KADIR HAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

BIG DATA PLATFORM DEVELOPMENT WITH A TELECOM DSL

CÜNEYT ŞENBALCI

May, 2013

BIG DATA PLATFORM DEVELOPMENT WITH A TELECOM DSL

CÜNEYT ŞENBALCI

B.S., Computer Engineering, Kadir Has University, 20011

M.S., Computer Engineering, Kadir Has University, 2013

Submitted to the Graduate School of Science and Engineering

In partial fulfillment of the requirements for the degree of

Master of Science

In

Computer Engineering

KADIR HAS UNIVERSITY

May, 2013

KADIR HAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

BIG DATA PLATFORM DEVELOPMENT WITH A TELECOM DSL

CÜNEYT ŞENBALCI

APPROVED BY:

Asst. Prof. Zeki BOZKUŞ (Thesis Supervisor)	Kadir Has University	_____
Asst. Prof. Taner ARSAN	Kadir Has University	_____
Prof. Dr. Selim AKYOKUŞ	Doğuş University	_____

APPROVAL DATE: 14/05/2013

BIG DATA PLATFORM DEVELOPMENT WITH A TELECOM DSL

Abstract

The amount of data in our world has shown exponential growth in recent years. This creates a very large collection of data sets –so called big data- in many organizations. Enterprises want to process their own big data to generate values from data to improve productivity, innovation and customer relationship better than their competitors. However, big data is so large and complex that it becomes difficult to process using traditional database management techniques. In this paper, we present a system which can be used to analyses for big data of telecom industries. Our system consists of three parts: Domain Specific Language (DSL) for telecom industries, parallel programming platform by using map reduce programming model and a viewer to present the results for human analysis. We integrated these three components by using a Distributed File Descriptor (DFD) to pass file information among each other. Our DSL offer many statements which are essential for telecom industries such as telephone call records, network logs and web link analysis. The platform component can perform highly parallel computations asked by DSL by using many different clusters of computers in data center. Our viewer component uses web browser to present result with many different graphics styles. Our solution for big data provides a comprehensive solution: Our DSL is much higher level than SQL. We do not ask programmer to write low level traditional code with Java or C by using Map Reduce techniques. We provide our own viewer.

Keywords: Big Data, Domain Specific Language, Distributed File Descriptor, Parallel Programming, Map Reduce.

TELEKOM DSL İLE BÜYÜK VERİ PLATFORMU GELİŞTİRME

Özet

Son yıllarda dünyamızdaki veri miktarı katlanarak artmaktadır. Bu durum şirketler içerisinde büyük veri olarak adlandırılan yapıların ortaya çıkmasına neden olmaktadır. Günümüz şirketleri rakiplerinin önüne geçebilmek adına gerekli olan verimlilik, yenilik ve müşteri ilişkileri gibi analiz sonuçlarını kendi bünyelerinde bulunan verileri işleyerek elde etmek isterler. Ancak büyük veri gerçek anlamda çok büyük ve karmaşık olduğundan ötürü geneleksel veri yönetim sistemleri ile işlenmesi imkansız denecek kadar zordur. Bu çalışmada size telekom firmaları için geliştirilmiş olan büyük veri sistemini sunacağız. Sistemimiz üç ana bölümden oluşmaktadır: DSL adı verilen Telekom alanına özgü bir dil, Map Reduce programlama modeli içeren paralel programlama platform ve sonuçların kullanıcıya sunulduğu bir arayüz. Bu üç ana bölüm birbirleri ile dağıtık dosya tanımlayıcısı olarak adlandırdığımız -DFD- framework'ü kullanarak haberleşmektedir. Önermiş olduğumuz DSL çözümümüz telekom firmalarına özgü telefon kayıtları, ağ kayıtları, link analizleri gibi verilerin paralel olarak işlenmesine olanak sağlar. Ayrıca veri merkezinde dağıtık yapıda bulanan cihazlar üzerinde işlemlerin paralel olarak çözümlenmesini sağlar. Web tabanlı sonuç gösterim ara yüzü ile işlenen verilerin efektif olarak gösterilmesi amaçlanmıştır. Tanımlamış olduğumuz DSL dili, SQL dilinden oldukça basit bir dildir. Kullanıcının dosyalar üzerinde herhangi bir paralel işlem yaptırması için Map Reduce tekniklerini içeren C, Java gibi kodları yazmasına gerek olmamaktadır. Aynı dil ile sonuç gösterimini kullanmak mümkündür.

Anahtar Kelimeler: Büyük Veri, DSL, DFD, Paralel Programlama, Map Reduce.

Acknowledgements

I would like to offer my special thanks to my supervisor Asst. Prof. Zeki Bozkuş who always helped me during the thesis period. Thank you for your useful advices and encouragement.

In addition, I would like to thank Bahtiyar Karanlık and Halit Olalı from Elkotek for their assistance. They provided Petaminer platform for our usage.

I would like to express my very great appreciation to Mr. Mustafa Yelmer for helping me to get through the difficult times. Thank you for his valuable and constructive suggestions during the planning and designing of this research work.

Last, but not least, I would like to thank my family for their unconditional support and encouragement. I am also grateful to my best friends Aykut and Mustafa for their useful advices and unconditional support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DSL	Domain Specific Language
DFD	Distributed File Descriptor
RDBMS	Relational Database Management Systems
DWH	Data Warehouse
GFS	Google File System
HDFS	Hadoop File System
MR	Map Reduce
SQL	Structured Query Language
NoSQL	Not only SQL
JSON	Java Script Object Notation
EC2	Elastic Cloud Computing
S3	Simple Storage Server
IT	Information Technologies
Mngt	Management
CSV	Comma Separated Value

Chapter 1

Introduction

In this project, I will describe a system that offer a special big data analysis platform for telecom industries. This platform has three main parts that suggests a new kind of domain specific system for processing and visualization of large data files for telecom organizations. These are Domain Specific Language (DSL), parallel processing/analyzing platform for big data and an integrated result viewer. Also, to pass information between these modules and organize communication, Distributed File Descriptor (DFD) is designed. By using this structure, steps of data processing/analyzing and visualization of results can be performed easily without writing complex queries or C, Java codes. This also prevents consuming too much time to perform some basic operations.

To find out benefits of this domain specific solution, we have to examine standard framework of big data concept carefully. At the beginning, I will describe Big Data subject and workflow of processing and analyzing phases.

Global data usage has been increasing exponentially since last ten years [1]. If we look at the type of these data, we can show that huge amount of them are generated in our daily life. For example; we share 50 million tweets per day, 700 billion minutes are spent on Facebook each month, 2.9 million number of emails are sent every day, 75 million hours video are uploaded YouTube every minute, 1.3 exabytes data are sent and received by mobile users, 24 petabyte data are processed by Google each day and 72.9 items are ordered on Amazon per second. With this example, we can obtain that data sources of enterprises are changing and increasing. Customer feedback, call detail records, billing, social media analyzing, emails, web server logs, databases are some of the most important information for enterprises.

Collection and correlation of different kind of data is not an easy operation. Traditional storage and processing systems cannot be used to handle these large datasets [2]. All of these large datasets are called as Big Data. This concept has special infrastructure and tools to perform for data storing, processing, analyzing operations. This infrastructure can be grouped 4 different part: Infrastructure, Programming Models, High Performance Schema Free Databases, Processing & Analyzing.

Infrastructure creates the base of Big Data concept. Distributed parallel processing and storing tools are placed on this structure which is generally known as distributed servers or cloud. These are easy to manage virtual systems that are served as IaaS by big companies such as Google, Amazon, and Microsoft. On the other hand, storage is another issue to store all datasets on distributed platform. Especially, Amazon-S3 and one of the most popular open source tool Hadoop Distributed File System.

Traditional programming models cannot be designed for large datasets. Therefore, Google is developed a new kind of programming framework that is called Map Reduce [3]. This programming model helps problems to divide multiple tasks and solve them in parallel way. Most and important part of Big Data tools (analyzing and processing) use this algorithm to increase efficiency of solving complex tasks like Hadoop [4], Hive [5], Pig [6], Cascading, Cascalog, Sawzall, Dremel and so on.

If companies want to manage large datasets, relational database management systems are not enough to achieve this target. Therefore, High Performance Schema Free Databases –generally known as NoSQL databases- are designed to perform data operations efficiently and rapidly. BigTable, Hbase, Cassandra are some of the most popular NoSQL database systems.

When companies deal with big data, it is very hard to obtain valuable information in it. So it is very important to process and analyze of big data on the distributed parallel systems. There are some special kinds of tools to overcome this issue like Hive, Pig, Mahout [7], R and so on.

Nowadays lots of organizations try to adopt big data systems. However, there are still some disadvantages in Big Data concept for midsize companies. First of all, integration of these technologies requires great software engineering skills. Secondly, it is not business friendly because customers need to write Java, Pig, Hive scripts. Last and the most important disadvantage is management of parallel tasks with right big data tools.

Although there are lots of advantages of Big Data concept, it is still very difficult to manage these systems for many enterprises. Therefore, this study suggest a new higher level language –called as DSL- which helps enterprises to process big data without writing any complex low level traditional parallel processing codes, a new kind of result viewer and this paper also presents a Big Data solution system that is called Petaminer.

1.1. Thesis Structure

Thesis has two main parts: Theoretical information about Big Data concept and a solution to simplify Domain Specific data operations.

