Using compression tables to improve HiveQL Performance with Spark A Case study on NVMe Storage Devices

2022 26th International Computer Science and Engineering Conference (ICSEC) | 978-1-6654-9198-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSEC56337.2022.10049309

Youppadee Intasorn
Computer Science Program,
Faculty of Science and Technology
Songkhla Rajabhat University,
Songkhla, Thailand
youppadee.in@skru.ac.th

Kritwara Rattanaopas*
Computer Science Program,
Faculty of Science and Technology
Songkhla Rajabhat University,
Songkhla, Thailand
kritwara.ra@skru.ac.th

Yanapat Chuchuen
Information Technology Program,
Faculty of Science and Technology
Songkhla Rajabhat University,
Songkhla, Thailand
yanapat.ch@skru.ac.th

***Abstract*— Query language execution is widely used in big data. The SQL standard is the major query language. Big data has a lot of SQL-like tools, for example: Spark-SQL, Hive, Drill, and Presto. This paper focused on Hive with the Spark engine. To increase Hive's query performance in a case study, NVMe Solid State Devices, we proposed the compressed Parquet file including SNAPPY, gzip, and Zstandard (zstd). Query workloads use TPC-H benchmark. Thus, this compression codec can reduce the main transaction table of TPC-H benchmark by 56%, and some queries have lower CPU usage than Text file. However, the Hive on Spark engine with our proposed compression codecs for Parquet files has lower CPU usage than Text file in some TPC-H queries. Thus, NVMe storage with the Parquet file compression codec is more efficient than text files for improving query performance on the Spark engine.**

Keywords-Spark engine; Hive; Hadoop; parquet file;

I. Introduction

Data warehouses based on Structured Query Language (SQL) are now widely used for big data processing frameworks. A Hadoop cluster is the popular framework. It stores data in a distributed file system, which is called the Hadoop Distributed File System (HDFS) [1]. MapReduce [1] is the default paradigm of Hadoop applications. It has two phases, including Map phase and Reduce phase. Currently, Apache Spark has been presented at Berkeley AMPLab since 2009. Since Hadoop version 2.7, the Spark framework has been used to investigate Hadoop clusters. Spark SQL [2] builds on top Spark core which is designed for data processing with multiple programming language including Scala, Python and Java. This paper presents an alternative method by using Hive on Spark engine. It can be used SQL query with HiveQL on Hive data warehouses like other MapReduce and Tez. It can only execute HiveQL queries with Spark-Hadoop version 2.4 lower [2].

Although Hive on Spark engine [1] can process HiveQL queries, This alternative engine has more performance than the MapReduce engine [4][5]. A business solution based on the TPC-H benchmark can create a Decision Support System (DSS). TPC-H has 22 decision support queries for business solution with eight tables. It has dbgen tool, which is data generator to Hive's database. Hive on Spark engine is supported text file and parquet file format. It can compress data size with various compression codes including SNAPPY, GZIP, deflate, and BZIP2. Snappy is the default. The Spark version 2.4 [6] has ZStandardCodec (zstd), which is the new compression code for parquet file.

For the Hadoop ecosystem of this paper, we installed an NVMe Solid State Drive, which is high-speed storage for HDFS and Hive data warehouses. For distributing data blocks in expensive storage, HDFS storage sets only one block or no replication. This is a challenge designed to improve query performance with a compression codec. This paper's goal was to propose a method for increasing query execution rate with Hive on Spark engine in private cloud hosting with Hadoop cluster. The results found that Spark with parquet compression can reduce execution time and CPU usage in some queries. Some queries operate multiple HiveQL statements, which can increase execution time compared to a single HiveQL statement with large-sized tables. Furthermore, queries with compression codec use less CPU than tables stored as text files on the Spark engine (discussed in Section 5).

In the design and implementation of the Hadoop ecosystem with Hive on Spark engine, we described all the research components in section II. Hardware and software are installed in the Hadoop ecosystem, as illustrated in the methods and materials section. Our results of the TPC-H benchmark are shown in the results and discussion section. The profit of compression data with Hive on Spark engine is described in conclusion.

II. Background and Related Work

A. Hive

Apache Hive [1] stores databases in HDFS (Hadoop). It likes SQL data warehouses and operates query languages like SQL, which is called HiveQL. It also has a few functions like SQL-92. HiveQL queries can be run in Hive CLI, Beeline, and Hivecommand. It is built around batch and parallel processing on distributed databases. Hive can be scaled to meet user demand.

