# THE COSTS AND BENEFITS OF TURNING DATA INTO INFORMATION USING BIG DATA SYSTEMS

A Thesis

by

Melih Koca

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Computer Engineering

in the
Department of Computer Engineering

Özyeğin University
Aug 2014

# THE COSTS AND BENEFITS OF TURNING DATA INTO INFORMATION USING BIG DATA SYSTEMS

Approved by:

---

Professor İsmail Arı, Advisor
Department of Computer Engineering
*Özyeğin University*

---

Professor Hasan Sözer
Department of Computer Engineering
*Özyeğin University*

---

Professor Ali Özer Ercan
Department of Electrical and Electronics Engineering
*Özyeğin University*

Date Approved: 22 August 2014

*To my supportive parents...*

# ABSTRACT

This thesis explains problems of and solutions for storing and processing big volumes and streaming types of data in a cost-effective way for enterprise companies. There are several problems like operational, infrastructural and usability problems about these new concepts of Big Data. The basic data processing concepts are not new, but the data generation volumes and velocities are pushing the limits of centralized architectures. Distributed systems using distributed programming models such as the Hadoop framework are used today to handle big data problems. This thesis will try to combine the structural, architectural and financial issues to address big data storage and processing problems and will give practical examples based on real-life experiences from several big data applications in different sectors including mobile telecommunications, finance and oil-gas fields.

# ÖZETÇE

Bu tez kurumsal firmalardaki büyük miktarlardaki verinin, işlenmeside ve saklanmasındaki zorlukları ve çözümlerini açıklmaktadır. Yeni ortaya çkan büyük veri kavramında işletimsel, mimari ve kullanımsal olarak birçok problem bulunmaktadır. Temel veri işleme yöntemleri yeni olmamasına karşın, oluşan veri miktarı ve hızı, merkezi mimarilerin sınırlarını zorlamaktadır. Hadoop gibi, dağıtık işleme modellerini kullanan, dağıtık mimariler günümüzde büyük verinin işlenmesi ve saklanmasında anahtar rol oynamaktadırlar. Bu tez büyük veriyi işlemede, yapısal, mimari ve finansal olarak sorun teşkil eden durumların gerçek hayat deneyimleri üzerinden, telekominikasyon, finans ve petrol rafinerileri gibi sektör bazlı çözümlerini açıklamayı amaçlamaktadır.

# ACKNOWLEDGEMENTS

Thank you İsmail Arı, for giving me a solid academic training, not just about computer science, but work ethics and humanity. He is also one of the best teacher and human being in my life.

Thank you Ali Özer Ercan and Hasan Sözer for supporting me during my Masters program and thesis stage.

Thank you Serkan Uçkun Uçkunlar and Selçuk Sözüer for their support and work.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABBREVIATIONS

| | |
|---|---|
| **GPS** | **G**lobal **P**ositioning **S**ystem |
| **RDBMS** | **R**elational **D**ata **B**ase **S**ystems |
| **CPU** | **C**entral **P**rocessing **U**nit |
| **PoC** | **P**roof **o**f **C**oncept |
| **I/O** | **I**nput - **O**utput |
| **ETL** | **E**xtract **T**ransform **L**oad |
| **SAN** | **S**torage **A**rea **N**etwork |
| **RAID** | **R**edundant **A**rray of **I**ndependent **D**isks |
| **JBOD** | **J**ust a **B**unch **O**f **D**isks |
| **NAS** | **N**etwork **A**ttached **S**torage |
| **OS** | **O**perating **S**ystem |
| **rpm** | **r**evolutions **p**er **m**inute |
| **Mhz** | **M**ega **h**ert**z** |
| **GPU** | **G**raphical **P**rocessing **U**nit |
| **SaaS** | **S**oftware **a**s **a** **S**ervice |
| **PaaS** | **P**latform **a**s **a** **S**ervice |
| **IaaS** | **I**nfrastructure **a**s **a** **S**ervice |
| **SMB** | **S**mall-**M**edium **B**usinesses |
| **SETI** | **S**earch for **E**xtra-**T**errestrial **I**ntelligence |
| **p2p** | **p**eer **to**(2) **p**eer |
| **CSV** | **C**omma **S**eparated **V**alues |
| **XML** | e**X**tensible **M**arkup **L**anguage |
| **API** | **A**pplication **P**rogramming **I**interface |

| | |
|---|---|
| **MPI** | **M**essage **P**assing **I**interface |
| **HDFS** | **H**adoop **D**istributed **F**ile **S**ystem |
| **PC** | **P**ersonal **C**omputer |
| **THP** | **T**ransparent **H**uge **P**ages |
| **jdbc** | **j**ava **d**ata **b**ase **c**onnector |
| **ODBC** | **O**bject **D**ata **B**ase **C**onnector |
| **SMS** | **S**hort **M**essage **S**ervice |
| **GSM** | **G**lobal **S**ystem for **M**obile |
| **NSN** | **N**okia **S**iemens **N**etwork |
| **WAP** | **W**ireless **A**pplication **P**rotocol |
| **ZTE** | **Z**hongxing **T**elecommunication **E**quipment Corporation |
| **OPEX** | **OP**erational **EX**penses |
| **CAPEX** | **CAP**ital **EX**penses |

# CHAPTER I

# INTRODUCTION

In modern enterprises there are two types of data sources, which are human generated data, machine generated data. The total amount of the data is growing increasing exponentially. GPS systems, base stations of a telecommunication company, smart roads, smart cities, smart electricity meters, sensors in an Oil & Gas company, Internet activities of a person, social media data, banking activities, etc. are generate enormous amount of data every day. In certain cases there is no need to store the incoming data, but to process them to generate real time alarms and inform people about anomalies. However in most cases companies have to store the data for reporting, analysis or regulative purposes. In both cases, there is huge amount of data to be dealth with, and traditional systems such as mainframes and RDBMS's do not scale in a cost effective or affordable way.

## *1.1 Big-Data*

What is Big-Data? and Why is it a problem? Is it just about the data volume? This question was answered by IBM researchers in a book called "Understanding Big Data"[1]. According to them, when the data have high volume, velocity, variety, and/or veracity properties, we call this data as Big-Data.

- **Volume:** Usually the high volume aspect will be the necessary condition. Because with this property the data becomes unmanageable easily.
- **Velocity:** If the data is piling fast and the processing environment cannot catch up with its arrival speed, the system becomes inefficient or looses data.
- **Variety:** In most Big-Data applications there are more than one data source

and type. For example for targeted advertisement applications, most of the companies are enriching their customer info with social media posts, comments etc. to generate personalized campaigns. Structured and unstructured data are also mixed together.

- **Veracity:** The data quality must be preserved while the data volume is increasing. Especially the sensor type of data, can be broken and needs to be validated first before processing.

## *1.2 Types of Data Processing Tasks in Enterprise Companies*

Almost every enterprise today, not just technology based or related companies realised the power of information. There are lots of data analysts and computer engineers that try to find ways to grow their business by utilizing the computing analytics's tools. Some data processing tasks are CPU intensive, and some are I/O intensive. I will try to explain these processing types in the following subsections.

### 1.2.1 CPU Intensive Data Processing

CPU intensive tasks usually do not require Big-Data to work on. There is small amount of data and the most of the computing done inside the CPU. For example, assume a banking example where, there is a customer table and a table for stocks of the public companies. We have done this kind of work as a PoC (Proof of Concept) for Turkey's biggest bank. The purpose of the PoC is to determine, which stocks to buy together for highest profit. This PoC gets two inputs, one is the lists of stocks of all their customers, that the customers are already have the stocks and the lists of available stocks. These two inputs occupies about 100 KB of text data, but there are about $2^{28}$ calculations and this task was finished by four IBM server Hadoop cluster in 1450 minutes (1 Day and 10 minutes). This kind of calculations one being used by banks, statisticians, cryptologists (to get big prime numbers to encrypt data safer).

### 1.2.2 I/O Intensive Data Processing

In I/O bound processing, input and output devices such as hard disks, network etc. are much more important than CPU because, the work itself requires scanning the data from top to bottom. For example in ETL (Extract Transform Load) type of jobs, the work begins with reading the data line-by-line because the application needs the extraction of the required fields or tuples. the next thing about the process is the transformation of data in another format, and the final step is loading the transformed data into another system or computer. In these kind o processes, CPU is not so important because for today's technology almost every CPU is faster than any kind of I/O devices. These kind of processing is being used by banking for governmental regulative purposes, for all industries to backup and restoration purposes, and system integrations for adaptability purposes (an ETL job works as an adaptor between two systems).

## *1.3 Data Storage Technologies*

Processing Big-Data solves just one part of the problem. In most cases there is a storage part and again in most cases the data is stored for future analysis, reporting, batch processing, querying purposes. Data storage is the most expensive (price wise) part of the Big-Data. We will try to explain storage types and technologies to get an idea why this part is the expensive one. There are two server-ready and one experimental usage of storage types:

- **SAN:** SAN (Storage Area Network) technology developed to create disk arrays to get maximum bandwidth with distribution of the data across the disk array. SAN appliances do not have usually public network connection, but it connects directly to the server via special SAN cards and those cards sold separately with a yearly renewal license. This technology allows to allocate block storage spaces (slice of storage space from each disk) for lots of servers. This technology comes

with reliable storage and on-line backup capabilities. Because of the special card and port licensing method, this technology can store 1 TB with a price of 10,000 USD. For most of the Enterprise companies this is a huge expense, but this technology comes with lots of capabilities for data integrity and recovery.

- **JBOD:** JBOD (Just a Bunch Of Disks) is a simple technology. It is a network shared disk array. There are no advance storage technologies in it, so in most cases this storage technology could not give reliable service to its users. There are network shared usage, so technology limited by the network speed and architecture, although it is an affordable technology, it is not suitable for Big-Data and Batch processing purposes.

- **NAS (Experimental for Big-Data Platforms):** NAS (Network Attached Storage) technology does not include built-in reliability or distribution functionality, but it has network interface and the user can mount this storage type to a server or computer as a local storage device. NAS technology is not used for handling Big-Data but, we experimented NAS storages with Hadoop to see performance results because the technology is affordable and has potential to get benefit from distribution. For Hadoop its important to increase distribution for both for machine wise and the distinct storage device wise. All of the Linux OS distributions support multiple mounting points for network devices so, in theory the user can mount unlimited NAS devices to get distribution and increasing bandwidth, but in reality the user is limited to the network bandwidth. Also almost all of the NAS devices run with 7200 rpm commodity disks with powerless CPU included (about 600 Mhz operating CPU of the NAS box). This is not a high-end technology but, limitations permit to get benefit from NAS technology for Hadoop typed distributed systems. May be in a controlled and specialized network architecture, this technology could produce high I/O throughput, but

this architecture will require dedicated network (separated from public network) and it will not be a cost effective or ideal solution for distributed systems.

## *1.4 Computational Technologies*

There are so many data types generated by machines and people, such as structured, unstructured, text based, binary, video, audio etc. all of these data types may processed by one system or programming logic, but if the user wants to go beyond the singular hardware unit's limit, some programming and architectural logics generated to get the best performance and benefit from the hardware to generate affordable and efficient systems. For example, to get performance and scalable architectures for matrix types of data, such as video, image, or statistical matrix data there are GPU based solutions because the problem is divisible to chunks and all of the chunks could be processed with a powerless small CPU's inside the graphic card (part of the GPU) and after processing of each chunk is done we can merge all of the individual chunks to solve the original problem. GPU is a specialized hardware, that developed to solve matrix type of problems and it has its own programming logic. This approach enables us to get benefit of performance and efficiency for just one type of the problem with specialized hardware. This section aims to explain computing approaches and their pros and cons for data type, affordability, efficiency and operational difficulties.

Big-Data is today's hottest computational problem, but it is not a new concept. Today we are talking about terrabytes, petabytes or even exabytes, but our computational power and technologies creates this much data. The concept of the Big-Data existed almost with invention of the computer. At the beginning 1 MB was "big", because the computational power was not enough to handle this much data. As we say, the Big-Data was a problem since the invention of the computer, there would be a solution for the problem. There are two approaches, first one is an old technology called Scale-Up or centralized architecture, and the second one is a relatively new

concept called Scale-Out or distributed architecture.

### 1.4.1 Centralized Architectures

This architecture type based on a powerful single mainframe (basically it is a server, with lots of CPU, disk and memory slots). When the mainframe can not respond to the client's needs, performance could be increased by adding extra memory, CPU and disk etc. As long as this architecture is based on a single powerful mainframe, there will be some limits in terms of available slots, compatibility etc. If all the slots are full, the client can not increase the performance, and client have to buy new mainframe and have to migrate data, applications and the other systems to the new mainframe. Scale-up architecture has useful in some specific areas but it has limits and almost any type of Big-Data will exceed its limits eventually.

