Testing of big data analytics systems by benchmark

Mingang Chen
Shanghai Key Laboratory of Computer Software Testing and Evaluating
Shanghai Development Center of Computer Software Technology
Shanghai, China
cmg@ssc.stn.sh.cn

Wenjie Chen, Lizhi Cai
Shanghai Key Laboratory of Computer Software Testing and Evaluating
Shanghai Development Center of Computer Software Technology
Shanghai, China
cwj@ssc.stn.sh.cn, clz@ssc.stn.sh.cn

***Abstract*—With the rapid development of big data technologies and applications, various big data analytics systems have been released by open source communities and industry. So testing and evaluating the overall performance of these big data analytics systems has become an important research topic. The paper analyzes in detail the challenges of testing big data analytics systems and proposes the method and strategies for the testing. Furthermore, the paper presents two cases of testing big data analytics systems by benchmark.**

Keywords—testing; big data; benchmark; TPC-DS; TPCx-BigBench.

I. Introduction

In recent years, big data has become a hot topic for governments and enterprises, and it is considered as a new driving force for innovation in the information era. This is based on the following two facts: firstly, in the past ten years, the speed of data generating is becoming faster and faster, and we have already entered the big data era; secondly, big data contains huge values, and has brought about revolutionary developments in many fields, such as e-commerce, finance, transportation, medical and health service, etc.

However, the "3V" characteristics (volume, variety, and velocity) of big data make challenges for data processing and analytics. Recently, industry and academia have launched a variety of big data analytics system to cope with the challenges, such as open source Apache Hive [1], Apache Spark [2], and commercial Transwarp Inceptor, Cloudera Impala, IBM Big SQL and so on. More and more enterprises or organizations use big data analytics system to build the business application and obtain decision support from data. Therefore, testing and evaluating big data analytics systems has become one of the important research subjects of the big data fields.

Testing of big data analytics system mainly has the following three roles. (1) We can verify the correctness of functionalities and the reliability of the big data analytics system before it is deployed and put to use. (2) We can carry out a fair comparison of the performance of different big data analytics systems. (3) We can optimize the performance of big data analytics systems by testing.

Presently, testing of big data analytics system mainly uses benchmarks, and by benchmark testing, we can analyze and evaluate the functionalities, performance, reliability, and compatibility of the system. There are three categories of benchmark in the testing of big data analytics systems. The first category is the micro benchmark. This category of benchmark principally aims at testing a certain component of the big data analytics system thus is also called component-level benchmark. Such as TeraSort can only be used to test the system's performance for sorting text data, and GridMax can only be used to test the performance of various MapReduce job in the Hadoop clusters. Therefore, the micro benchmark cannot evaluate the performances of big data analytics system entirely. The second category is the comprehensive benchmark. This category of the benchmark can test more than one components of big data analytics system. For example, Hibench is a comprehensive benchmark, and its workload including micro benchmarks, web search, SQL query and machine learning [3]. The third category is the application oriented benchmark, which is characterized by simulating the scenario of big data applications in the enterprise. TPC-DS is a benchmark for testing big data decision support systems [4, 5]. TPCx-BigBench [6, 7] is the first end-to-end, application-level big data benchmark based on TPC-DS. Due to the standardization and usability of TPC-DS and TPCx-BigBench, more and more organizations begin to use these two benchmarks to test, evaluate and compare the overall performance of big data analytics systems.

This paper will discuss in detail the challenges of testing big data analytics systems in Part II, and propose method and strategies of how to test big data analytics systems in Part III. In Part IV, two cases of testing will be presented, that is testing of Transwarp Inceptor by TPC-DS and performance comparison of Hive and Spark SQL by TPCx-BigBench. In addition, some preliminary analysis will be made on how to optimize the performance of Spark SQL by benchmark testing. Finally, we conclude the paper in section V.

II. The Challenges of Testing Big Data Analytics System

Due to the "3V" characteristics of big data and the complexity of big data analytics system, this brings about challenges for testing big data analytics system.

First is the complexity of the technologies on big data analytics system. It generally adopts distributed architectures, such as master-slave or peer-to-peer. And factors that will

This work was funded by Science and Technology Commission of Shanghai Municipality Program (16511101202, 17411952800).

affect the performance of the system under test are complex, such as network environment, hardware configurations, system configuration parameters, and virtualization etc. For instance, Hadoop system has over 200 configuration parameters.

