of several configuration factors under customer control, such as the cloud provider, the region, and the size of the virtual machines.

- Our analysis provides insights into the communication infrastructure leveraged by cloud providers, also showing the existence of phenomena generated by the management strategies which may impact both the performance experienced by the customers and the results of research investigating these networks.

- Performance results have also been compared to provider-imposed fees, in order to give useful guidelines to customers willing to deploy distributed applications onto the cloud.

- Empirical outcomes confirmed that providers often rely on their own dedicated infrastructure to connect geographically distributed sites, but also show that, in some circumstances, they depend on third-party networks, being forced to provide cloud customers with lower performance at higher costs.

- Our results are compared to those found in previous work, highlighting the changes in terms of performance figures, and analyzing the trend that these infrastructures are subjected to over time.

The chapter is organized as follows: § 5.1 details the reference architecture; § 5.2 describes the scenarios of interests; § 5.3 surveys the results from the most-related literature; § 5.4 shows the results of our work for the two providers taken into account.

5.1 Reference architecture

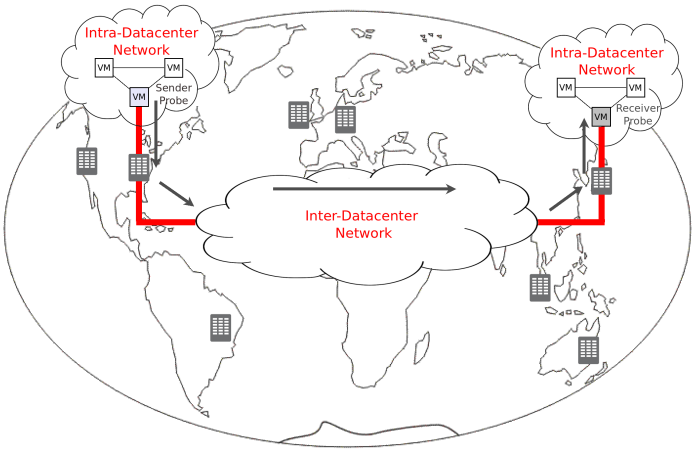


Figure 5.1: Reference architecture. Before being delivered to the receiver VM, synthetic traffic traverses different layers, i.e. the intra-datacenter network at sender side, the inter-datacenter network, and the intra-datacenter network at receiver side.

We aim at measuring the performance of the network paths interconnecting VMs deployed onto geographically distributed public-cloud sites. According to the reference architecture reported in Figure 5.1, the traffic directed from one side of the communication to the other traverses different and distinct layers. The traffic generated by a sender VM normally traverses (i) the devices composing the intra-datacenter, high-performance network at sender side first. Then, it enters and traverses (ii) the inter-datacenter WAN, and, before being delivered to the receiver VM, it passes through (iii) the intra-datacenter network at receiver side. Note that the internals of both intra- and inter-datacenter networks are out of our knowledge, as we adopt the point of view of the general customer. In fact, our approach is aimed at measuring the performance experimented by customers' traffic. Although different layers exist, the inter-datacenter network performance is assumed

Table 5.1: Summary of factors and considered values.

Factors	**Values**
Provider	Amazon, Azure
Region	Europe (EU), North Virginia (US), South America (SA), Asia-Pacific (AP)
VM Size	medium (M), extra-large (XL)
Transport Protocol	TCP, UDP

Table 5.2: Cost for transferring data to another region, as of Sep.'15. Only for Azure, data-transfer costs scale with volume.

Region	**Amazon (€/GB)**	**Azure (€/GB)**
EU	0.02	0.0734–0.0422
US	0.02	0.0734–0.0422
SA	0.16	0.1527–0.1350
AP	0.09	0.1164–0.1012

to be the bottleneck of the communication due to practical, technological, and physical limitations. Our results have therefore to be intended as related to these networks, if not stated otherwise.

For our experimentation, geographically distributed VMs have been instrumented with Ubuntu 14.04 operating system and all the necessary measurement and diagnostic tools needed for estimating the network performance. As already described for in intra-datacenter performance analysis, we have leveraged the CloudSurf platform (see Chapter 3) and used the network measurement tool named *nuttcp* [113] to inject synthetic traffic into the network from a sender to a receiver VM.

5.2 Scenario selection strategy

The reference architecture described above has been modified in different ways, according to factors described in the following section, giving birth to a set of scenarios of interest.

The inter-datacenter network performance can be measured and monitored in different scenarios when adopting the point of view of the general customer. In fact, a set of factors exists that may heavily influence the perceived performance, as confirmed by the

outcome of our experimentations. In this perspective, our work significantly extends the surveyed literature carefully analyzing the impact of these factors. Without claiming to be exhaustive, we believe an important contribution of our work is the identification of the factors to be carefully taken into account when performing similar analyses. These factors—summarized in Table 5.1—will be shortly discussed in the following.

According to latest reports about public IaaS cloud computing [17], the market is dominated by only a few global providers among the huge number of offers. In this work we take into account the IaaS of two **providers**: *EC2* for Amazon [19] and *virtual machines* for Azure [69]. The former is the clear market leader (over a million active customers in more than 190 countries), while the latter is the only clear challenger, also due to the continual investments in the latest infrastructure technologies.

Both providers are steadily expanding their global infrastructure whose growth is backed by billion investments: infrastructural expansion is claimed to be a priority because of the direct benefits generated for the customers (primarily latency reduction and throughput increase). For our experimental campaigns we have identified four **geographic regions**, where both providers have deployed datacenters: *Ireland (hereafter EU)*, *North Virginia (US)*, *Sao Paulo (SA)*, and *Singapore (AP)*. Being forced to select a subset of all possible regions—mainly due to cost constraints—we have picked a region per continent, in order to ensure geographical diversity to our dataset. We investigated the network performance of all the paths connecting all the regions in this selection. Hereafter, we will adopt the notation $A \rightarrow B$ to refer to the path from region A to region B. $A \leftrightarrow B$ will be used to refer to both the paths A→B and B→A at once—i.e., when both directions are taken into account. Note that traffic moving outside from a region is subjected to costs that vary with such source region (see Table 5.2). As, Amazon customers can further choose an **availability zone** once a region has been selected, in our study we have taken into account also the impact of different availability zones inside a region.

In terms of type, we used last-generation, general-purpose VMs for both providers. In terms of size, we considered two different ones named *m3.medium* and *m3.xlarge* for Amazon, and *A2* and *A4* for Azure. Hereafter we simply refer to them as *medium* (M), and *extra-large* (XL), respectively, for both providers. Table 5.3 reports further characteristics and the costs of the VMs adopted in our analyses. Note that both providers provide details only regarding RAM and CPU. Regarding the network characteristics, Amazon only provides a qualitative description of the expected performance, Azure completely

Table 5.3: Selected sizes and details (price may vary with regions).

	VM size	Type and Size	CPU cores	RAM (GB)	Network Performance	Min-Max Hourly Cost (€/h)
Amazon	**M**	m3.medium	1	3.75	*Moderate*	0.070–0.098
	XL	m3.xlarge	4	15	*High*	0.280–0.392
Azure	**M**	A2	2	3.5	n/a	0.1192–0.1460
	XL	A4	8	14	n/a	0.4767–0.5839

hides this information.

Finally, we have considered two **L4 protocols** in our experiments: UDP and TCP.

Dataset details. According to the non-cooperative approach we have proposed, we did not rely on provider-restricted information. All our analysis is based on the data collected between March and November 2015 adopting the methodology introduced above. The collected process required more than 790 hours of traffic generation (i.e. we have injected synthetic traffic into the inter-datacenter network for more than 790 hours). We considered the 12 combinations of the four regions selected for each provider. Experiments have been run between VMs of the same size (M or XL). Repeated, 5-minute-long experiments have been performed in the same conditions, equally spaced in 24-hour intervals.

Our experimentations have been subjected to providers' fees, and according to their terms of service, inter-datacenter traffic is subjected to volume-based charging (see Table 5.2). We limited the number of UDP runs—especially for larger VM sizes—because of budget constraints and the high volumes transferred during experiments. Table 5.4 reports more details about the dataset collected. Beside performance measures, path information has also been collected to complement the view on the performance.

5.3 Summary of the results in the literature

Traffic in public-cloud inter-datacenter networks is rapidly growing, as estimated by recent reports [15]. Moreover, novel applications are critically relying on it [48, 83, 47]. Unfortunately limited and poor information is available today about the performance attained and attainable by this traffic.

Most of the works in the literature aimed at providing a broad characterization of the performance of public clouds, i.e., did not directly focus on network performance. Most

Table 5.4: Experimental dataset details. Number of 5-minute-long experimental samples and overall experiment duration for each combination of VM size and L4 Protocol, for both providers.

	VM size	**TCP**	**UDP**
Amazon	**M**	3456 (288 h)	1740 (145 h)
	XL	3456 (288 h)	24 (2 h)
Azure	**M**	432 (36 h)	108 (9 h)
	XL	288 (24 h)	36 (3 h)

of these works only benchmarked the intra-datacenter network performancewith different purposes and often providing conflicting results [27, 28, 78, 37, 107]. A limited set of works took into account the performance of inter-datacenter networks.

Chen *et. al* [73] focused on the interplay of multiple datacenters performing a passive analysis on Yahoo! network flow dataset.

Li *et al.* [34] in their broader analysis, benchmarked also cloud inter-datacenter networks, but considering only TCP throughput performance between two US datacenters. They adopted general purpose instances of just one size for their experimentations. They found that the throughput across datacenters is much smaller than the one within the datacenter for all the providers considered, and that both Amazon and Azure show median values of inter-datacenter TCP throughput larger than 200 Mbps. Morever, they reported a variation of throughput across datacenters larger than the one measured within the same datacenter.

Feng *et al.* [83] performed an experimental evaluation of Amazon network paths interconnecting seven different datacenters to support their study on a set of algorithms to minimize operational costs of inter-datacenter video traffic. They considered datacenters in North California, Oregon, Virginia, Sao Paulo, Ireland, Singapore, and Tokyo. They used medium instances and monitored network performance for only 3 minutes. Their results revealed very different throughput values, ranging from 9.6 Mbps (for the path between Sao Paulo and Singapore) up to 545.1 Mbps (for the path between North California and Oregon). Values of end-to-end latency measured were lower than 587.3 ms for the 90% of the paths, while the average was about 349.1 ms. The same authors [48] proposed a protocol to deliver packets in video conferencing, designed for the inter-datacenter net-

work, and tailored to the needs of a cloud-based service. Measurement results supporting this work and obtained adopting small instances placed in the regions reported above ranged from 20.9 Mbps to 130.8 Mbps for throughput and from 11.3 ms to 441.7 ms for latency. Finally, Garcia-Dorado and Rao [38] presented a framework that exploits cloud-pricing schemes to construct overlay distribution networks for bulk-data transfer that proved to be effective from the customer-side perspective. To evaluate their approach they conducted experimentations on both Amazon and Azure, leveraging medium VMs and measuring TCP bandwidth and latency performing two-minute-long measurements during one day. They found low variation of throughput values, especially in paths exposing better performance. results, providing interesting insight to motivate these results.

Our work significantly differs from the others in recent literature dealing with the performance of the inter-datacenter networks. In spite of the analysis presented in [73], our work does not rely on provider-restricted information. The methodology we propose is completely based on active measurements performed from the point of view of the cloud customer: therefore it is independent of provider's will to disclose information, guarantees the results to be independent from it, and allows to replicate the study at any time. Differently from the analysis in [34], our work explicitly focuses on the performance of inter-datacenter networks. This gives the opportunity of deepening the aspects strictly related to the measurement process. Thanks to this, we investigate also the interesting traffic engineering practices and their impact on both the measurement process and the QoS perceived by cloud customers. Finally, with respect to the measurement data presented in [48], [83], and [38], our study is more systematic, details a repeatable methodology, compares the performance of multiple providers, and takes into account the specific management strategy they enforce.

5.4 Experimental results

In the following we discuss the most interesting results stemming out from the analysis of data we have collected. Firstly, we provide an assessment of the performance of the network interconnecting geographically distributed cloud sites for the two providers, comparing their performance, and also showing how it is influenced by a set of factors. This broad analysis provides useful information for customers willing to deploy distributed ap-

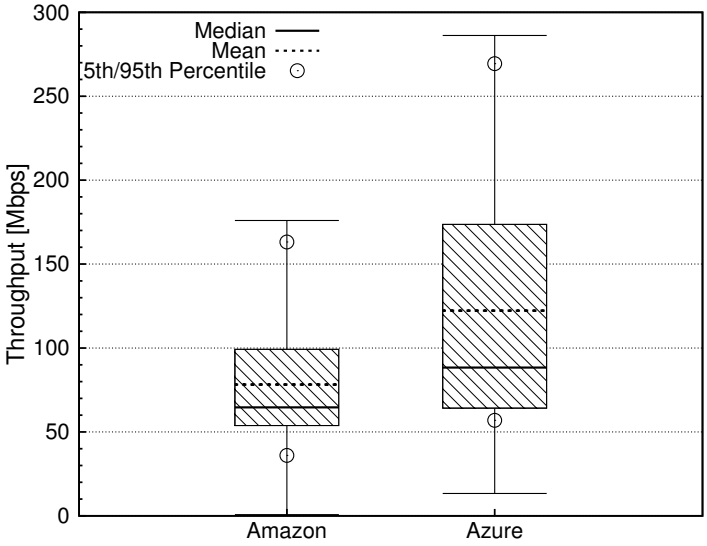


Figure 5.2: TCP throughput distribution across different regions. Each sample represents the mean of a 5-minute-long experiment. Azure performs better on average (+56%).

plications onto the cloud, and help them with their setup choices, according to application needs. Moreover, this analysis provides insights useful for researchers and practitioners willing to perform similar analyses. Secondly, we further deepen the analysis for interesting cases, providing insights into the communication infrastructures providers leverage for inter-datacenter communications.

5.4.1 TCP throughput

In this section we provide an assessment of the performance of the network interconnecting geographically distributed cloud sites for the two providers, comparing their performance, and also showing how it is influenced by a set of factors.

Figure 5.2 reports an overall picture of the throughput for both providers in all the experiments (see Table 5.4). Each sample in the plot represents the mean of a 5-minute-long experiment. It is worth noting that even values far from the global average of Figure 5.2 well represent the samples collected during that particular 5-minute measurement. In fact, the *coefficient of variation* (CoV) within each experiment is very low: the 95th percentile of its distribution along all the experiments is about 0.2. Figure 5.2 shows that Azure inter-datacenter network performs better than Amazon's one in terms of network throughput, achieving TCP throughput values 56% higher (78.2 Mbps vs. 122.2 Mbps, on average). Almost the same proportion is kept when considering maximum throughput

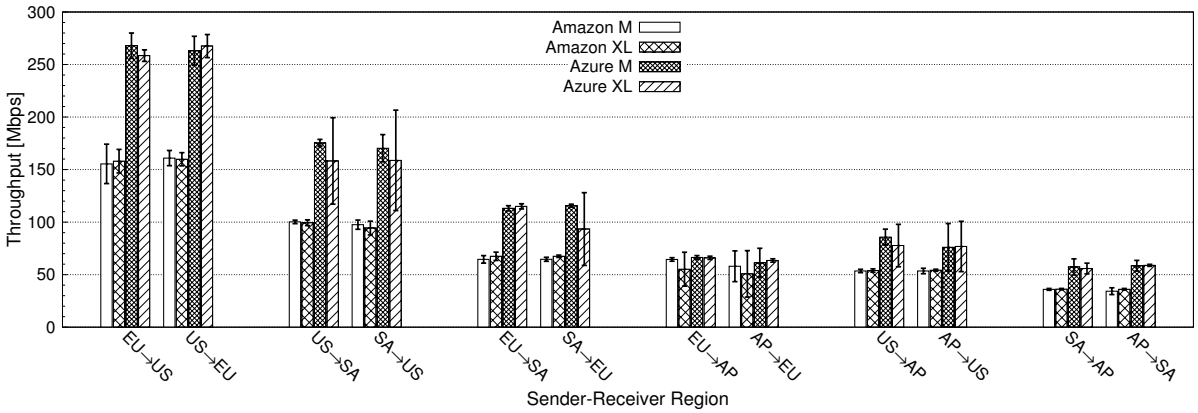
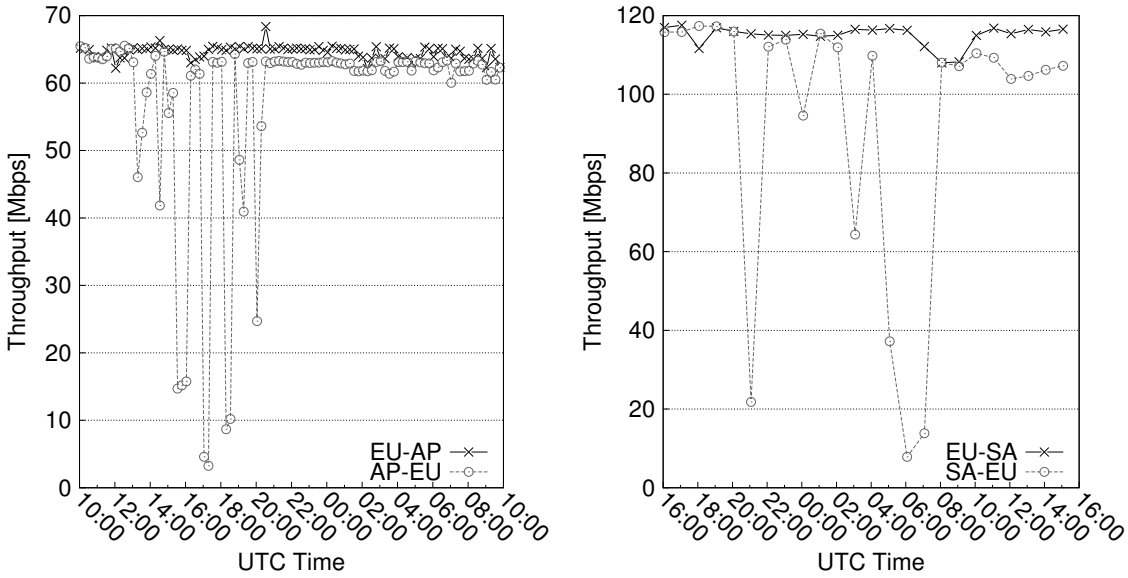


Figure 5.3: TCP throughput breakdown on different region pairs (mean and standard deviation), for different providers and VM sizes. For both providers, region pair is the factor with the highest impact, while VM size appears non influential.

achieved (286.2 Mbps vs 176.0 Mbps). Interestingly, only 25% of samples collected on Amazon's infrastructure have values higher than 99 Mbps, while 95% of samples collected on Azure's one have values higher than 57 Mbps. Finally, TCP throughput values for Amazon can be as small as 1 Mbps, while for Azure they are never smaller than 13 Mbps.

Figure 5.3 provides a breakdown of TCP performance. The bar chart reports mean and standard deviation across different regions and different VM sizes, for the two providers. A few important observations can be done looking at this figure. We can immediately observe the large performance differences across different regions, up to about 80% in the worst case. Interestingly, ordering the regions according to the achievable throughput, we obtain the same ranking for the two providers, with the only exception of US↔AP pair, which performs better than EU↔AP pair for Azure, on average. The achievable TCP throughput is not clearly affected by the size of the VM, for both providers, despite the different fees imposed. Note that small performance differences between differing sizes are observed. But, they are not always biased towards the larger, and they are always associated to higher variability. The standard deviation inside a region pair is normally very low, although some pairs show a higher throughput variability only for Azure XL VMs (e.g., SA→US, US→SA, and SA→EU).

Recall that M and XL VMs are advertised to have *Moderate* and *High* network performance respectively, according to Amazon's documentation. Our results show that differing performance figures across different VM sizes, highlighted in the previous chapter, are achievable only by communications that involve VMs both deployed inside the



(a) AP↔EU Amazon. (b) SA↔EU Azure.

Figure 5.4: Relevant examples of performance asymmetry for different directions.

same region, while they do not hold for inter-datacenter TCP performance. This is likely due to TCP dynamics.

Finally, performance figures appear to be roughly symmetric in the majority of the cases examined, although during our experimentations we also encountered severe degradations involving only one direction of the communication. Figure 5.4 reports some of these interesting cases. As shown, intermittent but heavy performance degradation—exposing throughput settling down to less than 10 Mbps—has been observed between AP and EU for Amazon and between SA and EU for Azure. Interestingly, although referring to different days and providers, these results are both related to only one direction of the communication, i.e. the downlink of VM inside EU.

We believe this broad assessment can be very useful to cloud customers willing to draw upon public clouds to deploy their distributed architectures. Thanks to these results, customers can wisely select among regions, when possible. Otherwise, this analysis provides them with a quantification of the significant network performance differences among regions. Moreover, relying on larger sizes to increase TCP inter-datacenter performances has not effect. Also, comparing the two providers we found that Azure performs better on average, while asking higher costs for the VMs and for the data transfer. Due to this trade-off, we believe that the choice of the provider should be tailored according

to regions of interest and should be driven by the specific characteristics of the application. Finally, considerations about performance symmetry may also be considered to properly place nodes in the different regions, according to the specific application the inter-datacenter network is leveraged for, and to the different roles of the counterparts involved in a communication.

In general, our analysis revealed TCP throughput values smaller than the ones reported in previous works. Indeed, authors of [34] observed TCP (median) throughput higher than 200 Mbps for both Amazon and Azure. Results are hard to compare because the authors disclosed no information about the VM size adopted for the experimentations and restricted inter-datacenter throughput analyses to only two regions placed in the same continent (United States). Several interpretations are available for this discrepancy, therefore. Different performance figures may be explained by the fact that experimentations in [34] involved datacenters separated by a lower distance and hence backed by infrastructures implementing technologies guaranteeing different performance levels. Another potential cause is related to the presence of less competing traffic across the inter-datacenter networks at the time when experimentations conducted in [34] were performed, i.e. around 5 years before ours [15]. Interestingly, the performance reduction observed after these 5 years is larger for Amazon than for Azure. They could be further justified by different catchment areas for the two providers, considering also the impact of the higher number of customers Amazon has on TCP congestion-control dynamics. Finally, authors of [34] also found throughput variability markedly higher than ours—regardless the fact we consider the variability within each experiment or between different experiments. This conclusion holds although analysis in [34] refers to data collected over a more limited observation period (one single day) and is related to measurements between two datacenters both placed in the US. This reduced variability is in line with the increase of the competing traffic already hypothesized above.

5.4.2 UDP throughput and end-to-end path capacity

Before digging into the details of this analysis and of the related results, we show how UDP measurement accuracy may be biased by specific phenomena related to the resource management enforced by providers. Properly understanding the impact of these phenomena allows also to improve the quality of the analysis.

In more details, we found how UDP throughput values measured for Azure inter-

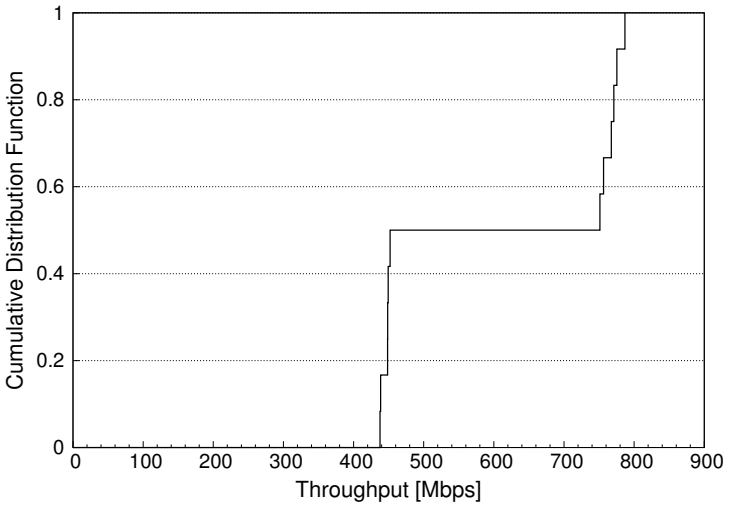
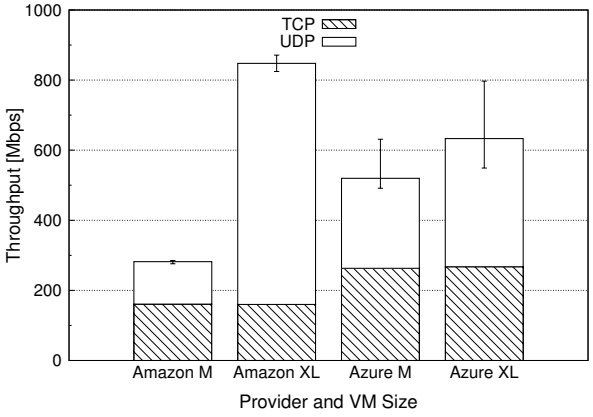


Figure 5.5: Empirical cumulative distribution for UDP throughput (Azure, XL VMs, EU→US). Each sample is the average of a 5-minute-long experiment. VMs have been terminated and recreated at each experiment.

