Cost aware real time big data processing in Cloud Environments

By

Cristian Montero

Under the supervision of

Professor Rajkumar Buyya
and
Dr. Amir Vahid

A minor project thesis submitted in partial fulfilment of the requirements of the degree of

Master of Information Technology

June 2014

Department of Computing and Information Systems
The University of Melbourne

Abstract

BIG data processing environments demand a considerable amounts of cloud resources. With the variety of cloud resources and volatility of Big-data application workloads, it is difficult to decide when and how to scale up and down to satisfy user QoS requirements of budget and deadline. Therefore, in this project we propose a resource provisioning and scheduling approaches (which utilizes optimization algorithms) that can supply sufficient resources to meet the users QoS requirements.

Declaration

I certify that

1. This thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

2. The thesis is 8000 words in length (excluding text in images, table, bibliographies and appendices)

Cristian Montero, June 2014

Acknowledgements

I would like to express my special appreciation and thanks to my supervisors Professor Rajkumar Buyya and Dr. Amir Vahid. To Professor Rajkumar Buyya for the opportunity to work and learn what I needed to start this project. Furthermore I would like to thank Dr. Amir Vahid for introducing me to the topic as well for the support on the way. His ideas, feedback and guidance were important to complete this project.

Also, I would like to thank my loved ones, who have supported me throughout entire process regardless the distance, their calls and words were helpful keeping me motivated. I will be grateful forever for your love.

Cristian Montero
June 2014

Contents

List of Tables

List of Figures

Chapter 1
Introduction

Nowadays, every single minute a huge amount of data is created from different sources, such as: scientific applications and sensors, GPS, social networks. For example, approximately 600 million times per second, particles collide within the Large Hadron Collider (LHC). Each collision generates particles that often decay in complex ways into even more particles. Electronic circuits record the passage of each particle through a detector as a series of electronic signals, and send the data to the CERN Data Centre (DC) for digital reconstruction. The digitized summary is recorded as a "collision event". Physicists must sift through the 15 petabytes or so of data produced annually to determine if the collisions have thrown up any interesting physics. The Data Centre processes about one petabyte of data every day - the equivalent of around 210,000 DVDs. The centre hosts 10,000 servers with 90,000 processor cores. Some 6000 changes in the database are performed every second. The Grid runs more than one million jobs per day. At peak rates, 10 gigabytes of data may be transferred from its servers every second[5].

Another example of this impressive data growth rate are social networks. where every minute Twitter users send over 100.000 tweets, Facebook users share 684,478 pieces of content, YouTube users upload 48 hours of new videos, and so on.

Big data not only focuses on large data (*volume*), but also on properties like *variety* and *velocity*. Big data is coming from a greater variety of sources in structured, semi structured or unstructured formats. In addition, big data can be defined in terms the frequency of data generation. A good example is the stream of data coming off sensors or social networks. [9][20].

Stream Processing System emerge as a solution[13][10] to enable real-time processing of continuous data streams. The idea is to process all incoming data and transfer it to the main memory. This is impossible with traditional batch processing because it requires the data to be stored first. It starts capturing the data, storing it and finally processing it to retrieve the results[25].

When we are dealing with the Big data processing applications, users (specifically for time-critical applications) require to have the processing results by a deadline. However, the velocity of data varies and that can significantly affects the processing time. One possible solution is procure resource for data bursts and maximum demand which require considerable capital investment on infrastructure. This solution is not feasible for SMEs and even for larger organizations comes with high cost of electricity and resource wastage. Alternatively it is possible to utilize Cloud services and add more resources on-demand via internet to satisfy deadline requirements. Cloud services include but limited to computing, storage, network, and software services[21][16]. The cloud adoption is not enough to solve the problem thoroughly. The reason is that, in addition to the deadline requirement, users also have budget constraint and their objective is to minimize the cost. Therefore, it is difficult to decide when and how to scale up and down to satisfy user QoS requirements of budget and deadline together. We Investigate the aforementioned issue in the thesis.

1.1 Contributions

As we mentioned earlier process all information available in real time comes at a cost incurred by cloud providers. In majority of SMEs a common aim is to adopt the cutting edge technology to accomplish objectives of the company more efficiently and at the minimum cost and big data processing with Cloud is not an exception. Therefore the main contribution of this thesis is proposing a resource provisioning and scheduling approaches (which utilizes optimization algorithms) that can supply just enough resources for real-time processing of big data and also meet the users QoS requirements.

1.2 Methodology

To demonstrate the effective of our optimization algorithm a real testbed has been developed with the following component.

- **Twitter data collector:** We are using Twitter data as stream data which is accessible through APIs provided by Twitter via Twitter4j library. The library used to capture data in JSON format for a period of 3 weeks and stored in a NO-SQL database for experiment.

- **Real-Time Twitter data analyzer:** Although there are few tools available for stream processing such as Apache S4 and Splunlk we decided to use Twitter Storm. The reason is that it is open source, fast, and fault-tolerant. In addition it has been benefited from a comprehensive documentation and an extensive community support.

- **Auto-scaling Optimizer:** This component first predict data growth rate first. It then based on the trade-off between cost and performance of cloud resources provides just enough resources which satisfies users deadline and minimizes the cost of processing.

The plan is to compare our auto-scaling policy with the one offered by Amazon EC2 which only uses virtual machine utilization for making scaling up and down decision. The objective is to test whether we can decrease the cost while we are adhering to the deadline constraint.

In this work we focus on static storage support for data streaming. The paper starts with an introduction to the terms we will use during this whole document. Next, we present the design of the scenario for the experiment and its architecture, tools and software. Then the results obtained during the experiment are presented. Finally, the conclusions and future work related to this research are discussed.

Chapter 2

Literature Review

Evolution of computing started in the early 1950's when mainframes appeared in universities and corporations to supply the processing capabilities using terminals, which did not have internal processing capabilities to perform tasks. Later, in the 1990's, computing became popular with the evolution of point-to-point data circuits called Virtual Private Networks (VPN). At this point it was defined exactly what provider and users are responsible for and that was the beginning of cloud computing.

Cloud computing is a new computing model, where an application runs , at the same time, on interconnected computers. Those computers would be Physical or Virtual Machines (VM). Both applications and computers would be shared by many applications of different organizations as a service in either private or public mode. Their consumption defines one of the main characteristics of cloud computing, the on-demand model. For these reasons, it is possible to classify different kinds of services in cloud computing: Infrastructure as a service, Platform as a service and Software as a service.

In the age of Big Data it is important to understand how invaluable it is to manipulate this data in order to use that information to for example, predicting building fires, anticipating the best moment to buy a plane ticket, seeing inflation in real time and monitoring social media in order to identify tendencies or needs in order to provide accurately either products or services to customers.

Every second it is generated a high volume of data that has to be analyzed. Processing that amount of data involves a high consumption of computing resources to get results as faster as possible without exceeding budget. This is where cloud computing can help to provide powerful computing services on-demand.

This chapter provides an overview of all the terms this work involve for a better understanding of their definition, characteristics, taxonomy and their practical application .

2.1 Cloud Computing

Cloud Computing is an on-demand model where all computer resources are configured into a shared pool and available when it is necessary with minimal or no effort at all. Those resources include servers, storage space, network, applications and services[21][16].