Second is the complexity of test datasets. The test datasets of big data analytics system need not only to meet the "3V" characteristics of big data but also to represent typical business scenes.

Third are the challenges of testing methods and tools, such as the traditional testing tools can no longer be appropriate, lacking automatic testing methods and the customization of testing and diagnosing schemes. Different modules in the big data analytics require different testing techniques. For example, we test the performance of Spark SQL by SQL's queries, while we test throughput and latency of Spark Streaming by loading streaming data.

Fourth, the testing of big data analytics system requires more professional and more comprehensive testing abilities. Testers not only need to have the testing expertise but also need to master the big data analysis and processing technology. For example, testers need to know how to load data from Hadoop HDFS into a Hive table and verify if the loading is correct.

III. Benchmark Testing method of big data analytics system

The testing of big data analytics system with benchmark can generally be divided into 6 phases, that is requirement analysis for testing big data analytics systems, preparing the testing environment, preparing the test datasets and workload, loading the test datasets, testing for the big data analytics system and analysis of the testing result, as shown in Fig.1.

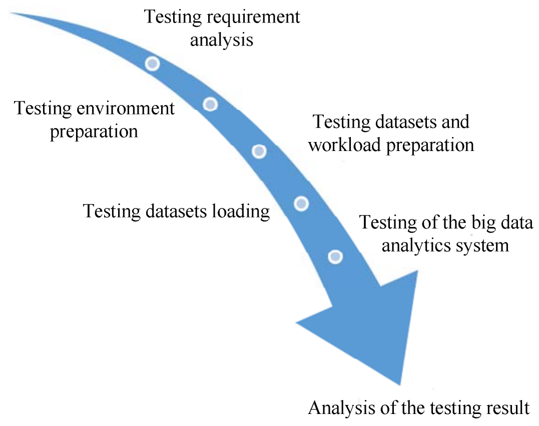
Testing requirement analysis
Testing environment preparation
Testing datasets and workload preparation
Testing datasets loading
Testing of the big data analytics system
Analysis of the testing result

Fig.1. Benchmark testing method of big data analytics system

A. *Requirement analysis of testing for big data analytics systems*

The phase of requirement analysis for testing big data analytics systems is by and large same as traditional software testing, including specifying the objects of testing, the purposes of testing, the environment of testing, the datasets of testing, technology and tools of testing and the risk of testing, etc. But the key point of testing big data analytics system is the performance and reliability of the system. For example, how efficient is the system's processing and analysis of data with large-scale datasets? Whether tasks of data processing can be migrated automatically or not when a node in the cluster goes down? Will the data be lost in a distributed environment when a node crashes?

B. *Preparing the testing environment*

In order to test a big data analytics system, we need to prepare a cluster of distributed data storage and computing, at the same time a sufficient storage space is required to store and analyze the large-scale datasets. It is worth noticing that the storage space here not only refers to the hard disk space but also memory space, especially in testing Apache Spark, due to the 60% occupation of memory is used for buffering RDD (the data structure of Spark), so enough memory space should be set apart for the testing program. We should be careful that, the testing environment should be ensured "clean". In other words, we should ensure that there is no other applications running in the cluster, the CPU and memory of the node in the cluster are both at their minimum utilization.

C. *Preparing the test datasets and workload*

The datasets for testing big data analytics system comes from two sources: one is the real data from business, such as data from weblogs or database of business; the other is simulated data generated by big data benchmarking tools. TPC-DS and TPCx-BigBench are two benchmarks that have been nominated in the industry. It should be noted that we should set appropriate data scale, data type, and data model according to the requirement of the testing. The workload is the core of performance testing of big data analytics system. It needs to reflect business scenarios and data analytical techniques. The workload in TPC-DS or TPCx-BigBench is the set of queries to be executed against the test datasets.

D. *Loading the test datasets*

During the phase of loading test datasets, we should verify if the data has been loaded correctly into the distributed storage system. For example, whether the data is loaded into the right HDFS storage directory? Is the size of data file correct? If the data need to be loaded into the distributed database system, we should verify if the data can be load into the table in the database correctly.