First part summarizes evolution of Big Data concept:

- Big Data Characteristics
- Big Data Infrastructure
- Types of Big Data Tools

Second part is about Big Data platform and higher level processing language that called as Domain Specific Language for telecom industries:

- Information about Petaminer Big Data Platform
- DSL and Result Viewer Solutions
- DFD Analysis

Chapter 2

Big Data Technologies and Characteristics

Nowadays, quintillion bytes of data are created and this data exponentially grows day by day. Traditional databases, management and analysis tools, algorithms and processes cannot be easily applied these large datasets which is called as Big Data [8]. These are gathered from everywhere such as social networks, sensors, digital signals, health, finance, science and so on. In other words, almost every dataset has great potential to become Big Data because it has higher importance to get valuable information for many organizations. However, there are great challenges to achieve this because of storing data that is unstructured format, processing and analyzing without using traditional RDBs (Relational Databases) and analyzing tools. Therefore, there are special systems to perform these kinds of operations. All these system is generally called as Big Data concept. This concept has three main characteristics:

- Volume
- Variety
- Velocity

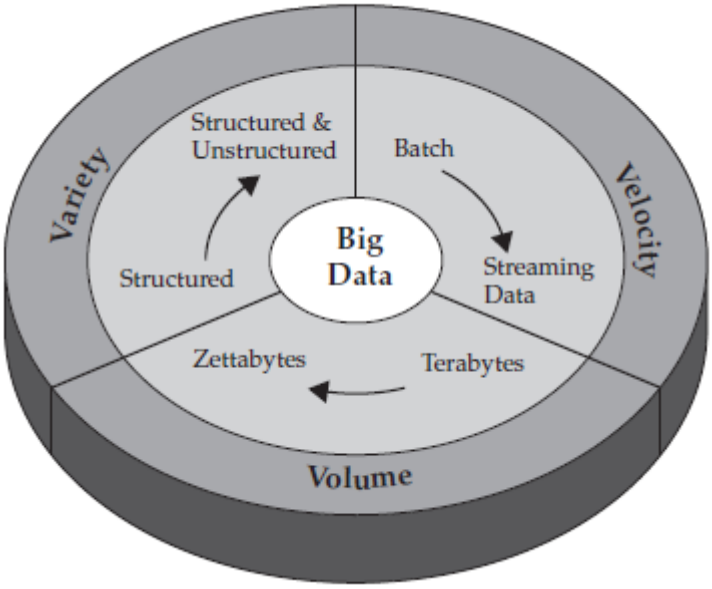

Figure 1**:** IBM characterizes Big Data by V^3

2.1. Characteristics of Big Data Concept

2.1.1. Volume

Volume of data that is stored to analyze is extremely increasing. Today, there are almost 2.7 Zettabytes of data in the digital world. If we look at social media, 100 Terabytes of data are uploaded on Facebook and 230 million of tweets are written every day. Also 2.9 billion emails are sent in everyday routine [9].

Every year structured and unstructured data grow %60 and 35 Zettabytes of data are expected to be generated annually by 2020 [9].

Day by day volume of data change terabytes to petabytes and will be zettabytes. Therefore to handle and store this large datasets become very important for organization over Big Data concept.

2.1.2. Variety

Volume of data is increasing and Big Data concept is not only deal to handle these. It has to deal with its variety. Because data come from everywhere such as sensors, mobile devices, web, social networks, emails, log files and so on. All of them have their own types. Therefore there is not only structured data types but also semi structured and unstructured which is not suitable to store in traditional RDBMS.

To combine and store structured and unstructured data types is another important characteristic and big deal for Big Data concept.

2.1.3. Velocity

Another characteristic of Big Data concept is time of between arrival, to be stored and also to be processed of data. It is very important to handle and to process data in this short interval. For example, Google, which have an extreme Big Data infrastructure, has a great search algorithm. When we type something its search engine, millions of results return in milliseconds. This is related with how organizations are efficiently stored and processed data as quick as possible.

These are 3 main characteristic which called as V^3 [2]. According to organization infrastructure, number of characteristics may be increased. However, all of them are always related with Big Data concept.

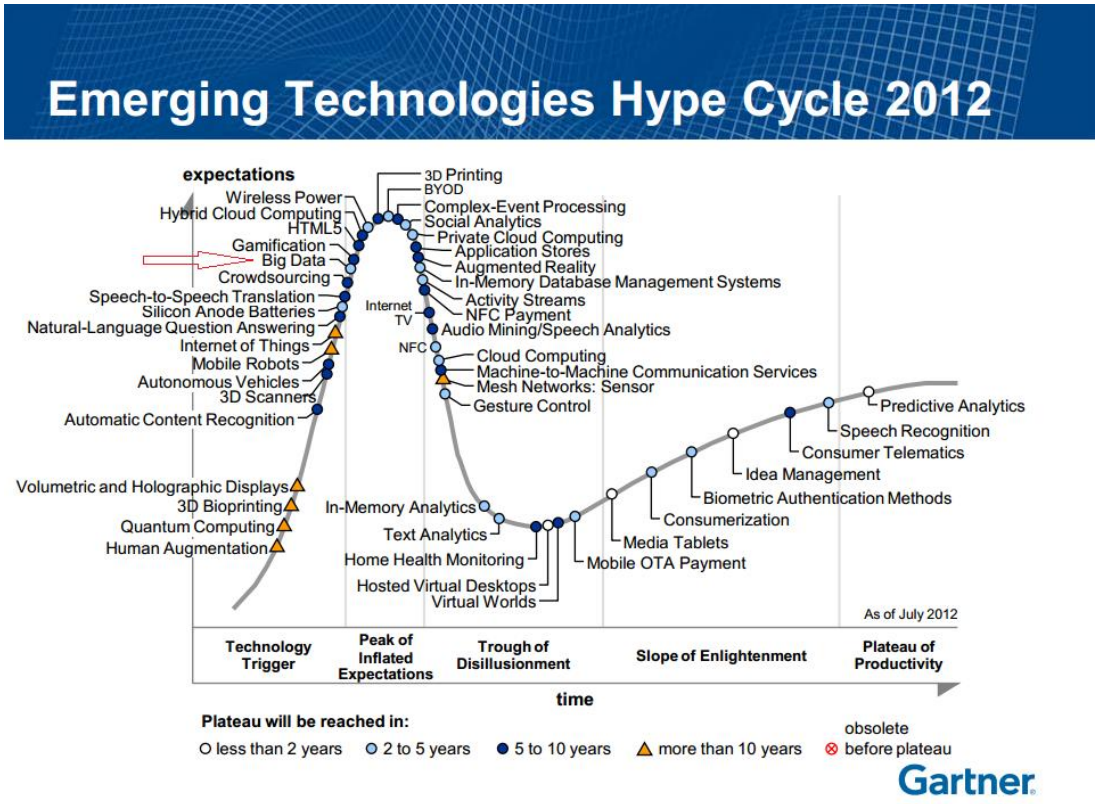

Figure 2: Emerging Technologies Hype Cycle 2012

2.2. Types of Tools in Big Data Concept

Big Data concept is just not deal with traditional approaches as we mentioned before. Therefore some tools and algorithms which are specific for Big Data tools are created to perform large datasets in this concept. There is some important factors to determine which type of tools are suitable:

- Infrastructure (Distributed Servers / Cloud / Storage)
- Programming Models (Distributed Algorithms)
- High Performance Schema Free Databases (NoSQL DBs)
- Processing & Analyzing (Data Mining etc.)

2.2.1. Infrastructure

Big Data needs big infrastructures such as lots of distributed servers, cloud systems and storage tools. Many organizations do not need to set up this physical system because big vendors (such as Google, Amazon, etc.) of the market serve this platform as IaaS. These vendors provide flexible, scalable, fail-safety virtual systems [1]. Infrastructure of Big Data systems has two parts:

- Distributed Servers (Cloud)
- Distributed Storage

2.2.2. Distributed Servers

Distributed Servers, which is generally called Cloud Servers, hold computing and storage units on different machines. They provide cluster of machines that can be easily manageable and customizable. Therefore any user of this kind of server vendors does not need to change anything physically when storage and number of processes are rapidly increasing. Because Distributed Server Vendors just offer a web UI, that customer can change his distributed virtual system by changing number of machine and its properties. Some important Distributed Server infrastructures are:

- Amazon-EC2 (Amazon Elastic Compute Cloud)
- Google App Engine
- Windows Azure
- Elastic
- Beanstalk
- Heroku

2.2.3. Distributed Storage

Processing of large datasets cannot allow performing special data operations by using traditional file systems that are not design for it [1]. Because, large sets of data are written or read multiple gigabytes at once while dealing big data. To solve this issue, large datasets are stored across multiple machines that are called Distributed Storage. There are two different Distributed Storage infrastructures:

- Amazon-S3 (Amazon Simple Storage Server)
- HDFS (Hadoop Distributed File System/Server)
- GFS (Google File System)

GFS and Amazon-S3 are designed for specific usage. For example, GFS is optimized for Google's search operations. Amazon-S3 is generally used for personal storage and process operations. However, Amazon-S3 can offer Hadoop infrastructure according to customer's needs.

On the other hand, HDFS is open source solution for processing distributed large datasets. Also it has very important differences from other distributed systems. First one is detection of faults and other one is that HDFS does not need high-cost hardware.

Hadoop File System

HDFS consists of a main NameNode and worker DataNodes. NameNode manages some file operations like read, write and so on. It also organizes mapping of blocks to DataNodes. DataNodes are responsible for processing file operations. They also manage creation of the new operational blocks.

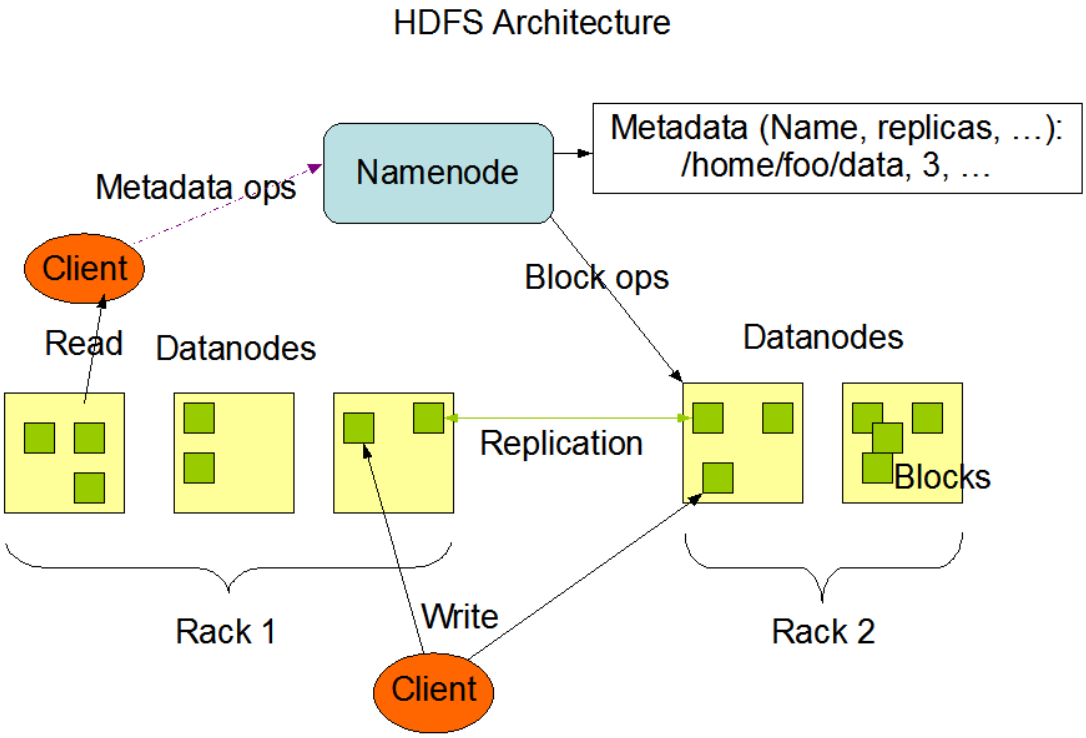

Figure 3: HDFS Architecture

(Source: http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html)

There are some key goals of HDFS system. It has powerful and quick fault detection infrastructure. It enables to access data quickly with streaming data access support. It deals with large datasets. Modular system of HDFS can be portable many heterogeneous software and hardware systems.