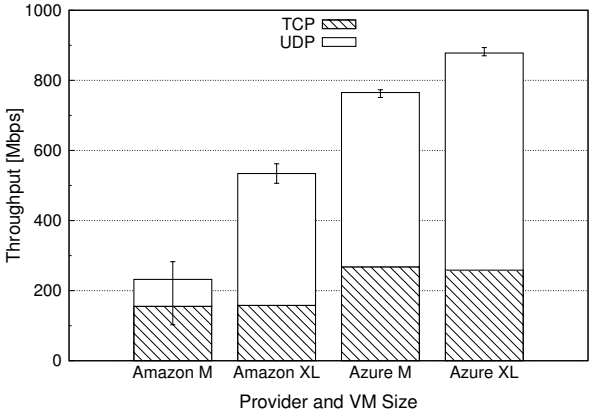
datacenter paths can be impacted by intra-datacenter limitations already documented in the previous chapter. TCP performance analysis is immune from these limitations instead, since the end-to-end bottleneck located along the paths proved to be tighter than them. Figure 5.5 shows the distribution of the UDP throughput measured in different 5-minute-long experiments between two XL VMs (EU→US). The value of the end-to-end throughput measured along the path between the same pair of regions settles to two well-defined values when adopting different VMs pairs: 450 Mbps and 800 Mbps. The former appears to be not related to inter-datacenter performance as it reflects intra-datacenter limitations. Note that the measured value does not change if the VMs involved in the measurement process are not terminated and then launched again from scratch. In the specific case shown, we encountered the lower value 50% of the times: this may induce to heavily under-estimate the maximum end-to-end UDP throughput if not properly taken into account. Although the influence of intra-datacenter limitation cannot be avoided *a priori*, restarting the VMs involved in the measurement process helps refine the estimation done. Note that the phenomenon described, although not representing a peculiarity of the inter-datacenter network, can impact the performance transparently perceived by the cloud customer. With this aim, we purposely iterated the VM termination and recreation process to bypass intra-datacenter limitations and unveil the peculiarities of the inter-datacenter network. We believe that the one adopted represents a good practice to adopt

when performing this kind of analyses for Azure infrastructure.

We can now analyze the obtained results. In summary, they show that UDP throughput proved to be significantly higher than TCP's for all the source-destination pairs considered. Although this result was expected (UDP protocol is not subjected to congestion control dynamics typical of TCP), interestingly we have identified cases in which UDP inter-datacenter throughput durably reaches the intra-datacenter performance figures reported in the previous chapter, and thus appearing limited by bottlenecks imposed by providers at source.



(a) US→EU.

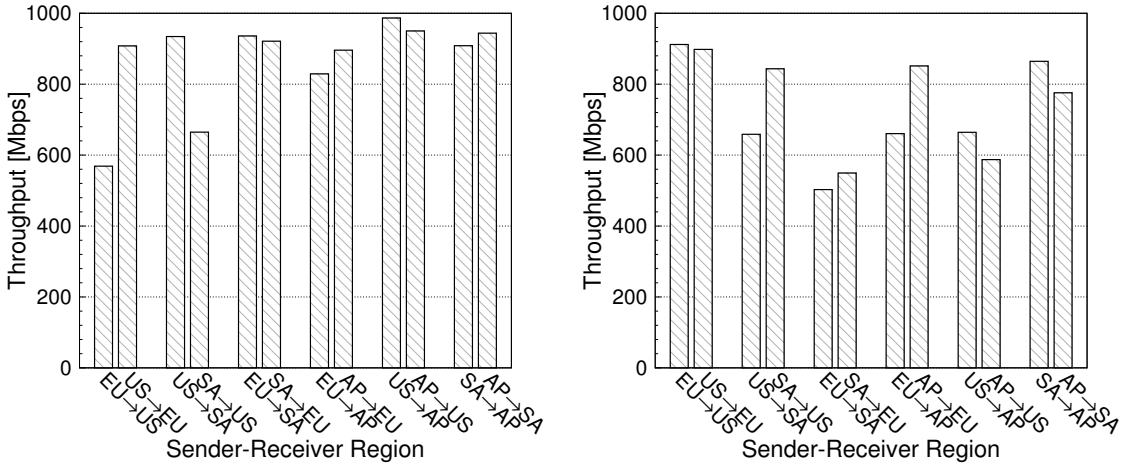


(b) EU→US.

Figure 5.6: TCP and UDP inter-datacenter average throughput for US↔EU with M and XL VMs (whiskers report maximum and minimum). While VM size does not affect the achievable TCP throughput, UDP is able to reach higher end-to-end performance, giving evidence of path capacities as large as more than 800 Mbps.

In detail, Figure 5.6 compares UDP and TCP average throughput obtained between

the pair of regions with the best performance for both providers: US and EU. While TCP performance does not vary with the size of the VMs, UDP throughput reaches much larger values. Note also how UDP maximum values are compatible with limitations imposed by providers at source and based on VM size. On the one hand, these results suggest that worse TCP performance is determined by network congestion across datacenters. On the other hand, the better performance of UDP gives evidence of the network capacity of the inter-datacenter paths. We can find further justifications for this empirical result considering the impact of the higher number of customers Amazon has on TCP congestion control dynamics. Although VMs are allowed to inject traffic into the inter-datacenter network at a rate as high as the throughput measured in Chapter 4, and the inter-datacenter network is able to deliver traffic at this speed, network congestion represents the main bottleneck when relying on TCP. This result is generalizable across different regions, even if actual UDP throughput values change from case to case,



(a) Amazon. (b) Azure.

Figure 5.7: Lower bounds of path capacity for inter-datacenter paths, measured as the maximum UDP throughput achievable between two regions.

We estimated the capacity of each path as described in the following to better understand how network resources are deployed across regions. We calculated the average throughput over 5-second-long non-overlapping windows considering the timeseries of each UDP experiment involving XL VMs in the dataset. This approach allowed us to mitigate the effect of throughput spikes—potentially caused by queuing and buffering dynamics generated by the coexistence of multiple layers of virtualization—that could ex-

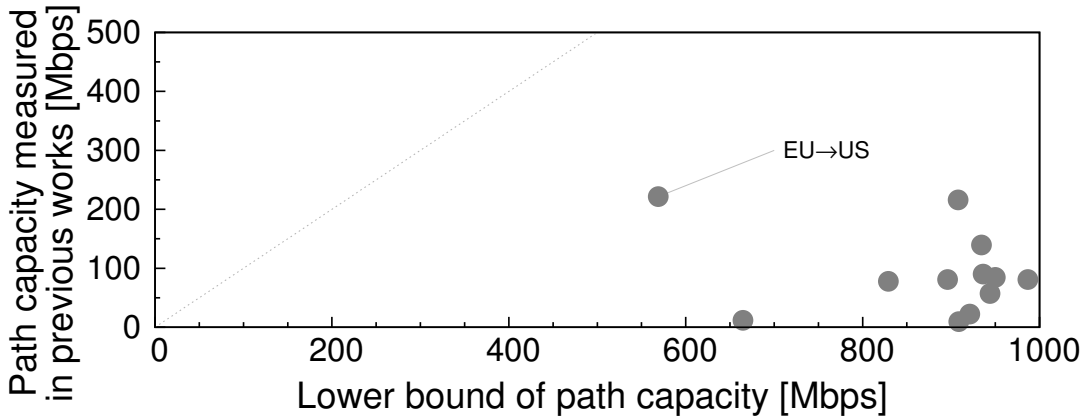


Figure 5.8: Path-capacity comparison. Lower bounds estimated in this work are higher than values reported in one of the previous [83].

ceed end-to-end capacity [26]. We then estimated the end-to-end capacity of each path extracting the maximum of this time series. Note that the values obtained with the approach proposed, represent a lower bound of the capacity, i.e. the end-to-end path that a user can leverage is able to deliver *at least* traffic at this throughput. The results are shown in Figure 5.7: Figure 5.7a and Figure 5.7b show the lower bounds for Amazon and Azure inter-datacenter paths, respectively. Estimated capacities are always larger than 800 Mbps for most cases. More in details, lower bounds for Amazon capacities ranged from around 560 Mbps to 986 Mbps, with only two cases exposing values smaller than 800 Mbps. For what concerns Azure, estimated capacities were lower on average: they ranged from 502 Mbps to 912 Mbps.

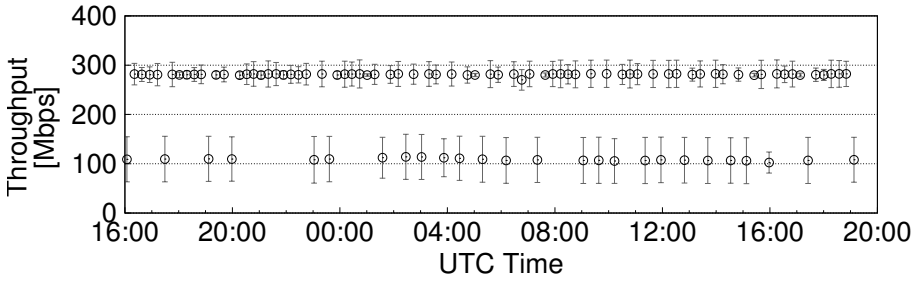
Interestingly, Amazon capacity values reported in Figure 5.7a result significantly larger than the homologous reported in [83], always lower than 300 Mbps. Figure 5.8 compares these values. As shown in the figure, EU→US surprisingly exposed the largest value according to the analysis performed in [83] but the lowest in ours. The results of our analysis show larger values than previous ones also limiting our dataset to data related to M-sized VMs (whose UDP throughput values reach 280 Mbps in all the circumstances considered).

Differing values could be justified by substantial infrastructure enhancements. However, we believe that they could also be impacted by the measurement methodology adopted in [83] (single 3-minute-long experiments leveraging medium VMs). This further highlights the need to detail the adopted methodology, to compare and validate results through further analyses.

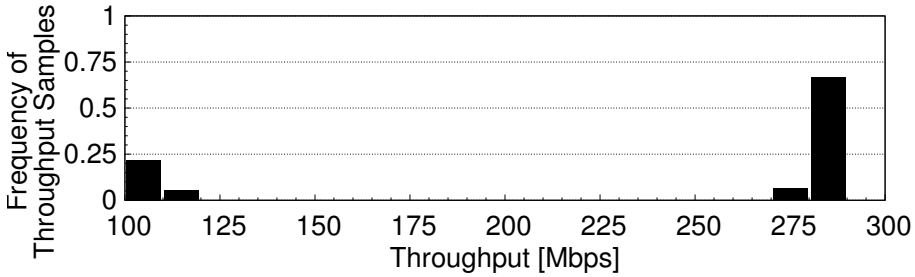
5.4.3 Throughput variability

Our further analyses have shown that inter-datacenter performance for both providers may not be simply regulated by limitations imposted at source: restrictions can be imposed by providers also along the path, also because of the several layers traversed, none of which is under the direct control of the cloud customer. Our dataset revealed some interesting cases in which UDP throughput measured over time is not stable. They are mainly related to Amazon M for EU→US and Azure M and XL for US→EU. This inflates the variance of the throughput, as reported by the larger error bars in Figure 5.6. We discuss these representative examples in the following.

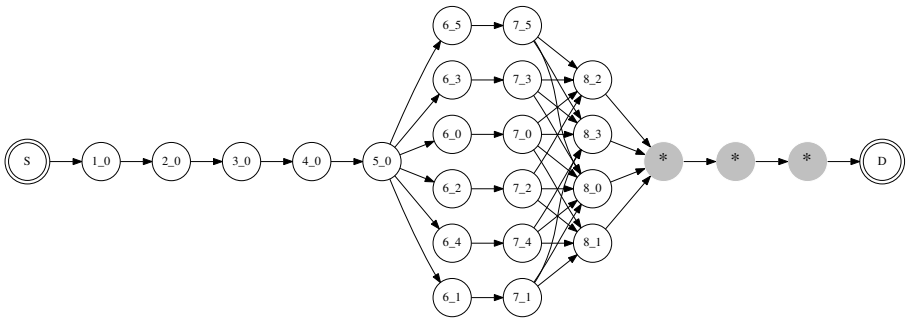
For what concerns Amazon VMs, we found a number of interesting cases in which UDP throughput is not stable, showing changes in the perceived path capacity over the time. An example is related to the path connecting M-sized VMs between EU and US regions, whose minimum, average, and maximum values are reported in Figure 5.6b. Deeper details for the spotted phenomenon are shown in Figure 5.9. UDP throughput dramatically changes its value during the time—but not during the same experiment (Figure 5.9a). It switches between two well-defined values around 280 Mbps and 110 Mbps (Figure 5.9b). While the former is in line with maximum performance achievable by M-sized Amazon VMs, the latter reflects a clear and systematic path capacity decrease. This result is in line with management practices commonly adopted [110] (current network technologies, such as SDN, allow to change network configuration on the fly, based on system state and needs). On the other hand, it can also be explained considering the multiple paths that we have identified between these two regions. We uncovered them using state-of-the-art technologies such as *Paris Traceroute Multipath Detection Algorithm (MDA)* [114]. The characteristics of the path inferred by adopting MDA for the case examined are reported in Figure 5.9c. Each node in the graph represents a unique IP address discovered by the tool along the path from the source VM (S) to the destination VM (D). Stars represent anonymous hops, i.e. associated to devices whose ICMP error messages did not reach S. According also to common datacenter topologies [35], they can be mapped to IP nodes placed inside the intra-datacenter network, and the root cause of their appearance is likely related to the filtering of incoming ICMP messages enforced at the border of the US datacenter. Edges in the graph connect nodes discovered by leveraging the same traffic flow. As shown in the figure the sole node appearing at the 5th hop acts as a loadbalancer at IP level, systematically distributing incoming traffic among different interfaces (hops



(a) UDP throughput time series (error bars report the standard deviation).



(b) UDP throughput empirical distribution.

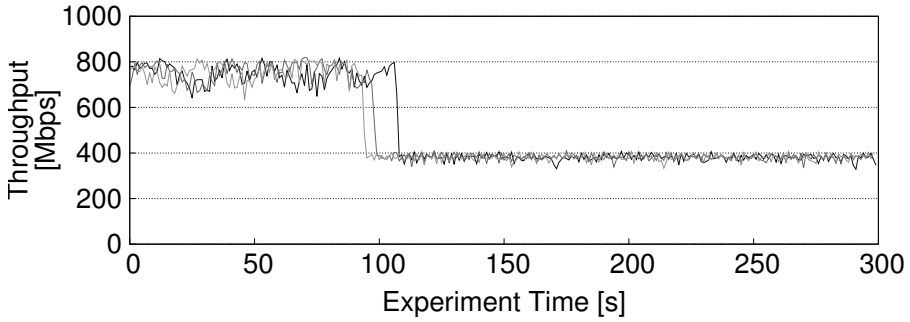


(c) Load-balanced paths inferred by adopting MDA.

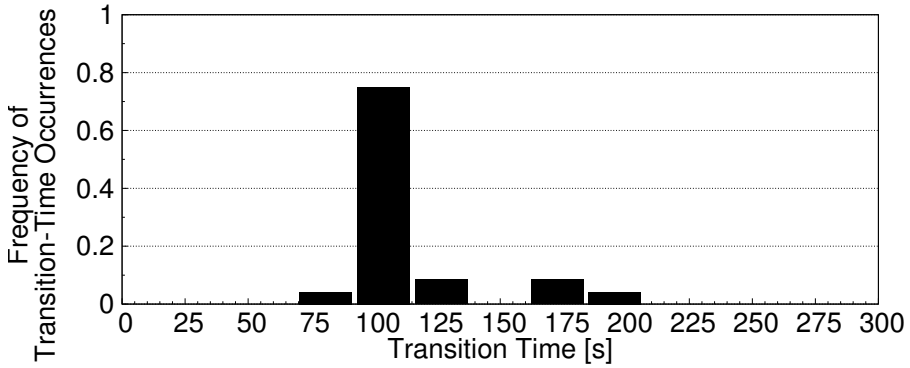
Figure 5.9: Amazon, EU→US, M-sized VMs. UDP throughput switches between two different values over time, suggesting the dynamic variation of the capacity of the path. Note how the distribution of the throughput samples follows a binomial shape. A likely cause is the existence of multiple paths between the source and the destination.

6, 7, and 8). The observed characteristics are confirmed also by the router-level graph obtained through alias resolution—using state-of-the-art techniques [115, 116]—applied to the output of MDA. Note how in this case performance variation occurs only across different experiments—even though no termination and recreation have been enforced—while within each of them, UDP throughput settles to a well-defined value for all the duration of the experiment.

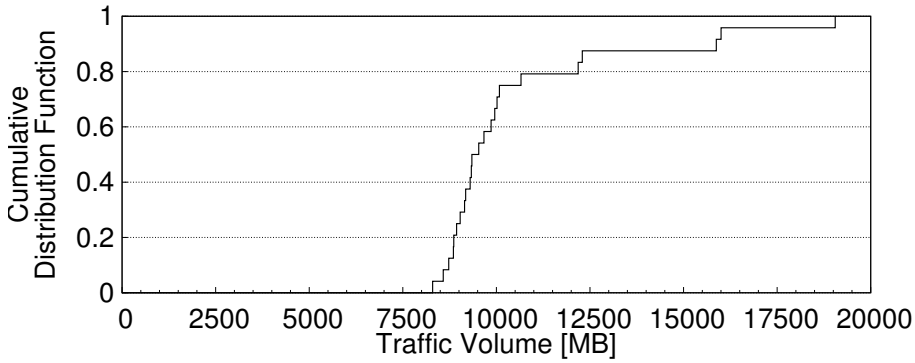
On the other hand, the performance variability exposed by the paths interconnect-



(a) Instantaneous throughput evolution of three 5-minute-long experiments.



(b) Distribution of the high-to-low transition time.



(c) Cumulative Distribution of the volume of traffic transferred before the transition happens.

Figure 5.10: Azure, US→EU. UDP throughput within 5-minute-long experiments typically switches from a high to a low stable value of 400 Mbps (a). The transition typically happens around 100 s (b). The transferred traffic volume ranges from 8,000 to 10,000 MB for 75% of the cases (c).

ing Azure M or XL VMs between US and EU regions has a significantly different nature. Indeed, the variability spotted is due to performance variation *within* each of the 5-minute-long experiments. Figure 5.10 shows some experimental evidences for this phenomenon.

Considering three experiments related to these two regions, Figure 5.10a shows how the throughput typically varies within each experiment: interestingly, two well-defined throughput values can be easily identified. All the experiments between this pair of regions shows the same pattern: the throughput dramatically switches from a high value (between around 700 and 850 Mbps, depending also on the VM size) to a low one (around 400 Mbps). Also the stability of the throughput samples significantly changes after the transition: before the transition the throughput appears very unstable, while after the transition, it stably settles down to about 400 Mbps. The CoV is almost halved after the transition.

The dramatic throughput variation described above happens *on the fly*, i.e., when a communication is active. Moreover, the high-to-low transition may happen at differing points in time for different experiments. As a consequence, different mean values have been observed, which also generate the larger variability range in Figure 5.6a. The distribution of the transition time is shown in Figure 5.10b. Interestingly, in more than 75% of the cases, the transition happens at around 100 s, thus exposing a certain deterministic behavior. Differently from the case of Amazon reported above, this empirical result could be mapped to mechanisms that restrict the maximum capacity available to a customer based on the traffic volume previously generated. Figure 5.10c shows the distribution of the volume of the traffic transferred before the high-to-low throughput transition happens. For about 75% of the cases, volume transferred before the transition ranges from $8,000$ to $10,000$ MB. Note that, small differences in the transition time reflect in larger discrepancy in the transferred volume because of the high throughput. Besides Azure, similar cases have been identified also for the paths connecting Amazon XL VMs.

We verified that no significant variation of the actual traffic injected in the network by the sender VMs has been identified for the cases discussed above. Lower throughput values are reflected by a proportionally higher packet loss. Therefore, the differing performance levels identified are not caused by the traffic generation capabilities of the VMs. This further suggests that the observed phenomenon is the consequence of traffic management policies enforced by providers along the paths that interconnect one region to the other. It is worth noting that this phenomenon may impact both the results of the measurement process and the user experience. Firstly, measurements shorter than 100 s are not able to spot the throughput transition. Secondly, longer measurements could lead to misinterpret performance variability if not associated to a deeper analysis. Finally, dra-

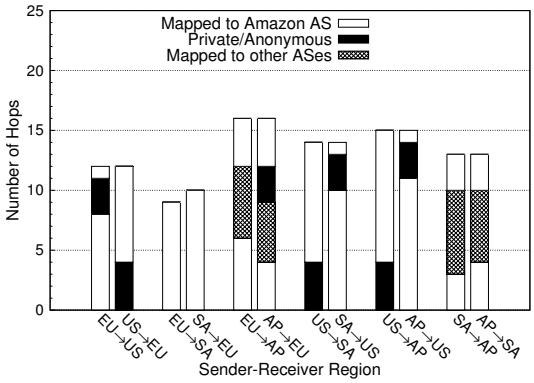
matic throughput drop (around −50% in the example proposed) can cause non-negligible troubles to customers, heavily impacting the perceived Quality of Service.

5.4.4 Performance vs. fees

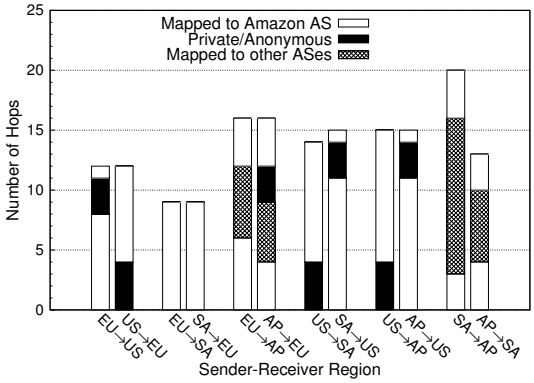
Our empirical results show that worse performance typically involves two specific regions: AP and SA. Interestingly, these two regions are also the ones with the highest data-transfer costs for the customers (see Table 5.2), thus representing unfavourable choices for them. Indeed, data transfer from AP and SA is subjected to higher costs with respect to EU and US regions, which amounts to 8× and 4.5× for Amazon, and up to 3.2× and 2.3× for Azure, respectively. To better understand these aspects, additional experimentations have been performed. We have traced IP paths between regions by adopting *traceroute*. Note that this analysis has been performed only for Amazon, due to ICMP filtering policies likely enforced at the borders of the Azure datacenters, which prevent the adoption of *traceroute*. The experimental campaigns we set up were designed to trace the region-to-region path while running the performance test. Then we have mapped each IP address identified along the paths to the autonomous system it belongs to. With this aim, we have applied IP to autonomous system number mapping (*IP-to-ASN mapping*) to each IP address collected, by relying on free external services [117] and on the IP-address ranges publicly advertised by the provider itself [118].

Figure 5.11 reports the results of this analysis for both M and XL VMs. The bar charts show for each path the number of hops it is composed of, also classifying the hops along the path from the source (hop 0) to the destination (whose position depends on the length of the path). More precisely, they also show the results of the IP-to-ASN mapping, by splitting the hops composing each path in three sets: (i) the ones mapped to AS owned by Amazon itself (#16509/#38895), reported in white; (ii) the ones impossible to map being anonymous or associated to private IP addresses, reported in black; (iii) the ones mapped to ASes other than Amazon, reported in grey. Results appeared stable over multiple experimental repetitions.