E. *Testing of the big data analytics system*

Testing of the big data analytics system needs to focus on system's functionality, performance, reliability, and compatibility.

1) *Functionality testing*

The Functionality testing of big data analytics system mainly verifies whether functions of the system in data storage, data processing, data I/O etc. are correct? For example, whether data processing based on MapReduce is correct? Whether the results of SQL queries on the SQL-On-Hadoop system are correct? And whether the data I/O is complete?

2) *Performance testing*

The performance testing of big data analytics system needs to test the performance of data I/O, data processing and analytic and the performance of SQL query on the system and so on. For example, we can test the reading and writing performance of Hadoop HDFS using single large data file or multiple large data files. For SQL-On-Hadoop systems, the performance of SQL query is the most important performance metric.

3) *Reliability testing*

The reliability testing of big data analytics system needs to focus on the following two aspects:

- If the task can be automatically migrated when a task of data analytics failed at a certain node (may be due to lack of memory), so as to ensure the task is executed correctly?
- If one or some nodes in the cluster go down, will the task of the data analytic be executed correctly due to the fault-tolerant mechanism of the system?

4) *Compatibility Testing*

The compatibility testing of big data analytics system needs to verify the compatibility of the file system, the compatibility of data storage format, the compatibility of SQL syntax and so on.

F. *Analysis of the testing result*

During the phase of analysis of the testing result, we need to analyze system's testing metrics (functionality, performance, reliability, and compatibility) comprehensively according to the testing requirement and finish the testing report.

IV. CASES OF TESTING BIG DATA ANALYTICS

According to the test method described in Part Ⅲ, in this section, we present two cases of testing big data analytics system.

A. *Testing for Transwarp Inceptor by TPC-DS*

1) *Requirement analysis of testing Transwarp Inceptor*

The purpose of testing Transwarp Inceptor is to verify the functionality of ETL，and evaluate the performance of SQL query and compatibility of SQL syntax through automated testing scripts. The method of testing follows the TPC-DS specification.

2) *The system under test and environment*

a) *Transwarp Inceptor big data analytics system*

Inceptor is a commercial big data analytics system developed by Transwarp Technology Co., Ltd. It provides high-speed SQL analytics based Apache Spark. It can help businesses to build high-speed, scalable data warehouses, and perform interactive analysis, real-time reporting, and visualization of data. Transwarp Inceptor has a three-tier structure from bottom to top: the storage layer, the distributed computing engine layer and the interface layer, as is shown in Fig.2.

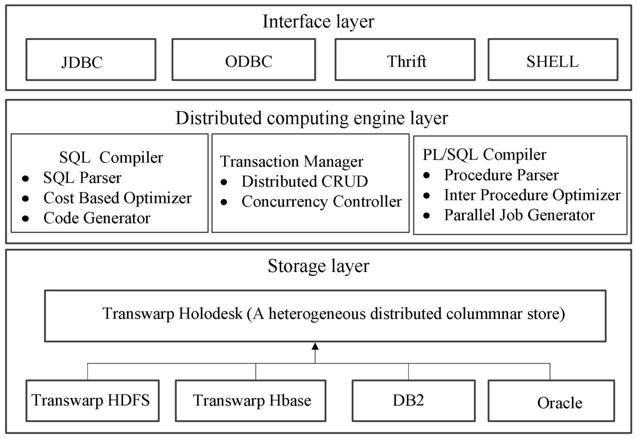

Fig.2. Architecture of Transwarp Inceptor

b) *Test environment*

The test environment consists of four physical servers, and the configurations of servers are same, as is shown in Table I. Four servers make up a Transwarp cluster through Gigabit network.

