

SCALABILITY OF VIDEO TRANSCODING ON A CLOUD COMPUTING PLATFORM

Mohamed Amini Alaoui
maminial@abo.fi - 39123

Master of Science Thesis
Supervisor: Dr. Sebastien Lafond
Embedded Systems Laboratory
Faculty of Science and Engineering
Åbo Akademi University
March 2019

ABSTRACT

Millions of videos are being shared between users every day; these videos have different formats and codecs and are being played in different platforms with different apps. The need to convert these videos into a format that could be read in every platform and could be used by any user is the key, and this is where transcoding comes into play. The operation of video transcoding demands a large amount of computing power, thus a source of computing resources which are ready at all times to be used, scalable, and reliable is necessary to achieve the goal. Therefore, cloud computing is worth examining as a prominent candidate to be considered as the platform for video transcoding.

This thesis work includes a presentation of cloud computing and its features, explaining the advantages and disadvantages of using it, as well as elaborating on the topic of the overheads related to this technology. Related work has been taken into consideration by presenting the outcomes of a study concerning the management of performance overhead of virtual machines in cloud computing. The experiment and its setup are explained in detail, and the result of the thesis work is presented at the end with figures showing the execution time of each step used of the script. The script is based on FFMPEG libraries for the transcoding, and the open-stack based cloud computing platform chosen for this experiment is cPouta from SCS “Center for Science”. Further discussions and future work are given at the last chapter of the thesis work.

Keywords: cPouta, Cloud computing, FFMPEG, Transcoding, Virtual Machine.

CONTENTS

ABSTRACT	2
LIST OF FIGURES	5
1 INTRODUCTION	7
Thesis Structure	9
2 CLOUD COMPUTING	10
2.1 Overview	10
2.2- Public, Private, and hybrid Clouds	11
2.2.1- Public Clouds	12
2.2.2- Private Clouds	12
2.2.3- Hybrid Clouds	13
2.3- Service models	14
2.3.1- XaaS – Everything-as-a-Service	15
2.3.2- IaaS – Infrastructure-as-a-Service	15
2.3.3- PaaS – Platform-as-a-Service	17
2.3.4- SaaS – Software-as-a-Service	18
2.4 Pros and Cons of Cloud Computing	18
2.5- AMDAHL’S LAW	20
3 RELATED WORK	22
3.1 Managing Performance Overhead of Virtual Machines in Cloud Computing	22
4 SETING UP THE ENVIRONMENT	26
4.1 The environment	26
4.1.1 Virtual Machines:	26
5 CPOUTA	29
5-1 OpenStack	29
5.1.1 Nova Compute	30
5.2 Instances creation	32
5.2.1 Setting up cPouta instance	32

5.2.2	SSH protocol connection	32
2.3	Create an instance	36
5.2	Instance access	39
5.2.1	Accessing the master instance	39
5.2.2	Accessing the other instances	41
6	EXPERIMENT AND RESULT	44
6.1	The script	44
6.1.1	FFMPEG	44
6.2	Video conversion	44
6.2.1	Running the script	44
6.3	The result of the experiment	50
6.3.1	Results	50
6.3.1.1	Splitting the video	50
6.3.1.2	Copying the chunks	51
6.3.1.3	Transcoding	52
6.3.1.4	Copy back	53
6.3.1.5	Merge	54
6.3.1.6	The total time	55
7	CONCLUSION AND FUTURE WORK	56
	BIBLIOGRAPHY	58
	ANNEX 1	61

LIST OF FIGURES

Figure 1: The general view of the destined consumer of the public clouds.....	12
Figure 2: The general view of the private clouds	13
Figure 3: The general view of the hybrid clouds	14
Figure 4: Cloud computing service models arranged as layers in a stack	14
Figure 5: IaaS providers most known and generating most interest	16
Figure 6: PaaS providers in Public Cloud	17
Figure 7: Localized application execution (top left) compared to an application executed on the cloud (top right and bottom). NIC: network interface controller.....	21
Figure 8: Typical shared architecture in the datacenter	24
Figure 9: Classification of causes and mitigation techniques of VM performance overhead from the viewpoint of IaaS cloud hierarchy	25
Figure 10: The services provided by Microsoft Azure	27
Figure 11: Depiction of the OpenStack framework.....	29
Figure 12: The architecture of Nova.....	31
Figure 13: SSH connection protocol.....	32
Figure 14: Create Key Pair button	33
Figure 15: Create Key Pair button details.....	33
Figure 16: Create Key Pair result	34
Figure 17: The SSH folder structure.....	34
Figure 18: Security group button.....	35
Figure 19: Security group result	35
Figure 20: IP address having access to the instances.....	36
Figure 21: The instance details	36
Figure 22: creation of an instance from command line	38
Figure 23: nova list command	39
Figure 24: security addition to the key	40
Figure 25: adding password to the key	41

Figure 26: access to the master instance.....	42
Figure 27: The Config file commands.....	43
Figure 28: The limit summery of the instances	45
Figure 29: The specs of the instances	45
Figure 30: The steps performed by the script	48
Figure 31: Details on the video used for the experiment.....	49
Figure 32: The execution time of the splitting script.....	50
Figure 33: The execution time of the copying script.....	51
Figure 34: The execution time of the transcode script.....	52
Figure 35: The execution time of the copying script.....	53
Figure 36: The execution time of the merge script.....	54
Figure 37: The total execution time of the script.....	55

1 INTRODUCTION

With a smartphone today, with thousands of times more processing power than the first rocket landed on the moon, is not a coincidence. Digital content usage is growing exponentially in the world of connected everything, which requires one – as a smartphone user – to at least have a decent level of computing power.

Most of the computational power needed is going to be used for video operations. According to a Cisco report [1], videos will constitute a large amount of the digital content shared on the internet, with an estimated 82% by 2020. The emerging trend is a global phenomenon, billions now have access to the internet and the number will double in the next few years. Everyone now can make, edit and share videos in just a few steps. The user has now access to more social media applications than at any time before, for example, short-to-long time video content in “youtube”[2] and “dailymotion”[3], short-to-medium in “instagram”[4], short in time, small in size and high in quality for “snapchat”[5]. Many more platforms are around the corner and will appear any time soon.

Video delivery is growing fast, Full High Definition (1080p) videos are the minimum quality that can be found in all TVs in every house and office, 4K definition is already setting the new standard in most of consumer places. Clearly, the demand for video content is enormous. With this tool’s availability, and easy access to internet in most countries of the world, more focus should be put on behind-the-scene operations, which make video coding and encoding possible for these large set of video formats. These operations running in the background are called video transcoding.

Video transcoding is also referred to as video encoding, and it is basically the process of converting a video from one format to another, in order to view it in different platforms and devices. In principle, it adapts the multimedia content to bandwidth, various network protocols, device resolutions and standards. Users today may have different platforms and devices, tailoring the media content, and especially video content, to their preferences has become a necessity, especially after many research studies published lately, which have tried to understand the consumption level of video content of the new wave of video services. Active life span, growth pattern, popularity, request pattern and the implication of the development of large-scale video-on-demand services [6] [7] are the main focus of these studies.

Transcoding, however, is a very computationally intensive operation and the knowledge level needed to run this process is advanced. Many factors come to play when it comes to video transcoding; the platform, format, and device used to view the video are the most important.

Behind all this, there is a need for more sophisticated means of computing. Traditionally, scripts have been written for line-to-line computation. A problem is divided into known series of instructions; instructions are executed sequentially one after another on a single processor, and only one instruction may be executed at a moment of time. This is clearly a bottleneck. However lately, with the introduction of multicore processors in the last decade, parallel programming has become the standard when it comes to heavy computational operations, and it is exactly what video transcoding requires. Simply, parallel computing is the use of multiple computing resources simultaneously. A problem is divided into known parts which can be run concurrently. These parts are further divided into several instructions that can be run in different processors simultaneously. Control and coordination of the processor can be enhanced by the use of neural networks and AI, to get the most of the execution process optimized. Companies such as Nvidia [16] and Intel [17] have already introduced these mechanisms in their latest bleeding-edge processor technologies.

Parallel Computing might be the right solution to tackle these problems but, if misused, the results will not be satisfying. Thus, the idea of running the scripts on the consumer side is not the smartest way to do it. Instead, it can hinder the user from enjoying an experience as basic as watching a video. The consumer needs fast, responsive and easy-to-use products.

Content providers have already opted for a solution, and they are relying on public and private cloud computing servers in order to have large-scale video transcoding platforms. Cloud computing is a sort of on-demand computing and a solution to use dynamically scalable and configurable shared resources as a service. The content providers are using this service as a new business model that further optimizes the flow of development, as they pay per use, when required. As a result, using cloud computing for video transcoding has advanced to a new level, and made it easy for the content providers to deliver the most enjoyable experience for the end users.

Throughout the thesis work, I will tackle some of the known problems that appear on the way. As mentioned earlier, the heavy computational power needed is also exposed to many factors, hindering the expected flow along the way. Performance overheads and communication overheads are the main concerns of most video transcoding computations. Scaling-out, or in another word, increasing the number of virtual machines used, with the same processing power in the cloud computing platform, is a crucial step to take, but, at what point scaling-out might turn to be inefficient? Also, what is the limit of scaling-out in this situation? This thesis will try to answer these questions in detail, supported by insights and further discussions to clarify and understand the problem at hand.

Thesis Structure

The thesis contains two parts, the first part consists of the background theory of cloud computing, and it is compiled in Chapters 1, 2 and 3. The second part which contains Chapters 4, 5 and 6 is dedicated to the actual work that has been done. Chapter 1 is an introductory piece of information, to help the reader understand the direction of the thesis. Chapter 2 focuses on the basics of cloud computing, explaining the different services. It also shows the advantages and disadvantages of using the technology, finishing by mentioning Amdahl's law. Chapter 3 focuses on the related work. In chapter 4 and 5, the experiment setup is explained in detail. Chapter 6 is dedicated to the results of the experiment. Finally, a conclusion is presented together with discussions for potential future work in the last chapter.

2 CLOUD COMPUTING

2.1 Overview

Over the past two decades, we have seen a significant change in the economy of most countries of the world, especially the US, with a decline in manufacturing-oriented in favor of service-oriented economies. In 2010, service-oriented companies constituted 80 percent of the U.S economy, leaving just 15 percent for manufacturing and 5 percent for agriculture and other areas.

The advances in Information and Communication Technology (ICT), have led to the development of cloud services, and made computing ranked high in the utility ladder. This is because the actual world is in need of uninterrupted, reliable access to computing power at all times. Back in 1960, researches already showed and predicted that computing will be a utility in the future [8].

The emerging term for this kind of computing today is called cloud computing. It was mentioned before, but under many other names such as grid computing and service computing. These naming conventions have come from the fact that this type of computing was more designed for application-oriented and infrastructure building, but now it has shifted to service-oriented economy [9].

Since it is a new paradigm, Cloud Computing has many proposed definitions from different standards, such as the definition from The National Institute of Standards and Technology (NIST) [10] on page 2: “ Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. ”

Amazon Inc., as a main provider of the service [15] defines it in page 3 as: “Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.”

From the above definitions of Cloud computing, a simple short description would be that Cloud Computing provides a simple way to access servers, databases, storage, and a large number of application services over the Internet.

The main features of Cloud Computing could be summarized in:

- Dynamic configurability where users are capable of measuring and dynamically reconfigure the provided Cloud resources.
- WEB 2.0 protocols based abstraction interfaces for easy interaction with the end users.
- Economics of scale where the set of resources are efficiently utilized by many users leading to cost reduction.

The cloud is evolving rapidly, including sophisticated devices; it is no longer only a place to store data, but also it can serve as a platform to process information. It has emerged as a heterogeneous ecosystem that allows the configuration of resources in the most effective manner.

The vast adoption of cloud computing in the IT world today, has led to the immense growth of complications around it. Nevertheless, the potential errors have also increased. To have less downtime and maintenance requires the content providers to keep a localized infrastructure, which results in higher uniformity and cost efficiency. Over and above that, with the growth of mobile usage, the benefits of cloud computing for businesses turn to be obvious.

The term "Cloud Computing" includes many areas of technology; one of the most used terms to refer to it, is "Everything-as-a-Service" or "XaaS". Generally, everything means that the cloud could provide everything a user needs, from Infrastructure-as-a-Service, "IaaS", to Platform-as-a-Service, "PaaS", and to Software-as-a-Service "SaaS".

2.2- Public, Private, and hybrid Clouds

As mentioned before, cloud computing has emerged from the grid computing, to the Everything-as-a-Service concept. Grid computing is more similar to clusters. Basically, it is a collection of computers working together to perform various tasks. Each computer (or node) on a grid has its own resource manager, using a software called middleware to connect the nodes, which in its turn translates the information communicated between the systems into a standard and recognizable format.

The cloud computing provides computing resources as a service, with the idea of configurability and a handful of options to optimize the use of the computing power tailored to the need, and to budget, which makes cloud computing a high-throughput computing (HTC) model.

2.2.1- Public Clouds

A public cloud is accessible by subscription, which means that anyone can access it as a service from a service provider. Virtual machines (VMs), applications or storage, are all ready to use through the Internet. Some public cloud services are free to use, but for demo perspectives. Many public cloud providers like Google App Engine (GAE), Microsoft Azure and Amazon Web Services (AWS), are famous commercial providers of publically accessible remote interface for configuring and managing virtual machines across the Internet within their own infrastructure. However, public clouds might be a bottleneck when it comes to high-performance computing (HPC) applications, according to some studies [16]; many public cloud platforms have slow network connections between virtual machines, which questions the efficiency of these platforms to adapt to various use cases. Figure 1 illustrates the end consumer of the public clouds.

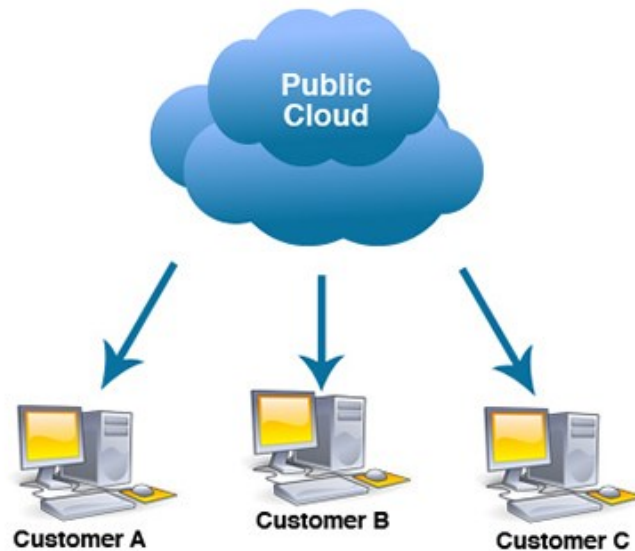


Figure 1: The general view of the destined consumer of the public clouds [18].

2.2.2- Private Clouds

In contrast to public clouds, private clouds are built around a specific domain of an intranet, which is owned by the company itself. Only clients and their partners could have access to the services. The purpose is not to deliver access through the Internet by means of intuitive interfaces. The amount of control the client has in private clouds is rather significant compared to public clouds,

therefore, the client is able to customize the services to the finest, leading to more efficient exploitation of the services. Many enterprises choose to have private clouds, since that enables them to harvest cloud benefits while vaulting their mission-critical data and software under the governance of their security policies.

Figure 2 illustrates a general view of the private cloud.

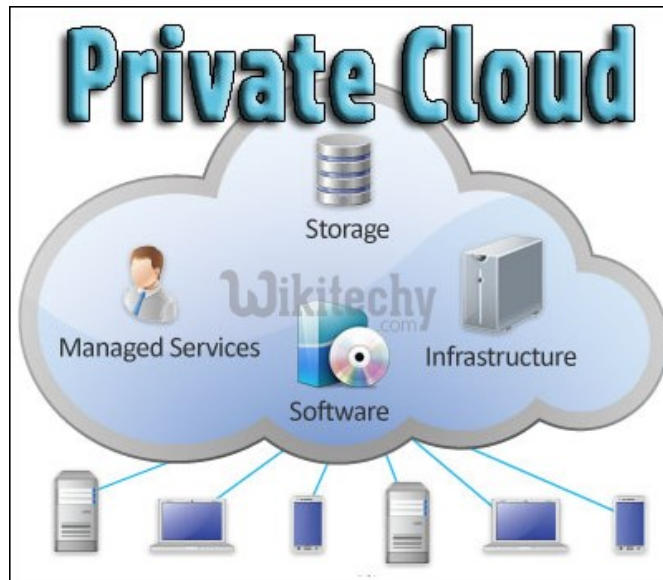


Figure 2: The general view of the private cloud [19].

2.2.3- Hybrid Clouds

The simplest definition of a hybrid cloud is a combination of public and private clouds. It incorporates cloud services from both private and public clouds to carry out distinguishable purposes within the same enterprise. As an example, the IBM Research Compute Cloud (RC2) is a private cloud used by IBM and other clients and partner networks. The clouds are spread across Europe, Asia, and the United States.

