Telecom Data processing and analysis based on Hadoop

Guofan Lu, Qingnian Zhang*, Zhao Chen

Wuhan University of Technology, Wuhan 430063,China

Received 1 October 2014, wwwcmnt.lv

Abstract

For call tracking system to adapt to the needs of large data processing, combined with a strong competitive advantage in recent years in large data processing Hadoop platform, designed and implemented a Hadoop-based call tracking data processing model, in order to verify its feasibility. The call tracking processing system model contains an analog data source module, data processing module, and a GUI interface. Analog data source module from real data samples in the simulated data, and the data is written directly to the Hadoop distributed file system, then using Hadoop's MapReduce model to write appropriate Mapper and Reducer function, the distributed processing of the data. Detailed study based on the system design and implementation, system deployment topology, hardware and software conditions, and designed several comparative experiments to analyze some static indicators of system performance.

*Keywords: h*adoop, MapReduce model, data processing, call tracking

1 Introduction

With the increasingly sophisticated and popularity of the third-generation communication networks, the size of the network is drastically becoming larger and the equipment complexity is also greatly improved. So how to quickly resolve the call failure and how to ensure the quality of network operation becomes the most urgent task for network maintenance personnel. Call trace system, a new feature derived from signaling trace, as an important subsystem of the network management system, is a powerful mean to solve call failure problem and guarantee the quality of the network operation [1, 2].

This paper deals with the demand of large amounts of call tracking data. In the paper, data and environment are analyzed to find their impact on system performance, combined with Hadoop platform, which is more used recently in the field of large-scale data processing. A call tracking simulation system based on Hadoop was designed to simulate real application scenarios. So the analysis research of algorithm and the assessment of performance can be done.

Hadoop is currently the most used distributed computing platform. The core part of the distributed storage system consists of HDFS and the distributed computing framework MapReduce. HDFS creates multiple copies of data blocks of replication to guarantee reliability. And put data blocks in the cluster's compute nodes, and then the application program will be broken down into many small tasks to realize the parallel processing. It processes the data block on its node where the data blocks are, using moving computing to substitute for data moving, greatly improving the efficiency. Hadoop is designed as a framework which can use simple programming model realizing distributed processing of large data sets with good scalability and high fault tolerance [3]. Streaming data access pattern is used to access the storage file system. This way is suitable for processing large amount of data application. The relationship between HDFS and MapReduce is shown in Figure 1.

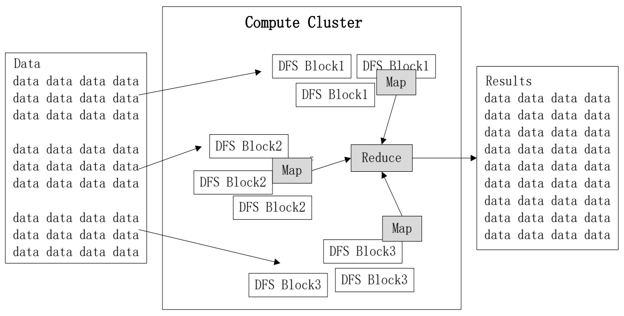

FIGURE 1 HDFS and MapReduce relationship Diagram

MapReduce framework adopts a master-slave structure, which is composed of one JobTracker and several TaskTrackers. JobTracker is the master node, responsible for scheduling the jobs that submitted. Each job is divided into a number of tasks by JobTracker. Then JobTracker assigns tasks to TaskTrackers in the cluster. In addition, JobTracker will also be responsible for monitoring the implementation of the tasks and timely feedback to the user. TaskTracker is the slave node, which keeps in communication with the JobTracker through the heartbeat mechanism. It informs JobTracker its own state and executes the tasks that assigned by JobTracker [4].