TABLE I. THE HARDWARE CONFIGURATION OF THE TESTING SERVERS

	Node1	Node2	Node3	Node4
Model	Dell PowerEdge R720			
CPU	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (2 CPU x 6 cores)			
Memory (GB)	256	256	256	256
Storage	24 TB HDD hard drive			
Operating System	Red Hat Enterprise Linux 6.5			
Hadoop	Transwarp DataHub v3.4 Hadoop 2.2			
Inceptor	Transwarp Inceptor v4.0			
Roles	Primary NameNode, Inceptor Server, DataNode	Secondary NameNode, Inceptor MetaStore, DataNode	DataNode	DataNode

3) *Generating test datasets and workload by TPC-DS*

a) *TPC-DS*

TPC-DS is testing benchmark for decision support system proposed by TPC (Transaction Processing Performance Council). TPC-DS models the decision support functions of a retail product supplier. The business model of benchmark simulates sales and returns of the three main channels (stores, online retailers, and catalogs). The business model contains 7 fact tables and 17 dimension tables, and tables are organized by star and snowflake mixed model. A reduced business model of TPC-DS is shown in Fig.3.

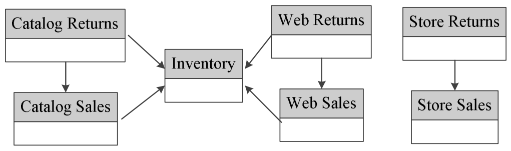

Fig.3. TPC-DS database schema

TPC-DS allows users to generate the different scale of datasets from 100G to 100T according to the user's test requirements and test environment. In general, the TPC-DS benchmark has following characteristics:

- A large amount of business data and test cases (SQL queries) can answer real business problems.
- A total of 99 SQL queries follow the SQL 99 and SQL 2003 core syntax standard, and SQL queries are complex.
- The test cases include a variety of business models, such as interactive query, statistical analysis, iterative OLAP and data mining.
- Almost all of the test cases need high I/O loading and CPU computing.

b) The generation of test datasets and workload

In this phase, we use the data generation and query generation tools (DSTools v1.3.0) provided by the TPC-DS benchmark to generate 500GB test datasets and 99 SQL queries through automated shell scripts, and the script fragment is as follows.

```
# Generate 500GB test datasets in the specified HDFS directory
1: dbgen2 -scale 500 -dir HDFS_LOCATION

# Generate 99  queries compatible with Oracle syntax for 500GB
datasets through the query template
2: qgen2 –query99.tpl –directory QUERY_TEMPLATE –dialect
oracle  -scale 500
```

The 500GB test datasets consist of 24 tables of the database (7 fact tables and 17 dimension tables) mentioned above. The 99 SQL queries implement business intelligence by answering real business questions.

4) Data loading

In the data loading phase, we first create 24 tables in Transwarp Inceptor to build the data warehouse for testing. The schemas of tables are provided by the TPC-DS benchmark. Then we load the datasets that have been generated in the HDFS into tables. The following script fragment shows how to load datasets in HDFS into the inventory table.

```
# load inventory.dat into the inventory table
1: LOAD DATA  inpath '/tpc_ds/data/inventory.dat' INTO TABLE
inventory;
```

5) Testing for Transwarp Inceptor

The core of the TPC-DS based benchmark testing is the execution of 99 SQLs one by one. In testing, we verify the correctness of the test results and record the execution time of SQL. We execute 99 SQLs with automated scripts by three rounds and take the average time of three rounds as SQL's execution time. The following script fragment shows how to execute 99 SQL queries sequentially in Transwarp Inceptor.

```
# Execute all 99 SQL queries one by one
1: for(i = 1; i<=99; i++ ){
2:   sql = "query"+  i + ".sql";
3:   system( "transwarp -t -h localhost  -f ./sql/" + sql);
4:}
```

6) Testing Analysis

In the case of the 500GB test datasets, the four categories of SQL execution time are shown in Table II. Test results show that 96 out of 99 SQL queries can be run directly in Transwarp Inceptor. There only 3 SQL queries need minor modification to be compatible with SQL compiler of Transwarp Inceptor. Considering that the TPC-DS specification allows SQL's minor modification, so Transwarp Inceptor has good compatibility with SQL 2003 standard.

TABLE II. SQL QUERIES' EXECUTION TIME OF TRANSWARP INCEPTOR

SQL Categories	The number of SQL	The total execution time (seconds)	The average execution time (seconds)
Interactive query	9	197	21.9
Statistical analysis	69	7705	111.7
Iterative OLAP	10	4232	423.2
Data mining	11	3502	318.4

B. Testing Hive vs. Spark SQL by TPCx-BigBench

1) Requirement analysis of testing Hive vs. Spark SQL

Testing Hive vs. Spark SQL has two purposes. One is to utilize TPCx-BigBench as a benchmark for evaluating and comparing the performance of two SQL-On-Hadoop analytics systems. The other is to tune system parameters for optimizing analytics system's performance.