Also, it is possible to define cloud computing as a type of parallel and distributed system where a set of interconnected computers, either virtualized or not, are presented as a unique resource, based on SLA, over the network [4].

Resources available can serve multiple customers using a multi-tenant model. In other words, based on the usage rate that each customer has it is possible to provide more or less resources, such as, storage, processing, bandwidth, in order to, perform tasks on time.

Cloud Computing delivers services over the internet starting from infrastructure passing by platforms until applications or services solve tasks. These three kinds of services are called: Infrastructure as a Service (*IaaS*), Platform as a Service (*PaaS*) and Software as a Service (*SaaS*) [21][15].

All those services would be deployed in several environments: a *private cloud*. It means infrastructure is managed by a single company. It could be the company itself or a third party organization. A *Community cloud* is shared by some companies with security policies to guarantee the safety of the information. A *Public cloud* is provided and managed by a specialized company that provides these services not just to a single organization. Finally, with a *hybrid cloud*, the cloud components consists of two or more different environments each contributed by different organization. [14]

In the following sections are described in detail all those characteristics and services.

2.1.1 Services

In Cloud Computing it is possible to identify three different kinds of services based on the computing resources they provide. They would be network infrastructure, servers and clustering platforms or applications.

Those services use *pay-as-you-go* (PAYG) as a business system. It means they can be rented from the cloud provider, and be accessed over the Internet, in order to create or use applications. It is similar to paying for the consumption of services such as electricity or gas.

For cloud service management, providers offer a friendly graphical interface (Using Web 2.0) to have complete control of computing resources. However, they also provide a customer service which would be by phone, email or chat. The three services are shown in Figure 2.1 and described in the next subsections.

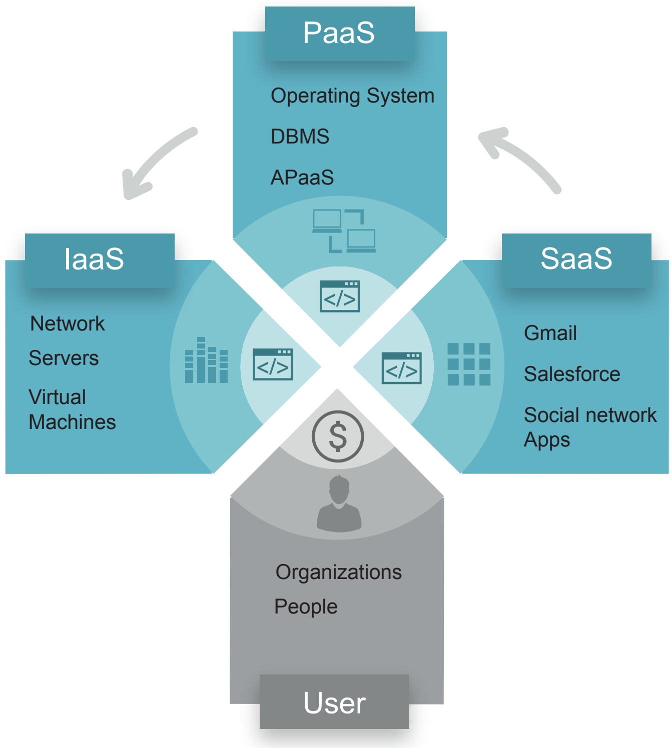

Figure 2.1: Services provided in Cloud Computing. Customers rent those computing resources and manage those resources by themselves using applications provided

Infrastructure as a service (IaaS)

Infrastructure as a Service refers to the computer resources such as storage, servers, virtual server space, network connections, bandwidth, IP addresses and load balancers. Physically, it is the pool of hardware resource of servers and networks usually distributed across numerous data centers in different locations. This infrastructure scales on-demand; it is an advantage for large organizations with workloads that fluctuate rapidly. Even though all those sources are available to be rented, it is possible to use current companies' infrastructure (private clouds) together with the purpose of shared responsibility in cloud management and also to save money.

There are many companies providing these services, such as: Amazon Elastic Compute Cloud (EC2) [1] [1], Google Cloud Storage [2], Microsoft Azure [3]. Although it is possible for any organization to contract these services, SME (Small and Medium Business) and start-ups highly take advantage of the benefits because they do not need to invest in their own infrastructure; saving money, space and people working to keep that infrastructure operational.

Platform as a service (PaaS)

Platform as a Service represents a middleware, and is called also Application Run-time Environment (ARE), that allows user to build, deploy, test and run applications without the challenge of maintaining the hardware and software infrastructures. In other words, it is a faster and more effective model for application development and delivery. It includes operating systems, DBMS, Server software, application platforms, integration and business process management which have to fulfil the typical requirements such as scalability, reliability and security [2].

As a platform it is available online from everywhere as well as all data being safe due to security policies, backup and recovery services. Moreover, it is possible to have developers working together spread across various locations.

[1] https://aws.amazon.com/ec2/
[2] https://cloud.google.com/products/cloud-storage/
[3] http://azure.microsoft.com

Although all those services are provided in PaaS, Application platform as a service (aPaaS) stands out because it offers development and deployment environments for application services together. This helps supplying frameworks to implement scalable applications. Some examples are: Aneka [11] [4], Hadoop [5], Salesforce [6], Google App Engine [7]. Another aPaaS feature is a rich set of API's to develop extensions also called add-ons for the software company's core application in order to include more functionality.

Software as a service (SaaS)

Customers make use of any software hosted by a service provider and at the end customers only pay for the time using that particular software. There are plenty of examples of this kind, namely: Google, Facebook, Twitter, Dropbox, among others. SaaS has many characteristics and benefits that are described as follows.

- Those applications are accessed over the Internet from any device such as PCs, mobile phones, tablets. For these reasons no additional hardware is required.
- It does not need an initial setup payment because applications are ready to download / access to use.
- Every application is rented instead of purchased. It means that you only pay for the time you use it. This is usually on monthly payment basis and also depends on the number of users.
- Applications are updated automatically. Users do not have to do anything if their devices to access latest versions of any application.

2.1.2 Cloud Computing Environments

As mentioned in Section 2.1.1 there are many companies that provides IaaS and PaaS services. For this project, following cloud environment is used:

[4] http://www.manjrasoft.com/products.html
[5] http://hadoop.apache.org/
[6] http://www.salesforce.com/au/platform/overview/
[7] https://developers.google.com/appengine/

Amazon EC2

Amazon provides IaaS and PaaS services. The name of the product is Amazon Elastic Compute Cloud (Amazon EC2), which is a web service that provides resizeable compute capacity in the cloud. It is designed to make web-scale computing easier for developers [1].

Using web service interfaces, it is possible to launch instances (Virtual Machines) with a variety of operating systems, load them with custom application environment, manage network access permissions, and run images using as many or few systems as desire. Those VM Templates, are called also Amazon Machine Image (AMI). Security settings are available to control access to any instance as well.

Instances would run in multiple locations. Those locations are organized in regions and availability zones. You can deploy your applications in any location you want in order to avoid failures of single location.

Cloud watch is the service that provides monitoring for Amazon resources. It shows usage, operational performance and overall demand patterns of AWS resources and applications.