The entire implementation process of MapReduce should include the following four parts: the submission of a job, assignment and execution of Map tasks, assignment and execution of Reduce tasks and the completion of the job. User configures the relevant configuration files before submitting the job. As soon as the job is submitted the user could no longer intervene, since it enters an automatic process. In the configuration it should ensure that there is

* ***Corresponding author's*** e-mail: 616024597@qq.com

at least the execution Map and Reduce code, which are finished by the user according to their needs. The core parts of MapReduce are Map operations and Reduce operations.

In Map process data process is in parallel, which is to say the input data should be separated into several parts. The main function of Map is to convert the input data into key/value pairs [5,6]. However, Reduce puts the separated data generated by Map together through sorting and merging. Then the new key/value pairs will be generated. In the user's perspective, the designation of the Map and Reduce function is associated with the type, just as shown in Equation (1) and (2) below.

$$Map(k1, v1) \rightarrow list(k2, v2)\,, \tag{1}$$

$$Reduce(k2, list(v2)) \rightarrow list(v2)\,. \tag{2}$$

This association is achieved in the internal MapReduce programming model. The framework hides the complexity of the procedure. So for users, there is no difference between the distributed computing model system of MapReduce and the serial computing algorithm we usually use. In actual process, the input data is divided into several blocks. Then the blocks are randomly assigned to the data nodes. When a job is submitted, JobTracker will distribute Map and Reduce tasks to TaskTrackers according to TaskTrackers' status. Generally, one data block has a corresponding Map task. Input data is processed by Map task using the user-defined Map function and converted into key/value pairs. Then the frame will get the intermediate results associated with each Reduce from the output of Map through HTTP, This section is called Copy section. There will be a sort section before Reduce task. In Sort section the intermediate results get by various Reduce will be grouped according to their keys. And then for each unique key, it produces the final result according to user-defined Reduce function. Generally, it does sorts while doing copy. The specific implementation process of MapReduce is shown in Figure 2.

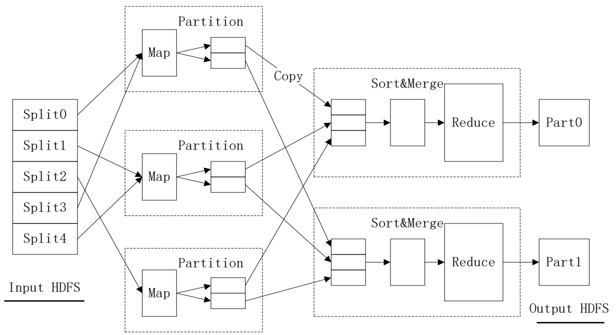

FIGURE 2 Implementation Process of MapReduce

2 Design and analysis of the system

In the real communication network, RNC will generate a large amount of call trace data. The source packets of these data will be generated at the specified path on the specified server. The data processing module will process these data and do further analysis for the results. In this paper, the process flowchart of call trace in the network management system is shown as Figure 3.

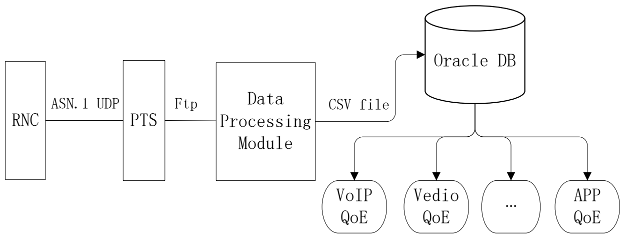

FIGURE 3 Flowchart of call trace

RNC (Radio Network Controller) will generate UDP data packets in ASN.1 format. PTS (Package Transferring System) receives the data packets which are generated in RNC and then the call trace with ASN.1 coded format is wrote into one file according to specified time duration. This file is the source packet which will be handled by the data processing module. Then the data processing module will get the data packet through FTP service and generate the result into a CSV format file after processing. The results will be upload to CT (Call Trace) server and be stored in the database for further analysis and application or be called by other modules in the network management system.

2.1 OVERVIEW

The cluster scale of Hadoop is generally to be larger in the real application. But the hardware environment used in this chapter is just a DL380 Medium server and a DL380 Small server as our main purpose is only to verify the availability of call trace's data processing. The configuration of these two servers is shown in Table 1.

