International Journal of Computer Science and Engineering Open Access
Research Paper Volume-2, Issue-1 E-ISSN: 2347-2693

Analyzing Failures of a Semi-Structured Supercomputer Log File Efficiently by Using PIG on Hadoop

Madhuri Srinivas Palle[1*], Konisa Jyothsna[2] & B. Anusha[3]

[1*]*Computer Science Engineering, National Institute of Technology, Warangal, India, madhuri.palle@yahoo.com*
[2]*Computer Science Engineering, National Institute of Technology, Warangal, India, jyothsna1503@gmail.com*
[3]*Computer Science Engineering, National Institute of Technology, Warangal, India, anushabattula321@gmail.com*

www.ijcaonline.org

Received: 12/12/2013 Revised: 20/12/2013 Accepted: 28/12/2013 Published: 31/Jan/2014

***Abstract*—** Data sets used to fuel the recently popular concept of 'business intelligence' are becoming increasingly large. Conventional database management software is no longer efficient enough however; parallel database management systems and massive data-scale processing systems like MapReduce indeed look promising. Although, MapReduce is a good option, it is difficult to work with, as the programmer would have to think at the mapper and reducer level. In this paper, we present a simple yet efficient way to mine useful information where a program can be written as a series of steps. We have queried a supercomputer log file using Apache's Hadoop and PIG, obtained results as to when and why the supercomputer had failed and compared these results to that of a traditional program.

Keywords- Big Data, Parallel Processing, Hadoop, MapReduce, Data Mining, Business Intelligence, PIG, Log file analysis, Supercomputer

I. INTRODUCTION

Nowadays, business intelligence has become a popular trend. What used to be trial and error has now developed into decisions based on past experiences and other data. The efficacy of mining useful information is ever changing and hence, more and more methods to make mining efficient are being proposed.

In this paper, we will propose a method to analyze the log file of a supercomputer from the bio-informatics background. Log files of this supercomputer are not only vast, but are also semi-structured. The method proposed does not only prove to be more efficient than the ones already existent, but also happens to be much simpler. One of the most efficient solutions to vast log file analysis is using a parallel processing system. However, without using a querying language, the programmer will need to have in-depth knowledge about MapReduce. In many situations, the programmer either does not have such experience or would like a solution in which much training is not required. Also, resultant data is usually required within a short interval as most real-time data worked with has high velocity, which requires programmers to provide instant analysis. We present a solution that allows people to be able to mine information efficiently on a parallel processing system without requiring much training or background knowledge. As a PIG program is nothing but a series of steps, or in other words stepwise instructions, it easy to use, keeping in mind the general population. We have implemented this querying language into our solution for analyzing the failures of a supercomputer.

Corresponding Author: *Madhuri Srinivas Palle*

The objective of the paper is to analyze the failures of a supercomputer by mining its log file in two ways and to compare the results of both the methods. The first way is a traditional program written in Java that will read the file and output the required, relevant information. The second method is to store the files in the HDFS (Hadoop Distributed File System), write a PIG query on MapReduce that will give the semi-structured log some schema to work with and output the required results back to the HDFS. The exact implementation has been explained in the respective section of the paper. By mining both ways, the time taken by both the methods for different sizes of data was compared to see which method is more efficient and feasible. As hypothesized, the method involving the usage of MapReduce and the PIG querying language has proven to be much faster. Statistics have been presented in the results section of the paper.

In the next section, we have written the background relevant to our research. This contains a brief description about MapReduce and PIG. Section III and IV contain the need of the problem statement and the implementation respectfully. Results have been portrayed in the Section V and section VI contains the conclusion and scope for further research. The paper ends with a list of references used for the research of our paper.

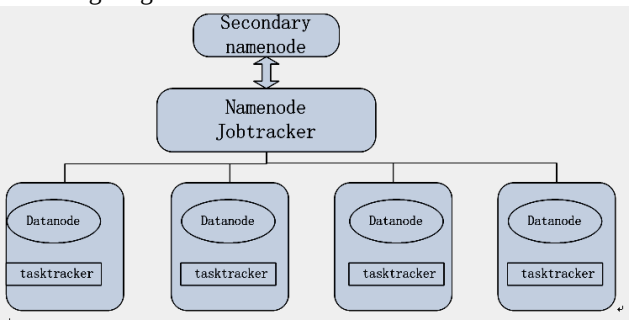
II. BACKGROUND

Big Data, as per the Gartner definition, refers to data that has one or more than one of the following three assets; volume, velocity and variety. The analysis of such data is known to be useful in business intelligence and decision-

making. Over the years, data mining has been a hot trend all around the world. Data analysts mine large amounts of data to spot some kind or reoccurring trends that may just provide a competitive edge.

