Applied Mechanics and Materials
ISSN: 1662-7482, Vol. 855, pp 153-158
doi:10.4028/www.scientific.net/AMM.855.153

Submitted: 2016-06-20
Revised: 2016-08-19
Accepted: 2016-08-23
Online: 2016-10-25

A Comparison of ORC-Compress Performance with Big Data Workload on Virtualization

Kritwara Rattanaopas[1,a*], Sureerat Kaewkeerat[2,b] and Yanapat Chuchuen[1,c]

[1]Computer Department, Faculty of Science and Technology,
Songkhla Rajabhat University, Thailand

[2]Business Computer Department, Faculty of Management Sciences,
Songkhla Rajabhat University, Thailand

[a]kritwara.ra@skru.ac.th, [b]sureerat.ka@skru.ac.th, [c]yanapat.ch@skru.ac.th

Keywords: ORC file, performance optimization, Hadoop, Hive

Abstract. Big Data is widely used in many organizations nowadays. Hive is an open source data warehouse system for managing large data set. It provides a SQL-like interface to Hadoop over Map-Reduce framework. Currently, Big Data solution starts to adopt HiveQL tool to improve execution time of relational information. In this paper, we investigate on an execution time of query processing issues comparing two algorithm of ORC file: ZLIB and SNAPPY. The results show that ZLIB can compress data up to 87% compared to NONE compressing data. It was better than SNAPPY which has space saving 79%. However, the key for reducing execution time is Map-Reduce that were shown by a less query execution time when mapper and data node were equal. For example, all query suites in 6-node(ZLIB/SNAPPY) with 250-million table rows has quite similar execution time comparison to 9-node(ZLIB/SNAPPY) with 350-million table rows.

Introduction

The Huge volume of information is called "Big Data". The variety of Big Data analytic systems can be largely categorized into two groups based on database types: SQL for relational database and NoSQL for non-relational database. Big Data infrastructure [1] has data warehouse and data processing facility. A Hadoop with HDFS is one of popular storage for the largest database and a fully replicated distributed database. In data processing facility, for example, R framework is compatible with statistic computing while Hadoop-Hive/Hadoop-HBase is compatible with query processing. In this paper, we evaluated the performance of Hadoop-Hive infrastructure on virtualization platform with Big Data solution. Hive infrastructure is based on Hadoop Distributed File System (HDFS) that provides a database query for Big Data. Recently, Hadoop was developed as an Enterprise Hadoop Eco system for commercial use [2], for example, Cloudera (www.cloudera.com), Hortonwork (hortonworks.com), and MAPR (www.mapr.com). Hive can process query with Map-Reduce technique. It used Map method for filtering and sorting data. Reduce() method is used for summarizing data. Both number of Map and Reduce node for each execution can be calculated by size of data in HDFS. Hive is set to bring a new compress file format of database, which is called "Optimized Row Columnar (ORC)" [3]. ORC file has only 2 standard algorithms: ZLIB and SNAPPY. Both is high level compress algorithm which have different method and size of output file. Considering the benefits of Hive and ORC files, we chose them to improve query processing performance with weather station data in virtualization environment.

The rest of the paper is organized as follows. In Background Review and Related Works, we describe background review and related work including Hadoop, Hive and Optimized Record Columnar (ORC) file. In Methodology, we describe the research methodology including experimental setup and workloads. Evaluation section presents experiment results. We also discuss the experimental results in results and discussion. Finally, conclusions section concludes and reveals possibly future works.

Background Review and Related Work

In recent years, Big Data with Hive have rapid development pace; Hive has been significantly updated by new features since original Hive version 0.10.0 available on 11 January, 2013. From the point of view, Hive is hot research topics which can be run SQL query in Hadoop's environment that we descried in this section.