TABLE 1 Experimental hardware configuration

	DL380 Medium	**DL380 Small**
CPU	2.93G 2*8core	2.93G 1*8core
Memory	4*18G	2*18G
Disk	HDD 12*300G 10000rpm	HDD 6*300G 10000rpm
Network	5GB interface	5GB interface

Two experimental environments (pseudo-distributed and small cluster) is designed using our Existing hardware conditions mentioned above to analysis how the configuration of the hardware affects the system performance.

DL380 Medium server is chose for pseudo-distributed environment as the master node and the slaver node will be configured on one machine in pseudo-distributed environment. The network topology of the pseudo-distributed environment is shown as Figure 4. The DL380 is both the name node and the data node, which means it plays the role of JobTracker also the role of TaskTracker at the same time. DL380 Medium will have five daemons after starting Hadoop, which are JobTracker, SecondaryNameNode, NameNode, DataNode and TaskTracker respectively.

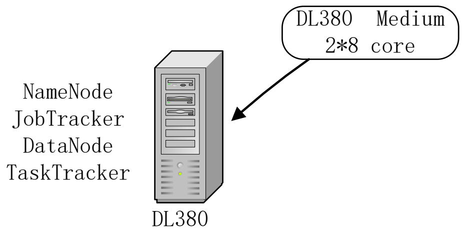

FIGURE 4 Network topology of pseudo-distributed environment

Three configuration files are needed under hadoop-0.21.0/conf/ on the system, which are core-site.xml, hdfs-site.xml and mapred-site.xml, respectively. The detail of the configuration is shown as Table 2.

TABLE 2 configuration table of pseudo-distributed system

Configuration file	Configuration content
core-site.xml	<name>fs.defalut.name</name> <value>hdfs://localhost:9000</value>
hdfs-site.xml	<name>dfs.replication</name> <value>1</value>
mapred-site.xml	<name>Mapred.job.tracker</name> <value>hdfs://localhost:9001</value>

Analysis of System Performance The procedure of data process is split into four sections and we use T_i, T_m, T_r, T_o representing the input data time, Map time, Reduce time and the output time. The calculation of the total processing time is shown in Equation (3).

$$T_{Total} = T_i + T_m + T_r + T_o . \quad (3)$$

In the following, three sets of comparative experiments are used to analyze the time of each section. All experiments will be made three times and then take the average value.

2.2 COMPARATIVE EXPERIMENTS OF FULL MR DATA VS NON-FULL MR DATA

In this part, a group of non-compressed data and pseudo-distributed test environment are used. The specific information of experimental data is shown in Table 3.

For the experimental results, the average time spent of various sections and the size of output file is collected. The specific experimental results data is shown in Table 4.

TABLE 3 Full MR Vs Non-Full MR Experimental Data

Data Type	Number of Information	Number of MR Information	File Size
Non-Full MR Data	6,262,800	349,943	703M
Full MR Data	6,262,800	6,262,800	613M

TABLE 4 Full MR Vs Non-Full MR Experimental Result

Data Type	Input file Time	Map Time	Reduce Time	Output file Time	All time	Output file size (M)
Non-Full MR	7s	16s	9s	5s	37s	60M
Full MR	3.7s	42s	72.3s	67s	185s	1057M

It Can be seen from Table 3 the size of the two data packets is similar and containing the same number (6,262,800 pieces) of information. In the non-full MR data package the MR data ratio is 5.6%. So in the actual data process, the ratio of the number of MR information processed using full MR data packet and non-MR packet is 17.9: 1.

2.3 COMPARATIVE EXPERIMENTS OF PSEUDO-DISTRIBUTED VS SMALL CLUSTERS

Different hardware environments inevitably have an impact on the performance of system. In theory, the cluster system has more hardware resources, the performance should be better than the pseudo-distributed test environment. The following set of experiments is designed to verify this. Four sets of Non- Compressed and Non-Full MR experimental data are prepared. The specific experimental data information is shown in Table 5.