Distribute incoming request among the cloud (EC2 load balancing). It decrease significantly the chance of fault because it detects if an instance is either busy or not healthy and redirect all the traffic to another instance.

Huge workloads require many cloud resources to accomplish objectives. High performance computing (HPS) clusters present in Amazon EC2 allows, benefiting of elasticity, flexibility and cost characteristics, to handle big amount of transactions at the same time taking advantage of Cluster Compute, Cluster GPU, and High Memory Cluster instances which are designed to increase throughput.

2.2 Big Data

These days, growth of data and data variety produce an effect that makes impossible to process and analyze information with traditional tools and procedures. It represents a real challenge to be addressed.This phenomenon is known as ”Big Data”.[9]

Integrated circuits are becoming cheaper. It helps to add intelligence to almost everything. Every day any kinds of devices are more and more interconnected. For example: cars, rails, health sensors, mobile phones. All those devices store information related with their functions. This is the main reason of increase in data *volume*. Even though it is possible to store anything companies want, For example, GPS, weather sensor, even social network and so on. It is challenging to access all information in real-time, due to *variety* of semi structured and unstructured format. In addition, from business perspective, time it takes to analyse the data streams with different velocity is also important [20]. It means that no matter what process the information has to go through, results should be immediately available for people.[25]

2.2.1 Characteristics

Big Data consists in three characteristics (Figure 2.2): *Volume* which refers to the amount of data. *Variety* of data formats such as structured and unstructured. Finally, *Velocity* at which data is processed . More details are shown in following sections.

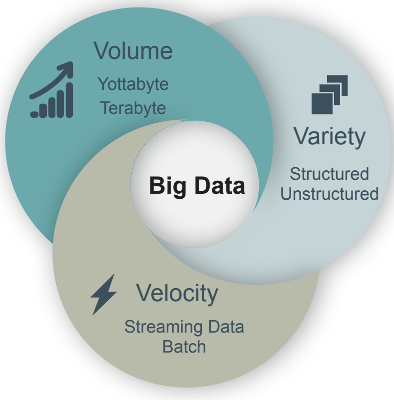

Figure 2.2: Big Data. The Three Vs

Volume of Data

Today we store all kind of data: environmental, financial data, medical, surveillance, and the list goes on and on. This data comes from any kind of devices: Computers, sensors, mobile phones, tablets, etc. It generates a big amount of data every minute. Such increment make us that we passed from terabyte to petabytes. However, soon we will need more prefixes to represent the amount of data we will use.

For this reason, International System of Units (SI), organization that defines prefixes names and prefix symbols of the decimal multiples and submultiples of SI units [18], until 1990 the larger number represented was in the order of Exabytes. However, with resolution 4 of the CGPM (1991) [17] were introduced prefixes for values up to 10^{24} as it is shown in Table 2.1.

Factor	**Name**	**Symbol**
10^3	Kilobyte	KB
10^6	Megabyte	MB
10^9	Gigabyte	GB
10^{12}	Terabyte	TB
10^{15}	Petabyte	PB
10^{18}	Exabyte	EB
10^{21}	Zettabyte	ZB
10^{24}	Yottabyte	YB

Table 2.1: Orders of magnitude of data

Variety

As information comes from different sources. Each one saved it in different formats. It includes not only traditional relational data, but also raw, semi structured, and unstructured data from web pages, web log files (including click-stream data), search indexes, social media forums, e-mail, documents, sensor data from active and passive systems.

In fact, majority of data is unstructured or semi structured. That is why is really important for organizations to take advantage of the information being able to analyze both structured and unstructured data.

Velocity

It refers to the speed rate of data arriving, flowing, being saved and its associated rates of retrieval. Organizations must be able to handle variety (Section 2.2.1) and volume

(Section 2.2.1) of data as closer to real time as possible, in order to find insights in this data while it is still in motion, not just when it is stored.

There are two approaches that should have to be consider in this point: Batch Processing, which consists in take incoming data and save it before start processing it. Stream Data Processing, where incoming data is processed at the time it arrives, before it will be save. More details of this approaches are shown in Section 2.3.

2.2.2 Importance of Big Data

Big Data solutions importance lies on the following aspects: To analyze different type of Data from a wide variety of devices (Section 2.2.1). Also, this analysis is perfect for iterative and exploratory when business measures on data are not predetermined.

Another scenario for applying Big Data solutions is when it is required to analyze all, or most, the data available instead of sampling the data. In the same way, there are scenarios where taking a sample of the data is possible to get valuable results.

To conclude it is possible to say that any analysis, investigation, study or calculation that cannot be fitted in a relational database approach, it is possible to be handle with Big Data approach.

2.3 Big Data Processing

In the age of Big Data it is important to understand how invaluable it is to manipulate this data in order to use that information to, for example, predict building fires, anticipate the best moment to buy a plane ticket, see inflation in real time and monitor social media in order to identify tendencies or needs in order to accurately provide either products or services to customers.

Every second high volume of data are generated that has to be analyzed. Processing that amount of data involves a high consumption of computing resources to get results in real-time without exceeding budgets.

The structure of a system that performs big data processing on the cloud should have subsystems for decision making, analysis and data aggregation and storage with the capacity of managing, provisioning and scheduling many servers to control massive event streams.

decision making means that it should be able to handle a bunch of events per second and make an immediate judgment without crashing when an unexpected variation of requests appear. For such analysis, it is required to have as many servers as necessary to process all data to do all statistical calculations in real time. Finally, all generated data has to be saved in a flexible infrastructure which allows for expanding of its capacity according to the amount of data that has to be saved. Distributed Data Store is a technique that could be used to handle a big amount of write and read requests. An example of this technique is distributed Key Value Store (KVS). It is a data structure which contains keys and its respective values; both stored in some servers in a distributed way. As all data are not stored in a unique server it helps to ensure a high performance even if there are many requests at the same time [9].

2.3.1 Batch Processing

Batch processing is a *store-and-process* model, where data are collected, entered, processed and then the batch results are produced. There are two kinds of this model. First, when results are produced by a scheduled task it is called *traditional batch processing*. On the other hand, there is the *Service-Oriented Architecture* or simply SOA Batch processing. In this model tasks are performed when a user requires the results of an specific batch. It is widely used in relational data. Both models are shown in Figure 2.3 over.

Batching processing is used in environments where it is required to process all the data at a time [13]. However, processing all information at a time presents disadvantages. First, it has turnaround time. This means the time taken between submitting the job and getting the output, which includes the information, related to the jobs are included in the job. For this reason it is a highly time consuming model when there exists huge amount of data.

Also, it is a time-varying process, as was described before. All processes would produce a nonlinear behavior and that is what produces uncertainty about the execution time.

For these reasons batch processing is commonly used in scenarios where there is a large amount of data processed and saved. Also, when a real time response is not required.