Related Works. Several researches involved Hive has previously focused on query performance. Both of Yin Huai [4] and Pouria Pirzadeh [6] surveyed and presented method to improved query execution performance that using the same approach with Avrilia Floratou [7] who purposed query execution on Impala, Hive(MapReduce) and Hive(Tez) with online transaction processing workload including TPC-H and TPC-DS. Their result found a better execution performance on Impala. Closely this research Liping Zhang [8] presented a new compression algorithm which is call "LZMA", that give better IO throughput with ORC file on both of Hive and HBase that similar with Songlin Hu [9] research which combines dualtable of Hive and HBase. This paper presented other approach for execution performance comparison between ZLIB and SNAPPY with Hive(MapReduce) on a virtualization environment of Hadoop.

Hadoop. This software is popular to solve solution about data set in Big data. Hadoop is portable software developed for Linux operating system that included Java platform or Standard Edition (Java SE). In Fig. 1, a storage part of Hadoop is called "Hadoop Distributed File System (HDFS)". A processing part of Hadoop is called "MapReduce". It splits files into large blocks and distributes to data node in a cluster. The Hadoop Ecosystem has a lot of software built on top including Hive, R Connectors, Mahout, Pig and Oozie.

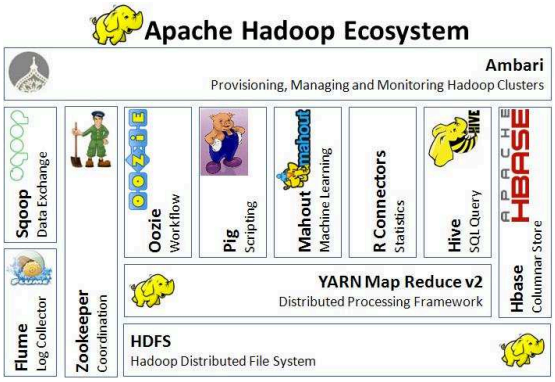

Fig. 1 Hadoop Ecosystem [2]

Hive. Hive software is first SQL interface on Hadoop that is an advantage including ability to scale out and high availability. Hive can be executed SQL query as follows: "SELECT…WHERE", "SELECT…ORDER BY", "GROUP BY" and "JOIN". It also supports SQL functions including sum, avg, count, min, and max. HiveQL statement are parsed, complied and optimized to produce a query execution plan which is a Directed Acyclic Graph (DAG) of map and reduce jobs. Hive supports two types of join method including a repartitioned join and join (key, value) pair.

Optimized Record Columnar File. Hive has a file format which is called Optimized Record Columnar File or ORC File. It significantly differences from the original RCFile. In Fig. 2, ORC has 2 steps to compress a raw data to compressing data. First, raw data is transfered to encoding stream. Second, it divides stream and compressing using ORC compressing type.

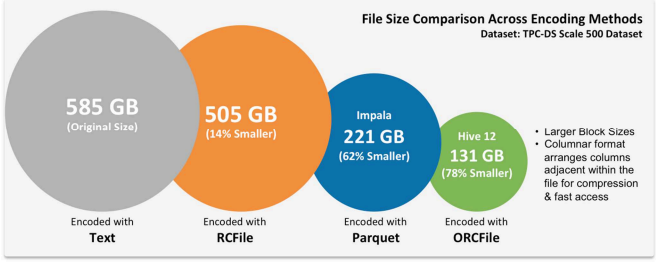

Fig. 2 Example size of file type on Hive [4]

ORC used lightweight compression algorithm e.g. Zlib, Lzo or Snappy Codec. We experimented with both of zlib and snappy algorithms that are available for Hive tables. Our results show that snappy compression provides slightly better query performance than zlib. Similar result with TPCH workload are present in [4].

Methodology

In research methodology section, the testing scenarios of weather station data are presented. We investigated on the performance of query processing in HiveQL, which provides a SQL-like interface on Hadoop system. Our experiment focused on desktop computer with KVM's virtualization environment that could solve the Big Data of weather data problem.