TABLE 5 Pseudo-distributed Vs small clusters experimental data information

No. of experiment	1	2	3	4
Number of all the information	6,262,800	12,525,600	25,040,400	50,080,800
Number of MR information	349,943	699,886	1,399,163	2,798,347
File Size	703M	1.4G	2.8G	5.6G

Set the same size of data block and start the same number of Reduce tasks in DL380 pseudo-distributed and DL380 small cluster environment and collect the statistical data of two experimental environments. The experimental result of input section is shown in Figure 5. In the pseudo-distributed environment, all the data is copied to the local disk. But in the small cluster, the two servers use gigabit Ethernet and since the amount of data to be copied is not huge, so the time of input section is almost the same in the two experimental environments.

The results of Map section are shown in Figure 6 and Figure 7. The total number of CPU cores in DL380 small cluster is 24, pseudo-distributed 16. So the CPU core number of DL380 small cluster is 50% (8 core) more than the pseudo-distributed test environment. In Figure 6, it can see that the Map section time of small cluster using DL380 is only about 70% of the pseudo-distributed environment. Map efficiency is almost about 1.4 times higher than pseudo-distribute. That is to say, when processing the same data, small clusters can reduce the time of map section and improve the map efficiency compared to pseudo-distributed environment.

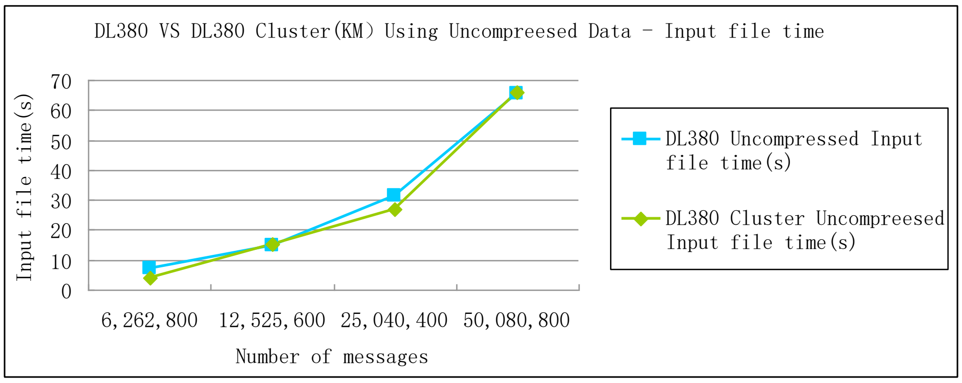

FIGURE 5 Time comparison of input section

The results of Map section are shown in Figure 6 and Figure 7. The total number of CPU cores in DL380 small cluster is 24, pseudo-distributed 16. So the CPU core number of DL380 small cluster is 50% (8 core) more than the pseudo-distributed test environment. In Figure 6, it can see that the Map section time of small cluster using DL380 is only about 70% of the pseudo-distributed environment. Map efficiency is almost about 1.4 times higher than pseudo-distribute. That is to say, when processing the same data, small clusters can reduce the time of map section and improve the map efficiency compared to pseudo-distributed environment.