HDFS Failure Check System

HDFS has a great failure check system to prevent any data loss. Especially, data disk failure, data integrity failure, and metadata disk failure are always controlled by HDFS failure check system.

Data disk failure is checked by NameNode. In HDFS structure, every DataNode sends Heartbeat messages to the NameNode periodically. Therefore, NameNode knows which DataNode is working and which is not. Therefore if one of the disks fails, NameNode creates a replication of this DataNode on a healthy disk.

Data integrity is another important issue for HDFS. Dealing with streaming data always requires checking failure or corruption of content. Thus, HDFS has a checksum system for every block of file to save data integrity. In that case, every DataNode verifies all content by using related checksum before data processing.

Metadata files have many replications on HDFS. When any file is changed in there, other replications of files automatically are changed.

2.2.4. Programming Models

Standard programming models generally do not deal with how much data is processing. In that case, to simplify complex programming tasks, Map Reduce algorithm is developed by Google and now it is most popular programming model approach all around the world [10].

Before usage of this algorithm, hundreds or thousands of machines are needed to process large datasets in an acceptable time In that case, parallelize, distribution of data, fault tolerant and load balancing are the most complex operations while dealing with large number of machines. Map Reduce library offers powerful interface that enables to process on distributed systems without handle these kinds of operations.

There are some examples that are easily solved by using Map Reduce algorithm: Count of URL Access Frequency, Reverse Web-Link Graph, Term-Vector per Host (importance level of a part in large sets), and Inverted Index (for search operations on datasets).

Map Reduce is generally called a framework that main target is to solve problems by using parallel processing approach. There are three main step:

- First the problem is divided into smaller problems. (Map Step)
- Second, these smaller problems are solved in parallel way.
- Finally, all solutions which come from small parts are combined to create original problem solution. (Reduce Step)

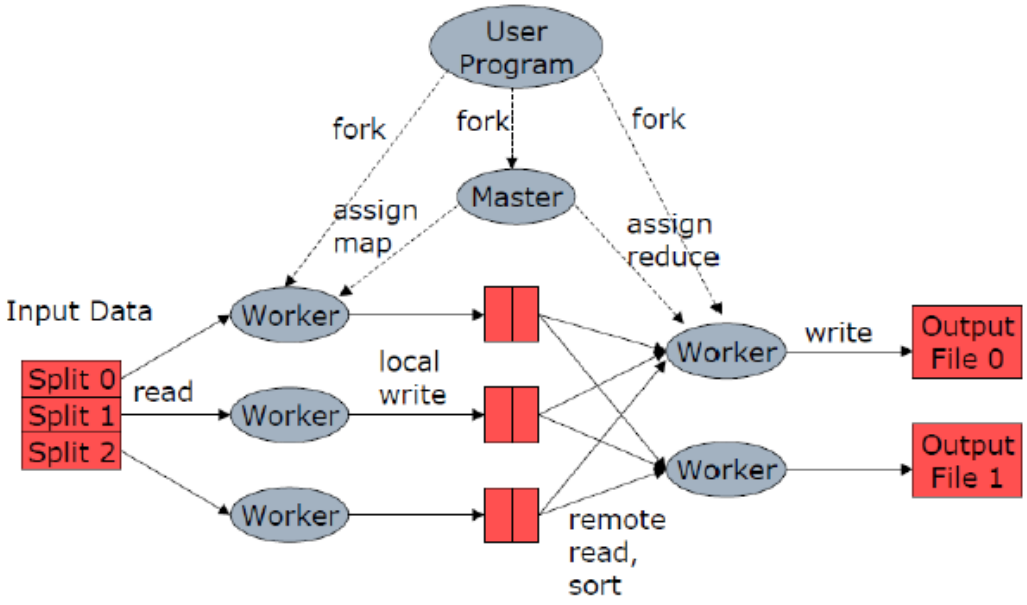

Figure 4: Map Reduce Algorithm

When user program creates map and reduce tasks, there are some limitations to increase the efficiency of the system. Number of maps is generally calculated by using input size and block size. Therefore, 1 TB data on the 128 MB block size are

usually needs approximately 8000 maps. On the other hand, number of reduces is related with number of nodes multiplied by 0.95 or 1.75.

The Map Reduce framework works with <key, value> pairs. Therefore, it gets the input as a set of <key, value> pairs and generates output as a set of <key, value> pairs at the end of the job. There is a sample code to find out general structure of Map Reduce:

```
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();

public void map(Text value, OutputCollector<Text, IntWritable> output){
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
    word.set(tokenizer.nextToken());
    output.collect(word, one);
    }
}

public void reduce(Text key, Iterator<IntWritable> val, OutputCollector<Text, IntWritable> output){
    int sum = 0;
    while (val.hasNext()) {
        sum += val.next().get();
    }
    output.collect(key, new IntWritable(sum));
}
```

Figure 5: Example Map Reduce WordCount Code

Map Reduce is the most popular algorithm, so many Big Data systems (analysis and processing tools) are based on Map Reduce framework such as Hadoop, Hive, Pig, Cascading, Cascalog, Sawzall, Dremel, mrjob, Caffeine, S4, MapR, Acunu, Flume, Kafka, Azkaban, Oozie, Greenplum.

2.3. High Performance Schema Free Databases

When organizations deal with large datasets, relational database systems are not enough to perform data operation and processing. Therefore, High Performance Schema Free Databases that are called as NoSQL (Not Only SQL) Databases are designed to perform data operations efficiently and quickly. There are some characteristics of NoSQL DBs that traditional databases cannot support:

- Support to process large datasets at once.
- Work on distributed systems.
- Different from standard SQL interface.
- Based on key value storage type like JSON (Schema free, no need to predefine).
- Fast, flexible, easy to write queries.

Some important implementations are BigTable, HBase, MongoDB, Cassandra, Redis, CouchDB, Hypertable, Voldemort, ZooKeeper and so on.

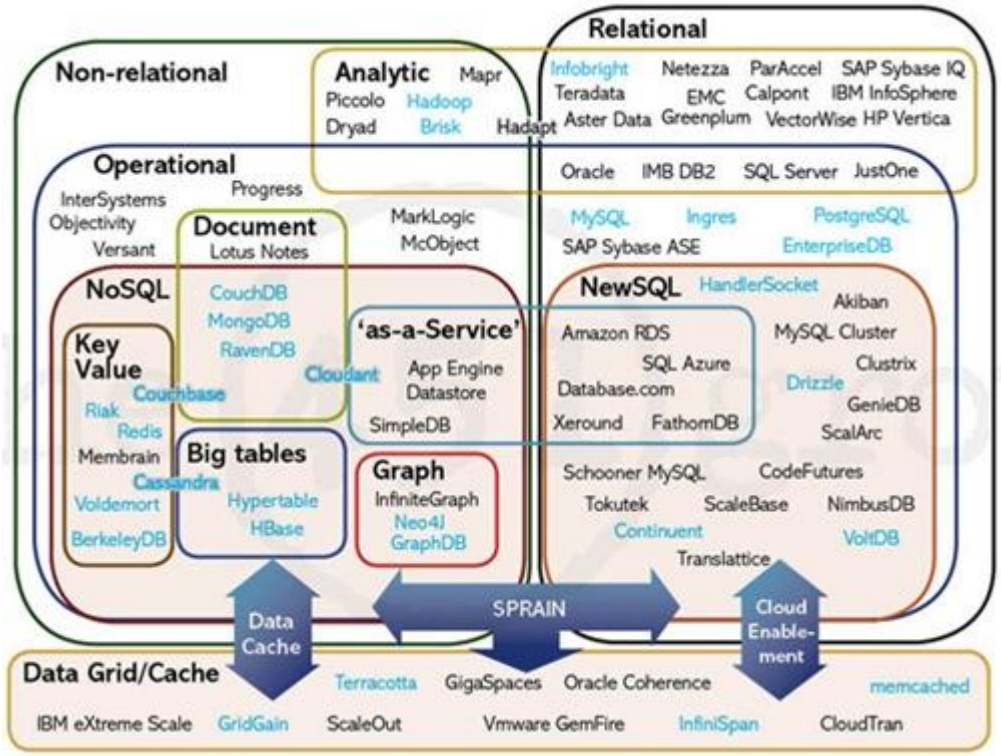

Figure 6: NoSQL vs RDBMS

(Source: http://cdi-mdm.blogspot.com/2011/07/nosql-newsql-and-mdm.html)

DB Type	Specifications
NoSQL	High Performance
	Linear Scalability
	Schema Free/Flexible
RDBMS	Complex Joins
	Predefined Schemas
	Declarative Syntax
	Transactions (ACID)
NoSQL & RDBMS	Deals with data and its operations

Table 1: NoSQL vs RDBMS Basics

2.4. Processing & Analyzing

The amount of data has been increasing exponentially. Processing and analyzing these large datasets are most important to obtain valuable information from this sea of data. However it is big challenge to analyze growing volume of information that meaningful patterns are just handled by using special kind of processing and analyzing tools. Therefore there are lots of processing and analyzing tools such as Hive, Pig, Mahout, R and so on.