2) Systems under test and test environment

a) Hive

Hive is one of the first data analytics engines to be built on top of MapReduce. It was originally developed by Facebook to support data analysts to analyze large datasets in Hadoop by queries in a SQL-like declarative query language. This SQL-like language is called HiveQL and is based on the SQL language, but does not strictly follow the SQL 99 standard. Hive has now become the foundation of new SQL on Hadoop projects, such as Impala, Presto, and Spark SQL. Hive metadata has become the de facto standard for users to store and manage metadata (table names, column names, and types, etc.) in Hadoop ecosystem.

Although Hive is a widely used project, historically its biggest drawback has been performance. Most of the performance problems can be attributed to Hive's use of MapReduce as its execution engine. MapReduce is not a good choice for running ad hoc, interactive queries. The main reason is that MapReduce reads and writes to disk extensively, and there is a high startup cost for MapReduce jobs.

b) Spark SQL

Apache Spark is a cluster computing platform designed to be fast and general-purpose. Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and stream processing. One of the main features of Spark is to be able to run computing in memory, so Spark has faster computing speed than MapReduce.

Spark SQL [8] is the component that Spark uses to manipulate structured data. It allows querying data via SQL as well as the HiveSQL and it supports many sources of data, including Hive tables, Parquet, and JSON. Spark SQL is fully compatible with Hive. Spark SQL supports HiveSQL and Hive metastore, so we can compare the performance of Hive and Spark SQL under the same test datasets.

Spark SQL also seamlessly integrates with Spark machine learning libraries MLlib and Spark ML. For example, in a machine learning application, the DataFrame API provided by Spark SQL can easily be used for data cleaning and feature engineering.

c) Test environment

The test environment is a Cloudera Data Hub (CDH) cluster with 4 nodes connected directly through Gigabit network, and detail hardware and software are shown in Table III. Cloudera CDH 5.10 with default configurations was used for all tests.

TABLE III. TEST ENVIRONMENT FOR TESTING HIVE VS. SPARK

	Node1	**Node2**	**Node3**	**Node4**
CPU	Intel(R) Xeon(R) CPU E5-2695 v3 @ 2.30GHz (8 cores)			
Memory (GB)	64	80	80	80
Storage	4TB HDD hard drive			
Operating System	CentOS 6.7 x86_64			
Hadoop	Cloudera Data Hub 5.10.0 (Hadoop 2.6.0)			
Hive	Hive 1.1.0			
Spark	Spark 2.1.0 (--driver-memory 10g –execuotr-memory 20g)			
Roles	HDFS NameNode, ResourceManager	HDFS DataNode NodeManager		

3) Generating test datasets and workload by TPCx-BigBench

BigBench covers the “3Vs” characteristics of the big data system. The initial implementation of BigBench was at the Teradata Aster platform in 2014. Later on, BigBench was standardized by TPC in Nov. 2016, and TPC released TPCx-BigBench v1.2.0 as the benchmark for big data analytics system. BigBench benchmark consists of the data model, the data generator and the specification of the workload.

a) Data model of BigBench

The data model of BigBench includes structured data, semi-structured data, and unstructured data, as shown in Fig.4. The structured data of BigBench is adapted from TPC-DS. The semi-structured data is composed of clicks made by customers and guest users visiting the retailer’s website. The unstructured data is covered by product reviews submitted by actual customers or guest users. Therefore, BigBench satisfies the “variety” property of big data.