Figure 3 shows a general view of the hybrid cloud.

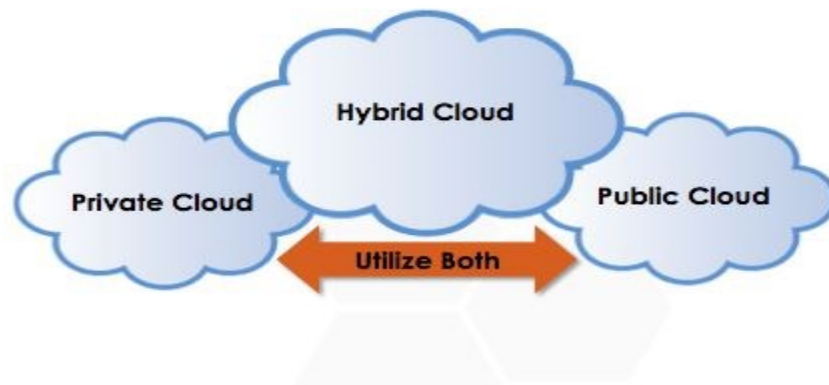


Figure 3: The general view of the hybrid clouds [20].

2.3- Service models

As already stated, according to NIST, the core services of cloud computing could be described by “Everything-as-a-Service” or “XaaS”, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Figure 4 illustrates the layers of cloud computing, and it also shows the service models arranged as layers in a stack.

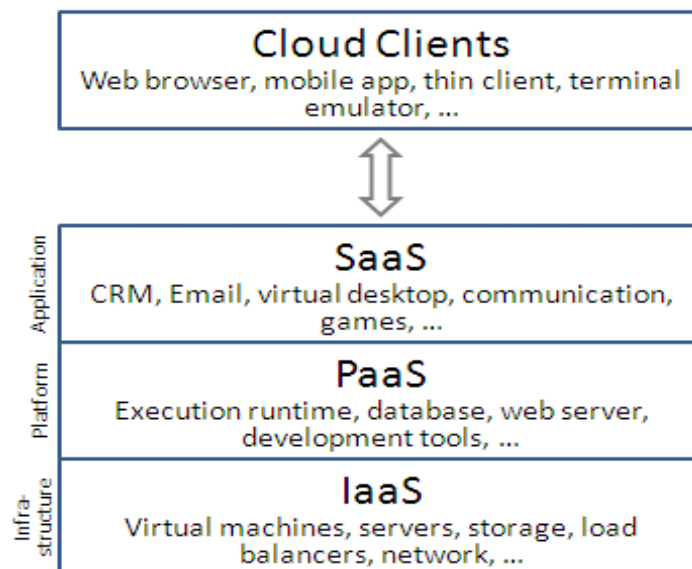


Figure 4: Cloud computing service models arranged as layers in a stack [21].

2.3.1- XaaS – Everything-as-a-Service

To imagine a world where a client is able to obtain everything as a service is quite utopic, but we are not far away from that. The concept has already started to emerge in reality, and for a few years, it has been tested, deployed and exhaustingly consumed.

The “X” in XaaS applies to “anything” or “everything”. It basically means that at any point in time, the client is able to receive the vast number of products, technology, and tools needed in the most efficient way possible. Reasons are stated previously, but why was such a service a necessity? According to Dan Rahko, Founder & CEO at SBT Partners [35]; “We got here by the convergence of several major technology innovations: the flawless Internet with high-speed networks and broadband found everywhere and the rise cost-effective server virtualization leading powerful computing.”

Lately, many new kinds of services have emerged, making XaaS not limited to online services. Uber [22] and Lyft [23] are proposing the Transportation-as-a-Service. Grocery-as-a-service and Accommodation-as-a-Service are provided by Safeway and Airbnb respectively. This is only the beginning of a new wave of services soon to arrive.

2.3.2- IaaS – Infrastructure-as-a-Service

The services provided here are to supply processing, network, storage and principal computing power needed for the clients to execute and deploy their own software. The software could range from operating systems to applications. The control does not cover the fundamental cloud infrastructure and occasionally the main networking components like firewalls.

As mentioned before, these services are available to the clients in the form of pay-as-you-go services. The service-level agreement (SLA), which is the contract between a service provider and its internal or external customers, includes the performance guidelines that the provider has to fulfill. The SLA introduces the terms concerning the performance, security, services availability and data protection.

Infrastructure-as-a-Service allows users to get hold of IT resources, networking and storage for computing purposes. In a nutshell, this service is rented out to users as infrastructure, so that they can deploy and execute their software and application on top of their operating system of choice. With that being said, the service, therefore, includes many sub-services, such as Instance-as-a-Service, which provides the computing instances, Communication-as-a-Service “CaaS” and Storage-as-a-Service. In addition to the well-known IaaS providers, i.e., Amazon Web Services and Google Apps, many start-ups have seen the light lately providing public clouds with Infrastructure-as-a-Service. Figure 5 shows the most popular public clouds providers.

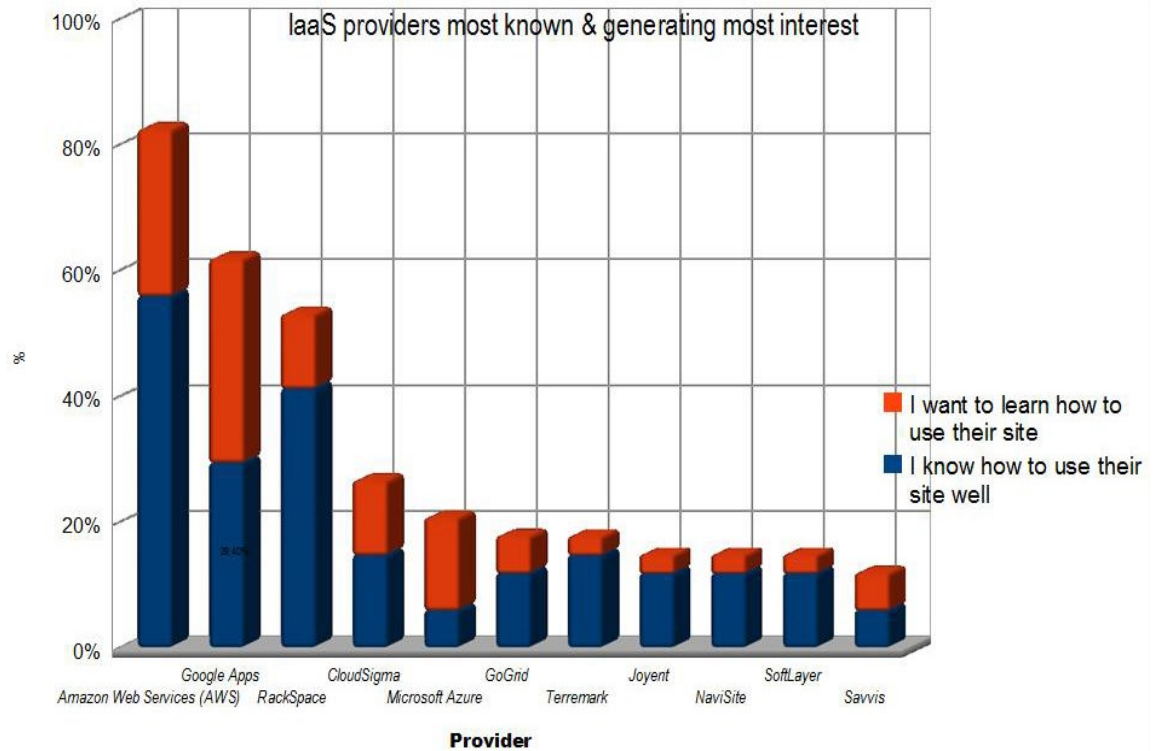


Figure 5: IaaS providers most known and generating most interest [24].

Video transcoding is a heavy computing process; more sophisticated means of computing are needed for a very cost effective and real-time resource management. The large number of video streams requires distributed systems with a large number of clusters and computing resources. Infrastructure-as-a-Service today introduces a tremendous amount of computing resources, such as virtual machines (VMs), storage, and network bandwidth, which is ready to be used in order to form a dynamically scalable cluster of video transcoding servers. The freedom to choose the resources and organize them in an optimal way provides opportunities for video transcoding to be performed in real-time, almost glitch free.

There are many ways a video transcoding operation can be performed. For example, one video transcoding process could be mapped to one virtual machine, however, it requires a large number of virtual machines to transcode several concurrent videos. Transcoding is also dependent on the quality and size of the video; high-resolution HD video streams can take more time than a regular SD video. Other available methods are to split the video streams into chunks, and after that, transcode them single-handedly [17]. The splitting is done by assigning multiple chunks of a stream to many virtual machines, and one virtual machine can also transcode multiple chunks from different streams.

The expected load from the user and the required performance predict the number of virtual machines and the size of the storage possible for a specific project.

2.3.3- PaaS – Platform-as-a-Service

NIST definition of PaaS, page 2: “The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment [10]”.

The end customer, at some point, should be able to use the operating system, libraries, and applications of choice. For this sake, Platform-as-a-Service has been created to bridge the gap between the infrastructure and the client.

Both hardware and software infrastructures are part of the cloud platform, the client can develop and deploy applications using well-known programming languages and software tools supported by the cloud provider. The limitation of managing fundamental cloud infrastructure also applies here.

The most famous PaaS providers are Google App Engine, Heroku and Red Hat’s OpenShift. Figure 6 shows the differences between the providers with respect to many factors.

Cloud	Tool and Programming Languages	Target Application
Google App Engine	Python, Java	Web Applications and BigTable storage
Microsoft Azure	Apex, Eclipse-based IDE	Business applications, CRM
Aneka	.Net, stand-alon SDK	.Net enterprise applications, HPC
Amazon Elastic MapReduce	Hive, Pig, Cascding, JavaPerl, Runy	Data Processing and e-commerce

Figure 6: PaaS providers in Public Cloud [25].

2.3.4- SaaS – Software-as-a-Service

End clients, whether they are businesses or individuals, generate enormous outcomes using the clouds. Low cost turns out to be a great benefit, since expensive licenses of software would not apply when using Software-as-a-Service.

Software-as-a-Service provides software application and tools as a service. For the provider, the cost is low compared to traditional hosting of the user application.

The most potent aspect of Software-as-a-Service is the fact that when everything is processed on the cloud, many people can interact with the same bit of information at the same time. For example on Facebook or YouTube, when someone likes a friend's post, another friend in another place might be commenting on it.

Microsoft SharePoint, Google Gmail, and CRM software are the most famous SaaS providers. These services are mostly free of charge, and they are used by the providers to promote the other paid services.

2.4 Advantages and Disadvantages of Cloud Computing

Companies such as Spotify [33], Foursquare [34] and many giants are rushing towards adopting this new trend. According to the latest surveys, it makes them more profitable and more cost effective. Many reasons have made them switch to cloud-based computing. Among them we can find:

- Flexibility in costs:

Cloud computing cost is really low compared to the traditional IT infrastructures. By using the pay-per-use rule, companies can optimize the use of the utility and reduce cost whenever possible.

- Availability:

Uptime availability is the key for success of the cloud-computing trend. Cloud services are extremely reliable, and the maintenance work is done on a regular basis. The redundant nodes distributed on the system ensure that the requirements are fulfilled.

- Mobility:

The client is now able to use the services on the go; all the user needs is a laptop and a stable internet connection.

- Scalability:

Scaling-out is the fact of adding multiple virtual machines to upgrade the hardware.

Obviously, the above reasons have changed considerably in the software and hardware industry, but at the same time, there exist many disadvantages of this trend. The following are some of the major disadvantages of this technology:

- Security and privacy:

Security and privacy must be addressed any time a discussion about data is invoked, mainly when managing critical data. Some large companies have closed down because of a major hacking attack, which results in deleting users' data. Companies are ready to cloud manage all the data they own as soon as the data privacy and security are firm enough.

- Downtime:

This might be the biggest disadvantage of the system. Invulnerability to service interruption is not easy to neglect from the client and the provider side. The system is internet based, thus, the reasons to fail are multiplied.

- Platform dependencies:

Migration between platforms introduces a risk as well. The “vendor lock-in” or the dependency to the platform of the first provider makes it difficult and sometimes impossible for the client to reconfigure the applications for the new host. However, data are clearly vulnerable to all sorts of risks in the process of migration.

Those were a few drawbacks of the cloud computing from a top-level perspective. Digging further in this technology and its implications, we could find more practical problems that the tenants (individuals or companies) face in everyday usage of the IaaS in particular. Some specialists call these problems deep drawbacks, but throughout this thesis work, they will be called cloud computing overheads for simplification. Thus, what do these overheads have to do with cloud computing and why are they so important? Also, why are they taking all the attention in today's discussions? To answer these questions, we must first clarify the nature of these overheads and discuss where and how they take place in the execution of the program in the cloud-computing platform.

- Performance overheads:

As the name indicates, they are the unpredictable execution times of the applications running on the cloud.

- Communication Overheads:

The time consumed on the communication between different resources on the virtual machine, and sometimes in different virtual machines, in distributed data centers for instance.

Other overhead types will be covered in detail in the next chapter, showing related works and research by other universities that cover the same topic.

2.5- Amdahl's Law

Amdahl's law is a formula used to find the maximum improvement possible, by improving a particular part of a system. In parallel computing, Amdahl's law is mainly used to predict the theoretical maximum speed-up for program processing, using multiple processors. It is named after Gene Amdahl, a computer architect from IBM and the Amdahl Corporation.

The speed-up is defined as the time it takes a program to execute in serial (with one processor), divided by the time it takes to execute in parallel (with many processors).

Amdahl's law can be formulated in the following way:

$$S_{\text{latency}}(s) = \frac{1}{(1 - p) + \frac{p}{s}}$$

Originally, Amdahl's law was intended for the comparison between one-processor and multi-processor execution times, but it can also be used in other systems. In a complex system like cloud computing, as illustrated in Figure 7, Amdahl's law can be applied to obtain an estimate of how fast the cloud could be, in comparison to a local machine. The clouds have always the cutting-edge technology embedded in their servers, which implies that there will always be an advantage to use the clouds compared to traditional servers.

The green arrows 1 and 3 in Figure 7 below show the communication time between the client machine and the clouds (sending and receiving). This time should not be neglected in the calculations, and it is called communication overheads. As stated before, the other overhead

involved here is the performance overheads, Amdahl's law will also be used to figure out the number of performance overheads, using a set of experiments throughout the thesis work.

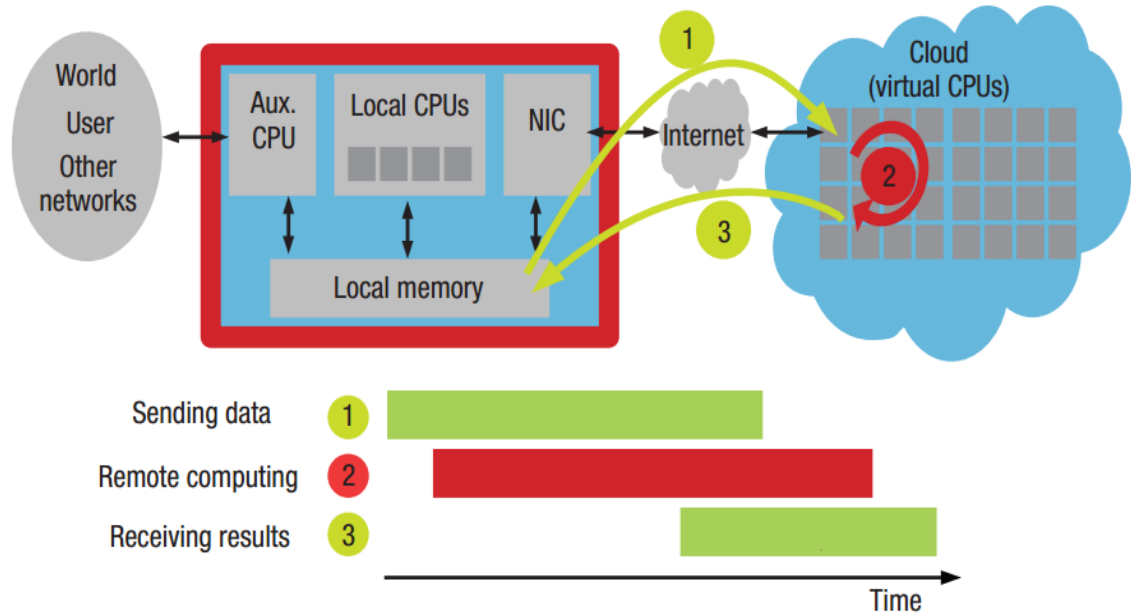


Figure 7: Localized application execution (top left) compared to an application executed on the cloud (top right and bottom). NIC: network interface controller [26].