B. Spark engine on Hive

Apache Spark [3] can analytical engine for executing data of petabytes size. Multiple languages are supported by Spark including Python, Scala and Java. Hive can be used Spark engine same MapReduce engine. This paper runs HiveQL with Spark engine.

C. *Apache Parquet*

Apache parquet [1] on Hive warehouse is a major type of table store. It is a column-oriented which is called columnar storage. It can be compressed parquet on Hive table with compression codec including SNAPPY, GZIP, deflate, BZIP2, and Zstd. The challenge to improving query performance we proposed compression codec with Parquet on Hive data warehouse.

D. *Related work*

Big data has various data storage and processing tools. Hadoop is the most popular analytics tool for big data [7]. It has more analytic tools, including open-source and commercial. For example, Hive, Impala, and Spark provide SQL queries over HDFS. HBASE provides only NoSQL. This paper focuses on Hive on Spark engine on the other hand, with Xiaopeng Li's research [4] comparison of three-type query tools with only 7 queries of the TPC-H benchmark in CDH5 (Cloudera Hadoop). Impala with Parquet has the least CPU and memory of the others. Tatsuhiro Chiba [8] studies all TPC-H queries on Spark SQL with the Hive data warehouse. SQL-on-Hadoop uses Hive with three different execution engines, including MapReduce, Tez, and Spark [9][10][11][12]. The TPC-H benchmark is widely used in that research [4][5][8][9][10][11][12] the same study The TPC-H benchmark has a flexible database size by using the dbgen command. Moreover, it is the more popular benchmark of TPC Professional Affiliates.

III. METHODS AND MATERIALS

In this section, this paper describes the research Hadoop ecosystem architecture and DSS workload of the TPC-H benchmark. The Hadoop ecosystem has been installed, including the Hadoop framework, Hadoop-Hive, and Spark engine for Hadoop. It can be installed on the virtualization host like a private cloud hosting. The TPC-H workload is stored in the Hive warehouse and processed by using the Spark engine for Hive in cloud computing.

A. *Experimental Setup*

This Hadoop ecosystem architecture has Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz 64GB of RAM and SSD M.2 PCIe 240 GB. Worker node virtual machine resource like as P2 type of Amazon EC2 instance [13]. This cluster system is described follows:

1) *The Hadoop master node and Spark master*: This virtual machine has 4 virtual cores and 8 GB of RAM
2) *The Hadoop data nodes and Spark worker nodes:* It has 3 nodes and virtual resources same Hadoop master node.

The software on physical host machine installed AlmaLinux release 8.5 (Arctic Sphynx) and QEMU 4.2.0. We create 4 instances, including Master, and Workers. Master installed AlmaLinux release 8.5 (Arctic Sphynx) same physical machine and on top software, including hadoop version 2.10.1, apache-hive version 2.3.9, and spark version 2.4.8. Also, Those Workers installed AlmaLinux release 8.5 (Arctic Sphynx) and on top software, including hadoop version 2.10.1, apache-hive version 2.3.9, and spark version 2.4.8. This paper use 3 Spark workers and set some Spark parameters, including driver.cores, driver.memory, executor.cores, and executor.memory.

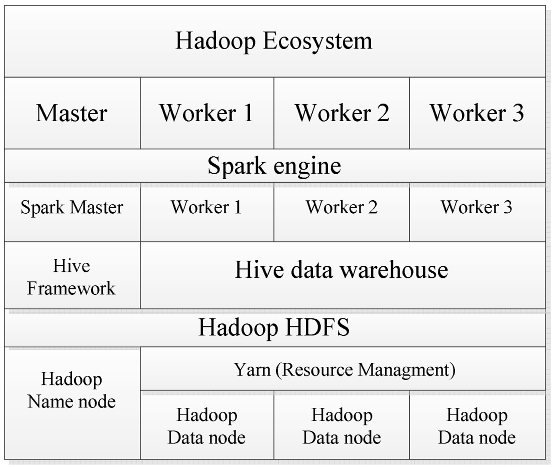
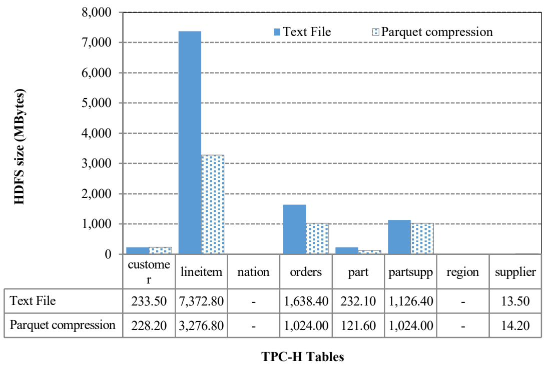


Fig. 1. The Hadoop ecosystem architecture of this research.