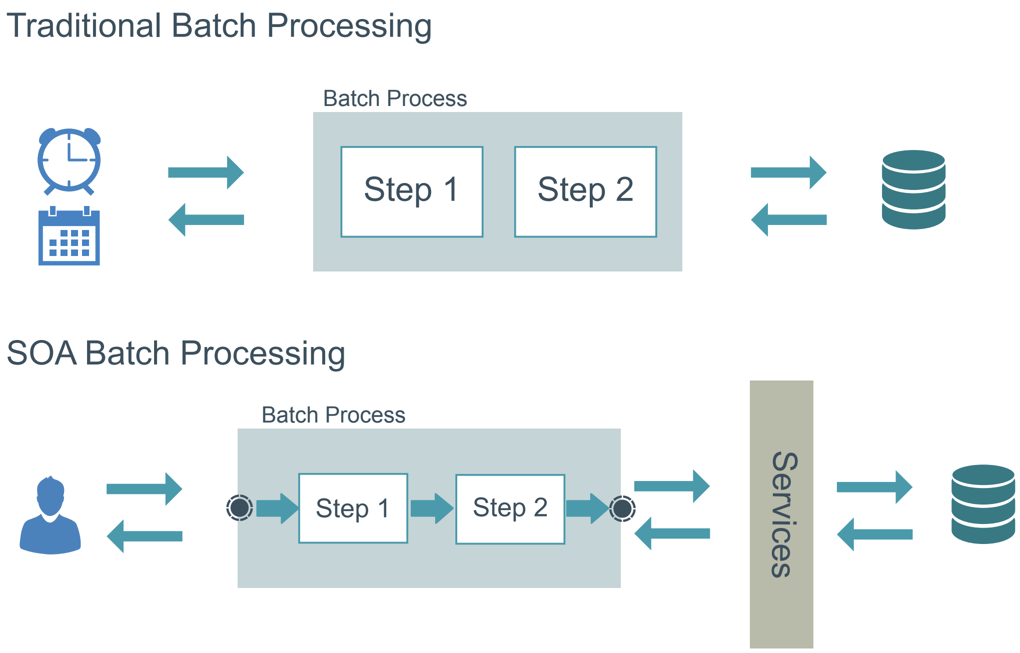

Figure 2.3: Batch Processing. Two kinds: Traditional (scheduled task) and SOA (Perform task under request)

Hadoop

The Apache Hadoop [8] software library is a framework that allows distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering computation and storage. Also, it includes error detection and handling. It consists of two main components described as follows.

First, a *Hadoop Distributed File System (HDFS)*. It is the storage component. It provides high throughput access to application data and it works better for large data sets. To do so, it handles large block sizes and data location reducing transfer speeds on networks.

[8] http://hadoop.apache.org/

Also, it is fault tolerant and applies data replication in order to have scalability and availability.

The other component refers to a framework called *MapReduce* [6] which allows parallel execution of data processing tasks. It takes a dataset and divides it in some smaller pieces to be processed over multiple nodes. Some characteristics are scalability in which it is possible to add worker nodes across the cluster, high throughput to run on many nodes at low cost, fault-tolerant to handle situations where any nodes crash it has procedures for recovery of information that is why is is used in many DBMS.

Finally, there are two kind of nodes present in Hadoop. *Master* node is the one in charge of data division, work distribution and results collection. And, *Worker* nodes are the ones executing process and sending results to the master node. Both working passing messages to know the status of each node as well as process completion to guarantee cluster integrity.

2.3.2 Stream Processing

Data stream processing is one of the ways to support real-time response processing of big data, present in systems such as financial data analysis and sensor networks. Even though it helps managing big amounts of information, it requires an optimization at the moment of using cloud resources. Dynamic management of cloud resources helps us to reach objectives in terms of time and budget, using required resources to insure low latency and using them just in the time needed. This helps to reduce the cost of operation of the task being executed [8].

It consists of a continual input, process and output of data. This is useful when data has to be processed in a small stipulated time. For example, to analyze incoming information to detect attacks such as DoS(Denial-of-Service), synchronous concurrent algorithms also data flow, reactive, signal processing [23]. Figure 2.4 shows how real time processing works.

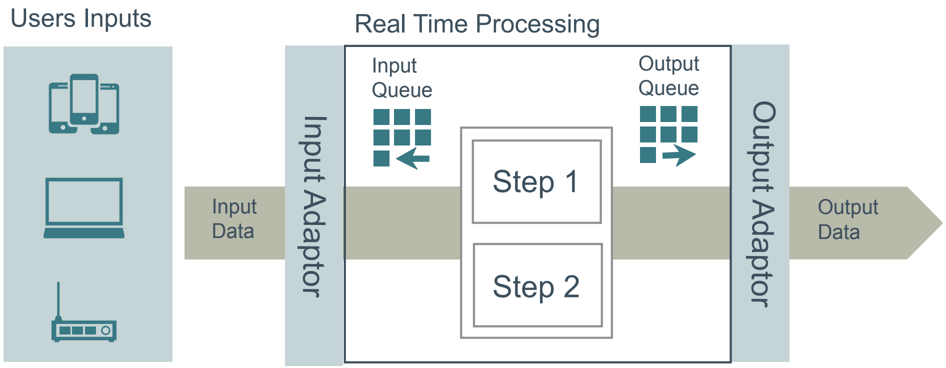

Figure 2.4: Stream Processing. Information is processed before it is saved on database

Storm

Storm is a distributed real-time computation system. It provides a set of general primitives for doing realtime computation. Storm is simple and can be used with any programming language [12].

It is very similar to Hadoop and has the same functionality but for batch processing (Previous section).

Components

The main component is the *Topology*; it is a process that runs forever or at least until the process is killed. This main process consists of many worker processes spread across the cluster [12].

Storm uses two kinds of nodes: **Master** nodes which execute a daemon called *Nimbus*. A Master node is in charge of code distribution around the cluster; also, monitoring any problem during execution and, finally it assigns tasks to **Worker** nodes. Those worker nodes listen for work assigned from the Nimbus node. Also, the Worker node is responsible for starting and stopping worker processes to execute a part of a topology depending on a Nimbus request. In worker nodes runs a daemon called *Supervisor*.

To work together Master and Workers nodes run on Zookepper [9] [7], which makes

[9]http://zookeeper.apache.org/

the coordination of all the work across the cluster. Zookepper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications.

Figure 2.5 shows all components described before.

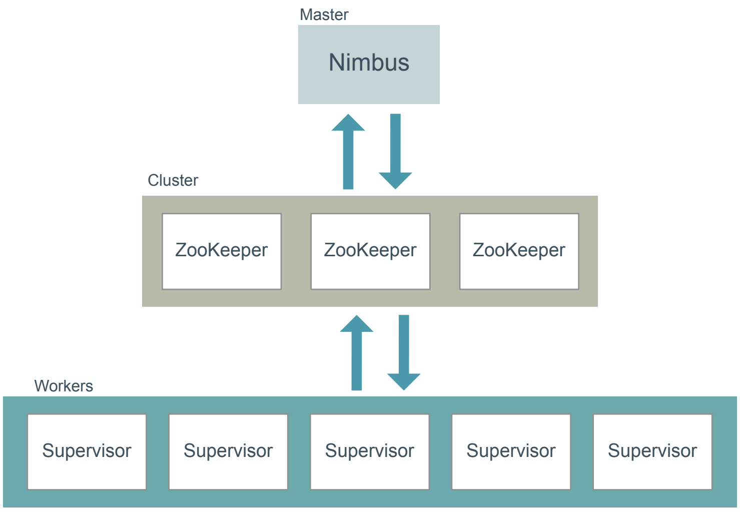

Figure 2.5: Storm Components. Master Node (Nimbus) send request of execution of task to Worker nodes (Supervisor) through Zookeeper cluster

Streams

Stream is the main abstraction in Storm. It is a set of unbounded tuples and allows to transform a set of those tuples or some of them into a new stream. To do so, it implements two basic primitives: *spouts*, which are a source of streams; for example, a queue of tweets to be sent as a new stream. Also, *bolts* are implemented that consume any amount of streams and do some processing and can emit other streams or not.