3 RELATED WORK

3.1 Managing Performance Overhead of Virtual Machines in Cloud Computing

The famous survey realized by Xu et al. [14], called “Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions”, states that resource contention between virtual machines is the main culprit for unpredictable performance. The resource contention leads to the degradation and variation, also known as “performance overhead’ in Infrastructure-as-a-Service (IaaS). The survey has realized a model for quantifying the performance overheads in three main scenarios:

- Single-server virtualization.
- Single large datacenter.
- Multiple geo-distributed datacenters.

The paper explains the main issues faced by large companies, trying to move their critical data into the clouds. It continues by elaborating on the low-level problems occurring at the CPU level. It compares the latest solutions for addressing performance overhead in the three cloud scenarios noted above; it also weighs the benefits and costs, as well as research challenges.

The scope of the survey is limited to the solutions that alleviate overheads in favor of both tenants and cloud providers.

The paper then continues by explaining the major techniques used to diminish the performance overheads in each scenario. It starts from a single server virtualization, where the fundamental technique is resource isolation. The paper suggests that other techniques, such as enhanced modeling methods, could cost-effectively alleviate the performance overheads. Current techniques have provided good performance isolation mechanisms on sharing resources like central processing unit (CPU) cores, memory, and disk capacities among collocated VMs. These isolation mechanisms are implemented by CPU schedulers. Contention occurs among server resources, in particular, cache and memory bandwidth on a physical server.

Along with performance overheads caused by single server virtualization, there are routine operations of the IaaS cloud data center that further complicate the issue, and can degrade the performance of VMs. There are three cases where this occurs:

The concurrent live migration of VMs that enables load balancing and power saving can cause overheads on the migration source and destination servers.

Both public and private clouds suffer from overhead issues caused by the concurrent deployment and snapshotting of multiple VMs, because of network traffic and interference.

The inability to isolate shared cloud network and storage resources among multiple VMs is another cause of performance overhead.

Under the multiple geo-distributed data centers scenario, new challenges arise: how to manage overhead caused by live VM migration or storage migration over the WAN connection.

The survey then tackles the modeling issue related to all these scenarios, and explains the different possibilities and techniques, to retrieve the most accurate data possible from these measurements.

As usual, they have started with how to alleviate the VM performance overhead caused by single-server virtualization. Currently, the hypervisors provide a baseline of performance isolation on CPU, memory, and disk resources, but the cache and I/O bandwidth resources shared among multiple collocated VMs cannot be easily isolated. To deal with this problem, several approaches have been proposed to mitigate the overheads, including resource isolation among collocated VMs, and optimization of the VM assignments (finding the optimal mapping of VMs to physical servers).

For mitigating the VM performance overhead within a single data center, one should note that the routine operations of an Infrastructure-as-a-Service cloud data center can severely take down the performance of the VMs. The first performance overhead mitigation technique is the resource isolation on virtual machines sharing the same location. The technique could be hardware and software oriented. On one hand, the interference on the network bandwidth of these virtual machines could be diminished by installing multiple network adapters on a physical server. On the other hand, the software solution includes manually setting the bandwidth cap before starting the virtual machine, or reducing the bandwidth limit using “tc” utility in the Linux kernel.

Concerning the methods to mitigate the virtual machine’s performance overheads on a single data center, as mentioned before, the live migration, deployment and snapshotting of multiple virtual machines are some of the reasons for overheads. Many solutions are been provided mainly to solve the migration problem. Joining forces to reduce the amount of the data transferred during live migration of virtual machines by using, for example, advanced compression techniques called MECOM and Delta compression technique that transfers only data between the present and the actual memory pages over a slow network.

The contention of multiple virtual machines on shared storage resources, as stated before, is a type of overheads that occurs especially on a single data center, as shown in Figure 8. To isolate the storage input/output resources within multiple virtual machines, the most common method is to manage and schedule the request in input/output queues on each server and storage facility. More approaches like PRADA control the length of the input/output queue on the servers in regards to the weight of the virtual machine. Further discussions concerning overheads were about contentions of multiple virtual machines on a shared network.

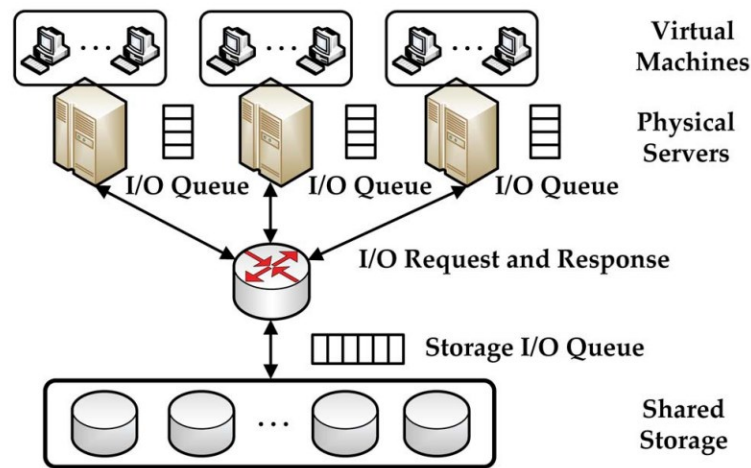


Figure 8: Typical shared architecture in the datacenter [27].

The geo-distributed data centers, according to the paper, could be divided into two segments, the migration over WAN of the virtual machine, and the migration over WAN of the virtual machines storage.

The proposed methods of migration over WAN of the virtual machine are numerous, but the most straightforward is the Write throttling mechanism. This mechanism reduces the number of dirty pages by limiting the write operations counts to a well-known number. VMFlockMs is another mechanism that reduces the time of migration by data prioritization. It sends the data responsible for starting the virtual machine and launching the application as a first class choice. CloudNet is mainly about the memory pages copied to the new host of virtual machines, it basically restricts the copying to only the difference between the current and the previous pages.

Figure 9 shows the classification of causes and mitigation techniques of VM performance overhead from the viewpoint of IaaS cloud hierarchy.

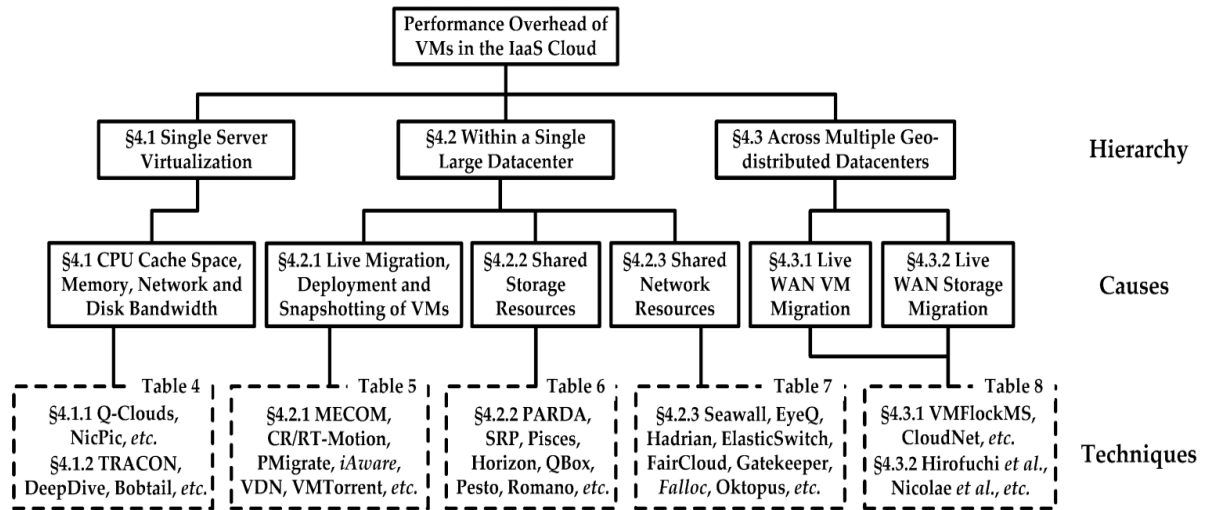


Figure 9: Classification of causes and mitigation techniques of VM performance overhead from the viewpoint of IaaS cloud hierarchy [28].

As a summary, the research paper concludes that the overheads introduced could be mitigated, using several methods and techniques. By dividing the issue into three main areas, single server, single data center and geo-distributed data centers, the authors were able to present solutions and suggestions to limit the overhead issues.

4 SETTING UP THE ENVIRONMENT

4.1 The environment

As mentioned in the previous chapter, the environment used for the thesis work is the cPouta cloud computing service. The work starts by creating many virtual machines that could be accessed from other computers, so that it could be possible to execute and run scripts on the clouds.

4.1.1 Virtual Machines:

A virtual machine (VM) is an operating system that emulates a specified computer environment (architecture and software); the user has the same level of experience as in dedicated hardware.

The execution of the thesis experiment will be in the virtual machine environment. A virtual machine provides a platform that executes programs in an independent environment, it is also considered as a substitute for an actual real machine. The system requirements that are needed to operate a program are also provided, in addition to the hypervisor that manages the hardware and the software instances. The hypervisors primarily control the virtualization, and the users can basically create instances of the machine of their own choice.

Many choices of virtual machines exist. Throughout the research phase of the thesis, an extensive list of cloud computing providers was found to fit the description; first and foremost Amazon Web Services (AWS).

Amazon Elastic Compute Cloud, which is known as EC2, is part of the services provided by Amazon Web Services, and it provides the possibility to manipulate instances on the servers as a service on demand. The Elastic Compute Cloud is considered to be one of the most important aspects of the AWS that provides the instruments for the clients.

The client is able to provision a multitude of choices of server images; the facility provides the users also with the ability to create their own virtual machines images to be used in the EC2. These services are offered as Infrastructure-as-a-Service. Amazon offers other services to businesses. The following is a summary of remote computing services offered by Amazon.com:

- Amazon EC2
- Amazon S3
- Amazon RDS
- Amazon ElastiCache
- Amazon Elastic MapReduce

In addition to AWS, Microsoft Azure provides a set of services as Infrastructure-as-a-Service. In this case, Microsoft Azure (shown in Figure 10) is a more business-oriented platform, and it is surrounded by a magnitude of services that make it a comprehensive solution for businesses and startups; below are the services offered:

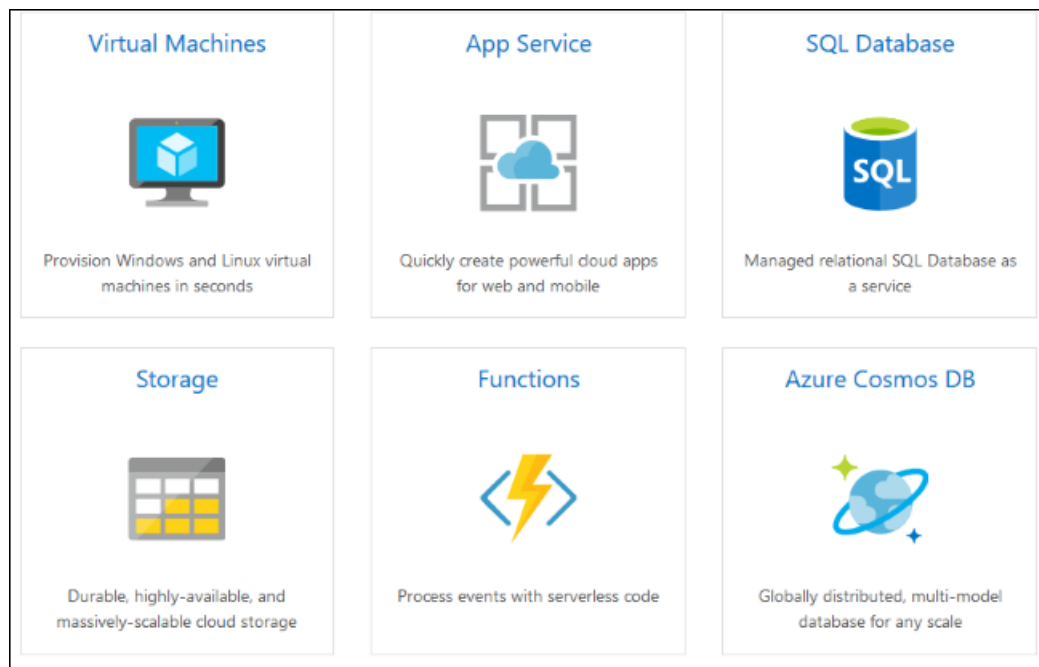


Figure 10: The services provided by Microsoft Azure [29].

After a thorough investigation to identify the most suitable cloud-computing provider for the project at hand, a service called cPouta IaaS Cloud, from the SCS was chosen for this thesis work.

SCS short for “Center for Science”, and it is an expertise center from Finland that provides high-quality ICT expert services to research, education and culture in the country. It also provides services to enterprises and public services.

CPouta is suitable for the project in many regards. Firstly, it is free of charge for Finnish universities to use. It is credit based, and the number of credits allocated to a project or thesis can be extended according to the required amount from the project.

Secondly, the service is provided from Finland, which results in a high-speed and secure connection to the clouds. Also, because this service is designed for researchers in particular, the platform is designed intuitively to suit the needs of engineers. More importantly, issues could be solved directly and quickly through direct contact with engineers and technicians from SCS.

SCS provides seminars and tutorials to the users. Most of the tutorials are open and free, and could be found on their YouTube channel [36].

In the next chapter, we will go through setting up a cPouta cloud computing platform, creating virtual machines' instances, setting up secure connections, and creating security groups in detail.

5 CPOUTA

CPouta uses a well-known platform for managing and configuring the clouds which is called OpenStack. An introduction and thorough explanation of this platform are presented further in the thesis.

5-1 OpenStack

OpenStack is a cloud operating system, which has been made to control a large set of servers, often called Pool Compute, and to manage their storage and networking resources. The control is done utilizing a data center. Figure 11 illustrates the architecture of the OpenStack framework.

Provisioning the resources is done via a simple website, mostly using an intuitive web interface that could be understood quickly. The user may, therefore, control the components of the system through the web interface, having all the necessary administration rights and privileges for a particular type of project.

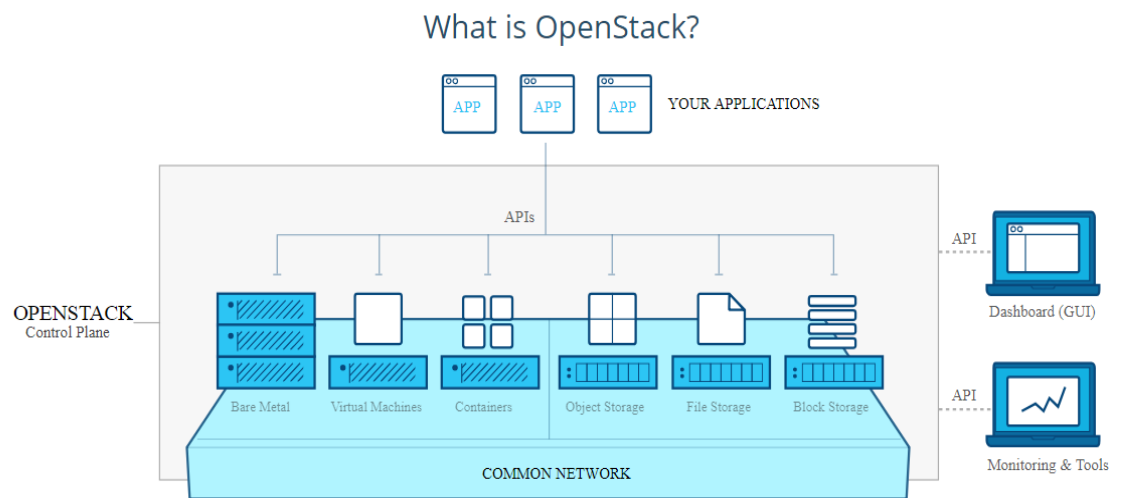


Figure 11: depiction of the OpenStack framework [30].

The provisioning of the resources is done through a set of open source software, which is a large project that started in 2010 by both The NASA and Rackspace. Since a significant growth in

the hardware industry has been made, the need for open-source software has emerged, and that allowed the providers to offer cloud-computing services to the majority of the world.

Like any other open-source project, developers are writing code permanently, mainly in Python programming language, and the result is always free to use under the Apache 2.0 license.

Some of the advantages of OpenStack are listed below:

- Manageability
- Availability
- Agility
- Efficiency
- Rapid development
- Testing
- Deployment massively scalable applications
- Competitive price
- Significant competitive advantage for enterprises.

Managing and provisioning time is decreased significantly from weeks to minutes, and this could also count as an advantage for this platform.

OpenStack has different components; each one of them is responsible for executing a set of tasks, for example, computing, changing the IP address of a machine, or starting a virtual machine. Each component produces log files after each operation it performs. Those log files are sometimes large and generally difficult to understand by the user, finding the root cause of a problem could turn to be a difficult task. However, paying attention to log files is very important in order to troubleshoot the issue that may occur.

5.1.1 Nova Compute

Nova Compute is an essential part of the controller in OpenStack, which in turns is considered the main section of the Infrastructure-as-a-Service (IaaS). The main job for this part is to control a set of computers, mainly described as a pool of computers. This part is also compatible with various kinds of virtualization technologies. Other types of technologies like high-performance and bare-metal are also compatible and can be configured.