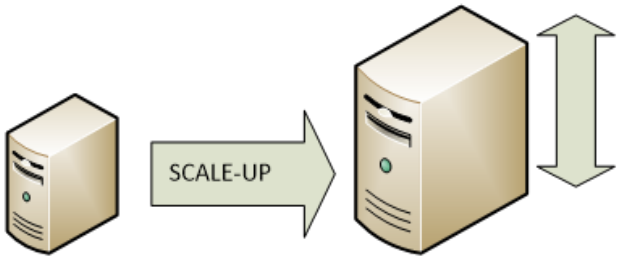


**Figure 1:** Scale-Up Architecture

### 1.4.2 Distributed Architectures

Scale-Out architecture is commonly being used by distributed systems, token-ring architectures with p2p communication, grid computing and new generations of cloud computing environments. This architecture offers flexibility and scalability features with adding new commodity servers (affordable low-end servers) to get performance increase. This architecture flexible because, there is no important component or server for the system and if one of the servers becomes unreachable, the system remains running. This architecture scalable, because as long as the system owner

adds new servers, performance will increase linearly. There are modern cloud based frameworks like Hadoop, that are using this kind of architecture to keep working despite unexpected hardware issues.

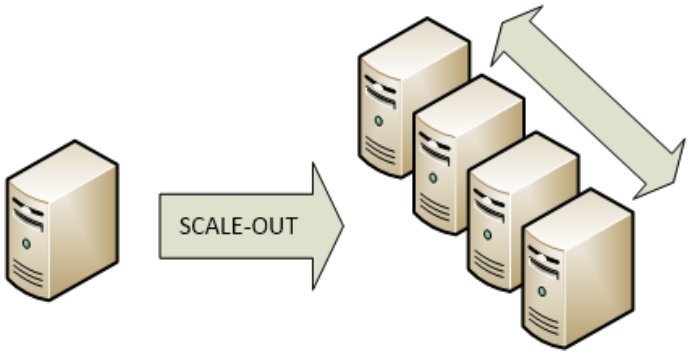


**Figure 2:** Scale-Out Architecture

The definition of distributed computing, is making a cluster of computers to work as one and shared computer system. The power of the distributed computing comes from the flexibility and scalability of the architecture. The owner of the system can add or remove bunch of servers to manage the usability of the system.

Distributed computing based on "Divide and Conquer" strategy. Algorithmic strategy, can be applied to specific type of problem sets, those can be solved by dividing the problem into smaller chunks of problems, but these chunks have to be the same type of problem with the original problem. All of these chunks becomes small problems, those can be solved by one computer, are distributed all over the clusters of computers. After solving all the small problems, the merging process will begin to get the original problem's solution. This concept is being used by recursive algorithms, grid computing, graphic card computing, and distributed systems such as Hadoop.

### 1.4.3 Cloud Computing

Definition of cloud computing is giving applications (SaaS), platforms (PaaS) or infrastructures (IaaS) as a service on a shared asset (hardware, platform or software) to a private group or everyone who pays the service tuition. Cloud computing gives chance to small groups, companies or even an ordinary person to make their calculations or job without any big investments for hardware, data center, software, service or operational costs. With cloud computing, the location does not matter, because the service is accessible from everywhere via the Internet and the users can reach their data or application where ever they want.

The main benefit of the cloud computing is the sharing of resources, because in most cases the systems are running with low utilization. With shared usage, servers are running with high utilization and this causes efficiency for electricity, hardware and owning costs.

There are three types of cloud services, that are differentiates access and usage types:

- **Private Cloud:** This type of cloud has limited access to some group or organization's members. Private cloud architectures are commonly used by large enterprise companies and the access is limited to company employees. Usually this type of cloud services contain company specific or privacy sensitive data so, private clouds are not connected to the Internet for security and for some cases governmental privacy regulations. Also since the owner of the cloud is an enterprise company, private cloud services has operational and owning costs too.

- **Public Cloud:** Public cloud services, as the name implies, are public access to every one who agrees to pay the tuition. Almost all public cloud service providers do not use monthly or yearly licensing fees, but they are using "pay


**Figure 3:** Private Cloud Example

per use" or "pay per hour" pricing types. This pricing strategy gives chance to SMB's (Small Medium Business) for competing with larger competitors. Because there are no owning, operational or electricity costs for them and they can use their resources efficiently and they are not paying while they are not using the system.

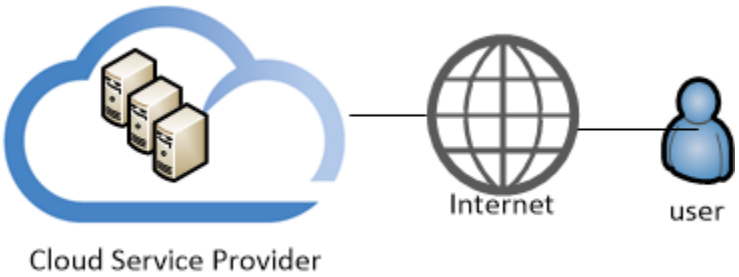


**Figure 4:** Public Cloud Example

Also the scaling problem is solved by the service provider with adding new computers (Virtual or Physical servers). This feature is the most beneficial property of the public cloud for the seasonal systems like or on-line video broadcasting company. For example during the world cup of football (soccer) they can allocate more servers and after the world cup ends they can release the servers, which they do not need any more.

- **Hybrid Cloud:** Hybrid cloud is a combination of public and private cloud

systems. The private data, like customer data of a bank, remains in the private cloud of the bank and when they want to do some data enrichment process with social media data for targeted marketing campaign they can use both data with advantages of the both systems. The customer data could be used for other operational needs and it cannot be removed, but when the marketing campaign is over, bank could close the public part of the cloud.

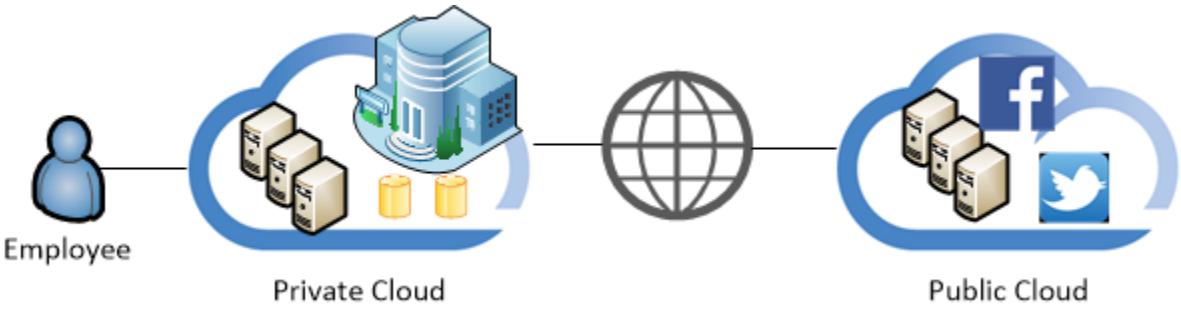


**Figure 5:** Hybrid Cloud Example

### 1.4.4 Grid Computing

Grid computing is a kind of parallel computing with loosely coupled computers trying to solve large problems like SETI@home[2] project. SETI project aims to collect data from space with radio telescopes distributed all over the world and after collecting the data, SETI institute processes the data and looks for meaningful radio signals, that might be generated by the intelligent life forms. There are lots of radio telescopes scanning the space continuously and produces terabytes of data for every day and this data needs to be processed in one day. This is a hard job when there is continuously generated new data, because of this issue University of California, Berkeley University decided to start SETI@Home project. SETI@Home is simply a screen saver application, when the user is not using the computer, screen saver turns on and computer makes calculations about radio data, that is collected from the radio telescopes and returns its results back to the SETI institute.

# CHAPTER II

# DATA PROCESSING

## *2.1 Batch Data Processing*

Batch data processing is very important for most of the technology companies, ETL jobs, data migrations, targeted advertisements, backup and restring processes for the new generations of archiving systems etc. This kind of processing begins with a pile of data, that waits for the processing phase. Almost for every case there is an easy task to do, but the data volume, velocity and variety of the data complicates this simple task. For example in Turkey, there are governmental regularities and all of the Telecommunication companies have to give their customers' Internet access, call, SMS logs to the government for every day in a certain format. Telecommunication company buys their hardware from variety of telecom vendors, such as Ericcsson, Huawei etc., and all of the hardware produces log data in different format (CSV, XML etc.), and these are not acceptable for government because their format is common and all the telecom companies have to transform their data into the common data format with ETL jobs. The raw log data is collecting from systems and a periodically working ETL job processes the data batch-by-batch.

### 2.1.1 Hadoop

Hadoop is a distributed, scale-out architecture based computing and storage framework. The framework allows its users either write applications through its API or extend capabilities by installing software packages, such as Hive, HBase etc.

The idea behind the Hadoop is getting benefit from power of the distribution and solving problems with dividing them. The parallelism or parallel computing itself is not a new concept, it is being used by MPI[3] and alike applications since late

1992, but parallelism done by programmer. Programmer have to think about, CPU management, synchronization, dividing the problem into smaller chunks etc. there are so many parameters to write successful parallel applications. Hadoop solved all of the parallelism, synchronization, resource management issues and it has a flexible, reliable and scalable architecture with high capacity storage capabilities.

### 2.1.2 Little History About Hadoop

The Hadoop's story begins with two publications from Google Labs. The first publication was about specialized large scale distributed file system called GFS[4] in 2003 and the other publication was about the data processing with large scale clusters[5] in 2004. These publications were the base of the Hadoop project. After these publications, two Yahoo members, Doug Cutting and Mike Cafarella, started to develop Hadoop, and after two years of hard working they released the first working version of Hadoop and in 20007 they open-sourced the project to the Apache foundation.

### 2.1.3 Virtual Architecture of Hadoop

There are two architectural views for the Hadoop, first one is the virtual view. This is the conceptual an algorithmic view of the Hadoop. There are two components. These are: Map-Reduce and HDFS. This view represents the conceptual and high level architecture of the Hadoop.

- **Map-Reduce:** Map-Reduce is the processing engine of the Hadoop, and it is a programming style. The logic behind the Map-Reduce comes from divide and conquer algorithm strategy.

  In Hadoop, every processing data must be a key-value pair. If the file format is text based, Hadoop gets all the text line-by-line and assigns the offset byte number as a key to the related line. After this loading and dividing process, Hadoop distributes all the lines to the mappers (or machines). Mapper is a

specialized method, that takes key-value pair list process the data and pass the resulting data to the reducers. For ETL type of jobs mapping phase would be the end of the job (Map-only job). After Mapping phase is done for a task there is a automatic phase called shuffle. Shuffle phase simple includes sorting and hashing the mapper outputs. Hashing needed for the reduce phase because Hadoop decides which data will be reduced by which reducer (or machine) with simply hashing the mapper output and the hash result maps to a specific reducer. After the reducer has done its job, writes results back to the disk (HDFS). There is a word count Map-Reduce example shown in figure 6.

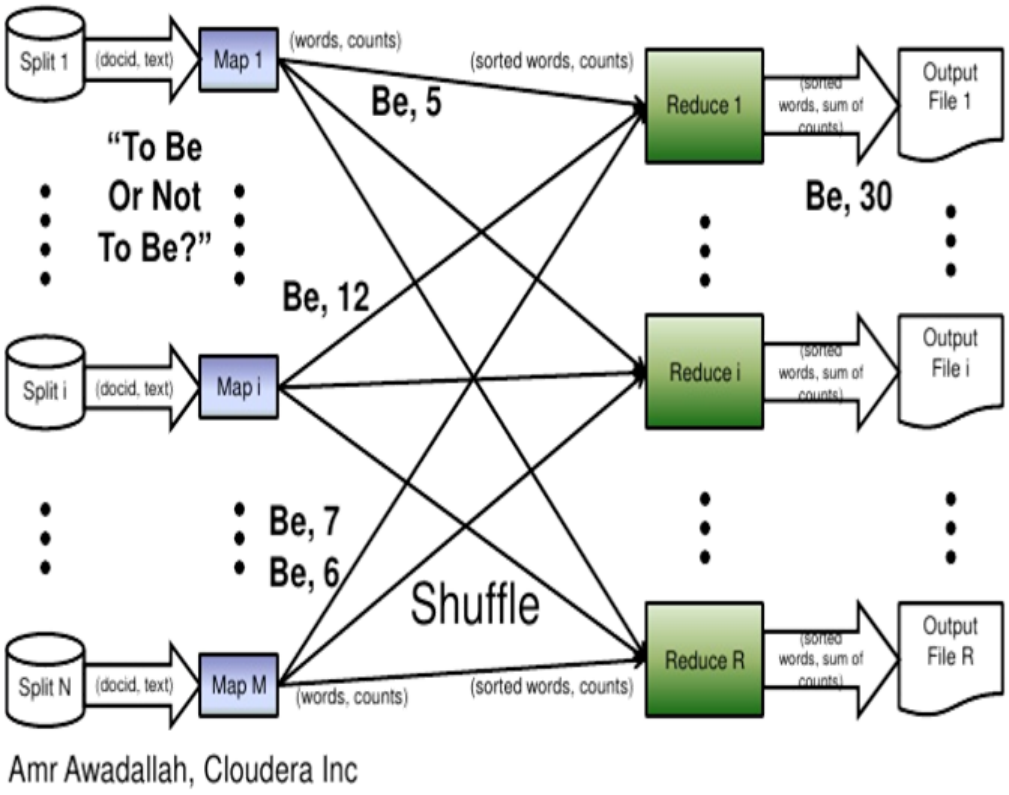


**Figure 6:** Word Count Example Diagram.
Source: http://www.slideshare.net/cloudera/hadoop-distributed-data-processing

- **HDFS:** Hadoop Distributed File System (HDFS) is the storage part of the Hadoop. HDFS designed for big data and it is a virtual file system. Virtual

means, for every disk of the all cluster's nodes, there is a real file system under the HDFS. HDFS's main purpose is managing all of the cluster's total storage capacity as a one big storage system.