Experimental Setup. To employ full-virtualization of KVM, we experimented with Hadoop-Hive over 4 desktop computers with Intel i7-2600 Quad core @ 3.40 GHz which support Intel-VT, 4 GB DDR3, 1.0 TB SATA disks and RTL8111 Gigabit Ethernet. Our system architecture for Hadoop-Hive shown in Fig. 3 is described as follows:

- Master Node employed only 1 big virtual machine in Host01 with 3 vCPU and 3 GB RAM
- Data Node involved 3 virtual machines on each Host with 1 vCPU and 1.5 GB RAM

Host OS: KVM platforms run on CentOS 7.2 64 bits. The virtual machine were stored on XFS partition

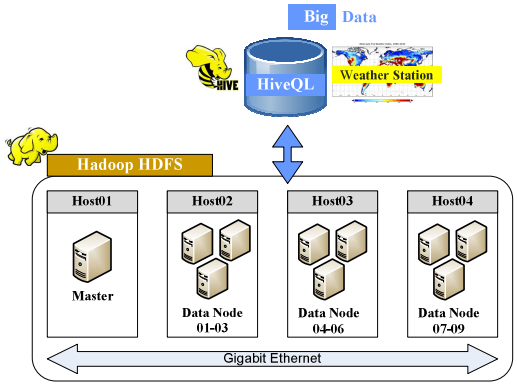

Fig. 3 Hadoop-Hive Architecture for Big Data solution

In Fig. 3, both name node and data node installed Hadoop version 2.6.4 with replication factor is 3. Hive version 1.2.1 was installed only in name node. We created and transferred weather station data to Hive with HiveQL on name node.

Workload. The query execution performance of weather station dataset that was used the same weather station table schema in Table 1 on Hive. We also created weather station table in ORC file using difference compress type including NONE, ZLIB and SNAPPY that was a major effect on the table rows which described in Fig. 4.

Table 1 Weather station table schema

Field	Stationcode(Primary Key)	station	datefield	prcp	tmax	tmin
Type	STRING	STRING	STRING	DOUBLE	DOUBLE	DOUBLE

Table 1 gives details of database schema. There was six fields: station code; station name; record date as "datefield"; precipitation weather as "prcp"; maximum temperature as "tmax" and minimum temperature as "tmin".

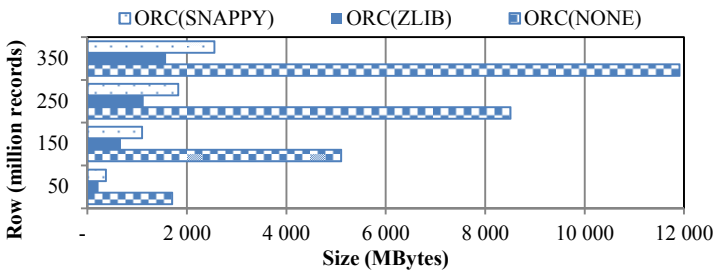

Fig. 4 Size of weather station table on each ORC compress types

Fig. 4 shows size of each ORC compress type of weather station table. Overall, it can be seen that the quality of data compression using ZLIB had the highest compression ratio at 7:1 compared to NONE, while SNAPPY's compression ratio 4:1. All result inclined toward the same ratio on every table rows.

We performed all experiments based on different query function. Each query suite was devide into 3 cases that was excuted on HiveQL interface 3 times which is described as follows:

- Case 1: Finding maimum of "tmax" separate by group station on each day. We used "SELECT station, datefield, MAX(tmax) FROM weather_orc GROUP BY station, datefield";
- Case 2: Finding average "tmax" separate by group station on each year. We used "SELECT station, cast(SUBSTR(datefield,1,4) as INT), AVG(tmax) FROM weather_orc_tmp GROUP BY station, cast(SUBSTR(datefield,1,4) as INT); "
- Case 3: There are 3 steps in this case. First, this step is same as Case 2. Second, calculating maximum of "tmax" as "SELECT station, MAX(tmax) FROM weather_orc GROUP BY station". Finally, joining station code from both table from previous step.