The Nova tool is written in Python and is composed of the following sections:

- Concurrent programming is provided by Eventlet external library.
- AMQP communication is made possible using Kombu.
- Database management provided by SQLAlchemy.

The basic idea of this technology is to design an architecture working on top of any standard hardware, without specifying any requirements on the type of hardware used for the software at hand, this facilitates the integration with legacy systems and other third-party technologies. Below in Figure 12, the architecture of Nova is shown.

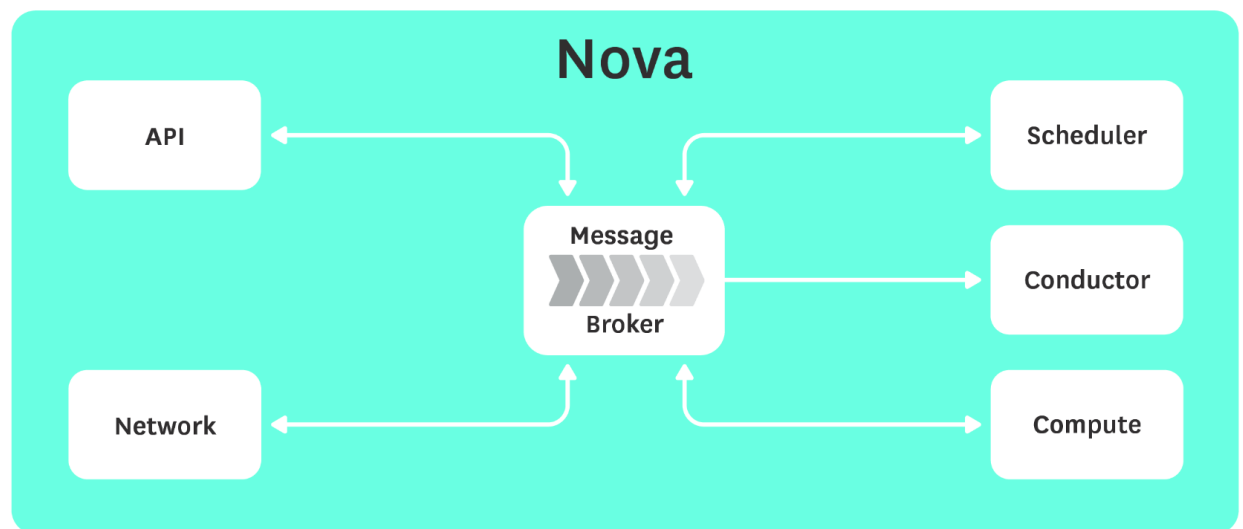


Figure 12: The architecture of Nova [31].

Nova is divided into three main categories:

- Hypervisor metrics.
- Tenant metrics.
- Nova server metrics.

Each metric of the above gives unique data collection capabilities; hence facilitating the work with Nova. The hypervisor metrics, for example, give a thorough view of the job performed by the hypervisor and its tools. Tenant metrics are essential to understanding the user's resource usage. The Nova server metrics provide an interface of the virtual machine instances and their tools.

Combining these metrics with other third-party metrics like RABitMQ would give a thorough understanding while developing any application. The final result could be a detailed description of the system, and full control of the OpenStack.

5.2 Instances creation

Before creating an instance under the chosen environment, a few steps should be taken into consideration in order to ensure that the communication to the clouds is successful.

5.2.1 *Setting up a cPouta instance*

There exist two different ways to create an instance in the cPouta environment, one of them is to use the Command Line Instructions, making use of Nova compute. A necessary connection from the client computer should be created, and some parameters from the server side should be set up.

5.2.2 *SSH protocol connection*

SSH is the abbreviation of Secure Shell; it provides a secure connection between the client computer and the cloud servers or the virtual environment. This secure connection lets the user securely operate the network services using an open network. The port used is the standard TCP port 22. Figure 13 depicts the architecture and communication steps used during an SSH connection.

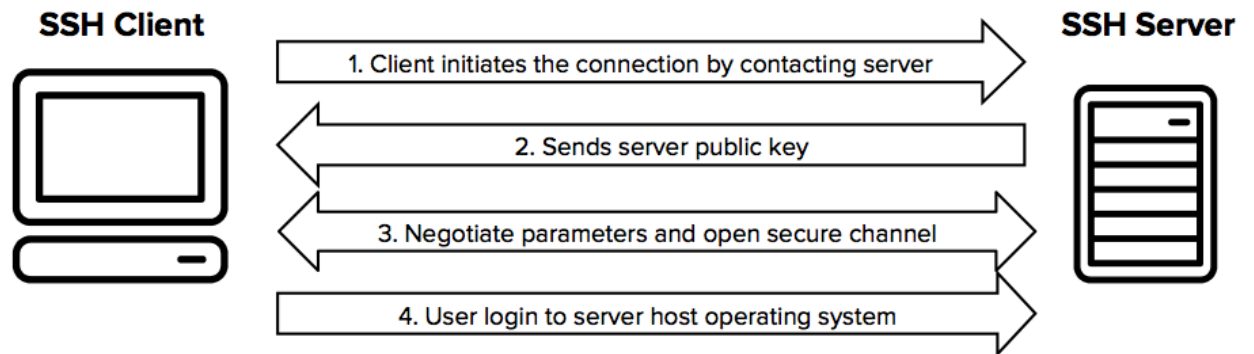


Figure 13: SSH connection protocol [32].

SSH is the first thing to set up between the client computer and the clouds, for this to operate, we need to create a private key. The key needs to be known between the two ends, and it should be identical.

To create the key, we use the web interface provided by CSC. The web interface is straightforward and easy to use and control. The steps are the following:

After login in, navigate to “access security” menu, and press the “Create Key Pair” button as illustrated in Figure 14.

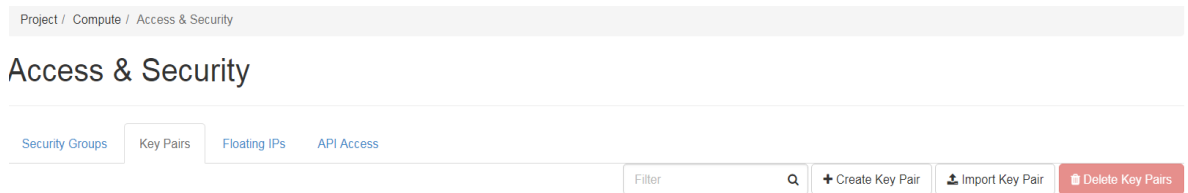


Figure 14: Create Key Pair button.

The next window is for the name of the key; here we have chosen “transcode” as the name of the key used in the thesis, as shown in Figure 15.

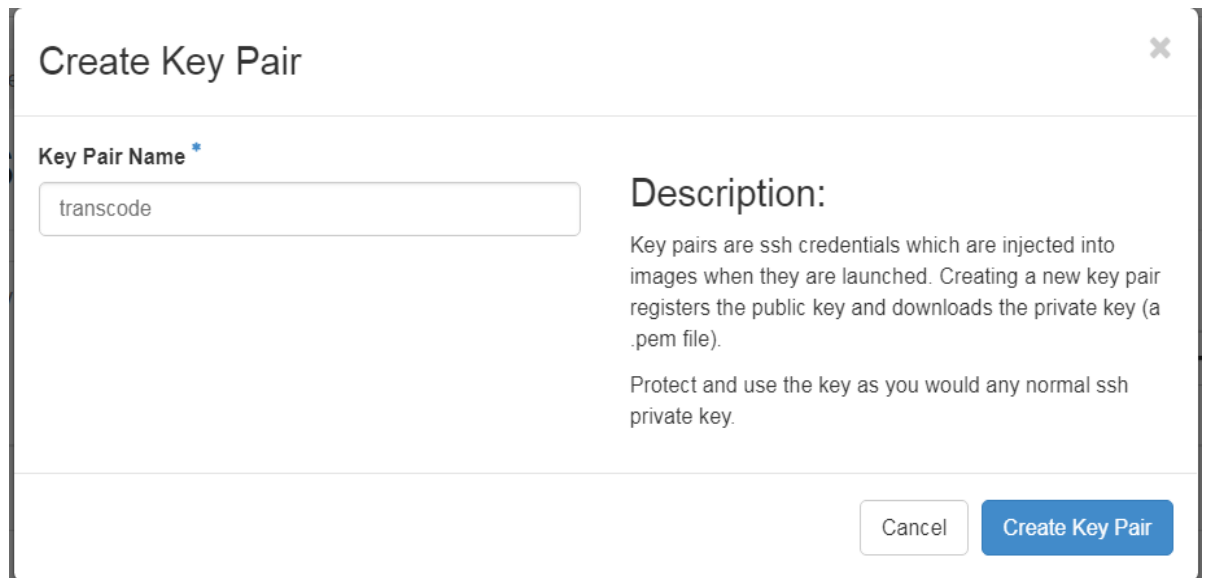


Figure 15: Create Key Pair button details.

Once created, an automatic download will be launched, and the key will be shown as ready to use on the interface, as it is shown in Figure 16.

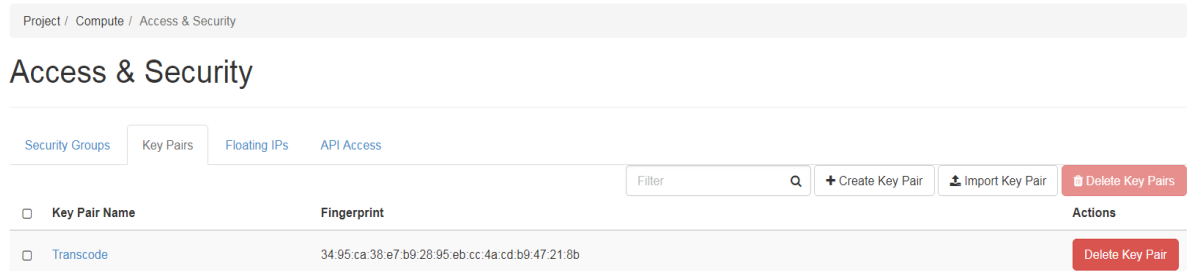


Figure 16: Create Key Pair result.

The downloaded key should be placed in the SSH folder. Figure 17 shows the SSH folder used here.

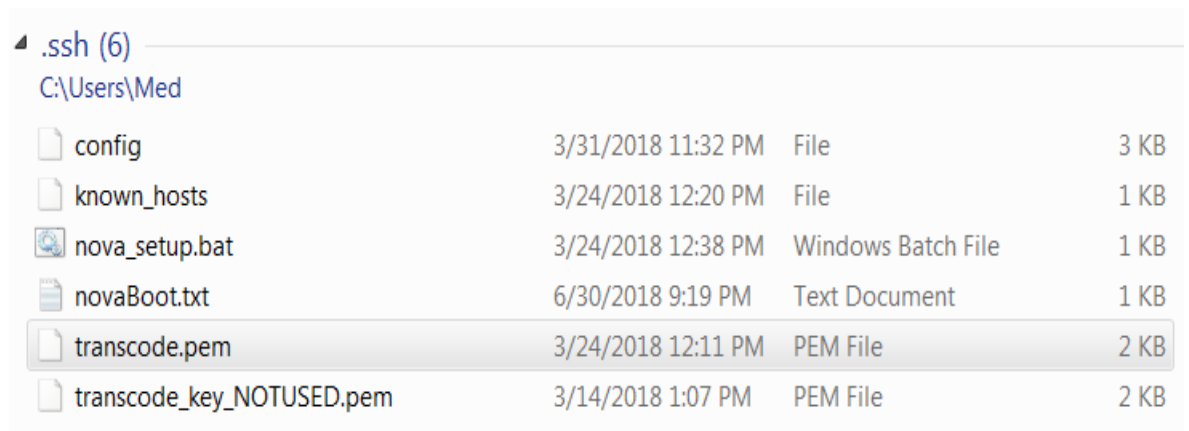


Figure 17: The SSH folder structure.

The key will be used from the client computer to create a secured connection.

One more thing to create in addition to the key is the “Security Group”. The security group settings allow the configuration of the clouds, granting access to the client computer using its IP address.

The web interface includes a tool to create the security groups that could be found in “Access&Security” menu, then the “Security Group” tab as shown in Figure 18.

Access & Security

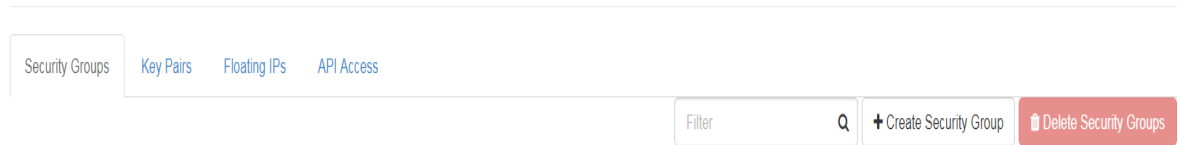


Figure 18: Security group button.

To perform this operation, we use the “Create Security Group” button; a window will be shown to enter the name of the security group and a description, as illustrated in Figure 19.

 The image shows a dialog box titled 'Create Security Group' with a close button (X) in the top right corner. The dialog has two main sections: 'Name' and 'Description'. The 'Name' section has a text input field containing 'Goteborg_laptop'. The 'Description' section has a larger text area and a descriptive text block that reads: 'Description: Security groups are sets of IP filter rules that are applied to the network settings for the VM. After the security group is created, you can add rules to the security group.' At the bottom right of the dialog are two buttons: a white 'Cancel' button and a blue 'Create Security Group' button.

Figure 19: Security group result.

After entering all the necessary information, the client needs to manage the rules of the security group created. For that, we need to enter the IP address of the client that will be using the virtual environment.

The first step is to click on “Manage Rules” button, and then we need to add the IP address of the client as shown in Figure 20.

Manage Security Group Rules: Goteborg_Laptop (855ff087-2018-40bf-87ba-dd22f4718c07)

<input type="checkbox"/>	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
<input type="checkbox"/>	Egress	IPv6	Any	Any	:::0	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	80.217.203.196/32	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	192.168.1.16/22	-	Delete Rule

Figure 20: IP address having access to the instances.

Here the IP address 80.217.203.169/32 was added, so it will be easy to access the clouds from this computer.

2.3 Create an instance

As mentioned before, there are two ways to create an instance in cPouta, the first one is through the web interface, and the second is through the command line.

We will start describing the first type which could be done quickly through the interface on CSC.

To begin, we need to navigate to “Instances” menu, and to click on the “Launch Instance” button. The resulting instance is shown in Figure 21. We also need to add a floating IP address to the first instance created in order to be able to connect to it from the client computer, through the shell script.

<input type="checkbox"/>	Transcode	-	<ul style="list-style-type: none"> 192.168.1.10 Floating IPs: standard.small	Transcode	Active	nova	None	Running	4 months, 1 week	Create Snapshot
			<ul style="list-style-type: none"> 193.166.25.70 							

Figure 21: The instance details.

The instance called “Transcode” will be the main instance that is connected to my computer, and it will be used throughout the project. The “Transcode” instance will contain all the scripts and will be connected to the rest of the instances that will be created later on this project.

The “Transcode” instance will also be called “Master” in the shell script.

The flavor used in all the instances is “standard.small”. This flavor contains two virtual CPUs, two gigabytes of RAM and 80 gigabytes of storage.

The second way of creating the instances in the OpenStack environment is through Command Line Interface (CLI). OpenStack contains a large number of libraries that could be used to manage and control the whole environment, easily accessible by CLI.

In order to create the instance using the “Nova” command line tools, we need to set up a nova connection to the clouds. The following Nova variables need to be set as follows:

```
set OS_USERNAME=username
```

```
set OS_PASSWORD=password
```

```
set OS_PROJECT_NAME=name_of_the_project
```

```
set OS_AUTH_URL=https://pouta.csc.fi:5001/v3
```

A snapshot of the main instance needs to be also created, in order to have the same capabilities and the same size as in the other instances. Once this snapshot is created, we get its ID; this ID is used in the creation of the other instances.

“Nova boot” is used to create a new instance as shown in Figure 22.