A topology can group spouts and bolts. It is a high-level abstraction to be sent to the cluster. In a topology every node is a spout or bolt. Also, all nodes run in parallel

2.4 Big Data processing Systems

There are many systems for big data processing. Among them we can mention:

ElasticStream is a system that uses cloud environment and dynamically adjust the number of nodes in the cloud, either private or public. For this purpose, it uses a prediction algorithm called Sequentially Discounting AutoRegression (SDAR) and it is used to predict load for every period of time called *TimeSlot*. Which can be from few minutes to few hours. ElasticStream receives incoming data stream, then splitting the data up for multiple computational nodes, and finally processing them in parallel. Although ElasticStream optimizes latency, it [8] does not consider a deadline constraint to complete task execution.

Fujitsu Laboratories has developed a distributed data store and parallelization of complex event processing (CEP). CEP refers to technology that processes and analyzes real time complicated and large amount of data. They have developed a fast allocation resource system, that add or remove resources without interruption in service. Also, it benefits from a language and programming model that process big data. It is an efficient system that focused in geological and seismology system [24].

ESC (pronounced Escape) is a distributed stream processing platform for data mining in real-time. It defines a simple programming model in which programs are specified by directed acyclic graphs (DAGs). In addition, it is distributed and fault tolerant which is hidden to users. The engine dynamically adapts to different workloads. Furthermore, in the cloud, ESC is able to add and remove machines to adjust according to the current requirements. Finally it is focus on latency and it does not take consideration of deadline constraint [22].

Although all those systems support big data processing in cloud; they do not consider customer constraints of deadline or budget. The aforementioned systems focused on latency. It means that they aim at improving throughput by adding instances with the purpose of finishing the task as faster as possible. In our architecture along with latency, we try to minimize the cost of execution by choosing the best combinations of resources in the cloud at the right moment.

Chapter 3

Big Data Processing Architecture

The main objective of the architecture presented in this chapter is using least number of cloud resources as possible to process real-time data before deadline approaches. Figure 3.1 shows a representation of this architecture and next sections will explain in detail every component of it.

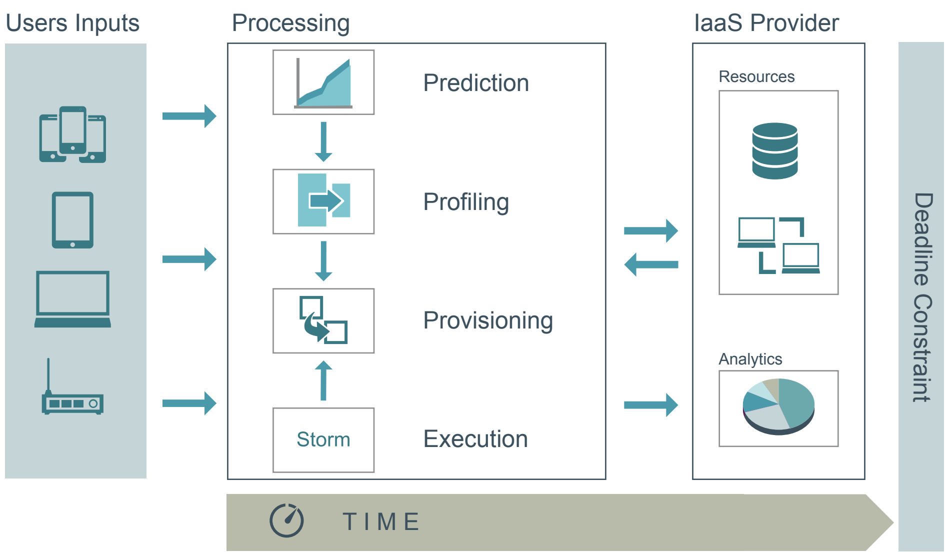

Figure 3.1: Big Data Processing Architecture. Data streams are required to be processed in cloud environment. Before that an estimation and assignment of resources are done in order to reach uses satisfaction

3.1 Big Data Processing System

For this project a data stream processing toolkit is required in order to accomplish expected goals of real-time processing. As was mentioned in Section 2.3.2, **Storm** is a scalable, fault-tolerant distributed computing system. It provides a distributed stream processing system, based on a architecture composed of bolts and spouts, that together form a topology. What this system does for real-time processing, Hadoop does for batch processing (Section 2.3.1).

3.2 Prediction Engine

As shown in the Architecture, our cloud resources provisioning consists in a time-series forecasting engine which is predicting the rate of generated data (Bid-data velocity). The prediction helps us to estimate how many cloud resources are required for the next time-slot to process the data before the dead-line .

prediction is accomplished through Weka[1] version 3.7.3, which has a dedicated time series analysis environment that allows forecasting models to be developed, evaluated and visualized.

Weka uses a machine learning / data mining approach to model time series by converting the data into a form that standard propositional learning algorithms can process. Weka accomplishes via replacing time-dependent feature with other fields that are automatically calculated and added. Once the data has been transformed, any of Weka's regression algorithms can be applied to learn a model. It is possible to apply multiple linear regressions, powerful non-linear methods such as support vector machines for regression and model trees or any method capable of predicting a continuous target.

Figure 3.2 shows the workload fluctuation in collected days. Also, it is possible to identify the highest peak of 1000 tweets; that information is useful at the moment that we will be looking for the most suitable resource.

In Figure 3.3, forecasting number of tweets for the last thirteen hours represented with blue line. On the other hand, the red line represents the real workload for the same period

[1] http://www.cs.waikato.ac.nz/ml/weka/downloading.html

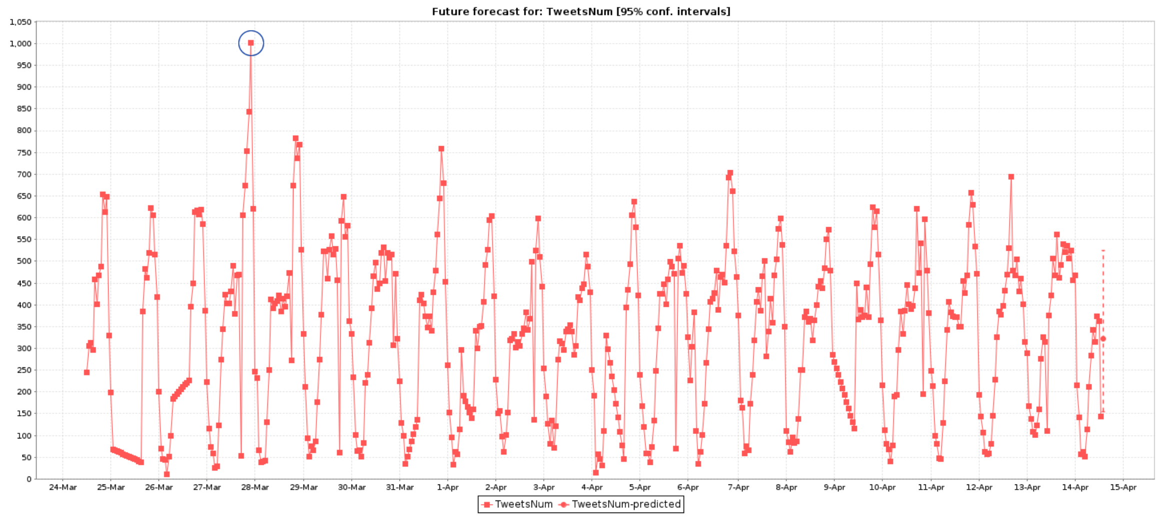

Figure 3.2: Data collection flow behavior

of time. According to that information it is possible to say that our prediction component works accurately; obtaining closer results in order to accomplish our objectives.