HDFS does lots of management tasks at once. When a file located to the HDFS, if the file is bigger than the HDFS block size (independent from the local file system's block size and by default it is 64 MB), HDFS splits the file into block sized chunks and distributes to all over the cluster by itself (see figure 7). This blocking mechanism is the only part of the Hadoop, which is written in native code (C and C++). Distributed systems have some issues about unbalanced or different machine combinations. HDFS is smart enough to manage the storage and balance the entire cluster in storage wise. Also storing data reliably is another issue, and HDFS replicates (by default replication factor is 3) the data and redistributes the replica data to other nodes, so if one node becomes unreachable the system remains working.

### 2.1.4 Physical Architecture of Hadoop

The second architectural view is the physical view of the Hadoop. This view represents the master-slave hierarchy and it has five major components.

In most cases there is no difference between master and slave servers in terms of hardware. The installation of the components determines the master and the slave nodes. Hadoop has five main components. These are:

- Name Node: This is a master node and HDFS component. Name node is a metadata and management server. It keeps track of the blocks and files. The other job of the name node is managing the file locations, keeping the cluster balanced and providing a reliable storage system.

- Secondary Name Node: This is a snapshot server, and gets snapshots from the name node periodically (by default one hour) to restore system from the name

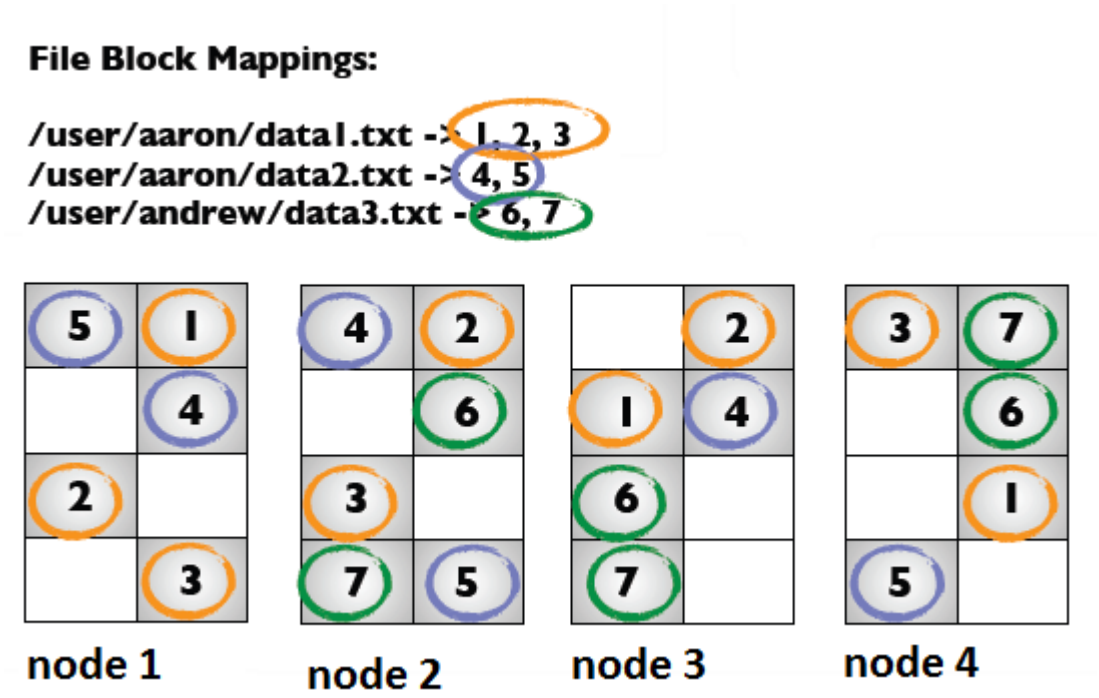


**Figure 7:** HDFS Replication Example.

node failure situations.

- Data Node: This is the slave node and HDFS component, and this is the where the data actually storing. It is a bridge from node's local file system to name node (HDFS).

- Job Tracker: This is a master node component for the map-reduce part of the Hadoop. Its main job is dividing the Big-Data job into smaller tasks and assigning these tasks to the nodes in a balanced way.

- Task Tracker: This is a slave node component for the map-reduce part of the Hadoop. This is where mappers and reducers run on, and basically it is a processing engine.

## *2.2 Big-Data Warehousing*

Storing the Big-Data, solves just one problem, but almost all of the enterprise companies gets some reports, and do some analytical processes. Writing Map-Reduce jobs require lots of programming knowledge, and the most of the analysts are do not have this knowledge, so they can learn or already know the SQL interface. Therefore some Hadoop packages emerged to develop for warehousing, reporting and analysing purposes. There are two open-sourced Hadoop packages for warehousing.

### 2.2.1 Hive

Apache Hive is a distributed warehousing software, and it is developed by Facebook in 2008 to add SQL and warehousing capabilities to Hadoop. The main purpose of the system is reading certain schema based formats as an input and runs HQL queries on the data. HQL is a specialized version of SQL_92 standard with extra Hadoop specific commands added. Because of extension of the SQL_92 standard, they called as HQL.

Hive simplified the Hadoop operations for structured data. There are no prerequisites except the SQL knowledge, easy to use and control. After the installation process, the user could open the command line interface and start writing HQL commands. Extensional commands are for file data import and export commands, and these commands required by the distributed usage nature of the Hadoop and they are required.

### 2.2.2 Impala

Former Cloudera Impala, now called Apache Impala is an indexing mechanism for Apache Hive to add low latency capabilities to the Hive. It has its own indexing service and the user can start using Impala with just running one indexing job (this is a one time job, and it is for importing the index data to the memory). After indexing job finishes, the user can use the Impala like using Hive.

## *2.3 Stream Data Processing*

Stream data processing is a must, where low latency is critical and it sis required to get on action in a short time after or even before the event happens. If the event has a pattern, the streaming engines could monitor for the pattern, and if the pattern comes up, the system could inform the people, who is in charge.

Almost every device is generating data and streaming the data to a central place. For example in Istanbul, almost all public transportation vehicles have a GPS device and in every certain amount of time they are generating a GPS data and sending to a public transportation center. The data may not seem very important, but just this GPS data could generate lots of useful applications, like when the bus comes to a station?, or if there is a problematic situation like engine failure and the bus does not move the system could generate an alarm and send help to the bus, or the bus following the right route.

Also there are lots of applications for the streaming data like, wearable technologies, with hearth monitor, and controls the patient every time, and if there is a irregular pulse, the system could catch the irregularity, and send paramedics to the patient, before the patient gets a hearth attack.

### 2.3.1 Complex Event Processing

Today's complex sensor networks, mobile systems, GSM enabled devices etc. generate huge amount of data, in some the data needs to be processed on the fly to get immediate results and actions. There are lots of applications, like customer experience management systems, alarming systems, targeted marketing systems which are mostly using complex event processing engines to catch events from the data with a set of predefined rules. These rules could be user defined rules or algorithm generated rules, like the output of the rule growth algorithm.

### 2.3.2 Rule Mining

Rule mining[1] is the task of finding interesting relationships in large datasets. These interesting relationships can take two forms. The first form is frequent item sets or association rules. Frequent item sets are a collection of items that frequently occur together. The second form to view interesting relationships is association rules. Association rules suggest that a strong relationship exists between two items. In rule mining, first of all we need to find frequent patterns after that we create association rules base on frequent item patterns.

There are two most important metrics in rule mining that measures the relationships in a collection of items.

- **Support:** The support of an item set is defined as the percentage of the dataset that contains this item set. Support applies to an item set, so we can define a minimum support and get only the item sets that meet that minimum support.

- **Confidence:** The confidence is defined for an association rule like {A} $\rightarrow$ {B}(Let's say A and B are products in a grocery. We know that, if someone buys item A, and also he/she buys item B in same transaction/shop ). The confidence for this rule is defined as support({A, B})/support({A}).

The support and confidence are ways we can quantify the success of our rule mining analysis.

There are many algorithms to discover relationships between items in a collection of items. Apriori and FP-Growth are well known algorithms in this field. These algorithms generate frequent item sets. To find association rules, we use those item sets that those algorithms produced.

- **Apriori Algorithm:** As mention before, in rule mining, we first need to find the frequent item sets, and then we can find association rules. The way to

[1]Rule mining explanations brought from http://ylzhj02.iteye.com/blog/2078673

find frequent item sets is the Apriori algorithm. The Apriori algorithm needs a minimum support level as an input and a data set. The algorithm will generate a list of all candidate item sets with one item. The transaction data set will then be scanned to see which sets meet the minimum support level. Sets that do not meet the minimum support level will be thrown away. The remaining sets will then be combined to make item sets with two elements. Again, the transaction dataset will be scanned and item sets not meeting the minimum support level will be thrown away. This procedure will be repeated until all sets are finished.

- **FP-Growth Algorithm:** FP-Growth algorithm is another algorithm that is used for finding frequent item sets. It builds from Apriori but uses some different techniques to accomplish the same task. That task is finding frequent item sets or pairs, sets of things that commonly occur together, by storing the dataset in a special structure called an FP-tree.

  The FP-growth algorithm is faster than Apriori because it requires only two scans of the database, whereas Apriori will scan the dataset to find if a given pattern is frequent or not Apriori scans the dataset for every potential frequent item. On small datasets, this is not a problem, but when you are dealing with larger datasets, this will be a problem. The FP-growth algorithm scans the dataset only twice.

  The basic approach to finding frequent item sets using the FP-growth algorithm has two steps. The first step is building the FP-tree, and the second step is mining frequent item sets from the FP-tree.

  To find association rules, we first start with a frequent item set. We know this set of items is unique, but we want to see if there is anything else we can get out of these items. One item or one set of items can imply another item. For

example if we have a frequent item set, {A, B, C}, one example of an association rule is A B $\rightarrow$ C. This means, say, if someone purchases A B, then there is a statistically significant chance that they will purchase C.

Similar to frequent item set generation , we can generate many association rules for each frequent item set. It would be good if we could reduce the number of rules to keep the problem tractable. So, we use metric confidence for reducing rules.

### 2.3.3 Clustering

Clustering is all about organizing items from a given collection into groups of similar items. These clusters could be thought of as sets of items similar to each other in some ways but dissimilar from the items belonging to other clusters.

Clustering a collection involves three things:

- **An algorithm:** This is the method used to group the items together.
- **A notion of both similarity and dissimilarity:** A metric to determine how item belongs to same group or not.
- **A stopping condition:** When should stop the grouping process.

Clustering is simply a process of putting things into groups. To do more than simple grouping, you need to understand the different kinds of problems in clustering. These problems and their solutions fall mainly into four categories:

- **Exclusive Clustering:** In exclusive clustering, an item belongs exclusively to one cluster, not several. We could have simply associated a book like Harry Potter only with the cluster of fiction books. Thus, Harry Potter would have exclusively belonged to the fiction cluster.

- **Overlapping Clustering:** What if we wanted to do non-exclusive clustering; that is, put Harry Potter not only in fiction but also in a young adult cluster as well as under fantasy. So an item can belong to more than one cluster.

- **Hierarchical Clustering:** Now, assume a situation where we have two clusters of books, one for fantasy and the other for space travel. Harry Potter is in the cluster of fantasy books, but these two clusters, space travel and fantasy, could be visualized as sub-clusters of fiction. Hence, we can construct a fiction cluster by merging these and other similar clusters. The fiction and fantasy clusters have a parent-child relationship.

- **Probabilistic Clustering:** A probabilistic model is usually a characteristic shape or a type of distribution of a set of points in an n-dimensional and adjust the model's parameters to correctly fit the data. Such correct fits rarely happen; instead, these algorithms give a percentage match or a probability value, which indicates how well the model fits the cluster.

### 2.3.4 Related Work

There are lots of researches about cost effective enterprise cloud solutions with Hadoop, streaming engines and distributed systems. When we talk about cost effectiveness, this topic includes for owning cost (CAPEX), and operational costs (OPEX). The owning cost could be just a comparison for having a cluster, appliance etc., but the operational costs includes energy consumption, having a support personnel and being in a data center.

In enterprise data economy journal paper[9], W. Lou researched the data costs for Big-Data for the enterprise companies. Also there are lots of researches for enterprise readiness of the Hadoop like, F. Tongke researched[10] for a Hadoop platform for Telecom companies. Tongke focused on the data growth of the telecommunication companies and how companies could handle this data growth with Hadoop based

platforms. There are performance based researches, about optimizing workload with understanding the characteristics of the job[11] and get maximum throughput from the jobs. This research is done with a 2000-node Hadoop cluster at Taobao. One of the aims of the research is finding the majority of the jobs that is more important to process and changing the scheduler with the Fair4S scheduler instead of using fair scheduler (default scheduler of the Hadoop). According to their research they decreased the waiting time for small jobs by a factor of 7 compared with the default fair scheduler.

