Big Data Platform Development with a Domain Specific Language for Telecom Industries

Cüneyt Şenbalcı, Serkan Altuntaş, Zeki Bozkus, Taner Arsan*
Kadir Has University
Computer Engineering Department
Cibali Kampusu 34083 Istanbul, Turkey
* arsan@khas.edu.tr

***Abstract* – This paper introduces a system that offer a special big data analysis platform with Domain Specific Language 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 parts are Domain Specific Language (DSL), Parallel Processing/Analyzing Platform for Big Data and an Integrated Result Viewer. In addition to these main parts, Distributed File Descriptor (DFD) is designed for passing information between these modules and organizing communication. To find out benefits of this domain specific solution, standard framework of big data concept is examined carefully. Big data concept has special infrastructure and tools to perform for data storing, processing, analyzing operations. This infrastructure can be grouped as four different parts, these are infrastructure, programming models, high performance schema free databases, and processing-analyzing. 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.**

***Keywords* – Big Data Analysis, Domain Specific Language, Parallel Processing and Analyzing**

I. INTRODUCTION

Steps of data processing/analyzing and visualization of results can be performed easily without writing complex queries or C, Java codes, by using this platform. This also prevents consuming too much time to perform some basic operations.

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. 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.

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.

Traditional programming models cannot be designed for large datasets. Therefore, Google has 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 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.

When it comes to 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, MongoDB, and CouchDB 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.

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.

Traditional databases, management and analysis tools, algorithms and processes cannot be easily applied these large datasets [8]. This concept has three main characteristics: Volume, Variety, and Velocity.

Volume of data that is stored to analyze is extremely increasing. Today, there are almost 2.7 Zettabytes of data in the digital world. [9].

Also Big Data concept has to deal with its variety. There is not only structured data types but also semi structured and unstructured which is not suitable to store in RDBMS.

978-1-4799-2569-8/13/$31.00 ©2013 IEEE

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.

II. Types of Tools In Big Data Concept

There are some important factors to determine which types of tools are suitable:

1) Infrastructure

Most of the existing wireless bandwidth estimation solutions focus on either probing techniques or cross-layer techniques and require either significant bandwidth resources or protocol modifications. The novel aspects in comparison with other works include the fact that no probing traffic is required and that no modification of Media Access Control (MAC) protocol is needed.

2) 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].

3) High Performance Schema Free Databases

High Performance Schema Free Databases that are called as NoSQL Databases are designed to perform data operations efficiently and quickly.

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. There are lots of processing and analyzing tools such as Hive, Pig, Mahout, R and so on.

III. Higher Level Domain Specific For Big Data Concept

While dealing with big data first job 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 job is finding the best cost / performance ratio.

Petaminer is big data management solution for telecom industries and developed by Elkotek. It consists of four main layers: Collect, Manage, Analyze, and Report. Besides all features it also has a Domain Specific Language Infrastructure, which helps business users to run complex queries while staying in their domain eliminating to learn writing scripts in Java, Pig etc.

When customers of Elkotek 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. Customers of big data platforms can easily analyze on their own.

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

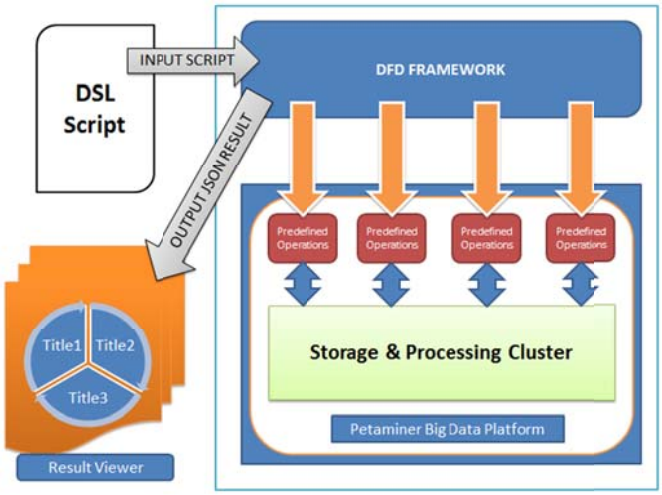

Figure 1: DSL Solution for Big Data Platform

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.

