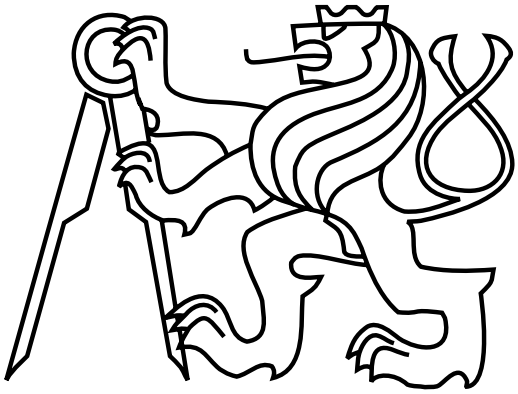
Czech Technical University in Prague
Faculty of Electrical Engineering



Master Thesis

Network-based cloud benchmarking

Vojtěch Uhlíř

Supervisor: Dr. Lukáš Kencl

Study Programme: Electronics, Energetics and Management

Field of Study: Management of Electrical Engineering and Economics

2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra ekonomiky, manažerství a humanitních věd

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Uhlíř Vojtěch

Studijní program: Elektrotechnika, energetika a management
Obor: Ekonomika a řízení elektrotechniky

Název tématu: Benchmarking cloudových řešení na základě dat ze síťového provozu

Pokyny pro vypracování:

- deploy and improve the CLAudit benchmarking tool for different cloud platform - Amazon Web Services to be able to run the latency measurements

- carry out the benchmarking tests and perform data analysis to discuss results based on different metrics

- compare these results with data and pricing model from another cloud solution - the Microsoft Cloud

- the aim of the thesis is to deploy, measure and compare cloud services from two service providers from a technical and economical standpoint and discuss the observations with regards to the impact of cloud service performance and cost on used applications

Seznam odborné literatury:

O. Tomanek, L. Kencl: CLAudit: Planetary-Scale Cloud Latency Auditing Platform, 2nd IEEE International Conference on Cloud Networking (IEEE CloudNet), November 11-13 2013, San Francisco, CA, USA

Raj Jain: Art of Computer Systems Performance Analysis: Techniques For Experimental Design Measurements Simulation and Modeling, Wiley, 2st edition (2015)

Vedoucí diplomové práce: Dr. Lukáš Kencl – ČVUT FEL, K 13132

Platnost zadání: do konce letního semestru akademického roku 2016/2017

L.S.

Prof.Ing. Jaroslav Knápek, CSc.
vedoucí katedry

Prof.Ing. Pavel Ripka, CSc.
děkan

V Praze dne 11.2.2016

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Lukáš Kencl for all advises given and his leadership, to Ing. Ondřej Tománek for the close collaboration and his amazing help during the whole year of my work as well as to the department of my program EEM that allowed me to study related courses and work on this project.

I'd like to also thank to Mahshid R. Naeini, Ph.D. and Jorge E. Pezoa, Ph.D. who helped me to work on the thesis abroad at the Texas Tech University and who also provided me a valuable discussions and fresh ideas for the methodology.

Last but not least I would like to thank to my family and my friends who supported me during the long journey of education at the university towards this Master Thesis and my diploma. My work was also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS15/153/OHK3/2T/13.

Declaration

I hereby declare that I have completed this Master Thesis independently in compliance with Methodical guideline for adhering to ethical principles for academical theses and that I have listed all the literature and publications used.

In Prague on Jan 5, 2017 .....

Abstract

This Master Thesis proposes an innovative methodology how to benchmark global cloud services from the network latency point of view. For this purpose, I have significantly redesigned pre-existing CLAudit platform to perform long term measurements allowing cloud service provider comparison research. The proposed benchmarking methodology consists of a sequence of steps that allows to obtain a ranking value per each provider. To this end multiple global points of presence and various metrics are used, as suits a particular application use case.

In the experiments, two well-known providers, Amazon and Microsoft are used. I have derived a theoretical foundation of the method and demonstrated its application with obtained real measurements to get a practical cloud service provider ranking based on various metrics.

Keywords:

Network latency, cloud, benchmarking, CLAudit, Amazon AWS, Microsoft Azure

Abstrakt

Předmětem této diplomové práce je inovativní metoda pro porovnání cloudových poskytovatelů z hlediska síťové odezvy globálních cloudových služeb. K získání reálných dat byla zasádně rozšířena a vylepšena existující platforma CLAudit. Metoda pro porovnání - tzv benchmarking, je posloupnost několika kroků na jejímž konci je hodnocení poskytovatele na základě různých metrik. Tyto metriky mohou být upraveny tak, aby modelovaly požadavky aplikací nebo jejich nasazení v cloudovém prostředí.

Data jsou získána měřením dvou největších cloudových poskytovatelů, Amazon a Microsoft. V práci je odvozen teoretický a matematický základ pro danou metodu, která je následně aplikována na data z měření reálných cloudových zdrojů. Výsledky poskytují nové informace a hodnocení jako podklad pro rozhodnutí o výběru cloudového poskytovatele z pohledu síťové latence.

Klíčová slova:

Síťová odezva, cloud, benchmarking, CLAudit, Amazon AWS, Microsoft Azure

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing is a major movement in IT and software engineering business nowadays. Customers more and more consider benefits of this approach, a selection of complex cloud services is increasing, number of providers is growing, their budgets are booming. Cloud provides a perfect scaling feature - adding and removing resources have not been ever that easy. This provides a substrate taking form of an underlying infrastructure - it provides a real and flexible resource pool for more innovations, scaling option for products' growth and it is not limit for a company's business.

With this infrastructure shift, different challenges are emerging. Main cloud parameters were targeted with tests and a thorough investigation - a compute power, a memory data throughput and some others. Somehow, cloud resource access time, or network latency, did not get that much attention, even though such evaluation is needed. With current trend of using personal devices and their services - like maps and a car navigation for example, users expect results in magnitude of hundreds of milliseconds, without long access times. In a high frequency online trading business, even single milliseconds matter when deciding about stock trades or betting on a particular situation. Despite that, cloud providers do not publish sufficient information in this field and application testing of different resources would be time and money consuming. This makes businesses indecisive and gullible.

In this work, I present an innovative benchmarking methodology, based on latency measurements of cloud resources, using a redesigned global measurement platform. Network latency is specifically chosen as it is among key performance parameters for the vast majority of applications. As research results show, provider selection and the application deployment location have significant impact when considering quality of service or optimizing its performance.

To achieve the task, this thesis has two major goals. Firstly, as a practical part, it is to significantly upgrade an existing CLAudit measurement platform [1], by extending it to two main cloud providers - Amazon and Microsoft. This extension allows to perform and to collect long-term network latency measurements via active probing at multiple layers of the network protocol stack for different providers using world-wide cloud resources.

The second goal is to propose a benchmarking methodology, enabling cloud providers comparison and resulting in a provider rank value. A theoretical background is introduced to cover all mathematical operations for data preprocessing and provider rank calculation

using different metrics. The benchmarking methodology contains multiple steps enabling to tailor the approach to a specific application needs and business use cases. The single rank value gives an option how to evaluate a cloud provider performance from a global network connection perspective. Demonstration of the method applicability is performed using data from the CLAudit platform resulting in real examples relevant to different use-cases. Discussion of the method results and measurements themselves opens ideas how these findings can help in business negotiations or quality of service evaluations.

With these goals in mind, the content of this work is divided accordingly:

- Chapter 2 contains a cloud computing overview, current state of the art in network latency research field and motivational use cases for this work.
- Chapter 3 introduces the CLAudit platform, its architecture, current deployment, description of my own practical work as well as a dataset which was used for the methodology demonstration and real data examples.
- Chapter 4 describes CLAudit measurement infrastructure cost and economical insight to cloud resource pricing.
- Chapter 5 presents the whole benchmarking methodology, starting with terminology introduction and data preprocessing followed by the step by step manuals how to get the cloud service provider rank, enabling provider comparison.
- Chapter 6 contains results when applying the methodology on real data from the CLAudit platform, and a discussion of the measurements and of the cloud resource performance itself.

Chapter 2

Cloud benchmarking

2.1 Cloud computing

The word "Cloud" is a terminology for a technique and a technology trend, not any more as a buzzword, but a usage set to continue. According [2], by 2020, a Corporate "No-Cloud" Policy Will Be as Rare as a "No-Internet" Policy Is Today.

Origin of the term "Cloud" is unclear, one source [3] defines cloud computing as a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using Internet technologies. According to another one [4], the "cloud" stands for a model enabling 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.

Figure 2.1: A data center building example, Google Inc., Oklahoma.

With application infrastructure resources in a cloud, administrators and developers can leave management of the bare metal computing hardware to someone else and focus on their actual work - application administration and development. For that there are different cloud providers that offer the infrastructure as a business service for prospective customers. In following lines I describe these and other terminology terms that are being used in this specific field. The review proceeds bottom up, from a building with computers to a cloud operations techniques.

A building where hardware resources, computers and a cloud underlying infrastructure are located, is called *Data Center (DC)*. Usually, it is a large and hi-tech building with very solid and reliable power resources, a cooling infrastructure and an internet connectivity. Inside, there is a complex infrastructure with high performance computers. These computers, usually called servers, represent such physical, hardware-based computing resource. An example of such data center building is shown in Fig. 2.1 [5], an arbitrary insight of DC interior is depicted in Fig. 2.2 [6].

Figure 2.2: A data center interior example, Facebook, Inc., Sweden.

A firm utilizes these data centers for its own business purposes and offers various cloud services for its customers. Therefore, related names as a *cloud provider* or a *cloud service provider (CSP)* are being commonly used for such business units. This CSP tries to utilize the infrastructure as much as possible to have the resources fully occupied. To do so, a cloud provider uses different techniques that allows him to share one hardware resource for various different clients. This is also the main approach when considering cloud - not to build and run underutilized on-premise resources. One technique enabling this resource sharing

is called *virtualization* [7], when a power of a real server is virtualized for other purposes, e.g. to represent a set of virtual servers. Then, each of these virtual servers are offered to different customers for their needs and applications.

With respect of this cloud concept, customers of such virtualized servers are usually called tenants. A *tenant* is a customer who consumes cloud provider's infrastructure by buying various services. These tenants are business units with own use-case. They can be large corporate firms, fresh new start-up companies or a single person with her own motivation. All of these can use resource offering as an arbitrary computing power for internal apps, storage or testing machine as well as for a software which serves content for their actual end-users, run some data analytics, business intelligence etc. Tenants can access their resources in various different ways - using just a simple remote access, via a provider's online management website or with a web-based command line tool, for example.

When speaking about tenants, a term *multi-tenancy* [8], is also commonly used in a cloud related discussions. Multi-tenancy is a reference to the mode of operation of software where multiple independent instances of one or multiple applications operate in a shared environment. The instances (tenants) are logically isolated, but physically integrated. The degree of logical isolation must be complete, but the degree of physical integration will vary. The more physical integration, the harder it is to preserve the logical isolation. Tenants (application instances) are representations of units that obtained access to the multi-tenant environment and they have applications competing for shared underlying resources.

A tenant deploys her application to a chosen data center of a particular cloud provider. From that instance of the application, an actual content is served to end user customers. There are also different techniques and technologies how the application can be deployed. It is a compromise of a degree of control over the deployment and application management simplicity. It is very common that providers offer not only infrastructure, but as well as software, database solutions or complete software platforms to use. Main differences between these models are described in the naming terminology. For that, it is very common to use acronym XaaS, which stands for *something-"X" As A Service*. Thus, the main cloud service models are:

- IaaS - Infrastructure As A Service
- PaaS - Platform As A Service
- SaaS - Software As A Service

But not all providers offer all of these options. Some providers offer an IaaS or PaaS solution only, for example. Degree of virtualization and major differences between these approaches are depicted in Fig. 2.3 [9]. This diagram comes from Microsoft cloud provider, which offers all types of these services with its cloud solution called Azure.

Other terms widely used in cloud related discussions are a *public* and *private* cloud. *Public cloud* from a cloud provider is a solution, as the name suggests, for public usage. Anyone seeking a cloud infrastructure can open an account or make a deal with the provider and start using the product. It is a shared environment of a computing infrastructure across different users, business units or businesses. Usually it is an internet facing, so all connections are coming in and out through the internet.

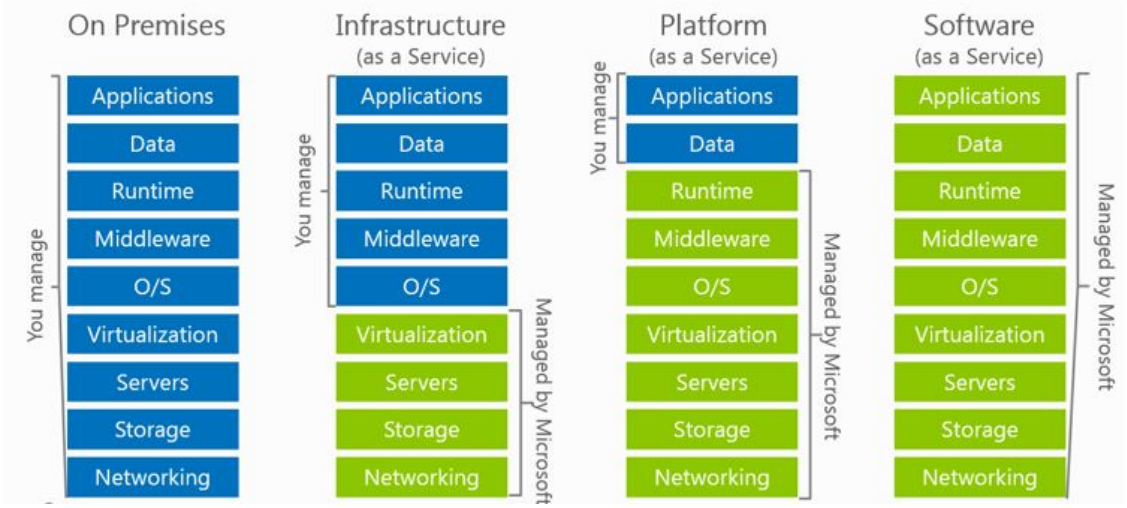


Figure 2.3: Cloud service model differences; IaaS, PaaS and SaaS; Microsoft example.

However, these internet facing, shared computing environments might not be suitable for all businesses and production usage where an isolated environment is needed to have the infrastructure, resources or networking setup under full control.

Private cloud is a model of cloud computing that delivers similar advantages as public cloud, including scalability and self-service, but through a logically isolated architecture. A private cloud is dedicated to the single organization which has complete control over the infrastructure management, especially networking. It includes selection of your own IP address range, creation of subnets or configuration of route tables and network gateways. Private cloud is usually offered by a cloud provider running on its infrastructure (usually called virtual private cloud) or managed in a separate data center. There are 3rd party companies, which offer service of managing private clouds at an arbitrary infrastructure based on customer preference [10, 11].

Based on different studies about market share in cloud environment [12, 13], Microsoft with its solution called Azure takes second place between the biggest public cloud providers. The first place belongs to Amazon and its Amazon AWS cloud solution. Overall, the most well known public cloud providers are:

- Amazon - cloud solution called Amazon Web Services (AWS)
- Google - Google Cloud Platform (GCP)
- Microsoft - Microsoft Azure
- Rackspace, Alibaba or IBM.

Cloud solutions are suitable for small applications as well as for large world-wide services with thousands of customers. For example, Netflix service, the most popular online video streaming portal in the USA, is running its infrastructure at AWS cloud [14, 15].

2.2 Latency-based benchmarking - State of the Art

Network latency is an important, yet often underestimated aspect of the nascent cloud computing scenario. A cloud service based on processing in remote data centers may exhibit latency and jitter which may be a compound result of many various components of the remote computation and intermediate communication. The inherent delay in computer-communication networks has been a long-standing problem and even accelerated nowadays by the boom in distributed applications related to the Internet and cloud computing growth. The key problem is the great demand of the new applications, both for computing resources and for the quality of the communication networks. While the more obvious problem of computing-resources allocation is being deeply investigated, research in improving network-quality parameters, throughput apart, appears not to get as much attention. As a consequence, end-users' time and patience is often the price, and deployment of latency-sensitive applications (e.g. games) in the general cloud has been slow, often favoring proprietary solutions. Cloud computing customers — end users and tenants — thus still stand in wait of satisfactory answers to their application-performance requirements.

Cloud service latency may play a role in many kinds of applications. Starting with simple web-based solutions all the way to collaborative applications, ranging from documents sharing across voice and video telepresence to haptic-operated distributed games or interactive shared 3D worlds. There can be orders of magnitude differences between their latency requirements, ranging from units of seconds to units of milliseconds. Low latency demands are not limited to those applications, but also to the practices and designs inherent to the cloud computing infrastructure, such as replication, task distribution, sharing, synchronization, offload or rapid scaling [1, 16].