In Fig. 1, the research Hadoop ecosystem architecture based on Hadoop HDFS and Yarn resource management. This HDFS set replication factor to 1 or none replication, which data distributed data block over data nodes. Hive data warehouse store file on HDFS and Hive on Spark engine must set Spark's parameters in Hive configure file. Also, Hive uses this command "set hive.execution.engine = spark;" or set spark engine in "hive-site.conf".

B. *TPC-H*

Research's workload used TPC-H benchmark. It is a decision support workload with 22 ad-hoc HiveQL queries from eight tables. TPC-H for MPP DBMS evaluation and TPC-H for Hive are available for download on github [14]. This experiment can run TPC-H queries and load data from Hive onto HDFS across multiple nodes. HDFS's size of Hive tables is illustrated in Fig. 2. TPC-H benchmark can generate testing data by using the dbgen command. In addition, the hive table size in this paper is 30 GB, implying that the dbgen scale factor is used with "-s 30." There are three large size tables in the eight tables: lineitem (59,986,052 records), orders (15,000,000 records), and partsupp (8,000,000 records). Moreover, TPC-H ad-hoc queries have 22 queries and describe operation in Table 1. On each query has more groups by and join tables based on business decision model.



	customer	lineitem	nation	orders	part	partsupp	region	supplier
Text File	233.50	7,372.80	-	1,638.40	232.10	1,126.40	-	13.50
Parquet compression	228.20	3,276.80	-	1,024.00	121.60	1,024.00	-	14.20

Fig. 2. TPC-H table size (Megabytes) comparison between test file and parquet file in HDFS.

TPC-H table has eight tables. The large size tables are lineitem, orders, and partsupp. Some tables have the category name, including nation and region, that their records do not exceed 10 records. Thus, Parquet file with compression codec can reduce HDFS size form the Text file by 56% (lineitem), 37% (orders), and 29% (partsupp). This large size tables has more affect to CPU usage during execution process by Spark engine as illustrated in section 4. Total HDFS size of TPC-H table as Text file is 10.37 GB and TPC-H table as compression Parquet file size is 5.56 GB. Also, TPC-H table with compression codec can reduce HDFS size to 46.42%.

TABLE I. HIVEQL OPERATION OF ALL OF TPC-H QUERIES

Query	Query name	SQL operations		
		Jobs	Tables	Large Table
Q1	pricing_summary_report	1	1	lineitem
Q2	minimum_cost_supplier	3	5	partsupp
Q3	shipping_priority	1	3	orders, lineitem
Q4	order_priority	2	2	orders, lineitem
Q5	local_supplier_volume	1	6	orders, lineitem
Q6	forecast_revenue_change	1	1	lineitem
Q7	national_market_share	2	5	orders, lineitem
Q8	volume_shipping	1	7	orders, lineitem
Q9	national_market_share	1	6	Orders, partsupp
Q10	returned_item	1	4	orders, lineitem
Q11	important_stock	3	3	partsupp
Q12	shipping	1	2	orders, lineitem
Q13	customer_distribution	1	2	orders
Q14	promotion_effect	1	2	lineitem
Q15	top_supplier	3	3	lineitem
Q16	parts_supplier_relationshi	3	3	partsupp
Q17	small_quantity_order_revenue	2	2	lineitem
Q18	large_volume_customer	2	3	orders, lineitem
Q19	discounted_revenue	1	2	lineitem
Q20	potential_part_promotio	5	4	lineitem, partsupp
Q21	suppliers_who_kept_orders_waiting	3	4	orders, lineitem
Q22	global_sales_opportunity	4	2	orders

HiveQL operations and a description of the TPC-H benchmark are shown in Table 1. Each table shows the name of the large size in HiveQL operation. Lineitem tables, which are the largest tables, were used in Q1, Q3-Q8, Q10, Q12, Q14, Q15, Q17, and Q19-Q21. Also, queries have more joined tables, including Q2, Q5, Q7-Q10, Q20, and Q21. Moreover, queries have only one HiveQL line, including Q1, Q3, Q5, Q6, Q8–Q10, Q12–Q14, and Q19. This operation of all queries has an effect on the execution time results with the Parquet compression codec as illustrated in Section 4.