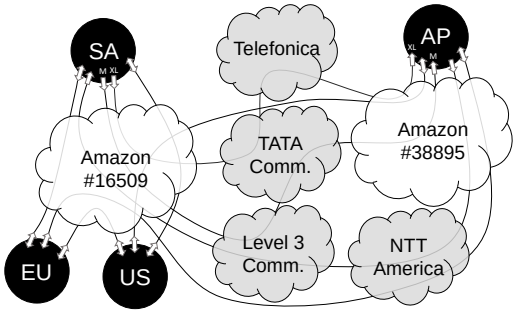
As shown in the figure, we can infer that the length of the path is symmetric in most of the cases—i.e., it does not change when switching source and destination regions. A few cases showing asymmetry have been spotted. For instance: we counted 9 or 10 hops for EU↔SA (M VMs); we counted 15 or 14 and 13 or 20 hops for US↔SA and SA↔AP, respectively (XL VMs).



(a) Length and hop classification for paths between M VMs.



(b) Length and hop classification for paths between XL VMs.



(c) AS level graph. VM size impacts the traversed path only where explicitly stated. Incoming and outgoing paths can be discerned on the basis of the direction the arrows reported.

Figure 5.11: IP-to-ASN mapping and length for Amazon inter-datacenter paths. Paths lay on provider-owned infrastructure for four out of six cases.

The black part of the bar charts shows that anonymous hops normally appear close to the edges of the path, and may therefore be intuitively mapped to Amazon (i.e. intra-datacenter hops). Hence, we can consider as part of Amazon infrastructure the hops reported in white and black in Figure 5.11a and Figure 5.11b. In this hypothesis, our results show that four out of the six paths allow to deliver traffic among geographically distributed datacenters without going out from the infrastructure owned and managed by Amazon. The remaining two paths instead imply the transit through external ASes, from which Amazon probably buys transit bandwidth. All these ASes are tier-1 (namely, *Level 3*, *TATA Communications*, *NTT America*, and *Telefonica*). A schematic view over AS-level-path graph is reported in Figure 5.11c. As shown in the figure, is some cases (i.e., where explicitly stated) the chosen VM size may impact the AS path traversed. Results are summarized in Table 5.5. While the number of hops traversed ranges from 9 to 20 hops, the number of domains traversed varies from 1—when only Amazon AS is traversed—to 3.

It is worth noting that AP and SA are the worse-connected regions in terms of external ASes to be traversed. It is known that the growth of cloud infrastructures is driven by multiple factors such as appealing environmental, financial, and political climates. The high concentrations of Internet exchanges and high-performance network infrastructures are not necessarily the most relevant factors to be taken into account to determine the location of a new datacenter to be deployed. Indeed, the existence of friendly governments, tech sectors, or highly educated populations are also key factors commonly considered. Therefore, the evolution of the communication infrastructure could be also a result of the deployment of the datacenters. This is in accordance with the fact that AP and SA are the regions made available most recently among the ones considered—being launched in 2010 and 2011, respectively.

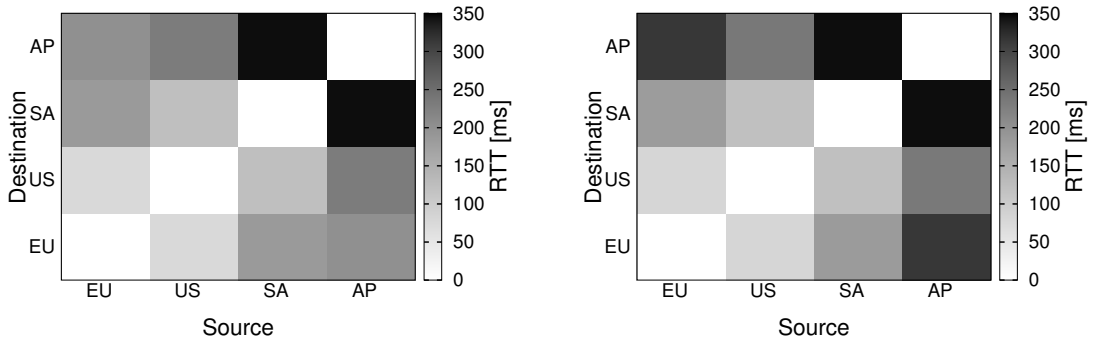
Experimental results obtained with different VM sizes (AP→SA) suggest the adoption of different routing policies for VMs of different sizes. The adoption of external network providers in case of AP or SA region gives also a possible explanation for higher data-transfer costs. They are probably aimed at discouraging intensive network usage by cloud customers, and at guaranteeing proper revenues to the cloud provider according to its business plans also in regions where proprietary network infrastructures have not been deployed yet.

Table 5.5: IP-level hops and domains traversed for each pair of regions in Amazon wide-area inter-datacenter network. IP-level path length and AS-level are symmetric if not explicitly stated otherwise.

Inter-datacenter path	Number of hops traversed	Number of domains traversed	Domains traversed
EU↔US	12	1	Amazon.com, Inc.;
EU↔SA	9/10	1	Amazon.com, Inc.;
US↔SA	14	1	Amazon.com, Inc.;
US↔AP	15	1	Amazon.com, Inc.;
EU↔AP	16	2	Amazon.com, Inc.; NTT America, Inc.;
SA→AP (M)	13	3	Amazon.com, Inc.; Level 3 Communications, Inc.; TATA Communications (AMERICA) Inc.;
AP→SA (M/XL)	13	3	Amazon.com, Inc.; Level 3 Communications, Inc.; NTT America, Inc.;
SA→AP (XL)	20	3	Amazon.com, Inc.; TATA Communications (AMERICA) Inc.; Telefonica International Wholesale Services, SL;

5.4.5 Latency

Our experimental data shows that latency and throughput are in general not highly correlated. In many cases, high throughput implies lower latency, but low throughput does not necessarily imply high latency and vice versa. In general, latency could not be considered the main cause of the throughput degradations.



(a) Amazon. Average values in ms (towards destination): 154 (EU), 143 (US), 218 (SA), 258 (AP).

(b) Azure. Average values in ms (towards destination): 193 (EU), 147 (US), 217 (SA), 298 (AP).

Figure 5.12: Latency between different regions. Azure exposes slightly higer values than Amazon. For both providers, US region is the one with the lowest average latency towards the considered destinations, while AP expose way higher values.

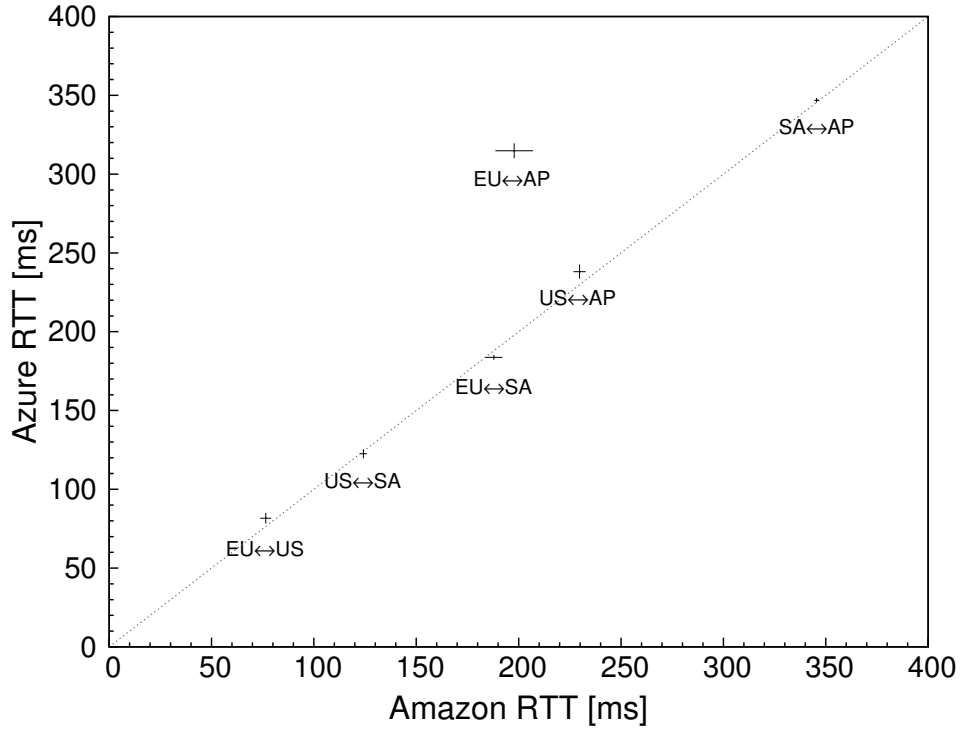


Figure 5.13: Comparison of inter-datacenter latencies experienced when relying on different providers. Crossing points identify mean values, while bars report standard deviation for both Amazon (x axis) and Azure (y axis). Latency across homologous pairs resulted to be comparable except than for EU↔AP. Little variability has been observed.

Figure 5.12 provides an overall view over performance in terms of latency (RTT) for both providers. As expected, latency values appear to be symmetric, although non-negligible differences exist across different regions. US region is the one with the lowest average latency towards the considered destinations, while AP exposes way higher values.

Figure 5.13 compares the mean latencies experimented between homologous regions for Amazon and Azure, and their variability. Average latency is equal to 193.61 ms and 214.67 ms for Amazon and Azure, respectively. On average, RTT values appeared to be smaller than the ones reported by previous works [48, 83]. Indeed, these values resulted to be also compatible with the delay constraints hypothesized for the application proposed in [48]. In general, as shown by the limited standard deviation, latency proved to be very stable over time, for both providers. Figure 5.14 investigates the variability of the latency from a different angle, showing the cumulative distribution of the CoV of the RTT for different experiments. For both providers we observed CoV values smaller than 0.05 for more than 80% of 5-minute-long experiments, and smaller than 0.1 in all but two

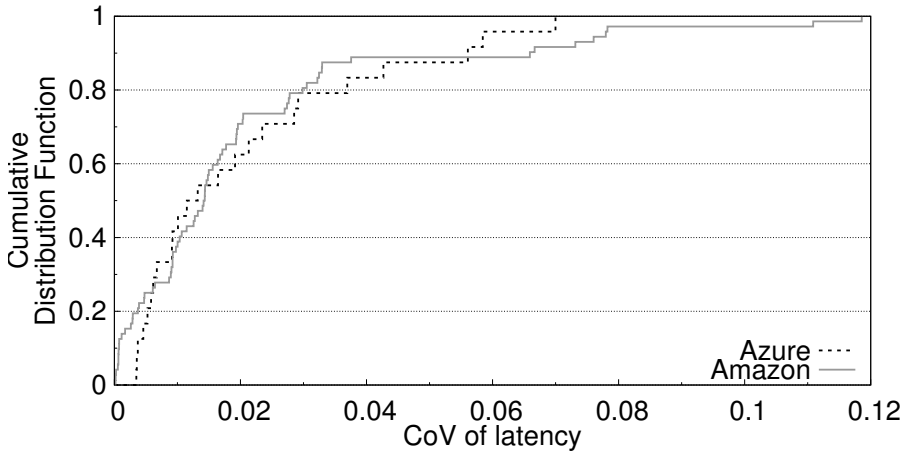


Figure 5.14: *CoV* distribution of latency (RTT) across different 5-minute-long experiments. Both providers expose little variation for latency.

of around 70 experiments for Amazon. Interestingly, the worst-performing region pair in terms of latency (i.e. SA↔AP) exposed the stablest performance over time.

In more details, while experimented RTT is similar across providers for five out of six region pairs, EU↔AP shows a markedly higher latency for Azure (197 ms vs. 315 ms). Interestingly, experimental data shows how for Azure the latency measured for EU↔AP is roughly equal to the sum of the latencies measured for EU↔US and US↔AP, thus disclosing that a probable strategy implemented by the provider to send traffic form EU to AP consists in routing it through US. Although the lack of data about the path does not allow to verify this conjecture, this would also motivate the higher latency with respect to Amazon.

In the following, we propose the results of analyses aimed at further investigating the nature of the delay perceived by the cloud customer. Figure 5.15 shows how the average RTT varies with the geographic distance between datacenters. As expected, distance proved to have large impact on RTT performance. However, propagation delay amounts for just a limited portion of the overall delay.[1] Indeed, Figure 5.15 shows how the delay without the propagation quota (filled squares and circles), can be coarsely clustered into 4 ranges: (i) 0–50 ms; (ii) 50–100 ms; (iii) 100–150 ms; (iv) 200–250 ms.

[1]In more detail, the propagation delay has been evaluated as $D_p = \frac{d}{c}$, where d is the distance (calculated by adopting *Vincenty's formulae*) between the source and destination datancenter (whose location has been obtained from [119]), and c is the speed of light. Since the one considered is the *minimum distance* and the actual propagation speed is slightly smaller than the speed of light, the one evaluated represents a lower bound of the actual propagation delay.

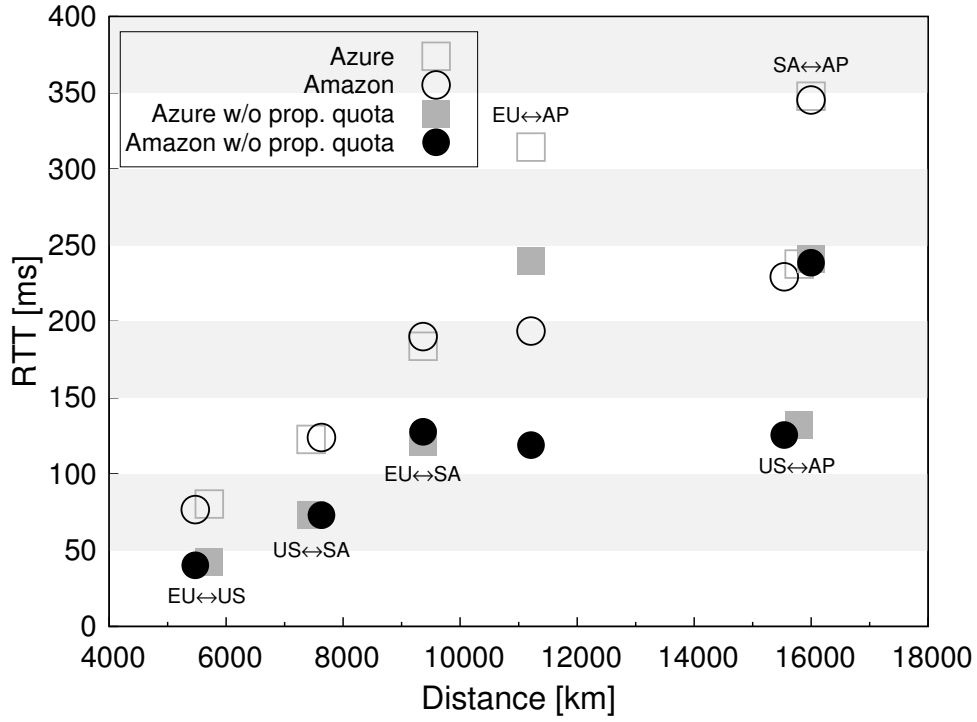


Figure 5.15: Latency vs. distance. As expected, latency is impacted by geographical distance between datacenters (empty squares and circles). However, propagation delay amounts for just a limited portion of the overall delay. Also excluding the propagation quota—known to be the quota mainly impacted by geographic distance—the delay grows with the distance (filled squares and circles). Results suggest that technologies with different performance levels are deployed.

This delay—considering negligible the computation done at the destination—is given by transmission, elaboration, and queuing quotas. All these quotas basically depend on the characteristics of the end-to-end connections (link speeds, computational capabilities of the devices along the path, congestion of the queues, etc.)—i.e. on the *performance of the technologies adopted*. They are therefore influenced by the economic investments done in the communication infrastructure. Interestingly, Figure 5.15 shows how the geographic distance indirectly influences also the delay without the propagation quota: the larger the distance, the larger the delay, even without the propagation quota. This can be explained by the impact of the distance on the technological choices done by the providers: due to technological or budget constraints, longest paths are provided with less performing technologies. We cannot exclude the traffic across that specific paths is subjected to more complex elaborations (e.g., due to the presence of middleboxes along the path performing specific tasks). We left the analysis of these specific aspects as a future work.

5.4.6 Impact of availability zones

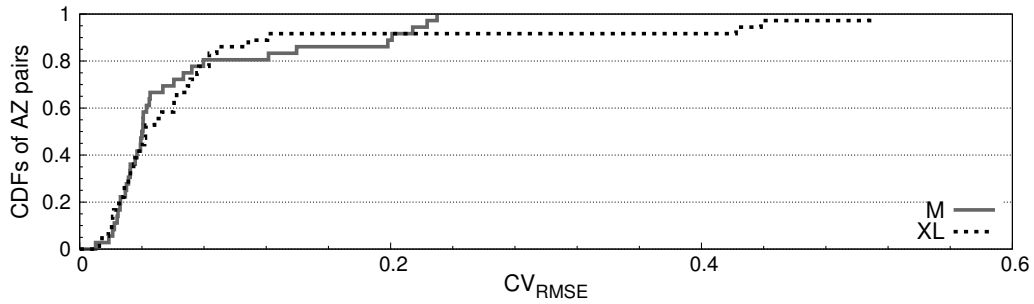


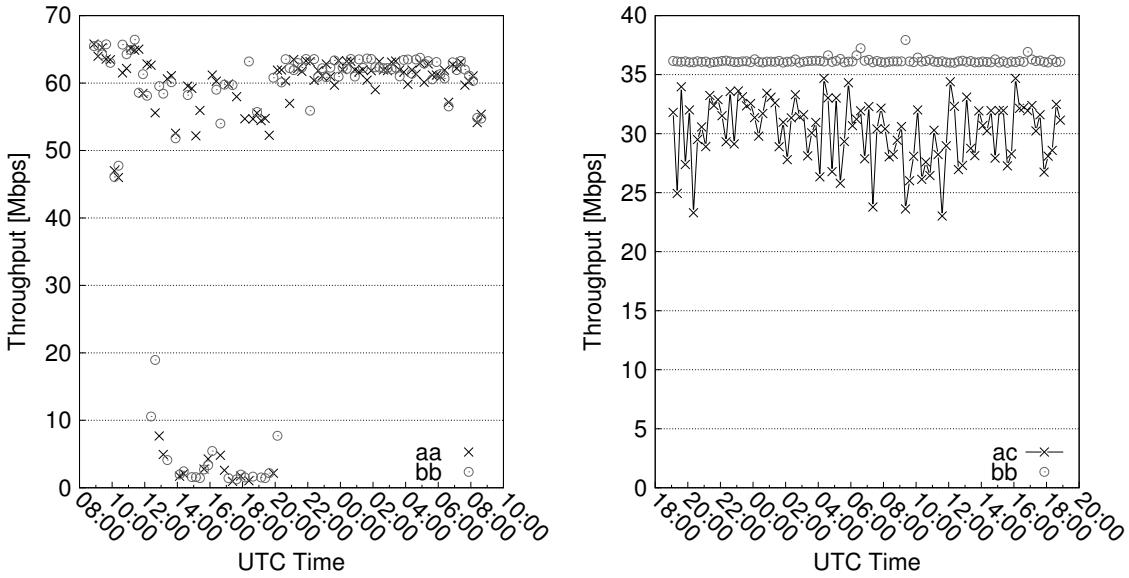
Figure 5.16: CV_{RMSE} distribution of the throughput across different Amazon AZs. 90% of samples has values lower than 0.2, showing how the choice of AZ is non-influential for throughput performance in most of the cases.

In our study we have also taken into account the impact of selecting different availability zones (AZs) inside a region, i.e. we investigated how performance may vary when choosing one of the isolated locations made available inside a region by Amazon.

We set up the experimentations such that 5-minute-long experiments have been launched over multiple distinct AZ pairs at the same time. Therefore we have information about the performance evolution over the time for multiple AZs in parallel. The main outcome of this analysis is that changing AZ gives no clear advantage in terms of achievable throughput. Figure 5.16 reports the distribution of the *coefficient of variation of the root mean square error*[2] (CV_{RMSE}) as an indication for the difference of throughput performance perceived along paths between different AZs. The figure shows this result for both M and XL VMs: CV_{RMSE} results lower than 0.2 for 90% of the samples, underlining how throughput performance along paths connecting differing AZs in the same region is the same, on average. In the following, cases showing higher values for CV_{RMSE} are deepened (Figure 5.17).

In a limited number of cases, severe performance degradation lasting for several hours has been identified, showing throughput dropping down to values smaller than 5 Mbps. An example for AP→EU is reported in Figure 5.17a. Pairs of homologous samples report different throughput values for different AZs (thus justifying the high CV_{RMSE}). However, in the period between 2:00 p.m. and 8:00 p.m., all the tested logical paths connecting

[2] $CV_{RMSE}(X,Y) = \frac{\sqrt{E[(X-Y)^2]}}{E[E[X],E[Y]]}$ where X and Y are the empirical distributions of the throughput values collected considering two distinct pairs of availability zones.



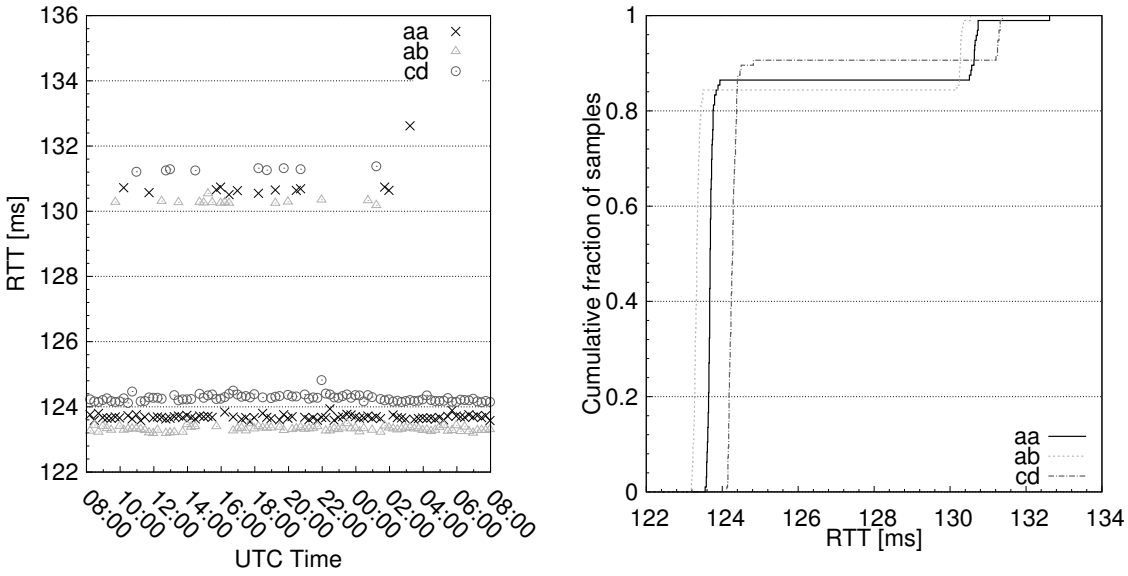
(a) Performance degradation involving all the AZs (AP→EU).

(b) AZs having consistently worse performance (AP→SA).

Figure 5.17: Examples for the interesting cases.

disjoint AZ sets (namely `aa` and `bb`) show a sever degradation of performance. This example shows how AZs although guaranteeing site isolation, revealed to be not completely independent from the network point of view. We have observed different cases similar to the one described. On the other hand, we have also observed cases in which the degradation involves only one AZ pair and not the others. This happened for a single pair of regions in our dataset (AP→SA). Figure 5.17b shows how the performance monitored for the AZ pair identified by `ac` is consistently lower than the homologous identified by `bb` during the entire 24-hour-long observation period.

Finally, extending latency considerations to Amazon AZs, led us to point out some interesting patterns, for which an example is reported in Figure 5.18. The figure shows that different AZ pairs present consistently but slightly distinct latency values. Considering that AZs are independently mapped to identifiers for each customer account [19], and that latency values proved to consistently depend from AZs, we believe that latency information proves to be useful to cloud customers to identify the actual AZ assigned inside a region. This information can be also leveraged to set up resources into the most convenient AZ, according to potentially existing performance discrepancies found. Even more interestingly, intermittent latency deterioration has been spotted in several circumstances,



(a) Timeseries. (b) Distribution.