Latency is of the utmost importance to the cloud, but the complexity and diversity of this environment prevent the conventional Internet measurement techniques to be easily adjusted to fit cloud computing needs. The problem partly being the scale, because the larger the scale the greater the impact of latency variability and the need for reliably low latency [17, 18].

Each application with its purpose, design and an implementation is different. Customer applications differ in QoS requirements, traffic nature, burstiness and performance parameters. With CSP's offerings homogenizing, competitive differentiation starts to take place at the service quality level. However, CSPs only reveal insufficient amount of technical information about their service, leaving tenants indecisive where to deploy the app resources or which products and solutions to buy. But, in fact, as some results in this thesis show, there are big differences in service quality among CSPs and even among single–CSP's data centers. One technique that enables decision support for selection of CSP and cloud resources is benchmarking [1].

The most accurate benchmarking method will always involve a trial deployment of the actual application, but that usually comes with significant costs, extensive configurations, setup times and need to redo everything for different CSPs. Cross-sectional studies and quick shallow benchmarking of cloud resources via a test suite might partly solve the problem, but have a number of limitations. Specifically, these are not in-depth and, as such, provide a little or no explanation; restrictions imposed by CSP often lead to invalid comparisons

and a short measurement timeframe leads to inaccuracies. Also, these often cannot answer questions related to global distributed applications.

Miscellaneous cloud measurements and analyses were conducted [19], often on platforms and tools designed in academia (like *Fathom* [20] or *Flowping* [21], often using PlanetLab [22]). Deriving traffic characteristics of flows inside a DC was a focus of [23] or [24]. Specific measurements concerning cloud performance include [25] and [26]. End-user-perceived cloud-application performance measurements were discussed for example in [27, 28, 29]. General network delay tomography and specific blackbox latency predictions are the topic of [30, 31], respectively.

Observations from multiple Vantage Points have been used previously [32, 33, 34], but these do not measure back-end or latency at multiple protocol layers. Active probing of cloud resources [35] confirms need for sophisticated measurement, but is availability-oriented and does not document trends. The common goal of all these efforts is a fair comparison of public CSPs using relevant metrics, as well as improving network behavior based on feedback, as implemented in SDN (Software Defined Networking) controllers.

Despite of the wide cloud service adoption between businesses and users, there is still ongoing research looking for a new improvements or solutions of existing cloud related issues. For example, data transfer bottlenecks [36], performance unpredictability [37] and latency [29] are among major obstacles to cloud growth. Benchmarking helps to reveal these, as was shown through its many use cases [38].

A number of cross–sectional (also called *breadth and snapshot*) CSP benchmarking tools considering network performance have been introduced previously: *CloudCmp* comparator [39] measures computation, storage and network resources, the latter using a TCP throughput and end-to-end response times. They developed own CloudCmp benchmarks to stress out each of these components of the Cloud available resources at the particular field. For each services they present different performance metrics to measure. For example for computing part it is benchmark finishing time (to stress out the CPU, memory and I/O), monetary cost to complete the benchmark task or scaling latency. For network measurements it is a throughput and latency measurements. They are focused more on benchmarking a particular services and infrastructure recourses at a short period of time. They run stress tests to measure a pure performance.

Smart CloudBench framework [40] deploys a transactional web–services benchmarking suite and, by measuring response times and recording error codes, estimates the cost–to–performance ratio. They have created a framework which can deploy the benchmarking application on user selected VMs at different providers' infrastructure. For actual performance measurements, they use a widely used benchmarking suit (TCP-W). During the experiments, they collect following metrics for result comparison: average response time (ART), maximum response time (MRT), total number of successful interactions (SI) and total number of timeouts (T) during each benchmark cycle. To offer a cost/performance ratio for a customer, they relate these data with a running cost.

Custom–tailored benchmarking suites for testing CSPs can be created using [41]. In contrast, CLAudit based network–latency–oriented approach works with weeks-to-months of collected RTTs of relevant ISO/OSI layers. Throughput is not considered, to keep monetary costs low and avoid overloading Internet and inter–Cloud links.

For cloud service status verification, various online dashboards exist, either run by a CSP itself (Amazon [42] or Microsoft [43]) or a third–party (*CloudHarmony* [44]). These are insufficient, as they do not offer in–depth comparisons.

Using the benchmarking output, providers can be ranked using multi–criteria decision–making techniques (as done by *CloudGenius* [45]) or utility theory and preference policies (as done by *CloudBroker* [46]).

Related to benchmarking, there are various cloud network performance estimation and prediction techniques. DNS infrastructure [31] or routing topology measurements [30] can be exploited to estimate latency between arbitrary Internet hosts. Performance of a hypothetical cloud service running across global DC deployment can be estimated via measurements [26]. In–house recorded usage traces can be used by *CloudProphet* to predict cloud response times of web applications under migration consideration [47]. Large–scale active probing of cloud in [35] confirms the need for sophisticated cloud measurements, but is availability-oriented only.

The CLAudit platform uses of observations from multiple Vantage Points, which was done in other works – mainly to measure Content Distribution Networks [32, 34] and Internet paths [27, 33, 48].

A number of (often PlanetLab–based [22]) tools for obtaining suitable RTT measurements for latency–based benchmarking exist, namely *CLAudit* [1], *Fathom* [20] or *FlowPing* [21]. Measurement transformations, which the Latency–based benchmarking applies, are described in [49]. Adaptations of different metrics and statistics used to describe application requirements, are based on [50].

2.3 Cloud services - use cases

Network latency plays significant role when considering applications when clients need to have a fast access to data stored at the application server. I present a few use cases when the knowledge of network latency plays a crucial role for business or for a customer usage. It demonstrates when a provider benchmarking can give a valuable information for decisions about application deployments. These use cases provide an insight for understanding that network latency research is a real issue in these days. Latency-based benchmarking will allow a comparison of different CSPs and how to evaluate different solutions for a particular use case. For each use case there are different metrics, which might be particularly important in that situation.

2.3.1 Financial trading

Once upon a time, stock exchanges were packed with traders running, shouting, and elbowing one another on an open trading floor. Today, virtually all stock trading is done through massive, globally interlinked computer systems [51]. In this article, author describes a phenomenon in financial business called high-frequency trading where the network latency response is a crucial part of the communication between trader's company and stock exchange servers.

Demonstration of network latency as a major factor is, that one company invested in an installation of a new shortest route between New York and Chicago stock exchanges. The round-trip travel time of a signal along of a new cable, 13.3 milliseconds, was 3 ms faster than competitors were offering. Lowering it from 16.3ms, it's an 18% improvement. From this example, it is clear to see that even a few milliseconds matter and that it can give a significant competitive advantage. If a customer has a business in this field, he seeks for globally and persistently lowest possible latency.

Trading is executed continuously which means that this low latency is requested in long period of time and in a stable manner, not just as a single minimal value. A threshold value for such latency can be derived from standards on the market or competitor's offers for a particular location. Occasionally or periodically observed high latency peaks deviated from the maximum tolerant value can result in a monetary or a business loss. If a customer has an information earlier than the competitor, he can react on the deal better - with a lower price, a better counter-offer etc. Long term, network latency measurements unfold such information and provide very valuable data for businesses.

Furthermore, in today's globalisation era, such a trading company might be dispersed over different cities to cover business hours in multiple timezones or countries. Benchmarking of different cloud resources around the world can offer an information where to deploy an internal system, for example, which resource will have an acceptable latency response to a global stock exchanges.

2.3.2 Online gaming platforms

Cloud environment provides a convenient scalability of resources, for example depending on a number of connected users. This situation was recognized by gaming studios which start to deploy an infrastructure to a cloud. Furthermore, thin clients have become increasingly popular in recent years, primarily because of the high penetration rate of broadband Internet access and the use of cloud computing technology to build large-scale data centers. The massive computation and storage resources of data centers enable users to shift their workload from local workstations to remote servers [52].

Latest games usually come with high requirements on a computer performance to play the game, including all new features, using of some brand new functions of the latest graphics card generation. This can meet an anxiety of users, gamers, who are forced to upgrade their machine or a gaming console regularly. Instead, these high intensive computations can be shifted right to the cloud, when the traditional fat client is replaced with a thin, interactive client.

But to meet high client expectations and ensure the same quality of experience as clients are used to, there is a high demand on cloud computing infrastructure parameters. They need to meet the strict network latency requirements necessary for an acceptable game play for many end-users.

Client's game stream should report a stable network latency for a smooth experience. This can be addressed by statistical metrics as a variance or a standard deviation. A threshold value which is still acceptable for a smooth game flow can be empirically measured. For action games this threshold can be very high demanding (very low latency value) due to rapid scene changes, for a turn-by-turn game on the other hand, this threshold can be more

flexible. If the network latency has some extremes in values, a game player might some video lags when she stays where she was an then jumps somewhere else. This has a strong impact on the user experience of course.

Online gaming is usually world wide based - one massive online game is played around the globe. This opens a demand for overall measurements and data which would represent clients in those locations. Traffic can be also routed to one specific application server location or split to a client nearest/best suitable server. World-wide data center latency benchmarking can offer aggregated data for such questions and to make a decision about application deployment easier.

2.3.3 Navigation systems

Let us consider an application deployment which serves maps and other navigation information to the clients. This got very popular recently, due to the fact that a lot of clients use GPS navigation systems with online data connection on their smartphones. Using a smartphone and a data connection, it is very easy to find a map of surrounding area where the user is and she wants to go. This comfort for smartphone users comes with a heavy load of queries against the application provider resources. This is a perfect use case for global latency measurements from a multiple following reasons:

- Masses of people requesting map details in a traffic peak hours are very geo-collocated in a relatively small area. A best fit resource/data center location can be found. A benchmarking approach can find and to propose the best fit data center for a location where most of clients are located.
- The peak hours are not everywhere at the same time. Cloud resources can be easily turned on and off when needed. So, for example, a central Europe can be served from a running resources in an European data center, meanwhile in a timezone of America's East Cost, the resource pool in an American data center can be down. Once time passes during the day and peak hour in Europe is over, these resources can be suppressed. On the other hand resources in the American data center serving this region might be extended to serve the upcoming peak hour traffic. From a long term measurements, a median metric can answer how fast a map on the smartphone will retrieve requested data. A 95th percentile statistic metric can answer the worst case for a loading time for most of users.

For both these situations, a network-based cloud benchmarking can resolve concerns and questions related to a best customer experience. With a lot of different apps and solutions on the market, failing to fulfill customer satisfaction can result in a customer and business loss, as she can easily switch to a competitive app.

Chapter 3

Network latency measurements

A prerequisite for conducting the latency–based cloud benchmarking is a collection of some amount of RTT (Round Trip Time) delay measurements between desired sources and destinations. A greater accuracy is achieved with, firstly, a large amount of recent measurements and, secondly, RTTs measured at multiple protocol layers. For such a measurement a data collection platform was created – *CLAudit* [1]. Online information, near real-time data analysis or measurements data are available at *http://claudit.feld.cvut.cz* [53]. Using this collection platform resulted in a research, as [54] and [16], that has demonstrated how this approach can bring an innovative data and results. But the platform had only one-purpose locked original architecture and with a focus only on measurements of a single cloud service provider (CSP), sufficient for research at that time. But for a next research areas, the latency based benchmarking for example, the platform had to be extended. For a benchmarking approach with two CSPs, it was necessary to redesign the architecture and tune up the internal algorithms for more thorough latency measurements and additional capabilities.

First, I introduce the platform itself and it's architecture. Then, I describe my own work and my improvements implemented into the platform. Also, there is a description of current deployment and an overview of the dataset which was used to apply the benchmarking theory from chapter 5.

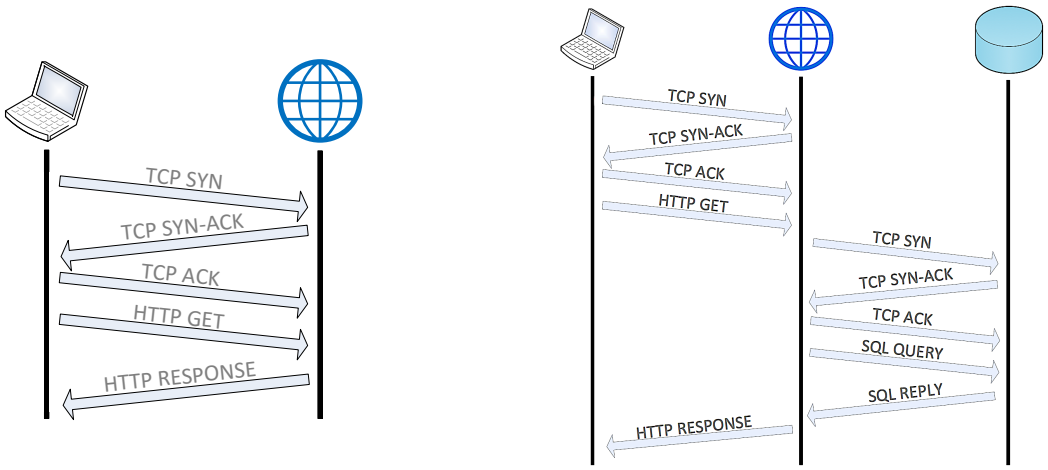
3.1 Cloud Latency Auditing platform

3.1.1 Architecture

CLAudit alias Cloud Latency Auditing Platform is a system for collecting and evaluating multidimensional measurements. By *measurements* are meant RTTs of individual protocol exchanges, processing times and a query latency, as shown on a data retrieval processes in Fig. 3.1. *Multidimensional* means measurements capable of being looked at from a point of view of Vantage Points, data centers and/or protocol layers of networking stack.

The request-response nature of many existing protocols allows measuring RTTs and deriving latency. That is just what CLAudit does - in a continuous and large-scale distributed manner. Several time intervals and metrics (shown in Fig. 3.1) are recorded by active probing of protocol layers between every *(VP, front-end)* and *(front-end, back-end)* instance pairs.

There is neither a front-end, nor a back-end selection algorithm, as all the pairwise combinations are measured. Thus, values of the Internet, intra-DC and inter-DC latencies are measured. Besides protocol–layer latencies, CLAudit collects an *overall* latency, which is an end–user perceived latency of the interaction. The overall latency, although most accurate for user–experience–driven applications, is difficult to obtain, as it requires deployment of the entire application for benchmarking purposes – something a tenant wants to avoid.



(a) Static webpage retrieval (b) Dynamic webpage retrieval

Figure 3.1: Network interactions during CLAudit latency measurements.

The CLAudit platform consists of components, which reflect a typical cloud computing application setups – i.e. Internet–connected client devices and DC–hosted front–end and back–end servers. Fig. 3.2 shows a subset of CLAudit deployment. A detailed description of the these components follows:

Vantage Points (VPs). Remotely-controlled VPs emulate real client appliances for interacting with cloud-hosted applications and services. They collect, at various protocol layers, latency measurements of what end users perceive when utilizing cloud. VPs are geographically dispersed to obtain end-user perspectives from around the globe. To maintain comparability of the measurements, VPs are homogeneous in terms of OS and software (i686 Fedora Core 8 PlanetLab hosts). For the sake of redundancy and validation, VPs are deployed in triplets in every region (two VPs in a single location and a third, backup VP in a different location nearby).

Cloud front-end. Servers that serve client requests (e.g. in this context respond to VP requests). Front-ends are geographically dispersed across CSP DCs. For the purpose of the experiments, front-end servers are implemented as PaaS web applications within a shared usage tier (a single shared instance without backup per DC). Implemented as PaaS means that a provider's service to deploy the code without managing an operation system/web server resource is used. Only platform source files are provided, the whole management of spawning a server or the web server configuration is done internally by the provider.

Web server consists a two parts. The first one is a static web web-page served for front–end latency measurements initiated from a VP. Process is depicted in Fig. 3.1a. The second

one is a dynamic script which is executed when a back-end measurement is called, Fig. 3.1b.

Cloud back-end. Servers that provide data when front-end servers need them to compose the client response (e.g. in a case of dynamic webpage in Fig. 3.1b). The back-end is, by definition, not involved in every interaction, although cloud applications often consist of both the front-end and the back-end. In the setup, several back-ends are co-located with front-ends within a single DC, whereas other back-ends reside in DCs elsewhere. That is to take into account scenarios like remote storage, georedundancy, data processing regulations etc. For the purpose of the experiments, back–end servers host a MySQL database.