Pig and Hive are most popular tools on Hadoop platform. Pig is firstly developed by Yahoo and Hive developed by Facebook. Then, both of them are taken under Apache open source platform foundation.

Pig

Pig is the one of the most important high level processing platform for creating Map Reduce programs. This platform has own language that is called as Pig Latin. It is very similar with SQL. It has also User Defined Functions (UDF) module that allow to user writing own functions by using Java, Phython, Javascript, Ruby or Groovy [11].

```
data =
    LOAD 'input'
    AS (query:CHARARRAY);

queries_group =
    GROUP data
    BY query;

queries_count =
    FOREACH queries_group
    GENERATE
        group AS query,
        COUNT(data) AS total;

queries_ordered =
    ORDER queries_count
    BY total DESC, query;

queries_limit =
    LIMIT queries_ordered $n;

STORE queries_limit INTO 'output';
```

Source: http://pig.apache.org/docs/r0.8.1/pigunit.html

Figure 7: Example Pig Script

Pig Latin is a language that fits between low level styles of Map Reduce and declarative style of SQL. Even if it seems like SQL type, Pig Latin is procedural [15].

Hive

Hive is a data warehouse system that can be integrated with hadoop for analysis of large datasets. It provides a language like SQL that is called as HiveQL. It has full support of Map Reduce operations. Thus, there are some important properties of Hive. Indexing is used for increasing speed of operation. Hive also supports different storage types (HDFS, TEXT, SEQUENCEFILE, HBase, etc.). UDFs can be used to handle some new operations in Hive platform.

```
hive> INSERT OVERWRITE TABLE events SELECT a.* FROM profiles a;
hive> INSERT OVERWRITE TABLE events SELECT a.* FROM profiles a WHERE a.key < 100;
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/reg_3' SELECT a.* FROM events a;
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_4' select a.invites, a.pokes FROM profiles a;
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_5' SELECT COUNT(*) FROM invites a WHERE a.ds='2008-08-15';
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_5' SELECT a.foo, a.bar FROM invites a;
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/sum' SELECT SUM(a.pc) FROM pc1 a;
```

Source: https://cwiki.apache.org/confluence/display/Hive/GettingStarted

Figure 8: Example Hive Script

2.5. Case Study: Hadoop

Big companies like Google, Microsoft, Amazon have their own big data solutions. However, they are not open source and not suitable for middle-scale organizations. At that point, Doug Cutting developed an open source version of this concept called Hadoop which is based on Map Reduce algorithm [10]. Then, Yahoo and other organizations support this framework and Hadoop became a distributed parallel processing framework which has own distributed database, file system; and also processing, analytic and visualization tools. Now it is one of the most powerful, scalable, flexible parallel computing frameworks that supports many open source processing and analyzing modules. Hadoop framework has these main modules [4]:

- Hadoop Common: Utility tool that supports all Hadoop modules
- Hadoop Distributed File System (HDFS): File system of Hadoop to manage Big Data
- Hadoop YARN: Cluster management system of Hadoop
- Hadoop MapReduce: Parallel processing modüle of Hadoop
- Ambari: Hadoop cluster monitoring module
- Avro: Data serialization system.

- Cassandra: A high performance schema free database for Hadoop (NoSQL DB)
- Chuckwa: Distributed system management tool.
- HBase: A high performance schema free database for Hadoop (NoSQL DB)
- Hive: Data analyzing tool for Hadoop
- Mahout: Data mining library for Hadoop
- Pig: High level language for execution of parallel operations
- ZooKeeper: Coordination service for Hadoop system.

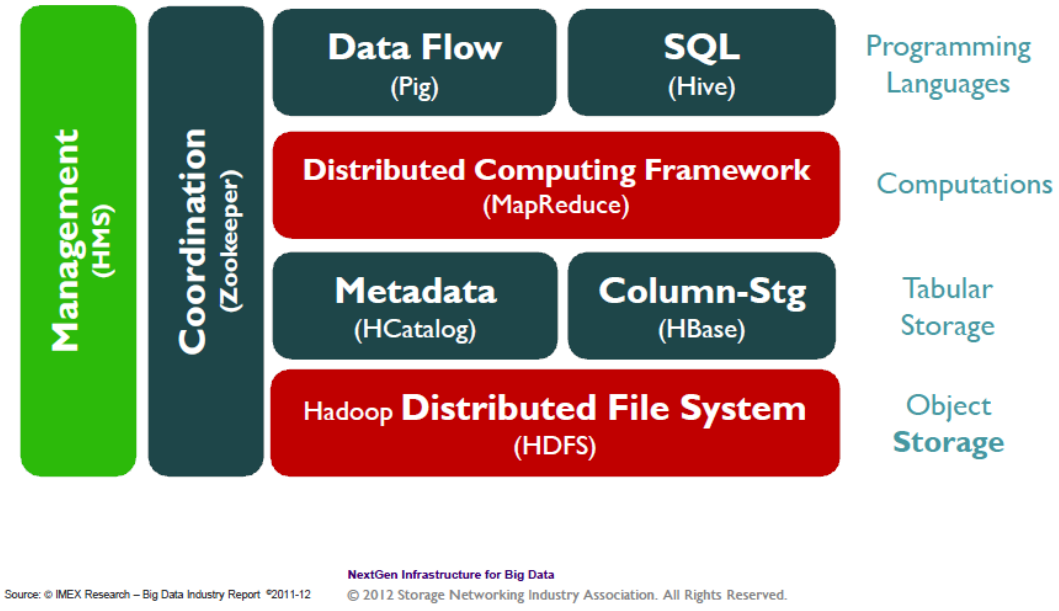

Figure 9: Hadoop Infrastructure

Chapter 3

Higher Level Domain Specific Language For Big Data Concept

3.1. Big Data Solution For Telecom Industries: Petaminer

Global data usage has been extremely increasing day by day. Therefore many enterprises want to collect these data and try to get best analysis for product strategy, targeting sales, business performance optimization, prediction & recommendation, resource management and so on.

Most of enterprises try to adopt their systems as Big Data concept. However, there are some key industries that will really gain advantage from this infrastructure.

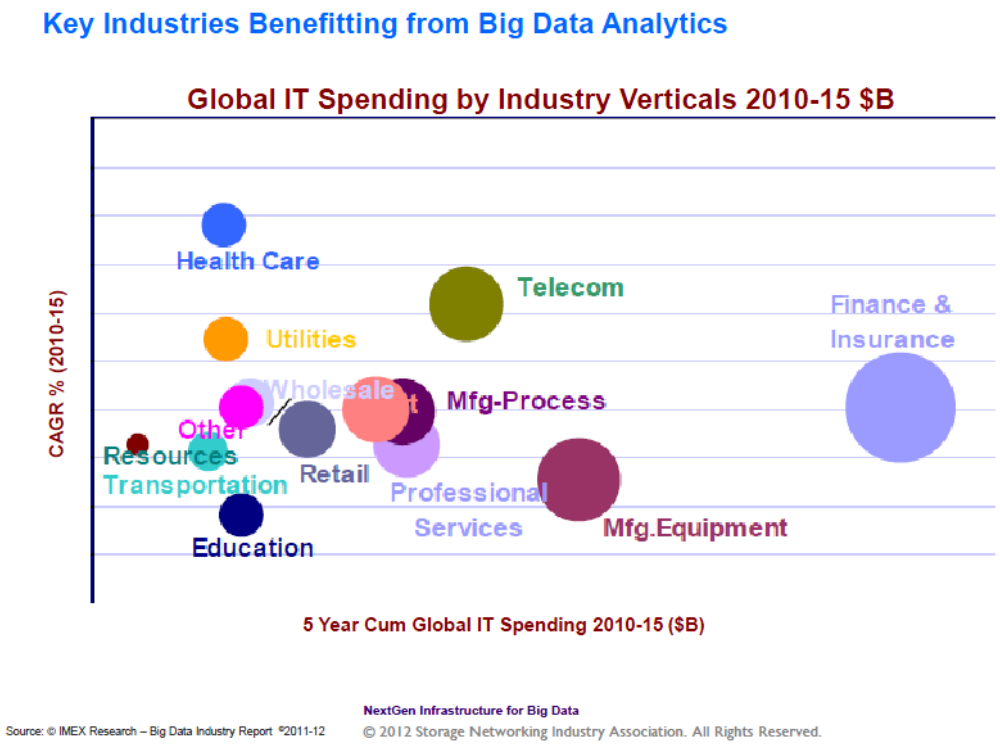

Figure 10: Global IT Spending by Industry Verticals 2010-2015

Main benefits of big data infrastructure are generally same for all of these industries:

- Improvement of service quality
- Prevent long time running queries for large datasets
- Get valuable information from multiple dimension and unstructured data
- Storing and accessing Cold Data is very effective in lower cost environments
- Single platform with modular tools
- Linear scalability

On the other hand, when organizations deal with big data, there are some challenges that traditional data management and processing tools cannot handle. First of them is collecting data sources from different kind of platform. Second is formatting unstructured data into a format. Third one is processing and analyzing huge amount of data. Final of them is to find best cost / performance ratio for the organizations.

In this section, I'll introduce a big data system of Elkotek for telecom industries that is called Petaminer. Then, I'll present a new higher level language –called DSL- for processing and visualizing specific telecom operations easily. Finally, I'll share related works about similar big data processing language solutions.

Petaminer is big data management solution for telecom industries. It is developed by Elkotek. It has 4 main big data aspects:

- Scalable
- Reliable
- Cost Effective
- Built on Industry Standards

This big data management system can allow increasing processing and storage capacity on demand. Therefore, it is not a big issue to start a small system and grow

afterwards. Configurable data storage settings and task failure handle system make this platform very reliable. Moreover, users -organizations- do not need to buy any physical system or invest on any data modeling. Finally, it uses most powerful Hadoop infrastructure and Google's Map Reduce algorithm.