Figure 5.18: Latency between different AZs (SA→US). A constant additive latency offset is observed in case of performance degradation.

that equally affected the different AZs, i.e., causing a fixed latency shift. Figure 5.18 shows an instance of this phenomenon. This behavior suggests how the root cause of this performance deterioration is placed in the portion of the communication path shared by all the traffic between the two regions, thus not depending from the specific AZ chosen.

Chapter 6

Cloud-to-user network performance

In this chapter, we focus on the performance of the cloud-to-user network. In more details, we take into account the performance of a cloud storage application (Amazon S3 [120]) as an interesting case study.

The chapter is organized as follows: § 6.1 introduces S3, the related concepts and nomenclature, and briefly summarizes the results of prior works that investigated it; § 6.2 details the adopted methodology; finally § 6.3 presents the results of our investigation.

6.1 The Amazon S3 case study

Cloud storage denotes a family of increasingly popular services for archiving, backup, and even primary storage of files, heavily promoted by recent advances in networking technologies. The increase in the need for storage resources has prompted many organizations to outsource their storage needs. In this section we describe the services considered in our experimental study, also introducing the names and concepts adopted in the following. We then discuss the related literature.

6.1.1 Background

Amazon Simple Storage Service (S3) is the general purpose storage as a service provided by Amazon [120]. User data is organized by means of *objects* stored in *buckets*, i.e. logical units of storage uniquely identified and belonging to one of the AWS regions.

Costs for the customer depend upon the storage class (standard, infrequent access, or long-term archive) and the geographic region in which the bucket is placed, according to a pay-as-you-go model. In more details, cost is calculated as the sum of three quotas: (i)

storage (depending on the size of stored objects), (ii) number of requests (e.g., upload or download), and (iii) data transfer (depending on the volume of traffic transferred from the bucket) [120].

CloudFront (CF) is the global Content Delivery Network (CDN) service offered by Amazon [121] and integrates with S3 in order to distribute contents to the end users with low latency and high data-transfer speeds. Data is distributed to the users through the global network composed of the AWS *edge locations* spread all over the world [19]. CF can be leveraged in combination with S3, by simply activating it, with no need for further configuration. Notably, CF is associated to storage and data transfer costs similar to S3. Request costs are markedly higher (around 2×).

6.1.2 Related literature

As S3 has been the first Amazon web service publicly available, a number of works tried to shed light on its performance [122, 79, 76, 34, 74, 123]. Differently than most of the studies above [122, 79, 76, 34], our work aims at focusing on the performance of content remote delivery, i.e., it aims at investigating the quality of service perceived by users that retrieve contents from vantage points (VPs) not placed into the cloud. With this goal in mind, our approach only leverages active measurements. On the one hand, this characteristic frees our study from the need of any privileged point of view [74, 123], thus also guaranteeing easier repeatability. On the other side, the methodology we enforced allows to better evaluate the impact of factors under the control of the customers, e.g., the region hosting the content, not limiting the validity of the study neither to the service usage patterns observed in traffic captures [74, 123] nor to a specific geographic zone [122, 79]. Indeed, in order to obtain a significant characterization of the performance of the cloud storage delivery service with respect to geographically distributed users, differently than previous works [76, 34] we leverage a set of geographically distributed VPs. In addition, our study provides an up-to-date view of the performance of the service under investigation, in front of both (i) the evolution of the cloud network infrastructure over the time, and (ii) the new services not considered in most of the previous works as not available at that time (e.g., a larger number of regions in which contents can be placed, or the integration with CDN services).

Table 6.1: Summary of factors and considered values.

Factors	**Values**
Storage Classes	Standard CloudFront
Cloud Regions	North Virginia (US) Ireland (EU) Singapore (AP) Sao Paulo (SA)
File Sizes	1B 1 KiB 1 MiB 16 MiB 100 MiB

6.2 Methodology

In this section we describe the setup and the choices implemented in our experimental study. We have taken into account a number of factors that may impact the service performance experienced by users. These factors are summarized in Table 6.1 and will be briefly discussed in the following.

6.2.1 Factors of interest

In order to take into account different use cases possibly generated by the needs of different categories of applications, we have considered two different **storage classes** e.g., Amazon S3 standard (hereafter simply S3) and CF. The latter is expected to have better performance at higher costs.

As of today, Amazon has datacenters in 12 regions around the world. Due to experimental cost constraints, for our experimental campaigns we have identified a subset of 4 **cloud geographic regions** among all possible ones: North Virginia (hereafter US), Ireland (EU), Singapore (AP), and Sao Paulo (SA). We have picked a region per continent, in order to ensure geographical diversity to our dataset. In each of these regions, we have created a bucket that contains objects of various sizes, from 1 B to 100 MiB as shown in Table 6.1. **Object sizes** have been selected to assess network performance against objects of different nature, possibly related to diverse use cases. We have used the standard *HTTP GET* method to download these files from the buckets, in order to emulate the

Table 6.2: Selected VPs and detailed locations.

Node ID	**Location**	**Label**
pl2.cs.unm.edu	Albuquerque (New Mexico, US)	US1
planetlab1.acis.ufl.edu	Gainesville (Florida, US)	US2
planetlab1.unl.edu	Lincoln (Nebraska, US)	US3
planetlab01.cs.washington.edu	Seattle (Washington, US)	US4
planetlab1.postel.org	Los Angeles (California, US)	US5
ricepl-2.cs.rice.edu	Houston (Texas, US)	US6
planetlab1.cs.du.edu	Denver (Colorado, US)	US7
pl1.rcc.uottawa.ca	Ottawa (Ontario, Canada)	US8
planetlab1.ifi.uio.no	Oslo (Norway)	EU1
planetlab2.inf.ethz.ch	Zurich (Switzerland)	EU2
planetlab1.diku.dk	Copenhagen (Denmark)	EU3
planetlab3.cslab.ece.ntua.gr	Athens (Greece)	EU4
planetlab-2.man.poznan.pl	Poznan (Poland)	EU5
peeramidion.irisa.fr	Rennes (France)	EU6
planetlab1.cesnet.cz	Prague (Czech Republic)	EU7
pl1.sos.info.hiroshima-cu.ac.jp	Hiroshima (Japan)	AP1
planetlab4.goto.info.waseda.ac.jp	Tokyo (Japan)	AP2
planetlab-2.scie.uestc.edu.cn	Chengdu (China)	AP3
planetlab-1.sjtu.edu.cn	Beijing (China)	AP4
planetlab-n2.wand.net.nz	Hamilton (New Zealand)	AP5
pl1.eng.monash.edu.au	Monash (Australia)	AP6
ple2.ait.ac.th	Bangkok (Thailand)	AP7
pl2.zju.edu.cn	Hangzhou (China)	AP8
planetlab1.cs.otago.ac.nz	Dunedin (New Zealand)	AP9
planetlab2.pop-mg.rnp.br	Belo Horizonte (Brazil)	SA1
planet-lab1.itba.edu.ar	Buenos Aires (Argentina)	SA2

common behavior of the vast majority of cloud-storage customer applications [79].

Moreover, to take into consideration the heterogeneity of users of the cloud-storage services, and therefore the ability of these services to serve users spread world-wide, we have adopted geographically distributed VPs, leveraging the facilities made available by the PlanetLab infrastructure [124]. We have selected 26 geographically distributed PlanetLab nodes that have been labeled according to the VP region they belong to, using the same nomenclature adopted for cloud regions: North America (US), Europe (EU), Asia-Pacific and Oceania (AP), South America (SA). Table6.2 reports the selected PlanetLab nodes and their detailed locations. Nodes have been selected according to PlanetLab availability (no more than two stable nodes were available in the SA region during the experimentation period).

6.2.2 Experimental campaign and dataset

All the results presented in this section refer to experimental campaigns conducted between January and February 2016. In order to collect the dataset we refer to, the VPs have been instructed as detailed in the following.

Each VP repeatedly performed *download cycles* over 12 days. Each cycle is composed of 40 sequential download requests spaced out by 10 seconds and uniquely identified by a combination of factors in Table 6.1, i.e. storage class, cloud region, and object size.

Download cycles are repeated from each VP every three hours. In order to avoid an excessive number of simultaneous requests towards the same storage bucket, we have split the VPs in three groups whose experiments started with a ± 0.5-hour offset. In addition, the order of the download requests randomly changes at each cycle. After every download request, a *TCP-traceroute* is performed toward the IP address that has served the download, in order to trace the information related to the path and estimate the RTT to the bucket.

6.3 Experimental results

In this section the main results stemming out from the experimental campaign will be discussed.

6.3.1 General overview of the performance.

Our experimentations confirm that the size of the object to retrieve through the network heavily impacts the measured performance [34, 79], independently from the VP. We report here the distribution of the performance in terms of the average *goodput* per download request calculated over all the dataset. As shown in Figure 6.1, the larger the size, the higher the measured goodput is. In more particulars, 1 MiB, 16 MiB, and 100 MiB reported values of the same order of magnitude, whereas 1 B and 1 KiB led to markedly lower values (5 and 2 orders of magnitude lower, respectively).

Accordingly, users deal with different performance levels in terms of goodput when adopting the S3 service to retrieve contents of different sizes. For instance, from the performance point of view, it is convenient for the user to download a single 100 MiB blob containing clustered contents (e.g., an archive containing a hundred pictures) instead of

a hundred 1 MiB files (e.g., single pictures), as the total time needed to retrieve contents would be smaller.

From the monitoring viewpoint, the slight variation observed on average between 16 MiB and 100 MiB sizes, suggests that 100 MiB is enough to obtain a good estimation of the maximum goodput achievable. On average, a limited error is done when considering 16 MiB objects (10.4%), while larger errors when referring to 1 MiB contents are done (50.3%, on average). In the following analyses, if not explicitly stated otherwise, we will restrict the results presented to 100 MiB objects, as being the size reporting the best performance observed, on average.

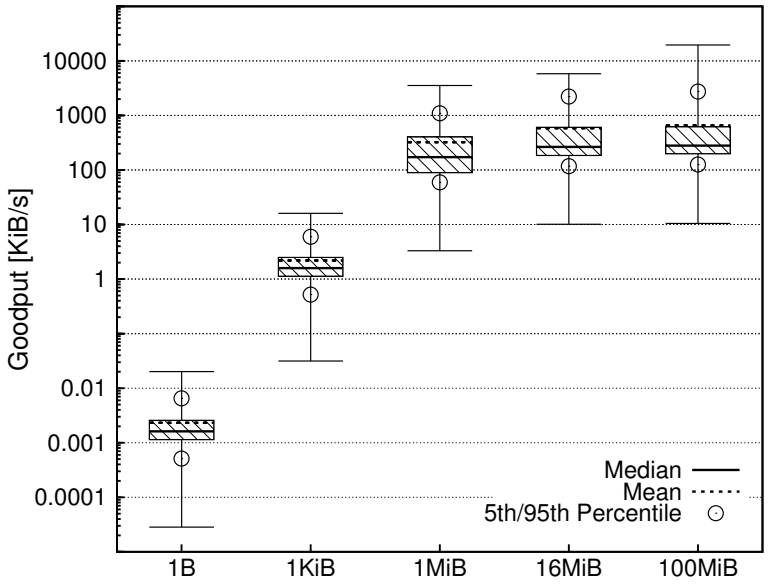


Figure 6.1: General overview of S3 performance grouped by file size. Goodput heavily depends on object size: the larger the size, the higher the goodput.

6.3.2 Impact of the geographic region

Our results reported that the measured performance may be heavily impacted by the placement of both the bucket and the VP. In order to evaluate how the performance changes when relying on cloud datacenters placed in different geographic regions, in this section we compare the performance of the four cloud regions considered, as observed from the 26 distributed VPs. Results are reported in Figure 6.2. Considering the goodput average values, US, EU, SA, and AP cloud regions reported 793.8, 779.8, 531.2, and 503.1, KiB/s, respectively. Therefore two performance classes can be easily identified:

US and EU versus SA and AP, where the former performs 34.3% better than the latter. Counterintuitively, AP and SA are also associated to higher network-transfer costs with respect to EU and US.

On the basis of the adopted cloud deployment strategies that often see customers leveraging a single cloud region [88, 123] these results are of interest for optimally choosing the cloud region to rely on. Indeed, if the cloud customer has no knowledge of the location of the users willing to retrieve contents, US and EU represent the best available choices, on average.

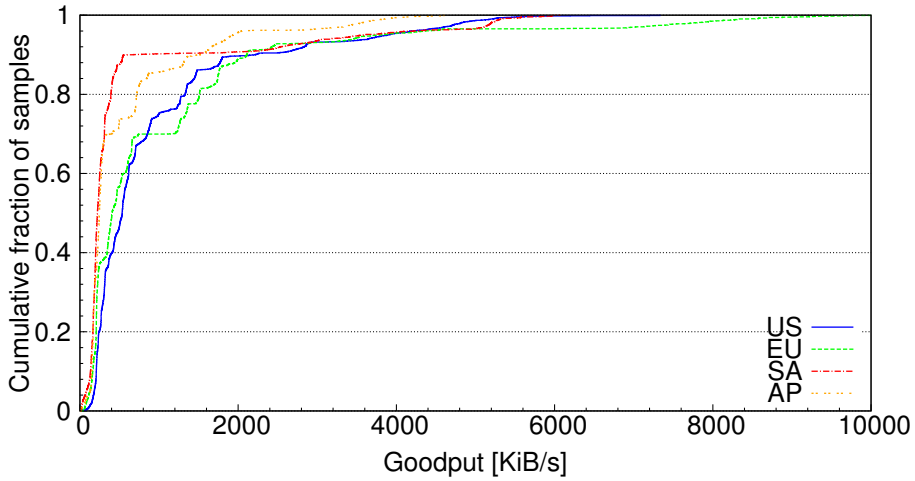


Figure 6.2: S3 performance for 100 MiB objects, grouped by cloud region.

As expected, results strongly depend also on the placement of the VP. Figure 6.3 reports detailed information about how the goodput is subjected to changes on varying source VP and cloud region. While in some cases, for a fixed VP the performance is not subjected to significant variations when relying on buckets placed in different cloud regions (e.g., AP9, SA2, or US5), in other cases a non negligible discrepancy is measured on changing buckets (e.g., AP7, SA1, or US3). It is evident in some of these cases how the globally optimal choice (EU or US region) is dramatically outperformed by local optimal choices.

Figure 6.4 summarizes the general trends observed by considering the aggregated values for both VPs and bucket regions. As already observed for detailed results, lighter boxes placed on the diagonal show that the best performance for a given VP placement region is obtained relying on a bucket placed in the same geographical zone, on average. This aspect is markedly visible for AP, EU, and SA regions, whereas is less evident for

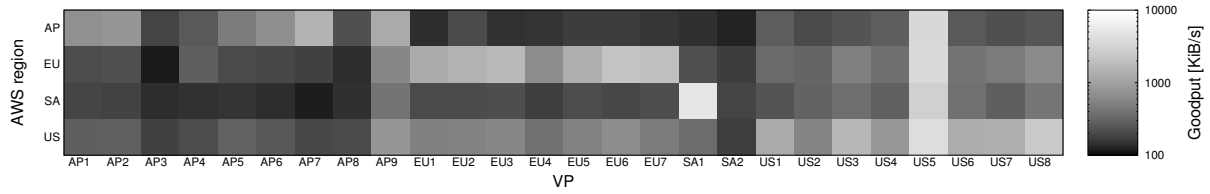


Figure 6.3: S3 goodput performance for 100 MiB objects. For each pair (VP, cloud-region) the average goodput is showed.

the US region. Interestingly, the best average performance is obtained considering VPs placed in SA retrieving contents from a bucket in the same region. Conversely, the worst performance, on average, is associated to VPs placed in SA retrieving contents from AP, and vice versa. More in general, the graph denotes performance symmetry with the only notable exception of the pair composed by SA and US. Indeed, the performance offered by the SA cloud bucket to VPs placed in US is markedly better than the one perceived by SA VPs retrieving contents from the US bucket.

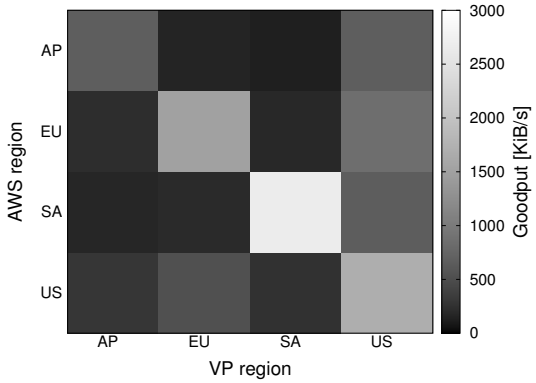


Figure 6.4: S3 performance for 100 MiB contents. For each pair (VP-region,cloud-region) the mean goodput is showed.

6.3.3 Evolution of the performance over time.

Considering fixed (VP, cloud-region) pairs, the performance is stable over time in most of the cases.

Figure 6.5 shows the distribution of the CoV of the goodput calculated for the download requests which refer to 100 MiB objects. As shown in the figure, the CoV is lower than 0.2 for the 80% of the (VP, cloud-region) pairs. This guarantees that the observations made in the above sections are consistent over time. The long tail of the distribution is

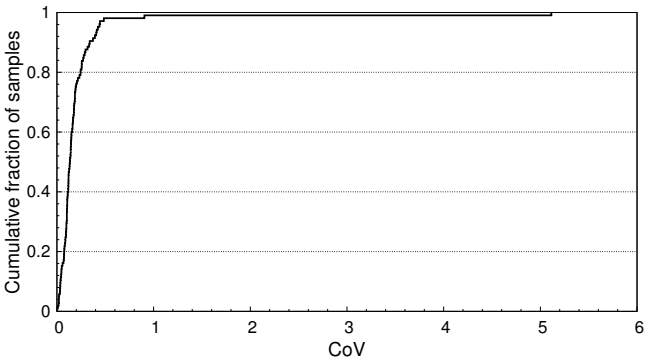


Figure 6.5: Distribution of performance variability over time in terms of CoV for differing (VP, cloud-region) pairs.

generated by the only two pairs whose CoV is higher than 0.4, associated to two outliers.

In spite of the low CoV values on average, a number of VPs exists for which the best-performing region changes over time. A notable example is reported in Figure 6.6. As shown, for the VP considered (AP3), the US region reported better performance for non negligible periods of time although the AP bucket performs better both on average and on median. These results give useful hints to the customer, and show how deployments relying on multiple buckets can be profitable not only from the availability and security point of view, but also for what concerns the available performance.

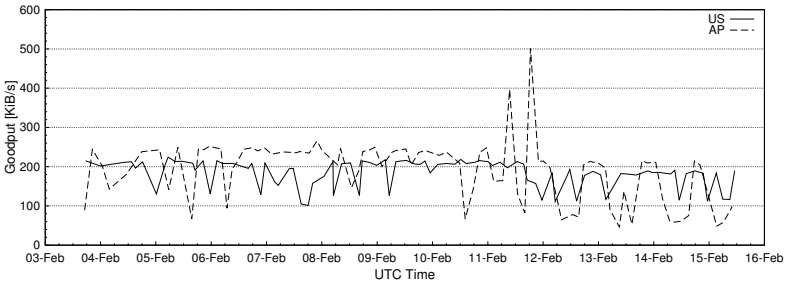


Figure 6.6: Time-series of S3 goodput for AP3.

6.3.4 Impact of CDN service adoption

In our experimentations we also tested the performance variation achieved by enabling the content distribution through the CF service. Note that from the customer point of view CF does not require any additional configuration but its explicit activation and implicates additional costs.

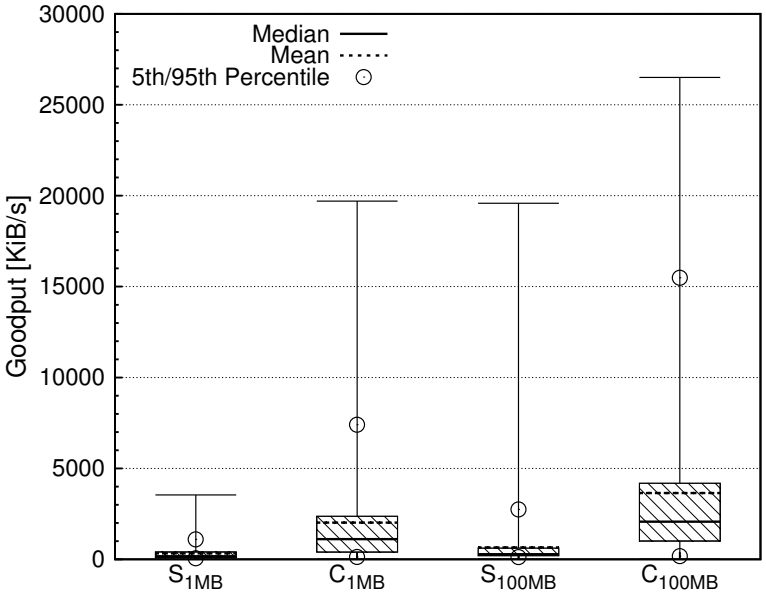


Figure 6.7: Comparison of S3 and CF goodput for 1 MiB and 100 MiB objects.

Figure 6.7 shows the distributions of the goodput obtained with both S3 standard and CF, considering two different sizes (1 MiB and 100 MiB). The beneficial impact of CF in terms of performance is evident, as it is able to deliver better performance on average (+522.7% and +458.6% when considering 1 MiB and 100 MiB objects, respectively).

Leveraging DNS names, each edge location observed has been associated to a geographic region. Figure 6.8 graphically shows how edge locations have been assigned to CF download request for each of the VP. In our experimentations we have been served by 46 out of the 54 available edge locations advertised by Amazon (21/21 in US, 14/16 EU, 2/2 in SA, and 9/15 in AP).

In most of the cases indeed, the content is downloaded from an edge location placed in the same geographic area (pink boxes in Figure 6.8), although some variability in the association has been observed. When a VP is not served by an edge location placed in the same geographic zone, the content is always retrieved from a US bucket. This is in accordance with the policies advertised by the provider, explicitly saying that when needed, requests may be redirected to an edge location belonging to a cheaper geographical zone. We found that in these case US region is the always chosen. Interestingly, we found that the VP placed in Beijing (AP4) has been always served by edge locations placed in the US.