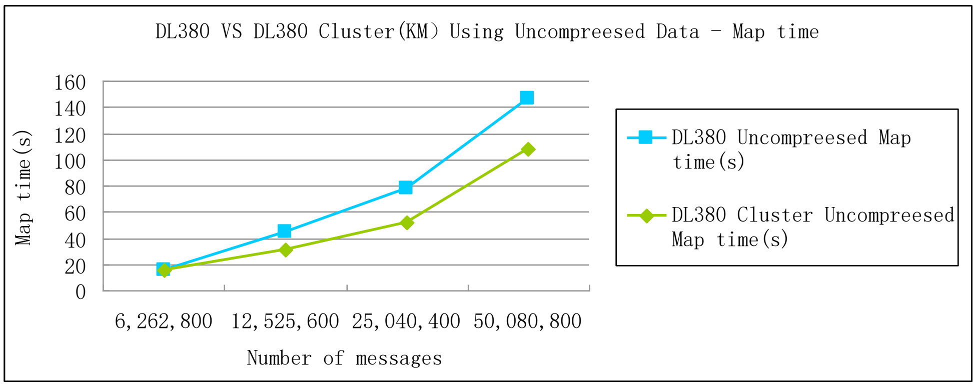

FIGURE 6 Time comparison of map section

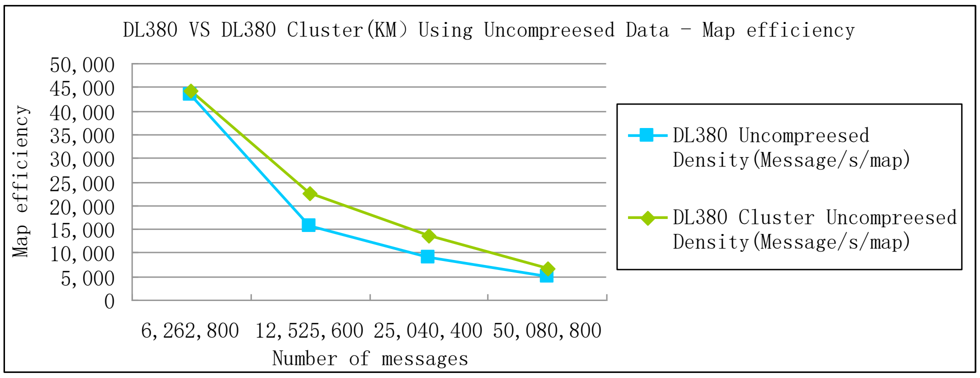

FIGURE 7 Comparison of map's efficiency

The results of Reduce section are shown in Figure 8. Since the input file size and the data block size are the same, so the number of Map tasks started is also the same. During the Reduce section, it gets Maps' intermediate results from the same number of Map tasks in the two environments. Moreover, the small cluster uses Gigabit Ethernet. So the Reduce time is almost the same in the two different environments.

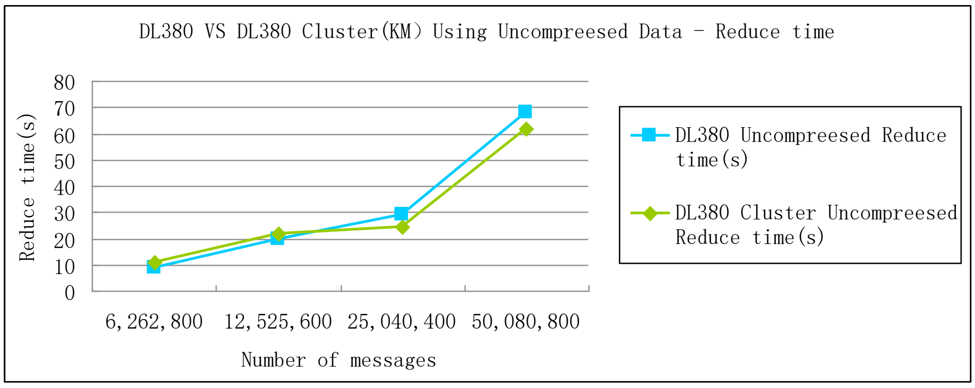

FIGURE 8 Time comparison of reduce section

The results of Output file section are shown in Figure 9. Since the size and content of produced result is the same, the time of output file section is almost the same.

The comparison analysis of the total time to run the entire job is shown in Figure 10. Time spent in Input file section, Reduce section and Output files section is almost the same in pseudo-distributed environment and small clusters according to the analysis of the previous four sections. However, the small clusters uses less Map section time than pseudo-distributed. Therefore, small clusters will spend less time than pseudo-distributed while processing the same job. In summary, small cluster using Gigabit Ethernet has better performance than pseudo-distributed environment.

