2. Background

Today's web is filled with browser-based applications that are used regularly by millions of users. Some applications have to deal with billions of server requests on a daily basis, and these numbers keep growing. Such applications need to be designed to scale, to expand onto improved and/or additional hardware, and to do this transparently (or at the least without having to take down the application for maintenance). The hardware that a web application runs on is an important component when it comes to dealing with large-scale applications. Applications such as Gmail, YouTube and Flickr that run across thousands of machines are good examples since they require different types of hardware, e.g. web-servers, databases, etc. However, as the number of users grows, the cost to keep the application up and running increases dramatically. The cost for maintenance, physical space, cooling, power, operations, increases for each server that is added to the environment, even if its resources are not fully used.

With very large applications, such as YouTube, that typically have more than 2 billion videos watched in a month, the costs associated with maintaining the required infrastructure is unpredictable. To manage this infrastructure, there are basically two options:

1. To provide resources based on peak situations, which basically means that, for the most part, resources will be idle;
2. Provide resources based on the average number of requests, which means that in some situations the servers will be overloaded and the quality of service will be affected.

None of the above alternatives is good from a business perspective. The first one is very expensive, as the cost is basically associated with the price of hardware itself, and not with its usage. In the second one, the quality of service may be impacted, and, in the long run, it may signify loss of clients and/or business opportunities.

In this scenario Cloud Computing appears as an interesting alternative, as its "everything as a service" model provides an economically attractive solution to demand variations.

2.1 Cloud Computing Paradigms

For the purpose of this thesis, Cloud Computing is defined as an Internet-based computing, where there is a large group of interconnected computers (cloud), that share their resources, software, and information (computing), on demand, according to the user needs [4]. Vaquero et al. attempt to pin down a suitable definition for clouds that describes how they differ from grids. Their proposed definition is thorough, but verbose:

"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs." [4]

In creating their definition, Vaquero et al. studied definitions from numerous experts, which featured many attributes of cloud computing such as immediate scalability, optimal usage of resources, pay-as-you-go pricing models, and virtualized hardware and software.

Cloud computing is a paradigm shift following the shift from mainframe to client–server in the early 1980s. Details are abstracted from the users, who no longer have need expertise in, or control over, the technology infrastructure "in the cloud" that supports them [5]. Cloud computing describes a new supplement, consumption, and delivery model for IT services based on the Internet, and it typically involves over-the-Internet provision of dynamically scalable and often virtualized resources [6,7].

Cloud computing infrastructures are typically broken down into three layers [4, 61]: "software", "platform" and "infrastructure". Each layer serves a different purpose and offers different products to businesses and individuals around the world, and, conversely, every layer can be perceived as a customer of the layer below [61].

The Software as a Service (SaaS) layer offers service based on the concept of renting software from a service provider, rather than buying it yourself. It basically refers to providing on-demand applications over the Internet. The software is hosted on centralized network servers and made available over the web or, also, intranet. Also known as "software on demand" it is currently the most popular type of cloud computing by offering high flexibility, enhanced scalability and less maintenance. Yahoo mail, Google docs, Flickr, Google Calendar are all instances of SaaS. With a cloud-based email service, for example, all that one has to do is register and login to the central system to start to send and receive messages. The service provider hosts both the application and data, so the end user is free to use the service from anywhere. SaaS is very effective in lowering the costs of business as it provides the business an access to applications at a cost normally less expensive than a licensed application fee. This is only possible due to its monthly fees based revenue model. With SaaS, users no longer need to worry about installation or upgrades [62].

Platform as a Service (PaaS) offers a development platform for developers. End users write their own code and the PaaS provider uploads that code and presents it on the web. SalesForce.com's is an example of PaaS. PaaS provides services to develop, test, deploy, host and maintain applications in the same integrated development environment. It also provides some level of support for the design of software applications. Thus, PaaS offers a faster and usually cost effective model for software application development and deployment. The PaaS provider manages upgrades, patches and other routine system maintenance. PaaS is based on a metering or subscription model so users only pay for what they use. Users take what they need without worrying about the complexity behind the scenes [42].

There are basically four types of PaaS solutions – *social application platforms*, *raw compute platforms, web application platforms* and *business application platforms* [43]. Facebook is a type of social application platform wherein third parties can write new applications that are then made available to end users.