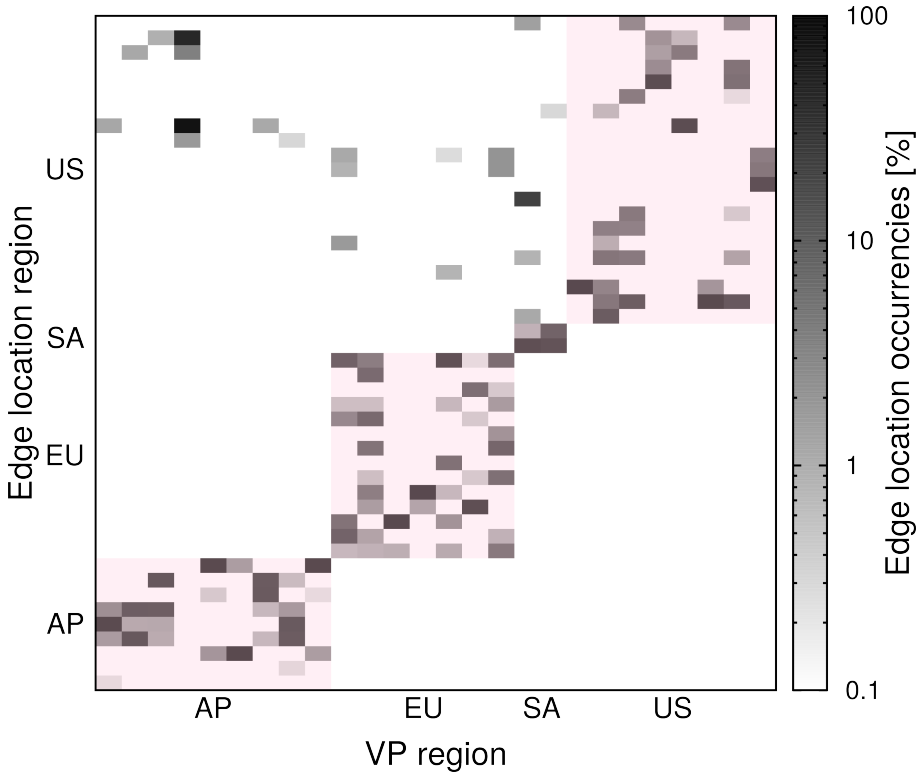
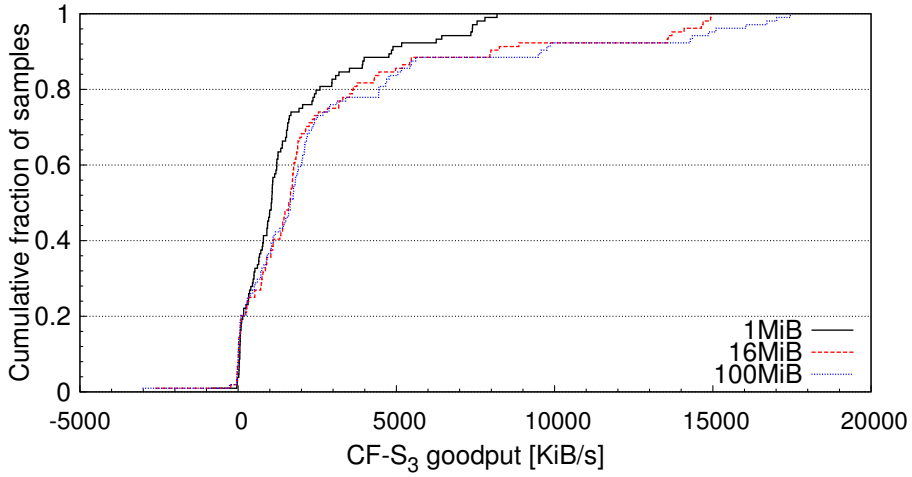


Figure 6.8: Occurrences of the associations of different edge locations (on the rows) to each VP (on the column) for objects of 100 MiB.

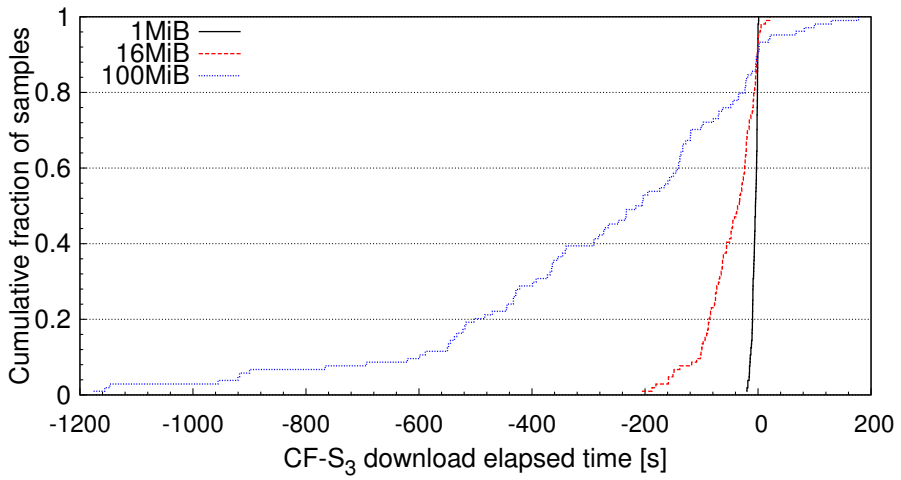
Considering the performance associated to each edge location seen from each VP, we found that only for 14 out of the 26 VPs the best-performing edge location on average, is the one seen with the highest frequency. In other words, for 45% of the cases, the strategy for associating an edge location to a VP leads to suboptimal results in terms of goodput [121]. The regions mainly subjected to this phenomenon are AP and EU (with 4 and 5 VPs, respectively), where this sub-optimal association leads down to 37% and 20% of the performance obtained in the best case, respectively.

We believe that the observed phenomenon is the result of the load balancing policies implemented by the provider to distribute requests across edge locations and are probably generated by edge-location overhead. Our results show that the strategy implemented may lead AP and EU VPs to severe performance degradation.

The distribution of the performance enhancement obtained enabling CF with respect to the different object sizes is reported in Figure 6.9. CF leads to a 2451.9 KiB/s mean improvement considering 1 MiB, 16 MiB, and 100 MiB sizes (i.e. +371.1%, on average). Improvements in terms of goodput up to 18,000 KiB/s have been observed for 100 MiB



(a) Goodput



(b) Download-time

Figure 6.9: Gain in terms of goodput and download time of CF against S3.

contents, whereas the gain observed for 1 MiB objects is always lower than 9000 KiB/s. However, 20% of the samples reported a negligible average improvement (i.e. lower than 181.4 KiB/s) with respect to the others. Most of them is related to VPs placed in the AP region.

Interestingly we found a number of cases for which adopting CF leads to worse performance than S3 (see the negative values in Figure 6.9). Considering all the combinations of file sizes, VPs, and cloud regions, we found 24 out of 312 cases in which S3 delivers, on average, better performance than CF, in terms of goodput and/or download elapsed time. All these cases refer to downloads performed from a VP placed in the same region

of S3 bucket, and involve 10 distinct VPs. While in some cases the performance degradation is negligible, in the worst case the negative variation in terms of mean goodput observed adopting CF, is equal to -60% (SA1 retrieving a 100 MiB object from a bucket placed in SA).

Final remarks. In this chapter we analyzed the performance delivered by the cloud-to-user network considering Amazon S3 as an interesting case-study. We found that performance varies with both the VP region and the cloud region. As expected, best performance on average is obtained when VPs retrieve contents from a datacenter placed in the same geographic region. However, considering the set of VPs taken in account as a whole, results show how EU and US datacenters are able to deliver better performance than AP and SA ones. Performance can be significantly improved leveraging the CDN service offered by the provider at additional costs. Surprisingly, a set of cases exists for which enabling the CDN service may even lead to performance degradation. We believe that these results are useful for both the provider and the customer, as offering an overall picture of the performance of the cloud-to-user-network, also highlighting cases in need of attention as suggesting that the enforced management policies are sub-optimal.

Chapter 7

Using monitoring data to automatically scale cloud resources

In this chapter we focus on the adoption of cloud monitoring data to implement strategies to automatically scale cloud resources. We propose a novel control technique that leveraging public-cloud monitoring information allows the customer to regulate the performance of a generic application to a desired and prespecified service level objective. The aim is to investigate the feasibility and effectiveness of adaptive approaches that rely only on the information available to the general customer (i.e., without requiring a performance model of the application, an estimation of the workload, or the knowledge of the infrastructure dynamics) for automatically scaling cloud resources in a real public-cloud environment.

7.1 Leveraging cloud resource elasticity to improve application performance

The *pay-as-you-go* pricing model is the characteristic that more directly captures the appealing economic benefit to the customer [25]. Indeed, the absence of up-front expenses allows capital to be redirected to core business investments. This is achieved through *resource elasticity*—i.e., the ability to add or remove resources at a fine grain and with a lead time of minutes rather than weeks—that allows closely matching resources to workload. For instance, a cloud customer can decide to start on-demand new servers or allocate more storage capacity just when needed, and without any up-front provisioning. In this way, it is possible to dramatically raise the resource utilization level (that is

estimated to be very low without cloud-based approaches—from 5% to 20% [125, 6]—due to typical *overprovisioning* needed to properly manage peak workload). Indeed, a wide number of different solutions has been proposed to take advantage of resource elasticity which range from social-network sentiment analysis [126] or market-based methods [127, 128], to control strategies with diverse degrees of complexity [129, 130, 131, 132, 133]. All methods propose different algorithms for resource scaling that can be implemented as either horizontal scaling (or *scaling out*, that consists in adding new server replicas to distribute load among all available replicas through load balancers) or vertical scaling (also known as *scaling up*, and consisting in changing the resources assigned to an already running instance, for example, allocating more computational or memory resources to a running VM). Note that the most common operating systems do not support on-the-fly changes on the available CPU or memory (*i.e.*, without rebooting), thus vertical scaling is practically unfeasible in most of the scenarios [134].

As building high-quality cloud applications is a critical research problem [135], *appropriately* dimensioning resources to applications in real time is a crucial issue in practical scenarios indeed, although elasticity theoretically allows cloud customers to dynamically scale resources according to changing demands. For example, since applications may face large fluctuating loads [134], it would be desirable to free the cloud customers from the burden of deciding how to adjust resources in presence of unplanned and unpredictable spike loads. In other words, it would be desirable to have an automatic strategy to adapt the amount of resources to be allocated on the base of the specific needs at any given time.

Approaches for automatically scaling out cloud resources (i.e. *autoscaling* approaches) like threshold-based rules and policies are very popular among cloud providers such as Amazon EC2 [19], or third-party tools like RightScale [136]. However, setting thresholds is a per-application task, and requires a deep understanding of workload trends. Thus, several solutions overcoming the limitations of threshold-based approaches have been proposed in the literature involving several components of the infrastructure, taking advantage of different kinds of techniques like queuing theory, reinforcement learning, workload prediction, or control theory [134, 137, 138, 139, 140, 141]. The most relevant are summarized in § 7.1.1.

Automatic resources allocation techniques deeply rely on the accurate real-time estimation of the actual conditions of the cloud system and of its performance, evaluated

through heterogeneous metrics representative of the QoS according to the specific application requirements [142, 143]. Usually resource allocation rules evaluate cloud performance from a single viewpoint, e.g., they consider just one metric at a time, such as CPU utilization or memory consumption [134].

In accordance with the recent literature, the metrics related to the shared intra-datacenter network infrastructures have to be also considered towards complete application scalability [144]. However, network resources and performance are often overlooked, although they could be critical either when different VMs share the same network and compete for resources in terms of available bandwidth, or when to each VM a dedicated network slice is associated [36]. In this sense, experimental results (see Chapter 4) confirm that commercial cloud providers such as Amazon or Azure guarantee to the newly instantiated VMs a fixed—although *a priori* unknown—network slice based on the their cost.

7.1.1 Related literature

Approaches for automatically scaling resources that leverage cloud elasticity have recently attracted the interest of the scientific community. In this section we discuss the most relevant ones, as more strictly related to the approach proposed in this chapter. For a more comprehensive review of all the approaches, we point the reader to [134].

Due to system complexity some of the works do not address the problem in real systems, but propose the extensive use of cloud simulators to mimic the dynamic response of the cloud system under the proposed control action. *Choi et al.* [145] proposed an autoscaling method—designed for hybrid infrastructures—that considers specific conditions such as application types, task dependency, user-defined deadlines, and data transfer times. *Fallah et al.* [146] seeks to offer an approach, based on learning automata, for the scalability of the web applications, in order to provide the best possible way for scaling up and down virtual machines. *Beloglazov and Rajkumar* [147] proposed a novel technique for dynamic consolidation of VMs based on adaptive utilization thresholds, exploiting the monitoring of CPU, RAM, network bandwidth, and storage.

The main drawback of these proposals is that they rely on performance models instead of reality, hence results strongly depend on the reliability of the simulations.

The experimental validation of cloud control strategies in a real environment has been often addressed by designing custom in-house testbeds such as server cluster or private

cloud deployments [129, 148, 149, 130, 131, 132, 133]. In all these works different control approaches, such as Proportional-Integral (PI), adaptive deadbeat, adaptive threshold mechanism, or feedback plus feed-forward have been analyzed to investigate the ability of the proposed strategies to regulate the average response time of the web application latency or the average CPU load. Since the control effectiveness has been only tested in the private experimental setups, the control design is often based on the precise knowledge of the cloud system implementation. This information is exploited to derive mathematical models of the system performance and/or for the prediction of the incoming variable loads acting on the cloud system. For example, *Padala et al.* [131] propose a Multi-Input-Multi-Output (MIMO) adaptive controller that uses a second-order ARMA to model the non-linear and time-varying relationship between the resource allocation and its normalized performance. *Xiao et al.* [150] designed and implemented a resource management system aiming at achieving good balance between overload avoidance (i.e. the algorithm has to avoid VM performance degradation caused by physical-machine (PM) overload) and green computing (i.e. the number of PMs has to be minimized to save energy). To achieve this goal the authors exploit their knowledge of the private cloud to design a load prediction algorithm that can capture the future resource usages of applications without looking inside the VMs. *Anglano et al.* proposed a cloud resource management framework exploiting feedback fuzzy-logic control able to adapt the CPU capacity allocated to the tiers of an application in order to match the needs dictated by the current workload. They implemented and evaluated their solution on a Xen-based virtualization environment.

Being tailored for in-house testbeds, all these approaches strongly leverage the precise knowledge of the inner mechanisms of the cloud systems under their control. Since the level of abstraction available in public cloud obfuscates it for commercial and security reasons, these approaches are not directly exploitable by common customers in public cloud environments.

Some of the literature is also devoted to the online prediction of the workload. For example, *Gambi and Toffetti* [132] exploited a Kriging model to predict the number of running VMs, the number of incoming requests, and the jobs enqueued. *Kalyvianaki et al.* [133] proposed workload prediction exploiting instead on-line Kalman filters. A different attempt to cope with workload variability has been instead proposed by *Bodik et al.* in [151]. Here a very complex strategy combines an integral control action with a performance models (to predict workloads) and a statistical machine learning techniques

to control internet data-centers. *Ardagna et al.* [152] proposed a distributed algorithm for managing SaaS cloud systems that addresses capacity allocation for multiple heterogeneous applications. The resource allocation algorithm takes into consideration a predicted future load for each application class and a predicted future performance of each VM, while determining possible SLA violations for each application type.

Obviously, the practical use of these approach is strongly limited from the correct implementation of these estimation techniques in real public-clouds environments. Indeed, in predictive (or proactive) approaches the prediction accuracy highly depends on the history window size, (i.e. the number of observations), and the adaptation window (i.e. the observation interval) [153]. Moreover, they require stable workloads to apply long learning [154].

It is worth noting that only few attempts have made in the very recent literature to solve the problem of cloud resources in public cloud for practical-use scenarios. For example, *Gergin et al.* [155] proposed a control architecture based on multiple fixed-gains Proportional-Integral-Derivative (PID) controllers. Note that these technique funding on a fixed structure with pre-established gain values, cannot adapt to highly time-varying and uncertain conditions. Furthermore as always happens with fixed-gain controllers, their effectiveness strongly depends on the tuning procedure which is heuristic and time-consuming. Moreover the control algorithm is able to allocate or deallocate only one VM at each time instant, so the control action is not able to rapidly counteract sudden peaks in the amount of requests.

Different control techniques try to overcome the limits of classical PID structure, proposing control actions that set their gains with respect to the working conditions or the control aggressiveness on the base of the management policy (*Ashraf et al.* [156]). Anyhow, all those approaches, do not consider heterogeneous metrics depending on the state of the network resources associated to the VMs, and implement resources scaling on the basis of computational aspects (mainly memory usage and CPU load).

7.1.2 Proposed approach

In this chapter, we propose a novel architecture based on a Fuzzy Logic Control (FLC) scheme to automatically scale out resources in public clouds. The proposed resource allocation strategy, takes into account hetreogeneous metrics related to different aspects of interest—i.e., computational or network capabilty—so that the resource provisioning

mechanism accomplishes the needs of the application from several points of view. These metrics are properly merged together by adopting recent methodologies designed to extract a representative QoS index from the heterogeneous observations.

In more details, the proposed control scheme implements a Proportional Integrative Derivative (PID) feedback control strategy to dynamically allocate resources at VM-granularity to applications running on public clouds and is aimed at guaranteeing a prespecified desired Service Level. The control strategy counteracts the presence of large fluctuating loads without the need of a previous knowledge of the system behavior and of the on-line estimation of disturbances acting on the cloud. It can be therefore implemented with only the information base available to the general customer. The control strategy also leverages *fuzzy logic* to implement a *gain-scheduling* policy so to adapt the control action with respect to the conditions of the public cloud which are not easy to predict since customers have limited or null visibility of the underlying management strategies enforced by cloud providers.

It is worth noting that when dealing with regulation processes in practical and unmodeled scenarios, PID controllers are usually the first choice [157]. Indeed, their major advantages are that they have a very simple structure, are easy to implement for on-line control, and show good results in terms of response time and precision, if control gains are well adjusted. However, they exhibit poor performance when applied to systems which are nonlinear, as controller tuning is difficult due to insufficient knowledge of the system dynamics [157], such as public clouds, whose implementation details and actual load may be heavily impacted by virtualization techniques and are not disclosed by providers. In these cases fuzzy-logic architectures are a good alternative as fuzzy logic is very suitable to solve practical control problems under uncertain and vague environments [158]. Therefore the proposed approaches aims at leveraging the benefits of both PID feedback controllers and fuzzy logic frameworks.

The main advantages of the proposed approach with respect to the state of the art can be summarized as follows.

- The proposed FLC approach does not require any detailed knowledge of the dynamics of the controlled cloud infrastructure or a performance model of the application (this makes this control solution flexible since it just relies on the knowledge available to the general customer and does not limit its use to private and in-house environments).

- The adoption of the fuzzy gain-scheduling algorithm provides robustness with respect to synthetic and real-trace workloads characterized by a high rate of variability, without requiring any *a priori* information on the current workload or its on-line measurement/estimation.
- Heterogeneous metric observations (properly merged together in a single performance index) allow to take into account the current state of the managed cloud application from different angles at once, according to the needs of the application.
- The proposed solution has been implemented and then extensively experimentally validated within the public cloud environment AWS EC2.

The remainder of the chapter is organized as follows. § 7.2 introduces the problem statement and presents the proposed approach, detailing the overall architecture and the design of the constituting blocks; finally § 7.3 shows the experimental setup used to evaluate the proposed solution in public clouds and all the results.

7.2 A scaling approach that leverages heterogeneous metrics

In this section, we detail the solution we designed and implemented to address the problem of resource scaling in public clouds. We first discuss the problem that the proposed control architecture aims to solve (§ 7.2.1) and the related reference scenario. Then we describe the overall architecture and each of its constituent block (§ 7.2.2–§ 7.2.5).

7.2.1 Problem statement

By leveraging resource elasticity, the *cloud customer* is able to decide at runtime resources leased by public the *cloud provider* and—consequently—the quantity of resources he/she pays for. In this framework, the proposed approach aims at *automatically dimensioning the set of resources allocated to an application*, in order to guarantee the desired performance level to *final users* in spite of the presence of a dynamically fluctuating *workload*. The cloud service paradigm we refer to is the IaaS and therefore resources are considered at VM granularity. VMs are activated or terminated on customer's request and thus they compose a cluster having dynamically changing size. Activating a number of VMs

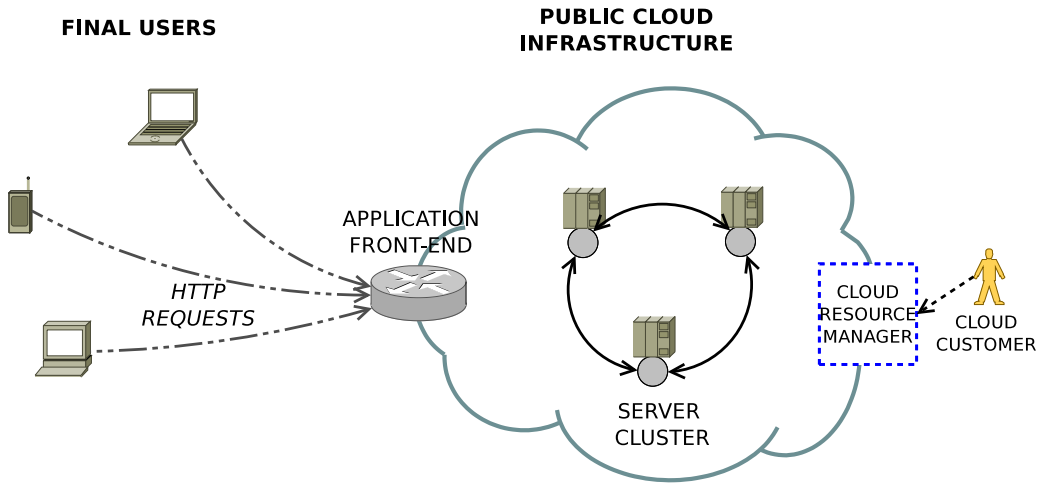


Figure 7.1: Reference scenario.

larger than the one strictly needed causes revenue loss to the cloud customer. Otherwise, if the set of VMs does not properly increase together with the workload, the performance perceived by the final user could dramatically fall down.

As an instance of the above described problem, in what follows we consider a generic application exposing a set of functionalities to final users through a web interface (see the reference scenario in Figure 7.1). Final users can submit tasks to the application through a *front-end*, that is in charge of scheduling users' requests and of forwarding them to the server cluster. These functionalities require communication among the VMs of the cluster for being accomplished. Note that this generic application is representative of some typical applications running onto cloud, such as applications for scientific computing (requiring communication among nodes to distribute shards of a complex task among a set of nodes) [159] or multi-tier applications (that separate roles—e.g., business logic and databases—into multiple layers which need to exchange data) [160, 161, 162].

According to the reference scenario, whose actors and terms are reported in Table 7.1, the problem statement to be solved is to keep the performance of the cloud application (i.e., its service level, SL) close to a predefined performance level (i.e., the service level objective, SLO).

7.2.2 Overall architecture design

In this section we describe the overall architecture of the system we have designed and implemented. Note that at each time instant the SL is evaluated merging the relevant

Table 7.1: Actors and terms.

Public Cloud Provider	• Offers cloud resources according to the *pay-as-you-go* paradigm
Cloud Customer	• Configures and manages the application by leveraging cloud resources • Is responsible for the service level guaranteed
Final User	• Takes advantage of the cloud services provided by the cloud customer through the application interface • Requires guaranteed service levels
Resources	• Consist in a virtual machine cluster of dynamic size (IaaS model) • Host applications in charge of executing tasks on final users' request
System Workload	• Consists in task execution started by requests generated by final users and executed by the VMs
Task	• Needs both computing and network communications among VMs to be accomplished
Service Level (SL)	• Is estimated by monitoring both CPU and network load of the VMs • Impacts tasks completion time and latency perceived by final users
Control Objective	• Keeping the VM cluster average CPU and network load close to the SLO in order to guarantee expected performance to final users and avoid revenue loss to the cloud customer

QoS parameter values through as a *Fitness Function* $\mathcal{F}$ (as detailed in § 7.2.5) hence the purpose of the regulation is to keep each of QoS parameters close to its desired optimal reference value in spite of unpredictable and time-varying disturbances (i.e., the workload). The overall architecture is shown in Figure 7.2.

Here, starting from the metrics observations gathered and processed by the *Monitoring Block* (M), the actual QoS index that represents the SL of the cloud application at each sampling time k, say y_k, is evaluated through the *Fitness Function Block* ($\mathcal{F}$). y_k is then compared to the optimal target, say the optimal QoS index value y_d, and the actual performance error, say $e_k = (y_d - y_k)$ acts as the input signal of the Fuzzy PID *Control Block* (C). Control gains (k_P, k_I, and k_D) are automatically and dynamically adapted by the *Fuzzy-Logic Block* (FL), on the basis of the actual error e_k and of its variation Δe_k, in order to dynamically counteract the effects of uncertainties and workload variations. The *Actuation Block* (A), connected to the public-cloud resource management interface, implements a scaling algorithm in order to actuate the control signal and starts or terminates a different number of VMs according to the control aggressiveness.

In the following the details for each of the block introduced will be provided.