Monitor. Monitor is a single dedicated server, that gathers measurements from all VPs; instructs and adjusts measurement setup, runs analytics (including benchmarking); archives and analyzes the past and near-real-time measurements. Monitor is hosted on a high-end server on premises of the Czech Technical University in Prague.

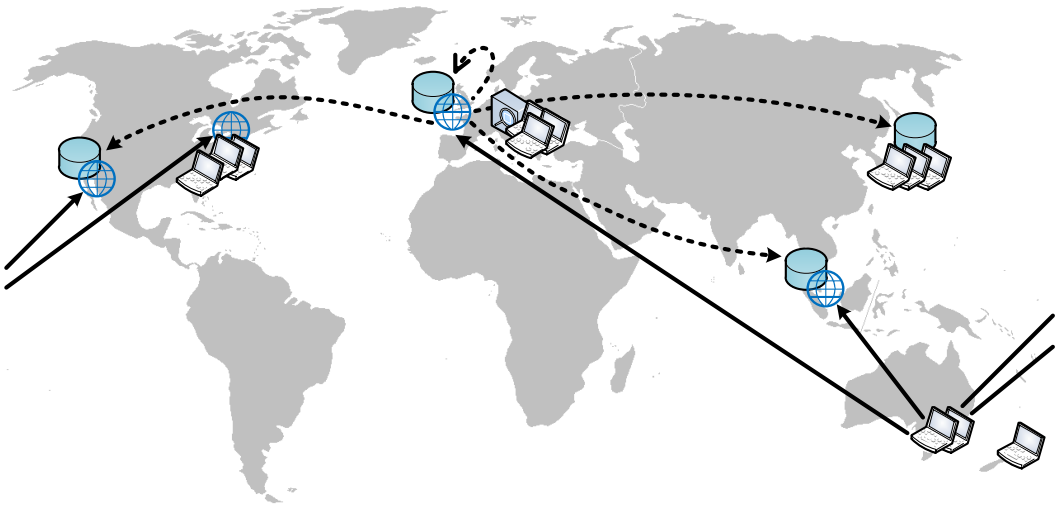


Figure 3.2: CLAudit global deployment and points of presence.

Part of the CLAudit infrastructure used for Latency–based benchmarking demonstration. VPs are represented by laptops, front–end servers by globe icons and back–end servers by cylinder icons. Benchmarking computation is done at a central server, represented by lens icon. Continuous and dashed curves depict VP to front–end and front–end to back-end measurements, respectively (only measurements triggered by Australian VP and Dublin front–end are shown).

3.1.2 Testbed

In a previous research and the CLAudit version, only resources from one provider were measured. That was a Microsoft Azure cloud solution. To extend research possibilities, to gain a new data and allow a benchmarking study, another provider was added. This brought a lot of difficulties with the technical solution, which I describe in chapter 3.2. As MS Azure is the 2nd biggest cloud provider, an obvious choice was to pick the biggest one and its solution for a comparison and benchmarking - Amazon and its AWS public cloud. Services

and technical details are of course different from provider to provider, I exploit services and resources which are used for the CLAudit platform in following lines.

Front-end resources. As I already mentioned, for front-end resources, a PaaS for web servers are being used. Each provider has a different name of such a service. In the AWS ecosystem, the service name is Elastic Beanstalk. A more simple name, called Wep App, is used at Microsoft Azure.

Both services spawn a web server with the actual code completely effortless. This also ensures a fair setup for both providers, it eliminates any possible issues with a virtual machine, an operating system, system drivers etc. One can assume that this deployment goes with the provider's best knowledge of its own underlaying infrastructure. The only thing what is configurable when requesting such a service, is to pick the right instance "power/size", the right performance scale for the unit. There are plenty of choices for the underlying power instances for such web-servers.

Usually, the selection of underlying server for this service matches with compute power instances offered for virtual servers. It scales on a number of CPU cores, memory size or storage volume attached, for a few examples. For the date of December 2016, AWS offers 54 server instance types, from a cheap, general purpose servers, up to expensive, very powerful servers for a big scale and a production use.

An example of a list with different instances from Amazon is in Tab. 3.1.

Instance Type	vCPU	Memory (GB)	SSD Storage (GB)
m3.medium	1	3.75	1x4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

Table 3.1: An example of compute resource instance types at Amazon AWS.

CLAudit platform does not perform any stress tests of the resources, so no large computing instances are required for the measurements. Even more, the purpose of such measurements is to provide an insight for a broad spectrum of use cases and applications. It's an obvious fact that for benchmarking between two solutions, there is a need to pick as similar resources as possible, to compare comparable. So the difficult part of a resource selection was to choose instance parameters which would be similar for both providers. Even a price is a factor which needs to be taken in account.

Based on these parameters and provider resource analysis, a following power tiers and instances were selected. Instances with similar cost on a shared infrastructure were chosen. The cost structure is explained in Chapter 4.

Provider	Instance Type	vCPU	Memory (GB)	Storage
Amazon	t2.micro	1	1	EBS Only
Azure	D1 Shared	Shared	0.5	1 GB

Table 3.2: CLAudit front-end resource instance types used.

Back-end resources. As it was described in the chapter 3.1.1, MySQL databases are used to represent back-end solution for the CLAudit application. In AWS cloud, a service providing SQL databases is named Amazon RDS (Relation Database Service). In Azure, it is simply SQL Database Service.

Selection approach for the right database instances was similar as in the previous case for front-end resources. Instances need to be as similar and comparable as possible, to ensure a fair measurements for both providers. Selected database instances are presented in Tab. 3.3. For example, Azure's *DTU (Data Transcation Unit)* metric is a complex variable so more detailed and up to date technical information about instances can be found at providers websites [55] and [56].

Provider	Database Instance Type	Performance note
Amazon	db.t2.micro	Instance with 1 vCPU and 1 GB memory
Azure	Standard S1	Max transactions 20 DTU

Table 3.3: CLAudit back-end resource instance types used.

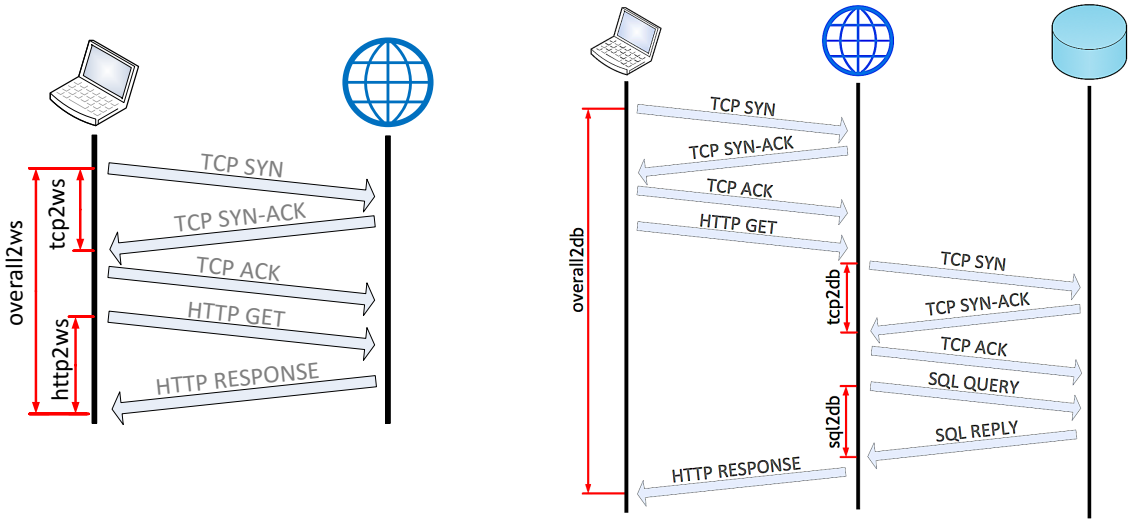
3.1.3 Measured variables

. A short overview of the CLAudit testbed measurement variables follows. In-depth description is presented in the [1]. Selected measured variables (or types) concerning the measurement sample interaction are depicted in Fig. 3.3 and include:

1. **tcp2ws**. Time between the client sending a TCP SYN packet to a web server and receiving the TCP SYN-ACK packet back from the web server.
2. **http2ws**. Time between the client sending a simplest HTTP request packet to a web server and receiving the simplest HTTP response packet back from the web server.
3. **tcp2db**. Time between a web server sending a TCP SYN packet to a database server and receiving the TCP SYN-ACK packet back from the database server.
4. **sql2db**. Time between a web server sending a simple packet containing an SQL query to a database server and receiving a packet containing the SQL response back from the database server.
5. **overall2ws, overall2db**. Total time between the client sending a TCP SYN packet to a web server and receiving the HTTP response packet back from the web server.

Note that database interaction is mandatory in the **tcp2db and sql2db** types and not present in the **tcp2ws, http2ws** types. Also note that data from the **tcp2ws, http2ws, tcp2db and sql2db** types represent the measured lower bound of the respective protocol RTT. Data from the **overall2ws, overall2db** type represent the user-perceived latency during a web-browsing scenario, with a possible intermediate query to a database.

As previous research shows [1], [16], the platform provided a valuable long-term data when considering a latency delay during a network communication to the cloud. These publications present latency research and observations how to exploits such a measurement. CLAudit



(a) Static webpage retrieval

(b) Dynamic webpage retrieval

Figure 3.3: CLAudit-measured variables during latency measurements

platform architecture providing measurements of a single provider (Microsoft Azure) was sufficient.

But that previous platform architecture was not maintainable any more, though. It did not provide sufficient versatility for future research neither it was suitable for multi provider measurements, which was needed for the benchmarking approach. Even further, as an only one-purpose written solution, almost everything was statically fixed inside of the code of the platform, which was hard to maintain for any extension, for different provides, different configurations etc.

3.2 My platform extensions for benchmarking approach

In this chapter, I describe my work on the CLAudit platform. First of all I had to get familiar with the architecture, with the whole environment, the VP setup, back-ends, front-ends and about the monitor node. This gave me an insight how the measurements work.

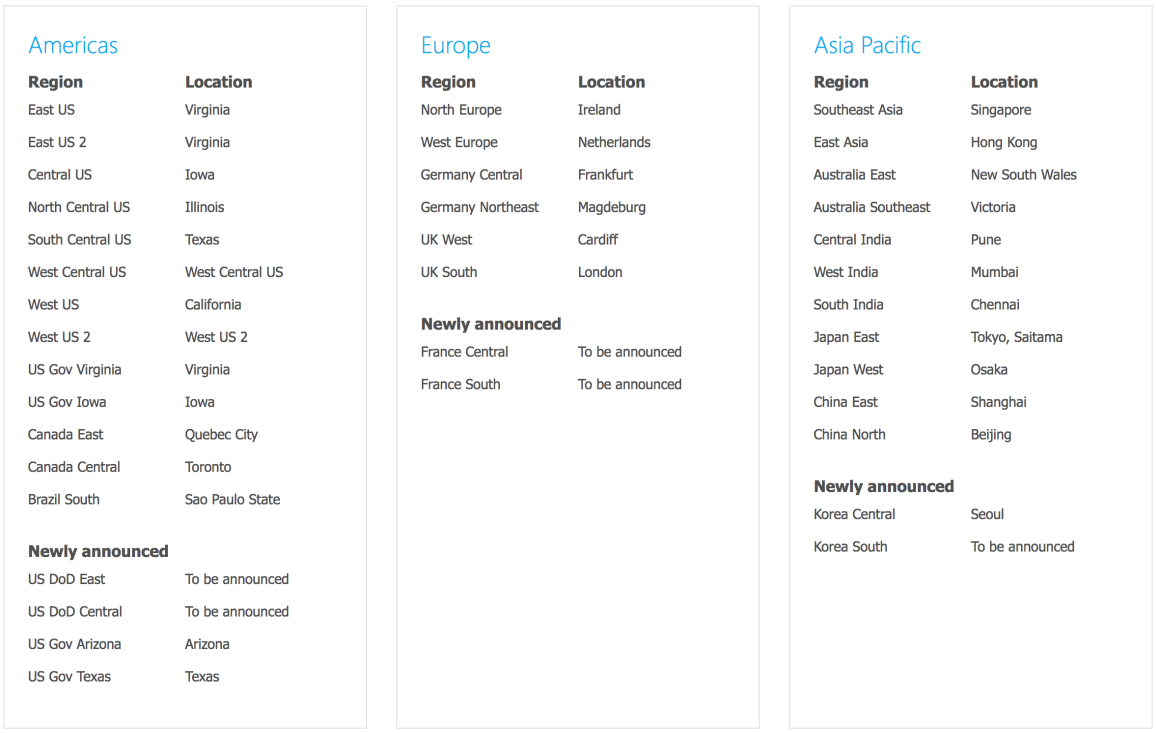
3.2.1 Data–center locations

As mentioned before, the previous research was based on measurements against Microsoft Azure CSP only. For the benchmarking approach the Amazon AWS cloud was chosen as the second provider. Naturally, a large number of prospective customers are deciding between these two. AWS is de facto leader in cloud technologies and it is widely adopted and known, followed by the Azure solution.

For the benchmarking use case it is necessary to measure as identical targets as possible, though. It was necessary to find as geographically close data-centers as possible for both

Amazon AWS and Microsoft Azure cloud solutions. One advantage when deciding about Azure data centers was existence of a free map with approximate location of all Microsoft data-centers. [57].

No similar map for Amazon data-centers existed that time (fall 2015) so I had to get an approximate locations from different resources, as [58] for example. I matched these sources with information from actual availability zones where one can deploy the resource in the AWS environment within the user account. One year later update (fall 2016), an official map for Amazon data centers also exists [59]. An example of a list containing all data centers used in Azure is in Fig. 3.4.



Americas

Region	Location
East US	Virginia
East US 2	Virginia
Central US	Iowa
North Central US	Illinois
South Central US	Texas
West Central US	West Central US
West US	California
West US 2	West US 2
US Gov Virginia	Virginia
US Gov Iowa	Iowa
Canada East	Quebec City
Canada Central	Toronto
Brazil South	Sao Paulo State
Newly announced	
US DoD East	To be announced
US DoD Central	To be announced
US Gov Arizona	Arizona
US Gov Texas	Texas

Europe

Region	Location
North Europe	Ireland
West Europe	Netherlands
Germany Central	Frankfurt
Germany Northeast	Magdeburg
UK West	Cardiff
UK South	London
Newly announced	
France Central	To be announced
France South	To be announced

Asia Pacific

Region	Location
Southeast Asia	Singapore
East Asia	Hong Kong
Australia East	New South Wales
Australia Southeast	Victoria
Central India	Pune
West India	Mumbai
South India	Chennai
Japan East	Tokyo, Saitama
Japan West	Osaka
China East	Shanghai
China North	Beijing
Newly announced	
Korea Central	Seoul
Korea South	To be announced

Figure 3.4: Microsoft Azure Data Center locations in December 2016.

Based on these available options, the following data center locations were chosen. Very similar locations for both providers and convenient for the research measurements are: Dublin (Ireland), North Virginia, California (East and West coasts of USA), Singapore and Tokyo (Japan). Once data center locations were fixed, I have created the resource pool in AWS cloud for the future platform code deployment and measurements. Also, the existing resource pool in Azure cloud had to be managed and reordered to be complaint with the new data center locations. A lot of CLAudit platform upgrades and code improvements were integrated to the platform before an actual code deployment and a real measurements launch.

3.2.2 Configuration file

The previous CLAudit version was significantly focused on a single provider measurements. To break that down I have suggested and developed a solution when all configurations, target names and other versatile parameters are separated to a special configuration file. Then, I have implemented functions that all parameters within in configuration file are loaded to the application when needed. This is a recommended approach from a software engineering in general where configuration parameters are out of the actual application.

I have decided to use a YAML [60] format for configuration files. This format is well structured, human readable and quickly to learn. These properties are very convenient for further upgrades or updates in the future. The configuration file example follows.

```
version: 4.00
measurement-data-folder : /home/cesnet_felcc/measurements/
measurement-providers:
 azure:
  ws-address-prefix: ws-azure-
  ws-address-suffix: azurewebsites.net
 aws:
  ws-address-prefix: ws-aws-
  ws-address-suffix: elasticbeanstalk.com
measurement-layers:
 - http2ws
 - tcp2ws
 - trace2ws
 - tcp2db
 - sql2db
 - overall
planetLab-locations:
 cz:
  nodes:
   planetlab2.cesnet.cz: prim
   planetlab3.cesnet.cz: sec
   planetlab1.cesnet.cz: ter
   db: dub
 ...
 ...
```

3.2.3 File naming notation

I have also proposed a new naming convention for files with the measurement data. The original naming convention was following: *{measurement type}_{VP location}_{Front-end target location}_{Back-end target location or null}_DDMMYY*. Due to a lack of information for primary, secondary and back up of VP assignment, all files were stored separately in three folders, accordingly.

