Письма в ЭЧАЯ. 2016. Т. 13, № 5(203). С. 1069–1073

EFFICIENT DATA MANAGEMENT TOOLS FOR THE HETEROGENEOUS BIG DATA WAREHOUSE

A. A. Alekseev [a], *V. V. Osipova* [a, 1], *M. A. Ivanov* [a], *A. Klimentov* [b], *N. V. Grigorieva* [a], *H. S. Nalamwar* [a]

[a] National Research Tomsk Polytechnic University, Tomsk, Russia

[b] Brookhaven National Laboratory, Upton, USA

The traditional RDBMS has been consistent for the normalized data structures. RDBMS served well for decades, but the technology is not optimal for data processing and analysis in data intensive fields like social networks, oil–gas industry, experiments at the Large Hadron Collider, etc. Several challenges have been raised recently on the scalability of data warehouse like workload against the transactional schema, in particular, for the analysis of archived data or the aggregation of data for summary and accounting purposes. The paper evaluates new database technologies like HBase, Cassandra, and MongoDB commonly referred as NoSQL databases for handling messy, varied, and large amount of data. The evaluation depends upon the performance, throughput, and scalability of the above technologies for several scientific and industrial use-cases. This paper outlines the technologies and architectures needed for processing Big Data, as well as the description of the back-end application that implements data migration from RDBMS to NoSQL data warehouse, NoSQL database organization, and how it could be useful for further data analytics.

PACS: 07.05.Hd; 07.05.Kf; 29.85.Ca

INTRODUCTION

The effectiveness of current-state-of-the-art Information Systems is largely determined by the result of the database design. The traditional approach to the database design is based on the principles of the relational data model, using strict operations of relational algebra. Thanks to this organization and normalization, the data structure of any complexity and for any domain can be formally designed. The relational database allows rapid collection and optimal allocation of data in the database, to ensure its completeness, relevance, and coherence during DML-operations. However, such SQL-systems are not designed for the most efficient, fast and multidimensional analysis, including for the very large amounts of data aka Big Data. Due to the complex and formalized structure of a query to obtain data that are big and not structured in various scenarios becomes difficult and not possible for RDBMS. Increasing tasks for processing Big Data in recent years have led to the need for using NoSQL-technology in IT-market [1].

[1] E-mail: vikosi@tpu.ru

VARIETY OF NOSQL-SYSTEMS

Depending on the data model, there are three classes of NoSQL-systems for processing Big Data: 1) Columnar; 2) Key-value (KV); 3) Document-oriented [2].

However, a complete refusing the relational database and moving into the NoSQL-technology does not solve problems with Big Data analysis and processing for reading as well as actualizing data. In this case the so-called heterogeneous data warehouse is suggested to develop for integrating the capabilities of these two technologies.

HETEROGENEOUS DATA WAREHOUSE

To create a heterogeneous data warehouse for processing Big Data, the special architecture is developed and implemented. The architecture of the heterogeneous data warehouse consists of three main components:

1) SQL-system — stores and processes actual data received from the source.

2) NoSQL-system — stores Big Data.

3) Data management system of the heterogeneous data warehouse — relates SQL and NoSQL systems.

The SQL-system (Oracle database) is used with a normalized relational data structure for the domain seismic geological exploration. The Oracle database works in cluster mode RAC [3]. Each NoSQL-system — Cassandra, Hadoop, MongoDB — is deployed on two separate servers and runs in cluster mode. The physical data structure consists of 23 relational tables according to the third normal form. The final component is the data management system of the heterogeneous data warehouse. The specific software is developed to export data from the database Oracle to the NoSQL-system and to analyze query performance to NoSQL-systems with getting further reports of the output data. For designing and developing the following system, several functions are implemented: data export (from Oracle to NoSQL), updating NoSQL data, query performance estimation (NoSQL), reporting, data uploading, and remote access to the system.

EXPERIMENTS

As a practical result, the heterogeneous data warehouse is designed and developed using three NoSQL-systems: Apache Cassandra, Apache Hadoop, and MongoDB.

All NoSQL-systems are installed on the servers HP Proliant DL 360 G6 with the following specifications: processor 2x Intel Xeon X5550 2.67 GHz, memory 12 Gb, HDD 500 Gb, Raid 1, OS Linux Ubuntu server edition 14.04.3 LTS.

The data management system of the heterogeneous data warehouse is developed and used to export data from Oracle and the NoSQL-system and to check load testing on NoSQL-systems, with further reporting to users.

By using the data management system of the heterogeneous data warehouse, data (about 4 million records) are exported from the Oracle database to the NoSQL-system. Later, based on the subsystem query performance estimation, a series of experiments were carried out to get the average execution time of each query compared to each NoSQL-system. In this regard,

by executing select queries to retrieve data from all the columns in the table, the following results were obtained at the end of the experiments.

1. **MongoDB.** For analyzing the performance of the NoSQL-systems, eight experiments were conducted with different number of queries ranging from 5 to 40 at interval of 5. The results of each experiment are shown in Table 1.

Table 1. **Experimental results for MongoDB**

Experiment No	Number of queries	Total execution time, s	Average execution time, s
1	5	14.57	2.91
2	10	24.70	2.47
3	15	45.33	3.00
4	20	49.74	2.48
5	25	61.28	2.45
6	30	73.08	2.43
7	35	97.19	2.78
8	40	97.50	2.44

The maximum, average, and minimum query execution time for MongoDB is 3, 2.62, and 2.44 s, respectively.