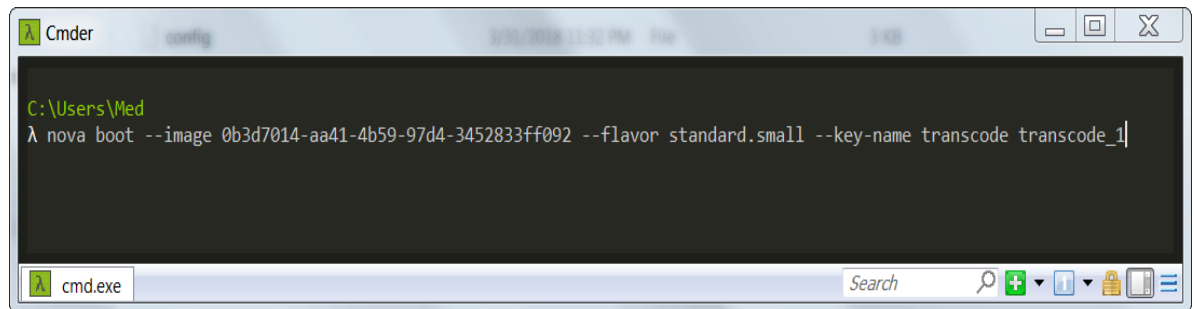
A screenshot of a Windows Command Prompt window. The title bar shows 'Cmder' and 'config'. The window content shows the current directory as 'C:\Users\Med' and a command being entered: 'λ nova boot --image 0b3d7014-aa41-4b59-97d4-3452833ff092 --flavor standard.small --key-name transcode transcode_1'. The taskbar at the bottom shows 'cmd.exe' and a search bar.

Figure 22: creation of an instance from command line.

The “--image” argument holds the ID of the snapshot of the Master instance.

The “--flavor” argument holds the flavor chosen at the beginning.

The “--key-name” argument holds the SSH key name.

The last argument is the name of the instance to be created. To check all the instances created, we can use “nova list” as shown in Figure 23. The result includes The ID of the instances, the name, the status and the network information that could be used to access the specific instance.

```

λ
C:\Users\Med
λ nova list

```

ID	Name	Status	Task State	Power State	Networks	CLI
c651eab9-4644-4c3c-9922-18a43cea31a9	transcode	ACTIVE	-	Running	project_2000748=192.168.1.10, 193.166.25.70	
0a51224a-01eb-43f4-b51f-04c473e603ff	transcode_1	ACTIVE	-	Running	project_2000748=192.168.1.16	
24d2188c-ddfd-415e-b455-6f34fc020f46	transcode_10	ACTIVE	-	Running	project_2000748=192.168.1.17	
902560e7-5372-4e1c-8d5f-2d20cea52509	transcode_11	ACTIVE	-	Running	project_2000748=192.168.1.20	
40fc74a0-9b4f-4c6c-9a8d-daf0401d36ff	transcode_12	ACTIVE	-	Running	project_2000748=192.168.1.24	
d7ed1a81-be70-4863-a3e1-510783865207	transcode_13	ACTIVE	-	Running	project_2000748=192.168.1.25	
1645acc3-8d82-43a3-b6f0-df319a193906	transcode_14	ACTIVE	-	Running	project_2000748=192.168.1.27	
fa7a7ee7-32c4-4932-98e3-25d45472925c	transcode_15	ACTIVE	-	Running	project_2000748=192.168.1.13	
04811d05-53b3-4db4-bd8d-1ed3eb17871e	transcode_16	ACTIVE	-	Running	project_2000748=192.168.1.22	
0e00113b-f14b-4191-b073-c4c3b77a0e72	transcode_17	ACTIVE	-	Running	project_2000748=192.168.1.28	
1cbd78d6-dd18-4e20-bb8b-c58b2599553b	transcode_18	ACTIVE	-	Running	project_2000748=192.168.1.32	
2987738e-3b17-44b1-ba6c-0846274b6a5c	transcode_19	ACTIVE	-	Running	project_2000748=192.168.1.25	
9193fed1-c8d5-4e1b-8b2c-fc3c5ba93b19	transcode_2	ACTIVE	-	Running	project_2000748=192.168.1.12	
2516b942-41db-46fd-a9df-b65065d7b957	transcode_20	ACTIVE	-	Running	project_2000748=192.168.1.8	
5c992f41-3e8d-4fee-a4a5-2ecfb4f1bda9	transcode_21	ACTIVE	-	Running	project_2000748=192.168.1.21	
23d9a56c-b7f7-4e44-8cc9-80151246bbe1	transcode_22	ACTIVE	-	Running	project_2000748=192.168.1.35	
76b8f2c6-6c73-471e-a87d-2af370abc710	transcode_23	ACTIVE	-	Running	project_2000748=192.168.1.36	
c93645a3-71a3-4310-80af-17696421a1f3	transcode_24	ACTIVE	-	Running	project_2000748=192.168.1.26	
d06059aa-803a-4294-9d3b-ae433a6e6dbd	transcode_25	ACTIVE	-	Running	project_2000748=192.168.1.38	
7cdc97ad-1e90-4efb-a526-9841964becf3	transcode_26	ACTIVE	-	Running	project_2000748=192.168.1.30	
94747c1e-dd43-463d-a951-8c75b06ffe07	transcode_27	ACTIVE	-	Running	project_2000748=192.168.1.39	
d1350f35-9a8d-404a-9467-cef00fc33945	transcode_28	ACTIVE	-	Running	project_2000748=192.168.1.29	
eb86a439-e108-4a39-b72b-7be98116e200	transcode_29	ACTIVE	-	Running	project_2000748=192.168.1.37	
e4fa9818-1097-40fe-bcc9-6507b6622fde	transcode_3	ACTIVE	-	Running	project_2000748=192.168.1.6	
32ae4da0-ec97-4432-abb1-7b4ef552d0db	transcode_30	ACTIVE	-	Running	project_2000748=192.168.1.7	
0a745623-e6da-4934-b4e5-9a2a24107597	transcode_31	ACTIVE	-	Running	project_2000748=192.168.1.23	
127e9012-2bf3-4b0f-87a2-cbb3fe7454bd	transcode_32	ACTIVE	-	Running	project_2000748=192.168.1.43	
03e42048-b030-4eb7-95a8-c5b6044bd92d	transcode_33	ACTIVE	-	Running	project_2000748=192.168.1.42	
0017fb32-60b4-4eea-aa4b-48fa072f17ee	transcode_34	ACTIVE	-	Running	project_2000748=192.168.1.47	
10547665-487f-4b79-92c0-9d249197c59c	transcode_35	ACTIVE	-	Running	project_2000748=192.168.1.34	
cfbaf3e0-20dc-419b-8876-f1c0479ff60e	transcode_36	ACTIVE	-	Running	project_2000748=192.168.1.45	
ec3975d7-a616-436e-8433-b9cbd4d115a0	transcode_37	ACTIVE	-	Running	project_2000748=192.168.1.40	
1461e338-3c87-4592-8ff2-a3d75f006337	transcode_38	ACTIVE	-	Running	project_2000748=192.168.1.41	
bed43cd1-3728-49ec-92af-db7781237336	transcode_39	ACTIVE	-	Running	project_2000748=192.168.1.44	
195587cf-0a6d-4e84-98de-616909b25a5f	transcode_4	ACTIVE	-	Running	project_2000748=192.168.1.19	

Figure 23: Nova list command.

5.2 Instance access

The tools and steps to access the instances are presented in this section. The SSH connection, as well as its instructions, are also described here.

5.2.1 Accessing the master instance

Having many instances in the setup provides the possibility to test the transcoding script freely, and to perform multiple test cases of the script.

The plan is to run the script using a different amount of identical virtual machines and to use the same environment and operating system.

In order to control the virtual machines through the master instance, we need to grant access to the master instance, here called “transcode”, so that it can communicate with the rest of the created instances.

The first step is to connect to the master instance using the SSH secure connection. From the SSH folder in the client computer, and using the SSH key generated earlier, together with the security groups also created in a previous step, we now have all the tools to connect to the master instance to the rest of the instances.

Once we download the generated key, we need to perform a security check to add a password and a level of security to the key; this is done using the commands “chmod” as shown in Figure 24. The password is added in the next step.

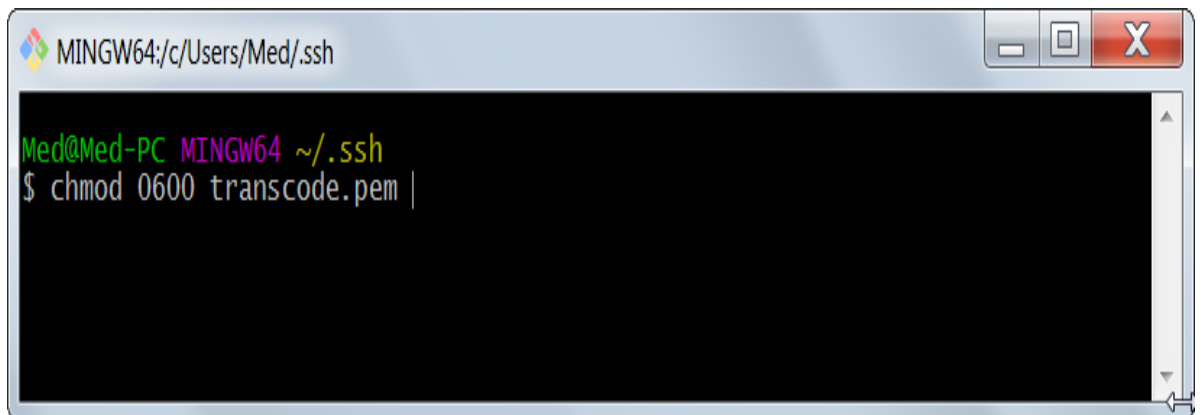
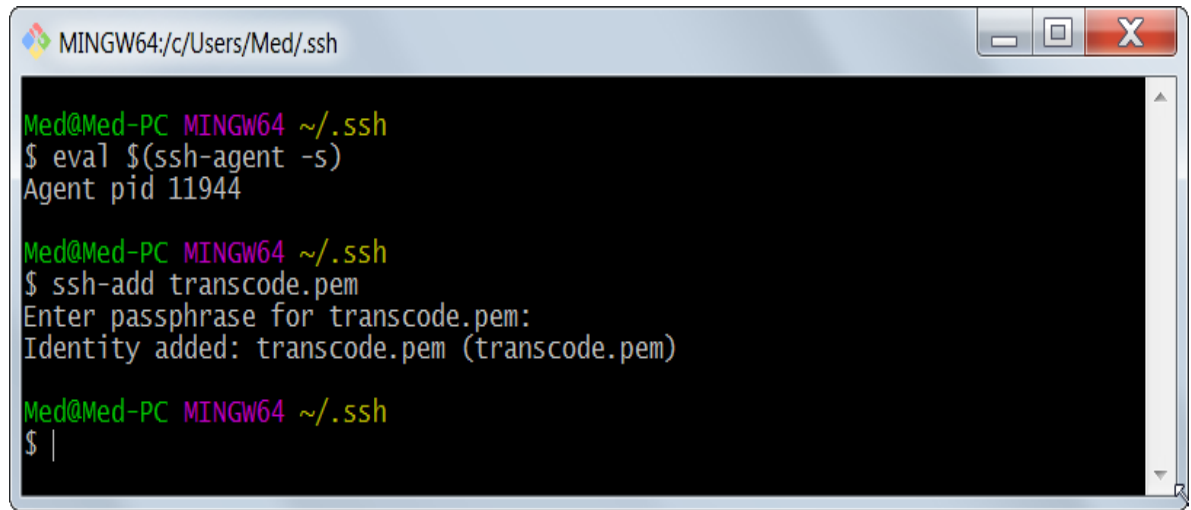
A screenshot of a terminal window titled "MINGW64:/c/Users/Med/.ssh". The terminal shows the prompt "Med@Med-PC MINGW64 ~/.ssh" and the command "\$ chmod 0600 transcode.pem |" being entered. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

Figure 24: security addition to the key.

After adding the security levels to the key, we can create an SSH agent to create the connection. Figure 25 shows the creation of the SSH agent and the association of the “transcode” key to the agent.

A terminal window titled 'MINGW64:/c/Users/Med/ssh' with standard Windows window controls. The terminal output shows the user 'Med@Med-PC' in a 'MINGW64' shell at '~/.ssh'. The first command is 'eval \$(ssh-agent -s)', which outputs 'Agent pid 11944'. The second command is 'ssh-add transcode.pem', which prompts for a passphrase and outputs 'Identity added: transcode.pem (transcode.pem)'. The prompt '\$ |' is visible at the end of the line.

```
MINGW64:/c/Users/Med/ssh
Med@Med-PC MINGW64 ~/.ssh
$ eval $(ssh-agent -s)
Agent pid 11944
Med@Med-PC MINGW64 ~/.ssh
$ ssh-add transcode.pem
Enter passphrase for transcode.pem:
Identity added: transcode.pem (transcode.pem)
Med@Med-PC MINGW64 ~/.ssh
$ |
```

Figure 25: adding password to the key.

Finally, the connection to the master instance is made via the command tools of the SSH library shown in Figure 26.

5.2.2 *Accessing the other instances*

The rest of the instances serve as the testing ground of the Python script. The transcoding of the video chunks will be performed there, as well as the copying operations.

To access the other virtual machines, we use the master instance as a hub or a tunnel. The master instance will be the mean of communication between the client and the whole virtual environment. Figure 26 shows how access to the master instance is done.

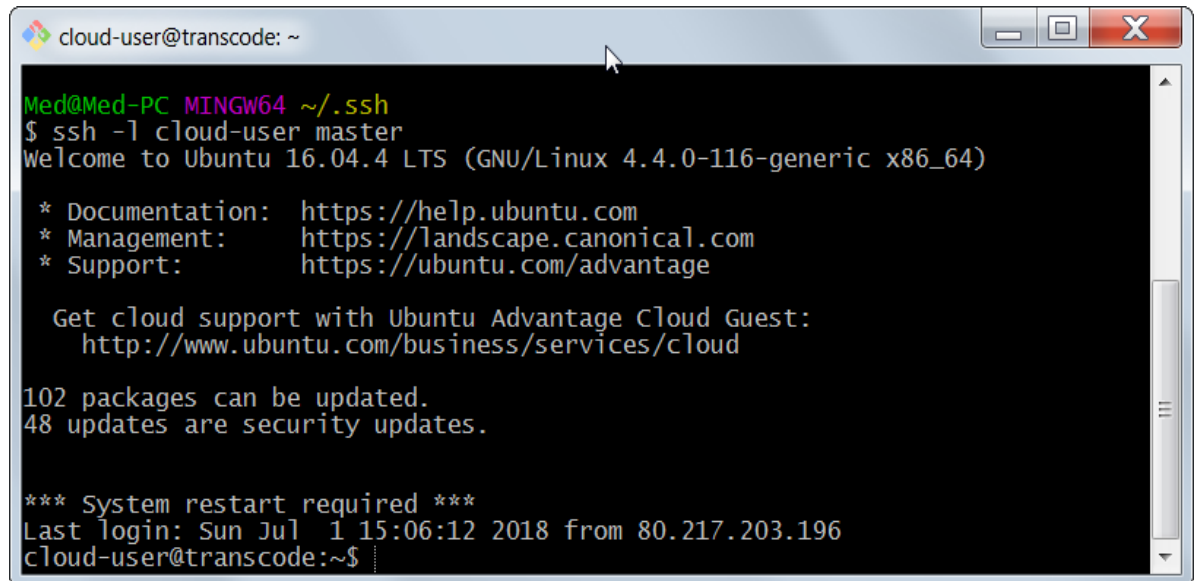
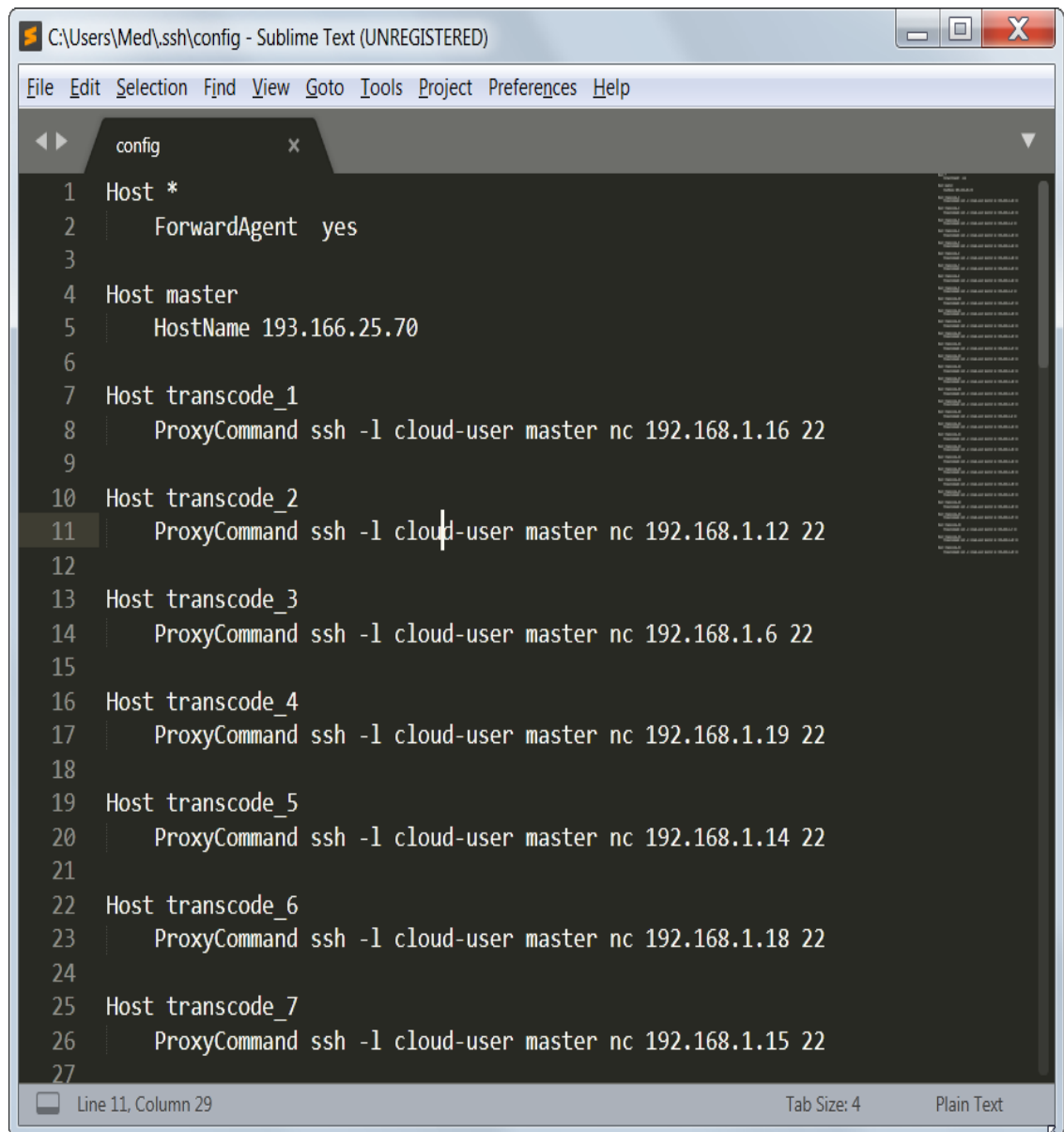
A terminal window titled 'cloud-user@transcode: ~' showing an SSH session. The user 'Med' on a 'Med-PC' (MINGW64) has executed 'ssh -l cloud-user master'. The terminal displays the Ubuntu 16.04.4 LTS login banner, including documentation, management, and support links. It also shows package update information: '102 packages can be updated. 48 updates are security updates.' and a system restart requirement. The last login is recorded as 'Sun Jul 1 15:06:12 2018 from 80.217.203.196'. The prompt returns to 'cloud-user@transcode:~\$'.