This format and file management had a few weak places. First, there was no information about the provider. It was sufficient for the previous research with one provider, but that was not sustainable any more. An information about provider had to be added to differentiate files from different CSPs. Then, there was the situation with the three folders structure. Files from same measurements (same network layer, source locations etc..) had exactly same names in these folders. By any mistake, results could be irretrievably interchanged. Therefore, I have advised and implemented a way with one measurement folder, but with appropriate filename mark. The *backup* naming style was deprecated, a set of primary, secondary and tertiary terms is used instead.

And for a third issue, a sequence for a date mark *DDMMYY* is not really convenient from sorting and browsing point of view. Since all files in any file explorer are displayed and sorted by comparing characters from the beginning of the file name, this approach makes an unorganized mess when looking at a long list of such files. And with the long term measurements nature and a pair permutation of all measurement layers, there are thousands of files. To resolve this, I proposed a sequence YYYYMMDD in the file name notation, which completely wipes out relates issues.

Based on all these concerns together, I have implemented to use following file name convention within the platform: *{provider name}_{measurement type}_{VP location}-{primary/secondary/tertiary level}_{Front–end target location}_{Back–end target location or null}_YYYYMMDD*. A file list thumbnail from a file system is shown in Fig. 3.5

3.2.4 VP locations

PlanetLab nodes serving as VPs (Vantage Points) for measurements (described in the chapter 3.1.1) had to be redesigned as well. For example, measurements from two nodes deployed in a New Mexico location (University of New Mexico (UNM)) had a unusually higher latency through all measurements. It was around a double value what was an expected value for USA. I have performed some trial deployments at other PlanetLab US located nodes and observed that the higher latency is caused just by the UNM nodes. All other values were at the similar level and performing within an expected range. This issue was resolved with new node locations. Also, the previous version of CLAudit platform worked with nodes in a Russia region as well. From past observations there were many inconsistent measurements, there were a lot of traffic interrupts and other abnormalities. This was unwanted situation and required an update of these nodes. Unfortunately, no better or more suitable PlanetLab node was found in Russia. Therefore there is no deployment in that region and no measurements from Russia are collected.

As a result, I performed trial deployments for all other suitable candidates at other locations to find out the best VP nodes for the long term measurements. Based on test results overview I have chosen final destination for the VP clients. The list is following:

1. North America region - primary and secondary nodes are in Atlanta city area, USA. A tertiary node is at North Carolina area, USA.

2. Asia region - primary and secondary nodes are in Hiroshima city area, Japan. A tertiary node is at Nagoya city area, Japan.

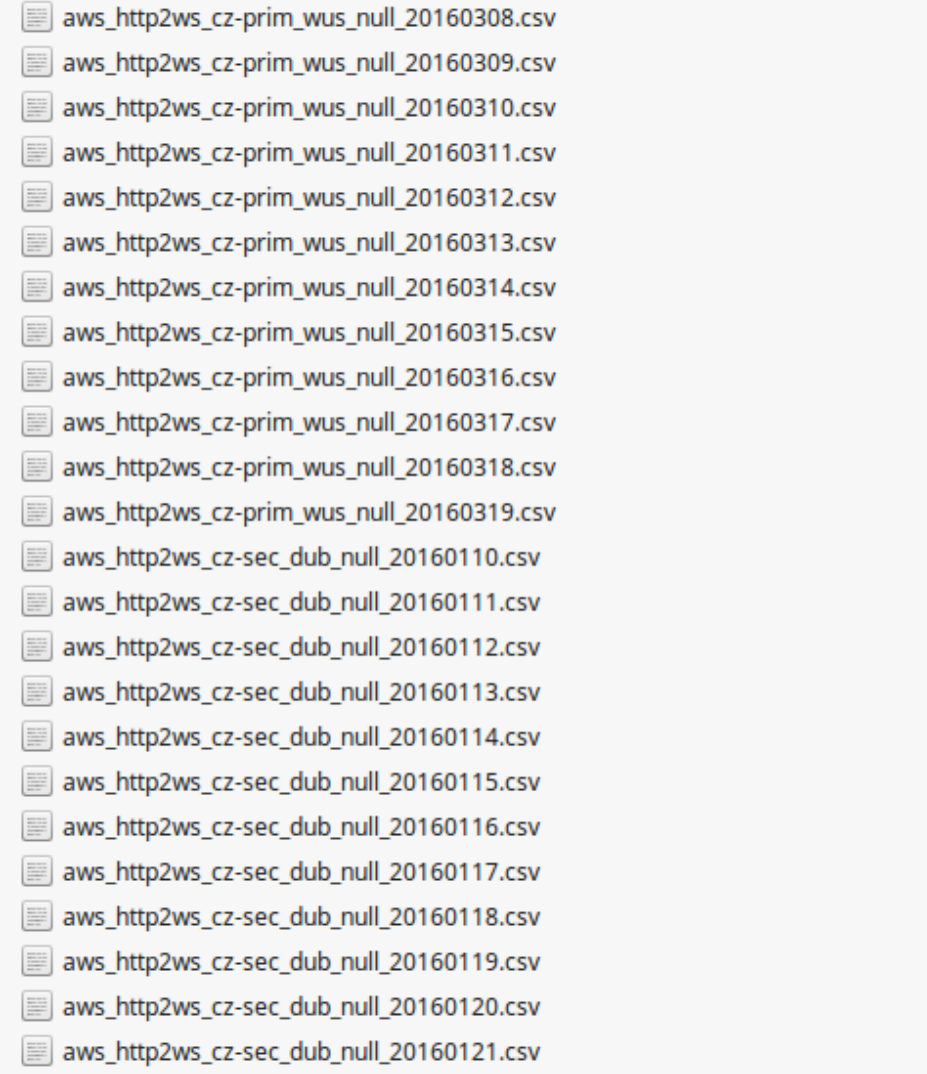


Figure 3.5: An example of a list of files showing the naming convention.

3. Australia region - primary and secondary nodes are in Melbourne city area, Australia. A tertiary node is in New Zeland area.

4. European region - primary and secondary nodes are in Prague city area, the Czech Republic. A tertiary node is in Brno city area, the Czech Republic.

3.2.5 Platform code redesign

Alongside with the configuration file creation, I have redesigned, improved and added a new code to the actual platform source code. I have integrated usage of that configuration file to an appropriate places. Also, I have changed the code to a way that any other and additional provider can be seamlessly added. I have also fixed a few bugs which were known at the previous version, but not fixed so far. A short preview of my code follows.

```
'''
Setup the measurement according parameters and
 content of the configuration file.
'''

def setup_measurement(hostname, host_region, config, provider):
  try:
    host_type = config['planetLab-locations'][host_region]/
                            ['nodes'][hostname]
    website_domain = config['measurement-providers'][provider]/
                            ['ws-address-suffix']
    ws_prefix =  config['measurement-providers'][provider]/
                            ['ws-address-prefix']
    webserver_regions = dict()
    for ws in config['ws-locations']:
        webserver_regions[ws]=ws_prefix+ws

        database_regions=dict()
        for pl in config['planetLab-locations']:
          db = config['planetLab-locations'][pl]['db']
          if db:
            database_regions[pl]=db

    output_folder = config['measurement-data-folder']
    m_order = config['planetLab-locations'][host_region]/
                    ['measurements-order']
    return host_type, website_domain, webserver_regions,/
                database_regions, output_folder, m_order

  except Exception, exc:
        print "Error in the configuration file:", exc
        sys.exit(1)
  ...
```

3.3 Measurement dataset

Once the whole platform was prepared, I deployed the measurement infrastructure and scripts to VPs and cloud resources and the measurements started. I have to tune up some parameters in the script to obtain a smooth data retrieval. In the final state, a measurement iteration of a single VP/front-end consists of a 5-request batch of every relevant protocol sent and up to 5 respective responses received, repeating every 4 minutes. There is no intent to conduct a stress test - neither PlanetLab's, nor CSP's acceptable user policy is violated. Partly because a generated network traffic is low and partly because it is scattered over

time due to unequal distances between VPs and DCs. Monetary cost of the measurements collection remains thus low, as it is described in Chapter 4.

Latency RTTs are measured in milliseconds, rounded up to a nearest integer. All measurement types have a reasonable 10 seconds timeout set, effectively treating timeouts and failures to respond the same way. The platform thus collects several hundreds of samples per *(measurement_ type, VP, front-end, back-end)* tuple.

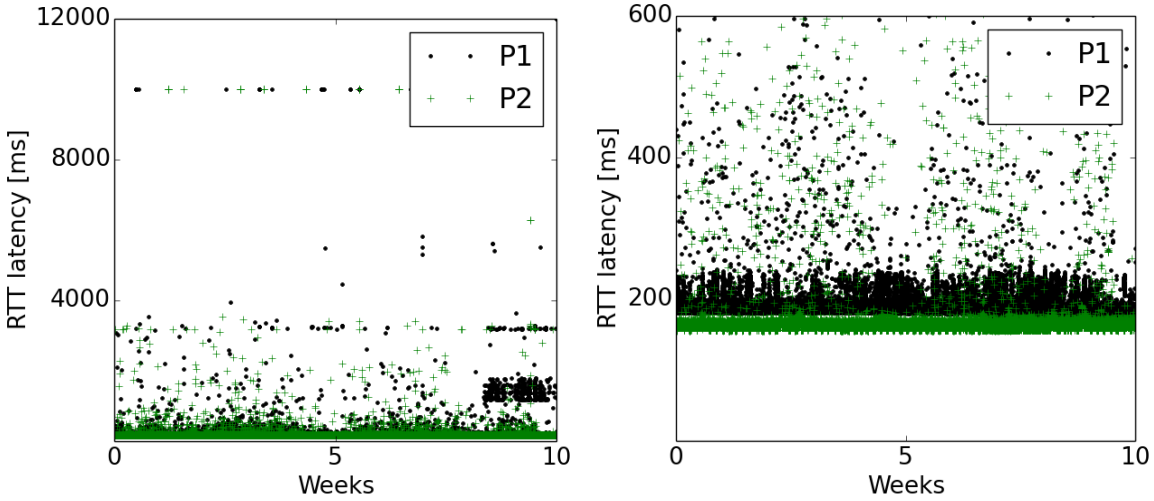
The following CLAudit deployment subset (depicted also in Fig. 3.2) was used to gather a dataset for the purpose of latency–based benchmarking demonstration:

- L = {TCP, HTTP, SQL} - Protocols used inside application that have round trips involved (TCP and HTTP between client web browsers and front–end web server; TCP and SQL between front–end web server and back–end SQL database in the example of dynamic web application)
- V = {AU, CZ, JP, US} - Internet Vantage Points, representing a global client base (Australia, Czech Republic, Japan, USA)
- F = {DUB, EUS, SING, WUS} - AWS and Azure front-end web server DC locations (Dublin, North Virginia, Singapore, California)
- B = {DUB, SING, TOK, WUS} - AWS and Azure back-end SQL database DC locations (Dublin, Singapore, Tokyo, California)
- P = {P1, P2}. CSPs being compared. CSP names are obfuscated and the alias was decided by a coin toss, as I intend to demonstrate the Latency–benchmarking methodology only.

For a practical demonstration of the benchmarking methodology introduced in Chapter 5 presented in Chapter 6, a fragment of the continuous measurements was used. A timeframe from January 10th 2016 to March 19th 2016 was selected (exactly 10 weeks). The main reason is the relatively good stability of PlanetLab nodes during that time and no abnormal demand across the world.

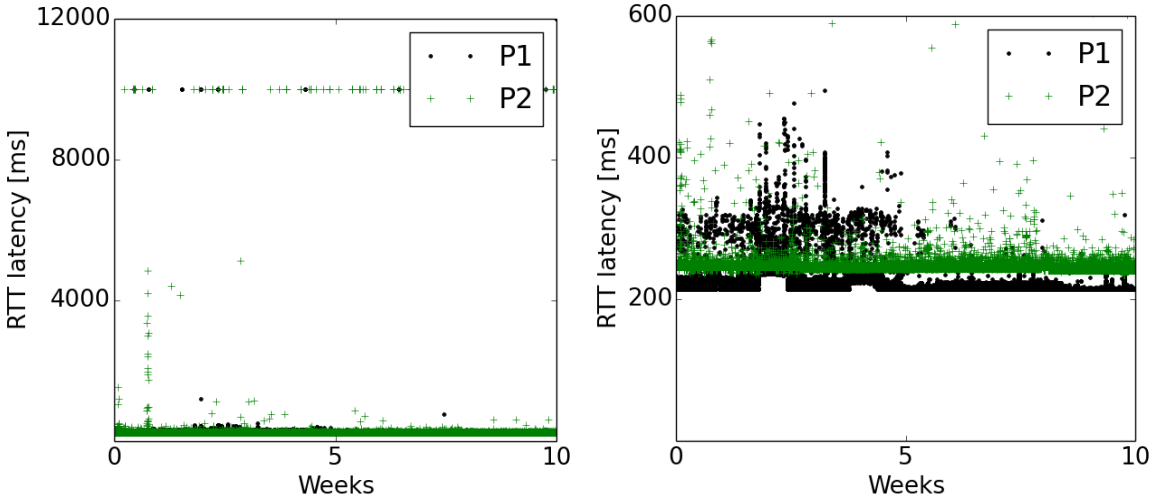
Examples of measurements and findings are in Fig. 3.6 and in Fig. 3.7. These examples show raw latency measurements, collected over 10 weeks at CSP1 and CSP2. As there are number of outliers and timed-out measurements (RTT = 10 000 ms), there are presented full scale graphs as well as scaled plots to encompass majority of measurements and highlight interesting differences in measurement timelines of the two CSPs.

Fig. 3.6 presents measurements on TCP layer between Prague VP (CZ) and front-end servers located in California region (WUS). In Fig. 3.7, there are examples showing SQL queries issued by a front-end web server hosted in Dublin (DUB) and answered by a back-end database server hosted in Tokyo area (TOK).



(a) Full scale view (b) Scalled view

Figure 3.6: Raw measurement example, TCP layer, VP in Prague - WUS front-end servers.



(a) Full scale view (b) Scalled view

Figure 3.7: Measurement of SQL queries from Dublin front-end servers to Tokyo back-ends.

Chapter 4

Measurement platform cost insight

Infrastructure cost is one of the main reasons behind a decision to move from on-premise solution to a cloud-based deployment. But to choose the best fit resources for the infrastructure or an application comes with significant cost and a work time consumption as well. With the CLAudit platform data from the long term measurements, obtained results can save money and time significantly. With multiple points of presence of data centers and clients, benchmarking results can give a first insight without an actual trial application deployment. In this chapter, I present cost of the CLAudit measurement infrastructure.

For a general cloud spending calculations, there are online calculators available, where it is possible to make a rough estimate for an overall deployment cost. Amazon calculator thumbnail is in Fig. 4.1 [61]. There is a similar web page for Azure cost estimate as well [62]. Both pages have limitations within the functionality and requires a deep knowledge of the deployment and service offering. Cloud cost structure is very wide topic, since there are tens or hundreds of different parameters, metrics or calculation approaches. It is beyond of the scope of this work. Here, I present only cost to run CLAudit platform and measurements.

4.1 Amazon - AWS

As it was described in Chapter 3.1.2, service called Elastic Beanstalk for all front-end deployments is used. For a back-end database servers, instances of Amazon RDS are being used.

When calculating the service cost, the Elastic Beanstalk service does not come with any additional fee of itself. When a service is used, tenant pays only for computing instances that power the application. There are four options related to payment and cost for Amazon EC2 (Elastic Compute Cloud) instances. There are *On-Demand, Spot instances, Reserved instances and Dedicated hosts.* For Back-end instances, there are only two options: *On-Demand* and *Reserved* database instances. From a nature of the CLAudit long term measurements, only On-Demand and Reserved instances were really relevant and interesting for consideration. They are suitable for applications that have steady state or predictable usage and targeting general public usage. Reserved Instances provide a significant cost discount compared to On-Demand instance. For long-term measurements what CLAudit does, it comes with significant savings [63].