Petaminer consists of 4 main layers:

- Collect
    - From different sources: DWH, RDBMS, FTP, Syslog..
    - In different formats: CDR, Free Text, Email, Social Media, Log File..
- Manage
    - Process: Workflow Mngt, Scheduling, Data Transformation, Alarm Mngt.
    - Store: Raw and generated data
- Analyze
    - Parallel Processing Algorithm: Map Reduce
    - Hadoop analysis modules: Hive, Pig, Mahout
- Report
    - Export the results: integration with RDBMS, DWH, FTP..
    - Enable real time query: Millisecond level query performance on provided API

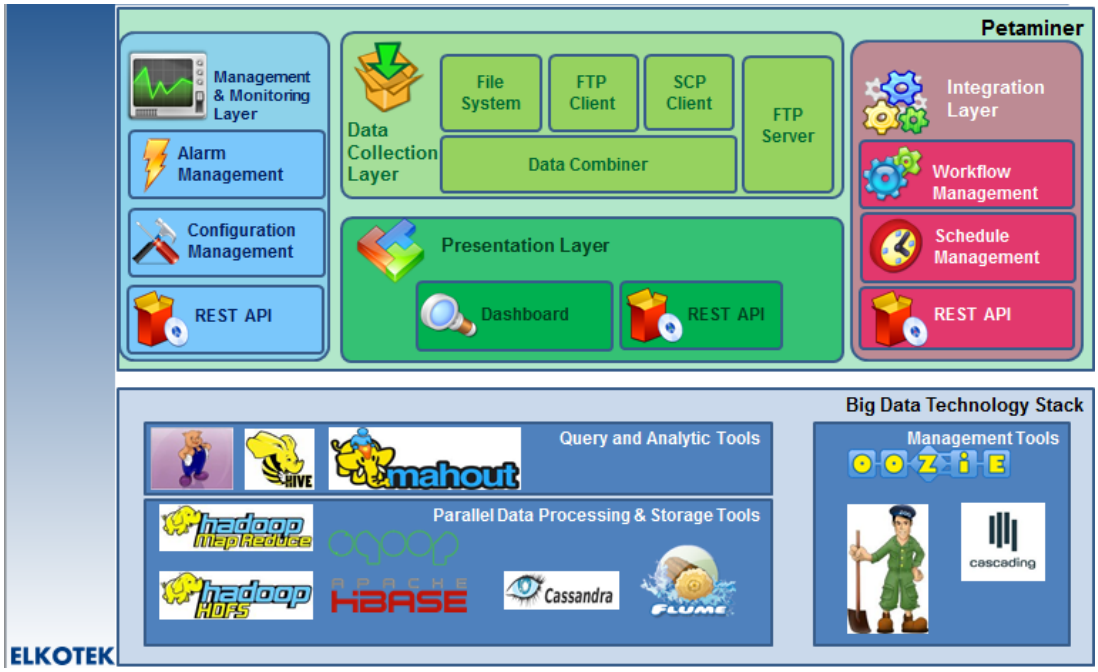

Figure 11: Petaminer Big Data Solution Platform

There are some excellence points that make Petaminer a better solution for Big Data concept in telecom industry.

- HDFS-aware FTP Server
    - Data, that is put on FTP, is automatically distributed on distributed HDFS cluster
    - If you have large datasets to push via FTP, you cannot deal with local disk sizes

- Selective Mapping
    - Map Reduce naturally iterates over all dataset
    - On the other hand, Petaminer's Selective Mapping layer increases query performance by running queries only on chunks that contains results
- Flexible Preprocessing
    - According to Metadata Binding concept, each data has more meaning in its context. Petaminer can store metadata about the context of the data such as server ports, generation time, folder name, properties and so on.
    - Data Combiner helps Hadoop to manage large datasets on distributed clusters.
- Parallel Compression
    - Compression ratios and performances are very important on data management and processing platforms.
    - Petaminer provides a parallel data compression and decompression method to increase disk utilization, query performance and decreases processing time.
- Domain Specific Language Infrastructure
    - Domain specific languages help business users to run complex queries while staying in their domain eliminating to learn to write scripts in Java, Pig or Hive languages
- Result (Report) Viewer Tool
    - DSL related result visualizer helps to get perfect visual results just writing simple DSL commands.

Lifecycle of Petaminer starts with store and search of big data. Then, search and analyze. Finally, analyze and data mining. All of these steps are applied on large datasets of telecom industries by Petaminer. This platform offers solutions such as [1]:

- Storage and Search
    - Application Logging
    - CG NAT Log Analysis
    - CDR Log Analysis
- Search and Analytics
    - DPI Reporting
    - Email Archiving and Analysis
- Analytics & Data Mining
    - Network Analytics
    - Social CRM

3.1.1. Storage and Search Solutions

Application Logging

Application logs are generally stored on RDBMS in many organizations. Large numbers of logs are created by enterprise applications and these are still increasing due to security concerns and regulatory requirements. However, traditional RDBMSs are not suitable for storing and searching operations because of large datasets. So these kinds of operations require new set of Technologies such as distributed infrastructure.

CG NAT Log Analysis

Telecom operators need to deploy CG NAT solutions when they need transition from IPv4 to IPv6. Logging IP usage records and being able to query them is a necessity for companies that deploys CG NAT solutions. Correlating this entire network logs with others help to identify customers and answer questions received from legal entities. This is another problem to require Big Data solution.

CDR Log Analysis

Telecom companies create huge amount of Call Detail Records (CDRs) every second. Processing, analyzing and serving CDR content requires real big processing power and storage capacity. Therefore, Big Data technologies offer to build more cost effective and high performance solutions.

3.1.2. Search and Analytics Solutions

DPI Reporting

DPI is one of the most popular analyzing concepts in telecom industry. Traditional database systems cannot allow inspecting all data source and dropping lots of information because of lacking database management system or data warehouse power and capacity. Big Data technologies prevent this kind of data lose and also enable to process at real time. This type of reporting gives lots of valuable information about customer behavior.

Email Archiving and Analysis

Emails have very important part of communication platform today. Many organizations do not offer an archival solution. End users generally save their

archives on their computers. Email Archiving and Analysis system offers a searchable email archive that handle complex queries for the organizations. It is very easy to detect usage patterns while dealing with their enterprise mailings.

3.1.3. Analytics & Data Mining Solutions

Network Analytics

Customers generally deal with various kinds of networks and systems. To define customer experience always requires processing and analyzing all these large network datasets such as structured or unstructured data, various kind of data and so on. Petaminer Network Analytics solution helps enterprises to collect, correlate and analyze their customer behavior. Different kind of customer data is always very useful to find hidden patterns in Big Data concept.

Social CRM

Social networks are one of the most important platforms to find out customers and potential customer's needs. Analyzing social platforms help organization to determine unhappy customers and solve their problems quickly, before customers think negative about your business. Furthermore, social analysis gives the power of determining marketing plans and identifies future expectation of organizations.

When customers of Elkotek –Petaminer platform- need to analyze a file, Elkotek must write necessary scripts every time. This is too much time consuming for Elkotek and their customers. This case is generally same as every big data platform for all industries.

At that point, we provide a new solution that is called as Domain Specific Language in that study. This is a new higher level language that customers do not need to write any complex Pig, Hive queries or Map Reduce codes. Therefore, customers of big data platforms such as Elkotek –Petaminer platform- can easily analyze their files on their own.

3.2. Domain Specific Language For Telecom Industries

Domain Specific Language is the one of the main part of this solution. This concept includes:

- Domain Specific Language - DSL
- Distributed File Descriptor - DFD
- Result Viewer

We designed DSL, DFD and result viewer by examining powerful big data analysis platform - Petaminer - of Elkotek. Integration of the whole system described below:

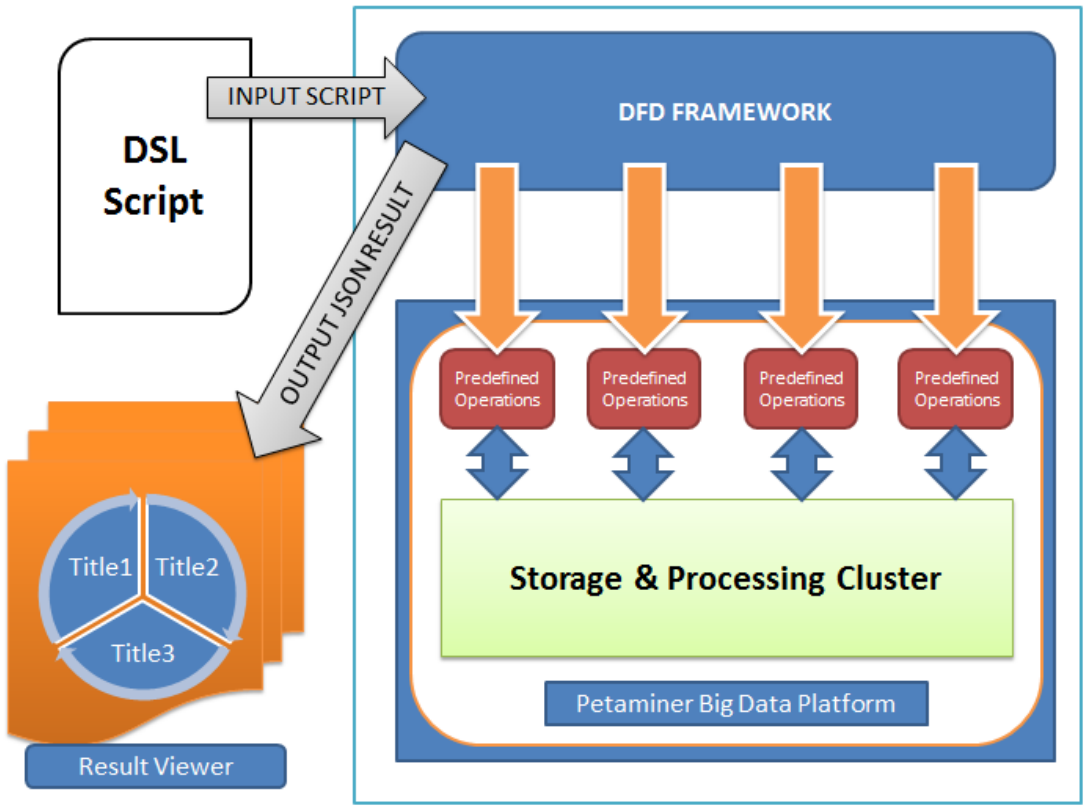

Figure 12: DSL Solution for Big Data Platform

First of all, user writes DSL script to analyze files. Then, this script is sent DFD module of Petaminer platform. In that section, DFD analyzes DSL script and converts it into right structure. Then DFD determines correct operation analysis frameworks and starts processing. Finally, generated output file information is sent to the result viewer as a JSON format. Result Viewer reads the JSON and shows related diagram for the user.