Domain Specific Language – DSL

System works with two kinds of files. These are input files and output files. There are predefined input files such as: *NETWORK_FILE, SUBSCRIBER_FILE, BILLING_FILE, CUSTOMER_COMPLAIN_FILE, COMPETITOR_FILE, PAYMENT_FILE, CROSS_SALES_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: *CROSS_SALE, MOST_PROFITABLE_CUSTOMER, MOST_PROFITABLE_TRAFFIC*

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	Exp.
fileType	“NETWORK_FILE”	Defines input files	-	-
location	HTTP url of file	Location of single input file	HTTP url of file	Single output location
location[]	{loc1, loc2, ...}	Location of multiple input files	{loc1, loc2, ...}	Multi output location
failureCheck	“YES” or “NO”	Need more process power and storage	-	-
filterParams[]	{par1, par2, ...}	Under dev acc. to customer	-	-
getColumns[]	-	-	{col1,col5, col9, ...}	Columns from output
titles[]	-	-	{title1, title2, ...}	Titles acc. column
graphType	-	-	“PieChart”, “LineGraph”, etc.	Graph type for Result Viewer
graphName	-	-	“My Graph”	Graph Name

Table 1:File Descriptor Attributes

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 results and sends them on Result Viewer in JSON format. DFD is an embedded solution like user-defined functions (UDF).

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.

IV. 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.

Processing and Analyzing Performance Results

Performance difference is observed when number of nodes and its map-reduce tasks are changed. 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 2). 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 2: Node Scalability Table

Number of mapper tasks decrease execution time and increase throughput. 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 2).

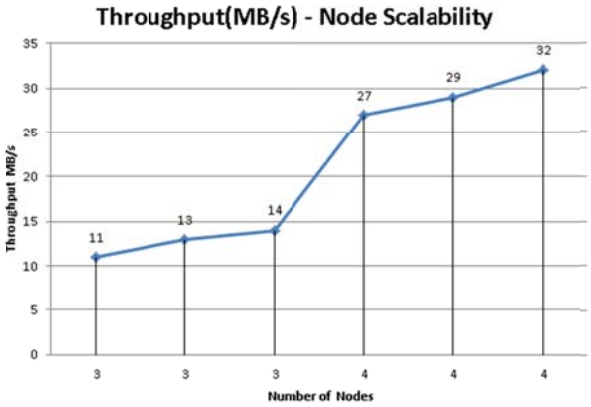

Figure 2: Node Scalability

Another important performance issue while dealing with big data is to decrease execution times. There are two sets: First is execution time result with 2 and 3 mappers on each of three nodes. Second are 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 2: Execution Time Table (n denotes num. of nodes)

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 3). There is an important time difference between while changing total number of nodes 4 up from 3 with fixed number of mapper tasks.

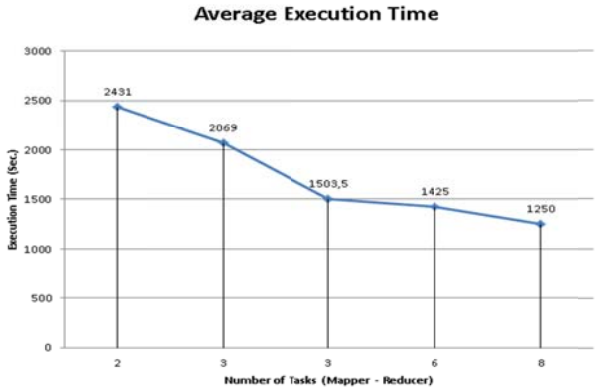

Figure 3: Execution Time

By using Table 2 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 3).

# 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 3: Speedup Table (n denotes number of nodes)

In Figure 4, it is possible to show execution time gain with speedup graph while increasing number of map/reduce tasks and nodes.

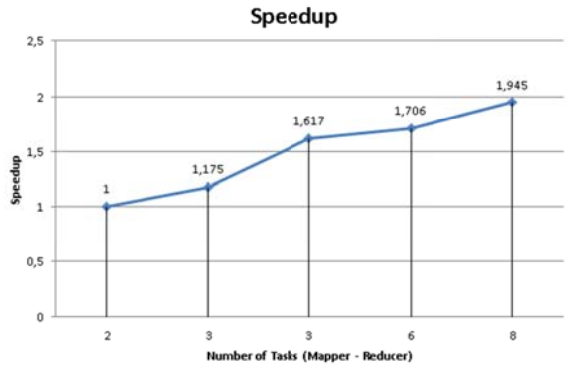

Figure 4: Speedup

HBase NoSQL DB Write Performances

Writing large datasets as quick as possible is important. HBase is designed to perform these kinds of operations efficiently. 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 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 5).