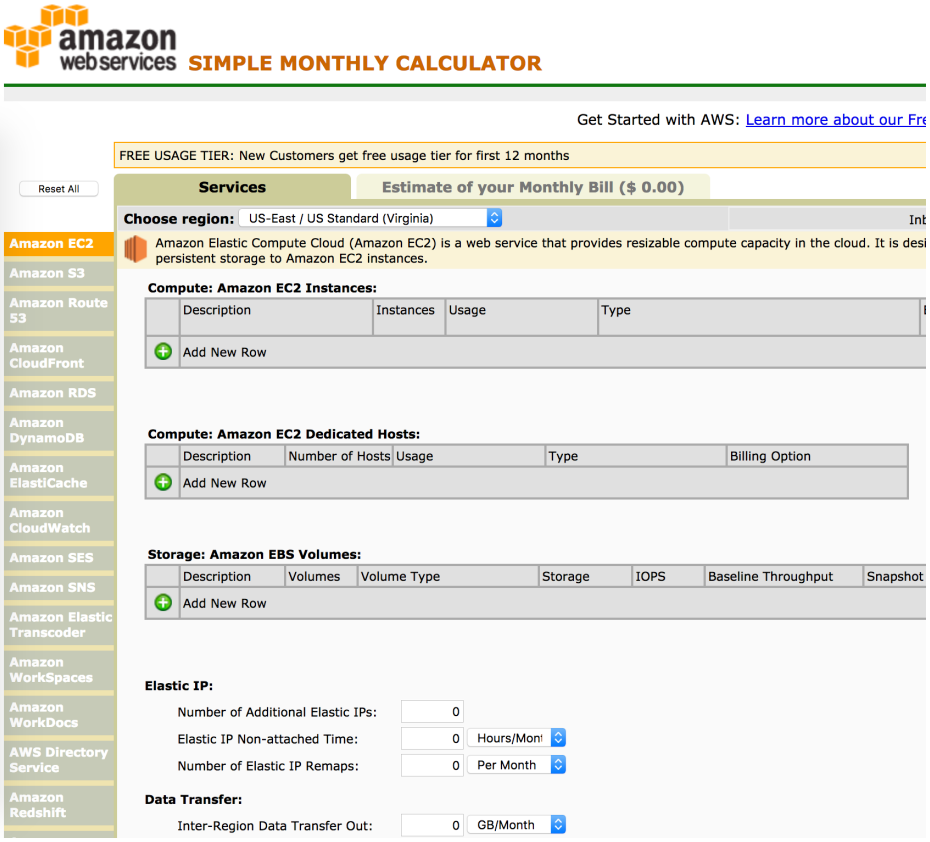


Figure 4.1: Amazon AWS cost calculator example.

When deploying the CLAudit infrastructure, a reserved instances for a three years contract were bought. Cost structure for front-end, EC2 server instances is shown in Tab. 4.1. Cost structure for back-end, database resources is in Tab. 4.2. The presented prices were paid at the moment of reserved instances purchase, the November 30th, 2015.

Instance Type	Data center zone and location	Term	Cost (including VAT)
t2.micro	eu-west-1; Dublin	3 years	$200.86
t2.micro	us-east-1; North Virginia	3 years	$182.71
t2.micro	ap-southeast-1; Singapore	3 years	$350.90
t2.micro	us-west-1; California	3 years	$252.89

Table 4.1: AWS compute reserved instances; 3-year contract cost overview.

Instance Type	Data center zone and location	Term	Cost (including VAT)
db.t2.micro	eu-west-1; Dublin	3 years	$261.36
db.t2.micro	ap-southeast-1; Singapore	3 years	$471.90
db.t2.micro	ap-northeast-1; Tokyo	3 years	$389.62
db.t2.micro	us-west-1; California	3 years	$343.64

Table 4.2: AWS database reserved instances; 3-year contract cost overview.

Based on cost shown in Tables 4.1 and 4.2 a total cost is $2453,88 for 3 years. That means $68.17 per month.

4.2 Microsoft - Azure

At Azure cloud, services used for running experiments are Web Apps for front-end servers and SQL databases as back-ends. There is similar situation as with AWS with cost delegation. PaaS services are free of charge and customer pays only for the underlying instances which power up the web server instances.

But there is one significant difference. Azure cloud does not offer any long-term prepaid plans. All costs are Pay-as-you-Go, which means that customer pays every month what she consumed. In another words, all resources are on-demand instances. Official prices per instance and location for front-end and a back-end instances are in Tables 4.3 and 4.4. Monthly price estimates are based on 744 hours per month.

Instance Type	Data center zone and location	Term	Cost (including VAT)
D1 shared	North Europe; Dublin	1 month	$9.67
D1 shared	East US 2; N. Virginia	1 month	$9.67
D1 shared	Southeast Asia; Singapore	1 month	$9.67
D1 shared	West US; California	1 month	$9.67

Table 4.3: Azure compute instances price overview.

Instance Type	Data center zone and location	Term	Cost (including VAT)
Standard S1	North Europe; Dublin	1 month	$29.98
Standard S1	Japan East; Japan	1 month	$33.93
Standard S1	Southeast Asia; Singapore	1 month	$29.98
Standard S1	West US; California	1 month	$29.98

Table 4.4: Azure Database instances price overview.

Based on presented prices Tables 4.3 and 4.4 a total cost per 1 month is $146.6. This result from the online price list is actually more than real month payments. Total payments per month are shown in Tab. 4.5. Although instances with similar standard on-demand cost were chosen, the difference between cost at AWS and Azure is the fact of significant discount at AWS when contracting instances for the 3 years long window.

Month Type	Actual payment (including VAT)
January	$137.77
February	$127.42
March	$138.09
April	$132.40
May	$137.62
June	$132.85
July	$136.67

Table 4.5: Total payments per month for resources at MS Azure.

4.3 Infrastructure cost comparison

To get rid of the bias caused by the long-term discount, there are comparisons for both CSPs with an hourly rate for on-demand pricing plans in Tables 4.6 and 4.7 as well. These prices are not discounted and show official numbers from providers' websites. Cost structure does not include cost of outbound traffic which is also paid, but with a small volume of data transfered using CLAudit platform, this cost is negligible (less than 5$ per month). All prices are valid in September 2016 and of course are subject to change anytime.

Data center zone and location	Term	Azure D1 shared instance (per hour)	AWS t2.micro instance (per hour)
North Europe; Dublin area	1 hour	$0.013	$0.013
East US 2, N. Virginia	1 hour	$0.013	$0.012
Southeast Asia; Singapore	1 hour	$0.013	$0.015
West US; California	1 hour	$0.013	$0.015

Table 4.6: Compute on-demand instances cost overview.

Data center zone and location	Term	Azure Standard S1 instance (per hour)	AWS db.t2.micro instance (per hour)
North Europe; Dublin area	1 hour	$0.0403	$0.018
Tokyo area, Japan East	1 hour	$0.0456	$0.026
Southeast Asia; Singapore	1 hour	$0.0403	$0.026
West US; California	1 hour	$0.0403	$0.022

Table 4.7: Database on-demand instances cost overview.

For data used in this work, a 10 weeks long window with dataset of continuous measurements mentioned in Section 3.3 was used. In Tab. 4.8, there is an actual cost of the platform for this period of time. Again, it reflects the significant savings of AWS discount on a prepaid plan.

Provider	10 weeks window cost
Amazon AWS	$170.408
Microsoft Azure	$288,312

Table 4.8: Platform cost for the 10 weeks measurement window.

Chapter 5

Benchmarking methodology

5.1 Measurements terminology

I have presented the CLAudit platform which executes measurement processes, measures a cloud resource access-time latency and collects measurement data in Chapter 3. In this part I describe a methodology how to get a cloud service provider rank (CSP rank) from measured values using a new benchmarking approach. This methodology has two major parts, first is a measurements data preprocessing and then an actual CSP ranking process to get the CSP rank. I propose following terminology for further methodology explanation:

A dataset file X contains $|X|$ single RTT latency data measurements x_m. A measurement x_m is described by the following (for definition of measurement categories, see Section 3.3):

- Measured protocol $l \in L$ (e.g. TCP);
- Internet Vantage Point location $v \in V$ (e.g. Planetlab host in Prague);
- Cloud front-end resource location $f \in F$ (e.g. Web server in Dublin DC);
- Cloud back-end resource location $b \in B$ (e.g. SQL database server in Singapore DC);
- Cloud Service Provider $p \in P$ (e.g. Microsoft).

Parameters v and b are mutually exclusive, reflecting that a measurement can only be conducted between v and f or f and b (i.e. Internet VP cannot directly reach a back–end resource). Using the above terminology, an input dataset for the Latency–based benchmarking is described as follows:

$$X^{L,V,F,B,P} = \{x_m^{l,v,f,b,p}\},\ m \in [1, 2, \ldots, |X|]$$

5.2 Data preprocessing

5.2.1 Data normalization

Global CLAudit-like distributed measurement platform obtains measurements from diverse network locations worldwide. Recorded latency values largely correspond to the geographical distances between souce and destination locations, resulting in varying scale that makes any direct comparisons impossible. Thus, there is a need to normalize the values to a new standard range. Two approaches are feasible for selection of the baseline used for normalization of the raw measured values:

5.2.1.1 Via optimal-propagation-delay approximation

This approach normalizes using an approximation of the optimal signal propagation delay between the measurement source and destination. Path length is given by the Great–circle distance - the shortest distance between two points on the WGS84 ellipsoid calculated using *geod* library, measured along the surface and ignoring differences in elevation [64]. An absolute ideal propagation delay is then calculated as this round trip distance divided by speed of the light. This is a theoretical lower propagation-delay bound that cannot be superseded. [1]. Resulting path length then gets divided by the speed of light in fiber. The optimal RTT delay is twice the resulting one-way delay approximation:

$$optimal\_rtt(src, dest) = \frac{2.gcd(src, dest)}{c_{fiber}}$$

There are two downsides to normalizing using this value. First, it might prioritize longer distances to shorter ones – in case of long distances, fiber is usually employed (e.g. transoceanic submarine cables), as opposed to short distances, where copper is common and a relative number of network hops is higher. This might result in inaccuracies.

Second, based on different experiments, resulting values are poorly normalized (mainly clustered around 150%, 170%, and 190% of the ideal value), which is not suitable for further measurements processing.

5.2.1.2 Via minimum-RTT latency

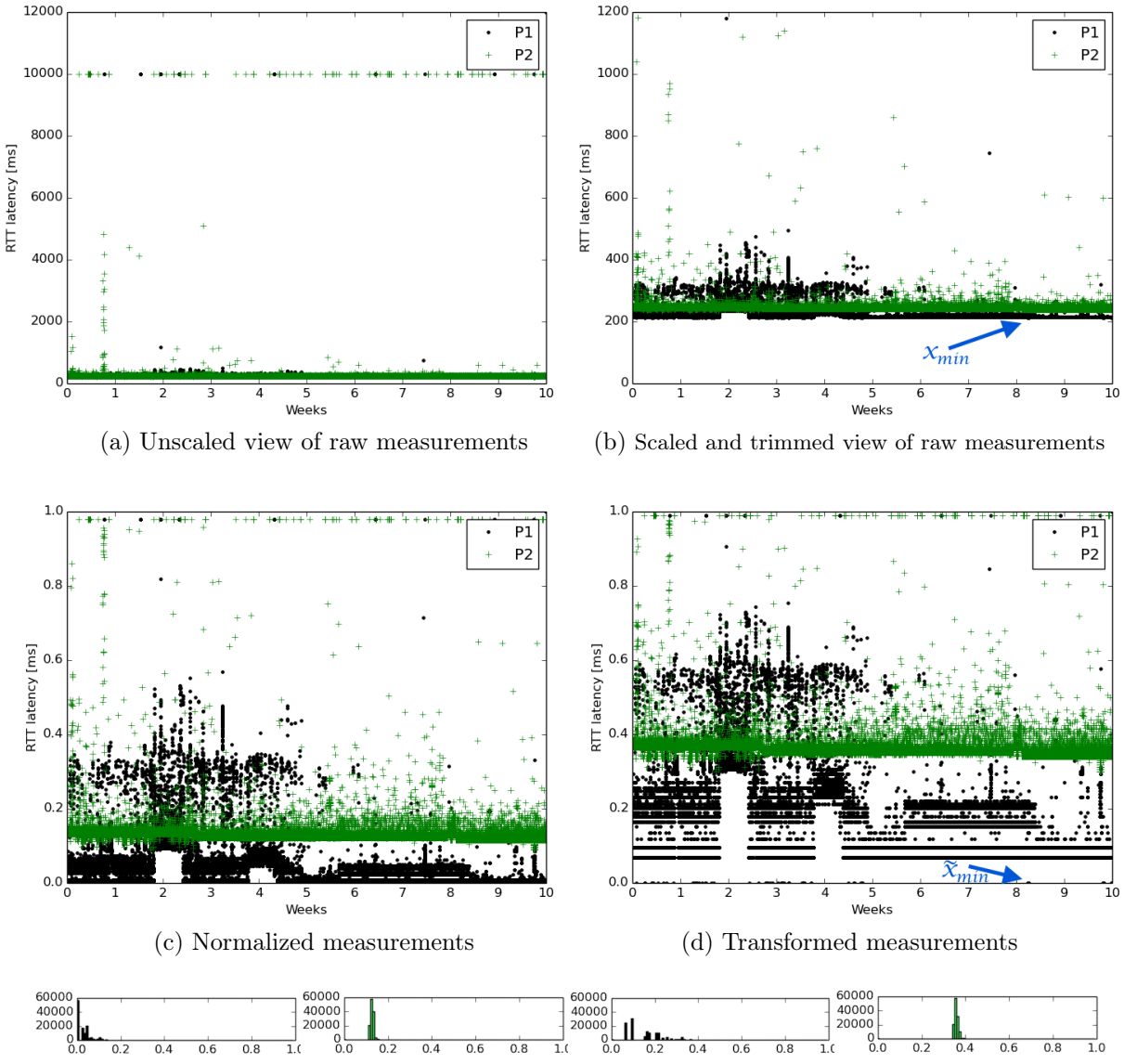
This approach starts with a step when it searches through the dataset for a minimum recorded latency value across all providers for one particular tuple (measured layer, source and destination):

$$x_{min}^{l,v,f,b} = \min_{m,p}\{x_m^{l,v,f,b,p}\},\ m \in [1, 2, \ldots, |X|]\,, p \in P$$

As this is a real measured value, it is feasible for normalization use. Minimum–latency–based normalization only suffers from clustering (addressed in 5.2.2) and, as such, is recommended for the latency benchmarking. Using the minimum latency, each dataset value (belonging to the same tuple) gets normalized using the following formula:

$$\widehat{x}_m^{l,v,f,b,p} = 1 - \frac{x_{min}^{l,v,f,b}}{x_m^{l,v,f,b,p}},\ m \in [1, 2, \ldots, |X|]$$

The measurements are now normalized to the standard $[0, 1]$ range, as shown on example in Fig. 5.1c. Subtraction from 1 is to retain the logical order, where the minimum value becomes 0 and outliers appear close to the other end of the range. The latency values are now comparable across different locations.



(a) Unscaled view of raw measurements

(b) Scaled and trimmed view of raw measurements

(c) Normalized measurements

(d) Transformed measurements

(e) P1 - Normalized

(f) P2 - Normalized

(g) P1 - Transformed

(h) P2 - Transformed

Figure 5.1: Data preprocessing process.

Latency measurements preprocessing for Latency–based benchmarking. Values get normalized using minimal recorded value x_{min} and transformed using square root, as shown in respective steps (a-d) and histograms (e-h). This example shows RTTs of SQL request–response interactions between Dublin DC front–end servers and Tokyo DC back–end server of both CSPs (P1 and P2), as recorded over 10 weeks.

5.2.2 Data transformation

Minimum–latency–based normalization results in highly–clustered measurements with different positions of central points across CSPs, locations and even protocol layers. This property is not desirable and, for the purpose of the subsequent benchmarking calculations, the values need to be spread out such that big latency values represent one end of the interval and small values the other end. The following square root transformation has the desired properties [49], as shown on transformation outcome in Fig. 5.1d. The entire preprocessing formula is as follows:

$$\widetilde{x}_m^{l,v,f,b,p} = \sqrt{1 - \frac{x_{min}^{l,v,f,b}}{x_m^{l,v,f,b,p}}},\ m \in [1, 2, \ldots, |X|]$$

Such a transformation changes the distribution of measurements. But, when applied consistently, it does not affect validity of the methodology.

5.3 CSP ranking method

The task of the benchmark is to report how well different systems perform under the given constraints. In practice, benchmarks are used to guide decisions about the most economical provisioning strategy as well as to gain insights into performance bottlenecks [38]. The specific need for benchmarking is given by concrete use cases such as financial trading or gaming systems. Whereas the former mandates very low latency with no tails, the latter allows for latency values below a certain psychological threshold. Such requirements are expressed via metrics, which are used throughout the benchmarking process.