As mentioned above, data can be of different formats. In order to analyze semi-structured and unstructured data efficiently, we can use Hadoop. Hadoop is an open source software framework. Its architecture can be broadly classified into two parts; the Hadoop Distributed File system (HDFS), which is it's storage mechanism and the Hadoop distributed computational mechanism, which is popularly known as MapReduce.

On a fully configured Hadoop cluster, there are five running daemons; the namenode, the datanode, the jobtracker, the tasktracker and the secondary namenode. The namenode and datanode have a master-slave relationship. The namenode is the master, contains file's metadata and keeps track of which blocks of data go to which node. The datanodes do the grunt work of reading and writing HDFS data blocks to file system. The secondary namenode takes screenshots of the namenode at regular intervals for failure recovery purposes. The jobtracker and tasktracker have relationship similar to that of the namenode and datanode. While the jobtracker monitors all the tasks, the tasktracker is responsible for individual tasks. The overall topology is depicted in the following diagram.



The MapReduce model's data flow contains two main steps; map and reduce. In brief, the Map step consists of the master node diving the data input into sub-problems and distributing these problems to the working nodes. The working nodes can further sub-divide their own problem and distributue it amongst their own working nodes in the form of a multi-level tree. At any iterative level, the working node computes the required output and reports back to it's master. The Reduce step is somewhat like a summary step. It involves the collection of all the sub-problem outputs, combines them and produces the originally required answer. Together, they form the framework of a distributed system.

PIG is an extension of Hadoop which is used to simplify the unnecessary complexity of MapReduce. It contains two main components; a high-level language PIG Latin and a compiler which is usually Hadoop. Complex tasks can be explicitly written as data flow sequences and hence makes it easier to write and maintain any particular program. The user is given the opportunity to concentrate on the semantics of the program rather than optimization as this is automatically done while using PIG.

Pig queries can be written in three ways. The first way is by using an interactive grunt shell. The second way is to write a script file, which is usually for large, repetitive programs and the third way is embedding the queries in a java program. Also, it runs on two different modes, the local and the Hadoop mode. In this paper, we are using PIG in the mapreduce mode.

Ways of Running Pig Latin Script

Method	Description
Grunt interactive shell	Generally used for ad hoc data Manually enter line by line
Script file	Used for large, repetitive pig programs Format: pig myscript.pig
Embedded queries in java program	Pig server class allows any Java program to execute query

A sample query in PIG is given below(fig.2.1). The LOAD command will load the file path as per the defined schema using the delimiter 'space'. Here the data type taken is the default chararray. The alias 'grp' contains the filtered tuples according to the condition specified. In this query, pattern matching was used to check the date of the tuple. The LIMIT operator limits the total number of tuples to 10 for convenience sake and the alias 'cntd' represents these 10 tuples. To see what is happening behind the scenes, PIG provides three diagnostic operators ILLUSTRATE, EXPLAIN and DESCRIBE.

```
grunt> log= LOAD  '/user/mady/project.log'
USING PigStorage(' ') AS
(month,day,time,info1,info2,info3);
grunt> grp= FILTER log BY (day matches '.*30.*');
grunt> cntd= LIMIT grp 10;
grunt> ILLUSTRATE cntd;
```

Fig.2.1

Each separate command can be distinguished by the 'grunt>' shell indication. The screenshot below(fig.2.2) shows how a few of the columns of the schema change by using the ILLUSTRATE command.

log	month:bytearray	day:bytearray	time:bytearray	info1:bytearray
	Apr	30	15:01:43	braf-mon
	Apr	29	18:46:14	braf-mon
	Apr	30	15:01:43	braf-mon
	Apr	30	15:01:43	braf-mon
	Apr	30	15:01:43	braf-mon
	Apr	29	18:46:14	braf-mon

grp	month:bytearray	day:bytearray	time:bytearray	info1:bytearray
	Apr	30	15:01:43	braf-mon
	Apr	30	15:01:43	braf-mon
	Apr	30	15:01:43	braf-mon
	Apr	30	15:01:43	braf-mon

he limit now in use, 1, may have been changed for ILLUSTRATE purpose.

cntd	month:bytearray	day:bytearray	time:bytearray	info1:bytearray
	Apr	30	15:01:43	braf-mon

Fig.2.2

One of the advantages of PIG over SQL is that it has a looser schema approach than that of SQL, which therefore, makes it more suitable for semi-structured and unstructured data.

III. Importance of the Study Problem Statement

Analysis of big data has become of utmost priority today. Specifically, analysis of log files is required in order to gauge and understand the failures of the supercomputer. By doing so, future failures and losses can be prevented. In this paper, we have proposed an efficient solution for analyzing the log files of a supercomputer compared to that of a traditional program. We have also taken into consideration the fact that people attempting to analyze these files may not have much experience and knowledge in the MapReduce domain. The solution proposed in this paper is important as it can be used be a large population, not just technical experts. We have targeted a more diverse population as potential users of this proposed solution. Also, this is an apt solution for not just structured logs, but semi-structured logs as well. Using pig we can compute on distributed environment so how much ever is the size of the input we can compute the results.