3.3 Profiling

Profiling is the process which helps us to identify the most suitable resource according our requirements. Key points are the performance and time execution of tasks in different instances. In this step, it is important to collect sample information regarding processing time of different number of tweets on different number of instances. This information is later used to train our machine learning engine which is used for resource selection.

Once, we have accurate predictions about workload; it is time to determine the resources we will need to process those predicted workloads. First, it is important to know the features of existing instances in Amazon EC2. Table 3.1 contains some information about each instance.

Sets of different data streams were generated each one with different amounts of data with the purpose of obtaining processing times in Amazon EC2 instances. Those data streams were defined in sets of 200 to 1000 tweets. For Small instances it takes about 60 seconds to process a peak of tweets. For Medium instances it takes about 40 seconds. Finally, for Large instances it would be 14 seconds. As we can see, it exists a big difference

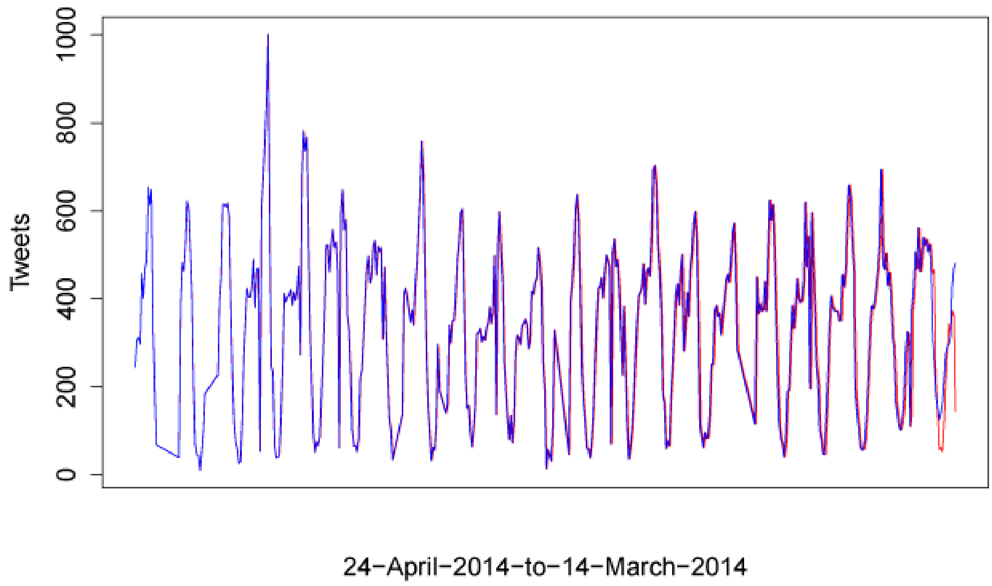

Figure 3.3: Workload prediction. Blue line represents forecasted workload and red line represents the real workload.

Instance	**vCPU**	**Memory (GiB)**	**Storage (GB)**	**Networking Performance**	**Processor**	**Clock Speed**
Small	1	1.7	1 x 160	Low	Intel Xeon Family	-
Medium	1	3.75	1x4 SSD	Moderate	Intel Xeon E5-2670	2.6
Large	2	7.5	1x32 SSD	Moderate	Intel Xeon E5-2670	2.6

Table 3.1: Amazon EC2 Instances characteristics [2]

in processing times. Those execution times of those different sizes of data streams are shown in Figures 3.4, 3.5 and 3.6.

3.4 Resource Selection

3.4.1 Training Sets and Query

As mentioned earlier the information collected in the profiling step is used to build necessary training set. In this section we describe this process in details. A training set consists of an input vector and an answer vector, and is used together with a supervised learning

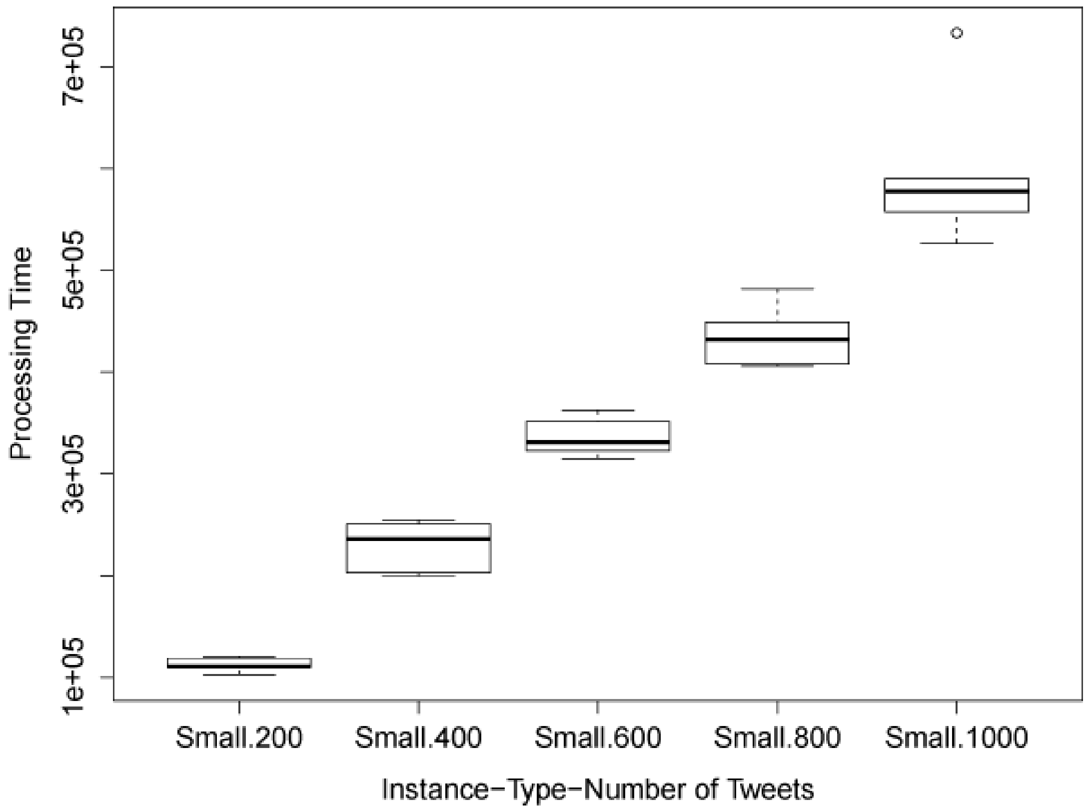

Figure 3.4: Execution time in small instances

method to train a knowledge database.