Streaming data is a strongly coupled feature with the Hadoop and Big-Data platforms, because most of the Big-Data is being generated by streaming platforms. For Complex event processing view, as we mentioned in section 2.3.1, there are two parameters for getting the rule and finding new patterns, the first one is support, and the other one is confidence. In late researches, there is one new parameter conceptually added called time confidence to decrease resultant rule count, to generate more clean results [12]. Time confidence is based on the appearance count in a time window and if the rule's time confidence is above the threshold value, the rule is accepted by the algorithm.

Autonomous stream processing and rule mining systems are really important for Telecom companies too, because finding rules and patterns could decrease faulty situations and increase the system reliability. For example with researching the alarms by their natures [13] (for GSM ecosystem), and utilizing the time constraints with the other patterns could increase the performance of the algorithm. Also doing some historical processing with vertical turning machine with a similarity function to compare, and validate the outputs of the turning machine is being used by some Telecom based alarming systems [14].

# CHAPTER III

# PC - LAPTOP BASED CLUSTERS

Hadoop and stream processing platforms are being used by lots of large enterprise companies and research organisations, such as banks, telecommunication companies, medical companies, finance and insurance companies, on-line shopping sites, genetic research groups, search engines etc. These applications have their own purpose and own specialized code running on their platforms. we are not studied on all of these fields individually, but we have touched some of these fields, those have Big-Data, and in this chapter, we will try to show what we have done so far.

We divided the Big-Data applications into two sections by the cluster installation types in terms of architecture. The first one was PC - Laptop based clusters and the other was the server based clusters.

Historically the companies and researchers developed the Hadoop clusters from their own resources, like laboratory PC's and retired laptops before making big investments. Therefore we followed this road and built our very first cluster from the University's computer lab PC's.

We have tried to use Hadoop in almost every architectural environments to see potential applications. Desktop PC's are seem to be powerless and useless for almost every Big-Data application, but the Hadoop's aim was combining relatively powerless servers (in this case PC's) to create a powerful system, because of this, we decided to find some applications for PC clusters.

## *3.1 Özyegin University Computer Lab Cluster*

The idea behind this is same as the Hadoop's distribution logic, and we wanted to see, if the scale-out strategy would create a powerful computational system from powerless

PC's. This was the early project of master period, and this study was good merit to compare the Hadoop installation types.

### 3.1.1 Set-up

The set-up has two parts. The first set-up, created with three (one node is in both master and slave roles) powerful IBM blade servers on H22 Blade chassis, and the second set-up developed with twenty (1 master + 19 slaves) laboratory PC's (See Table 2). Each side has one master and the other machines were configured as slave.

| | **Blade Servers** | **Laboratory PC's** |
|---|---|---|
| **Count** | 3 | 19+1 |
| **CPU** | Intel Xenon 2.4 GHz, 8 Cores | AMD Althron 1.7 GHz, 2 Cores |
| **Hyper Threading** | Yes | No |
| **Disk** | 74 GB, 15,000 rpm | 80 GB, 7,200 rpm |
| **Memory** | 24 GB | 4 GB |
| **RAID** | RAID1 | N/A |
| **Network** | 1 Gbps | 100 Mbps |
| **OS** | RedHat EL 5 | RedHat EL 5 |
| **Price (each)** | $ 6,000 | $ 500 |
| **Cluster Cost** | $ 18,000 (W/o chasis) | $ 10,000 |

**Table 2:** Blade Server - PC hardware comparison.

### 3.1.2 Test

We wanted to test the system for two merits. One was the utilization and the other one is time. In this purpose we have tried two examples in Hadoop's example set.

The first one was Teragen example, and the example simply generates random 4 Bytes unsigned integers and writes to the disk. In this case all of the nodes can generate integers and write them to the disk independently (Map-only job), so there are no noticeable network workload. We generated 10 GB of integers with both systems.

Luckily the first test's output is the input of the second test so we did not have

to produce new data for the second test. The second test was Terasort example, and it simply gets integer numbers as an input, and produces sorted output. Actually there is no useful code in this example, mappers takes the input and sends them directly to the reducers, and the reducers writes back the output of the mappers to the disk, because there is already a built-in sorting mechanism in shuffle mechanism. We expected to test three hardware parts of the computer with this example, these are CPU, network and disk. Sorting requires data exchange between nodes and, that will show us the importance of the network connection in distributed systems.

### 3.1.3 Results

| | **Blade Servers** | **Laboratory PC's** |
|---|---|---|
| **Average CPU Utilization** | ∼40% | ∼74% |
| **Average Disk Write Speed** | 62.6 MB/s | 22.3 MB/s |
| **Completion Time** | 53 secs. | 24 secs. |

**Table 3:** Teragen Test Results

In the first example, we though generating random numbers could be challenging for CPU's, so we expected to test CPU and disk, but we could not test CPU, because today's CPU's are much more faster than disk technology, therefore our test turned out to a I/O bound test. Every node has one local disk and there were nineteen disks produced about 423.7 MB/s writing throughput for PC cluster and, three disks produced about 187.8 MB/s writing throughput for blade servers, and there were no major network traffic because, all tasks can be done with the node' local resources independently.

| | **Blade Servers** | **Laboratory PC's** |
|---|---|---|
| **Average CPU Utilization** | ∼32% | ∼69% |
| **Average Disk Write Speed** | 58.2 MB/s | 8.1 MB/s |
| **Completion Time** | 57 secs. | 65 secs. |
| **Network Utilization** | ∼40% | ∼100% |

**Table 4:** Terasort Test Results

The second example concluded with blade's victory, because PC's have 100 MBps network cards, and Blades have 1 GBps internal network bus (not using regular network switch or router for intra chassis network traffic), and the terasort example produces a huge amount of network traffic.

### 3.1.4 Conclusion

As the results show, our PC cluster is much more faster in Map-only jabs, because they do not have powerful network cards, and when the reducers got involved to the process, the work progress is slowly increasing because of the network cards. For both tests, we expected to test CPU too but for both systems has faster CPU's than the other components. Also the importance of the count and the revolution speeds of the disks have shown in these tests. We have shown the bandwidth increase for disks even individual disk bandwidth is slow, the system's disk bandwidth increased with the disk count increase.

## *3.2 Customer Stock Portfolio Analysis for Banks*

Banks are doing so many computer aided calculations to operate their processes. These calculations can be regulatory, reporting, analysis and ETL etc. type of calculations or processes. For a bank the maximum profit calculations like customer stock portfolio analysis is very important because, customers can be put more money to banks to protect or increase their money.

The customer stock portfolio analysis has two parts and the input data is limited to the customer data, which includes the stocks, those the customer currently has it, and the other is the available stocks, those could be bought by the customer with banks advice. The analysis look at the maximum profit combinations of these stocks. For example, the customer has 3 stocks and there are 28 available stocks, the algorithm takes all the possible combinations of the stocks and calculates the possible profit, according to the possible combinations. This algorithm is a deterministic algorithm

because, it tries every possible combinations, and its running time can be determined with the number of the available stocks, and number of customers.

### 3.2.1 Algorithm

The algorithm is simple and makes CPU bound calculations. It takes customer info and stock list data separately. The customer data is a 100 KB text file, includes the customer code and stock codes, those owned by relevant customer, and the second file includes the available stocks list in a plain text format. Algorithm takes the customer data line-by-line, generates all the combinations with the available stocks and calculates the maximum profit option, and writes the maximum profit option back to the disk. We take the algorithm as a flowchart with lots of unknown parameters and variable names, those have to be solved by us, and the hardware (cluster) is the same blade servers, those used to compare the PC cluster test.

### 3.2.2 PC Version of the Usage

The bank's idea is running this algorithm in their employee PC's with Hadoop out of the working hours. There are 400 employee PC's and a powerful master could operate 400 PC's to run the customer portfolio analysis, and if there are more time slots, to run other algorithms. This idea was good, but we never got a chance to apply it.

Although the ides was great, but there would be some management problems, because there might be some employees (over-time workers) out of the working hours, or some PC's might be accidentally shut-downed by employees and these PC's could not be reached.

## *3.3 Queryable Long Term Storage Service for Sensor Data*

There are so many areas for sensor data collection, stream processing over this data and, storing to analyse the data for future processes or regulatory needs. Wearable sensor added clothes for health care systems, also sensor added sports clothing, smart

roads with sensors, smart electric meters and, for energy sector, measuring tank health with pressure and heat sensors etc. An Oil & Gas company have to monitor all the oil tanks, petrol production steps with sensors for 24 X 7, and there is no difference for the Turkey's biggest Oil & Gas company. They have to collect the sensor data and store for the reporting and analysis purposes. There are about 65,000 sensors for monitoring heat, temperature etc. and the sensors collect data for every 3 seconds. The data mostly composed by integer values with flags. The amount of data per day can be calculated as follows:

$$65,000\ sensors\ x\ 3\ seconds\ x\ 4\ Bytes\ =\ 780000 Bytes\ /\ 3\ Seconds$$

$$780000\ x\ 20\ data\ reads\ per\ minute\ x\ 60\ x\ 24\ =\ 20.92\ GB\ /\ Day$$

Even if there is no data enrichment like adding sensor number, date time etc., it is a huge amount of data for almost every system, but there is a data enrichment process, and this data amount groves exponentially. Therefore just storing the data is a huge process, and this storage phase is not the only part of the problem, because they have to query data for reporting and analysing.

The company did not want to invest money to a Hadoop system before they see the benefit of the system, so they proposed us to use old employee laptops, and they gave us 13 laptops to install Hadoop, Hive and impala. The laptop configurations were identical and like this:

| | **Laptop PC's** |
|---|---|
| **CPU** | 2 GHz Intel i5 4 cores |
| **Memory** | 4 GB |
| **Disk** | 500 GB 7200 rpm |
| **Network Card** | 1 Gbps |
| **OS** | CentOS 6.5 |

**Table 5:** Laptop PC configurations

These laptops are connected to each other with a separate Gigabit switch, and they are kept in the company data center (see figure 8) to control the temperature.

**Figure 8:** Laptop Cluster in Data Center.

There were two issues about this installation. The first problem is the slow laptop disks, and the second problem was the laptops were not designed to work 24 X 7.

### 3.3.1 Cluster

The cluster configured as 1 master node and 12 slave nodes, with about 5.5 TB available storage space, and 48 GB memory in total, and we installed the Cloudera version of Hadoop Distribution. This distribution adds some cluster management capabilities, such as, monitoring, changing configurations, adding and removing nodes or software packages from the web based interface.

### 3.3.2 Installation

We installed Cloudera with Hive and impala packages. Impala is a warehouse application with indexing capabilities for fast query performances, that runs on top of Hive warehousing package.

### 3.3.3 Application

There is no specialized application implemented for this service, but we had to do lots of configurations to get maximum performance from the system. The need was getting query results under 10 seconds. Data test data was produced by the 13,000 of 65,000 sensors for 1 year. The total data amount for the application was about 1.5 TB in a plain text format.

There are so many overheads for operating systems, because they are built for general purpose of computing. The main overhead is visualization of the system. Hadoop does not need desktop environment, and desktop environment occupies lots of memory spaces, so the first thing we have done, was removing the desktop environment. This released about 10% of the memory space. After this operation we removed all the unnessary applications and services, like mailing, printing etc. and the final move was changing some OS level flags like, THP feature of CentOS, this feature enables the large page management[6] for the memory. this feature could significantly improve for the memory indexed applications, that we were going to do with impala indexing.

### 3.3.4 Compression

We could get benefit from tuning the system by doing some configurations, but working with the Big-Data requires some data specific tuning for almost all individual systems to do real breakthrough. Therefore we have searched for some compression algorithms to improve the I/O performance and making a cost effective high volume storage system. There are some late experiments on compression algorithms on Hadoop environment[7], but back in the day there were none of them for Hadoop, so we have ran some tests to find best compression algorithm for this job. Our tests are focused on compression ratio, speed, and usability. The test data was the 10 GB of database style schemed text, with 10 columns and some repetitive rows in a single

file. I specially mentioned the test data was in a single file, because it effects the read/write performance, and the best case for the compression is reading from and writing to a single file. The test results are produced in four blade servers. The test results are shown in figure 6.

| | **GZip** | **LZO** | **Snappy** |
|---|---|---|---|
| **Ratio** | 1/8 | 1/6 | 1/8 |
| **Time** | 58 sec. | 120 sec. | 68 sec. |
| **Split-able** | No | Yes | Yes |

**Table 6:** Compression Algorithm Test Results

The test repeated for three times, and all the results above are the average values of these tree tests. For all three tests GZip compression algorithm was the winner. You might think the overhead of compressing the data could decrease the system performance, but actually it helps to improve the system performance, because it does less I/O from disk, and the disk is the slowest component of the computer, so CPU's can easily absorb the overhead of compression and decompression without performance loss.