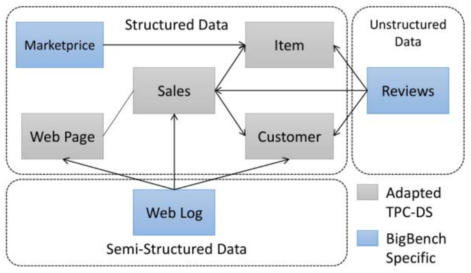

Fig.4. Data model of TPCx-BigBench

b) Data generator of BigBench

The data generator of BigBench is based on an extension of PDGF [9] and allows generating data in accordance with the data model. It can not only generate the structured data but also generate the semi-structured and unstructured data. PDGF is a parallel data generator that is capable of generating large amounts data based on a scale factor. So, the “volume” property of big data is reflected in BigBench. In addition, the “velocity” property of big data is implemented through a periodic refreshing scheme that continually adds new data to different tables in the data model. The following script fragment shows how to set data storage directory and generate 50GB datasets parallel by BigBench.

```
# Set dataset’s HDFS storage path in userSettings.conf
1: export BIG_BENCH_HDFS_ABSOLUTE_PATH
="/user/$BIG_BENCH_USER"
2: export BIG_BENCH_HDFS_RELATIVE_HOME
="benchmarks/bigbench"
# Generate 50GB test datasets with TPCx-BigBench
1: $INSTALL_DIR/bin/bigBench runBenchmark –f 50 –m 8  –i
DATA_GENERATION
-f  <scale factor of dataset>
-m [number of map tasks for data generation]
-i  <benchmark phases to perform >
```

c) Query workload of BigBench

The BigBench query workload includes 30 queries, which are defined as questions about the business model. Ten of them have been taken from the TPC-DS workload. The other 20 queries were adapted from a McKinsey big data use cases and opportunities report. The 30 queries of BigBench can be classified from two aspects: data types and analysis methods, as shown in Table IV and Table V. Analysis methods can be grouped into four categories: Pure Hive Queries(Pure HQL), Hive Queries with MapReduce programs, Hive Queries using natural language processing(NLP/UDF/UDTF), and Queries using Apache Spark MLLIB(Machine Learning).

TABLE IV. DATA TYPES OF BIGBENCH'S WORKLOAD

Data type	Queries	Number
Structured data	query1,query6, query7, query9, query11, query13, query14, query15, query16, query17, query20, query21, query22, query23, query24, query25, query26, query29	18
Semi-structured data	query2, query3, query4, query5, query8, query12, query30	7
Unstructured data	query10, query18, query19, query27, query28	5

TABLE V. ANALYTIC METHOD OF BIGBENCH'S WORKLOAD

Analytic method	Queries	Number
Pure HQL	query6, query7, query9, query11, query12, query13, query14, query15, query16, query17, query21, query22, query 23, query 24	14
MapReduce	query2, query3, query4, query8, query30	5
Machine Learning	query5, query20, query25, query26, query28	5
NLP/UDF/UDTF	query1, query10, query18, query19, query27, query29	6

4) *Data Loading in BigBench*

Data loading in BigBench refers to load test datasets into Hive tables. The following script fragment shows how to load test datasets created in the phase of "DATA_ GENERATION" into Hive tables. We can verify whether data loading was successful or not by Hive's shell command.

```
# Load test datasets into Hive tables
1: $INSTALL_DIR/bin/bigBench runBenchmark –i LOAD_TEST

# Verify the test datasets was loaded successfully
2: hive> use bigbench;
3: hive> show tables;
```

5) *Testing for Hive vs. Spark SQL*

In order to compare the performance of Hive and Spark SQL, we use Hive engine and Spark engine respectively. We execute 30 queries in sequence to compare the execution time, as shown in TableⅥ. The script fragment is as follows. It is worth noting that before using Spark engine we need to ensure that Spark had access to the tables in Hive.

```
# Test Hive performance with BigBench
1: $INSTALL_DIR/bin/bigBench runBenchmark –i POWER_TEST

# Test Spark SQL performance with BigBench
2: $INSTALL_DIR/bin/bigBench runBenchmark –i POWER_TEST
–e spark_sql
```

TABLE VI. EXECUTION TIME FOR ALL QUERIES WITH SF 50(50G DATA)