3.2.1. Domain Specific Language - DSL

This is very high level language. It is designed to prevent writing complex queries such as Java, C Map Reduce algorithms. Also, it is short language and easy to implement. Therefore, it prevents too much time consuming.

System works with two kinds of files. These are input files and output files. There are some predefined input files such as:

- NETWORK_FILE
- SUBSCRIBER_FILE
- BILLING_FILE
- PAYMENT_FILE
- CROSS_SALES_FILE
- COMPETITOR_FILE
- CUSTOMER_COMPLAIN_FILE

We can think predefined files as first class object called as File Descriptor -FD-. We define file objects like:

FD Input_FD;

Input_FD.fileType = "NETWORK_FILE";

Therefore analyzing of these predefined files needs some operators. All operators are in telecom domain and know about predefined input files. These are:

- Network Engineering Operators
    - DPI
    - NETWORK_TRAFFIC
    - NETWORK_BANDWITH
    - NETWORK_CLUSTERING
- Customer Support Operators
    - USAGE_ANALYSIS
    - CHURN
    - COMPLAIN_ANALYSIS
    - TOP_APPLICATION
    - ANORMALY_DETECTION
- Finance & Marketing Operators
    - MOST_PROFITABLE_CUSTOMER
    - MOST_PROFITABLE_TRAFFIC
    - CROSS_SALE

We can use predefined operators such as:

FD output_FD;

output_FD.location = "http://test.user/defined/output/path";

ouput_FD = DPI input_FD;

Predefined File Descriptor attributes for specific operations:

Attributes	Input File Descriptor		Output File Descriptor	
	Usage	Explanation	Usage	Explanation
fileType	“NETWORK_FILE”	Defines input files	-	-
location	“http:// server.com/file.xyz”	Location of single input file	“http:// server.com/newFile.xyz”	Single output file location
location[]	{location1, location2, ...}	Location of multiple input files	{location1, location2, ...}	Multiple output files location
failureCheck	“YES” or “NO”	Need more process power and storage	-	-
filterParams[]	{param1, param2, ...}	Under development according to customer needs	-	-
getColumns[]	-	-	{col1,col5, col9, ...}	Get the related column values from output files
titles[]	-	-	{title1, title2, ...}	Title names according to column that chosen
graphType	-	-	“PieChart” , “LineGraph”, etc.	Graph type for Result Viewer
graphName	-	-	“My Graph”	Graph Name

Table 2: File Descriptor Attributes

3.2.2. Distributed File Descriptor – DFD

DFD is the management system between DSL, Big Data platform -Petaminer- and Result Viewer. It gets script from DSL and convert it right structure to process on right operation framework. Then, it handles result files and sends them on Result Viewer in JSON format. DFD is an embedded solution on Big Data platform like user defined functions (UDF).

3.2.3. Result Viewer

After output files are generated, DFD send graph operations to the Result Viewer in JSON format. Result Viewer takes it and generates related graph according to user needs. Result Viewer will be implemented after all of these DSL and DFD structures are finished.

3.2.4. Case Study: DPI Operation

DPI is the one of the most popular topic in telecom industry. Both detect of service attacks –especially DoS- and customer behavior analysis, that is most hot topic these days, is related with DPI operations.

Our implementation works with predefined network files. These files are generated by using row-column style or CSV pattern standards. Therefore, user knows details about network file and how to apply right operations on it. After applying operations, output file has a predefined pattern and user can send this file to the Result Viewer. Result Viewer shows these values graphically. Example script;

```
FD FD_Output;

FD_Output.location=“http://server.com/networkTraffic_2013_final_Out.txt”;

FD FD_Input;

FD_Input.fileType = “DPI”;

FD_Input.location = “http://myserver.com/networkTraffic_2013_final.txt”;

FD_Input.failureCheck = “YES”;

FD_Input.filterParams = {noRepeat, between: 04/13-05/13};

FD_Output = DPI FD_Input;

FD_Output.getCols[] = {1,8,9};

FD_Output.titles[] = {VoIP-Traffic, Email-Traffic, X-Traffic};

FD_Output.graphType = “lineGraph”;

FD_Output.graphName = “DPI Traffic Analysis”;
```

This is an example DSL script which performs DPI operation on a specific file. After, DFD directs the output solution to the Result Viewer.

Chapter 4

Performance Results

We examined two kinds of performance results. First one is data processing and analyzing performances. Another one is database write performances with HBase. These are the key points to increase the general performance of the Big Data systems. Increasing performance of DSL operations is related with performance of Big Data platform. Therefore, we examined the some point of the Petaminer performance results to analyze effects of the processing and database units.

Processing and Analyzing Performance Results

First of all, we observed the performance difference when number of nodes and its map-reduce tasks are changed. Therefore, there are 6 different throughput results that first two of them have 2 mapper tasks, second two of them have 3 mapper tasks, fifth one has 6 and last one has 8 mapper tasks (Table 3). There are 10 GB of data per node that has 1024 MB memory.

# of Nodes	Execution (seconds)	Throughput (MB/s)
3	2629	11
3	2233	13
3	2069	14
4	1514	27
4	1380	29
4	1250	32

Table 3: Node Scalability Table

If we carefully look at the results, number of mapper tasks decrease execution time and increase throughput. On the other hand, increasing number of nodes

provides extremely rise of throughput. Throughput of 4 nodes with 3 mappers is twice of 3 nodes with same number of mappers (Figure 13).

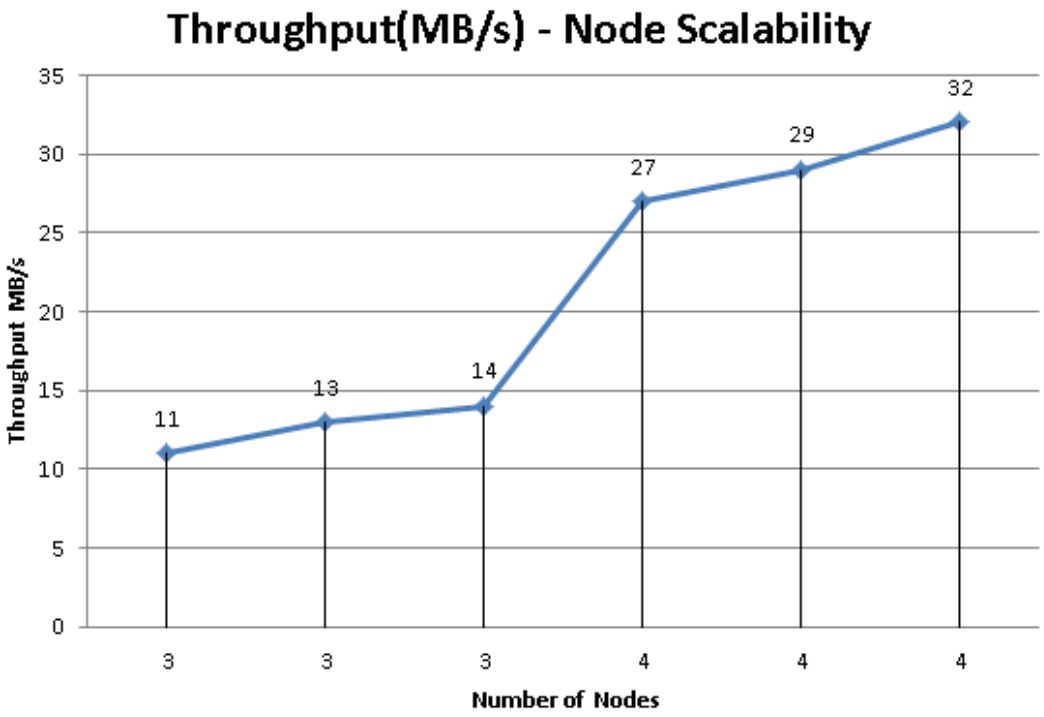

Figure 13: Node Scalability

Another important performance issue while dealing with big data is to decrease execution times. In that case, we examined execution time results. Like first performance results, there are two sets of results. One of them is execution time results with 2 and 3 mappers on each of three nodes. Another one is execution time results with 3, 6 and 8 mappers on each of three nodes (Table 4).

# of Map Reduce Tasks	Execution 1	Execution 2	Mean
2, n = 3	2629	2233	2431
3, n = 3	2069	-	2069
3, n = 4	1514	1493	1503,5
6, n = 4	1470	1380	1425
8, n = 4	1250	-	1250

Table 4: Execution Time Table (n denotes number of nodes)

We observed that increasing number of mapper tasks decrease execution time when number of nodes equal to 3 or 4 (n = 3, n = 4). Moreover, number of nodes has a great performance effect to decrease execution time (Figure 14). There is an important time difference between while changing total number of nodes 4 up from 3 with fixed number of mapper tasks.

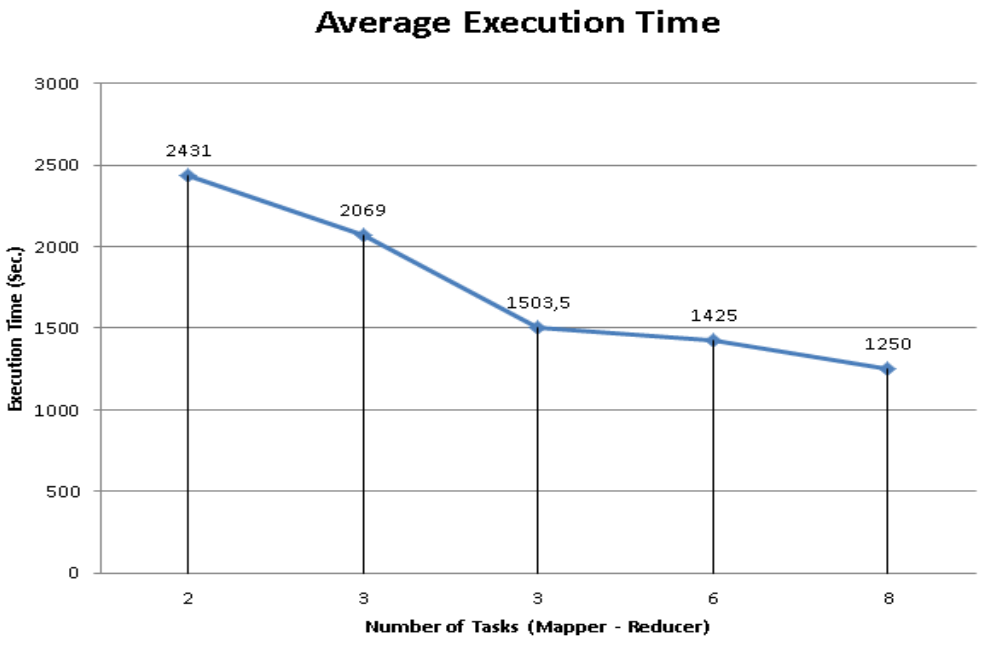

Figure 14: Execution Time

By using Table 4 values and variables, we created a speed up table to find out gain while dealing with difference number of map/reduce tasks and nodes (Table 5).