Figure 26: access to the master instance.

The configuration of the master instance to act as a tunnel is done using a Config file stored in the SSH folder. By placing the Config file in that folder, the required configuration is programmed automatically.

The Config file contains access rights to the other virtual machines, as well as NetCat forwarding. NetCat (often abbreviated to nc) is a computer networking utility for reading from and writing to network connections using TCP or UDP.

The configuration starts by setting the master instance as a Host Master, using its IP address, then set the other instances using NetCat as shown in Figure 27.



```
C:\Users\Med\.ssh\config - Sublime Text (UNREGISTERED)
File Edit Selection Fjnd Vjiew Goto Tools Project Preferences Help
config
1 Host *
2     ForwardAgent yes
3
4 Host master
5     HostName 193.166.25.70
6
7 Host transcode_1
8     ProxyCommand ssh -l cloud-user master nc 192.168.1.16 22
9
10 Host transcode_2
11     ProxyCommand ssh -l cloud-user master nc 192.168.1.12 22
12
13 Host transcode_3
14     ProxyCommand ssh -l cloud-user master nc 192.168.1.6 22
15
16 Host transcode_4
17     ProxyCommand ssh -l cloud-user master nc 192.168.1.19 22
18
19 Host transcode_5
20     ProxyCommand ssh -l cloud-user master nc 192.168.1.14 22
21
22 Host transcode_6
23     ProxyCommand ssh -l cloud-user master nc 192.168.1.18 22
24
25 Host transcode_7
26     ProxyCommand ssh -l cloud-user master nc 192.168.1.15 22
27
Line 11, Column 29 Tab Size: 4 Plain Text
```

Figure 27: The Config file commands.

6 EXPERIMENT AND RESULT

6.1 The script

The script used in the thesis work is presented in detail in this section, as well as the libraries and video-conversion techniques that were used. The script could be found in annex1.

6.1.1 FFMPEG

FFMPEG is open-source software; it contains many libraries and programs that perform video, audio and multimedia operations. It is based on command line processing of multimedia files and streams, and it is famously known to perform transcoding operations. Some of the other use cases of the FFMPEG libraries are trimming, concatenation and video scaling.

6.2 Video conversion

H264 is the most common video-compression method used in video distribution for the time being. It is oriented on the blocks of the videos, it can support up to 4k resolution, and it is based on video-coding format standard. H264 also called MPEG4 part 10 Advanced Video Coding “AVC” was created to provide decent video quality with slightly better/ lower bit-rate than the previous standard, and for this reason, the standard has been changed to be substantially flexible because of the high complexity of the video design.

H265 is the most recent standard; it is called High-Efficiency Video Coding (HEVC) and the next generation after H264. It is developed by the Joint Collaborative Team on Video Coding (JCT-VC). The goal for developing this new standard was to double the amount of compression efficiency compared to the previous standard.

6.2.1 *Running the script*

The script used for the thesis work is divided into four different sections, splitting the video, copy the chunk to the instances, transcode the chunks in the instances, then copy back to the master instance to merge.

Every step of the script is done in parallel; the parallelism, in this case, is used to make sure that we take advantage of the powerful virtual machines provided to this thesis work. Figure 28 shows the limit summery of the instances used in the project, providing the number of instances used, the VCPUs, the amount of RAM, Floating IPs and the number of security groups created.



Figure 28: The limit summary of the instances.

Figure 29 illustrates the different specs for each instance used, the flavor name, and the RAM and VCPU amount.

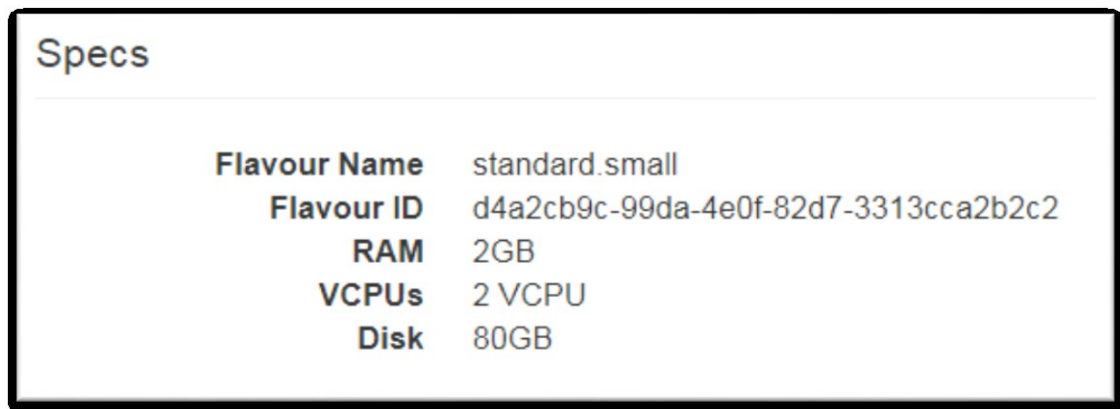


Figure 29: The specs of the instances.

The script is written in Python using threads; Python was chosen because it is easy to install and straightforward when creating the new instances. The instances are used in every step of the script. As mentioned before, the script is divided into five main sections: split, copy to the instances, transcode, copy back to the master instance, merge.

- The splitting

The splitting step is done in parallel on the master instance, and it depends on the number of available virtual machines. When using for example four virtual machines, the input video is split into 4 quasi-equal chunks. Using multithreads, the video could be split in a shorter time compared to using one thread.

The script, in this case, uses the FFMPEG library to split the video to the number of the chunks needed; this is done using the syntax below:

```
“ffmpeg -i Video_Name.mp4 -vcodec copy -acodec copy -ss xx -t xx output_name.mp4”
```

- Copy the chunks to the VMs

The next step is to copy the created video chunks into the virtual machines. The copy is also done in parallel, and it is done using the “scp” command in line 16 of the script. SCP allows files to be copied to/from different hosts. It uses SSH for the data transfer and provides the same authentication with the same level of security as SSH.

The SCP command is used as it is shown below:

```
“scp sample.mp4 cloud-user@ip:~/FFMPEG ”
```

- Transcode

The transcoding of the chunks is the most important step in the script, and it is executed in parallel; the parallelism is done on the instances (virtual machines). FFMPEG libraries are used for the transcoding, and the script is calling the FFMPEG as shown in line 26 of the script.

```
ssh -l cloud-user "+ ip +" \"cd FFMPEG && ffmpeg -i "+ sample + ".mp4 -vf scale=-1:576 -vcodec libx265 -strict -2 "+output+".mp4 && exit\"”
```

In this example, the script transcodes the chunk called “sample.mp4” from h264 into h265 compression standard, the output is called “output.mp4”.

- Copying back

After transcoding the chunks into the new H265 encoding standard, they are copied back the master instance where they are merged. The use of “scp” command is done in this case in reverse as it is shown below:

```
"scp cloud-user@" + ip + " :~/FFMPEG/" + sample + ".mp4 ~/FFMPEG"
```

IP: the IP address of the instance.

Sample: the name of the output from the transcoding step.

- Merge

After all the transcoded chunks are copied to the master instance, the last part of the script is to merge all of them into one video. The merging script also uses the FFMPEG libraries to merge the chunks, as well as a “txt” file that contains the names of the input files and their extensions.

The whole process of the script is described in Figure 30 below.

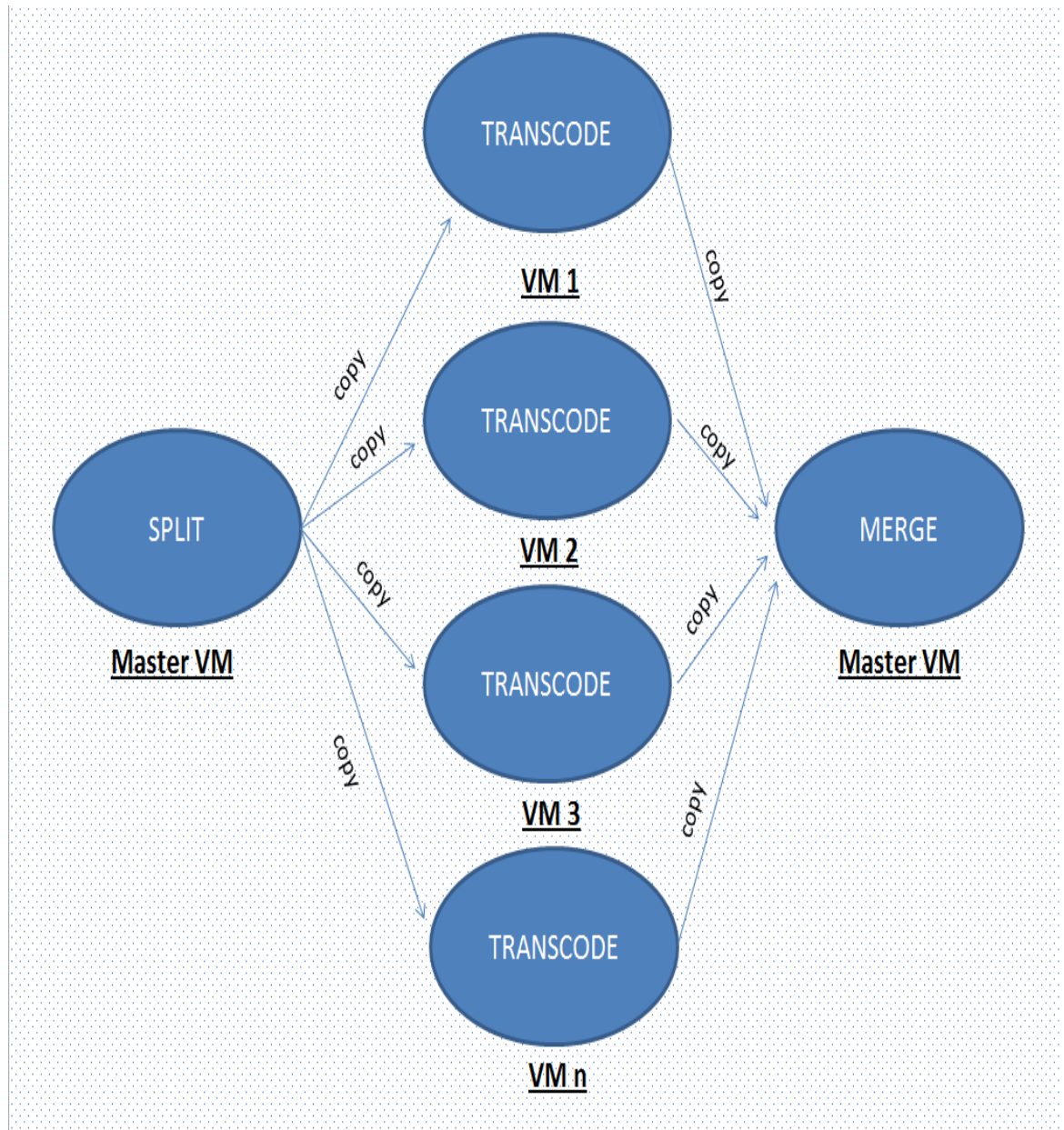


Figure 30: The steps performed by the script.

The video used in the experiment is a 10 min video with H264 codec, 1280 frame width, 720 frame height, and 25 frames per sec. More information is shown in Figure 31 below.

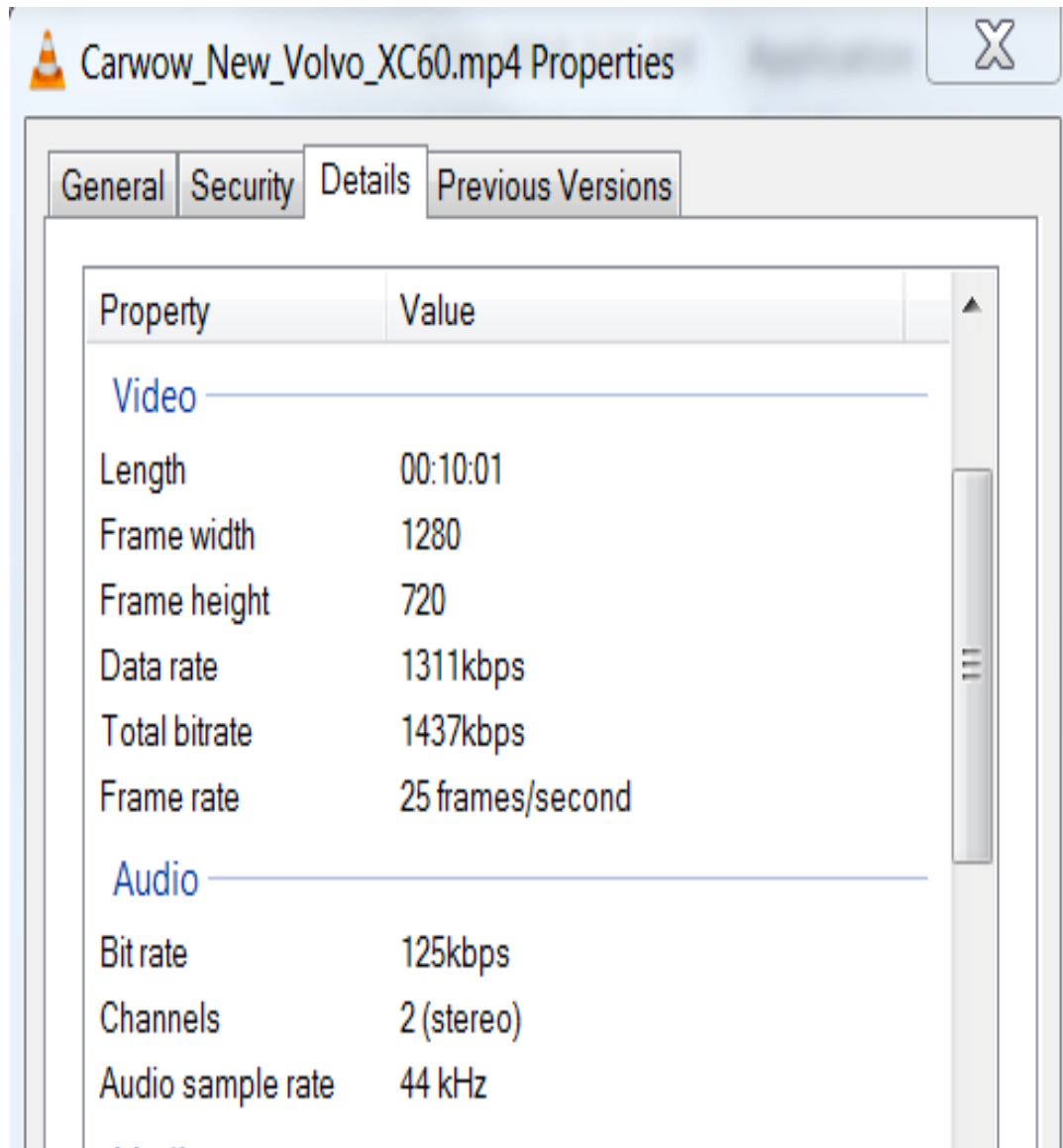


Figure 31: Details on the video used for the experiment.

