Correlation Aware Technique for SQL to NoSQL Transformation

Jen-Chun Hsu, Ching-Hsien Hsu
Dept. Computer Science & Information Engineering
Chung Hua University
Hsinchu, Taiwan
cklkmnbvc@gamil.com, chh@chu.edu.tw

Shih-Chang Chen, Yeh-Ching Chung
Dept. Computer Science
National Tsing Hua University
Hsinchu, Taiwan
scc@sclab.csie.chu.edu.tw, ychung@cs.nthu.edu.tw

***Abstract* - For better efficiency of parallel and distributed computing, Apache Hadoop distributes the imported data randomly on data nodes. This mechanism provides some advantages for general data analysis. With the same concept Apache Sqoop separates each table into four parts and randomly distributes them on data nodes. However, there is still a database performance concern with this data placement mechanism. This paper proposes a Correlation Aware method on Sqoop (CA_Sqoop) to improve the data placement. By gathering related data as closer as it could be to reduce the data transformation cost on the network and improve the performance in terms of database usage. The CA_Sqoop also considers the table correlation and size for better data locality and query efficiency. Simulation results show that data locality of CA_Sqoop is two times better than that of original Apache Sqoop.**

Keywords - NoSQL; Cloud computing; Big Data; Data locality; Sqoop

I. INTRODUCTION

With the improvement and advancement of internet technologies, people use the internet more and more in daily life. Thus, huge amount of data is generated with various formats. The ways to store data, analyze data and find useful information become important issues. Cloud Computing technologies are developed for those issues, e.g., Google File System (GFS) [6], MapReduce framework [5] and BigTable [4] proposed by Google in 2003, 2004 and 2006 for distributed file system, parallel and distributed computing, and NoSQL database, respectively.

In 1984, John Gage created the phrase "The Network is the computer" which described the concept of Cloud Computing today. However, the internet technologies were not good enough to realize this idea. The phrase "Cloud Computing" is not a technology but a concept of establishing a server cluster [3] in Cloud by virtualization technologies and processing the large amount of data stored in Cloud through the internet. Google proposed the MapReduce framework to split data into small pieces and execute the related jobs on nodes. The results will be collected from nodes, integrated and then return back to users. In this way, MapReduce transforms a single-node processing job to a parallel processing job to improve the execution efficiency.

Although GFS, MapReduce and BigTable were published, the source code was not released. After the community developed Hadoop [8], enterprises and programmers have a platform similar to GFS and MapReduce framework to develop MapReduce related technologies [7].

Most enterprises still use relational database (RDB) for business. However, as more and more data produced, RDB lacks the ability to handle such size of data. Using Cloud database is a possible solution and enterprises need a tool to migrate data from RDB to Cloud database for performance in terms of database usage.

Apache Sqoop was developed for data migration from RDB to NoSQL database. Sqoop becomes the top level project in 2012 which enable users to migrate large size of data to Cloud environment and to be accessed by Cloud technologies.

Different from the traditional means, files are split and distributed in different Virtual Machines (VMs). However, there will be an issue if a JOIN operation is performed and its data of tables is distributed and stored on different VMs.

Sqoop splits each table into four parts by default and use the Mapper of MapReduce framework to store data in the cluster via JDBC driver during data migration. Data of tables is then stored in the VMs where Hadoop executes the Mapper randomly. The data is therefore distributed in the VM cluster.

Due to execute Mapper randomly in VMs, the bad location of data in the cluster may increase the query time because some data has to be transformed on network.

This paper addresses above issues and enhances data locality by analyzing the log of queries. RDB logs everything including the queries which are submitted to access database. By analyzing the log, tables which are frequently used can be found and stored as closer as possible in the VM cluster to avoid possible data transformation and improve the overall performance.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces how Apache Sqoop works. In section 4, we present the proposed method, CA_Sqoop, to improve the data locality. Section 5 gives the performance evaluation. Finally, section 6 concludes this paper.

II. RELATED WORK

978-1-4799-4266-4/14 $31.00 © 2014 IEEE
DOI 10.1109/U-MEDIA.2014.27

With the coming of digital age, enterprises have to keep up with data growth explosion. RDB lacks the ability to handle such amount of data for real-time system, e.g., telecom billing systems. A research [10] compared MapReduce and SQL on large scale data processing. The result shows RDB had better performance with small data, but MapReduce performed better while size of data increased. LoadAtomizer [1] presented algorithms to solve the scheduling and load balancing problems.

JOIN is a frequently used database operation. Many researches had proposed solutions to improve the performance of JOIN [11, 12]. Hive [16] provides SQL-like query language, HiveQL, to access data in NoSQL storage, e.g., HBase. However, the MapReduce jobs of JOIN operations generated by Hive are not efficient.