Query No.	Analytic method	Execution time (seconds)	
		Hive	Spark SQL
query1	UDF/UDTF	296	124
query2	MapReduce	3904	1634
query3	MapReduce	1046	568
query4	MapReduce	3932	989
query5	Machine Learning	535	344
query6	Pure HQL	603	238
query7	Pure HQL	897	260
query8	MapReduce	680	251
query9	Pure HQL	1123	138
query10	NLP/UDF/UDTF	1133	1868
query11	Pure HQL	242	110
query12	Pure HQL	271	146
query13	Pure HQL	361	152
query14	Pure HQL	93	92
query15	Pure HQL	151	124
query16	Pure HQL	823	236
query17	Pure HQL	230	118
query18	NLP/UDF/UDTF	1066	903
query19	NLP/UDF/UDTF	401	317
query20	Machine Learning	341	322
query21	Pure HQL	613	175
query22	Pure HQL	160	128
query23	Pure HQL	254	145
query24	Pure HQL	307	118
query25	Machine Learning	483	350
query26	Machine Learning	249	291
query27	NLP/UDF/UDTF	121	201
query28	Machine Learning	456	510
query29	UDF/UDTF	237	154
query30	UDF/UDTF/MapReduce	3769	922

6) *Performance analysis of Hive vs. Spark SQL*

According to Table Ⅵ, Fig.5 and Fig.6, Spark SQL performance is 1-8 times that of Hive under 14 Pure HQL queries and 5 Hive queries with MapReduce. The main reason is that Spark SQL uses memory computing and optimized SQL engine. So Spark SQL is more efficient than Hive that uses MapReduce as a computing engine.

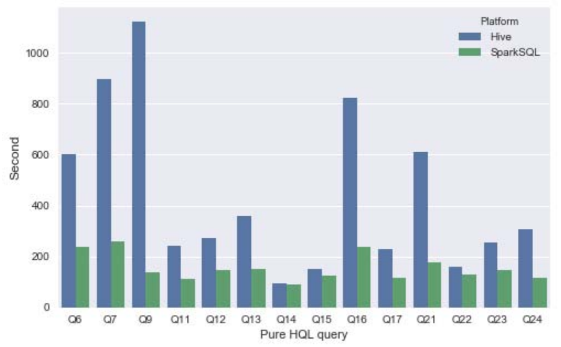

Fig.5. Hive and Spark SQL performance comparison by Pure HQL query

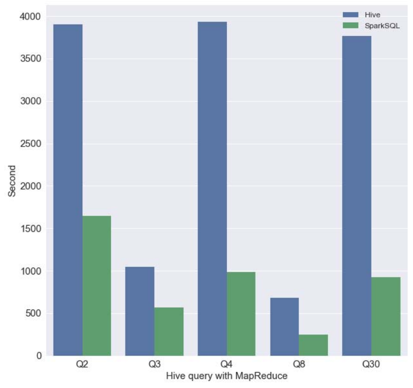

Fig.6. Hive and Spark SQL performance comparison by MapReduce query

For machine learning workload, Hive and Spark SQL are similar in performance, since both Hive and Spark SQL use Spark MLLIB as a machine learning engine, as shown in Fig.7.

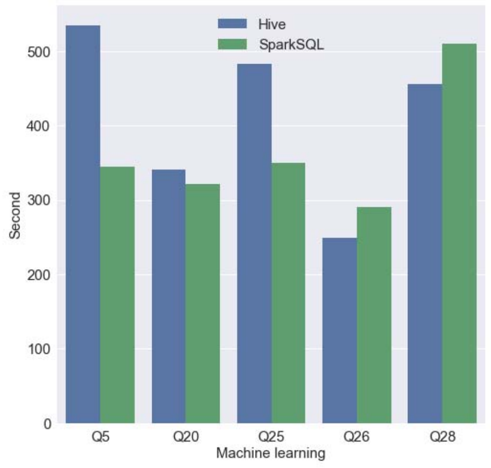

Fig.7. Hive and Spark SQL performance comparison by machine learning

Since NLP programs were written in the Python language, neither Hive nor Spark SQL can take advantage of the system's parallel computing features. As a result, for NLP/UDF/UDTF workload, Hive and Spark SQL performance's gap is not large, and Hive outperformed Spark SQL even on query 10 and query 27, as shown in Fig.8.