# of Map Reduce Tasks	Execution Time	Speed Up
2, n = 3	2431	1
3, n = 3	2096	1,175
3, n = 4	1503,5	1,617
6, n = 4	1425	1,706
8, n = 4	1250	1,945

Table 5: Speedup Table (n denotes number of nodes)

In Figure 15, it is possible to show execution time gain with speedup graph while increasing number of map/reduce tasks and nodes. Increment rate of speed up is at the higher level while increasing number of nodes as we expected.

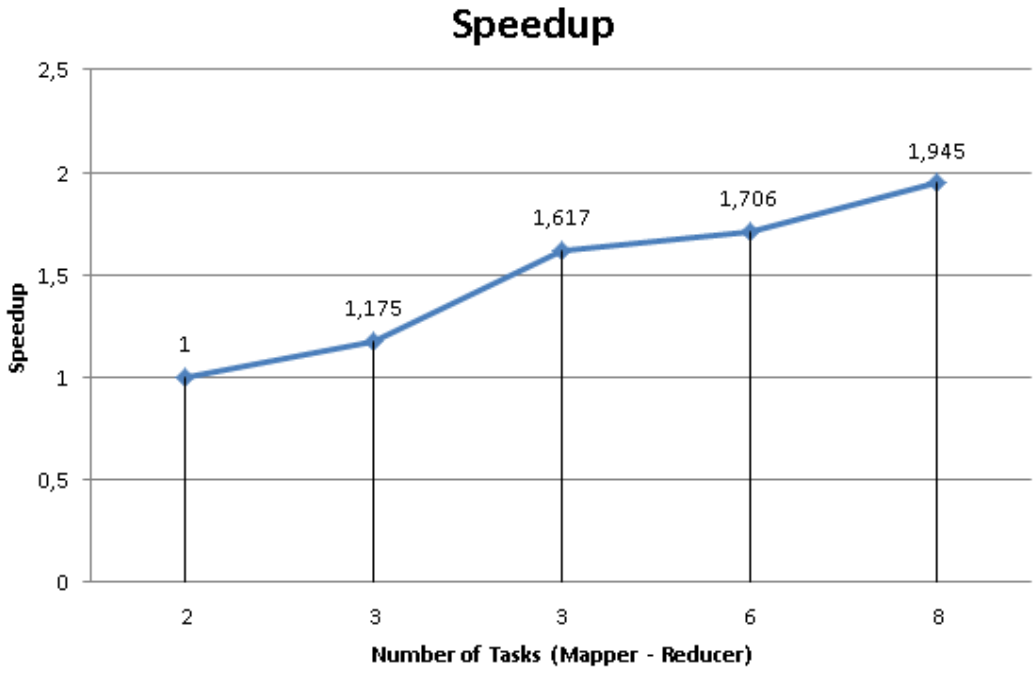

Figure 15: Speedup

HBase NoSQL DB Write Performances

Another important point for Big Data platforms is to write large datasets into DB as quick as possible. NoSQL databases like HBase are designed to perform these kinds of database operations efficiently. Therefore, some compression options like snappy driver and lzo increase the average import rate and also decrease the disk usage to increase HBase performance. In Table 6, there are some different samples such as 709 MB data, 1065 MB data and its compressed output, 95827 MB compressed data.

Data Size	# of Records	Total Time(sec)	Avg. Import Performance (MB/sec)	Disk Usage	Compression (LZO, Snappy)
709,11	10934599	73,995	9	1802	NO
1065	4863732	58,984	17	1331	NO
1065	4863732	49	21	530	YES
95827	437587321	4891	19	48784	YES
95827	437587321	5350	19	49029	YES
95827	437587321	5051	19	47246	YES

Table 6: HBase DB Write Performances

Average import performances are generally related with the data size and disk usage. Starting with smaller data (709 MB) without compression helped to figure out average import performance. Then approximately 1 GB of data is tested with compressed and uncompressed. There is average import performance difference between these samples that have same data size. Finally, average import rate of compressed big size of sample data (95827 MB) are examined to find out the effects of the large datasets (Figure 16).

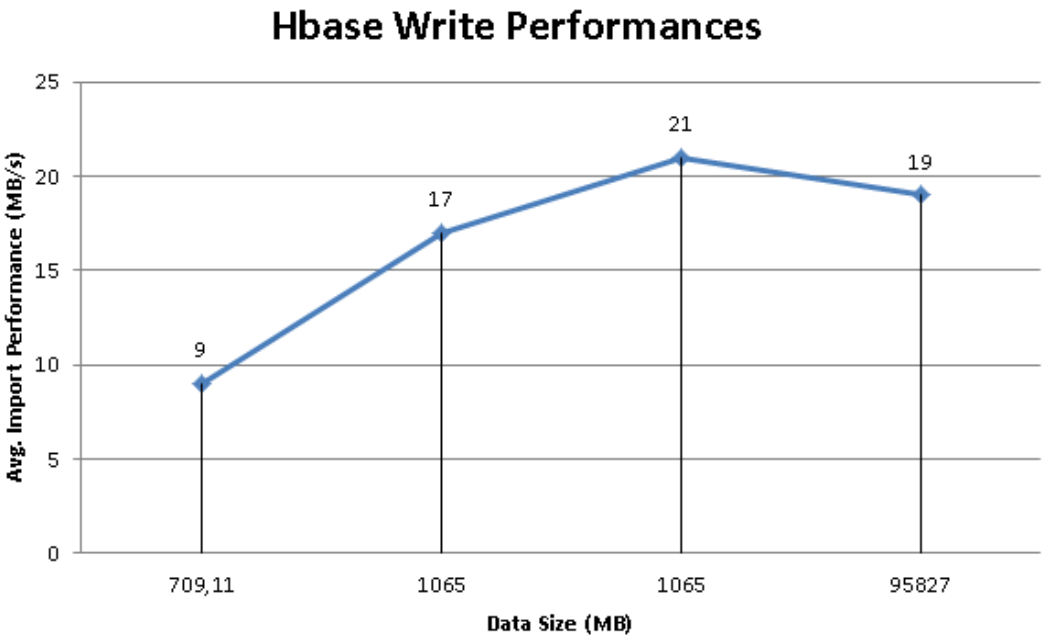

Figure 16: HBase Avg. Import Performance

It is very important to store big data efficiently. Therefore, increasing write performance of the system is related with data size and disk usage of these data. It is possible to decrease disk size some tools like lzo compression system. In figure 17 shows that disk usage of 1065 MB sample data is compared with its compressed sample.

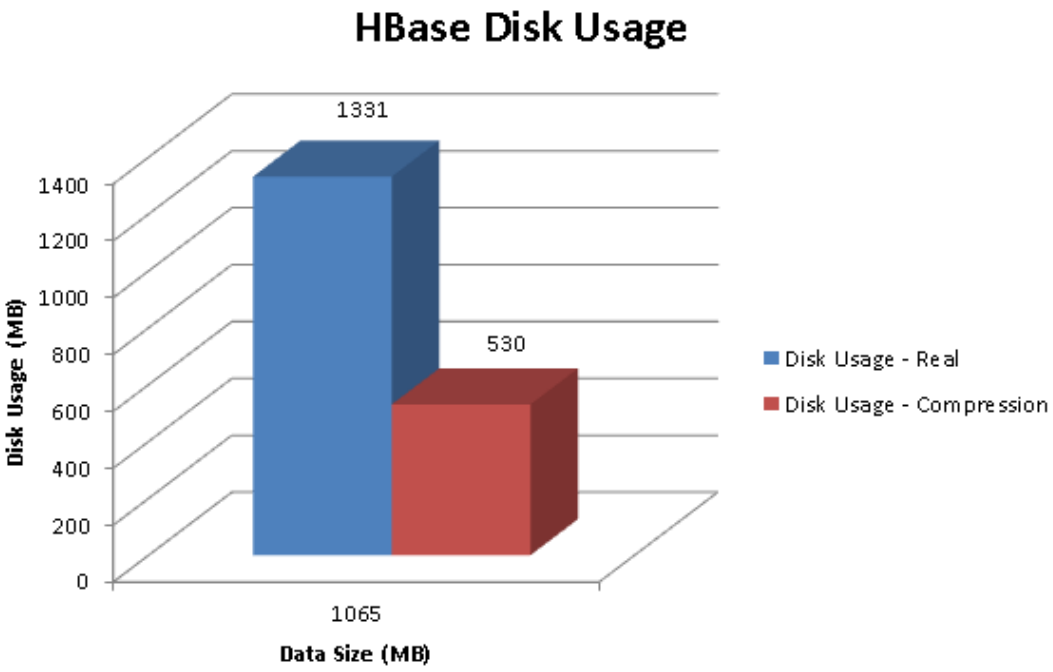

Figure 17: HBase Disk Usage – Sample 1

Disk usage of another sample that has large size (95827 MB) is tested three times with compression (Figure 18). Because, it is important to decrease disk size and increase average import rate while dealing with real big data.

In Figure 19, disk usage rate of compressed big sample is almost same as first sample in Figure 17. Therefore, it is possible to compress whole big data to decrease disk usage and increase average import rate.