Hive generates MapReduce jobs for queries one by one. Thus, the relation between queries is not considered. YSmart [15] addressed this issue and tried to merge operations according to the relation between queries. YSmart successfully avoided unnecessary JOIN operation, improved the query time and was integrated to Hive in 2012.

NoSQL databases were designed to process unstructured data [13, 14], but enterprises want to leverage the ability of NoSQL to process structure data. Clydesdale [9] proposed a framework on Hadoop for above issue without modifying the complicated architecture of Hadoop but presenting several techniques to speed up the operation of processing structured data.

III. IMPORT DATA TO NOSQL

In the age of information explosion, anything can be digitalized, e.g. books and figures. As time goes by, the size of digital data grows and is hard to know how large it actually is. IDC says the total data size of Digital universe is 0.18 ZB in 2006. RDB is obviously not able to processing such kind of data in terms of size. However, Cloud Computing is designed for the Big Data including data storing, processing and analyzing.

Hadoop, an Apache top level open source project, provides a distributed system architecture and is mainly consisted of Hadoop Distributed File System (HDFS) and MapReduce. HDFS was developed based on the concept of GFS to connect nodes together and form a large scale of distributed file system. HDFS was designed to process large amount of data and provide safe storage architecture to avoid hardware failure. It was also designed for write-once-read-many file access model, which provides simple consistency mechanism and increase the throughput of data transformation. HDFS duplicates the data on different data nodes to make sure users can still access their data if any data node crashes.

Fig. 1 shows Apache Sqoop system architecture. Sqoop focuses on migrating large scale of data between RDB and Hadoop. Through Sqoop, users can migrate data to HDFS or HBase in command line mode with ease. Sqoop became an Apache top level project in March, 2012.

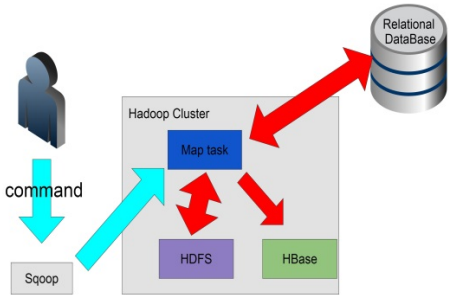

Figure 1. Apache Sqoop system architecture

With the rapid growth of data, processing and analyzing data with RDB, e.g., MySQL, becomes inefficient. MapReduce can be an alternative to improve the performance after Sqoop migrates data to Hadoop.

Apache Sqoop splits a table into four parts and migrate them to HDFS or HBase through JDBC by Mapper. However, the nodes for executing Mappers is randomly decided by Hadoop. Data is therefore stored on random data nodes which results in a bad data locality.

Queries are used to access databases and shown the data with different meaning. JOIN is a frequently used operation which merges two table according to specific columns. If the data of two tables is distributed on eight data nodes, data transformation on network cannot be avoided to perform the JOIN operation. The speed of data accessing on network is obviously slower than that in local disks. Therefore, data locality should be increased to avoid data transformation on network and enhance the performance of JOIN operation. Section 4 proposes an algorithm to allocate data to nodes according necessary information to address this issue.

IV. CORRELATION AWARE TECHNIQUE

Relational database usually has a log file to store operations including configuration, modification and query. The log file is usually used to monitor the database. In addition, the logged operations can be treated as the queries accessing to the database. Thus, once the log file is analyzed, we can know the tables which are frequently use by queries.

Table I demonstrates a matrix showing degree of Table Correlation (TC) by analyzing queries in the log file. Taking Table 1 and Table 2 as an example, the degree is 15 which means there were 15 queries accessing these two table simultaneously according to history records. Similarly, there were 45 queries accessing Table 4 and table 5 according to the log.

In addition to the degree of table correlation, the size

of tables can also be considered. In some cases, it is not necessary to put two relative small tables together even though the degree of table correlation is high. Hadoop assigns a task to one of the nodes, which owns more data to reduce the cost data transformation. Such concept is also applied in this work to improve the performance according to table sizes.

TABLE I. Example of table correlation

Table Correlation (TC)	Table1	Table2	Table3	Table4	Table5
Table1		15	25	63	21
Table2			82	24	34
Table3				72	12
Table4					45
Table5					

V. PERFORMANCE ANALYSIS

The simulation results of Sqoop and the proposed method, CA_Sqoop, are given in this section to show the performance while considering table placement. In the simulation, the effect of replication is not considered. The ranges of parameters in the simulation are given as follows.

- Table Size : 1~10 GB
- Table Correlation : 1~1000
- Node Capacity : 2~50
- # of Tables : 20~300
- # of Nodes : 60~100

Figures 2 and 3 are the results of with different number of node capacity while importing tables to clusters. Fig. 2 gives the improvement on data locality with a small cluster while Fig. 3 presents the results with a large cluster. CA_Sqoop overcomes Sqoop with better data locality even the node capacity is increased.