The final layer in the cloud computing stack is the infrastructure. Infrastructure as a Service (IaaS) is the process in which computing infrastructure is delivered as a fully outsourced service. Some of the companies that provide infrastructure services are IBM, Amazon.com, among others. Managed hosting and provision of development environments are the services included in the IaaS layer.

The user can buy the infrastructure according to his or her requirements at any particular point of time, instead of buying an infrastructure that may not be used for months. IaaS operates on a “Pay as you go” model, ensuring that the users pay for only what they are using. Virtualization enables IaaS providers to offer almost unlimited instances of servers to customers, and make use of the hosting hardware cost-effective. IaaS users enjoy access to enterprise grade IT Infrastructure and resources, that might be very costly if otherwise purchased. Thus, dynamic scaling, usage based pricing, reduced costs and access to premium IT resources are some of the benefits of IaaS. IaaS is also sometimes referred to as Hardware as a Service (HaaS). An Infrastructure as a Service offering also provides maximum flexibility because just about anything that can be virtualized can be run on these platforms. This is perhaps the biggest benefit of an IaaS environment. For a startup or small business, one of the most difficult things to do is keep capital expenditures under control. By moving the computational infrastructure to the cloud, one has the ability to scale up and down as needed.

In next section, we detail the Amazon Web Services Cloud Platform, as well as its main services, that provided the basic infrastructure for the video processing architecture proposed in this thesis.

2.2 Amazon Web Services Platform

Amazon Web Services (AWS) [48] is a group of cloud-based services provided by Amazon that differs from traditional hosting since its resources are charged by actual usage. These services provide cloud-based computation, storage and other functionality that enable organizations and individuals to deploy applications and services on an on-demand basis and at commodity prices [61]. The AWS platform is composed by several services that complement one another. Elastic Compute Cloud (EC2), for processing, the Simple Storage Service (S3), for binary storage, SimpleDB, for structured data storage, Relational Database Service (RDS), for relational databases, Cloud Front, for content delivery, are examples of such services. For the purposes of this project, EC2 and S3 are the most relevant services.

Amazon EC2 is a web service interface that provides resizable computing capacity in the cloud. Based on IaaS model, it allows a complete control of computing resources and reduces the time required to obtain and boot new server instances. Users of EC2 can launch and terminate server instances on a matter of minutes, as opposed to delays of several hours, days or weeks, typical of traditional hardware solutions. This feature is particularly interesting because it allows applications to quickly scale up and down their processing resources, as computing requirements change, while in the traditional hardware approach, it can take several weeks or even months to get a new server running.

Amazon EC2 provides developers with two APIs for interacting with the service, allowing intances administration operations, such as start, stop, reboot, query information, etc. One is the Query API in which operations send data using GET or POST methods over HTTP or HTTPS. The other is the SOAP [46] API in which operations send data using SOAP 1.1 over HTTPS.

The main concept behind EC2 is that of a server instances [8]. There are a number of different types of instances that users can choose from, divided into six categories: standard, micro, high-memory, high-CPU, cluster and cluster-GPU. Each type has several subtypes, with various levels of processing power, memory and storage, and users can choose between them according their needs.

Because AWS is built on top of heterogeneous hardware processing power, a standard measure Amazon EC2 Compute Units is used. One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. [9].

The storage provided on an instance (referred to by Amazon as “instance storage”) is volatile. Data will survive the instance rebooting, either intentionally or accidentally, but it will not survive an underlying hard drive failing or an instance being terminated. It is also possible to choose a non-volatile storage for the instance, called Elastic Block Storage (EBS). Amazon EBS allows users to create storage volumes from 1 GB to 1 TB that can be mounted as devices by Amazon EC2 instances. Multiple volumes can be mounted to the same instance. Using an EBS as instance storage, users can temporally stop their instances, without data loss.

The EC2 Instances are created by launching machine images known as Amazon Machine Images (AMI) [47], which contain the operating system that will be launched on the instance, along with software applications and their configuration. AMIs are stored on Amazon S3 and Amazon provides a number of pre-bundled public AMIs (with Linux, UNIX or Windows as the OS), that can be immediately launched by users, and do not require specific configuration.