IV. Implementation

We have used PIG on Hadoop version 1.0.4. Firstly, we have taken a sample semi-structured supercomputer log file, a part of which is shown below(fig.4.1). In this file, there is information like the date, time, kernel in question and a message. We have taken each piece of information as a column by defining a schema.

```
Apr 30 19:14:49 braf-mon nagios: Warning: Host 'cn041' has no services as
Apr 30 19:14:49 braf-mon nagios: Warning: Host 'cn042' has no services as
Apr 30 19:14:49 braf-mon nagios: Warning: Host 'hn1' has no services asso
Apr 30 19:14:49 braf-mon nagios: Warning: Host 'str1' has no services ass
Apr 30 19:14:49 braf-mon nagios: Warning: Host 'str2' has no services ass
Apr 30 19:14:49 braf-mon nagios: Finished daemonizing... (New PID=14258)
Apr 30 19:14:59 braf-mon nagios: SERVICE ALERT: Firstfloor-Printer;PRINTE
eout from host bio-printerff
Apr 30 19:14:59 braf-mon nagios: HOST ALERT: Firstfloor-Printer;DOWN;SOFT
Apr 30 19:15:00 braf-mon nagios: Caught SIGTERM, shutting down...
Apr 30 19:15:00 braf-mon nagios: Successfully shutdown... (PID=14258)
Apr 30 23:11:59 braf-mon kernel: usb 2-6.4: USB disconnect, address 82
Apr 30 23:12:00 braf-mon kernel: usb 2-6: reset full speed USB device usi
Apr 30 23:12:00 braf-mon kernel: usb 2-6.4: new full speed USB device usi
Apr 30 23:12:00 braf-mon kernel: usb 2-6.4: New USB device found, idVendo
Apr 30 23:12:00 braf-mon kernel: usb 2-6.4: New USB device strings: Mfr=0
Apr 30 23:12:00 braf-mon kernel: usb 2-6.4: Product: Sun USB Keyboard
Apr 30 23:12:00 braf-mon kernel: usb 2-6.4: configuration #1 chosen from
Apr 30 23:12:00 braf-mon kernel: input: Sun USB Keyboard as /devices/pci0
Apr 30 23:12:00 braf-mon kernel: generic-usb 0003:0430:00A2.0051: input,h
.4/input0
```

Fig.4.1

The log file was copied to the HDFS. While writing the query, we searched for all logs in which the supercomputer had failed. The query was executed in the MapReduce mode of PIG and was tested on the interactive grunt shell as well as by saving it as a script file. After MapReduce functions were performed, these logs were stored in a output file in the HDFS for viewing. We had also performed queries for user-defined queries such as which activities were going on at a particular user-defined time or on a user-defined date. A screenshot of the statistics for a sample file are shown below(fig.4.2).

```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt
1.0.4   0.11.1  mady    2013-06-05 01:12:49     2013-06-05 01:14:10

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime
ducetime        Alias   Feature Outputs
job_201306042208_0013   1       0       21      21      21      21
:54310/user/mady/fail_new,

Input(s):
Successfully read 981 records (96700 bytes) from: "/user/mady/projec

Output(s):
Successfully stored 8 records (1024 bytes) in: "hdfs://localhost:543

Counters:
Total records written : 8
Total bytes written : 1024
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_201306042208_0013


2013-06-05 01:14:10,309 [main] INFO  org.apache.pig.backend.hadoop.e
mady@ubuntu:~/pig-0.11.1/bin$
```

Fig.4.2

As seen in the statistics screenshot, various measures have been displayed including the job_id, maps, reduces, maximum map time, minimum map time, average map time, median map time, maximum reduce time, minimum reduce time, average reduce time, median reduce time, aliases and feature outputs. Also, various counters have been displayed such as the total number of records written.

To view the output records of the query, we access the HDFS. Below(fig.4.3) is the output of the sample log file.

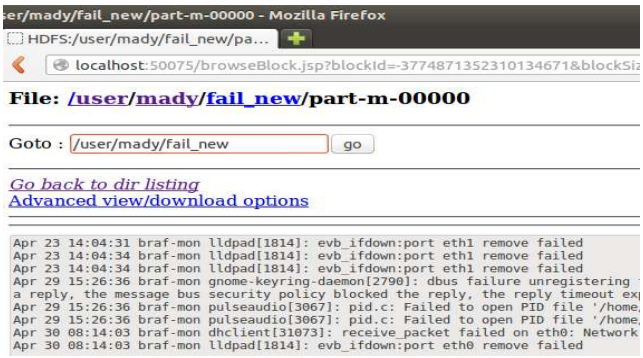


Fig.4.3

In this paper, we have determined the efficiency of using PIG compared to that of a traditional Java program by taking different sized log files and analyzing them both ways. Each time, we doubled the size of the log file in order to get an exact picture as to which method is suitable for which kind of file (i.e. small, medium, large). The results of the performance check are given in the next section of the paper.