Our training set consists in three main components:

1. **Number of tweets** which come from our prediction engine described before.
2. **Execution time** that represents the time required to complete data analysis.
3. **Instance type** which is a Cloud computing resource used to process the tweets.

3.4.2 Prediction Engine

If we collect enough data to build a proper training set, then we can use our prediction engine to predict which instance type is required to finish the task for a give deadline and give number of tweets. For this purpose we utilize a decision tree classifier called J48 in WEKA.

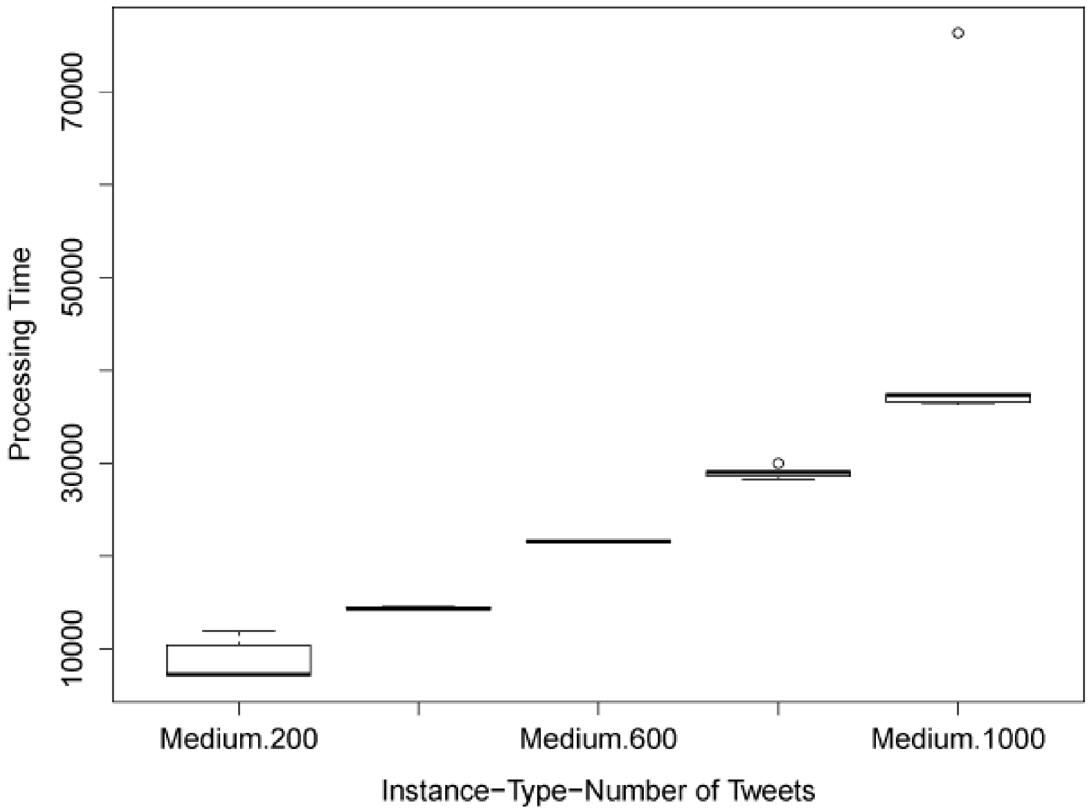

Figure 3.5: Execution time in medium instances

J48 is one of the most popular and powerful decision tree classifiers. J48 is an optimized implementation of C4.5. C4.5 can produce both decision tree and rule-sets; and construct a tree. C4.5 models are easy to understand as the rules that are derived from the straightforward technique [19].

J48 first creates the tree based oh following criteria. Identify if every class belong to the same class. Then, for each attribute it calculates information gain. Finally, look for the better attribute to be splitted. Once, tree is ready it calculates the measure of disorder of data, also called *entropy*. To complete the process it is important to pruning the tree, which allows to discard subsets of data that are not well defined [3].

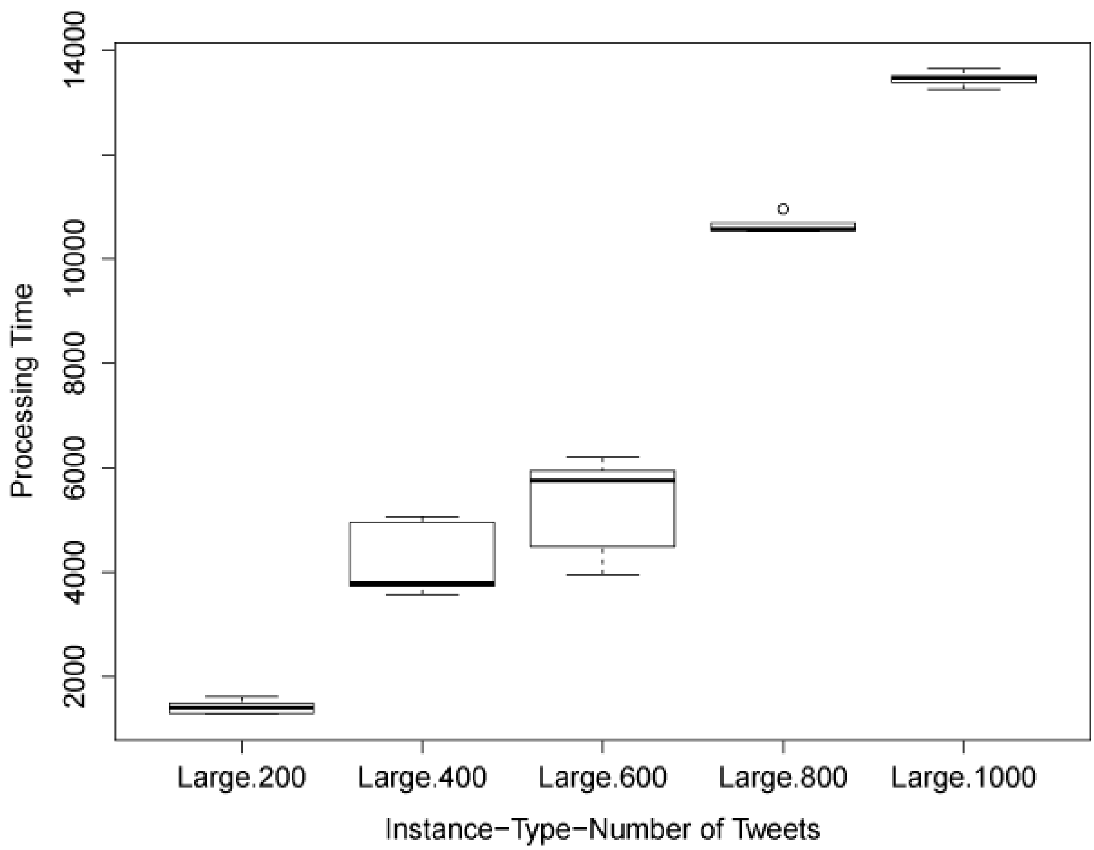

Figure 3.6: Execution time in large instances

3.4.3 Resource Selection

Once we have constructed our training set we can make a query that identifies which resource can process given number of tweets before deadline. In the next step when we have a list of resources which have aforementioned capacity, resources that minimize the cost are getting selected. In each time slot with the information provided by prediction engine the provisioning component add or remove resources from the cloud to satisfy user requirements with give constraints.