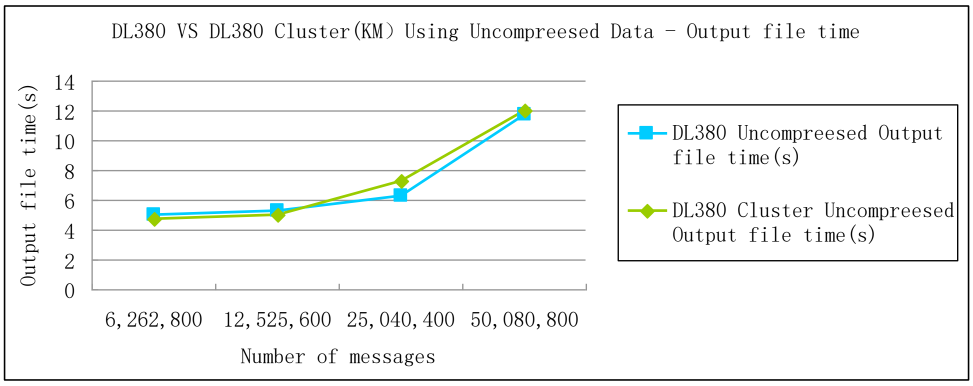

FIGURE 9 Time comparison of output section

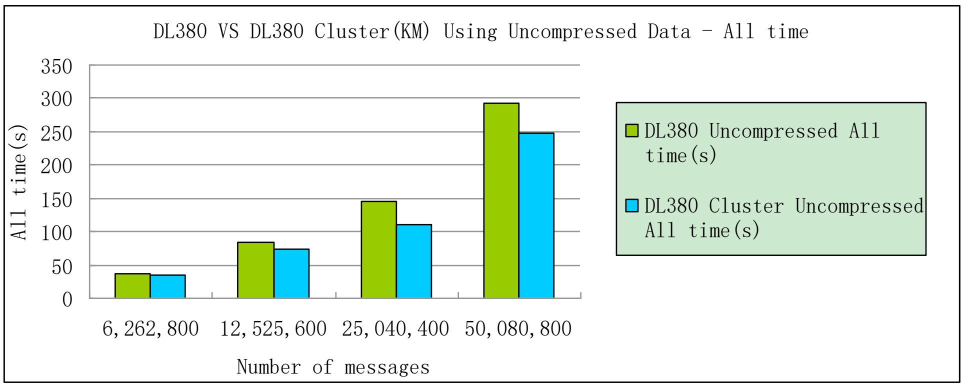

FIGURE 10 Time comparison of all sections

2.4 COMPARATIVE EXPERIMENTS OF COMPRESSED AND UNCOMPRESSED DATA

These sets of experiments were carried out on DL380 small clusters. Four sets of experimental data which contains different number of information and includes compressed and uncompressed Non-Full MR were prepared. The specific information of the experimental data is shown in Table 6.

TABLE 6 Experimental data information of compressed and uncompressed

No. of experiment		1	2	3	4
Number of all the information		6,262,800	12,525,600	25,040,400	50,080,800
Number of MR information		349,943	699,886	1,399,163	2,798,347
File size	**Uncompressed**	703M	1.4G	2.8G	5.6G
	Compression	137M	275M	549M	1.1G

In this part, four sets of experimental data were processed in small DL380 cluster environment. The statistic results were collected.

The results of Input section are shown in Figure 11. Time used writing file to HDFS is reduced while using compressed data since the size of compressed packet data is much smaller than the uncompressed. It can be calculated from Table 6 that the compression rate is about 20%, and the writing time after compressed is just about 20% of the uncompressed one. Therefore, compression can reduce the time of Input section and improve the efficiency of Writing file section.