Users can also create their own custom AMIs (private AMIs), either from scratch or using a public AMI as base. Private AMIs are created by a process called bundling, in which a machine image is compressed, encrypted and split, the parts of which are then uploaded to Amazon S3.

EC2 provides the ability to place instances in multiple locations. EC2 locations are composed of Regions and Availability Zones. Regions consist of one or more Availability Zones, are geographically dispersed. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region [61].

The Amazon S3 service provides a simple web service interface that can be used to store and retrieve data on the web, and provides a scalable data storage infrastructure [70]. It is designed to make storing and retrieving data on AWS as simple as possible. Data is stored using a straightforward flat model, on top of which users can build their own storage structures using hierarchies. S3 also features a simple, yet versatile, access control system, where objects can be made private, public or be made accessible by certain groups of users.

The two main concepts of S3 are buckets and objects [61]. Buckets are containers for data objects. All objects stored on S3 are stored in buckets. An object consists of four components: a value (the data being stored in that object), a key (the unique identifier for that object), metadata (additional data associated with the object) and an access control policy.

Each bucket has a name that is completely unique within S3. Bucket names are directly mapped to URLs for addressing data stored on S3. If a bucket is named *cloudencoding* then it can be addressed with the URL *http://cloudencoding.s3.amazonaws.com*. This URL can be appended with the name of an object to create an address for any object stored on S3.

The other main concept of S3 is an object. Object sizes vary from one byte to five gigabytes. There is no limit to the number of objects that a user can store on S3 and no limit to the number of objects that can be stored in a bucket. A bucket can be stored in one of several Regions. Users can choose a Region to optimize latency, minimize costs, or address regulatory requirements [61, 70].

S3 objects are redundantly stored on multiple devices across multiple facilities in an Amazon S3 Region. To help ensure durability, Amazon S3 PUT and COPY operations synchronously store your data across multiple facilities before returning. Once stored, Amazon S3 maintains the durability of your objects by quickly detecting and repairing any lost redundancy. Amazon S3 also regularly verifies the integrity of data stored using checksums. If corruption is detected, it is repaired using redundant data. In addition, Amazon S3 calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data [70].

The key is the name of the object and must be absolutely unique within the bucket that contains the object. Keys can be any size from one byte to 1,024 bytes. Keys can be listed by their bucket and a prefix. This allows users to use common prefixes to group together their objects into a hierarchy, meaning that the flat storage model of S3 buckets can then be turned into a directory-like model for storing data. Object keys can also be given suffixes, like *.jpeg* or *.mpeg*, to help make the key more.

The metadata of an object is a set of key/value pairs and is divided into two types: system metadata and user metadata. System metadata is used by S3 while user metadata can be any key/value pair defined by the user. User metadata keys and values can be any length, as long as the total size of all metadata (system and user) for an object is less than two kilobytes.

Access control on objects is managed by access control lists (ACL). Every object, as well as every bucket, has an ACL. When a request is made to S3, it checks the ACL of the object or bucket to check if the requester has been granted permission. If the requester is not authorized to access the object then an error is returned by S3. There are a number of different types of groups that can be granted permissions and a number of different permissions, such as READ, WRITE and FULL CONTROL.

S3 provides two APIs for making requests, the first uses a REST protocol and the second uses SOAP. The REST API uses standard HTTP headers and status codes, with some additional headers added in by S3 to increase functionality.

Amazon S3 also has the option of the Reduced Redundancy Storage (RRS) that enables customers to reduce their costs by storing non-critical, reproducible data at lower levels of redundancy than Amazon S3's standard storage. It provides a cost-effective solution for distributing or sharing content that is durably stored elsewhere, or for storing thumbnails, transcoded media, or other processed data that can be easily reproduced. The RRS option stores objects on multiple devices across multiple facilities, providing 400 times the durability of a typical disk drive, but does not replicate objects as many times as standard Amazon S3 storage, and thus is even more cost effective [70].

The Amazon AWS Platform also provides several additional services, as previously mentioned, which are not the focus of this research. One of them is the Elastic Map-Reduce, an implementation of Map-Reduce [10] algorithm built on top of the basic AWS infrastructure blocks (EC2 and S3). This feature is particularly interesting because EC2 and S3 alone, are not sufficient to provide efficient and scalable high performance processing architecture. To achieve these goals, one has to build applications that are able to take advantage of the characteristics of IaaS infrastructures, for example, the ability to automatically start and stop machines according to processing demands, or the ability to use several machines to simultaneously process parts of a content. One paradigm that deals with these issues is the Map-Reduce, detailed in the following section.