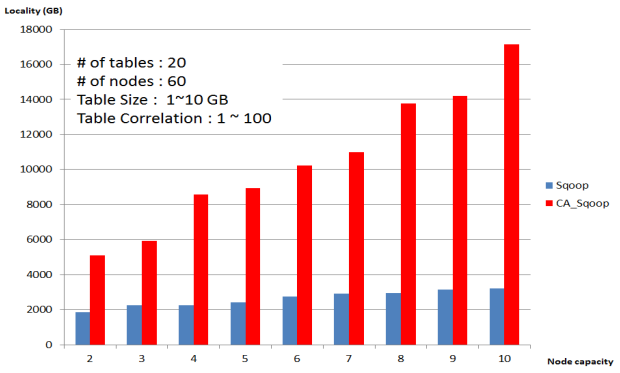

Figure 2. Importing twenty tables to a small cluster

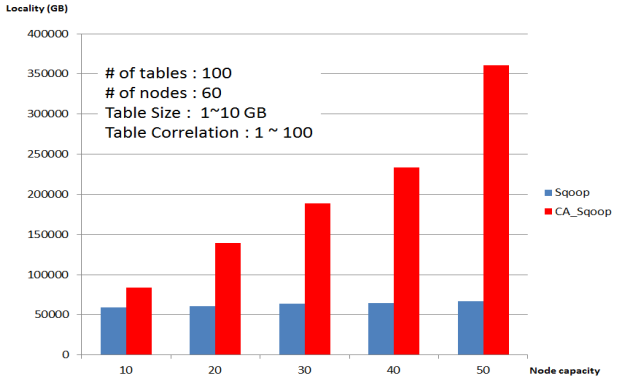

Figure 3. Importing one hundred tables to a large cluster

Fig. 4 presents smaller difference in data locality while the number of table is increased. The results also show a smaller space to be improved while there are more tables to be migrated from RDB.

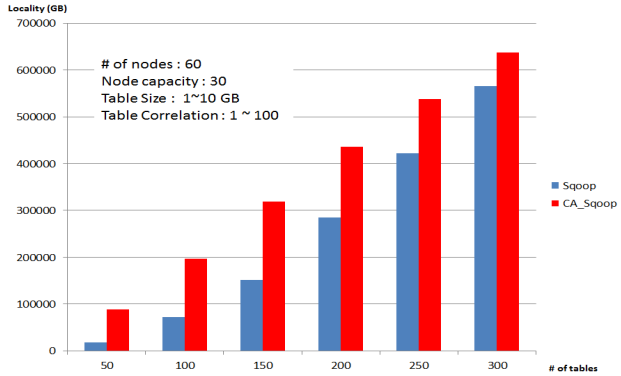

Figure 4. Importing different number of tables to a cluster

Fig. 5 shows that CA_Sqoop increases data locality with wider range of correlation. The reason is that CA_Sqoop aims at improving the operations related to larger tables.

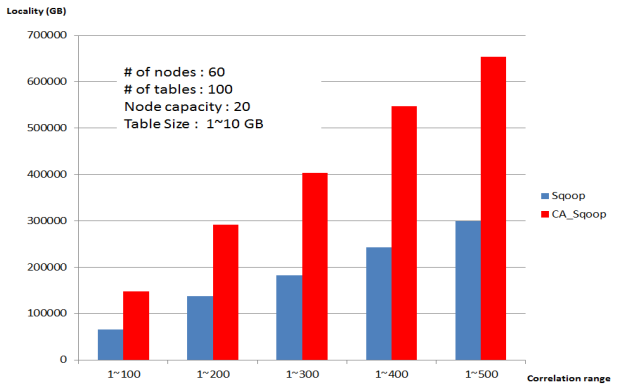

Figure 5. A database has different correlation range.

Fig. 6 gives the results of importing one hundred tables to clusters with different number of nodes. Although

original Sqoop has less data locality while number of nodes increased, CA_Sqoop can derive higher data locality in the same situation.

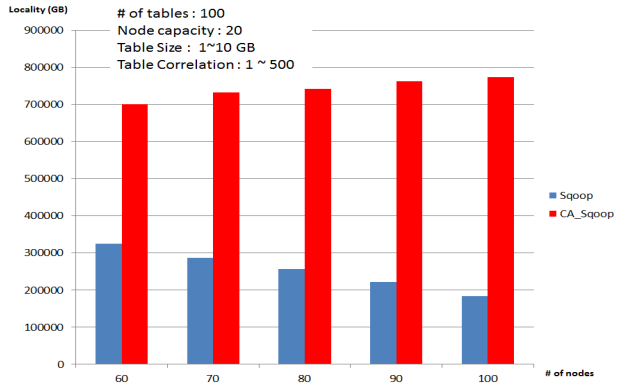

Figure 6. Impact on different sizes of cluster

Those simulations show that our approach CA_Sqoop can get more data locality than tradition Sqoop. We believe that locality can reduce data transmission by network and improve MapReduce join performance.