### 3.3.5 Indexing

Indexing required by the Apache Impala, and this one time indexing job generates the indexes for doing fast lookups to make fast queries. To index all the data to the memory, the user has to run a Map-Reduce job, after this process, Impala stops using Map-Reduce framework (but still uses HDFS), and uses its own agents to handle distributed queries. Because of Map-Reduce has high initialization step, Map-Reduce could not response the queries in low-latency fashion.

### 3.3.6 Partitioning

Partitioning is done by Hive's own partitioning mechanism, an it is simply moves the data in a folder hierarchy. For example if the user wants to partition the data by the date and if the example date is 24-01-2014, Hive stores this data in

/user/hive/warehouse/<table-name>/2014/01/24 folder and the date part of the data is not required to store in the text file. This approach saves storage space and improves the query performance, because if the user's queries are mostly time based queries, Hive looks for only the relevant folders, it does not scan all the data. Although if the single partitioned file is too small (like 2 MB each), both compression ratio and I/O performances decreases, because the compression algorithms work best with large files, and small file sizes increases data seek time on disk, an this decreases the I/O performance.

### 3.3.7 Querying

Impala has three types of query interface. The first interface is the jdbc interface. This interface allows the users connect to the Impala through Java based applications. The second interface is command line interface, which is an application, that connects through jdbc connector of the Impala and prints back the query results to the screen. The last interface is the ODBC interface. This interface actually is an adaptor, connects the jdbc interface of the Impala and serves the ODBC clients and makes conversions from ODBC to jdbc, and allows users to make object based connections (mostly using in Windows OS based systems). This interface is used to connect to the Impala from Microsoft Office Excel software to visualize the data.

### 3.3.8 Results

The system itself is not so powerful, because of that we had to squeeze all the system resources, we have made date based partitioning, indexing with Impala, OS flag changes and removing unnecessary applications from the system, compression. The goal was responding date based count queries under 10 seconds, and the final results are shown in table 7. We could not achieve the goal but we were close, and we have learned lots of new things with this Hadoop PC set-up. the results is additive, this means we have tried first the compression, and we partitioned the compressed data,

and finally added indexes to the final compressed and partitioned data.

| | **Raw Data** | **Compressed** | **Partitioned** | **Indexed** |
|---|---|---|---|---|
| **Date Based Count Query** | 619 sec. | 364 sec. | 229 sec. | 33 sec. |

**Table 7:** Hive - Impala Test Results

### 3.3.9 Conclusion

Laptop cluster with lots of configurations and computational methods derived the Oil & Gas company lots of savings in terms of software license and hardware costs, because all of the software were open sourced projects, and using the old employee laptops saved from the hardware costs. This project showed us, the importance of the partitioning, indexing, compression and hardware specialized performance tuning. Removing the unnecessary applications, saved us lots of disk and memory spaces, with free CPU cycles from background jobs and services. For some specific conditions, even laptops could derive unexpected performance with doing system specific tuning.

## *3.4 Conclusion About PC Clusters*

Although the PC cluster competed with the powerful blade servers in the supercomputer test, PC's have too many problems, because PC's are not designed to run 24 X 7 and when you do, there will be some consequences, firstly the electricity bill goes up, because of the power consumption. Blade chassis consumes just 2000 Watts per hour, but total electricity consumption of twenty PC's is 7000 Watts per hour. Second issue is the PCs' components are not designed to run 24 X 7, and they will stop running after just few months. Third issue is the physical storage area, the standard blade chassis occupies 7 Rack Unit which means 0,13 $m^3$ by volume (Rack Unit dimensions: 28.26 X 4.45 X 92.71 cm) in a rack, but 20 PC's occupies 1,74 $m^3$ by volume and this means the PC's need 13 times bigger physical storage area. There is a need for special storage rooms like data center, specialized cooling systems, specialized power supplies with backups.

The PC clusters are mostly good for Map-Only jobs with lots of special configurations, because all the resources are limited and not designed for high density calculations, and PC's could not operate high speeded (10,000 - 15,000 rpm) disks. Also the network connection is very important, if the network is not separated or only low transmission speed cards are available, probably, these cards will be the bottleneck of the system. Because of this we are recommending only Map-Only type of usage with PC clusters.

# CHAPTER IV

# SERVER BASED CLUSTERS

This type of clusters are more commonly used by enterprise companies. The aim of Hadoop is using the low-end commodity servers to build clusters. There are lots of usage types and applications for server based clusters. In a technology based or related company, there is s huge amount of data, streaming and processing for operational needs. In this section we will try to explain some of these usage areas, problems and solutions.

## *4.1 ETL and Warehouse Solution for Telecommunication Companies*

Data processing is a daily routine for a Telecom company. In a Telecom company's infrastructure, there are lots of machine generated data, those are collecting and processing from wide variety of vendors, systems, and machines for analysis, reporting, investigation purposes and for governmental regulatory obligations. Also because of the variety of vendors there are lots of format wise incompatible systems. Therefore ETL jobs are indispensable parts for a Telecom company's operations. In this section we will explain the problems and their solutions for the Turkey's third biggest GSM company's distributed ETL, and warehouse system.

In a telecom company, there are lots of data streaming, and storing for processing. Almost all the data is generated by the machines. Even one SMS's trip could generate 20 different logs in different systems, base stations, and gateways etc., and when thinking all the customer's SMS's, only the SMS service derives huge amount of data, that needs to be processed in one day due to the regulations, and in-company processes. Also call initiation is a complicated process and generates twice logs as

SMS service. There are so many operations, like call and SMS, so there must be a system for just governmental regulatory ETL jobs.

Government wants the logs in daily basis and if the log stream is late or broken, they could fine the amount of the company's 3% of endorsement, so this is a critical mission. Against to common sense, there is a increase for WAP usage, and the WAP log transformation processing time was near to exceed the one day limit with one machine, and long term storage part solved by Oracle Exadata system, which could store just last months data, and there was a need for storing last six month's data. Hadoop completely fits the problem, because it has both processing and storage units.

The WAP specific log data is 160 GB / day (uncompressed data), and unlike the common sense it is slowly growing. Wap was a dying technology since the invention of the social networks with the Telecom operator's social billing packets, and its usage is slowly growing.

### 4.1.1 Governmental Regulations at a Glance

In Turkey all the Telecom companies have to give their customer's Internet access records, SMS, call, 3G etc. to the government in a certain text format. This regulation applies to all Telecom companies, Internet service providers even to wireless Internet available hotels, restaurants etc. As well as providers are giving their customer Internet access log, they also store these logs for 5 years. This law creates two problems. The first problem is the format transformation, and the second one is long term data storage.

### 4.1.2 Variety of Data Sources

There are so many Telecom vendors like, NSN, Huawei, Ericcsson etc., and their systems (for each system of all vendors) have different log format for all types of operations, and in some operations, more than one vendor involves in the operation. Government also has its own regulation log format, and all the vendor system's (see

Gateways in 9) logs needs to be transformed to the governmental log format. In our case the company did choose Ericcsson, ZTE and their in-house gateway deve lopment for WAP logs, and each vendor has its own format. Ericcsson wap gateways produce XML, ZTE gateways produce CSV and aLabs wap gateways have CSV like format with different separator and column sequence.

### 4.1.3 Reporting, Operational Investigation, and Long Term Storage

The Quality Assurance team of the company is generating some daily reports from the WAP log data, such as top 100 queries, service quality, and maximum response time etc. So they asked to us, to create a web based warehouse solution for WAP log data. We thing the warehouse part can be handled by Apache Hive, and we can develop a web based query interface for Hive. Our Hive solution fits for the problem because, Hive can work with compressed files, works with Hadoop and It has a SQL like query language (HQL). Therefore we used Hive for reporting but, we had to improve Hive's capabilities with query scheduling, and automatic recovery. Company wanted to make daily queries scheduled, and finished by the start of the work hours, so they can analyse and process the reports (see section 4.1.5).

When a customer calls the customer care service and complains about his/her bill, customer care employee hangs up the phone, and opens a operational investigation ticket to the operational teams. Operational teams get the ticket and start to investigate on relevant system(s). In WAP case, the most tickets are opening for unused, but billed WAP usage. For the first phase of the project because of the architectural problems, Hadoop system could not respond quickly, so the customer care service, had to open a ticket for investigation, and operational teams look for the usage of the customer (with our query mechanism), and if there is no usage, the company gives the WAP usage entry of the bill. In second phase Hadoop could respond the WAP usage requests while the customer still at the line. There are lots of cases like, court

orders to understand a criminal activity with looking to WAP usage reports.

The main goal of the Hadoop solution was creating a cost effective long term query-able storage system, so the storage part was very important for the company. Hadoop has a distributed storage system (see HDFS 2.1.3). The distribution handled by the Hadoop but, the first phase (see section 4.1.7) of the project has limited SAN type storage resource. Because of the resource limitations and the cost effective requirements, we researched compression algorithms (see compression 3.3.4) and the compressed Hive usage types. This research increased the storage capability from 12 days to 92 days. Although the second phase has 10 TB of storage space, according to our research compression increases the performance too, so the second phase includes compression too.

### 4.1.4 Enterprise-Ready Hadoop Based Architecture

There is a high level architecture of our WAP solution (see figure 9). Although Hadoop seems to solve all of the distribution and storage problems, back in day there were lots of problems with using an open sourced application in an enterprise company's infrastructure. Enterprise companies have lots of requirements, such as alarming, web interface, job scheduling, KPI's, statistical informations etc. to use a system in their operation.

### 4.1.5 Enterprise Operational Needs

Job and query scheduling is a must for an enterprise company. There are so many daily reports, analysis and queries. Also scheduling Hadoop jobs is a must, because the system have to be automated. Job and query scheduling is done by our implementations. The scheduler starts the Hadoop ETL jobs for every hour and, after ETL jobs finished, Hadoop moves the output data to Hive and a bucket to pile up to send the data to the government. Also sending the data to the government is a daily scheduled job. The query scheduler has a web interface, and the user can either

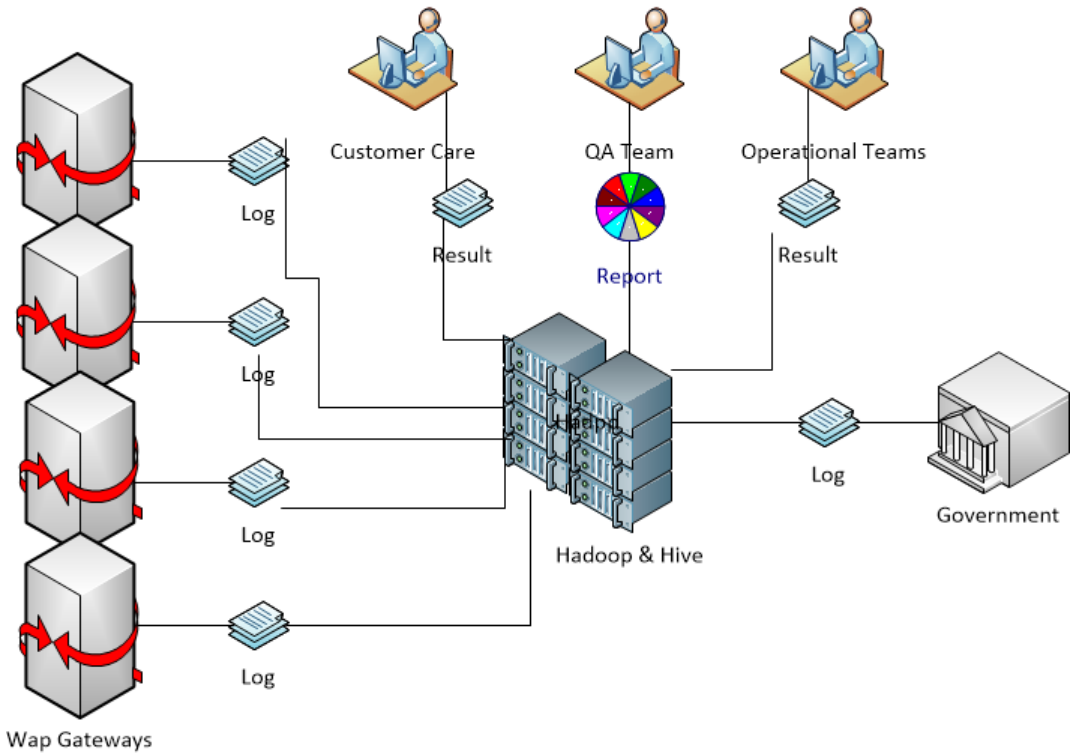


**Figure 9:** Wap Hadoop Architecture

schedule or run instant queries from the web interface. This part is very important to create an autonomous system.

This project has a governmental part, so for disaster situations informing the responsible people is very important. There are three alarming interfaces of the system. These are SMS interface, e-Mail interface and lastly HP OVO interface. HP OVO is being used by mostly enterprise companies to monitor applications, machines, systems with its monitor agents. In this installation HP OVO interface is used for alarming.