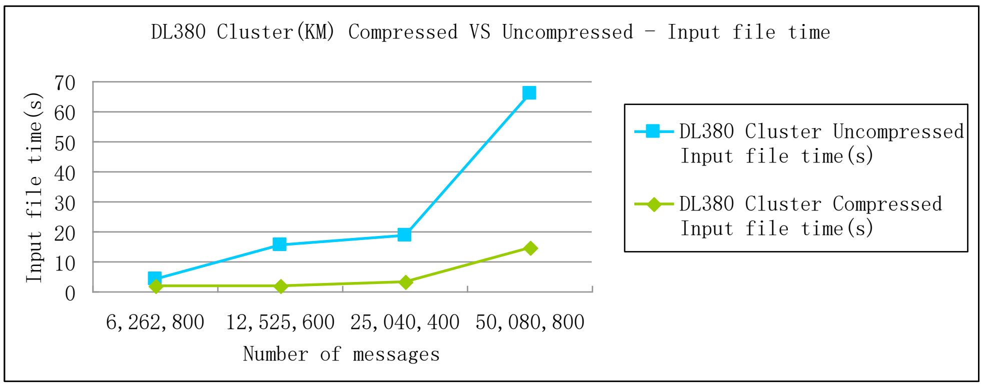

FIGURE 11 Time comparison of input section

The results of Map section are shown in Figure 12 and 13. It will start less Map tasks since for compressed file size of data containing same information is much smaller than the uncompressed. It can be seen that using compressed data can reduce the Map time (Except the first data point, since data size there is small and the compressed data need time to uncompress itself. So the advantages of compressed data in Map section won't be reflected). So the compressed input data could improve the Map efficiency. It can be seen from Figure 12, as the input data size increases, The Map efficiency reduces no matter using compressed or uncompressed data since the input data size exceeds the capacity of nodes.