Evaluation

In this section, we conducted experiments in 3 scenarios to investigate the differences of query processing time employing the compared number of data node and ORC compress types in Fig. 5, Fig. 6 and Fig. 7. We ran our experiments on Intel i7 architecture with KVM virtualization environment described in previous section. All figures show execution time compared to number of data node based on each ORC compress type on query suites. HiveQL-interface can comply all query suites on workload section into an optimized execution plan of map and reduces jobs shown in Table 2, Table 3 and Table 4.

Experiment I. In Experiment I, we used Case 1 query suite to evaluate execution time as follows 50-million, 150-million, 250-million and 350-million table rows with start data node from 3 to 9. All results executed only two table rows and last table rows repeated execution in next data node step, for example, 3-node(ZLIB) has a result of 50-million and 150-milion that 150-million repeat executed in 6-node(ZLIB).

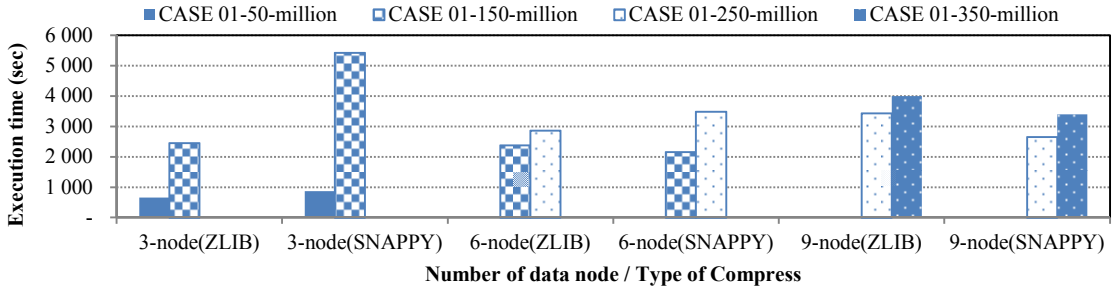

Fig. 5 Experiment I on Case 1 query suite

Table 2 Map-Reduce of Experiment I on each table rows and data node

3-node(ZLIB)		3-node(SNAPPY)		6-node(ZLIB)		6-node(SNAPPY)		9-node(ZLIB)		9-node(SNAPPY)	
50-m	150-m	50-m	150-m	150-m	250-m	150-m	250-m	250-m	350-m	250-m	350-m
M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R
1/1	2/3	2/3	4/5	2/3	4/5	4/5	7/8	4/5	7/7	7/8	11/11

* M-Mapper,R-Reducer,m-million

A graph in Fig. 5 and Table 2 show that execution time decreases, while the mapper and tables rows are increasing. 3-node(SNAPPY) has the highest execution time with 4 mappers in 150-million table rows. It has lower performance than 6-node(SNAPPY) which can decrease execution time to 2,154 second or 39.7%. On the other hand, 3-node(ZLIB) has similar execution time with 2 mappers in 150-million table rows compared to 6-node(ZLIB). Although the execution time decreases while the number of mappers and data node, 9-node(SNAPPY) with 11 mappers in 350-million has quite similar execution time with 6-node(SNAPPY) with 7 mappers in 250-million.

Experiment II. We used Case 2 query suite, which calculate average temperature for each weather station per year, using evaluate execution time as follows 50-million, 150-million, 250-million and 350-million table rows while data node start from 3 to 9. All results executed only two table rows same as experiment I.