As per the Latency–based benchmarking, application requirements are expressed using metrics, indicating needs such as low latency (e.g. interactive applications) or stable latency (e.g. media players). The benchmarking works in both forward and backward directions, i.e. a set of Internet clients may seek the optimal cloud server deployment or cloud servers need to locate clients they serve best. The same can be done for cloud front-end servers, who are the clients of cloud back-end servers.

A measurement source $s \in S$ and a destination $d \in D$ can be any of the v, f, b, but not all combinations are valid (as noted in 5.1). For simplicity, I use d is a CSP DC (i.e. f or b). The whole benchmarking process is outlined in Fig. 5.2. Raw data from the source-destination measurements are biased by its own location, when all measurement are affected by distance and routes between each other. These data can not be directly compared. As it was described in Section 5.2, the very first step is data preprocessing. Once data are normalized and transformed to a standard range, a CSP ranking process to obtain a performance rank for a provider (CSP rank) can be applied.

5.3.1 Statistical metrics

Given the transformed measurements with desired characteristics (5.2.2), the first step is a metric calculation. A metric $m \in M$ expresses latency–related application

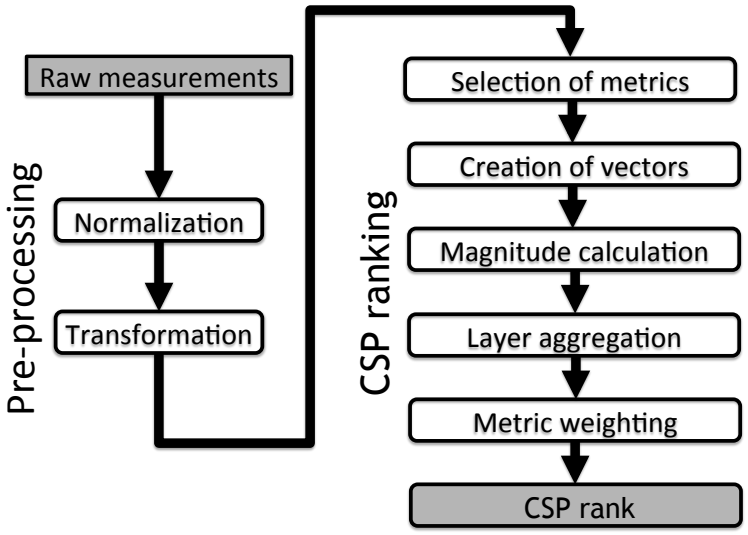


Figure 5.2: End-to-end benchmarking process overview.

requirements such as low or stable latency. Arbitrary set of metrics M' can be used that reflects application requirements well and for that I use basic metrics adopted from descriptive statistics [50]. The following examples show the Sample arithmetic mean $\overline{X} \in M'$, simple standard deviation or Coefficient of variation, which are used to express requirements related to the latency average value or latency stability:

Sample Arithmetic Mean

$$\overline{X}^{l,v,f,b,p} = \frac{1}{m} \sum_{m=1}^{|X|} \widetilde{x}_m^{l,v,f,b,p}$$

Sample Variance

$$s^2_{X\,l,v,f,b,p} = \frac{1}{m} \sum_{m=1}^{|X|} \left(\widetilde{x}_m^{l,v,f,b,p} - \overline{X}^{l,v,f,b,p} \right)^2$$

Sample standard deviation

$$s_{X\,l,v,f,b,p} = \sqrt{s^2_{X\,l,v,f,b,p}} = \sqrt{\frac{1}{m} \sum_{m=1}^{|X|} \left(\widetilde{x}_m^{l,v,f,b,p} - \overline{X}^{l,v,f,b,p} \right)^2}$$

Coefficient of variation

$$C.O.V = \frac{s_{X\,l,v,f,b,p}}{\overline{X}^{l,v,f,b,p}}$$

5.3.2 Metric vectors

Given a $|M'|$ metrics of interest, the next step is a creation of $|M'| \cdot |P| \cdot |L|$ vectors $\vec{v}$ in a $|S|$-dimensional space, where $|S|$ is a number of measurement sources s (see the example in Fig. 5.3).

The vector components are metric values, calculated per source location s_i for a provider p's destination d using a protocol l:

$$\vec{v}_{\overline{X}}^{\,l,d,p} = (\overline{X}^{\,l,d,p,s_1}, \overline{X}^{\,l,d,p,s_2}, .., \overline{X}^{\,l,d,p,s_{|S|}})$$

In this example, there is a vector consisting of $|S|$ Sample arithmetic means of latency of protocol l between $|S|$ sources and provider p's destination d. The magnitude of such resulting vector, calculated as Euclidean norm, summarizes the performance under the actual metric over all measurement source locations S:

$$\|\vec{v}_{\overline{X}}^{\,l,d,p}\| = \sqrt{(\overline{X}^{\,l,d,p,s_1})^2 + (\overline{X}^{\,l,d,p,s_2})^2 + .. + (\overline{X}^{\,l,d,p,s_{|S|}})^2}$$

The metrics, under which a CSP is performing well, have a vector magnitude close to 0. A magnitude close to 0 also occurs in the case of co-located resources, where latency is often around 1 ms (e.g. front-end and back-end in the same DC). When this happens, co-located deployment usually dominates the comparison, which is desired.

Note that an outstanding source does not influence the resulting CSP rank, as it hurts all CSPs the same way. Thus, there is no need to limit the set of considered sources only to clients of target application's interest.

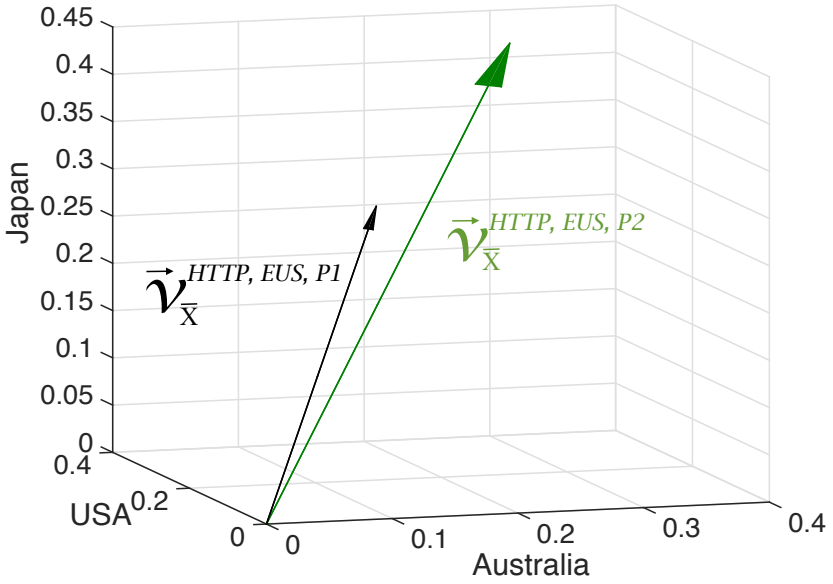


Figure 5.3: Metric vectors from CSP ranking.

Example metric vectors of providers P1 and P2 in a 3–dimensional space; Sample arithmetic means of HTTP latency measurements of similarly–located EUS front-end servers from 3 vantage points (Australia, USA, Japan) were used. The magnitude of P1's vector is smaller and, as per the methodology, benchmarking methodology favors P1 in the subsequent comparisons.

5.3.3 Protocol layer aggregation

As per the methodology, CSPs are benchmarked using requirements of an application, whose deployment is under consideration. Such an application is usually built on top of the standard network protocol stack, consisting of a hierarchy of protocol layers. Different protocol layers take turns in issuing round trips between application endpoints (as shown in Fig. 3.1). As such, every involved protocol $l \in L'$ is essential and needs to perform well. To reflect this in the ranking, an magnitude aggregation of the per–protocol–layer vectors using the following multiplication is applied, which ensures that all involved protocols perform well.

$$\|\vec{v}_m^{d,p}\| = \prod_{l \in L'} \|\vec{v}_m^{l,d,p}\|$$

This approach can be extended to prioritize some layers over others, which certain use cases may require. Also, because network protocol stack is less coherent in the cloud environment (e.g. TCP connections are terminated at TCP proxies, but application protocols at load balancers), a disagreement between layers is common [35] and penalization of misbehaving layers can be applied.

5.3.4 Metric weighting

Depending on tenant and its application needs, some metrics are of a higher priority than other. Weights are assigned from a range $[1, MAX]$. Weight is proportional to the metric's influence on application - i.e. weight 1 is assigned to a metric that stand for application requirements, which, if not satisfied, does not impact application significantly; and weight MAX is assigned to a metric having a strong impact. Given metrics of interest M', the $|M'|$ weights $w_i \in \langle 1; MAX \rangle, i \in [1, 2, \ldots, |M'|]$ are assigned. Weights heavily depend on use cases, tenant business and application. As such, the selection of number MAX and assignment of weights are based on one's needs. Weights get normalized as follows:

$$\widehat{w}_i = \frac{w_i}{\sum_{i=1}^{|M'|} w_i}$$

5.3.5 CSP rank

The resulting number f, used as a CSP rank, is then calculated as a weighted sum of vector magnitudes:

$$f^{d,p} = \widehat{w}_1 \|\vec{v}_{m1}^{d,p}\| + \widehat{w}_2 \|\vec{v}_{m2}^{d,p}\| + \cdots + \widehat{w}_{|M'|} \|\vec{v}_{m|M'|}^{d,p}\|$$

Once the value f is calculated for each provider, CSPs can be ordered by the magnitudes of their CSP ranks and compared:

$$f^{d,p_1} \leq f^{d,p_2} \leq .. \leq f^{d,p_p}$$

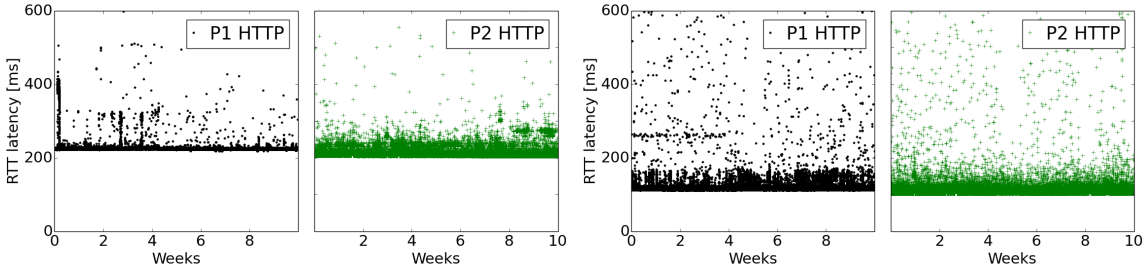
The recommended CSP is the one with the lowest f value.

Chapter 6

Benchmarking application

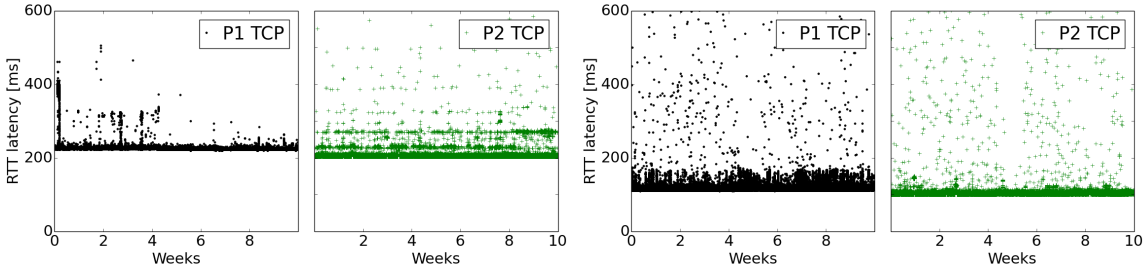
In this section, I apply the Latency–based benchmarking on a small–scale example of latency–sensitive application (approach used also in [39]).

This TCP/HTTP application is a web container serving only static web pages. A hypothetical tenant wants to migrate it from its premises and leverage the cloud environment. She considers the public CSPs P1 and P2 for the deployment on the US East Coast. The nature of the application requires low-to-moderate latency and preferably a stable latency in a sense of the following weights: $median = 5 = MAX$, $standard\ deviation = 1$, $coefficient\ of\ variation = 1$. I assume comparable provider prices.



(a) HTTP (Australia VP)

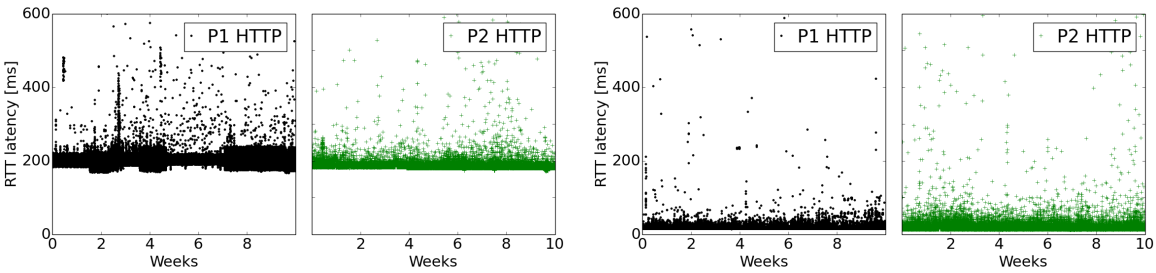
(b) HTTP (Czech VP)



(c) TCP (Australia VP)

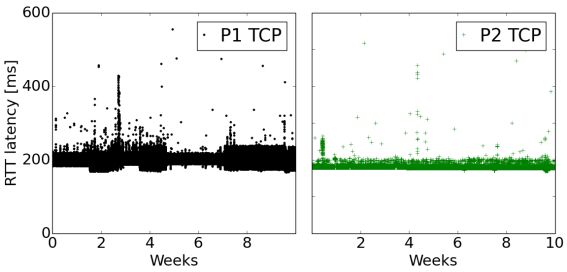
(d) TCP (Czech VP)

Figure 6.1: Actual latencies to EUS front-end servers as observed by the four VPs.

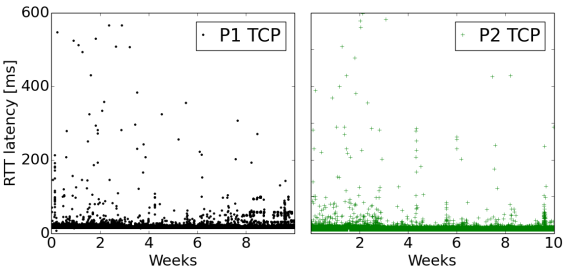


(e) HTTP (Japan VP)

(f) HTTP (US VP)



(g) TCP (Japan VP)



(h) TCP (US VP)

Figure 6.1: Actual latencies to EUS front-end servers as observed by the four VPs.

Provider P2's DC has a lower median latency observed by all VPs. Provider P1's DC on the other hand provides a more stable latency in some cases. These observations are quantified throughout various stages of benchmarking process.

6.1 CSP rank results

Given the transformed normalized measurements, benchmarking can now proceed according to the steps described in Chapter 5. First, a metric vector is created for median latency of HTTP protocol for CSP P1:

$$\vec{v}_{med}^{HTTP,EUS,P1} =$$

$$= (med^{HTTP,EUS,AU}, med^{HTTP,EUS,CZ}, med^{HTTP,EUS,JP}, med^{HTTP,EUS,CZ}) =$$

$$= (0.306, 0.368, 0.421, 0.433)$$

$$\|\vec{v}_{med}^{HTTP,EUS,P1}\| = \sqrt{(0.306)^2, (0.368)^2, (0.421)^2, (0.433)^2}$$

$$\|\vec{v}_{med}^{HTTP,EUS,P1}\| = 0.771$$

Analogically, such vector is also created for CSP P2 with the resulting magnitude:

$$\|\vec{v}_{med}^{HTTP,EUS,P2}\| = 0.518$$

From results, P2 has around 25% better median metric vector magnitude, i.e. it has lower latency. This particular comparison can visualized as two points in 4-dimensional vector space, analogically to Fig. 5.3. The same steps are now done for TCP protocol, with the resulting magnitudes:

$$\|\vec{v}_{med}^{TCP,EUS,P1}\| = 0.997$$
$$\|\vec{v}_{med}^{TCP,EUS,P2}\| = 0.726$$

Again, a notable difference in TCP latencies between the providers can be observed, suggesting that provider P2 provides lower latency across the network protocol stack. Next, I calculate vector magnitudes for a coefficient of variation:

$$\|\vec{v}_{cov}^{HTTP,EUS,P1}\| = 0.302, \|\vec{v}_{cov}^{TCP,EUS,P1}\| = 0.230$$
$$\|\vec{v}_{cov}^{HTTP,EUS,P2}\| = 0.774, \|\vec{v}_{cov}^{TCP,EUS,P2}\| = 0.836$$

In the case of coefficient of variation for both TCP and HTTP, P1 scored notably better, i.e. it provides a more stable latency. Next, standard deviation vector magnitudes is calculated – another measure of latency stability:

$$\|\vec{v}_{std}^{HTTP,EUS,P1}\| = 0.122, \|\vec{v}_{std}^{TCP,EUS,P1}\| = 0.092$$
$$\|\vec{v}_{std}^{HTTP,EUS,P2}\| = 0.195, \|\vec{v}_{std}^{TCP,EUS,P2}\| = 0.130$$

Once again, because of P1's better stability, its standard–deviation vector magnitudes are lower than P2's. Tab. 6.1 shows magnitudes of the selected metric vectors.