VI. CONCLUSIONS

The design of distributed file system provides the ability to execute jobs in parallel while data is split and imported to nodes randomly. However, this behavior may not be good for processing some data which is frequently used. To improve the data placement may enhance the performance in terms of database usage and is the motivation of this paper.

JOIN is one of the frequently used operations to database and requires much resource. While performing JOIN on a distributed file system, it is sensible to execute jobs on some nodes. If data is not distributed in these nodes, data transformed from other nodes through network is necessary and will affect the execution time of JOIN. The design of the proposed method, CA_Sqoop, is to first analyze the log to know which tables are frequently used for JOIN. Then generate TCS and distribute above tables on the same node if possible.

Simulation results show that CA_Sqoop can improve the data locality in all scenarios even importing 300 tables to the distributed file system. With CA_Sqoop, the time of data transformation and job execution can be significantly improved.

REFERENCE

[1] Masato Asahara, Shinji Nakadai and Takuya Araki, “LoadAtomizer: A Locality and I/O Load aware Task Scheduler for MapReduce, ” in 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 317-324, 2012.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt and Andrew Warfield, “Xen and the Art of Virtualization, ” SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, vol. 37, Issue 5, pp. 164-177, December 2003.

[3] Kasim Selcuk Candan, Jong Wook Kim, Parth Nagarkar, Mithila Nagendra and Ren-wei Yu, “Scalable Multimedia Data Processing in Server Clusters,” IEEE MultiMedia, pp. 3-5, 2010.

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C., Hsieh Deborah A., Wallach Mike Burrws, Tushar Chandra, Andrew Fikes, and Robert E.Gruber, “Bigtable: A Distributed Storage System for Structured Data,” 7th UENIX Symposium on Operating Systems Design and Implementation, pp. 205-218, 2006.

[5] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters, ” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[6] Sven Groot, “Jumbo: Beyond MapReduce for Workload Balancing,” Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on Cloud Computing Technology and Science, vol. 4, pp. 2675-2678, 2011. July.

[7] C. Jin and R. Buyya, “Mapreduce programming model for net-based cloud computing, ”in Proceedings of the 15th International Euro-Par Conference on Parallel Processing, Euro-Par (Berlin, Heidelberg), pp. 417–428, 2009.

[8] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz and Ion Stoica, “Improving MapReduce Performance in Heterogeneous Environments, ” 8th Symposium on Operating Systems Design and Implementation, pp. 29–42, 2008. Dec.

[9] Andrey Balmin, Tim Kaldewey, Sandeep Tata, ”Clydesdale: Structured Data Processing on Hadoop, “ACM SIGMOD International Conference on Management of Data, pp. 705-708,2012.

[10] Jenq-Shiou Leu, Yun-Sun Yee, Wa-Lin Chen,”Comparison of Map-Reduce and SQL on Large-scale Data Processing,” International Symposium on Parallel and Distributed Processing with Applications, pp. 244-248, 2010.

[11] Steven Lynden, Yusuke Tanimura, Isao Kojima and Akiyoshi Matono,” Dynamic Data Redistribution for MapReduce Joins,” IEEE International Conference on Coud Computing Technology and Science, pp. 717-723, 2011.

[12] Dawei Jiang, Anthony K. H. Tung, and Gang Chen,” MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters,”IEEE Transactions on knowledge and Data Engineering, vol. 23, no. 9, pp. 1299-1311, 2011.

[13] Hung-Ping Lin, ”Structured Data Processing on MapReduce in NoSQL Database, ”Master Thesis in National Chiao Tung University, 2010.

[14] Meng-Ju Hsieh, Chao-Rui Chang, Jan-Jan Wu, Pangfeng Liu and Li-Yung Ho, ”SQLMR : A Scalable Database Management System for Cloud Computing, ”International Conference on Parallel Processing (ICPP), pp. 315-324, 2011.

[15] Rubao Lee, Tian Luo, Yin Huai, Fusheng Wang, Yongqiang He, and Xiaodong Zhang, ” YSmart: Yet Another SQL-to-MapReduce Translator, ” International Conference on Distributed Computing Systems, pp. 25-36, 2011.

[16] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive - a warehousing solution over a Map-Reduce framework, ”PVLDB, vol. 2, no. 2, pp. 1626–1629, 2009.