6.3 The result of the experiment

The result of the experiment is presented in this chapter, together with charts and tables explaining in detail all the aspects of the result.

6.3.1 Results

The following section describes the results of the experiment; a detailed description will be shown for every step of the algorithm.

6.3.1.1 Splitting the video

The first step on the execution of the script is the splitting part, which is a command to split the input video into chunks of the same size. The nature of the mp4 videos does not allow the FFMPEG script to split the video into same-sized segments correctly, and that is because of the nature of the frames of the mp4 video, and how they are assembled.

The result of the splitting part is shown in Figure 32 below.

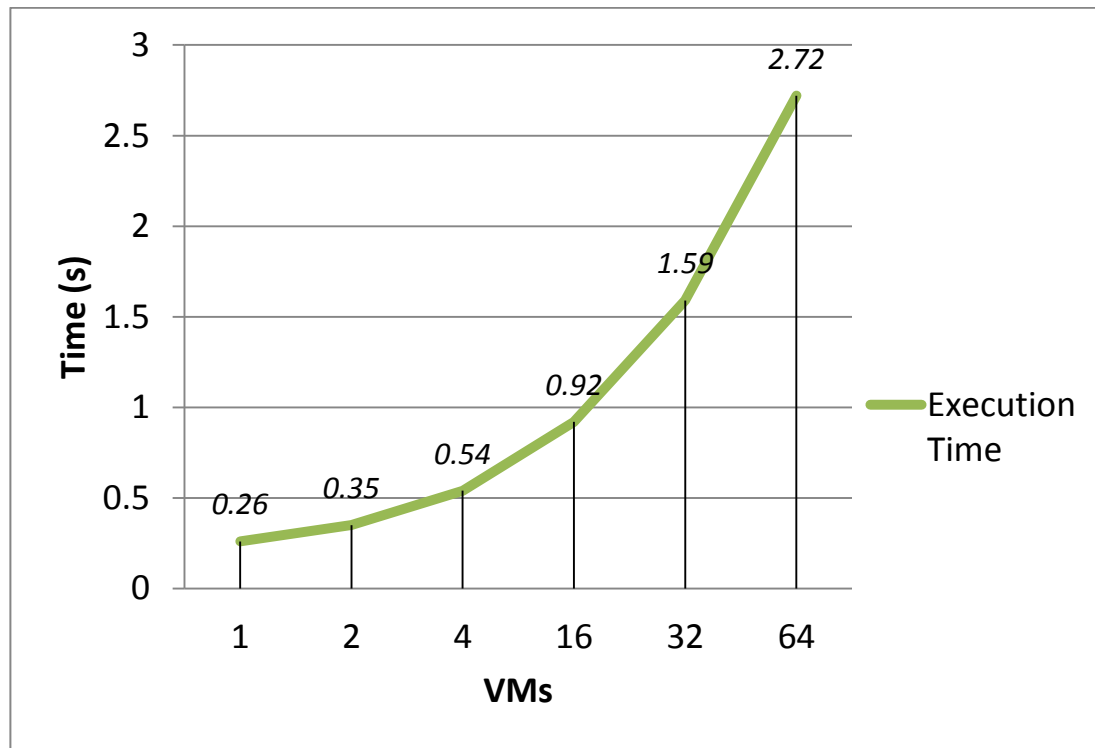


Figure 32: The execution time of the splitting script.

In each case, the X-axis shows the number of virtual machines and the Y-axis shows the execution time of the algorithm. As mentioned previously, the number of virtual machines is equal to the number of video chunks. As shown in Figure 31, although the size of the videos decreases with the increase in the number of virtual machines, the time spent to split the input video into smaller segments becomes higher, this shows that using FFMPEG to split the video is a rather time-consuming task, since it adds more overheads as the number of the chunks increases.

6.3.1.2 Copying the chunks

The second part of the script consists of copying the video chunks into the virtual machines. Briefly, it is performed by the “scp” command from the host to the virtual machines. This command uses the IP addresses of the virtual machines to access them. The mentioned IP addresses are stored in one config file which is located in a folder named “.SSH”. This file contains the IP addresses of the master host and the virtual machines. The config file enables the “scp” command to reach the virtual machines.

Figure 33 shows the execution time of the copying script.

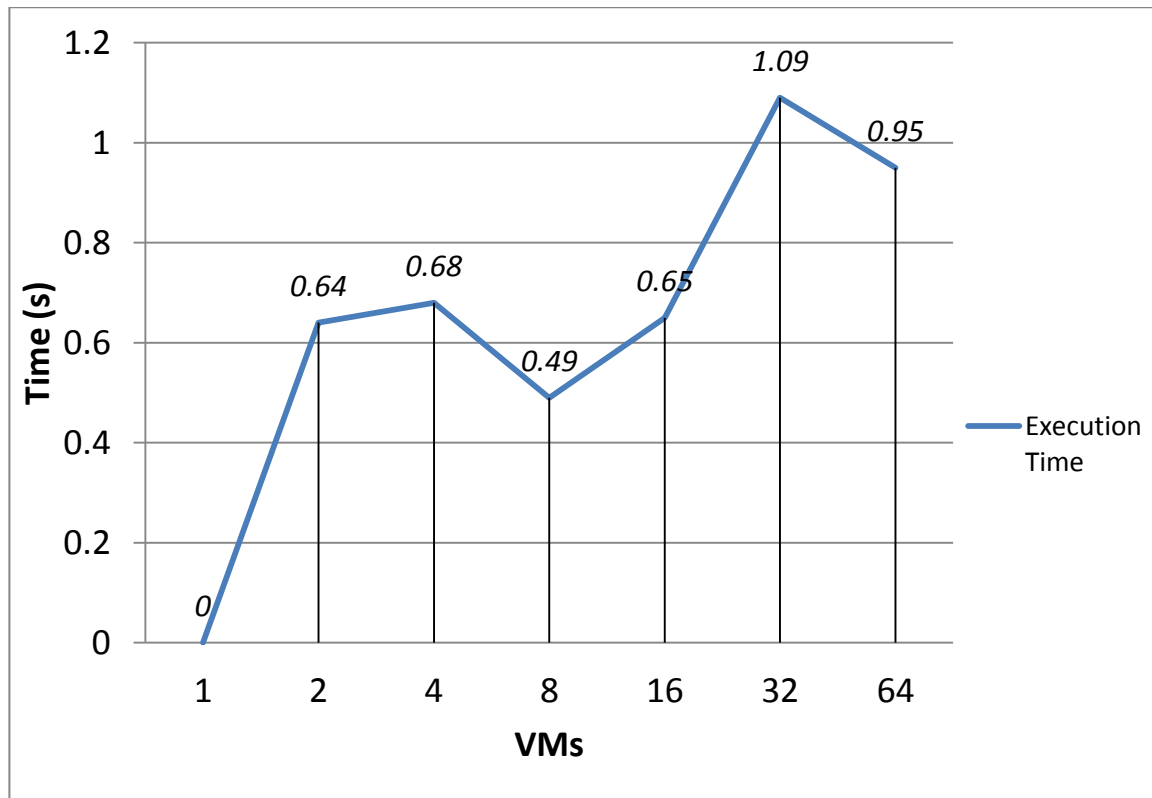


Figure 33: The execution time of the copying script.

Here, the two axes represent the same variables as shown in Figure 31. As shown in Figure 32, there are some fluctuations as the number of the virtual machines increases, but there is an overall rise in the required execution time by the script. The fluctuations reflect the unequal video segments encoded in an mp4 format, which has been explained before. This mp4 characteristic is shown when the test is run on eight virtual machines.

6.3.1.3 Transcoding

Transcoding is the most important phase of the script, the transcoding of the video chunks. As explained before; the chunks are copied to the virtual machines, which in turn are transcribed using the FFMPEG library to another coder format. By increasing the number of virtual machines, the time needed for the transcoding shrinks by a factor of two; this could be noticed in the experiment when using four virtual machines. The time consumed by the FFMPEG script is “04:31.10”, but when using eight virtual machines, the time decreases to “02:12.7” minutes. The trend continues as more virtual machines are added.

Figure 34 shows the execution time of the transcode script.

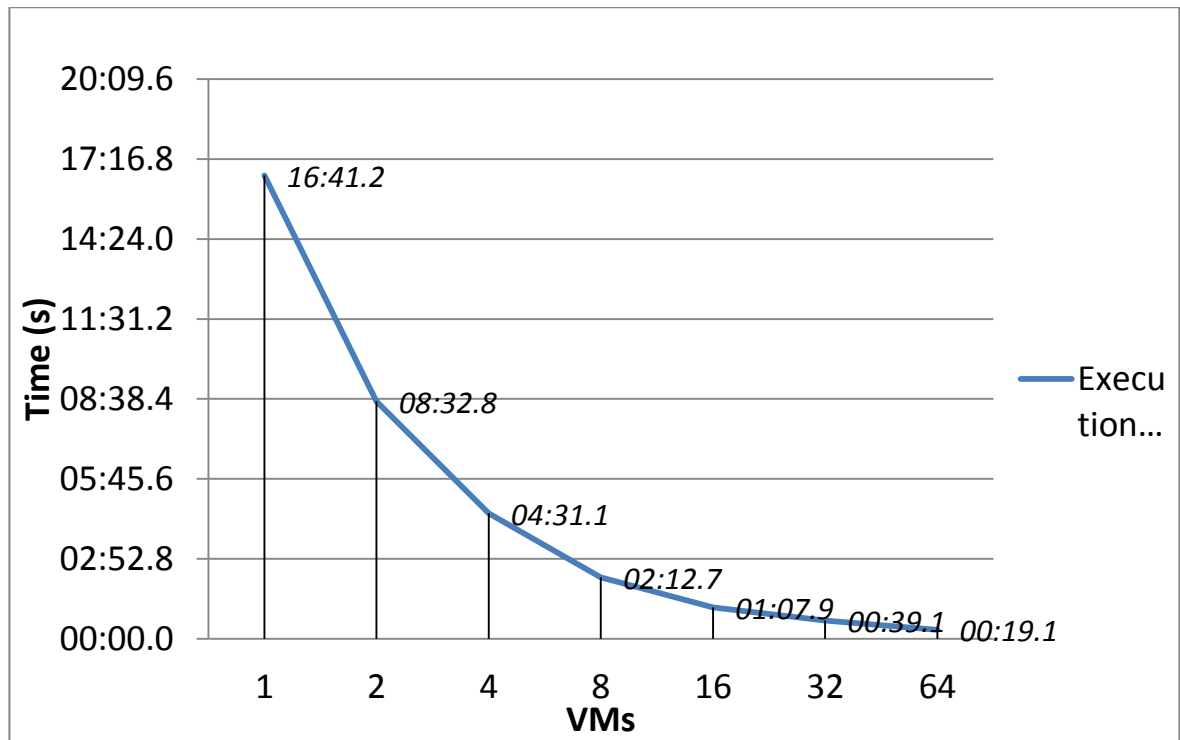


Figure 34: The execution time of the transcode script.

The maximum number of the virtual machines that were used in the project is 64 VMs, and yet there is a considerable decrease in time spent to transcode the video chunks by the FFMPEG script.

6.3.1.4 Copy back

The fourth step of the script is to copy back the transcoded video chunks to the master machine. Comparing the time consumed to copy the video chunks to the VMs and the time spent to copy the transcoded chunks to the master machine, we can see that the latter takes less time, and that is due to the smaller size of the transcoded videos.

Figure 35 shows the execution time of the copying script.

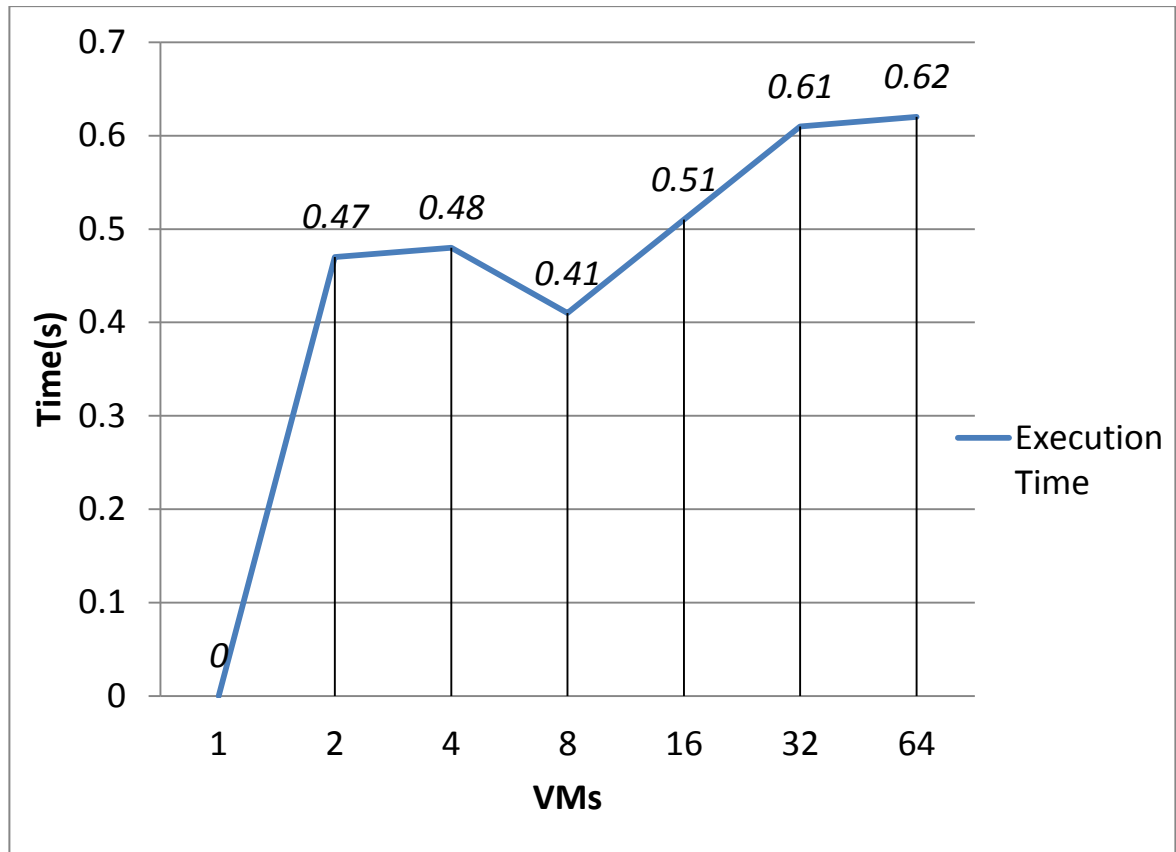


Figure 35: The execution time of the copying script.

6.3.1.5. Merge

Merging the video chunks is the last step of the script; it is where all the video chunks get merged using the FFMPEG script to create a high-quality video. The merging process is done on the master machine, where the splitting takes place. As the number of video chunks increases, a slight increase in the time spent to merge could be observed, this is reasonable because of the amount of the operations that are performed to merge increases when we have a higher number of videos.

Figure 36 shows the execution time of the merge script.