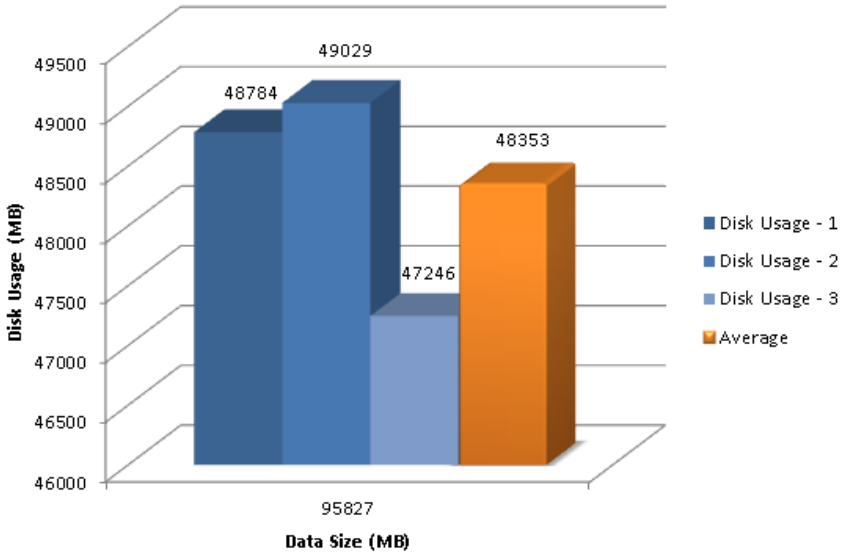

Figure 18: HBase Compression Tests of Sample 2

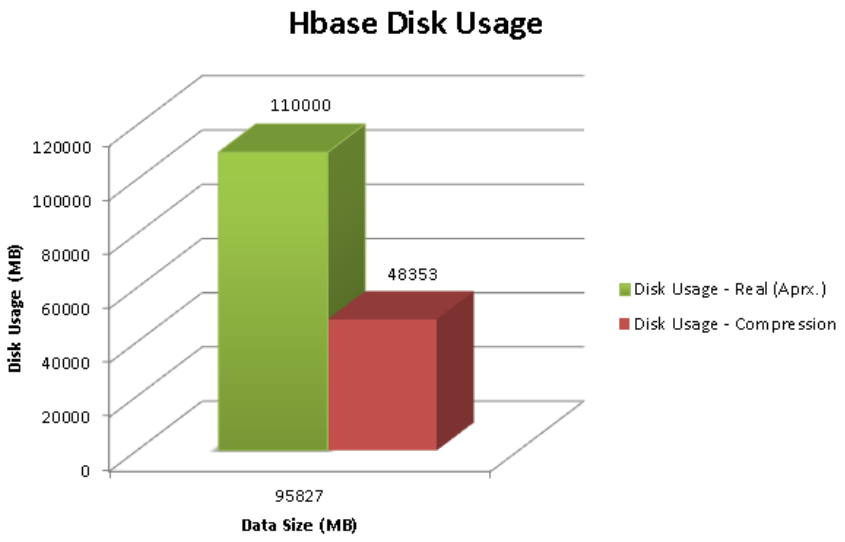

Figure 19: HBase Disk Usage – Sample 2

Chapter 5

Related Works

Hadoop [1] is one of the most popular open source modular and distributed system that processes datasets by using parallel computing approaches. It has lots of components to achieve this such as HDFS (Hadoop File System), YARN for cluster management, Map Reduce algorithm, Cassandra DB, Hive[2] for query execution, Pig[3] for high level implementation, Mahout for Data Mining and so on for specific operations. Our DSL solution is a high level language solution for this kind of Hadoop system but it has higher level implementation than Hadoop Hive, so it is very easy to implement specific operations. Also Amazon WS uses Hadoop infrastructure to perform high performance operations.

Map Reduce is a programming style that helps to process large datasets by splitting them among different machines. Therefore there are lots of analysis tools to create meaningfull knowledge by using Map Reduce approach. One of them is Dremel. Dremel is a query execution system that quickly analysis large datasets . It has similar architecture as Hive and Pig but provides powerful adhoc data analysis and fast query engine [12]. DSL provides a high level language above Map Reduce algorithms (such as Pig, Hive also Dremel) and prevent user to create complex SQL queries to analysis read-only datasets. On the other, DSL is a high level language to process specific operations (such as DPI, CHURN, so on…) on the file datasets.

FlumeJava is a library that increases the efficiency of Map Reduce tasks [13]. Unlike DSL solution, it is a low level library that helps to creates optimized Map Reduce operations to process large datasets.

Sawzal [14] is another domain specific approach which is a new language above Map Reduce framework. It is a language that easier to write Map Reduce programs. Like Sawzall, DSL is very simple to write for domain specific operations. However, Sawzall works above Map Reduce framework on the same layer. Our approach is higher level solution which just deals with writing file process operations. Therefore user not needs to change any process or algorithm in the Map Reduce layer for writing file processes. On the other hand, DSL requires a specific Map Reduce solution for distinct domains (such as telecommunication, health, science etc.).

Pig Latin is another data analysis language that works over Hadoop distributed computing system. It is developed by Yahoo and works between declarative SQL style and produceral low level Map Reduce style [15].

Spark is a parallel computing framework that uses RDD (Resilient Distributed Datasets) to decrease loss of data [16], [17]. Therefore, in every distributed machine when a partition is lost, the system generates same one for this specific cluster. DSL infrastructure has same option for fault tolerance. For Map Reduce operations DSL creates many copies of datasets to prevent the data loss.

Another large data analysis scripting language is SCOPE [18]. It has lots of features from SQL. In addition users can define new functions by using this scripting language. Like Sawzall and DSL, it has easy language to prevent writing many lines of code. However it owns a special parallel processing layer which called Cosmos. Another data analysis high-level programming language is DryadLINQ which is also under development at Microsoft [19]. This language uses .NET platform to write scripts and debugging them. It is non SQL based language like DSL.

All of above related works generally based on distributed file system (GFS, HFS, etc.) and a parallel framework (such as MR, Dryad, etc.). DSL also have same

architecture but this higher level language has a different framework that prevent user from complex distributed processes and writing complex queries. However, this language and its operations just works on specific domains that hold related algorithms for distributed systems.

Conclusion

In this thesis, we present a big data system for telecom industries which offers a new higher level language for domain specific operations. Firstly, the characteristics of big data are described. Then, criterias for determining big data tools are grouped into four different parts: Infrastructure, Programming Models, High Performance Schema Free Databases and Processing & Analyzing. After that, telecom domain big data solution with a DSL concept is explained detailed.

Writing Map Reduce programs is very difficult and complex operation. Because user has to write both processing and analyzing tasks by using Map Reduce algorithms. This is also too much consuming. There are lots of solutions to handle Map Reduce jobs very easily and effectively such as Dremel, Sawzall. However, although effectiveness of processes increases, they are not higher level solutions to write these queries easily. We provide a new higher level language that is called DSL and design an integration framework -DFD- for messaging with domain specific big data platform -Petaminer-. With this solution, user can easily perform predefined operations in telecom domain and get solution with a result viewer. Messaging protocol is designed with JSON which is very basic and common solution all around the world.

There are some limitations in this study. Firstly, DFD framework is not designed completely yet. And also result viewer is not implemented. Only DSL operators, objects and attributes are designed and its tests still continue.

As a further research, DFD framework and result viewer should be implemented. Then new operators and attributes can be defined to increase analyzing power of DSL solution.

References

[1] Warden, P., *Big Data Glossary,* O'Reilly Media Publications, USA, 2011.

[2] Eaton, C., Deroos, D., Deutsch, T., Lapis, G., Zikopoulos, P., *Understanding Big Data*, McGraw-Hill, USA, 2012.

[3] Dean, J., & Ghemawat, S. (2008). *MapReduce: Simplified Data Processing on Large Clusters*. Communications of the ACM, 51(1), 107-113.

[4] Apache Hadoop Project. Web site: http://hadoop.apache.org/

[5] Apache Hive Project. Web site: http://hive.apache.org/

[6] Apache Pig Project. Web site: http://pig.apache.org/

[7] Apache Mahout Project. Web site: http://mahout.apache.org/

[8] Critiques of Big Data execution. Web site: http://en.wikipedia.org/wiki/Big_data#Critiques_of_the_Big_Data_paradigm

[9] Driving Marketing Effectiveness By Managing The Flood Of Big Data. Web site: http://www.ibmbigdatahub.com/infographic/flood-big-data

[10] Lam, C., *Hadoop In Action*, Manning Publications, USA, 2010.

[11] Pig Languge. Web site: http://en.wikipedia.org/wiki/Pig_(programming_tool)

[12] S. Melnik, A. Gubarev, J. J. Long, g. Romer, S. Shivakumar, M. Tolton and T. Vassilakis. *Dremel: Interactive Analysis of Web-Scale Datasets*. PVLDB, 2010.

[13] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. Henry, R. Bradshaw, and N. Weizenbaum. *FlumeJava: Easy, Efficient Data-Parallel Pipelines*. In PLDI, 2010.

[14] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. *Interpreting the Data: Parallel Analysis with Sawzall. Scientific Programming*, 13(4), 2005.

[15] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. *Pig Latin: a Not-so-Foreign Language for Data Processing*. In SIGMOD, 2008.

[16] M. Zaharia, M Chowdhury, M. J. Franklin, S. Shenker, I. Stoica. *Spark: Cluster Computing with Working Sets*. June 2010.

[17] C. Engle, A. Lupher, R. Xin, M. Zaharia, H. Li, S. Shenker, I. Stoica. *Shark: Fast Data Analysis Using Coarse-grained Distributed Memory*. SIGMOD 2012, 2012.

[18] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. *SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. VLDB, 1(2)*, 2008.

[19] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U´ . Erlingsson, P. K. Gunda, and J. Currey. *DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language*. In OSDI, 2008.

Curriculum Vitae

Cüneyt Şenbalcı was born in October 9th, 1987, in Istanbul. He received his BS in Computer Engineering in 2011 from Kadir Has University. From 2011 to 2012, he worked as a freelancer Android Application Developer. Since the beginning of 2012 he is Mobile Technology Services Engineer in Turk Telekom Group which is the leading communication and convergence technology group in Turkey.