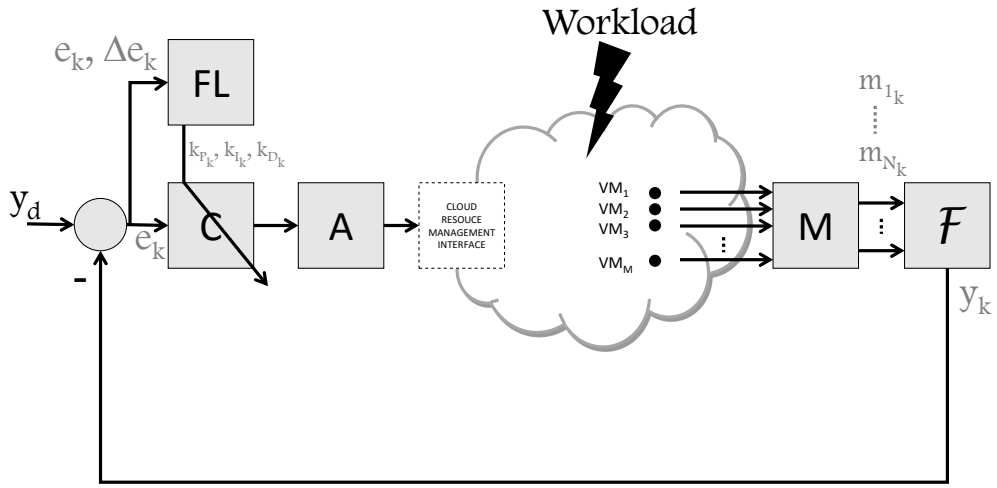


Figure 7.2: FLC architecture designed and implemented to solve the resource allocation problem in public clouds. Different blocks can be identified: the *Control Block* (C), the *Fuzzy-Logic Block* (FL), the *Monitoring Block* (M), the *Actuation Block* (A), and the *Fitness Function Block* ($\mathcal{F}$).

7.2.3 Control and actuation design

Since PID method cannot be applied to uncertain nonlinear systems with rapid changes of their dynamical behavior, resulting for example from the action of disturbances, the idea is to combine a PID structure with a fuzzy-logic control scheme in order to inherit the main advantages of both techniques. Essentially fuzzy logic provides a certain level of artificial intelligence to the conventional PID controllers, due to its ability of on-line adapt to system dynamics, by properly automatically adjusting its control parameters depending upon the error. With respect to our applicative framework, experimental instances disclose the large variability of public cloud behaviors under various operating conditions and workload variations, that can not be easily predicted. This suggested here to solve the regulation problem by complementing a PID with a fuzzy rule-based scheme, in order to automatically adapt the control action to the changing dynamics.

The approach keeps the simple structure of the PID controller and enables an easy connection between fuzzy parameters and operation of the controller. The knowledge base is combined by three separate rule bases, with simple rules for each different gain.

PID discrete time controller. The control strategy (see block C in Figure 7.2) is based

on a discrete PID structure that can be mathematically formalized as:

$$u_k = k_{P_k} e_k + k_{I_k} \Delta t \sum_{q=1}^{k} e_q + \frac{k_{D_k}}{\Delta t} \Delta e_k \tag{7.1}$$

being u_k the control action at time k; $e_k = y_d - y_k$ the closed-loop error (i.e. the difference of the desired output y_d and the measured output y_k); $\Delta e_k = e_k - e_{k-1}$ the error change (or its first difference) and Δt the sampling interval set as one cycle. The control parameters that modulate the control effort are adjusted at each time instant k depending upon the actual closed-loop error dynamics e_k and the error change Δe_k , i.e. $k_{P_k} = k_P(e_k, \Delta e_k)$; $k_{I_k} = k_I(e_k, \Delta e_k)$; $k_{D_k} = k_D(e_k, \Delta e_k)$.

Fuzzy Gain Scheduling. The fuzzy rules and reasoning block (see the block FL in Figure 7.2) is used to on-line select the control parameters based on the values assumed by the two inputs e_k and Δe_k.

It is assumed that control gains are in prescribed ranges, i.e. $k_{P_k} \in [k_P^{\min}, k_P^{\max}]$; $k_{I_k} \in [k_I^{\min}, k_I^{\max}]$; $k_{D_k} \in [k_D^{\min}, k_D^{\max}]$. For convenience, the current values of PID gains at time instant k are normalized into the range between zero and one, i.e.

$$\begin{aligned} k'_{P_k} &= \tfrac{k_{P_k} - k_P^{\min}}{k_P^{\max} - k_P^{\min}}, \\ k'_{I_k} &= \tfrac{k_{I_k} - k_I^{\min}}{k_I^{\max} - k_I^{\min}}, \\ k'_{D_k} &= \tfrac{k_{D_k} - k_D^{\min}}{k_D^{\max} - k_D^{\min}}, \end{aligned} \tag{7.2}$$

while the normalized error and its normalized increment are given by

$$\begin{aligned} \widetilde{e}_k &= K_e e_k, \\ \Delta\widetilde{e}_k &= K_{\Delta e} \Delta e_k, \end{aligned} \tag{7.3}$$

being K_e and $K_{\Delta e}$ some scaling factors to be opportunely chosen. The definition of the parameters regions and, hence, the computation of the minimum and maximum value of each PID gain is carried out by off-line solving a constrained optimization problem (which is described in the following).

The fuzzy logic gain scheduler is designed according to the classical architecture of fuzzy logic systems [158] and its structure funds on four main components, namely the *knowledge base*, the *fuzzification interface*, the *inference engine*, and the *defuzzification interface* [163] (see Figure 7.3). The knowledge base system contains all the information required for the fuzzy system, namely the fuzzy control rule base and the data base. The

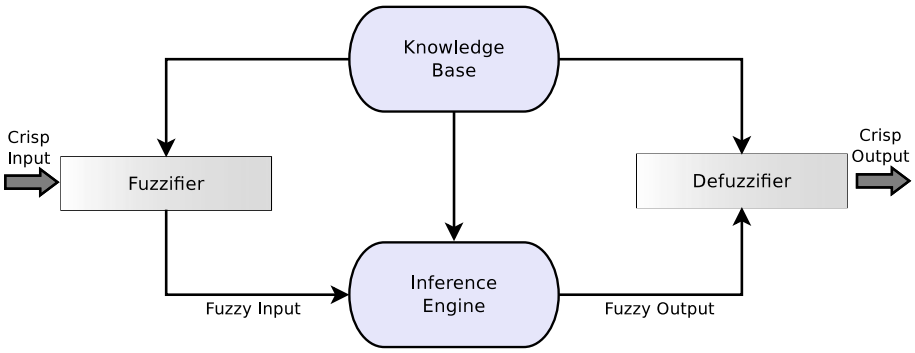


Figure 7.3: Basic configuration of a fuzzy system.

inference engine performs inference procedures upon the fuzzy control rules, while the fuzzification interface defines a mapping from a real-value space to a fuzzy space, and the defuzzification interface implements a mapping from a fuzzy space to a real-valued space. In what follows we provide details on each of the components for our two-inputs fuzzy PID configuration.

Given the normalized variables (eqs. 7.2–7.3), the rules base structure takes the following form:

$$\begin{cases} \texttt{If } \widetilde{e}_k \texttt{ is } A_i \texttt{ and } \Delta\widetilde{e}_k \texttt{ is } B_j \\ \texttt{Then } k'_{P_k} \texttt{ is } C_m,\ k'_{D_k} \texttt{ is } D_\lambda \texttt{ and } k'_{I_k} \texttt{ is } E_\gamma \end{cases} \tag{7.4}$$

where $i = 1, \ldots, I$ and $j = 1, \ldots, J$ represent the fuzzy states of the antecendents (being I and J the number of linguistic values associated to the antecendents); $m = 1, \ldots M$; $\lambda = 1, \ldots, \Lambda$; and $\gamma = 1, \ldots, \Gamma$ are the fuzzy states associated to the consequent (being M, Λ, and Γ the number of linguistic values associated with the controller outputs); and A, B, C, D, E are the fuzzy sets.

By opportunely setting K_e and $K_{\Delta e}$ in (7.3) (see Table 7.3), both the universes of disclosure of $\widetilde{e}(k)$ and $\Delta\widetilde{e}(k)$ have been scaled to $[-1, +1]$, while the corresponding membership functions are shown in Fig.7.4. Here *NB* stands for *Negative Big*, *NM* is *Negative Medium*, *NS* is *Negative Small*, *Zero* is *ZO*, *PS* represents *Positive Small*, and *PM* and *PB* stand for *Positive Medium* and *Positive Big*, respectively. Note that to cope with the real control problem, we adopt non-equal span membership functions since for highly nonlinear processes a fuzzy-logic controller with equal-span triangular membership function is not adequate to achieve good control results [164, 165].

With respect to the outputs of the gain scheduling fuzzy modules, C_m and D_λ can be either *Big* or *Small* (B and S, respectively), while E_γ can be instead *Very Big*, *Big*, *Small*

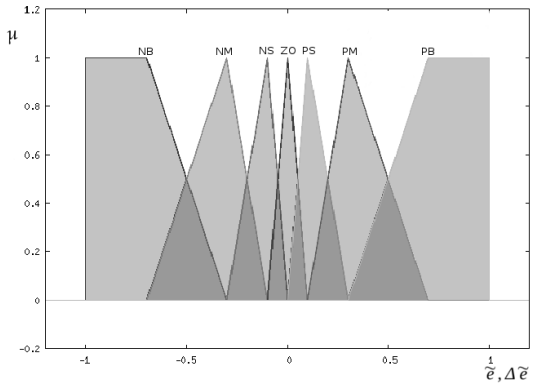


Figure 7.4: Membership function for $\widetilde{e}(k)$ and $\Delta\widetilde{e}(k)$.

and *Very Small* (i.e. *VB*, *B*, *S*, *VS*).

The derivation of the fuzzy rules in (7.4) is based on step time response of the process in order to fulfill both steady-state and transient requirements and, hence, to induce an ideal response of the process (in terms of rise time, over-shoot, settling time, and steady-state error), balancing control reactivity and achievable performance. In so doing the controller embeds the human operator experience and automatically applies it to the system. The resulting set of rules is given in Table 7.2a for $k_P^{'}$, in Table 7.2b for $k_D^{'}$, and in Table 7.2c for $k_I^{'}$. Such tables show that the fuzzy reasoning block incorporate 49 standard rules.

Concerning the fuzzy implication operator implemented by the inference engine, the choice fell out upon the zero-order Sugeno-type approach [166, 167] modeled by the operator `min(;)`, since this method has been shown to be computationally effective and to work well with optimization and adaptive techniques [168], which makes it very attractive for the implementation control approaches in a real world environments. Accordingly, the crisp normalized output is than generated through a weighted average of rule outputs [167]. Finally, the crisp control gains values to be provided at each time instant k are derived from the normalized ones as:

$$\begin{aligned} k_{P_k} &= (k_P^{\max} - k_P^{\min})k_{P_k}^{'} + k_P^{\min}, \\ k_{I_k} &= (k_I^{\max} - k_I^{\min})k_{I_k}^{'} + k_I^{\min}, \\ k_{D_k} &= (k_D^{\max} - k_D^{\min})k_{D_k}^{'} + k_D^{\min}. \end{aligned} \tag{7.5}$$

PID gains are computed according to equation (7.5) where usually, the minimum and the maximum values that determine the parameters range (i.e. $k_P^{\max}$, $k_P^{\min}$, $k_I^{\max}$, $k_I^{\min}$, $k_D^{\max}$, $k_D^{\min}$) are set according to the experience of designers. For example, higher value

Table 7.2: Fuzzy tuning rules.

(a) Fuzzy Tuning Rules for $k_P^{'}$.

		$\Delta\widetilde{e}_k$						
		NB	NM	NS	ZO	PS	PM	PB
$\widetilde{e}_k$	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	ZO	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

(b) Fuzzy Tuning Rules for $k_D^{'}$.

		$\Delta\widetilde{e}_k$						
		NB	NM	NS	ZO	PS	PM	PB
$\widetilde{e}_k$	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZO	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

(c) Fuzzy Tuning Rules for $k_I^{'}$.

		$\Delta\widetilde{e}_k$						
		NB	NM	NS	ZO	PS	PM	PB
$\widetilde{e}_k$	NB	VS	VS	VS	VS	VS	VS	VS
	NM	S	S	VS	VS	VS	S	S
	NS	B	S	S	VS	S	S	B
	ZO	VB	B	S	VS	VS	B	VB
	PS	B	S	S	VS	S	S	B
	PM	S	S	VS	VS	VS	S	S
	PB	VS	VS	VS	VS	VS	VS	VS

of k_P results in faster system response, but larger over-shoot; higher value of k_D results in slower system response, but smaller over-shoot. It is clear that the definition of gains regions based on heuristic methods, may result to be time-consuming, or not effective. Furthermore, often these methods do not provide any optimality of the solution, while a wrong selection can also induce a loss of stability in some critical cases. For these reasons the gain values determined by traditional Ziegler-Nichols method are here modified according to an optimization method to automatically select gains regions that improve the output performance of controlled system. Results of the optimization procedure are

Table 7.3: Values for the parameters.

$k_p^{\min}$	90
$k_p^{\max}$	120
$k_i^{\min}$	0.001
$k_i^{\max}$	0.05
$k_d^{\min}$	0.001
$k_d^{\max}$	0.002
$\#VM_{\max}$	4
K_e	0.1
$K_{\Delta e}$	0.05

shown in Table 7.3.

In our approach the number of VMs to be provisioned or terminated at a time k, say VM_k, is set according to the aggressiveness of the control signal. Note that the classical strategy of provisioning or terminating ± 1 VM at time (as done in [155]) can be ineffective in real scenarios since it may be too slow in scaling up in the case of sudden peak loads or it may result in unwanted longer provisioning periods during scaling down. In our approach, the number of VMs to be provided or terminated at time interval k, say VM_k, depends on the actual value of the control signal u_k according to the following dead-zone with saturation:

$$VM_k = \begin{cases} \#VM_{\max} & \text{if} \quad \bar{u} \leq u_k \\ \alpha u_k - \beta & \text{if} \quad \epsilon \leq u_k < \bar{u} \\ 0 & \text{if} \quad -\epsilon < u_k < \epsilon \\ \alpha u_k + \beta & \text{if} \quad -\underline{u} < u_k \leq -\epsilon \\ -\#VM_{\max} & \text{if} \quad u_k \leq -\underline{u} \end{cases} \tag{7.6}$$

where α and β determine how fast the controller adds and removes VMs and, as usual when dealing with nonlinear actuators, the amplitude of the dead-zone ϵ has been set so to prevent oscillations in the actuation signals [169]. Note that the actuation characteristic has been discretized with the classical sample-and-hold method.

7.2.4 Monitoring module design

In order to fulfill the needs of a generic application, the proposed approach takes advantage of heterogeneous metrics since each of them captures one of the different aspects of interest, namely the *CPU load* and the *network usage*. These are representative aspects of the state of the system: on the one hand, CPU load well captures the impact of computation on the performance as perceived by the end user [155, 170]; on the other hand, network usage

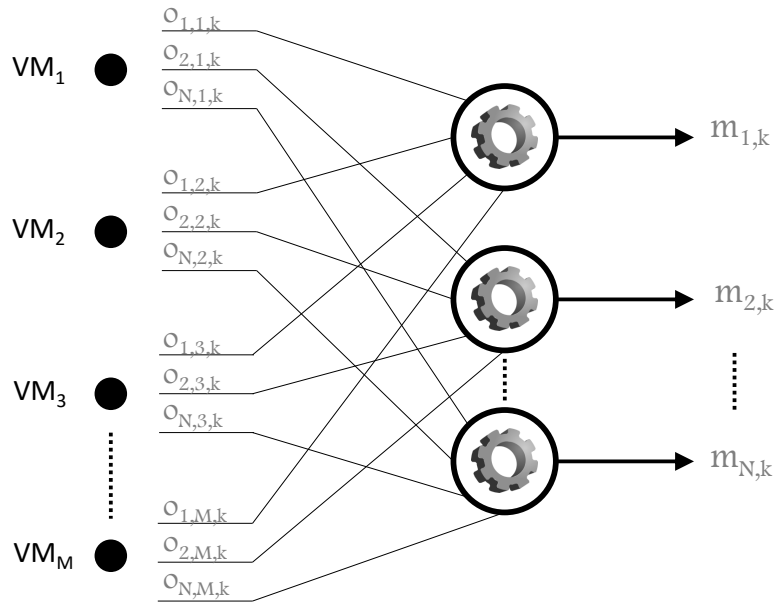


Figure 7.5: Monitoring Block.

gives hints about the state of the network interconnections among VMs in the cluster. Note that recent experimental works on public clouds, disclose the importance of this last aspect [144] to achieve complete application scalability in shared datacenters.

As shown in Figure 7.2, the Monitoring Block (M) is interfaced to multiple VMs and produces heterogeneous metrics. According to Figure 7.5, for each of the M VMs that are active at the sampling time k, the Monitoring Block extracts one sample for each of the N heterogeneous metrics, say $o_{i,j,k}$ where $i = 1, \ldots, N$, $j = 1, \ldots, M$. These M samples flow into N separate processing modules, that fuse them in a single metric observation, say $m_{i,k}$. Under the assumption that the samples concurring to compute a given metric observation are equal with respect to the active VMs, we have:

$$m_{i,k} = \sum_{j}^{M} \frac{o_{i,j,k}}{M} \tag{7.7}$$

Note that the value of M may change from cycle to cycle.

Given the general structure in Figure 7.5, in this work we consider three different metrics ($i = 1, \ldots, 3$) namely the CPU load (`CPU`) for the computational capability and the amount traffic injected into or received (`NETOUT` and `NETIN`, respectively) for the

communication network. The former is expressed as a percentage, while the others as volume of traffic per cycle.

Indeed, this approach is in line with the granularity of the resources considered and with the idea that each element of the cluster equally contributes to the amount of virtual resources allocated to the application (i.e. computation and communication resources).

In public-cloud environments, different approaches can be implemented to gather observations from different VMs [31]. The different choices available reflect different levels of flexibility and scalability. From the one hand, the cloud customer can adopt his/her own monitoring module (i.e. can adopt non-cooperative approaches). This approach can be applied to any public cloud, and completely relies on the customer, who is in charge of designing, developing, and deploying monitoring probes able to gather observations on each active VM. In this case, the features of the monitoring module are necessarily limited by the point of view a cloud customer can leverage. The main advantage of this approach consists in the higher flexibility (i.e. the customer can decide the metrics to monitor, and the granularity of the measure based on the application needs). On the other hand, the customer can also rely on monitoring modules directly supplied by the provider, when available. In this case the customer, being relieved of the design, implementation and deployment burden, is restricted in some implementation choices (such as the available metrics or the granularity of the measurements).

7.2.5 Fitness Function design

Based on the metric observations, the next step is to construct an index that captures the overall state of the cloud system at each sampling time k. To this aim here we design a fitness function block. The definition of a QoS index from heterogeneous metrics is a well known problem in the recent literature [171, 172, 173, 174, 175]. Here, with the idea in mind of evaluating how close the current state of the cloud system is with respect to an ideal reference behavior, we adopt the general approach proposed in [171] and we propose the following fitness function:

$$\mathcal{F}_k = \sum_{i}^{N} \alpha_i \frac{m_{i,k}}{R_i} \tag{7.8}$$

where $m_{i,k}$ and R_i are the metric observation at the sampling time k and the reference value, respectively; $\alpha_i \in [0,\ 1]$ are positive weights so that:

$$\sum_{i}^{N} \alpha_i = 1 \tag{7.9}$$

By construction, when the system is working at the desired value for each metric (i.e. when $m_i = R_i, \forall i$) we obtain $\mathcal{F}_k^* = 1$. Otherwise, the output of the function reaches values larger than 1 when each measure m_j is larger than the respective reference R_j, or values smaller than 1 when each measure m_j is smaller than the respective reference R_j. Given the overall constraints in eq. 7.9 the value to be assigned to each α_j is crucial for the evaluation process, and usually the priority to be given to each single metric that contributes to the QoS index depends on the specific application, as shown in the very recent literature on key performance indices [171, 174, 172, 173].

7.3 Experimental evaluation

In this section we first describe the experimental setup, then we present the results achieved thanks to the proposed FLC approach.

7.3.1 Experimental setup

In order to evaluate our proposal in a public-cloud environment we set up a testbed composed of three main elements: (i) the *cloud application*, deployed and running on a public cloud infrastructure; (ii) the *master node*, hosting all the blocks of the architecture proposed (see Figure 7.2) and managing cloud resources through their interaction; (iii) the *final-user emulation node*, in charge of issuing requests to the cloud application and imposing the workload to the system. In the following we present all the implementation details for each of the elements above.

Cloud application. Our solution has been extensively tested on a *Amazon EC2* IaaS environment. Although the approach we propose does not depend on the specific provider it represents a valid test bench for the proposed solution being claimed to be one of the leading providers [176].

The cloud application is deployed onto a group of VMs (i.e., the cluster) that can dynamically change its size at each sampling time k, as demanded by the master node. The

cluster is composed of a set of same-type VMs, connected to a load-balancer instructed to equally distribute the incoming requests across them. On each VM, a web server has been configured, in order to serve the HTTP requests generated by final users and forwarded by the load balancer. When a VM receives a request from the load balancer, it acts as *root node* for the request and gives birth to a number of CPU-consuming and network-intensive tasks, by distributing the burden of the request among all the VMs part of the cluster. Each VM communicates with other peers in the cluster: based on its cardinality, a different amount of traffic is generated towards and from each VM. Each VM at the end of the process, returns the control to the root node, that generates the reply directed to the final user.

For all our experimentations, we adopted general purpose micro VMs (`t2.micro`) representing a feasible choice for our long experimental sessions thanks to their limited cost. The selection of this particular kind of VMs does not represent a limitation, since the proposed control approach does not depend on the type of the VMs employed and no assumption has been made on provisioning dynamics. In order to distribute the incoming HTTP requests to the VMs of the cluster in a balanced fashion, we properly configured the *AWS Elastic Load Balancing* service made available by the provider itself. Finally, the network-intensive tasks are executed by synthetically generating real network traffic among the VMs of the cluster through the adoption of D-ITG traffic generator [177].

Master node. For what concerns the master node, two blocks have to be properly configured and tuned.

Monitoring Block. For the experimental evaluation, the samples to construct the metric observations have been gathered through the monitoring solution supplied by the cloud provider itself i.e. *Amazon CloudWatch* [65], since the metrics of interest for the proposed approach can be obtained through it. We also developed a provider-independent solution, in order to foster the replication of the proposed approach to other public cloud providers.

Fitness Function Block. With respect to the weights of the fitness function (see eq. 7.8) where not explicitly stated otherwise, we balance the computational capability and the network communication aspects, hence we assigned an overall weight of 0.5 to both the `CPU` and network-related metrics `NETIN` and `NETOUT`, (subdivided as 0.25 each). This is an exemplar priority choice, made for a generic application.

Final user emulation. To produce web requests that impose a workload on the sys-

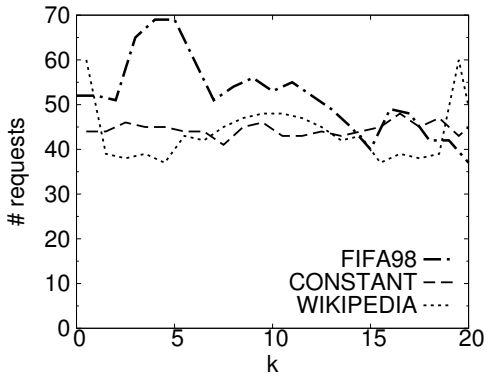


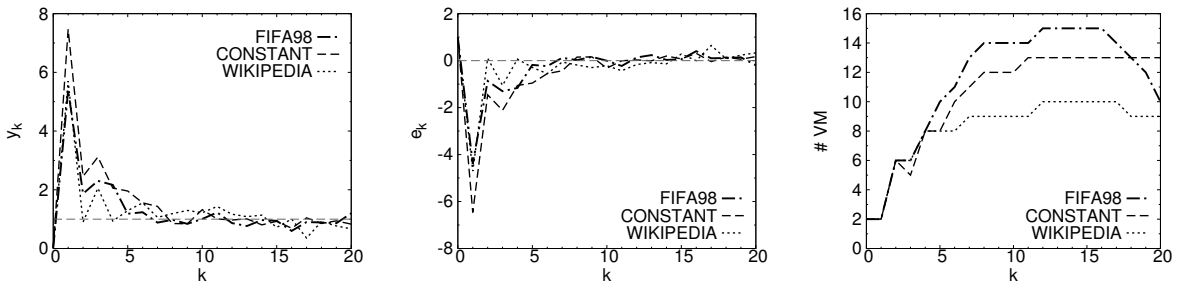
Figure 7.6: Different workloads adopted to emulate users' requests.

tem, a geographically separated node was configured. The realistic workload has been generated by exploiting *Httpmon*, an HTTP request generator purposely designed for executing experiments related to computing capacity shortage avoidance in cloud computing [178] [179]. The tool allows to emulate web users by generating request patterns in which the time between two consecutive requests is exponentially distributed. We instructed Httpmon to use the open model, i.e. to issue requests without depending on the system's response.