Chapter 4
Performance Evaluation

Tn order to realize how effective is the proposed architecture it is required to test it in several scenarios; For this purpose and for our experiment, we have make an environment which we can use to reproduce the experiments. To achieve this as a first step, a workload generator application that reproduce real data streams based on real collected tweets.

4.1 Experiment Setup

We aim at comparing our strategy with a strategy which is statically provision resources based on pick load. First, we define *provisioning for peak*. This approach takes identify the time slot that has highest number of tweets and provision an instance that can handle it. The instance then uses to process the whole workload until a new pick is detected. We can identify the pick in Figure 3.3, which is of 1000 tweets.

On the other hand, we have our strategy which uses times series prediction for load estimation and machine learning for resource selection. In other words, it will allocate resources according to prediction in an hourly basis and dynamically.

To complete the experiment we need to define deadline constraint. In this scenario we will use a deadline constraint of 10 seconds. This means that all data should be processed up to 10 seconds regardless of its volume.

For our evaluation we have to include costs of each instance. Table 4.1 shows costs obtained from Amazon web site.

Forecast period is of 13 hours. Which is the time we predicted using prediction engine

Instance	Cost per hour
Medium	0.098 AUD
Large	0.196 AUD
X-Large	0.293 AUD

Table 4.1: Cost per hour of Amazon EC2 instances [1]

Strategy	Cost (13 hours)
Peak	5.096 AUD
Proposed	1.862 AUD
Difference (Cost Saved)	**3.234 AUD**

Table 4.2: Results of experiment

described in Section 3.2.

4.2 Experiment Result

With all inputs defined in Section 4.1 the results we obtain are shown in Table 4.2

As we can see our proposed strategy processed the stream data spending 36.54% less money (3.234 AUD) than provisioning for peak strategy in 13 hours. With those results it is possible to estimate a the money saved in certain periods of time. For example, in a day, it would be possible to save 5.97 AUD, in a week 41.79 AUD, in a month 177.87 AUD and yearly 2134.44 AUD. It represents a high profit for companies that wants to accomplish its tasks spending a specific budget as well as by a defined deadline.

To conclude we can mention that the proposed architecture provides us of enough tools to accomplish our tasks both on time and within budget. However, it is important to notice that the correct implementation of prediction engines used in our architecture is very important to guarantee those results.

Chapter 5
Conclusion

Big Data refers to information that cannot be processed and analyzed with traditional process and tools effectively. With the new requirements of processing data from all types of devices, stored in various formats, real-time, we require novel techniques which we have discussed in this thesis.

At this point, stream processing emerged as a solution providing tools required to perform tasks in real-time. Using those tools in a Cloud environment it is possible to reach objectives of processing Big Data in real-time by scaling up and down and without capital investment. This environment allows us to process any tasks we want just paying the for the period which the resources have been utilized. However, we have to consider that users require to minimize their cost as well. Here relies the importance of our project.

Our proposed architecture minimizes the cost by dynamically allocate resources which are just enough to process the load we receive in the next time slot. To achieve this, it implements a machine learning approaches in order to both forecast the load and find the most suitable resources for our task at a specific point of time.

The effectiveness of our architecture was tested and compared with a provisioning for peak approach which provides resources to complete tasks based on peak workload. Based on experimental results we realize that our approach allows dynamic scale of resources while processing latency remains within desired levels which keeps the cost minimum.

Bibliography

[1] Amazon, "Amazon EC2," https://aws.amazon.com/ec2/, 2014, [Online; accessed 15-May-2014].

[2] D. Beimborn, T. Miletzki, and S. Wenzel, "Platform as a service (paas)," *Business & Information Systems Engineering*, vol. 3, no. 6, pp. 381–384, 2011.

[3] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *International Journal*, vol. 3, no. 6, 2013.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[5] CERN, "Computing," http://home.web.cern.ch/about/computing, 2013, [Online; accessed 07-April-2014].

[6] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] T. A. S. Foundation, "Apache ZooKeeper," http://zookeeper.apache.org/, 2014, [Online; accessed 21-May-2014].

[8] A. Ishii and T. Suzumura, "Elastic stream computing with clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 195–202.

[9] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big data processing in cloud computing environments," in *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on*. IEEE, 2012, pp. 17–23.

[10] D. Le Phuoc, "A native and adaptive approach for linked stream data processing," Ph.D. dissertation, 2013.

[11] Manjrasoft, "Aneka," http://www.manjrasoft.com/products.html, 2014, [Online; accessed 15-May-2014].

[12] N. Marz, "Apache Storm," https://github.com/apache/incubator-storm, 2014, [Online; accessed 21-May-2014].

[13] H. Matsuura, M. Ganse, and T. Suzumura, "A highly efficient consolidated platform for stream computing and hadoop," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 2026–2034.

[14] P. Mell and T. Grance, "Draft nist working definition of cloud computing," *Referenced on June. 3rd*, vol. 15, 2009.

[15] S. P. Mirashe and N. V. Kalyankar, "Cloud computing," *CoRR*, vol. abs/1003.4074, 2010.

[16] N. I. of Standards and Technology, "NIST Cloud Computing Program," http://www.nist.gov/itl/cloud/, 2011, [Online; accessed 07-April-2014].

[17] T. I. S. of Units, "Resolution 4 of the CGPM (1991)," http://www.bipm.org/en/CGPM/db/19/4/, 2014, [Online; accessed 15-May-2014].

[18] ——, "SI Prefixes," http://www.bipm.org/en/, 2014, [Online; accessed 15-May-2014].

[19] A. Rajput, R. P. Aharwal, M. Dubey, S. Saxena, and M. Raghuvanshi, "J48 and jrip rules for e-governance data," *International Journal of Computer Science and Security (IJCSS)*, vol. 5, no. 2, p. 201, 2011.

[20] P. Russom *et al.*, "Big data analytics," *TDWI Best Practices Report, Fourth Quarter*, 2011.

[21] N. Sadashiv and S. D. Kumar, "Cluster, grid and cloud computing: A detailed comparison," in *Computer Science & Education (ICCSE), 2011 6th International Conference on*. IEEE, 2011, pp. 477–482.

[22] B. Satzger, W. Hummer, P. Leitner, and S. Dustdar, "Esc: Towards an elastic stream computing platform for the cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 348–355.

[23] R. Stephens, "A survey of stream processing," *Acta Informatica*, vol. 34, no. 7, pp. 491–541, 1997.

[24] S. Tsuchiya, Y. Sakamoto, Y. Tsuchimoto, and V. Lee, "Big data processing in cloud environments," *Fujitsu Sci. Tech. J*, vol. 48, no. 2, pp. 159–168, 2012.

[25] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.