Measuring the performance is both hard and important part of any kind of operation. In our case, because of the most important part is sending data to the government, we decided to use sending time of data to the government as a metric. Every day the data has to be at the government's servers at a certain time, and this KPI have to be 100 % for every time.

Also keeping statistical information about the system, like system up-time, the amount of the input and output data etc., is important to understand, what amount of Big-Data the system is dealing with.

For an enterprise company, interfaces could be very important for ease-of-use and adapting (importing and exporting data) the system to other systems, so the system has web based graphical interface to, schedule a query or job, and monitor the system. Also the query mechanism can not be trigerred by only the web based interface, therefore jdbc, and ODBC connectors also helps to adapt the system with other systems such as Oracle Siebel.

### 4.1.6 Automatic Recovery, Backup and Restore

The system has a powerful recovery system for jobs, queries, and system failures, and this is not a Hadoop feature. If the job fails, it looks for the job status and if it is recoverable, continues from the last recovery point, if not system reschedules the job. If the query fails, system automatically reschedules the query and lastly for system failures the system has a unique mechanism. If the master node fails, it creates an alarm and informs the responsible people, after alarming completed, each slave node generates a random integer and throws to other nodes. The owner of the biggest number brings the system up as the new master and the system continues running.

Hadoop has not a backup and restore features for the HDFS, because it is a virtualization for the local file systems, so we decided to get regular file system backups for the HDFS. Also the restore part is done by the regular file system restore processes.

### 4.1.7 Phase 1 Solution

This phase designed to see the capabilities of the Hadoop framework, so the system created with servers and storage system, those are the company already have. The connections and the adaptors are the same with the second phase, so the high level picture (see figure 9) is valid for both phases, and phases are differentiated in terms

of machine count, cluster implementation, and storage types.

This phase has two separate clusters (see figure 10). the first cluster contains two powerful servers. This cluster is responsible from the ETL process, and keeping raw data for disaster scenarios. The second cluster has only one server and it is responsible for long term storage and querying. The cluster specifications are shown in table 8. This is not a ideal cluster type, because in Hadoop is a distributed system, and phase one does not have enough distribution, and it uses expensive SAN technology for storage.

This phase architecture could handle only 2.7 TB data (on Hive cluster), and it can run at maximum 70 % storage utilisation, so the system handle only 1.9 TB. The total system price is about $ 55,000 , and it can store just 1.9 TB, so this system is not ideal, not recommended and not wanted, but this was a great opportunity to develop our solution, and see Hadoop's behaviour in this kind of architecture.

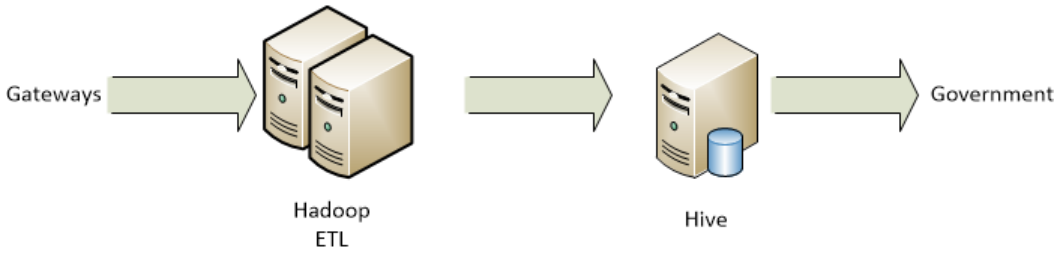


**Figure 10:** First Phase Architecture

| | **Hadoop Cluster** | **Hive Cluster** |
|---|---|---|
| **CPU** | Intel Xeon 24 cores | Intel Xeon 20 cores |
| **Memory** | 24 GB | 24 GB |
| **Network** | 1 Gbps | 1 Gbps |
| **Storage Type** | SAN | SAN |
| **Storage Capacity** | 500 GB | 2.7 TB |
| **Server Count** | 2 | 1 |
| **Estimated Price** | $ 20,000 | $ 15,000 |

**Table 8:** First Phase Server Specifications

### 4.1.8 Phase 2 Solution

The second phase architecture developed by us, with local storages and near commodity hardware (see table 9). This architecture created to be fit into distributed system's architectural logic. The problem of the first phase was, the speed difference with CPU and storage was too much, and the system could not utilize the CPU, and valuable system resources remained unused. Second phase's aim was to create cost and performance effective ETL and storage system. This phase has 5 servers, with 30 TB storage system with a cost of $ 30,000. This price is nearly the half of the first phase's hardware cost.

The architecture implemented to use ETL and storage services together. This simplified the system logic, and reduced to risk of loosing data, with multiple replicas. This system has 30 TB storage area, even we use 5 replicas, we can store almost 3 times bigger data in this environment. Although the replica count remained 3 because of the small cluster size, so we can store near 5 times more data. The query time, and cumulative system performance dramatically increased because of the increase of the I/O bandwidth.


**Figure 11:** Second Phase Architecture

| | **Hadoop & Hive Cluster** |
|---|---|
| **CPU** | Intel Xeon 6 cores |
| **Memory** | 24 GB |
| **Network** | 1 Gbps |
| **Storage Type** | Local 7,200 rpm |
| **Disk Count** | 6 |
| **Storage Capacity** | 6 TB |
| **Server Count** | 5 |
| **Estimated Price** | $ 6,000 |

**Table 9:** Second Phase Server Specifications

### 4.1.9 Tests and Results

The first phase solution was challenging because, Hadoop was a new software and we had to do lots of experimental test, configurations and new views for the performance increase and doing the job with limited resources.

As we mentioned before, there are lots of data sources and types for WAP log data. We started with the Ericcsson's XML log data. The data is compressed files with a length of 3-4 MB's per each file. As long as the Hadoop designed for big files, so we thought, we need to group the small files into bigger files than start the processing part[8]. We used 800 sample XML files for data load tests and there are 5 different loading and processing tests (see figure 12). The first test was the control group, every configuration, file types and formats were standard, with Hadoop's standard data loading interface. First test shows us the base results to see the improvements. The second test does not include the grouping procedure, this test is done to show the parallel loading difference. Instead of loading the data on just one computer, we decided to divide the data into the machine count and load all the data from each machine. Just doing the parallel loading task, saved us about 100 seconds for just 800 files. In second test there is no processing time improvement because the source data is not changed or grouped. All the remaining tests were executed to show the parallel loading and grouping affects for the data. The third test is the first data

grouping test, and there are 80 files, those are grouped ten-by-ten. The interesting thing about the fourth and the fifth tests, these two tests are completed almost in the same time, because we reached the I/O bandwidth limit after 100 groups.

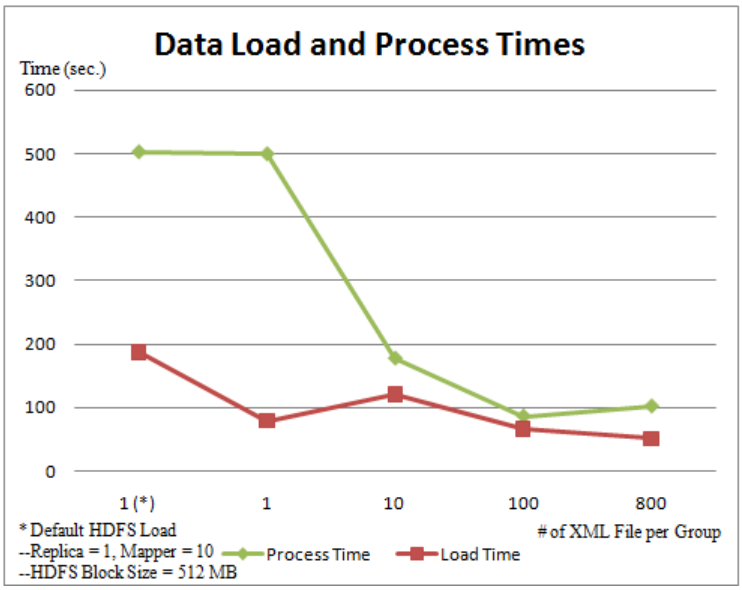


**Figure 12:** Parallel Data Loading

When a Hadoop performance benchmark released, they are just mentioning about the processing time, however for distributed systems, data importing is also very time consuming and problematic task, so we decided to release end-to-end benchmark (see figure 13). Test results showed us the importance of the input data format and size, because these tests are not related to a configuring Hadoop or making OS level optimizations. These tests are designed to use hardware resources efficiently. The test sequence starts with grouping process, and the second phase is the data loading process with again 800 sample WAP files, followed by processing phase, and the last part is getting data back to the local file system and sending to the governmental places.

The third test was the job pipelining. The idea behind the job pipelining is using all the system resources fully filled at once (CPU and I/O). For ETL jobs there are

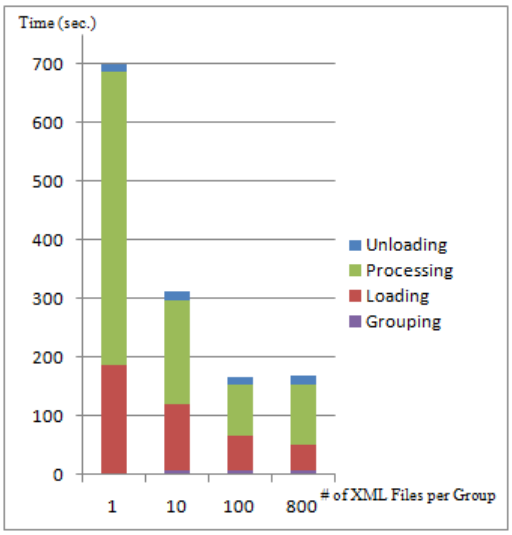


**Figure 13:** End-to-End WAP ETL Job Benchmark

three steps for the job completion. The data loading step, which is a I/O bound step, processing phase mostly CPU bound step, and getting the data back, which is also a I/O bound step, so we decided to run two independent ETL jobs by sliding the starting point of the step (see figure 14). The first job is a SMS log based ETL job and the second job is a WAP log based ETL job. The first test is running two jobs back-to-back (no overlap), and this is our control test for this test sequence. In scenario 2, the second job's loading step is overlapping with the first job's data unloading step, but this pipelinig could not make a significant difference, because overlapping steps are both I/O bound processes and resources are consuming by two different jobs. In third scenario there are two overlapping steps, the first job's processing task is overlapping with the second job's loading step. This test produced the best performance because, overlapping steps are mostly differentiates with their bounding types. The processing task is mostly a CPU bound task and the loading step is mostly I/O bound task, therefore the system resources could be used fully utilized and in most efficient way. The last test was about the full overlapping test.

The last part of this section is about the performance comparison of the previous

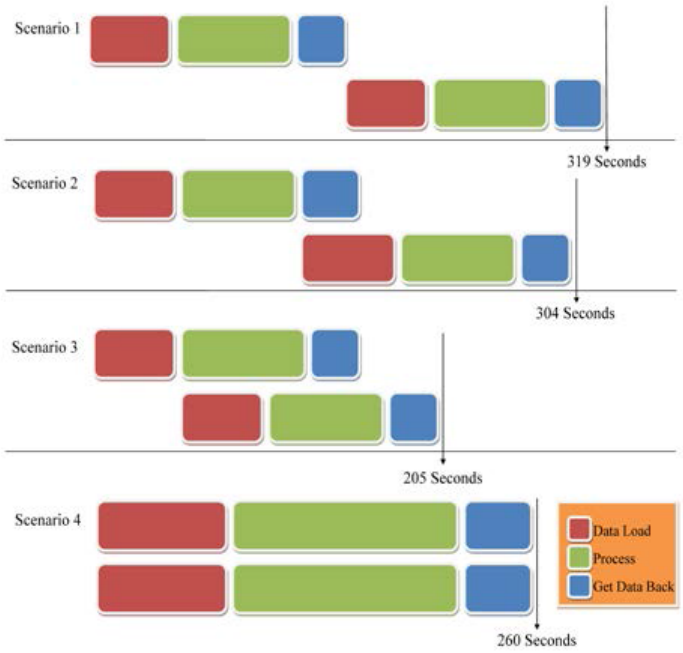


**Figure 14:** Job Pipelining Results

one machine solution of the company, first phase and the second phase for both ETL processing time and query times. The previous one machine solution does not have storage and query capabilities, so the storage and querying parts are covered by Oracle's Exadata warehousing appliance, so the for the ETL and query costs are separated in tables 10 and 11. Also the first phase handles the storage and the ETL parts separately so the prices separated in tables. Although the second phase solution includes both storage and ETL parts, so $ 30,000 is total price for the second phase solution.

| | **Non-Clustered** | **First Phase** [2] | **First Phase** | **Second Phase** |
|---|---|---|---|---|
| **ETL Time** | 24 Hours | 8 hours | 45 mins. | 15 mins. |
| **Cost** | $20,000 | $ 40,000 | $ 40,000 | $ 30,000 |