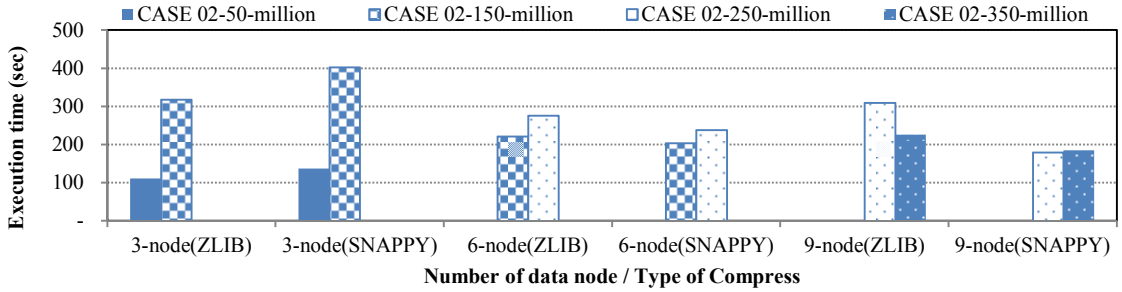

Fig. 6 Experiment II on Case 2 query suite

Table 3 Map-Reduce of Experiment II on each table rows and data node

3-node(ZLIB)		3-node(SNAPPY)		6-node(ZLIB)		6-node(SNAPPY)		9-node(ZLIB)		9-node(SNAPPY)	
50-m	150-m	50-m	150-m	150-m	250-m	150-m	250-m	250-m	350-m	250-m	350-m
M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R
1/1	2/1	2/1	4/1	2/1	4/1	4/1	7/1	4/1	7/2	7/1	11/2

* M-Mapper,R-Reducer,m-million

A bar graph in Fig. 6 and Table 3 show that execution time are quite similar to execution time comparing to experiment I. 3-node(SNAPPY) has the highest execution time with 4 mappers in 150-million table rows. It has lower performance than 6-node(SNAPPY) which can decrease execution time with 7 mapper to 204 second or 50%. However , 9-node(SNAPPY) with 11 mappers in 350-million better execution time than 6-node(SNAPPY) with 7 mappers in 250-million.

Experiment III. According to experiment I and experiment II, we used Case 3 query suite, which experimented with join multiple table, that evaluate execution time as follows 50-million, 150-million has 250-million and 350-million table rows with start data node from 3 to 9.

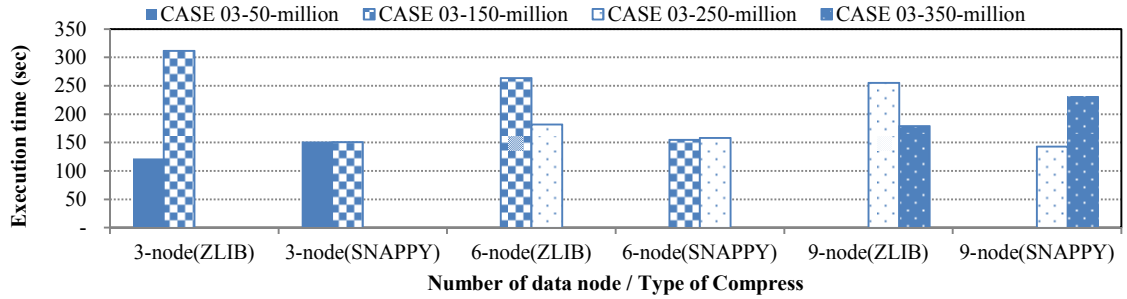

Fig. 7 Experiment III on Case 3 query suite

Table 4 Map-Reduce of Experiment III on each table rows and data node

3-node(ZLIB)		3-node(SNAPPY)		6-node(ZLIB)		6-node(SNAPPY)		9-node(ZLIB)		9-node(SNAPPY)	
50-m	150-m	50-m	150-m	150-m	250-m	150-m	250-m	250-m	350-m	250-m	350-m
M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R	M/R
1/0	2/1	2/1	4/1	2/1	4/2	4/1	7/2	4/2	11/1	7/2	11/1

* M-Mapper,R-Reducer,m-million

In Fig. 7 and Table 4, The execution time decreases as the mapper increasing. 3-node(ZLIB) has the highest execution time with 2 mappers in 150-million table rows. It has lower performance than 6-node(ZLIB) which can decrease execution time to 264 second or 84.6%. , Not only the execution time decreases as mappers increase but also data node increase, for example, 9-node(ZLIB) with 11 mappers in 350-million has a quite similar execution time with 6-node(ZLIB) with 4 mappers in 250-million. On the other hand, 9-node(SNAPPY) with 11 mapper has lower performance than 9-node(ZLIB) in 350-million.