To evaluate our approach, we consider the different workloads depicted in Figure 7.6, i.e. *(i)* a constant workload (CONSTANT) and two realistic benchmarks: *(ii)* the *World-Cup98 web-server workload* (FIFA98) [180]—note that it is a meaningful benchmark extensively used in the cloud scientific literature ([181, 182, 183, 184, 185, 186, 187]), and *(iii)* the *WikiBench workload* (WIKIPEDIA)—related to the application used to host `wikipedia.org` which allows one to stress-test systems designed to host web applications and cloud platforms. For each experiment, each workload has a duration of 120 minutes (i.e., 20 $\times$ 6-minute cycles). The overall experimental activity (control tuning and extensive validation) amounted to 100 hours.

7.3.2 Experimental results

Here the effectiveness of the proposed architecture is investigated. Through representative experimental examples (i) we evaluate the proposed FLC strategy against three different workloads (i.e. CONSTANT, FIFA98, and WIKIPEDIA workloads) also considering the impact on each metric individually; (ii) we disclose the stability and robustness of the approach with respect to different choices of the weights α_i that balance the differ-



(a) (b) (c)

Figure 7.7: Performance in the presence of the three different workloads in Figure 7.6 (start-up transient and regime). (a): time history of the current SL measured as $y_k = \mathcal{F}(\cdot)$; (b): time history of the error with the respect to the SLO $e_k = (y_d - y_k)$; (c): time history of the active VMs ($\#VM(k)$).

ent variables within the fitness function; (iii) we compare the proposed solution against previously proposed approaches; (iv) we evaluate the robustness in the presence of VM failures.

Performance in the presence of different workloads. Here we evaluate the ability of the proposed approach to regulate the current SL measured as $y_k = \mathcal{F}(\cdot)$ at its predefined desired SLO i.e. $y_d = \mathcal{F}^* = 1$, in the presence of the three different workloads introduced above. This analysis is fundamental since one of the practical features required to the resource management strategies is the automatic compensation of time-varying operative conditions without the need of on-line prediction of workload effects.

Regulation results are depicted in Figs. 7.7a and 7.7b. The controller always ensures a stable behavior also in the presence of highly time-varying operating conditions, with a steady-state percentage error of always less the 20%. Furthermore, the desired target is always reached in less of 9 time intervals from the start-up time (interval duration is 5 minutes).

The control signal (i.e. the number of active VMs) is depicted in Figure 7.7c and varies in order to counteract the effects of the workload. Note that overshooting is evident only in the very first cycles at the system start-up, when the architecture allocates a prefixed minimum number of VMs equal to two. This initial condition reflects the communication needs of the application, that at least relies on two VMs to run. Better switching-on performance can be easily achieved relaxing the management policies and the increasing the number of VMs active at start-up. Note that, because of the pay-as-you-go model,

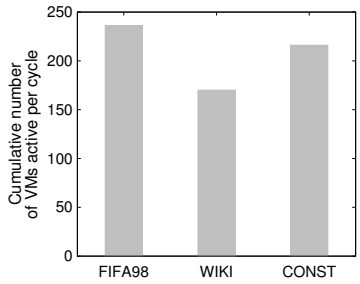


Figure 7.8: Control cost associated to each workload, calculated as the cumulative number of VMs active per cycle. The overall control cost depends on the workload. Best performance in terms of cost is achieved with the WIKIPEDIA workload.

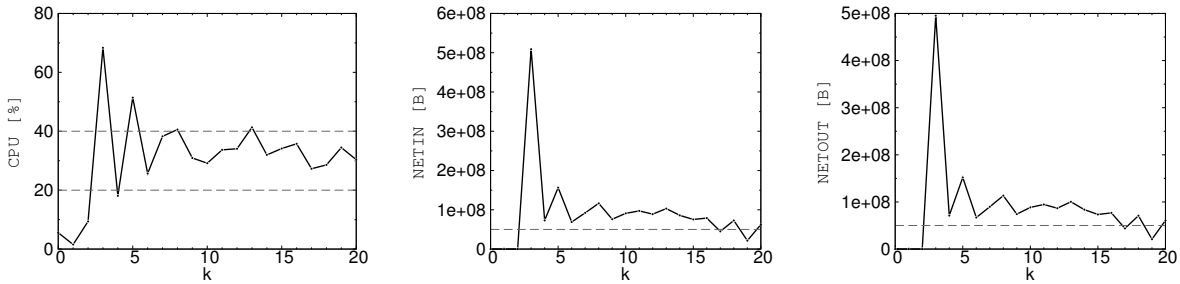


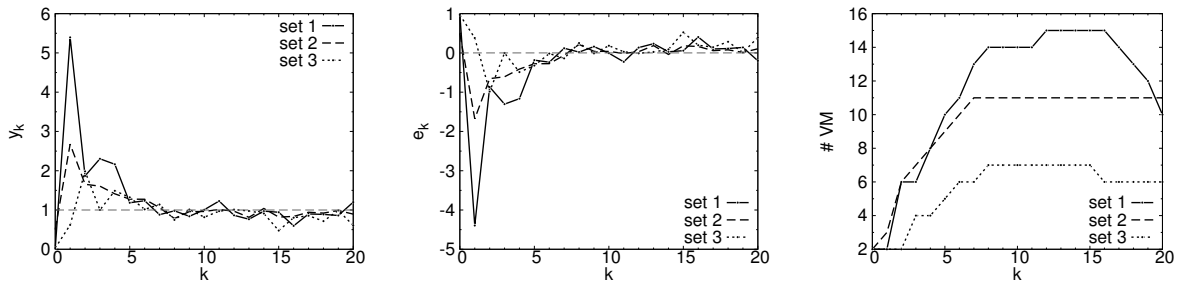
Figure 7.9: Performance with respect to each metric considered individually (WIKIPEDIA workload). (a): time history of `CPU`; (b): time history of `NETIN`; (c): time history of `NETOUT`.

the number of VMs to be allocated at the time 0, is the solution of a trade-off between maximum acceptable error and cost to be paid, which is beyond the scope of this chapter.

It is worth noting that the cost associated to the control action (evaluable as the area underlining the control signal e_k) depends on the workload and its variability (see Figure 7.8). The best performance in terms of cost is achieved with the WIKIPEDIA workload, as it is associated with the lowest cumulative number of requests, although it is characterized by a higher variability with respect to CONSTANT workload.

Performance with respect to each metric individually. Although the aim of the proposed approach is to guarantee a prefixed SL evaluated by a sole index computed merging heterogeneous metrics, in our experimental analysis we also evaluate its ability in appropriately regulating each of the metrics of interest individually. Results in Figure 7.13 refer as a representative example to the WIKIPEDIA workload case.

Specifically, the FLC approach guarantees that `CPU` converges to the target CPU (30%), with a negligible error as depicted in Figure 7.9a. A similar behavior is obtained



(a) (b) (c)

Figure 7.10: Performance with respect to fitness function weights (FIFA98 workload). (a): time history of the current SL measured as $y_k = \mathcal{F}(\cdot)$; (b): time history of the error with the respect to the SLO $e_k = (yd - yk)$; (c): time history of the active VMs ($\#VM(k)$.

for both `NETIN` and `NETOUT`, as reported in Figure 7.9b and Figure 7.9c (where 50 MB is the target values chosen for both).

Sensitivity analysis with respect to fitness function weights. To further analyze the flexibility of the proposed approach we also performed a sensitivity analysis with respect to fitness function weights. As exemplar cases, here we report results for two alternative weight sets (see Table 7.4) for the FIFA98 workload, which is the one with highest variability and average request level with respect to the ones taken into account. Similar performance can be achieved in all the other tested cases (for different workload and weight sets), omitted here for sake of brevity.

Results depicted in Figure 7.10a and Figure 7.10b show that the regulation goal ($y_d = 1$) is always achieved in any case again in less that 9 cycles with an average percentage error that never exceeds the 25%. Conversely, the history of the VM activation is different and, as expected, depends on the specific choice made for the sets (see Figure 7.10c). Note that in our case the control signal (i.e. the number of active VMs) is greater when we balance the CPU and network capability aspects. Less control effort is instead necessary for the application considered when a grater priority is given to the CPU load. As shown

Table 7.4: Different choices of the fitness function weights considered. While set 1 and 2 balance computational and network aspects, set 3 only consider `CPU`.

Set	α_{CPU}	α_{NETIN}	α_{NETOUT}
1	0.5	0.25	0.25
2	0.7	0.15	0.15
3	1	0	0

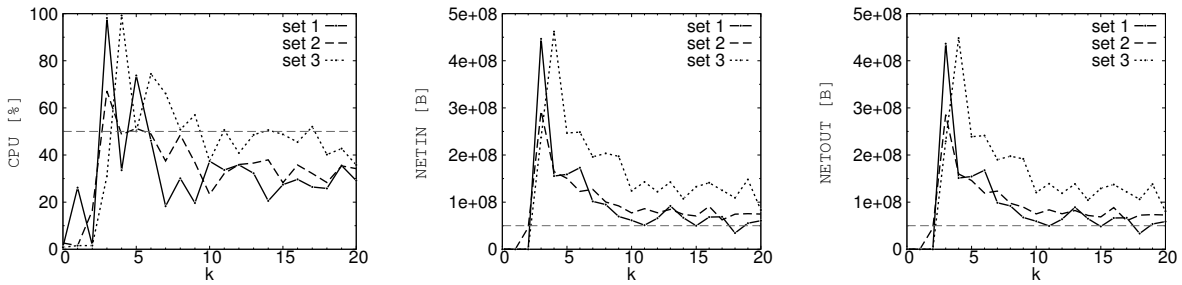
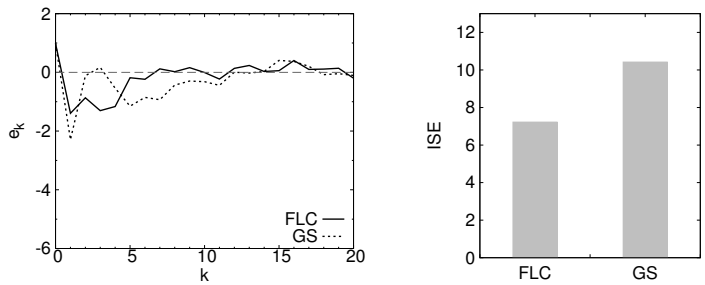


Figure 7.11: Performance with respect to fitness function parameters as in Table 7.4 (FIFA98 workload). (a): `CPU` time history; (b): `NETIN` time history; (c): `NETOUT` time history.

in the figure, the lower values of active VMs at each cycle is obtained for set 3. Hence results suggest that the metrics considered and their priority have an impact on the control effort and thus on the cost to be paid to guarantee performance, depending on the needs of the application. Further studies are needed to investigate the trade-off between costs and performance raising from the choice of metrics and weights, and its optimization.

Although the overall performance does not depend on the specific choice of the weights set, the regulation of each single metric to its desired value is impacted by the choice of priorities. Results in Figure 7.11 disclose this specific aspect, and here better goal achievement is obtained for the set 3 where only the CPU is weighted. Note that to highlight the flexibility of the approach with respect to the specific choice of the reference value, the CPU requirement for this experiment has been set to 50%. The similar results that can be obtained by arbitrary changing the reference value for each metric, are omitted here.

Comparison against Gain Scheduling strategy. In this section we evaluate the proposed approach against a past proposal in public clouds [188], where a Gain Scheduling algorithm experimentally outperforms other state-of-the-art approaches. It is worth noting here that the architecture presented in this chapter solves a more wide problem introducing heterogeneous metrics, while previous attempts in the state of the art (as already detailed in § 7.1.1) usually only considered CPU-related aspects. So, in order to perform a fair comparison, we choose to downgrade our architecture by selecting the trivial weight set (i.e., the set 3 in Table 7.4). For what concerns the workload, we select again the FIFA98 workload (see Figure 7.6) for the high variability and the amount of



(a) (b)

Figure 7.12: FLC approach vs. GS approach (FIFA98 workload). (a) Time history of the error ($e_k = (y_d - y_k)$) between the actual SL (measured in terms of `CPU`) and its SLO. (b) Integral of Squared Errors—ISE.

the requests.

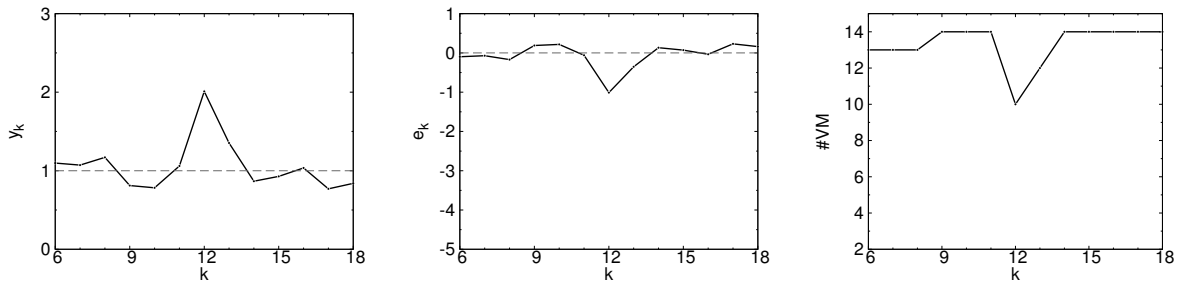
Results in Figure 7.12a show that the fuzzy adaptation of the control gains with respect to time-varying conditions of the public-cloud system, provides better performance with respect to a classical gain scheduling PID algorithm, due to its ability to cope with highly time-varying and vague environments. In order to better evaluate how the two approaches meet strict SLOs (the requirement is for the `CPU` to be $y_d = 30\%$) here we exploit the Integral of Squared Errors (ISE) [169] that better summarize the achieved results. Note that, the FLC approach guarantees a lower ISE than GS (see Figure 7.12b).

Robustness in the presence of VM failures. To analyze the robustness with respect to failures, we consider the FLC under the action of the CONSTANT workload (see Figure 7.6) to better capture the effects of sudden and unwanted VM termination.

The system is at its steady-state equilibrium point $y_k = y_d = 1$ when we cause a hard failure to happen: specifically more thn 1/3 of VMs that are running crash at time interval $k = 11$ (4 out of the 14 active VMs, see Figure 7.13c).

Due to this critical event, the regulation error increases (see Figure 7.13b) and, accordingly, the control action varies, on-line adapts its gains, and counteracts the effect of the failures (see Figure 7.13a). In so doing, the error is then again within $\pm 10\%$ at time interval $k = 14$, as shown in Fig.7.13b. **Concluding remarks.** In this chapter we have

proposed a feedback-based control approach to automatically scale out public-cloud resources only leveraging the base of knowledge available to the customer. The approach



(a) (b) (c)

Figure 7.13: Robustness in the presence of VM failures. (a): time history of the current SL measured as $y_k = \mathcal{F}(\cdot)$; (b): time history of the error with the respect to the SLO $e_k = (y_d - y_k)$; (c): time history of the active VMs ($\#VM(k)$).

takes as input heterogeneous monitoring metrics merged through a fitness function and leverages fuzzy logic to cope with cloud unpredictable and highly time-varying operating conditions. The results of the experimentation performed within Amazon public cloud environment show that the proposed approach is robust against different realistic workloads, also in presence of VM failures. Moreover by acting on the weights of the fitness function, the approach proved to be flexible, thus proving to be suitable for applications with different needs. Compared to already-proposed solutions, the proposed approach is able to provide better performance with respect to classical PID algorithms.

Chapter 8

Conclusion

Cloud computing more and more positively impacts ICT industry, as providing both technical and economical benefits. Accordingly, with the strengthening of the security aspects, an increasing number of organizations is moving its mission-critical applications and services on public clouds. As a results, cloud traffic is expected to dramatically increase in the next few years, reaching the three quarters of the total datacenter traffic by 2018. In order to properly manage this highly dynamic and heavy load, huge investments have been made by providers and networks of geographically distributed datacenters have been built. Cloud infrastructures have become more and more complex in the last years indeed, from both the qualitative and the quantitative point of view. To operate cloud resources the proper understanding of the state of these infrastructures is of the paramount importance. Therefore effective, efficient, and accurate monitoring activities are required.

Network resources are particularly critical, as their characteristics and performance may impact the performance of the overall system. Because of the nature of the cloud paradigm, the network is the only means to access cloud resources. Moreover, a number of applications heavily rely on the cloud network, as the high degree of virtualization naturally leads to implement functional separation of layers, with heavy communication needs. Also, applications and services are often designed to benefit from the high-performance network architecture that interconnects geographically distributed cloud datacenters, and therefore cloud applications increasingly exchange traffic among datacenters placed in two different sites on the globe.

However, providers expose very poor information about the cloud networks, in spite of their primary role. Although public-cloud datacenters intrinsically depend on high-performance networks to connect servers within the datacenter and to the rest of the

world, providers seldom make any promises about network performance, even though many cloud customers do want to be able to rely on network performance guarantees. Consequently, public-cloud consumers suffer from unpredictable network performance, which in turn generates both performance and cost issues for the application.

In this thesis we have investigated the techniques for monitoring the performance of public-cloud networks through non-cooperative approaches, i.e., without relying on privileged points of views, but acting as general customers. We have also shown how network monitoring is beneficial not only to the entities directly involved in the management of the cloud infrastructure, but also to the customers.

Our thesis includes several contributions to the field of cloud network performance evaluation.

We have designed and implemented *CloudSurf* a monitoring platform that enables the general cloud consumer to monitor public-cloud networking infrastructures from the customer viewpoint, and without relying on any information restricted to entities playing privileged roles in provisioning cloud services. The platform implements a number of desirable features (e.g., ease of use, experiment-cost estimation, and ability to share results with the community), and integrates a number of active monitoring tools. Moreover it allows to act on a number of factors under the direct control of the cloud consumer whose effect may influence the performance of the network. By varying these factors, it is possible to identify a number of different scenarios, thus leading to a better characterization of the datacenter network infrastructures. This drives to offer a clearer picture of the intra-cloud networks with respect to the blurred one coming from the scientific literature.

Experimental analyses have been performed with respect to the two leading cloud providers, i.e. Amazon Web Service and Microsoft Azure. Implementing non-cooperative approaches to monitor public-cloud intra-datacenter network performance, we have been able to deepen the understanding of cloud systems. First, we have characterized the impact that virtualization has on non-cooperative measurement techniques, also providing recommendations to properly deal with it. We found that providers—although offering comparable network performances—implement different management mechanisms which may differently impact performance variability. While stable and predictable performance has been observed for Amazon VMs, a higher variability has been found for the Azure network infrastructure. Further analyses show how this variability may be due to some factors under the control of the customer. Thanks to the gathered knowledge, some de-

ployment guidelines have been also drawn to cope with undesired performance variability. These guidelines allow the customer to improve the expected network performance.

Furthermore, the thesis depicts a clear picture of the cloud inter-datacenter network, also providing some insights into the communication infrastructures leveraged by cloud providers. Our study shows the existence of phenomena generated by the management strategies enforced, which may impact both the performance experienced by the customers and the results of analyses investigating these networks. We found that, thanks the investments done and the infrastructures deployed, inter-datacenter and intra-datacenter networks in some cases exposes comparable performance, even though the former have a larger geographical extension. In addition, Azure inter-datacenter network performs better than Amazon's potentially because of the smaller number of customers of the latter. Comparing performance results to provider-imposed fees, we found that their relation could be counterintuitive, as worse performance is associated to higher costs for the customer. Path analysis reveals that this is justified by the need for the provider to be served from external network carriers.

The thesis also analyzes the cloud-to-user network, taking into account the performance of Amazon S3 as being an interesting case study. We found that datacenter placed in specific regions (i.e., EU or US) are able to deliver better performance, on average. Network performance improves when utilizing CDN services, although examples of performance worsening have been observed, probably due to provider's suboptimal management.

Finally, also aiming at showing how monitoring data can be helpful also for the customers, our study also focuses on its adoption to implement strategies to automatically scale resources in public clouds. A novel control strategy approach has been designed, implemented, and extensively tested. The proposed strategy merges heterogeneous metrics to control resource allocation, thus enabling to horizontally scale the set of VMs allocated to an application. Leveraging metrics related to different aspects (i.e. computation and network), the proposed approach is able to take into account the current state of the managed application from different angles at once, thus meeting its different needs. The proposed approach does not require any detailed knowledge of the dynamics of the controlled cloud infrastructure or a performance model of the application. Therefore it can be applied to any public cloud environment.

Thanks to the proposed methodologies and to the realized platform we enabled the

scientific community to better address the study of performance in the context cloud networks.

Acknowledgments

I would like to express my special appreciation and thanks to my advisor Prof. Pescapé for having accepted me as a student, and for the support along the years of the doctoral course. Special thanks go to him and to all the other guys of the *Traffic* research group, Alessio, Giuseppe, Walter, to the newly arrived in the group, Antonio, and to the "free-time-researcher-at-least-for-now" Pietro who have contributed immensely to my personal and professional growth. Each with his own peculiarities and his point of view about life and research. Together they have been a source of friendship as well as examples and good advices. Lastly, I would like to thank my family for all their love and support. Thanks to my parents and my "little" brother Daniele for having supported me in all my pursuits. A huge thank you to Marianna, for her love, her patience, and above all, for having taught me to distinguish what is important from what is really important. I'm still on the way to learn about it, actually. But I am able to still establish contact with Planet Earth thanks to her priceless help, at least.

Bibliography

[1] Douglas F Parkhill. Challenge of the computer utility. 1966.

[2] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[3] Peter Mell and Tim Grance. The NIST definition of cloud computing. Special Publication 800-145. 2011.

[4] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. *Decision Support Systems*, 51(1):176 – 189, 2011.

[5] Federico Etro. The economic impact of cloud computing on business creation, employment and output in europe. *Review of Business and Economics*, 54(2):179–208, 2009.

[6] Ludwig Siegele. *Let it rise: A special report on corporate IT*. Economist Newspaper, 2008.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[8] Andrzej Bialecki, Michael Cafarella, Doug Cutting, and Owen O'MALLEY. Hadoop: a framework for running applications on large clusters built of commodity hardware. *Wiki at http://lucene. apache. org/hadoop*, 11, 2005.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[10] Apache hadoop website. http://hadoop.apache.org.

[11] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. NIST cloud computing standards roadmap. Special Publication 500-291. 2011.

[12] The Xen Project. http://www.xenproject.org/.

[13] Kernel Virtual Machine. http://www.linux-kvm.org/.

[14] wmware. http://www.vmware.com/.

[15] CISCO. Cisco Global Cloud Index: Forecast and Methodology 2014–2019. 2013.

[16] IDC Press Release. Worldwide public cloud services spending forecast to double by 2019, according to idc. https://www.idc.com/getdoc.jsp?containerId=prUS40960516.

[17] Lydia Leong, Douglas Toombs, and Bob Gill. Magic quadrant for cloud infrastructure as a service, worldwide. http://www.gartner.com/technology/reprints.do?id=1-2G2O5FC&ct=150519&st=sb, 2015.

[18] press release Synergy research. Aws remains dominant despite microsoft and google growth surges. https://www.srgresearch.com/articles/aws-remains-dominant-despite-microsoft-and-google-growth-surges, 2016.

[19] Amazon Web Services website. http://aws.amazon.com, March 2016.

[20] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The computer journal*, 53(7):1045–1051, 2010.

[21] Andy Hooper. Green computing. *Communication of the ACM*, 51(10):11–13, 2008.

[22] J. Baliga, R.W.A. Ayre, K. Hinton, and RodneyS. Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167, Jan 2011.

[23] Green Micro-electronic Packaging, Energy-Efficient Interconnect, Sustainable Living, and Green Computing. Technology with the environment in mind. *Intel Technology Journal*, 12(1), 2008.