P	L	med	std	cov
P1	HTTP	0.771	0.122	0.302
	TCP	0.997	0.092	0.230
P2	HTTP	0.518	0.195	0.774
	TCP	0.726	0.130	0.836

Table 6.1: Calculated metric vector magnitudes $\|\vec{v}\|$.

Next, I aggregate the layers via multiplication (Section 5.3.3) and calculate the weighted sum using the following weights:

$$\widehat{w}_1 = \frac{5}{7}, \widehat{w}_2 = \frac{1}{7}, \widehat{w}_3 = \frac{1}{7}$$

The weights reflect the primary need for low latency and a secondary need for stable latency. Weights are plugged into the weighted sum formula:

$$f^{EUS,p} = \widehat{w}_1\|\vec{v}_{med}^{EUS,p}\| + \widehat{w}_2\|\vec{v}_{std}^{EUS,p}\| + \widehat{w}_3\|\vec{v}_{cov}^{EUS,p}\|$$

That brings resulting marks: $f^{EUS,P1} = 0.56$ and $f^{EUS,P2} = 0.36$.

As $0.56 \geq 0.36$, CSP P2 is a recommended provider for hosting web–based application front–end in East USA region.

Fig. 6.1 confirms that the tenant gets a better service performance with CSP P2. Importantly, CSP P2 had a sufficiently lower latency and thus scored better overall, despite the more stable–latency environment at CSP P1. In Fig. 6.1, a lower median latency at P2 can be seen through order of tens milliseconds of difference at all VPs. Higher latency–stability at P1 is caused mainly by Australian and Czech VP's observations at both application and transport layers. In contrast, Japan VP experienced a higher–stability with P2. US VP's observations were least significant, owing to a physical proximity of this VP to the EUS DCs.

6.2 CSP rank - error function

The amount of measurements needed for accurate CSP benchmarking depends on the particular CSP's stability. Fig. 6.2 shows changing error in CSP rank (from the case study in this Chapter 6) with growing amount of the measurements. The error was $\leq 4.1\%$ at 3 weeks and $\leq 2.2\%$ at 6 weeks of measurements. In general case, stabilization may not arise even with a greater amount of measurements – due to changing CSP and network conditions. On the other hand, the long–term measurements of major public CSPs indicate that an amount of measurements smaller than 3 weeks is insufficient due to week–of–month and day–of–week distortions.

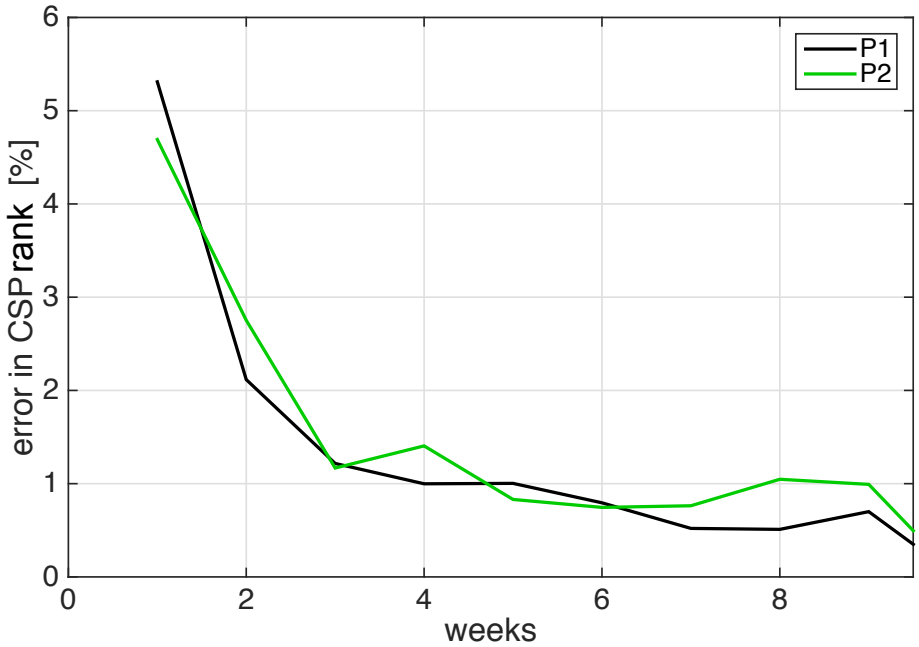


Figure 6.2: Decreasing error with growing amount of measurements.

Error function of benchmarked–CSP mark with growing dataset. The case study indicates that at least 3–6 weeks of measurements are recommended for the Latency–based benchmarking.

6.3 Measurements summary

To give an overview of the major public CSP–latency behavior, Tables 6.2 and 6.3 show a summary of computed magnitudes of descriptive–statistics–based metric vectors. In Tab. 6.2, it can be seen that the largest differences in both mean and median front–end latency between the providers are observed at US data centers. The highest latency variability and deviations were observed at Singapore DC, again with a big relative difference between providers (over 50% in the case of TCP latency variance). P2 also has an excess variance in HTTP latency at Virginia DC.

front–end summary (vector magnitudes)			front–end DC location							
			California		Dublin		Singapore		Virginia	
			P1	P2	P1	P2	P1	P2	P1	P2
Metric	Mean	HTTP	.823	.556	.544	.684	.689	.634	.780	.562
		TCP	.853	.514	.545	.590	.718	.596	.998	.736
	Median	HTTP	.809	.531	.511	.668	.693	.604	.771	.518
		TCP	.842	.491	.564	.585	.723	.571	.998	.727
	Variance	HTTP	.009	.014	.013	.008	.023	.031	.008	.024
		TCP	.009	.013	.014	.008	.016	.033	.005	.011
	Std. deviation	HTTP	.127	.155	.148	.125	.201	.230	.123	.195
		TCP	.126	.143	.152	.124	.171	.239	.093	.131
	COV	HTTP	.305	.754	.587	.440	1.810	.810	.303	.773
		TCP	.295	.789	.625	.524	.628	.902	.231	.836

Table 6.2: Summary of vector magnitudes for front-end resources.

Back–end results (Tab. 6.3) are heavily influenced by the DC and inter–DC network design. Big differences in latency deviation and relative variance were observed at Dublin DC (TCP and SQL latency) and Singapore DC (TCP latency). At California DCs of both providers, there was a big disagreement between SQL and TCP average latency and also the latency variability.

back–end summary (vector magnitudes)			back–end DC location							
			California		Dublin		Singapore		Tokyo	
			P1	P2	P1	P2	P1	P2	P1	P2
Metric	Mean	SQL	1.130	1.015	1.106	1.272	.422	.677	.447	.513
		TCP	.653	.768	1.114	1.301	.858	.986	.488	.640
	Median	SQL	1.136	1.018	1.107	1.258	.347	.680	.429	.534
		TCP	.551	.997	1.114	1.299	.797	1.149	.450	.636
	Variance	SQL	.014	.012	.033	.003	.039	.049	.011	.023
		TCP	.153	.169	.035	.007	.026	.154	.013	.020
	Std. deviation	SQL	.140	.140	.233	.066	.252	.242	.136	.167
		TCP	.414	.447	.245	.097	.205	.408	.153	.174
	COV	SQL	.613	1.613	.880	.340	4.522	3.337	.751	.738
		TCP	1.249	1.016	.893	.418	.706	.781	.754	.567

Table 6.3: Summary of vector magnitudes for back-end resources.

The magnitudes, listed in these Tab. 6.2 and Tab. 6.3, allow to assign arbitrary weights to metrics and observe changing benchmarking results. Another use is to benchmark an application distributed across DCs with multiple front–ends or back–ends. And locate clients that such a deployment serves best.

6.4 Measurements and business analysis impact

In the previous sections of this chapter I have presented results from the benchmarking method evaluating the performance of each provider. Using that, an arbitrary subset of end-to-end measurements can be chosen to compare latency performance for each provider.

The CLAudit based, long term measurements can be also used for a real overview if a CSP satisfies quality of service which he promises to a customer- in a SLA (Service Level Agreement) document for example. In the chapter 2.3, I have described three use cases which require different parameters for a particular business case. For a high-frequency trading, there is a need for low values of network latency with as little outliers as possible, for an online-gaming use case it would be a standard deviation metric the small as possible and for an online map provider, an information about average cases using a median metric, among others. Values for these performance thresholds always depend on the particular situation, on the application which is under consideration and other parameters important for the business.

Let us consider CLAudit latency measurements as values for a real communication which was perceived by a client. In Fig. 6.3, there is a communication on HTTP layer between an Australian VP and a front-end application web server located in the Singapore DC for two providers, P1 and P2. For a simplicity of this example, I consider a latency threshold which would be enforced by the use case. Overcoming this threshold value might result in a trade loss for the trading or undesired disadvantage for a game player, for example.

First, in Fig. 6.3 and Fig. 6.4, there are measurement results in a full scale as well as in a scaled view to show the major and interesting part of the latency measurements on this path. I choose this particular example, because the latency time series have a similar values, it is thus not easy to differentiate between these two. Therefore an analysis for a threshold can give interesting results.

For this measurement and the example, 100ms threshold was chosen. To have a complete picture and an analysis, two other thresholds, 150ms and 200ms were added. These values were chosen to encompass the vast majority of the measurement and to study differences on these consecutive levels. As mentioned, for each use-case or an app, the threshold is different. Thresholds are highlighted in measurements in Fig. 6.5 for 100ms, in Fig. 6.6 for 150ms and in Fig. 6.7 for 200ms.

Let us consider a condition which could appear in a SLA agreement between the customer and a provider about a level of quality of the service. This condition requires that a 95% of the network latency communication must be below the requested threshold value (100ms, 150ms and 200ms accordingly). In Tab. 6.4, I calculated a percentage which represents how many measurements satisfies this condition and to which extent was the threshold violated in the measurement window. Any other metric or parameter which can be calculated can be used for similar comparison.

For provider P1, results exceed the 95% condition and it satisfies the requirement. In case of provider P2, it does not meet the criteria for the first two threshold values. Even more, the differences are around 5-15%, which is a significant agreement violation. To this end, important fact is that data are of a long-term nature. It is not just a short-term sample, data are collected in uniform manner over 10 weeks.

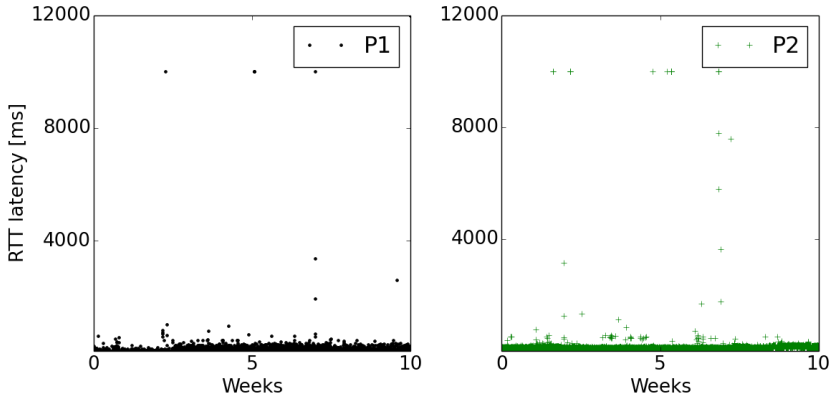


Figure 6.3: An Australian client connecting to Singapore front-end server on HTTP layer.

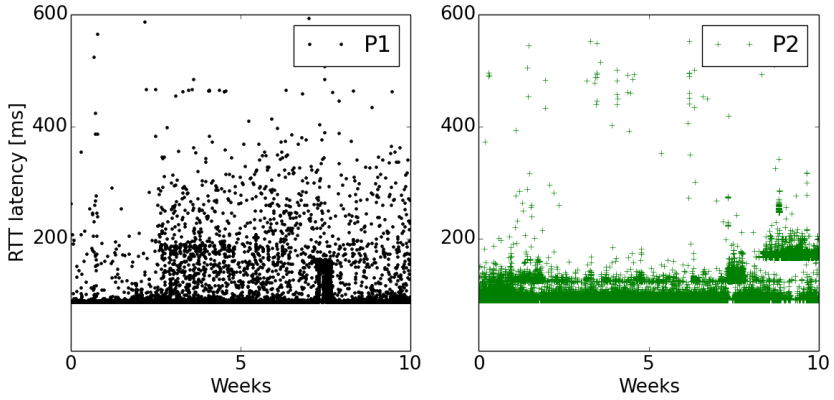


Figure 6.4: Scaled plots of measurements from Fig. 6.3.

Presented values might have a significant impact on the customer business, as it was already mentioned - a trade loss, a bad user experience etc. As such, this might have an influence on the customer-provider relationship and other business aspects. Customer can ask for some sort of price discounts or other service usage negotiation etc. For example, Amazon gives a *Service Credit* points to customer when SLA is not met. This credit is used as a compensation when billing the service [65].

Provider	threshold 100ms	threshold 150ms	threshold 200ms
P1	0.96273	0.97113	0.99642
P2	0.79710	0.89606	0.99696

Table 6.4: Percentage of measurements which falls under the threshold.

This was just a short insight how such long term measurements might help in a business, using a simple performance threshold. Similar analysis might use advanced metrics, as well as more complex data (different layers) for more thorough evaluation.

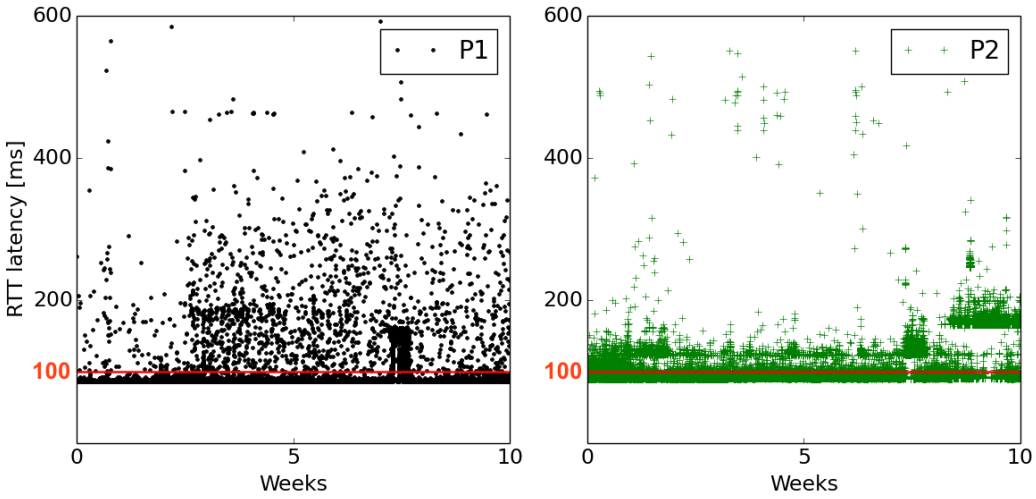


Figure 6.5: Threshold at 100ms highlighted.

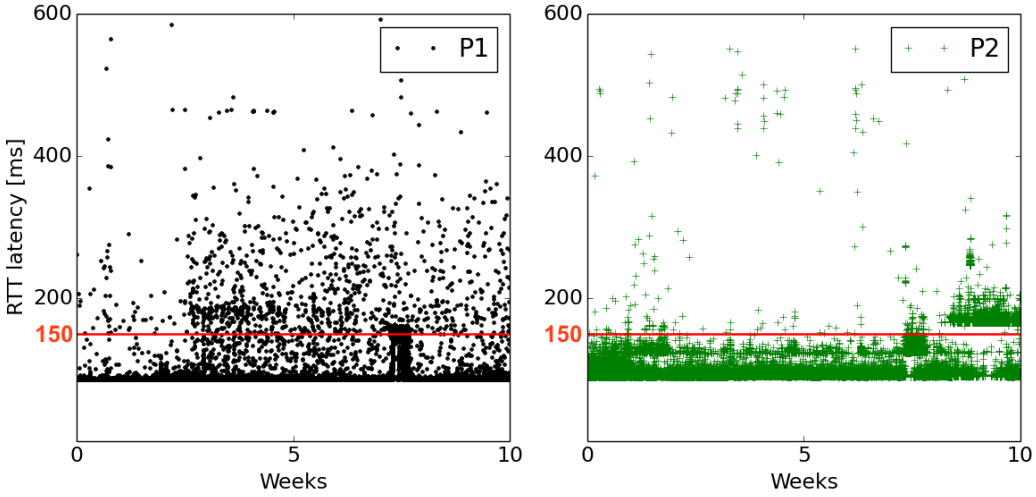


Figure 6.6: Threshold at 150ms highlighted.