Conclusions

We have investigated the HiveQL, which build on top of a virtualization Hadoop-HDFS, applying two compression of ORC file to improving a query processing with weather station big data and conducted three query suites for this experiment in previous section. In term of ORC file capacity – size of storage can be reduce, ZILB can be reduced capacity to 13% of NONE which was small than SNAPPY. The results show that the number of mapper which is a necessary factor in the decreasing execution time solution that can be concluded as follows:

When mapper was equal to data node, Hive could provide better query processing efficiency which is shown by a less query execution time. For example, all query suites in 6-node (ZLIB/SNAPPY) were a quite similar execution time comparison to 9-node(ZLIB/SNAPPY).

When mapper was lower than data node, increasing data node could not be effectively used. On the other hand, this key show that table rows can be scaled up. For example, both of 6-node and 9-node were a quite similar execution time in 250-million table rows.

When mapper was higher than data node, it does take advantages to increase data node for improve a query processing. For example, the 3-node with 4 mappers is worse performance than the 6-node with 4 mappers at 50% in query suite (case 2).

In conclusion, a virtualization Hive with big data solution can be scaled up the table rows by using mapper to optimized query plan. Moreover, this knowledge can help reducing unnecessary data node and optimizing data node with mapper in Amazon Elastic MapReduce (Amazon EMR), which is commercial provider of Apache Hadoop that has a cluster platform running big data frameworks, such as Apache Hive and Apache Pig.

Our future work will be analyzing and testing other significant factors which improve an execution time and performance of HiveQL in other virtualization environment.

Acknowledgement

The authors would like to acknowledge Faculty of Science and Technology, Songkhla Rajabhat University, Thailand for support of this research.

References

[1] V. Reynolds, Big Data For Beginners: Understanding SMART Big Data, Data Mining & Data Analytics For improved Business Performance, Life Decisions & More!. Kindle Edition, 2016.

[2] Information on http://thebigdatablog.weebly.com/ blog/the-hadoop-ecosystem-overview.

[3] Information on https://cwiki.apache.org/confluence/display/Hive/LanguageManual-ORC

[4] Information on http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/

[5] Y. Huai, A. Chauhan, A. Gates, G. Hagleitner, E. N. Hanson, O. O'Malley, J. Pandey, Y. Yuan, R. Lee, and X. Zhang, Major Technical Advancements in Apache Hive, Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 1235–1246.

[6] P. Pirzadeh, M. J. Carey, and T. Westmann, BigFUN: A performance study of big data management system functionality, in: 2015 IEEE International Conference on Big Data (Big Data), 2015, pp.507–514.

[7] A. Floratou, U. F. Minhas, and F. Özcan, SQL-on-Hadoop: Full Circle Back to Shared-nothing Database Architectures, in: Proc. VLDB Endow., vol. 7, no. 12, 2014, pp. 1295–1306.

[8] L. Zhang, Q. Chen, and K. Miao, A Compatible LZMA ORC-Based Optimization for High Performance Big Data Load, in 2014 IEEE International Congress on Big Data, 2014, pp. 80–87.

[9] S. Hu, W. Liu, T. Rabl, S. Huang, Y. Liang, Z. Xiao, H.-A. Jacobsen, X. Pei, and J. Wang, DualTable: A Hybrid Storage Model for Update Optimization in Hive, 2015 IEEE 31st International Conference on Data Engineering, 2015, pp. 1340-1351.