**Table 10:** ETL Performance Comparison

Table 10 shows the performance comparison for the ETL jobs for processing one

[2]Without doing any configuration

day WAP log data (about 160 GB data). The non-clustered solution is the previous one machine configuration is reached to its own limits for I/O and CPU, the only way to improve this solution is replacing the machine with a more powerful server. Because of this job is a daily job, this solution is not reasonable for the company. The first phase solution with optimizations and configurations, like OS configurations, pipelining and grouping for small file problem improved the system performance by a factor of about 10. Finally the second phase solution improved performance dramatically. Because of the distribution of disks and multiple disks for each machine is increased the data distribution and I/O bandwidth. The first phase's cost is $ 40,000 for ETL performance test because there are two separate clusters and the table includes the ETL cluster's cost.

| | **Oracle Exadata** | **First Phase** | **Second Phase** |
|---|---|---|---|
| **Query Time** | 12 mins. | 30 mins. | 58 secs. |
| **Storage Capacity (Month)** | 1 (1 replica) | 3 (1 replica) | 15 (3 replica) |
| **Cost** | $625,000[3] | $ 15,000 | $ 30,000 |

**Table 11:** Query Performance Comparison

Let we evaluate the second phase results. What does it mean 58 seconds for this kind of data. Daily unprocessed WAP data is 160 GB, and the daily processed data amount is about 100 GB. All of the test queries requires full data scan for a month's data, and the calculations are as follows:

$$100\ GB\ X\ 30\ days\ =\ 3,000\ GB\ processed\ data$$

$$3,000\ GB\ per\ month\ /\ 58\ sec\ =\ 51.72\ GB/sec$$

$$51.72\ GB/sec\ /\ 4\ machines\ =\ 12.93\ GB/sec/machine$$

$$12.93\ GB/sec/machine\ /\ 6\ disks\ =\ \mathbf{2.15\ GB/sec/disk}$$

The average read speed of a 7,200 rpm disk is 25-30 MB/sec not 2.15 GB/sec. This result is far beyond the throughput of the disk. This is the result of the partitioning,

indexing, compression and other fine-tunings for the OS and Hadoop. Without these optimizations, let's find how many servers we need to get this performance.

$$10\ X\ GB\ processed\ data\ X\ 30\ days\ =\ 3,000\ GB/month\ =\ 3,072,000\ MB/month$$

$$30\ MB/sec\ read\ speed\ X\ 6\ disks\ =\ 180\ MB/sec/machine$$

$$3,072,000\ MB/month\ /\ 180\ MB/sec/machine\ /58\ secs. =\ \textbf{294 machines}$$

This results shows us the importance of the optimization and the data oriented processing.

### 4.1.10 Conclusion

All these test results and experiments show us, hardware and software configuration, architecture and the data processing approach could make difference for performance and query time. The cost of the previous one machine solution with Oracle Exadata is about $ 645,000, but the ETL performance for a daily processing requires almost more than a day, and getting query results from just one month's query results in 12 minutes requires, a huge appliance with a huge price, and yearly renewal maintenance cost. Although the fist phase pseudo-distributed architecture is not the answer for desired query times, and distributed systems point of view. Because the first phase includes powerful servers with costly SAN storage systems. The second phase is the answer for all points of views. Second phase accomplished distributed architecture with 5 identical mid-range serves with 6 independent local hard drives.

Making an enterprise ready Hadoop cluster and giving a 24 X 7 service is a hard job to accomplish. There are lots of needs, and developments to integrate Hadoop into a Telecom company's operational environment. All these tests and experiments show us, distributed systems has lots of potential to create powerful system with less

[3]Oracle Exadata price brought from http://www.oracle.com/us/corporate/pricing/exadata-pricelist-070598.pdf. Storage capacity could change with configuration difference and/or hardware difference.

hardware, and the importance of the data oriented processing view, because data specific optimizations like, grouping small files into bigger files, makes real difference.

## *4.2 Alarm Correlation on Streaming Data*

Hadoop is suitable for Big-Data and batch processing, but today's market requires to be fast, so batch processing is not fast enough, so in most cases streaming engines have to be involved to the system.

The ability of processing the streaming data is a must for telecommunication, marketing, finance, banking and oil & gas companies. In section 3.3, we tried to explain to process and query the sensor data in an example. Processing or storing Big-Data is one thing, but combining the power of the stream processing and distributed systems is called as a complete system for today.

In our case the Telecom company wants to know all the problematic situations in their infrastructures including the base towers, and be prepared for these situations before the event is done. The difficulty about this project is, there are lots of data resources, and if an important data source brakes down, their business could effect in so many levels. For alarm correlation on streaming data, this kind of system is a must. The current system is based on human experiences, and when a new problem or alarm raises, they have to look all the logs and, find the problem cause. After finding cause, they have to be sure about the problem caused from the finding pattern and they have to define a rule to their systems.

### 4.2.1 Finding Alarms in an Autonomous Way

There are so many approaches about finding new rules and patterns in data mining field. Our approach was finding rules with a modified rule growth algorithm [15]. The algorithm need two parameters to run (see section 2.3.2). Our approach was adding a new parameter called time confidence and the aim of this parameter is reducing the rule count, those are produced by the rule growth, to get more dependable rules.

Time confidence looks the pattern and searches for the occurrence of this pattern in a time window, and if the pattern occurs in the exact time difference, the algorithm counts this occurrence as a successful catch. After doing all passes for all patterns, the patterns, those have values above the threshold value, the pattern accepted by the algorithm as a new rule.

The streaming architecture (see figure 15) is the combination of the streaming engine with the Hadoop. The streaming data is firstly processed by the streaming engine (CEP Engine) with registered rules if the CEP engine catches an event creates an alarm and informs the company employee. After the stream processing phase is ended, the data goes to the Hadoop system for storage and, further rule mining analysis. A distributed data mining engine called Mahout runs on top of the Hadoop and mines for new rules. If mahout finds a new rule, informs the company employee and if the rule found is a valid rule, the employee registers the rule to the streaming engine and the engine looks for the new rule too. Hadoop enables the long term storage, and distributed processing power for the data mining purposes. The main purpose of this project is finding the root causes of the problems and inform before the system fails or something problematic is done.

### 4.2.2 Rule Mining Strategy

Our rule mining strategy is based on rule growth algorithm with time confidence parameter included [12]. This approach is not used by the industry, but the number of the rules, those are found by the original rule growth algoritm, is very high, and finding the useful rules from the pile of not validated rules is like finding a needle in a hail stack. The time confidence parameter reduces the number of the resultant rules, this simplifies the system operator's job.

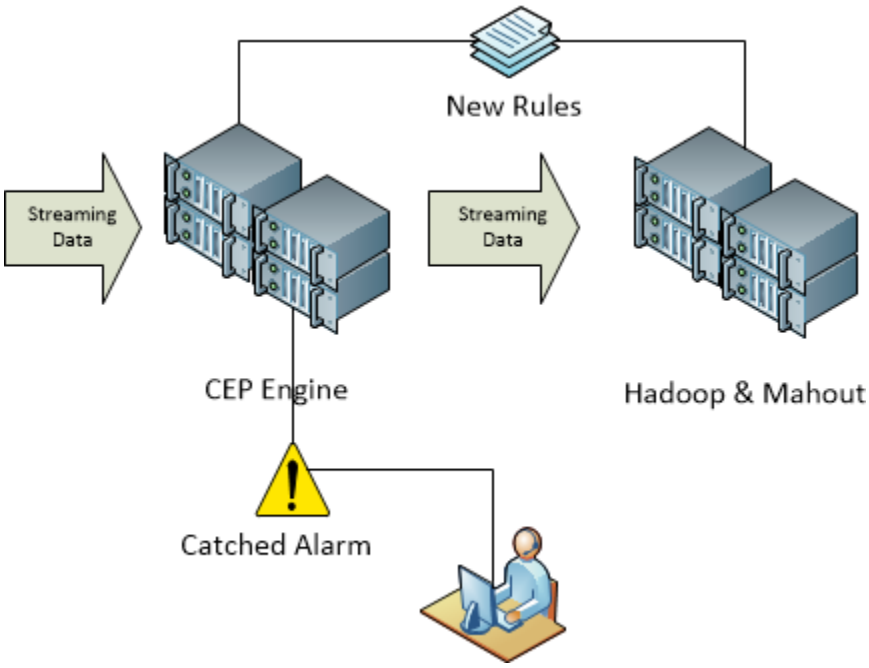


**Figure 15:** Streaming Engine Combined with Distributed Systems

## *4.3 Search Engine Project*

This Hadoop deployment is the one of the biggest Hadoop cluster in Turkey. There were 138 nodes for just one Hadoop cluster. The architecture is shown in figure 16. There are three parts of this project. The first part is fetching the Internet site data to the data center. This task requires lots of skills including the understanding of which part of a web site is useful and which is not. Obviously we cannot crawl all the web inside a data center, so we need to know the importance of a web page autonomously and select information and send it to Hadoop. The second phase is about creating indexes and storing the original data. This part is done by Hadoop cluster with Hadoop's HBase columnar database extension. The importance of the Hadoop is, creating searchable indexes for the usage of the search engine. There is a batch processing indexing job to create ready-to-import index files for Apache Solr Cloud. Finishing the indexing part brings us to the last part of the project, which is indexing and searching engine. This part is where the user interacts with the system.

Apache Solr Cloud is simply a indexing and searching engine based on Apache Lucine indexing engine. Solr enables, making searches over the lucine indexes.

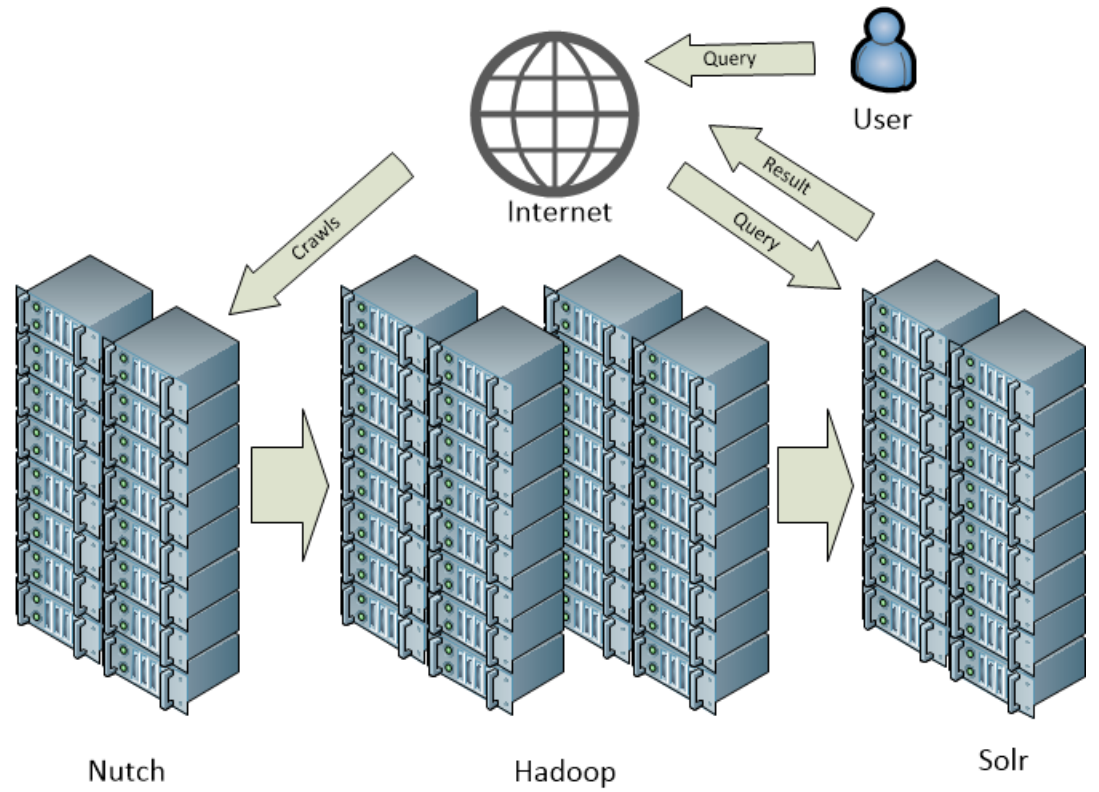


**Figure 16:** Search Engine Architecture

The high level architecture is very simple, but the node count makes the system unmanageable. There are lots of systems and applications to manage this kind of clusters, like salt project for sending commands to multiple nodes at once to simplify the management process. Also monitoring, and alarming is important for knowing the utilization of the cluster. Therefore Nagios got involved in the project for monitoring the clusters.

## *4.4 Discussions*

When I explain my thesis, I have mentioned lots of optimizations to improve system performance. There is a common way, that I created to make decisions about

optimizations. I will try to explain all of them.