C. Spark engine configuration

In order to execute SQL with Spark framework in Hive has a various method, including Spark SQL, and Spark engine. This paper focused on Hive on Spark engine. In master node, we configured Hive on Spark engine parameters, follows as:

- Spark master is yarn
- Spark driver-cores is 1 core (default)
- Spark driver-memory is 1 GB (default)
- Spark executor-cores is 1 core
- Spark executor-memory is 1.5 GB

In our experiment, Master node has 1 virtual machine (installed Hadoop master, Spark master, and Hive) and worker node has 3 virtual machines (installed Hadoop datanode, and Spark worker). Hive on Spark engine executed TPC-H queries with 11 Spark workers and 1 Spark Driver.

IV. RESULTS AND DISCUSSION

In order to present the CPU usage and execution time results of TPC-H benchmark. As a result, we executed and logged CPU resources on each Worker node by using Linux's SAR command. MapReduce processes the external table stored as Text file, which is base performance as illustrated in Fig. 3. It used the same Spark engine resoures. Spark engine executes both Text file and Parqute file with compression codecs, including SNAPPY, Gzip, and zstd. The results are the average CPU useage and the average execution time of a retrieve executed 10 times on each TPC-H query.

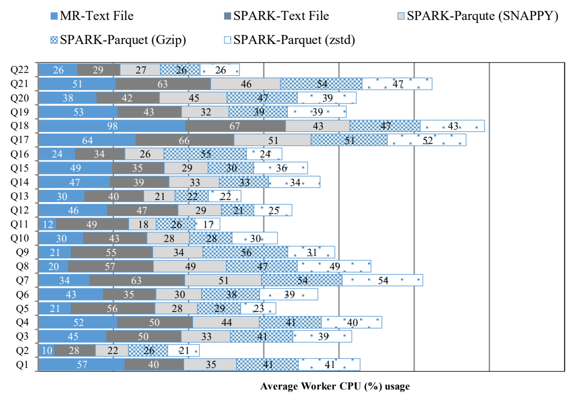


Fig. 3. CPU usage (%) of all of TPC-H queries on each executors.

Some TPC-H queries have CPU usage of Spark engine more than MapReduce in Text file including Q2, Q3, Q4, Q5, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q17, Q18, Q20, Q21 and Q22. However, Higher CPU usage of Spark engine can reduce execution time in all of queries as illustrated in Fig. 4. Thus, Spark engine with compression codec have lower CPU usage than Text file in all of TPC-H queries. SNAPPY and Gzip are the popular compression codec on Hive and Hadoop [8],[9]. In addition, zstd is an alternative compression codec to Parquet file. It has a quite similar CPU usage SNAPPY in some TPC-H queries.

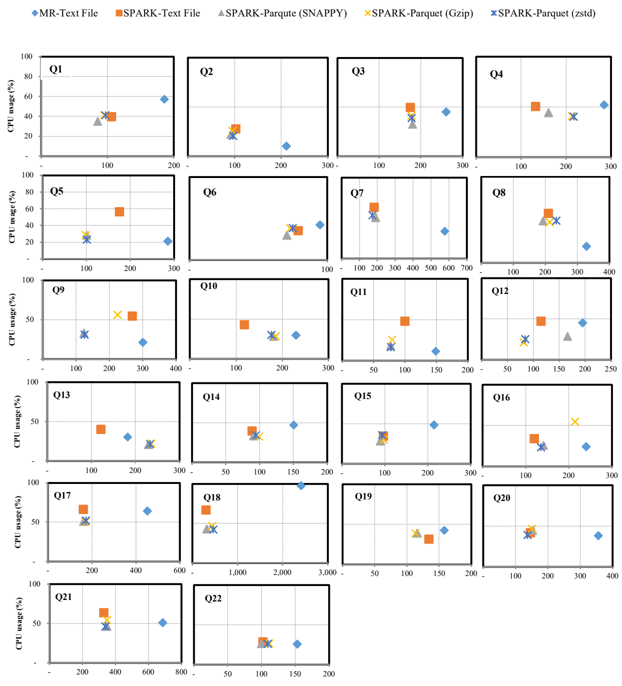


Fig. 4. CPU utilization (%) and Execution time (second) while running TPC-H queries.