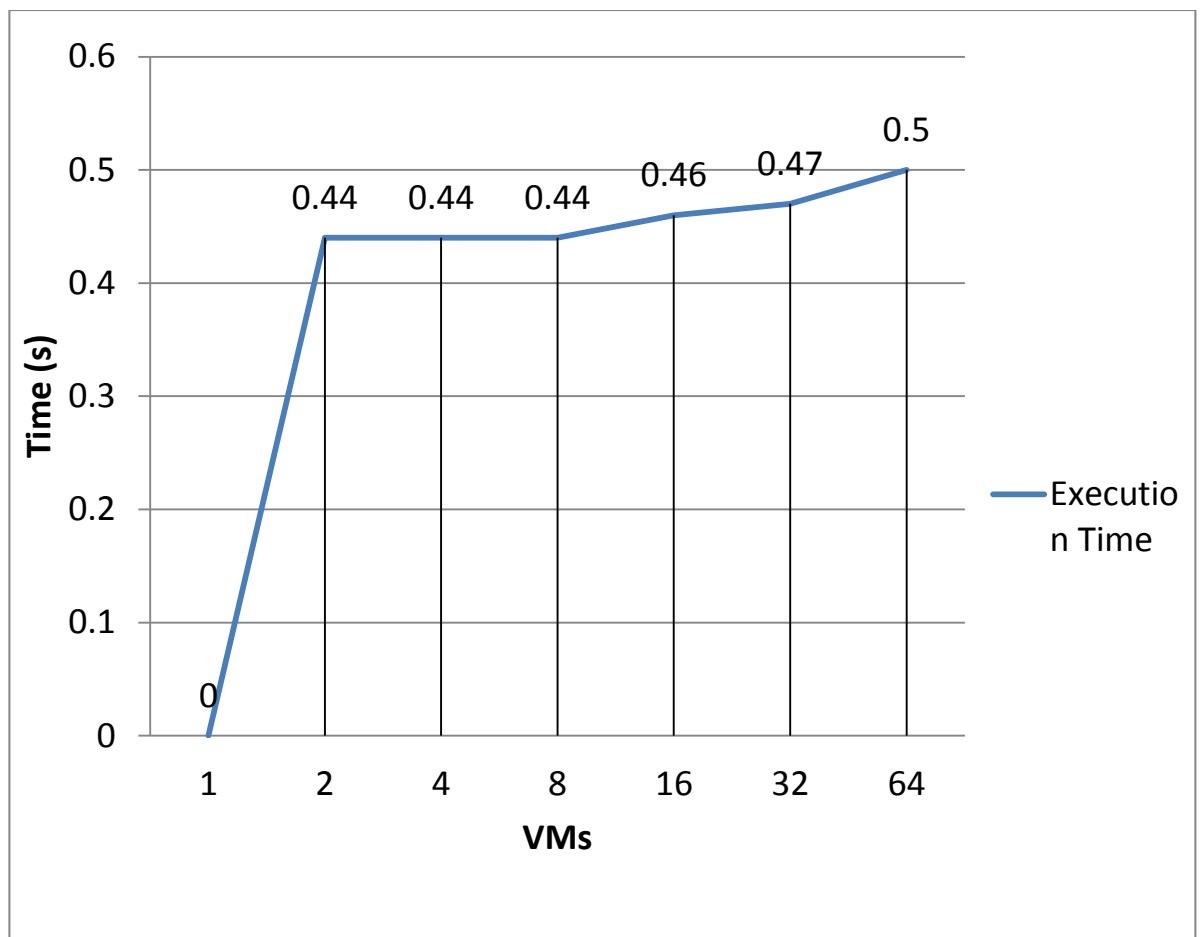


Figure 36: The execution time of the merge script.

6.3.1.6. The total time

As expected from the experiment, the execution time of the script is decreasing while adding more virtual machines; this is noticeable when going from 1 virtual machine to 2 VMs, where the timings are “16:41.2” and “08:34.6” consecutively as shown in Figure 37. The same rate of change is visible when shifting to a higher number of virtual machines each time.

The main factor of this increase is the computing power added while testing and the fact that the script is made to be run in parallel using all the virtual machines. However, when breaking down the results of each step of the algorithm, some interesting findings can be noticed.

As shown in Figure 37, after doubling the number of virtual machines in every step of the experiment, the execution time is almost cut by half. This behavior continues all the way to the point when 64 virtual machines are used.

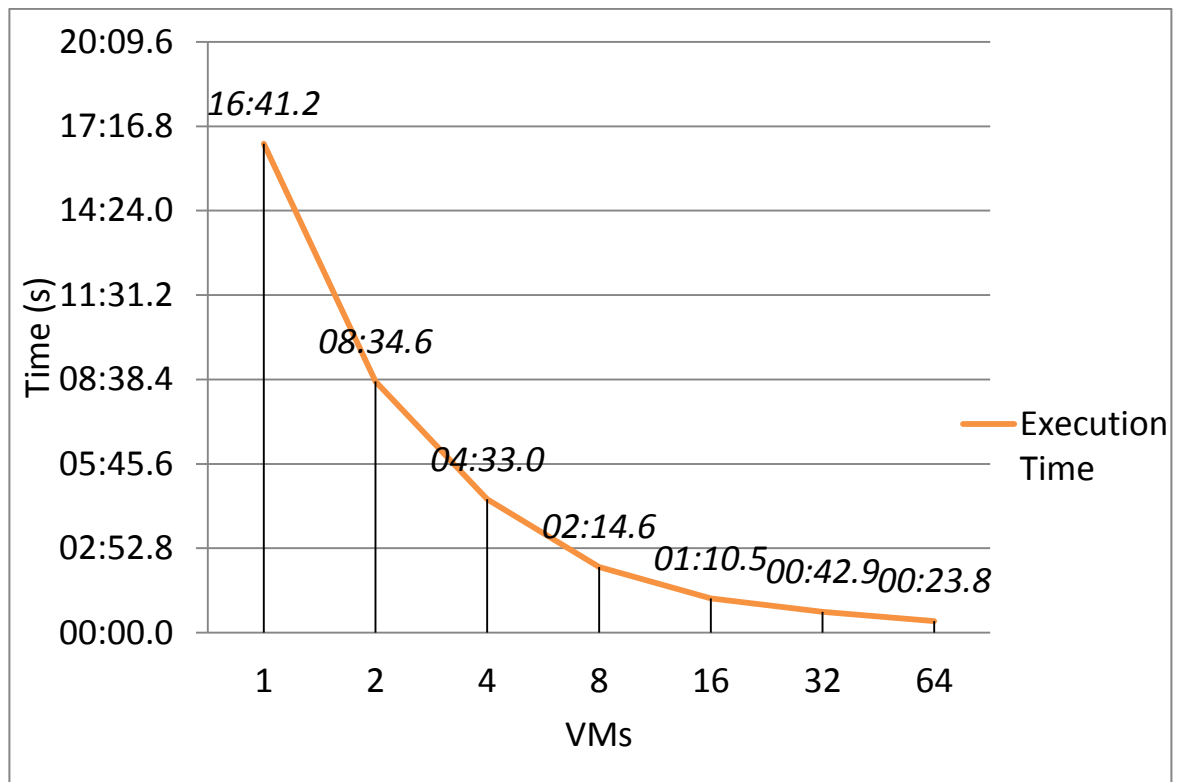


Figure 37: The total execution time of the script.

7 CONCLUSION AND FUTURE WORK

Throughout the thesis work, the discussion was mainly about the elaboration of basic notions and terms of cloud computing, the meanings and different types of the cloud platforms that exist in today's market, the choices that have been made for the most optimal solution for the experiment at hand, finishing by setting up the actual experiment with the right platform. Many of these steps and decisions have been taken in accordance to what is needed for the experiment, taking into account the availability of the resources which could be allocated to the experiment, and the budget that was decided for the thesis work by the university. Such decisions required extensive research and discussions with my supervisor throughout the thesis work. The final results are a reflection of what has been achieved with the available resources, and the reasoning of how the experiment should be conducted.

The thesis work resulted in a somewhat expected behavior of the transcoding script, i.e., the time spent to transcode the video chunks decreases once a new set of virtual machines (computing power) is added to the experiment. The decrease is not hundred percent linear as one might expect, but it is a noticeable decrease. It is important to mention that the experiment started with one virtual machine in the first place, in order to have an anchor and a point of reference. The experiment continued by doubling the number of virtual machines, so as the next step, the second virtual machine, was added to the experiment. Increasing the number of VMs resulted in a very noticeable decrease of the transcoding time. Doubling the number of virtual machines each time a new experiment was conducted and it showed some impressive results that will be discussed further in this chapter.

The decreasing execution time of the experiment is, as discussed before, due to the increase of computing power, hence the fact that the bigger the number of virtual machines the less execution time the script needs. At the same time, the execution time of subtasks other than transcoding seems to increase slightly by adding VMs. Taking for example, the timing result from the splitting script, we can notice there an increase from 0.26 seconds when using one virtual machine to roughly 3 seconds when reaching 64 virtual machines. This behavior is also noticed while merging the video chunks coming from different virtual machines. In this example, what caught the attention is the increase of the time spent to merge video chunks when using two virtual machines, which was 00:44 seconds, to 00:50 seconds when using 64 virtual machines. These overheads are also observed in the operation of copying the video chunks into other virtual machines. This step is crucial and is also dependent; the dependency of this operation is on the network speed if the clouds are situated in the multiple geo-distributed data centers, or the hardware speed if the virtual machines are situated in the same data center or the same machine. As observed in Figure 32, the time spent to copy the chunks increases when we reached 64 virtual

machines compared to when the experiment was conducted using two virtual machines, this overhead is very important to be discussed in detail and for a different type of cloud situations. For future work, with much more computing power and more clouds computing platform choices, these types of overheads could be calculated with more granularities.

All in all, the experiment showed that the more the number of virtual machines the less execution time the general transcoding of the input video requires. It also shows the increase of overheads in the other tasks without exception. With 64 virtual machines, the impact of overheads is roughly observed, and the amount of time spent in those overheads could be negligible.

For the experiment at hand, a point to discuss could also be the limited computing power appointed to the experiment, the script is only using from the range of one virtual machine to 64 VMs, and the virtual machines have the following set of specification, 2 GB of RAM, 2 virtual CPU, 80 GB of SSD storage. They are all created on the same server.

Some points to improve could be:

- Add more virtual machines to the setup, in order to get to the point to observe a clear manifestation of the overheads discussed in the previous chapters.
- Add virtual CPUs to the virtual machines, to improve the computing power of the system.
- Add more RAM.
- Experiment with different situations:
 - Large data centers.
 - Across multiple geo-distributed data centers.
 - Same server with more power than the one used in the current experiment.
- To investigate adding cores to the virtual machines. Together with increasing the number of virtual machines; one could calculate the optimal amount of power needed to transcode specific videos.

BIBLIOGRAPHY

- [1] White paper: Cisco VNI Forecast and Methodology, 2015-2020 [online]. Cisco official website.URL: <http://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/visualnetworking-index-vni/complete-white-paper-c11-481360.html>.Accessed 19 August 2016.
- [2] YouTube, LLC, www.youtube.com, [Accessed June 2018]
- [3] Dailymotion, www.dailymotion.com [Accessed July 2018]
- [4] Instagram, www.instagram.com [Accessed August 2018]
- [5] Snapchat, www.snapchat.com, [Accessed June 2018]
- [6] Hongliang Yu, Dongdong Zhng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large scale-video-on-demand systems. SIGOPS Oper. Syst. Rev., April 2016
- [7] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, youtube, everybody tubes: analyzing the world’s largest user generated content video system. IMC ’07, San Diego, California, USA.
- [8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud consuming and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 25(6):599–616, June 2009.
- [9] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In 2008 Grid Computing Environments Workshop, pages 1–10, Nov 2008.
- [10] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.
- [11] What is Cloud Computing?". Amazon Web Services. 2018-03-05. Retrieved 2018-03-06.
- [12] Shujia Zhou* , Ben Kobler, Dan Duffy, Tom McGlynn NASA Goddard Space Flight enter{shujia.zhou, benjamin.kobler, daniel.q.duffy. Case Study for Running HPC Applications in Public clouds.

- [13] A. Vetro, C. Christopoulos, and H. Sun, 'Video transcoding architectures and techniques: an overview,' *Signal Processing Magazine, IEEE*, vol. 20, no. 2, pp. 18 - 29, mar 2003.
- [14] Fei Xu, Fangming Liu, Member IEEE, Hai Jin, Senior Member IEEE, and Athanasios V. Vasilakos, Senior Member IEEE, *Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions*.
- [15] *Xen Users' Manual*, Citrix Systems, Inc., University of Cambridge, U.K., XenSource Inc., IBM Corp., Hewlett-Packard Co., Intel Corp., AMD Inc., 2008. [Online]. Available: <http://bits.xensource.com/Xen/docs/user.pdf>
- [16] Nvidia Corporation, www.nvidia.com [Accessed April, 2017]
- [17] Intel Corporation www.intel.com [Accessed August, 2018]
- [18] <https://cloudxchange.io/cloud-managed-services-for-public-clouds/>
- [19] <https://www.wikitechy.com/cloud-computing/cloud-types/3-types-of-clouds>
- [20] <http://www.paranet.com/blog/bid/128265/The-Four-Types-of-Cloud-Computing-Models>
- [21] <https://weblogs.asp.net/lduveau/iaas-vs-paas-vs-saas>
- [22] Uber Technologies Inc, www.uber.com [Accessed Mars, 2018]
- [23] Lyft, www.lyft.com [Accessed Mars, 2018]
- [24] <https://insidehpc.com/2011/02/survey-results-on-cloud-iaas-providers/>
- [25] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10, Nov 2008.
- [26] <http://www.rtc.us.es/>
- [27] <https://analyzingidentity.com/2010/06/08/the-anywhere-application-architecture/>
- [28] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, youtube, everybody tubes: analyzing the world's largest user generated content video system. *IMC '07*, San Diego, California, USA.
- [29] <https://www.westcoastcloud.co.uk/cloud-vendors/microsoft/azure/>
- [30] <https://www.openstack.org/software/>

[31] <https://docs.openstack.org/nova/queens/user/architecture.html>

[32] <https://www.addictivetips.com/windows-tips/best-ssh-clients/>

[33] Spotify, www.Spotify.com, [Accessed Mars, 2018]

[34] Foursquare, www.foursquare.com [Accessed April, 2018]

[35] <https://www.chargify.com/blog/xaas-everything-as-a-service/>

[36] <https://www.youtube.com/channel/UCFv-76jNZIBFp6O9umdneyDA>

ANNEX 1

```

1. import time
2. import datetime
3. import os
4. from threading import Thread
5.
6. def splitVid(output, part, start, end):
7.     exe = "time ffmpeg -loglevel panic -i Carwow_New_Volvo_XC60.mp4 -
vcodec copy -acodec copy -ss "+ start +" -t " + end + " " + output + ".mp4"
8.     print "-" * 100
9.     now = datetime.datetime.now()
10.    os.system(exe)
11.    print "SPLITTING THE PART " + part + " IN : ", datetime.datetime.now() - now
12.    print "-" * 100
13.
14.
15. def copyVid(sample, ip, part):
16.    exe = "time scp "+ sample + ".mp4 cloud-user@" + ip + ":~/FFMPEG "
17.    print "-" * 100
18.    now = datetime.datetime.now()
19.    os.system(exe)
20.    print "COPYING OF THE VIDEO FILE "+ part + " to VM" + part + " TIME IS.....
.. : ", datetime.datetime.now() - now
21.    print "-" * 100
22.
23. def transVid(ip, sample, output):
24.    print "-" * 100
25.    now = datetime.datetime.now()
26.    exe = "time ssh -l cloud-user "+ ip +" \"cd FFMPEG && ffmpeg -
i "+ sample + ".mp4 -vf scale=-1:576 -vcodec libx265 -strict -
2 "+output+".mp4 && exit\""
27.    retvalue = os.system(exe)
28.    print "TRANSCODING TIME for "+ sample + " IS..... ", datetime.datetime.now(
) - now
29.    print "-" * 100
30.
31. def copyBack(ip, sample, part):
32.    print "-" * 100
33.    now = datetime.datetime.now()
34.    exe = "time scp cloud-user@"+ip+":~/FFMPEG/"+sample+".mp4 ~/FFMPEG"
35.    os.system(exe)
36.    print "COPYING BACK THE RESULT part "+part+" IN ", datetime.datetime.now() -
now
37.    print "-" * 100
38.
39. def startEnd():
40.    t1.start()
41.    t2.start()
42.    t3.start()
43.    t4.start()
44.    t1.join()
45.    t2.join()

```

```

46.     t3.join()
47.     t4.join()
48.
49. #split the video
50. t1 = Thread(target=splitVid, args=("sample1", "1", "00:00:00", "00:02:30"))
51. t2 = Thread(target=splitVid, args=("sample2", "2", "00:02:30", "00:02:30"))
52. t3 = Thread(target=splitVid, args=("sample3", "3", "00:05:00", "00:02:30"))
53. t4 = Thread(target=splitVid, args=("sample4", "4", "00:07:30", "00:02:30"))
54. startEnd()
55. print "Exiting Main Thread"
56. # copy samples to VMs
57. t1 = Thread(target=copyVid, args=("sample1", "192.168.1.16", "1"))
58. t2 = Thread(target=copyVid, args=("sample2", "192.168.1.12", "2"))
59. t3 = Thread(target=copyVid, args=("sample3", "192.168.1.6", "3"))
60. t4 = Thread(target=copyVid, args=("sample4", "192.168.1.19", "4"))
61. startEnd()
62. print "Exiting Main Thread"
63. # transcode the samples
64. t1 = Thread(target=transVid, args=("192.168.1.16", "sample1", "output1"))
65. t2 = Thread(target=transVid, args=("192.168.1.12", "sample2", "output2"))
66. t3 = Thread(target=transVid, args=("192.168.1.6", "sample3", "output3"))
67. t4 = Thread(target=transVid, args=("192.168.1.19", "sample4", "output4"))
68. startEnd()
69. print "Exiting Main Thread"
70. # copy back the samples
71. t1 = Thread(target=copyBack, args=("192.168.1.16", "output1", "1"))
72. t2 = Thread(target=copyBack, args=("192.168.1.12", "output2", "2"))
73. t3 = Thread(target=copyBack, args=("192.168.1.6", "output3", "3"))
74. t4 = Thread(target=copyBack, args=("192.168.1.19", "output4", "4"))
75. startEnd()
76. print "Exiting Main Thread"

```