Розглянуто підхід до підвищення продуктивності розподілених інформаційних систем на основі спільного використання технологій кластера Hadoop та компонента PolyBase SQL Server. Показано, що актуальність вирішуваної в роботі проблеми пов'язана з необхідністю обробки великих даних, що мають різний спосіб подання відповідно до рішення різнопланових задач бізнес-проектів. Проведено аналіз методів та технологій створення гібридних сховищ даних на основі різних даних типу SQL та NoSQL. Показано, що в даний час найбільш поширеною є технологія обробки великих даних з використанням середовища розподілених обчислень Hadoop. Проаналізовано існуючі технології організації та доступу до даних в кластері Hadoop із SQL-подібних СУБД за допомогою конекторів. Наведено порівняльні кількісні оцінки використання конекторів Hive та Sqoop при експорті даних у сховище Hadoop. Проведено аналіз та особливості обробки великих даних в архітектурі розподілених кластерних обчислень на базі Hadoop. Наведені та описані особливості технології PolyBase як компонента SQL Server для організації моста між SQL Server та Hadoop даних типу SQL та NoSQL. Наведений склад модельної обчислювальної установки на базі віртуальної машини для спільного налаштування PolyBase та Hadoop для рішення тестових завдань. Розроблено методичне забезпечення установки та конфігурування програмного забезпечення Hadoop і PolyBase SQL Server з урахуванням обмежень на обчислювальні потужності. Розглянуто запити для використання PolyBase та сховища даних Hadoop при обробці великих даних. Для оцінки продуктивності системи запропоновано абсолютні та відносні метрики. Для тестових даних великих об'ємів приведені результати експериментів і проведений їх аналіз, що ілюструє підвищення продуктивності інформаційної системи – часу виконання запитів і величини тимчасових таблиць, що створюються при цьому. Проведений порівняльний аналіз досліджуваної технології з існуючими конекторами з кластером Hadoop, який показав перевагу PolyBase над конекторами Sqoop та Hive. Результати проведених досліджень можуть бути використані при проведенні наукових і тренінгових експериментів для вдосконалення бізнес-процесів організацій при впровадженні надсучасних IT-технологій

Ключові слова: Hadoop, MapReduce, HDFS, PolyBase SQL Server, T-SQL, розподілені обчислення, масштабування, масштабована група PolyBase, зовнішні об'єкти, Hortonworks Data Platform

UDC 004.75
DOI: 10.15587/1729-4061.2018.139630

ENHANCING THE PERFORMANCE OF DISTRIBUTED BIG DATA PROCESSING SYSTEMS USING HADOOP AND POLYBASE

S. Minukhin
Doctor of Technical Sciences, Professor*
E-mail: minukhin.sv@gmail.com
V. Fedko
PhD, Associate Professor*
E-mail: vvfedko@gmail.com
Y. Gnusov
PhD, Head of Department
Department of cybersecurity
Kharkiv National University of Internal Affairs
L. Landau ave., 27, Kharkiv, Ukraine, 61080
E-mail: duke6969@i.ua
*Department of Information Systems
Simon Kuznets Kharkiv National University of Economics
Nauky ave., 9-A, Kharkiv, Ukraine, 61166

1. Introduction

Modern data processing technologies address the challenges associated with scaling, flexibility of using different tools, access time and data query processing rate. It is necessary to store and process Big Data. Big Data are characterized by specific properties, which include their capacity – terabytes or petabytes of data; high real-time processing rate and natural diversity of their types, structured and unstructured by nature [1, 2].

Typicaly, different types of data are used to solve problems that are diverse in a contextual sense. For example, the existing database of activity of a company contains large amount of structured data. In contrast to it, operational information relating to market behavior and market trends has an unstructured form. In addition, with increasing capacities of stored and operational data, there appears a problem of unloading and using "cold" data from warehouses, capacities of which are terabytes.

One of the relevant directions of development of modern information systems is the transition to high-performance environments that uses distributed computing – grid systems, cloud platforms (Azure, Amazon, Google) and so on., ensuring enhancing efficiency of computation processes at the expense of scalability and distributed data warehouse technologies. The most common in modern scientific research are platforms Apache Hadoop (Hortonworks (HDP)), Cloudera Distributed Hadoop (CDH)), as well as software tools for working with clusters – Apache Sqoop, Apache HBase, and in cloud platforms environments (Elastic MapReduce – AWS, HDInsight-MS Azure). They are based on the project Apache Hadoop – a distributed programming environment and a data warehouse. With a view to improving the productivity of this ecosystem, the Ma-

pReduce software framework that provides implementation of such important characteristics in terms of operation of a complex system, such as ease of setting, fault tolerance, and scalability, was developed.