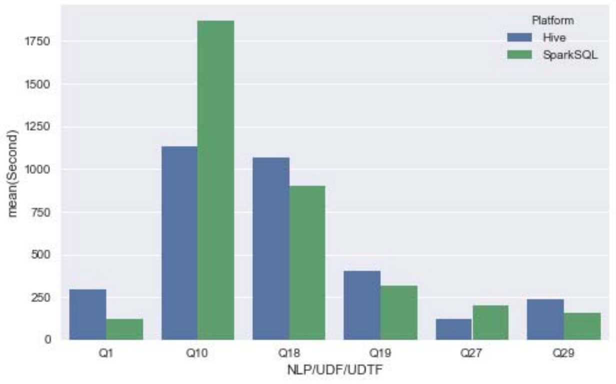

Fig.8. Hive and Spark SQL performance comparison by NLP/UDF/UDTF

For query10, we modify the parameter of *spark.sql.shuffle.partition* from the default of 200 to 50 to optimize the performance of Spark SQL. In Spark SQL, a large number of shuffle partitions means more tasks when shuffle operation occurs. More tasks in Spark SQL will increase the overhead of tasks startup and decrease the performance of the system. As shown in Fig.9, by optimizing Spark SQL's parameter, query 10 reduce its run time from 1868 seconds to 1376 seconds.

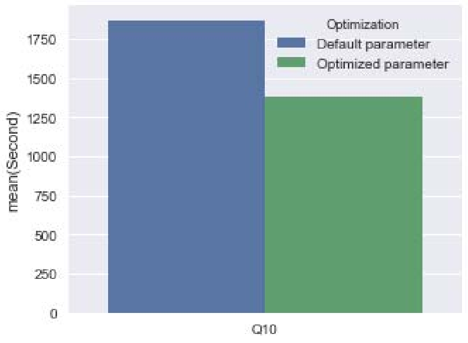

Fig.9. Spark SQL performance improvement through optimization

V. Conclusion

With the continuous development of big data applications and technologies, industry and academia pay more and more attention to the benchmark testing of big data analytics systems. It not only equitably compares the performance of multiple big data analytics systems, but also allows you to tune system parameters and optimize system performance. The paper analyzes the challenges of testing big data analytics system and summarizes the methods and strategies of the test. And the paper presents two cases of benchmark testing for big data analytics systems. In case 1, we present an automated system testing solution for Transwarp Inceptor by TPC-DS in detail, and the test includes system's functionality, performance, reliability and compatibility of SQL. In case 2, we test and compare the performance of Hive and Spark SQL by TPCx-BigBench, an application oriented end-to-end benchmark. Test results show that the performance of Spark SQL significantly better than Hive on the workload of pure HQL and query with MapReduce. In the future, we will further research new technologies of big data benchmarks [10], such as testing and evaluation of streaming analytics and graph analytics systems.

References

[1] A.Thusoo, J.S. Sarma, N. Jain, et al, "Hive-a petabyte scale data warehouse using hadoop", IEEE 26th International Conference on Data Engineering. IEEE, 2010, pp.996-1005.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, et al, "Spark: Cluster Computing with Working Sets", Usenix Conference on Hot Topics in Cloud Computing, Boston, USA, 2010.

[3] S. Huang, J. Huang, J. Dai, et al, "The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis". ICDE Workshops, 2010, pp. 41 - 51.

[4] R. O. Nambiar, M. Poess, "The making of TPC-DS", Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006, pp.1049-1058.

[5] M. Poess, R. O. Nambiar, D. Walrath,"Why you should run TPC-DS: a workload analysis", Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007, pp.1138-1149.

[6] A. Ghazal, T. Rabl, M. Hu, et al, "BigBench: towards an industry standard benchmark for big data analytics", Proceedings of the 2013 ACM SIGMOD international conference on Management of data, 2013, pp.1197-1208.

[7] TPCx-BigBench Standard Specification Version 1.2.0, November 2016, http://www.tpc.org/

[8] M. Armbrust, R. S. Xin, C. Lian, et al, "Spark sql: Relational data processing in spark", Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp.1383-1394.

[9] T. Rabl, M. Frank, H. M. Sergieh, et al, "A Data Generator for Cloud-Scale Benchmarking", TPCTC, 2010, pp.41-56.

[10] T. Rabl, M. Frank, M. Danisch, et al, "The vision of BigBench 2.0", Proceedings of the Fourth Workshop on Data analytics in the Cloud., ACM, 2015.