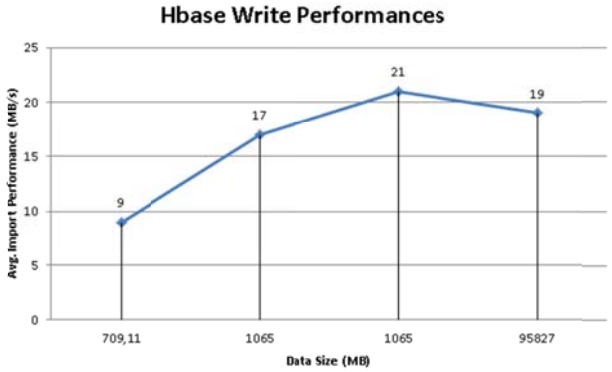

Figure 5: HBase Avg. Import Performance

Write performance of the system is related with data size. lzo compression helps to decrease disk size. In figure 6 shows that disk usage of 1065 MB sample data is compared with its compressed sample.

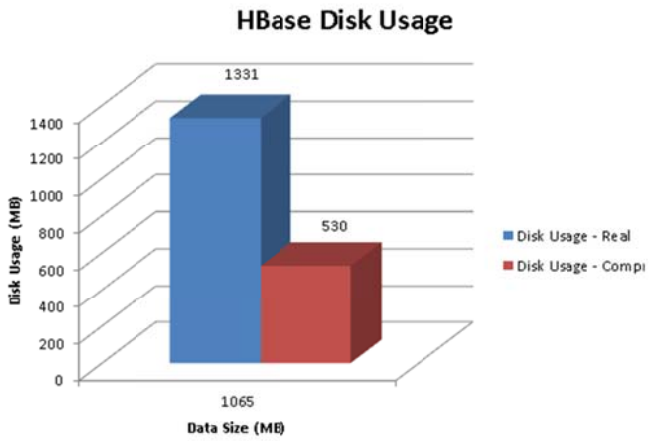

Figure 6: HBase Disk Usage – Sample 1

Disk usage of another sample that has large size (95827 MB) is tested three times with compression (Figure 7).

In Figure 8, disk usage rate of compressed big sample is almost same as first sample in Figure 6. Compressing whole data to decrease disk usage and increase average import rate.

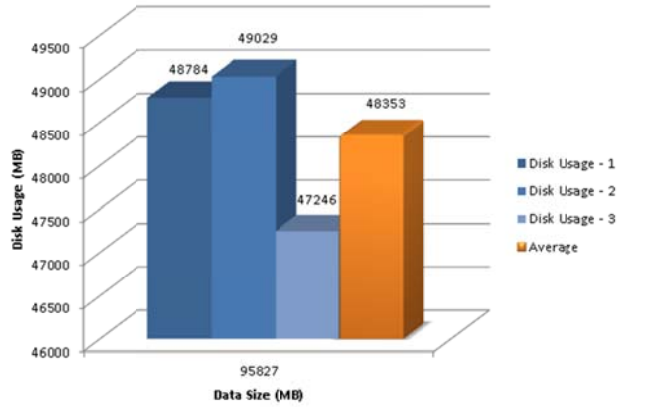

Figure 7: HBase Compression Tests of Sample 2

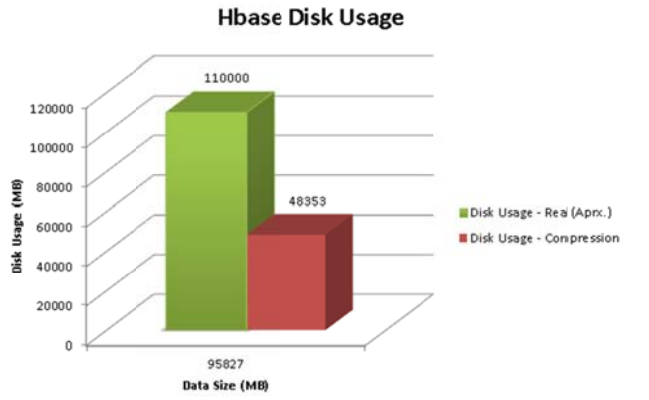

Figure 8: HBase Disk Usage – Sample 2

V. Research Contributions and Conclusions

In this study, we present a big data system for telecom industries, which offers a new higher level language for domain specific operations.

Writing Map Reduce programs is very difficult and complex operation. User has to write both processing and analyzing tasks by using Map Reduce algorithms. There are lots of solutions to handle Map Reduce jobs such as Dremel, Sawzall. 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.

VI. Acknowledgement

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

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] 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.