The problem of processing different types of data is the following. When data of one type are stored in bases, for data exchange, it is enough to specify a data provider library in a code and set a connection string for each database. For example, for SQL Server database, the data provider support is performed by specifying the namespace System.data.SqlClient, and connection strings and in the simplest case have the form, shown in [3, 4]. If data are stored in databases of different types (for example, NoSQL), the situation is complicated, the name spaces for the relevant data providers are determined in the code.

One of the key points of a solution to the problem of dealing with Big Data is the use of NoSQL technology. In the case where the type of NoSQL database is used, the application of the SQL language is impossible, except for libraries for specific data providers and connection strings. Comparative analysis of performance of the considered data types is presented in reviews [5–7], specific features of transition from using SQL data type to NoSQL data are considered in [8, 9].

Thus, the problem is to find possibilities for sharing different types of databases at implementation of industrial and research projects. Research into the areas of human genome [10], education [11], machine learning [12], solutions of engineering problems [13], social studies [14], and business [15] can be considered as examples.

The main directions in overcoming the considered problems in sharing the DB of SQL and NoSQL types are the development of technologies for bridges between them [16, 17], building integrated platforms for sharing different data types [18], converters (adapters) queries for SQL- and NoSQ-data interaction [20, 21], designing bridges when organizing warehouses of different types [19, 22, 23].

The latter of the mentioned are related to the creation of hybrid warehouses, constructed based on sharing distributed data warehouses, for example, based on Hadoop cluster, and industrial data warehouses (IDS (EDW, Enterprise Data Warehouses) [22, 23]. IDS typically use common parallel databases that support complex processing, updating and transaction of SQL queries. In this case, many companies store data in a distributed form with the possibility to process them in different ways – by intellectual data analysis, real-time SQL-queries, etc. In this connection, the technologies, which make it possible to process simultaneously data of large capacities and of different types in combination with technologies of distributed high-performance computing, are relevant.

The considered variety of tools and technologies which should be used simultaneously in applications, both in the process of their development and deployment, results in considerable difficulties in their application. That is why at the end of 2017, there emerged the concept of "translytical", introduced in the Forrester Wave report [24]. It defines the incorporation of an operating database and a data warehouse on one platform, thus, developing the concept of data warehouses construction.

It therefore seems relevant to develop approaches aimed at improving performance of an information system when working with Big Data of different types by creating a bridge between databases, stored in the distributed system Hadoop, and DBMS MS SQL Server. Using the distributed file system (HDFS) in Hadoop cluster, on the one hand, and the SQL-type database, on the other hand, allows creation of hybrid warehouses. This approach gives an opportunity to control on one platform large amounts of diverse information, operative and analytical data, at the expense of their distributed storage and processing. That is why, its application is also promising to solve the problems in modern data processing systems HTAP (Hybrid Transactions and Analytics Processing) [24]. Selection of MS SQL Server as operating database is caused by a high rating of MS SQL Server among the most popular operating DBMS in the world [25]. According to Gartner research, it has taken leading positions [26] in the world market of appropriate technologies.

It should be noted that developing modern data processing technologies in the framework of the studied performance of sophisticated data processing systems use methodologies for continuous deployment, development and integration (DevOps). In this connection, it should be noted that the methodology for deployment of separate components of software systems and frameworks for Big Data processing is an integral part of these methodologies, which is explained by the following factors: scalability; the need for continuous updating of both software and hardware parts of a system [27]. This, in turn, requires development of high-quality methodological support for implementation of all stages of installation and configuration of software and hardware platform of the developed and studied system.

2. Literature review and problem statement

Modern tendencies in the use of databases imply the solution of a relevant problem – where to store these applications if part of it is relational and another part is of NoSQL type [16]. Splitting data between SQL and NoSQL databases requires management of some data sources. One way to bridge this gap between SQL and NoSQL is to create an abstraction level for both types of databases that behave as a single database. In paper [17], the NoSQL data are converted into the triple format and are included in the SQL database as a virtual relationship. For each query, SQL extension, including a NoSQL query template, is used. This template describes the conditions for NoSQL data and allows the rest part of the SQL query to refer to the corresponding NoSQL data through variable links.

To enhance performance of data processing systems, the possibilities of high-performance computing (HPC) systems, in particular, the cluster system Hadoop, built on the distributed file system HDFS, are widely used nowadays. This architecture allows using capabilities of SQL-oriented DBMS with possibilities of high-performance computation, a short review of which is considered below.