2. **Apache Hadoop.** To analyze the performance of this NoSQL-system, two combinations are defined: Hadoop and Hive, Hadoop and Impala [4].

a) **Hadoop + Hive.** For analyzing the performance of the combination Hadoop and Hive, nine experiments were conducted with different number of queries ranging from 2 to 10 at interval of 1 (the interval value is determined on a lot of execution time of all queries). The results of each experiment are shown in Table 2.

Table 2. **Experimental results for Hadoop + Hive**

Experiment No	Number of queries	Total execution time, s	Average execution time, s
1	2	26.44	13.22
2	3	39.42	13.14
3	4	51.76	12.94
4	5	65.81	13.31
5	6	79.10	13.18
6	7	94.10	13.44
7	8	102.16	12.77
8	9	114.79	12.76
9	10	130.50	13.05

The maximum, average, and minimum query execution time for Hadoop + Hive is 13.44, 13.09, and 12.76 s, respectively.

b) **Hadoop + Impala.** For analyzing the performance of the combination Hadoop and Impala, eight experiments were conducted with different number of queries ranging from 5 to 40 at interval of 5. The results of each experiment are shown in Table 3.

The maximum, average, and minimum query execution time for Hadoop + Impala is 1.76, 1.41, and 1.26 s, respectively.

Table 3. **Experimental results for Hadoop + Impala**

Experiment No	Number of queries	Total execution time, s	Average execution time, s
1	5	8.81	1.76
2	10	14.52	1.45
3	15	19.73	1.31
4	20	27.91	1.39
5	25	33.22	1.33
6	30	42.05	1.40
7	35	46.66	1.33
8	40	50.59	1.26

3. **Apache Cassandra.** For analyzing the performance of Cassandra, eight experiments were conducted with different number of queries ranging from 5 to 40 at interval of 5, using the original drivers (JDBC driver for Cassandra/CQL). The results of each experiment are shown in Table 4.

Table 4. **Experimental results for Apache Cassandra using the original drivers**

Experiment No	Number of queries	Total execution time, s	Average execution time, s
1	5	2.41	0.48
2	10	0.71	0.07
3	15	2.06	0.14
4	20	3.56	0.17
5	25	2.80	0.11
6	30	2.85	0.10
7	35	5.18	0.14
8	40	4.36	0.11

The maximum, average, and minimum query execution time for Apache Cassandra is 0.48, 0.17, and 0.07 s, respectively.

RESULTS

The following results were obtained during experiments, the average value of the execution time of one query per sample for each of the systems to different conditions, in increasing order of time:

1. Apache Cassandra — 0.17 s;
2. Hadoop + Impala — 1.41 s;
3. MongoDB — 2.62 s;
4. Hadoop + Hive — 13.09 s.

There is an exhaustive list of requirements to the NoSQL-systems for the complex estimation. Each requirement has a rank determined by different criteria (Table 5).

For ranking the monitoring system the given formula is used:

$$R_i = \sum_{j=1}^{m} V_{ij} L_{ij},$$

Table 5. **Comparison and systems ranking results for NoSQL-systems**

NoSQL-system	Query execution time for fetching	NoSQL-system monitoring	Ease of writing queries	Additional tools for processing data	Ease of system configuration and deployment	Completeness of documentation and manuals	Rank
Hadoop + Hive	30	10	9	20	2	5	76
MongoDB	35	6	5	8	5	4	63
Cassandra	50	8	10	10	3	5	86
Evaluation criteria weight	50	10	10	20	5	5	100

where R_i is a rank of the ith monitoring system that takes values from 0 to 100, (i varies from 1 to n); V_{ij} is a rank of the jth requirement to the ith monitoring system; L_{ij} is a weight of the jth requirement to the ith monitoring system (we need to write R_i by using math word equation).

In accordance with the value of the rank, two NoSQL-systems show a good result. Cassandra (in particular, DataStax) shows the best result while running select queries and has a simple built-in query language (CQLSH), the drivers for developing software, and so on. Hadoop (such as Cloudera) has an excellent system for monitoring the server, which allows managing and monitoring services via web interface, and there are tools for importing data from different systems (including relational databases) and other external sources.

CONCLUSIONS

In this paper, the comparative analysis of relational databases and NoSQL-technologies on several different criteria is conducted to identify the important pros and cons of processing Big Data. Due to limitations in current technologies, the unique heterogeneous Data warehouse is developed to work with Big Data efficiently. Thus, the typical representatives of the NoSQL-technology are considered for data processing, and a series of experiments are conducted by these systems. Finally, the data management system for the heterogeneous data warehouse is developed, including the modules as functions to process Big Data by these technologies. The proposed system has been designed specifically for the concrete domain, and the results may vary depending upon the domain and structure of data.

REFERENCES

1. *Artemov S.* Big Data = Big Problem? // Jet Info. 2012. No. 6; http://www.jetinfo.ru/jetinfo_arhiv/big-data/bolshie-dannye-bolshaya-problema/2012.
2. *Vaish G.* Getting Started with NoSQL. Packt Publ. 2013. 142 p.
3. Oracle Database Concepts // Oracle Database Online Documentation 12c Release 1. http://docs.oracle.com/database/121/CNCPT/intro.htm#CNCPT001.
4. *Erickson J. et al.* New Benchmarks for SQL-on-Hadoop: Impala 1.4 Widens the Performance Gap. 2014. http://blog.cloudera.com/blog/2014/09/new-benchmarks-for-sql-on-Hadoop-impala-1-4-widens-the-performance-gap.