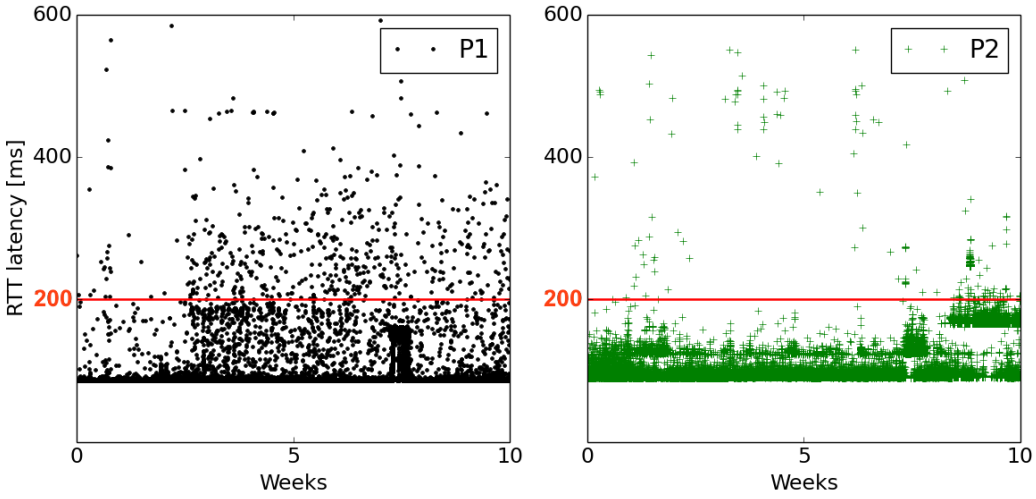


Figure 6.7: Threshold at 200ms highlighted.

Chapter 7

Conclusion

The aim of this work was to measure network latency to global cloud resources of different providers and to propose a method that evaluates and compares their performance - network based cloud benchmarking. As current trend shows, network latency is getting more and more important. A customer who is seeking the best fit cloud infrastructure for his applications remains indecisive due to lack of information from providers themselves or by lack of his own general measurements. For the purpose of this work I have extended an existing measurement infrastructure to measure the two largest cloud providers (Amazon and Microsoft) and proposed an innovative benchmarking methodology, resulting in a cloud provider performance rank.

In the more practical part, I have completely redesigned an existing CLAudit platform to allow measurements of the two cloud providers. I have chosen similar data center locations for both clouds and then deployed the infrastructure to the Amazon AWS cloud to run the measurements as well. Existing solution at Microsoft Azure cloud was updated for this purpose accordingly. The whole new testbed allows fully autonomous, continuous and provider fair latency-based cloud measurements of both providers. With the structural changes like a usage of configuration file, code improvements and measurements tuning, the platform is extended and prepared even for measurements of other cloud providers in the future.

For performance comparison, I present a benchmarking methodology, which has two parts. Firstly, data preprocessing normalizes and transforms raw data from word-wide measurements to a standard range, suitable for further processing. The subsequent CSP ranking part contains multiple steps and different metrics that can be applied to tailor the evaluation according to the use-case or the application under consideration. With such parametrization one metric can be more stressed with a stronger weight than another. All these steps are applied equally on data of both providers, so the final rank is a fair evaluation.

I have applied the introduced theory on the real data from the CLAudit measurement platform to demonstrate the applicability of such approach. It reveals interesting results for different metrics and various data center locations. These findings can be used for provider selection using diverse application requirements, provider performance estimations without any actual test deployments, traffic load balancing or identifying best-fit data center locations for a global client base, front-end or

back-end resources and vice versa. Also, I present a cost overview of the platform and an insight to a cloud provider pricing plans. Surprisingly, this approach discovers significant differences, even though cloud resources at similar power level were chosen.

This study clearly shows that selection of the best fitting cloud service provider is not straightforward. Detailed application quality-of-service requirements must be known to obtain a clear picture, as different services outperform others in different aspects or locations. An advantage of this work is that it is a demonstration of introduced theory on real data and measurements, as opposed to simulation-based approaches with assumptions.

Another strong contribution of this work is the fact, that the research is already accredited in the cloud computing community. The benchmarking methodology and the work results from this Master Thesis were used for a research paper named *Latency–based Benchmarking of Cloud Service Providers* authors *Vojtech Uhlir, Ondrej Tomanek and Lukas Kencl*. This paper was accepted to internationally well recognized conference - *9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2016), Shanghai, China*. Myself, Vojtech Uhlir, I had the honor to attend the conference in person and present the paper's findings to the conference auditorium.

Clearly, improvement of the methodology is possible - for example by integrating other measurable parameters such as network throughput or a service cost. Including the pricing plans to the algorithm could reveal an interesting cost/performance ratio. But the cloud cost structure is a large and complex issue, with a high number of different cloud instances, services or payments plans. Because of that I did not include this parameter to the calculations in this work, as it requires a more thorough research and providers' offerings evaluations. More cloud providers can be included to the platform as well, to cover even more of the market.

Furthermore, despite the practical application demonstrated, to fully confirm findings in this work a comparative test is needed. Since there are no similar studies yet, there are no data to measure improvement or a result precision. All these are open opportunities for the future or an extension of this work. CLAudit measurements data are published and freely accessible online at *http://claudit.feld.cvut.cz* and can be used by the research community. This also leaves a space for other research groups to use this data for further investigation and processing of the measured values.

Accomplishments

- Vojtech Uhlir, Ondrej Tomanek, and Lukas Kencl. Latency-based benchmarking of Cloud Service Providers. In *Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing - UCC2016*. Association for Computing Machinery (ACM), 2016.
- Ondrej Tomanek, Pavol Mulinka, Vojtech Uhlir and Lukas Kencl. Cloud Performance Analysis and Improvement. Project report for the *Grant Agency of the Czech Technical University in Prague, grant No. SGS15/153/OHK3/2T/13*. Czech Technical University in Prague, 2017.

Bibliography

[1] Ondrej Tomanek and Lukas Kencl. Claudit: Planetary-Scale Cloud Latency Auditing Platform. *2nd IEEE International Conference on Cloud Networking (IEEE CloudNet), San Francisco, CA, USA*, 2013.

[2] Gartner. By 2020, a corporate "no-cloud" policy will be as rare as a "no-internet" policy is today. [online], http://www.gartner.com/newsroom/id/3354117, 2016.

[3] Gartner. Cloud Computing Definition. [online], http://www.gartner.com/it-glossary/cloud-computing/, 2016.

[4] National Institute of Standards and Technology, Information Technology Laboratory. Cloud Computing. [online], https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf, 2016.

[5] Datacenterfrontier. Scaling Up: Google Building Four-Story Data Centers. [online], http://datacenterfrontier.com/google-building-four-story-data-centers/, 2016.

[6] DPR Construction. Facebook Sweden Data Center. [online], http://www.dpr.com/projects/sweden-data-center, 2016.

[7] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM*, 2003.

[8] Gartner. Multitenancy. [online], http://www.gartner.com/it-glossary/multitenancy/, 2016.

[9] Microsoft. When to move biztalk to Azure IaaS. [online], http://social.technet.microsoft.com/wiki/contents/articles/35800.when-to-move-biztalk-to-azure-iaas.aspx, 2016.

[10] Amazon. Amazon VPC. [online], https://aws.amazon.com/vpc/, 2016.

[11] Rackspace. Managed cloud. [online], https://www.rackspace.com/cloud/private, 2016.

[12] Barb Darrow. Amazon and microsoft are running one and two in two-cloud race, 2016.

[13] David Ramel. Microsoft no. 2 behind amazon in cloud market share, 2016.

[14] Amazon. Netflix Case Study. [online], https://aws.amazon.com/solutions/case-studies/netflix/, 2015.

[15] Alex Casalboni. Aws re:invent 2015 nettflix and aws, 2015.

[16] Ondrej Tomanek, Pavol Mulinka, and Lukas Kencl. Multidimensional Cloud Latency Monitoring and Evaluation. *Computer Networks*, 2016.

[17] Luiz Andre Barroso. Warehouse-scale Computing: Entering the Teenage Decade. *ACM SIGARCH, 2011.*

[18] Luiz André Barroso, Jimmy Clidaras, and Urscc Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture, Morgan & Claypool Publishers, 2013.*

[19] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *ACM international conference on Multimedia, 2011.*

[20] Mohan Dhawan, Justin Samuel, Renata Teixeira, Christian Kreibich, Mark Allman, Nicholas Weaver, and Vern Paxson. Fathom: A Browser-based Network Measurement Platform. In *ACM IMC*, 2012.

[21] Ondrej Vondrous, Peter Macejko, and Zbynek Kocur. FlowPing-The New Tool for Throughput and Stress Testing. *Advances in Electrical and Electronic Engineering*, 13(5):516, 2015.

[22] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using Planetlab for network research: myths, realities, and best practices. In *ACM SIGOPS*, 2006.

[23] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the ACM IMC*, pages 267–280, 2010.

[24] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The Nature of Datacenter Traffic: Measurements & Analysis. In *ACM IMC*, 2009.

[25] Abhinav Pathak, Y Angela Wang, Cheng Huang, Albert Greenberg, Y Charlie Hu, Randy Kern, Jin Li, and Keith W Ross. Measuring and Evaluating TCP Splitting for Cloud Services. In *PAM, 2010, Springer.*

[26] Y Angela Wang, Cheng Huang, Jin Li, and Keith W Ross. Estimating the Performance of Hypothetical Cloud Service Deployments: A Measurement-Based Approach. In *INFOCOM, IEEE, 2011.*

[27] Atef Abdelkefi, Yuming Jiang, Bjarne Emil Helvik, Gergely Biczok, and Alexandru Calu. Assessing The Service Quality of an Internet Path Through End-to-end Measurement. *Elsevier Computer Networks, 2014*, 2014.

[28] Marik Marshak and Hanoch Levy. Evaluating Web User Perceived Latency Using Server Side Measurements. *Elsevier Computer Communications, 2003.*

[29] Robert Minnear. *Latency: The Achilles Heel of Cloud Computing*, 2011. Cloud Computing Journal.

[30] Harsha V Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring, and Arun Venkataramani. A Structural Approach to Latency Prediction. In *SIGCOMM, ACM, 2006.*

[31] Krishna P Gummadi, Stefan Saroiu, and Steven D Gribble. King: Estimating Latency Between Arbitrary Internet End Hosts. In *ACM SIGCOMM Workshop on Internet measurement*, 2002.

[32] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. Analyzing the Performance of an Anycast CDN. In *ACM IMC*, 2015.

[33] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. Are We One Hop Away from a Better Internet? In *ACM IMC*, 2015.

[34] K.L Johnson, J.F Carr, M.S Day, and M.F Kaashoek. The Measured Performance of Content Distribution Networks. *Elsevier Computer Communications, 2001.*

[35] 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. In *Passive and Active Measurement*. Springer, 2014.

[36] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. Above The Clouds: A Berkeley View of Cloud Computing. *UCB/EECS*, 2009.

[37] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards Predictable Datacenter Networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.

[38] Enno Folkerts, Alexander Alexandrov, Kai Sachs, Alexandru Iosup, Volker Markl, and Cafer Tosun. Benchmarking in the Cloud: What it Should, Can, and Cannot be. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 173–188. Springer, 2012.

[39] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. *In Proc. of Conference on Internet Measurement (IMC)*, 2010.

[40] Mohan Baruwal Chhetri, Sergei Chichin, Quoc Bao Vo, and Ryszard Kowalczyk. Smart CloudBench - Automated Performance Benchmarking of the Cloud. *Proc. of Sixth International Conference on Cloud Computing (CLOUD2013)*, 2013.

[41] Alexander Lenk, Michael Menzel, Johannes Lipsky, Stefan Tai, and Philipp Offermann. What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-Service Offerings. In *Cloud Computing (CLOUD), 2011 IEEE International Conference*, pages 484–491. IEEE, 2011.

[42] Amazon. AWS Service Health Dashboard. [online], http://status.aws.amazon.com/, 2016. http://status.aws.amazon.com/.

[43] Microsoft. Microsoft Azure Status. [online], http://status.azure.com, 2016.

[44] CloudHarmony Inc. Cloud Harmony. [online], http://cloudharmony.com/, 2016.

[45] Michael Menzel and Rajiv Ranjan. CloudGenius: Decision Support for Web Server Cloud Migration. In *Proceedings of the 21st international conference on World Wide Web*, pages 979–988. ACM, 2012.

[46] Mohan Baruwal Chhetri, Quoc Bao Vo, Ryszard Kowalczyk, and Cam Lan Do. Cloud Broker: Helping You Buy Better. In *International Conference on Web Information Systems Engineering*. Springer, 2011.

[47] Ang Li, Xuanran Zong, Srikanth Kandula, Xiaowei Yang, and Ming Zhang. CloudProphet: Towards Application Performance Prediction in Cloud. *ACM SIGCOMM Computer Communication Review*, 41, 2011.

[48] Mosharaf Chowdhury, Rachit Agarwal, Vyas Sekar, and Ion Stoica. A Longitudinal and Cross-Dataset Study of Internet Latency and Path Stability. Technical report, 2014.

[49] J Osborne. Notes on the Use of Data Transformations. *Practical Assessment, Research and Evaluation*, 9(1):42–50, 2005.

[50] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[51] D. Schneider. Financial trading at the speed of light. http://spectrum.ieee.org/computing/it/financial-trading-at-the-speed-of-light, 2011 23 Sep.

[52] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. *Proceedings of the 19th ACM international conference on Multimedia*, 2011.

[53] Ondrej Tomanek, Pavol Mulinka, Vojtech Uhlir, and Lukas Kencl. CLAudit. [online], http://claudit.feld.cvut.cz, 2016.

[54] Pavol Mulinka and Lukas Kencl. Learning from cloud latency measurements. *3rd International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA) at IEEE International Conference on Communications (ICC15 - Workshops05), London, UK*, 2015.

[55] Amazon. Amazon RDS. [online], https://aws.amazon.com/rds/details/, 2016.

[56] Microsoft Azure. SQL Databases, 2016. https://azure.microsoft.com/en-us/pricing/details/sql-database/.

[57] Azure. Azure Regions. [online], https://azure.microsoft.com/en-us/regions/.

[58] DataCenterResearch. Amazon data center map. [online], http://www.datacentermap.com/cloud/amazon-ec2.html, 2016.

[59] Amazon. Amazon Global Infrastructure. [online], https://aws.amazon.com/about-aws/global-infrastructure/, 2016.

[60] YAML.org. Yaml language. [online], yaml.org, 2016.

[61] Amazon. Simple monthly calculator. [online], https://calculator.s3.amazonaws.com/index.html, 2016.

[62] Azure. Pricing calculator. [online], https://azure.microsoft.com/en-us/pricing/calculator/, 2016.

[63] Amazon. Amazon EC2 Service. [online], https://aws.amazon.com/ec2/, 2016.

[64] Gerald Evenden. *Proj. 4–Cartographic Projections Library*, 1990. Source code and documentation available from trac. osgeo. org/proj.

[65] Amazon. Amazon EC2 Service Level Agreement. [online], https://aws.amazon.com/ec2/sla/, 2016.

Appendix A

List of abbreviations

- **AUS** - location Australia
- **CSP** - Cloud Service Provider
- **CZ** - location Czech Republic
- **DC** - Data Center
- **DUB** - location Dublin
- **EUS** - location East of United States of America
- **EBS** - Elastic Block Store
- **HTTP** - Hyper Text Transport Protocol
- **IaaS** - Infrastructure As A Service
- **JP** - location Japan
- **PaaS** - Platform As A Service
- **RTT** - Round Trip Time
- **SaaS** - Software As A Service
- **SING** - location Singapore
- **SLA** - Service Level Agreement
- **SQL** - Structured Query Language
- **TOK** - location Tokyo
- **TCP** - Transport Control Protocol
- **US** - location United States of America
- **VP** - Vantage Point
- **WUS** - location West of United States of America