[24] R.R. Harmon and N. Auseklis. Sustainable it services: Assessing the impact of green computing practices. In *Management of Engineering Technology, 2009. PICMET 2009. Portland International Conference on*, pages 1707–1717, Aug 2009.

[25] M Armbrust, M Armbrust, A Fox, A Fox, R Griffith, R Griffith, Ad Joseph, Ad Joseph, Rh, and Rh. Above the clouds: A Berkeley view of cloud computing. *University of California, Berkeley, Tech. Rep. UCB*, pages 07–013, 2009.

[26] Guohui Wang and T. S Eugene Ng. The impact of virtualization on network performance of Amazon EC2 Data Center. *Proceedings - IEEE INFOCOM*, 2010.

[27] Costin Raiciu, Mihail Ionescu, and Dragos Niculescu. Opening up black box networks with CloudTalk. In *Proc. 4th USENIX Conference on Hot Topics in Cloud Computing*, page 6, 2012.

[28] K LaCurts and Shuo Deng. Choreo: network-aware task placement for cloud applications. *Proceedings of the 2013 . . .*, pages 191–203, 2013.

[29] E.M. Hanna, N. Mohamed, and J. Al-Jaroodi. The cloud: Requirements for a better service. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 787–792, May 2012.

[30] Gurdev Singh, Shanu Sood, and Amit Sharma. Cm-measurement facets for cloud performance. *International Journal of Computer Applications*, 23(3):37–42, 2011.

[31] Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio PescapÃ¨. Cloud monitoring: A survey. *Computer Networks*, 57(9):2093 – 2115, 2013.

[32] Eddy Caron, Luis Rodero-Merino, Frédéric Desprez, and Adrian Muresan. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. 2012.

[33] A. Viratanapanu, A.K.A. Hamid, Y. Kawahara, and T. Asami. On demand fine grain resource monitoring system for server consolidation. In *Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services, 2010 ITU-T*, pages 1–8, Dec 2010.

[34] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. CloudCmp: Comparing Public Cloud Providers. *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, page 1, 2010.

[35] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. VL2: A Scalable and Flexible Data Center Network. *SIGCOMM Comput. Commun. Rev.*, 39(4):51–62, 2009.

[36] Jeffrey C Mogul and Lucian Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.

[37] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, 3(1-2):460–471, 2010.

[38] Jose Garcia-Dorado and Sanjay Rao. Cost-aware Multi Data-Center Bulk Transfers in the Cloud from a Customer-Side Perspective. *IEEE Transactions on Cloud Computing*, 7161(c):1–1, 2015.

[39] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. *ACM SIGCOMM Computer Communication Review*, 41(4):74, 2011.

[40] Ondrej Tomanek and Lukas Kencl. CLAudit: Planetary-scale cloud latency auditing platform. *Proceedings of the 2013 IEEE 2nd International Conference on Cloud Networking, CloudNet 2013*, pages 138–146, 2013.

[41] Zi Hu, Liang Zhu, Calvin Ardi, Ethan Katz-Bassett, Harsha V. Madhyastha, John Heidemann, and Minlan Yu. The need for end-to-end evaluation of cloud availability. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8362 LNCS:119–130, 2014.

[42] N. Bitar, S. Gringeri, and T.J. Xia. Technologies and protocols for data center and cloud networking. *Communications Magazine, IEEE*, 51(9):24–31, September 2013.

[43] S. Azodolmolky, P. Wieder, and R. Yahyapour. Sdn-based cloud computing networking. In *Transparent Optical Networks (ICTON), 2013 15th International Conference on*, pages 1–4, June 2013.

[44] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[45] Pedro Casas and Raimund Schatz. Quality of experience in cloud services: Survey and measurements. *Computer Networks*, 68:149 – 165, 2014. Communications and Networking in the Cloud.

[46] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-li Zhang. Unreeling Netflix : Understanding and Improving Multi-CDN Movie Delivery. *Ieee Infocom*, pages 1620–1628, 2012.

[47] Feng Wang, Jiangchuan Liu, and Minghua Chen. Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities. In *INFOCOM, 2012 Proceedings IEEE*, pages 199–207. IEEE, 2012.

[48] Yuan Feng, Baochun Li, and Bo Li. Airlift: Video conferencing as a cloud service using inter-datacenter networks. In *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pages 1–11. IEEE, 2012.

[49] Greg Linden. Make data useful, 2006.

[50] Ron Kohavi, Randal M Henne, and Dan Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967. ACM, 2007.

[51] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards Predictable Datacenter Networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, pages 242–253, New York, NY, USA, 2011. ACM.

[52] Matthew L Massie, Brent N Chun, and David E Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[53] Nagios. https://www.nagios.org/.

[54] Zabbix. https://www.zabbix.com/.

[55] Harvey B Newman, Iosif C Legrand, Philippe Galvez, Ramiro Voicu, and Catalin Cirstoiu. Monalisa: A distributed monitoring service architecture. *arXiv preprint cs/0306096*, 2003.

[56] Sergio Andreozzi, Natascia De Bortoli, Sergio Fantinel, Antonia Ghiselli, Gian Luca Rubini, Gennaro Tortone, and Maria Cristina Vistoli. Gridice: a monitoring service for grid systems. *Future Generation Computer Systems*, 21(4):559–571, 2005.

[57] Peer Hasselmeyer and Nico d'Heureuse. Towards holistic multi-tenant monitoring for virtual data centers. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 350–356. IEEE, 2010.

[58] Gregory Katsaros, Roland Kubert, and Georgina Gallizo. Building a service-oriented monitoring framework with rest and nagios. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 426–431. IEEE, 2011.

[59] Chengwei Wang, Karsten Schwan, Vanish Talwar, Greg Eisenhauer, Liting Hu, and Matthew Wolf. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 141–150. ACM, 2011.

[60] Fatemeh Azmandian, Micha Moffie, Jennifer G Dy, Javed A Aslam, and David R Kaeli. Workload characterization at the virtualization layer. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 63–72. IEEE, 2011.

[61] Stuart Clayman, Richard Clegg, Lefteris Mamatas, George Pavlou, and Alex Galis. Monitoring, aggregation and filtering for efficient management of virtual networks. In *Proceedings of the 7th International Conference on Network and Services Management*, pages 234–240. International Federation for Information Processing, 2011.

[62] Stuart Clayman, Alex Galis, and Lefteris Mamatas. Monitoring virtual networks with lattice. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 239–246. IEEE, 2010.

[63] JiSu Park, HeonChang Yu, KwangSik Chung, and EunYoung Lee. Markov chain based monitoring service for fault tolerance in mobile cloud computing. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 520–525. IEEE, 2011.

[64] Mahendra Kutare, Greg Eisenhauer, Chengwei Wang, Karsten Schwan, Vanish Talwar, and Matthew Wolf. Monalytics: online monitoring and analytics for managing large scale data centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 141–150. ACM, 2010.

[65] Amazon cloudwatch monitoring service. http://aws.amazon.com/cloudwatch/. Online; accessed Nov '15.

[66] https://www.paraleap.com/AzureWatch.

[67] http://cloudmonix.com/.

[68] http://portal.monitis.com.

[69] Microsoft Azure website. http://azure.microsoft.com, March 2016.

[70] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208. ACM, 2009.

[71] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

[72] Rahul Potharaju and N Jain. An empirical analysis of intra-and inter-datacenter network failures for geo-distributed services. *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 335–336, 2013.

[73] Yingying Chen, S. Jain, V.K. Adhikari, Zhi-Li Zhang, and Kuai Xu. A first look at inter-data center traffic characteristics via yahoo! datasets. In *INFOCOM, 2011 Proceedings IEEE*, pages 1620–1628, April 2011.

[74] Ignacio Bermudez, Stefano Traverso, Marco Mellia, and Maurizio Munafó. Large scale observation and analysis of Amazon AWS traffic. (October), 2012.

[75] Gideon Juve, Ewa Deelman, G Bruce Berriman, Benjamin P Berman, and Philip Maechling. An evaluation of the cost and performance of scientific workflows on amazon ec2. *Journal of Grid Computing*, 10(1):5–21, 2012.

[76] Huan Liu and Sewook Wee. Web server farm in the cloud: Performance evaluation and dynamic architecture. In *Cloud Computing*, pages 369–380. Springer, 2009.

[77] Radu Tudoran, Alexandru Costan, Gabriel Antoniu, and Luc Bougé. A performance evaluation of azure and nimbus clouds for scientific applications. In *Proceedings of the 2Nd International Workshop on Cloud Computing Platforms*, CloudCP '12, pages 4:1–4:6, New York, NY, USA, 2012. ACM.

[78] Zach Hill, Jie Li, Ming Mao, Arkaitz Ruiz-Alvarez, and Marty Humphrey. Early observations on the performance of windows azure. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 367–376. ACM, 2010.

[79] Mayur Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. Amazon S3 for Science Grids: a Viable Solution. *Proc. ACM Int. Workshop on Data-aware Distributed Computing*, pages 55–64, 2008.

[80] Pavol Mulinka and Lukas Kencl. Learning from cloud latency measurements. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1895–1901. IEEE, 2015.

[81] Hamed Saljooghinejad, Felix Cuadrado, and Steve Uhlig. Let latency guide you: Towards characterization of cloud application performance. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 99–106. IEEE, 2015.

[82] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, pages 363–378. USENIX, 2016.

[83] Yuan Feng, Baochun Li, and Bo Li. Jetway: minimizing costs on inter-datacenter video traffic. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 259–268. ACM, 2012.

[84] Nimbus Project. http://www.nimbusproject.org/, March 2016.

[85] Zach Hill, Jie Li, Ming Mao, Arkaitz Ruiz-Alvarez, and Marty Humphrey. Early observations on the performance of Windows Azure. *Scientific Programming*, 19(2-3):121–132, 2011.

[86] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. On the performance variability of production cloud services. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 104–113. IEEE, 2011.

[87] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[88] Keqiang He, Alexis Fisher, Liang Wang, Aaron Gember, Aditya Akella, and Thomas Ristenpart. Next stop, the cloud: Understanding modern web service deployment in ec2 and azure. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 177–190. ACM, 2013.

[89] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM, 2009.

[90] Sk Barker and Prashant Shenoy. Empirical evaluation of latency-sensitive application performance in the cloud. *Proceedings of the first annual ACM SIGMM ...*, page 35, 2010.

[91] The CloudSurf platform webpage. http://traffic.comics.unina.it/cloudsurf.

[92] GNU Affero General Public License. http://www.gnu.org/licenses/agpl-3.0.html, 2007.

[93] Awspricingfull project. https://github.com/ilia-semenov/awspricingfull.

[94] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996. Updated by RFC 6761.

[95] Dave Mangot. Measuring ec2 system performance. http://tech.mangot.com/roller/dave/entry/ec2_variability_the_numbers_revealed, May 2009.

[96] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster Computing*, 16(2):249–264, 2011.

[97] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Cloud Computing*, pages 254–265. Springer, 2009.

[98] Terry Lam, Sivasankar Radhakrishnan, Amin Vahdat, and George Varghese. *NetShare: Virtualizing data center networks across services.* [Department of Computer Science and Engineering], University of California, San Diego, 2010.

[99] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C Snoeren. Cloud control with distributed rate limiting. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 337–348. ACM, 2007.

[100] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*, page 15. ACM, 2010.

[101] Alan Shieh, Srikanth Kandula, Albert G Greenberg, and Changhoon Kim. Seawall: Performance isolation for cloud datacenter networks. In *HotCloud*, 2010.

[102] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *WIOV*, 2011.

[103] Donald Kossmann, Tim Kraska, and Simon Loesing. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 579–590. ACM, 2010.

[104] Matei Zaharia, Andy Konwinski, Anthony D Joseph, Randy H Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI*, volume 8, page 7, 2008.

[105] Qiming He, Shujia Zhou, Ben Kobler, Dan Duffy, and Tom McGlynn. Case study for running hpc applications in public clouds. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 395–401. ACM, 2010.

[106] Public Cloud Usage Trend. https://www.cloudyn.com/papers/df-cloud-cost-benchmark-2013.pdf, 2013.

[107] Radu Tudoran, Alexandru Costan, Gabriel Antoniu, and Luc Bougé. A performance evaluation of azure and nimbus clouds for scientific applications. In *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, page 4. ACM, 2012.

[108] Jon Whiteaker, Fabian Schneider, and Renata Teixeira. Explaining packet delays under virtualization. *ACM SIGCOMM Computer Communication Review*, 41(1):38–44, 2011.

[109] Man page for `tracepath`. http://www.unix.com/man-page/linux/8/tracepath/.

[110] Google Inc. Inter-datacenter wan with centralized te using sdn and openflow. https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/cs-googlesdn.pdf, 2012.

[111] Netflix website, howpublished = http://netflix.com.

[112] CloudHarmony. https://cloudharmony.com/.

[113] B Fink and R Scott. nuttcp, v6.1.2.

[114] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring multipath routing in the Internet. *IEEE/ACM Transactions on Networking*, 19(3):830–840, June 2011.

[115] Pietro Marchetta, Valerio Persico, and Antonio Pescapé. Pythia: yet another active probing technique for alias resolution. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 229–234. ACM, 2013.

[116] Ken Keys, Young Hyun, Matthew Luckie, and Kim Claffy. Internet-scale ipv4 alias resolution with midar. *IEEE/ACM Transactions on Networking (TON)*, 21(2):383–399, 2013.

[117] Team Cymru. IP to ASN mapping service. http://www.team-cymru.org/IP-ASN-mapping.html.

[118] Amazon Web Services. AWS IP address ranges. http://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html.

[119] Gary Cook and David Pomerantz. Clicking clean: A guide to building the green internet. http://www.greenpeace.org/usa/wp-content/uploads/legacy/Global/usa/planet3/PDFs/2015ClickingClean.pdf, 2015.

[120] Amazon Simple Storage Service website. http://aws.amazon.com/s3/, March 2016.

[121] Amazon CloudFront—Content Delivery Network (CDN) website. http://aws.amazon.com/cloudfront/, March 2016.

[122] Simson L Garfinkel. Technical Report TR-08-07 : An Evaluation of Amazon's Grid Computing Services: EC2 , S3 and SQS. *Applied Sciences*, pages 1–15, 2006.

[123] Ignacio Bermudez, Stefano Traverso, Marco Mellia, and Maurizio Munafo. Exploring the cloud from passive measurements: The Amazon AWS case. *2013 Proceedings IEEE INFOCOM*, pages 230–234, 2013.

[124] Andy C Bavier, Mic Bowman, Brent N Chun, David E Culler, Scott Karlin, Steve Muir, Larry L Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating systems support for planetary-scale network services. In *NSDI*, volume 4, pages 19–19, 2004.

[125] Kash Rangan, A Cooke, J Post, and N Schindler. The cloud wars: $100+ billion at stake. Technical report, Tech. rep., Merrill Lynch, 2008.

[126] Andre Abrantes DP Souza and Marco AS Netto. Using application data for sla-aware auto-scaling in cloud environments. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on*, pages 252–255. IEEE, 2015.

[127] Mahyar Movahed Nejad, Lena Mashayekhy, and Daniel Grosu. Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):594–603, 2015.

[128] Zuling Kang, Hongbing Wang, and Lei Wang. Performance-Aware Cloud Resource Allocation via Fitness-enabled Auction. *IEEE Transactions on Parallel and Distributed Systems*, XX(X):1–1, 2015.

[129] Harold C Lim, Shivnath Babu, and Jeffrey S Chase. Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM, 2010.

[130] Martina Maggio, Cristian Klein, and Karl-Erik Årzén. Control strategies for predictable brownouts in cloud computing. IFAC, 2014.

[131] Pradeep Padala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, pages 13–26, New York, NY, USA, 2009. ACM.

[132] A. Gambi and G. Toffetti. Modeling cloud performance with kriging. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1439–1440, June 2012.

[133] Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proceedings of the 6th International Conference on Autonomic Computing*, ICAC '09, pages 117–126, New York, NY, USA, 2009. ACM.

[134] Tania Lorido-Botran, Jose Miguel-Alonso, and JoseA. Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, 2014.

[135] Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R. Lyu, and Jianmin Wang. QoS ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1213–1222, 2013.

[136] Rightscale. http://www.rightscale.com/. Online; accessed Mar '15.

[137] Eugen Feller, Louis Rilling, and Christine Morin. Snooze: A scalable and autonomic virtual machine management framework for private clouds. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 482–489, Washington, DC, USA, 2012. IEEE Computer Society.

[138] Horacio Andrés Lagar-Cavilla, Joseph Andrew Whitney, Adin Matthew Scannell, Philip Patchin, Stephen M. Rumble, Eyal de Lara, Michael Brudno, and Mahadev Satyanarayanan. Snowflock: Rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, pages 1–12, New York, NY, USA, 2009. ACM.

[139] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*, pages 89–96, Dec 2010.

[140] Hien Nguyen Van, F.D. Tran, and J.-M. Menaud. Sla-aware virtual resource management for cloud infrastructures. In *Computer and Information Technology, 2009. CIT '09. Ninth IEEE International Conference on*, volume 1, pages 357–362, Oct 2009.

[141] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo. Dynamic resource management using virtual machine migrations. *Communications Magazine, IEEE*, 50(9):34–40, September 2012.

[142] Hamzeh Khazaei, Jelena Misic, and Vojislav B Misic. A Fine-Grained Performance Model of Cloud Computing Centers. *IEEE Transactions Parallel and Distributed Systems*, 24(11):2138–2147, 2013.

[143] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling Web Applications in Heterogeneous Cloud Infrastructures. *Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E '14)*, 2014.

[144] Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45, 2011.

[145] Jieun Choi, Younsun Ahn, Seoyoung Kim, Yoonhee Kim, and Jaeyoung Choi. VM auto-scaling methods for high throughput computing on hybrid infrastructure. *Cluster Computing*, 18(3):1063–1073, 2015.

[146] Monireh Fallah, Mostafa Ghobaei Arani, and Mehrdad Maeen. NASLA: Novel Auto Scaling Approach based on Learning Automata for Web Application in Cloud Computing Environment. *International Journal of Computer Applications*, 113(2):18–23, 2015.

[147] Anton Beloglazov and Rajkumar Buyya. Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, (December 2010):6, 2011.

[148] Harold C Lim, Shivnath Babu, Jeffrey S Chase, and Sujay S Parekh. Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 13–18. ACM, 2009.

[149] Sang-Min Park and Marty Humphrey. Self-tuning virtual machines for predictable escience. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 356–363. IEEE Computer Society, 2009.

[150] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, 2013.

[151] Peter Bodık, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 12–12, 2009.

[152] Danilo Ardagna, Carlo Ghezzi, Barbara Panicucci, and Marco Trubian. Service provisioning on the cloud: Distributed algorithms for joint capacity allocation and admission control. In *Towards a Service-Based Internet*, pages 1–12. Springer, 2010.

[153] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 95–104. ACM, 2014.

[154] G.A.C. Santos, J.G.R. Maia, L.O. Moreira, F.R.C. Sousa, and J.C. Machado. Scale-space filtering for workload analysis and forecast. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 677–684, June 2013.

[155] Ian Gergin, Bradley Simmons, and Marin Litoiu. A decentralized autonomic architecture for performance control in the cloud. In *IEEE International Conference on Cloud Engineering (IC2E)*, pages 574–579. IEEE, 2014.

[156] Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres. Feedback Control Algorithms to Deploy and Scale Multiple Web Applications per Virtual Machine. *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 431–438, 2012.

[157] Karl J Astrom. Pid controllers: theory, design and tuning. *Instrument society of America*, 1995.

[158] Robert Babuška. *Fuzzy modeling for control*, volume 12. Springer Science & Business Media, 2012.

[159] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[160] Hadi Goudarzi and Massoud Pedram. Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 324–331. IEEE, 2011.

[161] Waheed Iqbal, Matthew N Dailey, and David Carrera. Sla-driven dynamic resource management for multi-tier web applications in a cloud. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 832–837. IEEE, 2010.

[162] Web application hosting in the aws cloud best practices. https://media.amazonwebservices.com/AWS_Web_Hosting_Best_Practices.pdf. Online; accessed Nov '15.

[163] Gang Feng. A survey on analysis and design of model-based fuzzy control systems. *IEEE Transactions on Fuzzy systems,*, 14(5):676–697, 2006.

[164] Han-Xiong Li and HB Gatland. A new methodology for designing a fuzzy logic controller. *IEEE Transactions on Systems, Man and Cybernetics,*, 25(3):505–512, 1995.

[165] Kwok L Tang and Robert J Mulholland. Comparing fuzzy logic with classical controller designs. *IEEE Transactions on Systems, Man and Cybernetics,*, 17(6):1085–1087, 1987.

[166] M. Sugeno. On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *Fuzzy Systems, IEEE Transactions on*, 7(2):201–224, Apr 1999.

[167] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics,*, (1):116–132, 1985.

[168] Rolf Isermann. On fuzzy logic applications for automatic control, supervision, and fault diagnosis. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,*, 28(2):221–235, 1998.

[169] Hassan K Khalil and JW Grizzle. *Nonlinear systems*, volume 3. Prentice hall New Jersey, 1996.

[170] Alex Zhang, Pano Santos, Dirk Beyer, and H Tang. Optimal server resource allocation using an open queueing network model of response time. *HP laboratories Technical Report, HPL2002301*, 2002.

[171] Yan Gong Yan Gong, Fangchun Yang Fangchun Yang, Lin Huang Lin Huang, and Sen Su Sen Su. Model-Based Approach to Measuring Quality of Experience. *2009 First International Conference on Emerging Network Intelligence*, pages 1–4, 2009.

[172] Hyun Jong Kim Hyun Jong Kim and Seong Gon Choi Seong Gon Choi. A study on a QoS/QoE correlation model for QoE evaluation on IPTV service. *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, 2, 2010.

[173] Sana Aroussi, Thouraya Bouabana-Tebibel, and Abdelhamid Mellouk. Empirical QoE/QoS correlation model based on multiple parameters for VoD flows. *GLOBECOM - IEEE Global Telecommunications Conference*, 2(1):1963–1968, 2012.

[174] Abdelwahab Hamam, Mohamad Eid, Abdulmotaleb El Saddik, and Nicolas D. Georganas. A Quality of Experience Model for Haptic User Interfaces. *Proceedings of the First International Conference on Ambient Media and Systems*, pages 1–6, 2008.

[175] Hyun Jong Kim, Dong Hyeon Lee, Jong Min Lee, Kyoung Hee Lee, Won Lyu, and Seong Gon Choi. The QoE evaluation method through the QoS-QoE correlation model. *Proceedings - 4th International Conference on Networked Computing and Advanced Information Management, NCM 2008*, 2:719–725, 2008.

[176] Lydia Leong et al. Gartner, magic quadrant for cloud infrastructure as a service, 2014.

[177] A. Botta, A. Dainotti, and A. Pescapé. A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios. *Computer Networks*, 56(15):3531 – 3547, 2012.

[178] Cloud control project, httpmon github webpage. https://github.com/cloud-control/httpmon. Online; accessed Mar '15.

[179] Cristian Klein, Martina Maggio, Karl-Erik Årzén, and Francisco Hernández-Rodriguez. Brownout: Building more robust cloud applications. In *Proceedings of the 36th International Conference on Software Engineering*, pages 700–711. ACM, 2014.

[180] Worldcup98 web site access log. http://ita.ee.lbl.gov/html/contrib/WorldCup.html. Online; accessed Mar '15.

[181] Hamoun Ghanbari, Bradley Simmons, Marin Litoiu, and Gabriel Iszlai. Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723. IEEE, 2011.

[182] Bradley Simmons, Hamoun Ghanbari, Marin Litoiu, and Gabriel Iszlai. Managing a saas application in the cloud using paas policy sets and a strategy-tree. In *Proceedings of the 7th International Conference on Network and Services Management*, pages 343–347. International Federation for Information Processing, 2011.

[183] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1, 2008.

[184] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 9–16. IEEE, 2010.

[185] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507. IEEE, 2011.

[186] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 514–521. IEEE, 2010.

[187] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic resource allocation for shared data centers using online measurements. In *Quality of Service–IWQoS 2003*, pages 381–398. Springer, 2003.

[188] Domenico Grimaldi, Valerio Persico, Antonio Pescapé, Alessandro Salvi, and Stefania Santini. A feedback-control approach for resource management in public clouds. 2015.