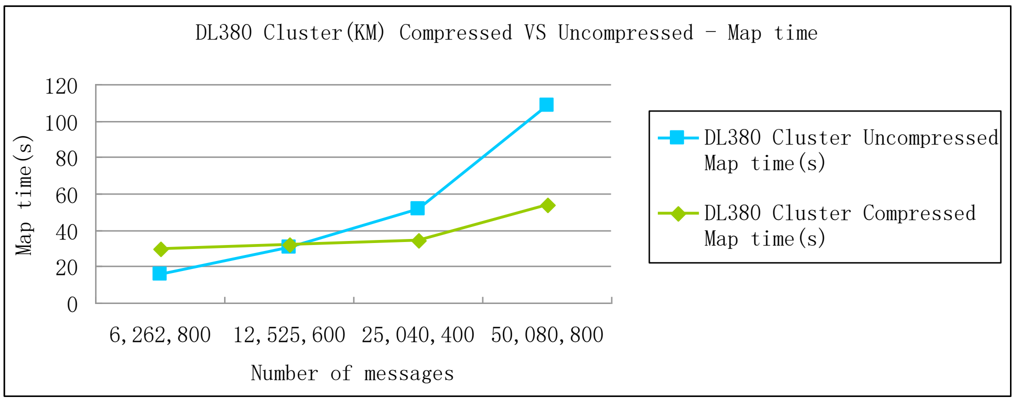

FIGURE 12 Time comparison of nap section

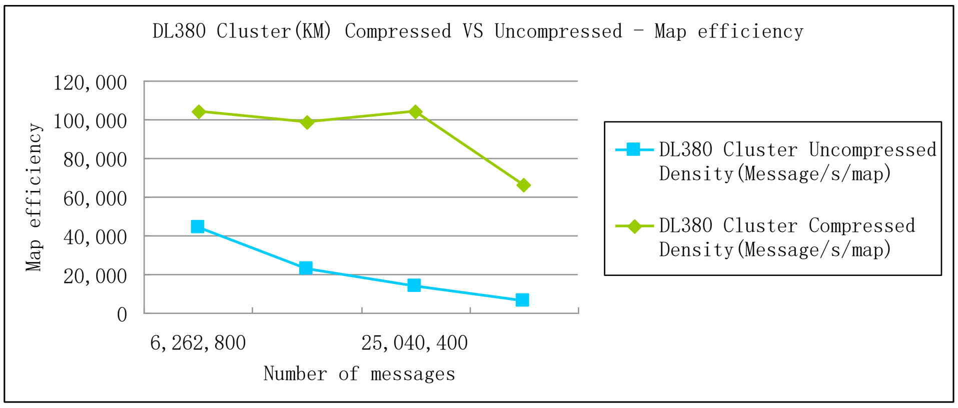

FIGURE 13 Comparison of map's efficiency

3 Summary

This paper describes the deployment of call trace system. One pseudo-distributed environment and one small cluster environment was implemented based on the provided experimental hardware. Job running process in Hadoop and operation process of call trace system was studied. According to the characteristics of the system three sets of experiments were designed to study how the test environment and the different data types impact the system performance. Finally, the experimental results were analyzed and draw the conclusion that compressed data and small clusters can improve the system performance compared to uncompressed data and pseudo-distributed environment.

Acknowledgements

The authors are grateful for the support from the National Science Foundation of China (51479159) and the Soft Science Project of China's Ministry of Transport (2013-322-811-470).

References

[1] Tian C, Zhou H, He Y, Zha L 2009 A Dynamic MapReduce Scheduler for Heterogeneous Workloads [C] *Eighth International Conference on Grid and Cooperative Computing* 218-24
[2] Lu P, Lee Y C, Wang C, Zhou B B, Chen J, Zomaya A Y 2012 Workload Characteristic Oriented Scheduler for MapReduce [C] *IEEE 18th International Conference on Parallel and Distributed Systems* 156-63
[3] Nguyen P, Simon T, Halem M, Chapman D, Le Q 2012 A Hybrid Scheduling Algorithm for Data Intensive Workloads in a MapReduce Environment [C] *IEEE/ACM Fifth International Conference on Utility and Cloud Computing* 161-7
[4] Liu L, Zhou Y, Liu M, Xu G, Chen X, Fan D, Wang Q 2012 Preemptive Hadoop Jobs Scheduling under a Deadline [C] *Eighth International Conference on Semantics, Knowledge and Grids* 72-9
[5] Kamal Kc, Kemafor Anyanwu 2010 Scheduling Hadoop Jobs to Meet Deadlines [C] *IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)* 388-92
[6] Yang S-J, Chen Y-R, Hsien Y-M 2012 Design Dynamic Data Allocation Scheduler to Improve MapReduce Performance in Heterogeneous Clouds [C] *Nineth IEEE International Conference on e-Business Engineering* 265-70
[7] Matei Z, Dhruba B, Joydeep Sen S, Khaled E, Scott S, Ion S 2010 Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling [C] *5th European conference on Computer systems* 265-78

Authors

	Lu Guofeng, was born in 1966, Jilin, China **Current position, grades:** PhD candidate in Transportation planning and management in Wuhan University of Technology, Wuhan, China. **University studies:** BSc in Highway and Bridge from Harbin University of Civil Engineering and Architecture, China, in 1993, MSc degree in traffic management from Jilin University in 2012. **Scientific interest:** traffic and transportation planning and transportation safety management etc.
	Zhang Qingnian was born in 1957, Xi'an, China. **Current position, grades:** professor of School of Transportation at Wuhan University of Technology. **University studies:** PhD degree in Machinery design and theories from the Wuhan University of Technology, China, in 2002. **Scientific interest:** traffic and transportation planning, optimization and decision making of transportation system, transportation safety management.
	Chen Zhao was born in 1969, Shaanxi Province, China. **Current position, grades:** PhD candidate in Logistics management at Wuhan University of Technology. **University studies:** BSc in Chinese language and Literature from China people's Police University, Beijing, China in 1992, MSc in Transportation management engineering from Wuhan University of Technology, China, in 2002 **Scientific interest:** parallel computing on Hadoop platform, optimization decision.