### 4.4.1 Indexing

This is a query oriented optimization, and not suitable for general purpose systems. If the system administrator indexes all of the fields, at some point the memory will be full and the system becomes unavailable. If the queries are mostly focused on some specific fields, such as date, time, user ID etc. indexing becomes reasonable. There is a basic indexing flowchart for indexing in figure 17. If the example data is like table

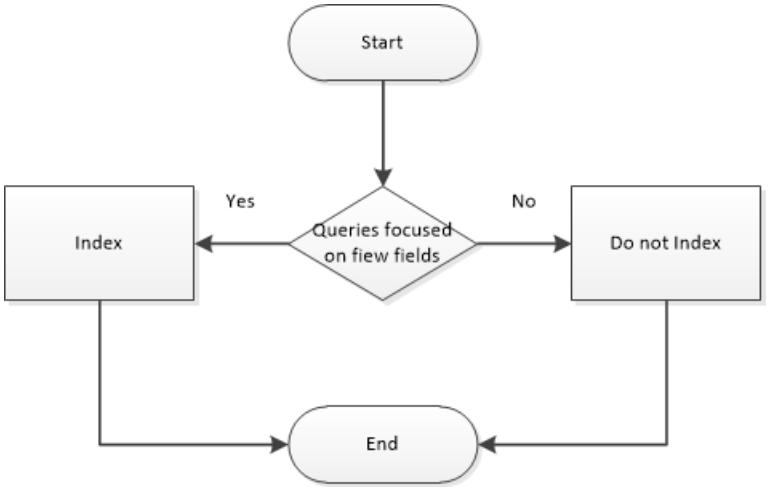


**Figure 17:** Indexing Flowchart

12 and if the system queries' where clauses are mostly focused on one field like:

| Sensor ID | Value | Date | Time |
|---|---|---|---|
| 18856 | 22.5 | 2014.03.24 | 18:56:23 |
| 18953 | 25.5 | 2014.03.24 | 18:59:15 |

**Table 12:** Example Sensor Data

*Select count*$(*)$, $'$*sensor_id*$'$ *from tbl_sensors where* **date = '2014.03.24'** *order by* $'$*sensor_id*$'$

indexing date field becomes reasonable, and the system gets performance increase.

### 4.4.2 Partitioning

For partitioning, decision making is more complicated than indexing, because there are two views to partition a table for Big-Data platforms. The first view is data view. As I explained in section 4.1.9, small files are problem for Big-Data platforms, especially for Hadoop, so if the data amount of one partition is too small, partitioning could decrease the overall system performance. Also the count of the partitions is very important because each partition occupies space in meta data server of Hive and the count must be lower than 2048. The second view is the query view. In this approach, we must react exactly same as the indexing (see section 4.4.1). Partitioning flowchart shown in figure 18.

### 4.4.3 Compression

Compression increases system performance for almost all text based data. Specially text based data could be compressed more efficiently, because for enterprise companies almost all of the Big-Data sources produce same type of data and data duplication is very high. Therefore in most text based data cases compression could both increase system performance, and reduce the storage space needs at once. If single compressed file size is smaller than the block size of the Hadoop, the system does not requires split-able compression algorithms. We recommend to use non split-able compression algorithms because, these algorithms are faster, and have higher compression rates than split-able algorithms.

For binary data types, it may not so beneficial because almost all the binary data formats, also include the compression. If the project needs to process uncompressed binary data types like *.wav audio files, split-able compression algorithms must be used to get benefit from distribution. the compression flowchart shown in figure 19.

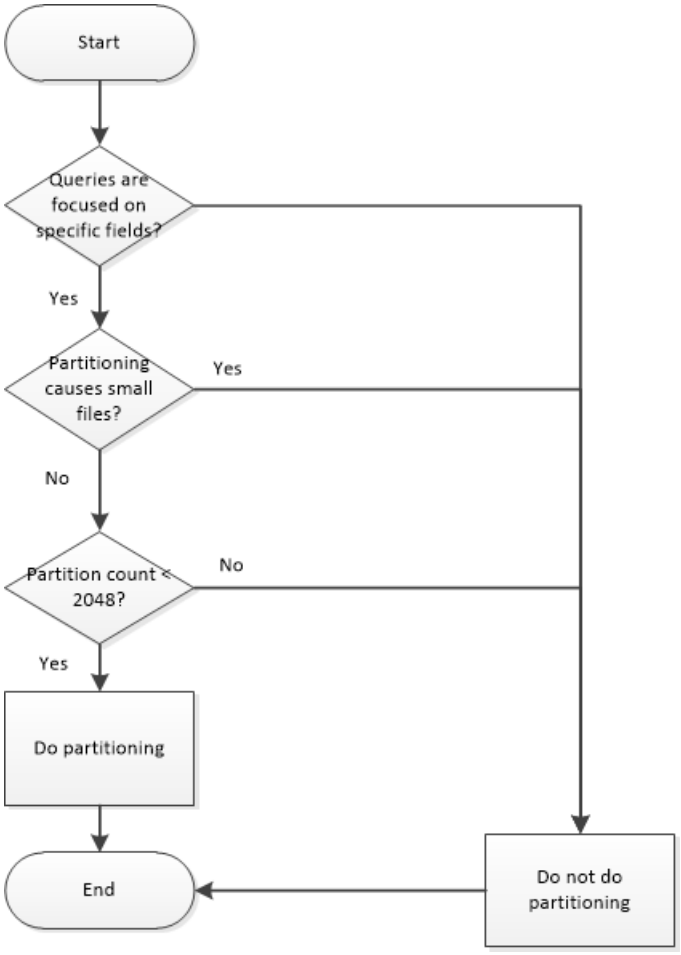


**Figure 18:** Partitioning Flowchart

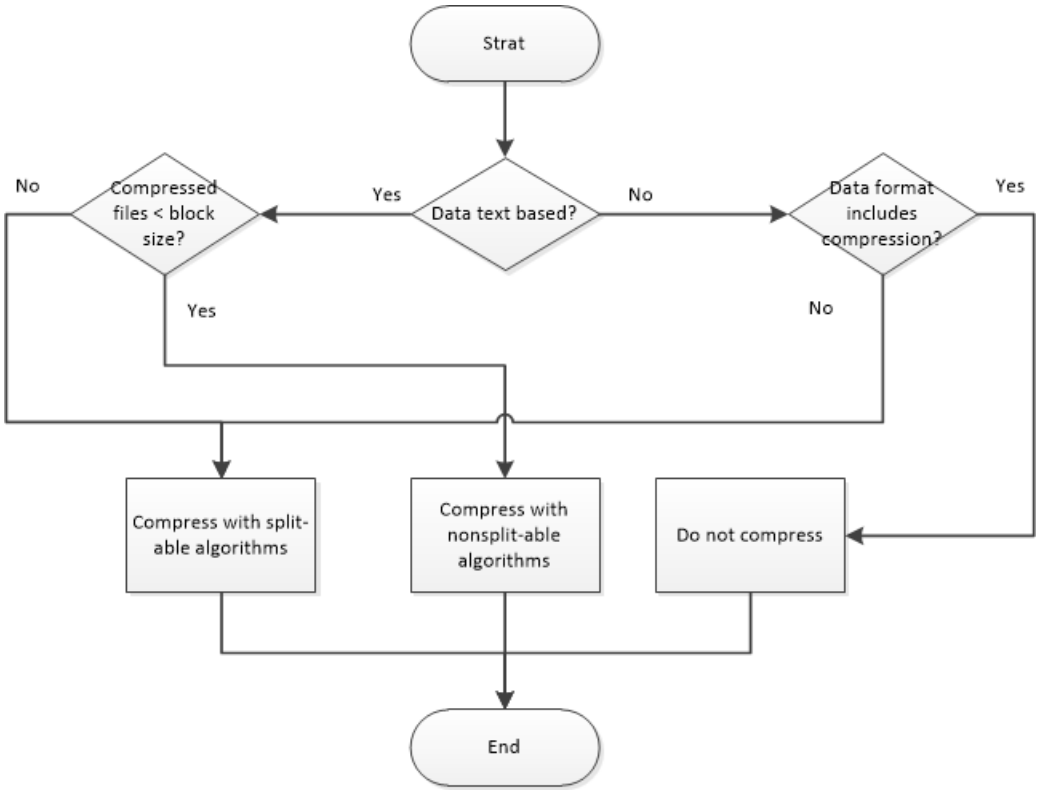


**Figure 19:** Compression Flowchart

### 4.4.4 OS Level Parameters

The OS level optimizations need for Linux expertise, so I will explain only some of common optimizations. An Operational system requires a non stop running environment, and Hadoop opens lots of files at once for processing, network communication and storage, and if the opened file descriptor count exceeds the maximum file descriptor count, Hadoop crashes. For Linux, the default file descriptor count is 1024, the command to increase the file descriptor count for system-wide is:

```
$ sysctl –w fs.file–max=100000
```

For RedHat based Linux distributions there is a security software called SELinux, to control file and directory access. This software is slows down the file access times, so closing this software is also increases the system performance. To close SELinux, open the file in path */etc/selinux/config* and change the line **SELINUX=enforcing**, to **SELINUX=permissive**, and after saving the file, restart the server.

A final hint for choosing the operating system, use RedHat for master nodes with RAID1 configuration for operating system disks, and use CentOS for slave nodes without any RAID configurations.

# CHAPTER V

# CONCLUSION

In this thesis, we report experiences, from real-life projects. Almost all the projects were owned by, Turkey's top Telecom, finance and Oil & Gas companies. In conclusion we have tried lots of techniques, and methodologies: PC clusters, server clusters, indexing, partitioning, compression, and OS level optimizations.

PC clusters are good at map-only Hadoop jobs. PC's are not designed for running 24X7, and they are not reliable for mission critical jobs. for instance recall the bank job (see section 3.2), the job was not mission critical and the customer portfolio analysis done in the employee computer while out of the business hours.

For Hadoop, the optimizations and the fine-tuning processes feels like I am sitting in a aircraft's cockpit, because there are lots of buttons, and parameters. Without knowing how the buttons and parameters work, the aircraft is still flying but slowly. The main purpose of this thesis is to prove the benefits and cost effective solutions could derive results, that almost no appliances could derive this kind of performance without special data driven optimizations.

Hadoop and its ecosystem can made a real difference in terms of performance. We proved the price is not always the best indicator for the performance for Hadoop. The architecture is very important for distributed systems. In section 4.1 the first phase's total price was $ 55,000 but the second phase's query performance was 32 times better, besides its price almost the half of the first phase's. Also the second phase's the ETL performance was three times faster than the first phase.

# Bibliography


[1] P. C. Zikopoulos, and C. Eaton, and D. deRoos, and T. Deutsch, and G. Lapis. Understanding Big Data 3–12, 2012. ISBN 978-0-07-179053-6

[2] D. P. Anderson, and J. Cobb, and E. Korpela, and M. Lebofsky, and D. Werthimer SETI@home An Experiment in Public-Resource Computing *COMMUNICATIONS OF THE ACM*, 45(11):56–61, November 2002.

[3] J. J. Dongarra, R. Hempel, A. J. G. Hey, and D. W. Walker. A proposal for a user-level, message passing interface in a distributed memory environment. *Technical Report TM-12231 Oak Ridge National Laboratory*, February 1993.

[4] S. Ghemawat, H. Gobioff, S. Leung. The Google File System. *SOSP03, October 1922, 2003, Bolton Landing, New York, USA*, 2003.

[5] J. Dean, and S. Ghemawat. MapReduce:Simplified Data Processing on Large Clusters. *OSDI 04: 6th Symposium on Operating Systems Design and Implementation, Google*, June 2004.

[6] D. A. Heger. Hadoop Performance Tuning - A Pragmatic & Iterative Approach. *DHTechnologies White paper*, 2013.

[7] S. Lovalekar. A Survey on Compression Algorithms in Hadoop. *IJRITCC*, 2(3): 479–482, March 2014.

[8] M. Koca, İ. Arı, U. Koçak, O. Çalıkuş, and C. Sezgin. Parallel and Pipelined Processing of Large-Scale Mobile Communication Data Using Hadoop Open-Source Framework. *Signal Processing and Communications Applications Conference (SIU), 2012 20th* April 2012.

[9] W. Lou. Enterprise data economy: A hadoop-driven model and strategy. *Big Data, 2013 IEEE International Conference, 2013* 65–70, October 2013.

[10] F. Tongke. Hadoop-Based Data Analysis System Architecture for Telecom Enterprises. *Computational and Information Sciences (ICCIS), 2013* 1277–1279, June 2013.

[11] Z. Ren, J. Wan, W Shi, X. Xu, M. Zhou. Workload Analysis, Implications, and Optimization on a Production Hadoop Cluster: A Case Study on Taobao. *Services Computing, IEEE Transactions, 2014*, 7(2):307–321, April-June 2014.

[12] O. F. Celebi, E. Zeydan, I. Ari, O. Ileri and S. Ergut. Alarm Sequence Rule Mining Extended With A Time Confidence Parameter. *Industrial Conference on Data Mining 2014(ICDM'14)* July 2014.

[13] P. Wu, W. Peng and M. Chen. Mining Sequential Alarm Patterns in a Telecommunication Database. *Proceeding of the International Workshop on Databases in Telecommunications 2001(VLDB'01)* September 2001.

[14] Y. Chen, J. Lee. Autonomous Mining for Alarm Correlation Patterns based on Time-shift Similarity Clustering in Manufacturing Systems. *Prognostics and Health Management (PHM), 2011 IEEE* June 2011.

[15] P. Fournier-Viger, R. Nkambou, V. Tseng. RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth. *Symposium On Applied Computing (SAC), 2011 ACM* March 2011.