V. Result & Analysis

On taking different sized files, we have observed that by using Hadoop and PIG, log files can be analyzed much more efficiently compared to that of a traditional Java program. Not only does the time taken for the file to be analyzed reduce considerably by using this method, but the complexity of programming with this method has also proven to be much more user-friendly. As Hadoop's architecture uses parallel processing, for larger files it is much more feasible. That is, to process larger files, more maps can be used in order to maintain efficiency. The graph shown in the next column (fig.5.1) was made by recording the time taken by different sized files to have their logs processed in milliseconds. This was done for both the methods in our comparison.

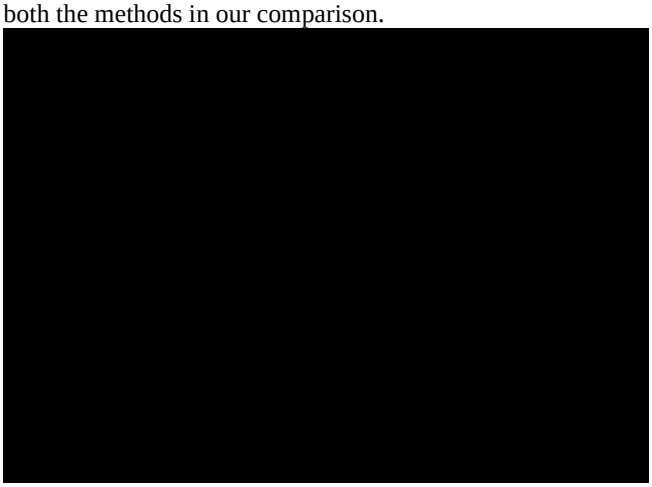


Fig.5.1

Below (table 5.2) are the values taken to draw the graph. In the Hadoop method, a few initial values are unstable. This is probably due to the difference in number of maps allotted, the Hadoop environment and number of resources available. The traditional method shows no such inconsistency.

Size (megabyte)	Traditional method (milli sec)	Hadoop (milli sec)
0.093	49	12
0.18	70	63
0.37	79	49
0.74	106	50
1.5	147	58
2.9	328	102
5.8	506	57
11.6	901	60
23.3	2093	63
46.6	3458	77
93	8126	80
188	18920	133
376	22326	324
752	36206	437
1504	76638	807

Table 5.2

VI. Conclusion

According to experimental results, the method involving using the querying language PIG on Hadoop has proven to be scalable, reliable ,faster,and efficient. One of the major advantages besides efficiency is the fact that it is simple and easy to use, hence targeting a wider audience. The method reliably analyzes the log files, and hence proves that it is useful for structured as well as semi-structured data.

Extensions of this paper could include a solution for analyzing completely unstructured data.

References

[1]. T. White, Hadoop: The Definitive Guide. Yahoo Press,2010.
[2]. Chuck Lam,Pig:Hadoop in Action.
[3]. J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters," Comm. of the ACM,Vol. 51, no. 1, pp. 107–113, 2008.
[4]. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: A Not-So-Foreign Language for Data Processing," Proc. of the 2008 ACM SIGMOD international conferenceon Management of Data, 2008, pp. 1099–1110.
[5]. Thomas Reidemeister, Mohammad Ahmad Munawar, Miao Jiang, Paul A.S.Ward, "Diagnosis of Recurrent

Faults using Log Files," Proc. of the 2009 Conference of the Center for Advanced Studies on Collaborative Research,November 2009, pp. 12-23 .

[6]. Apache. Hadoop: Open-source implementation of MapReduce. http://hadoop.apache.org.

[7]. Apache. Pig: High-level data ow system for Hadoop. http://www.pig.apache.org

[8]. Michael Cardosa, Chenyu Wang, Anshuman Nangia, Abhishek Chandra, Jon Weissman,"Exploring MapReduce efficiency with highly-distributed data" Proc. of the second international workshop on MapReduce and its applications",June 2011, pp. 27-34.

[9]. H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reducemerge: simplified relational data processing on large clusters,"proc. of the SIGMOD Conference, 2007, pp. 1029–1040.

[10]. A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava., "Building a High-Level Dataflow System on Top of Map-Reduce: The Pig Experience." Proc. of the VLDB Endowment, vol. 2,no. 2, 2009.

[11]. A tutorial on pig: http://www.pig-tutorial.blogspot.in/