In order to evaluate query performance, we present relation between CPU usage and exection time on each query of TPC-H benchmark in Fig. 4. Q1 is a close simila query performance between Text file and Parquet file with compression codec on Spark engine same Q1, Q2, Q3, Q6, Q7, Q8, Q14, Q15, Q16, Q20, Q21 and Q22. Those queries are load large size table only one HiveQL statement on each query. In addition, Q5, Q9, Q11, Q12, Q17,Q18 and Q19 have a better query perforamce in Parquet file with compression codec than Text file on Spark engine and Those queries statement load lagresize table one time. However, Q4, Q10 and Q13 used more CPU usage and high execution time than Text file on Spark engine. Thoes queries load more large size tables on HiveQL statement and All statements in whole query runtime. Only Q13 query found Mapreduce results lower than Parquet with compression codec on Spark engine because it is lower operation only left join and no condition.

V. Conclusion

In this paper, we conducted a query performance evaluation on Hive on Spark engine. In order to do so, we investigate NVMe storage, which has faster read and write speeds than standard Solid State Drives, in the HDFS data node. Research's Hadoop ecosystem has three virtual machines: Spark worker, HDFS data nodes. Hive tables are distributed across Hadoop clusters using only the HDFS replication factor. HDFS size of TPC-H tables stored as compressed Parquet files is only 5.56 GB, which is 53.58% of TPC-H tables stored as Text file. HiveQL queries are executed with 11 executors and 16 GB of RAM on 3 worker nodes, 1 driver and 1 GB of RAM.

The results found a high query performance with a compressed Parquet file on Spark engine in the case of a single HiveQL statement query, as illustrated in Fig. 4 and Table I. TPC-H queries with tables stored as compressed Parquet files operated only the large size table in statement and no join. Those queries have quite similar query performance tables stored as Text file. However, HiveQL operates multiple large-sized tables with joins in one statement. It has lower query performance in a compressed Parquet file on Spark engine. Some queries show a better query performance by using a compressed Parquet file. This proposed method uses compressed Parquet files on HDFS with NVMe storage. It has the benefit of improving query performance and reducing HDFS size in the case study of high-speed storage.

In the future, we plan to investigate Spark SQL with Hive in Spark version 3 later. It can use various types of hive files, including ORC and AVRO. We plan to evaluate query performance, other parameters, and system indicators.

Acknowledgment

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

References

[1] V. Deepak, Practical Hadoop Ecosystem, Springer Science+Business Media New York: New York, USA, 2016.

[2] L. Hien, Beginning Apache Spark 2, SAN JOSE, California, USA, 2018

[3] H. Szehon, "Hive on Spark: Getting Started", [Online] Available: https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark%3A+Getting+Started, 2018.

[4] C. Tatsuhiro, and O. Tamiya, "Workload characterization and optimization of TPC-H queries on Apache Spark," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS),* 2016. pp. 112-121.

[5] I. Todor, P. Matteo, "The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet," in *Concurrency and Computation: Practice and Experience*, Vol 32, 2020, pp 1-32.

[6] Apahe Spark 2.4.3 document, "Parquet Files", [Online] Available: https://spark.apache.org/docs/2.4.3/sql-data-sources-parquet.html

[7] GVL, R. M. K., and Sowmyarani, C. N. "Review of various Big Data analytics tools using Hadoop," in Vol 8, 2022, pp. 156-162.

[8] T. Chiba, and T. Onodera, "Workload characterization and optimization of TPC-H queries on Apache Spark," In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*,2016, pp.112-121.

[9] T. Ivanov, and M. Pergolesi, "The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet," in *Concurrency and Computation: Practice and Experience*, Vol *32*, 2020, pp 1-31.

[10] H. E. Ciritoglu, J. Murphy, and C. Thorpe, "Importance of data distribution on hive-based systems for query performance: An experimental study," In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020. pp. 370-376.

[11] Y. Shen, J. Xiong, and D. Jiang, "Using Vectorized Execution to Improve SQL Query Performance on Spark," In *50th International Conference on Parallel Processing*, 2021, pp. 1-11.

[12] T. M. Allam, "Estimate the Performance of Cloudera Decision Support Queries," In *International Journal of Online & Biomedical Engineering*, Vol 18, 2022, pp 127-138.

[13] Amazon Web Services, Inc., "Amazon EC2 P2 Instances", [Online] Available: https://aws.amazon.com/ec2/instance-types/p2/?nc1=h_ls, 2021.

[14] X. Reynold, " TPC-H_on_Hive" [Online] Available: https://github.com/rxin/TPC-H-Hive, 2011.