2.3 The Map-Reduce paradigm and Distributed Data Processing

The distribution of tasks in a cluster for parallel processing is not a new concept, and there are several techniques that use this idea to optimize the processing of information [49, 50, 51]. The Map-Reduce paradigm [10], for example, is a framework for processing large datasets of certain kinds of distributable problems, that makes use of a large number of computers (nodes), collectively referred to as a cluster. It consists of an initial Map stage, where a master node takes the input, chops it into smaller sub-problems, and distributes the parts to worker nodes, which process the information independently; following there is the Reduce stage, where the master node collects the solutions to all the sub-problems and combines them in order to produce the job output. The process is illustrated in Figure 1.

A popular Map-Reduce implementation is Apache's Hadoop [11], which consists of one Job Tracker, to which client applications submit Map-Reduce jobs. The Job Tracker pushes work out to available Task Tracker nodes in the cluster, which execute the map and reduce tasks.

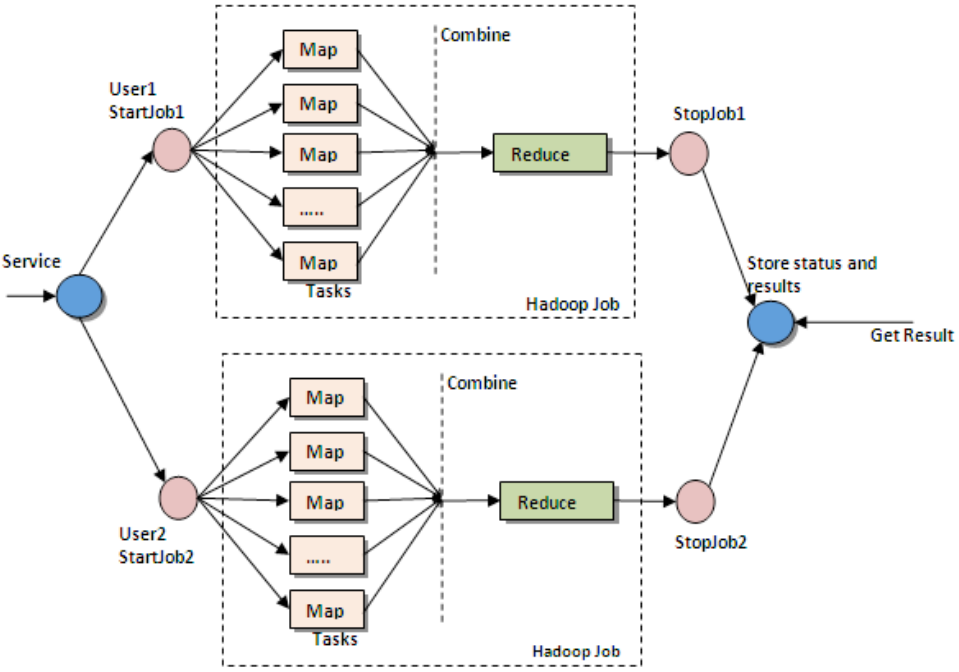

Figure 1. The Map Reduce Architecture [44]

Despite being a very appealing and efficient technique for processing large volumes of data, there are a number of challenges and shortcomings associated with the deployment of Map-Reduce architectures [46]. The first of them is the required infrastructure. To make the process truly effective, one needs several machines acting as nodes, which often requires a large upfront investment in infrastructure. This point is extremely critical in situations where the processing demand is seasonal. In addition, fault tolerance issues and the need of a shared file system to support mappers and reducers make the deployment of a Map-Reduce architecture complex and costly.

To understand how Map-Reduce could be useful to improve the efficiency of a video processing task, and, which of its characteristics we must preserve while building a high performance video processing architecture based on a Cloud platform, we first need to understand how video compression works, and which are the main steps of this process. Only then, can we discuss the required changes and adaptations to the Map-Reduce paradigm.

In the next chapter, we present a brief introduction to sequential video processing, followed by a discussion on the requirements for parallel video processing, and the required adaptations to the Map-Reduce paradigm so that it serves this specific purpose.