Sqoop (SQL-to-Hadoop, Sqoop) [29] is intended for interaction of relational databases and Hadoop. Sqoop is used to transfer the relational tables to/from HDFS and generation of classes to Java that allow MR to interact with the data from the relational DBMS import utility.

Hadapt (HadoopDB) [30] is the system, designed to support execution of SQL- queries for unstructured and structured data. In Hadapt, one copy of DBMS PostgreSQL deploys at each Hadoop node. Strings of relational tables are hash- elements placed on PostgreSQL instances on all

cluster nodes. HDFS system is used as a warehouse of unstructured data.

Hive [31] represents an abstraction above MR. It enables us to query data without development of MR tasks. To retrieve data from the warehouse, Hive applies SQL-like queries in the Hive language Query Language (HiveQL).

Apache PIG [32] is the platform for representation of stream data through the high-level language Pig Latin; it executes a series of MR tasks, managed by Hadoop.

Apache HBase [33, 34] is a scalable NoSQL database, running above the HDFS system. For MySQL data, Phoenix, creating clones of MySQL and SQL tables for an access to HBase, was developed. Instead of creating tasks in MR, Phoenix interacts with HBase with the coprocessor and thus reduces the query execution time.

Apache Cassandra [35] is a distributed database, designed to process Big structured data, using CQL language [36] for creating SQL-queries. Its idea is to develop a SQL-query language on top of Cassandra, bypassing the SQL interpreter as a whole due to the lack of compatibility with the SQL code.

Mongodb-Hadoop Connector (MDBHC) [37] is a connector to work with unstructured data in Hadoop HDP 2.1 platform. The connector represents MongoDB as the Hadoop-compatible file system, allowing the MR task to be read directly from the MongoDB without prior copying from HDFS, and thereby reduce communication delay. In addition to the existing MR, Pig, Hadoop Streaming and Flume streams, MDBHC launches SQL queries from the Hive by MongoDB data. The latest MDBHC version allows the Hive to access BSON hives with full support for MongoDB collections. MDBHC provides integration with MR tasks – the data aggregation from some input sources, or as a part of the Hadoop-based data warehouse or operating processes of ETL (SSIS).

The data adapter system, [38] is an architecture, which includes four components: a relational database, the NoSQL database, DB Adapter and DB Converter. The system is the coordinator between the applications and two databases, controlling the query stream and transformation processes. DB Converter is responsible for data conversion and reports conversion in the DB adapter for their subsequent processing.

Oracles SQL Connector for Hadoop [39], Greenplum [40] and Asterdata [41] use the external table mechanism, which provides a "relational" form to the data, stored in HDFS.

The use of high-performance distributed environment implies the existence of an interface between the traditional organization of SQL-queries and unstructured databases. In the future, the possibilities of PolyBase SQL Server to create a hybrid warehouse based on Hadoop cluster are explored in this capacity.

3. The aim and objectives of the study

The aim of present research is to increase performance of distributed information systems by sharing Hadoop computational cluster and PolyBase SQL Server.

To accomplish the aim, the following tasks have been set:

– to perform quantitative analysis of performance of SQL-oriented DB connectors with the Hadoop cluster when working with Big Data;

– to analyze the characteristics of Hadoop architecture and organization of interaction with the PolyBase connector when working with Big Data;

– to consider and describe the main types of queries that are used to access the database in PolyBase SQL Server;

– to develop methodical support for deploying and configuring joint running of Hadoop and PolyBase;

– to carry out experiments and analyze their results based on quantitative estimates of the developed metrics of performance of the studied data processing technology.

4. Technology of Big Data processing in the distributed system of Hadoop

One of the most high-performance systems for processing various types of data is the Hadoop cluster, designed for processing large data volumes (10 GB or larger [42]), stored in a large number of files.

In this article, Haddop is used as a warehouse of hybrid type data. In it, reference books are stored in the relational database, and operative data are loaded from Web sites to the NoSQL warehouse in the transformed and cleared form (Fig. 1). The fact that Hadoop does not support transactions results in unit-by-unit addition and reading big arrays. Warehouse in the Hadoop technology is provided by the distributed file system (HDFS), in which units of files are distributed among thousands of nodes in the cluster.

The data, presented in the form of units of 64, 128 and 256 MB, are in the DataNode nodes (called data nodes). If a file size exceeds the selected unit, the data that were not placed in it, fall to the next one. In the NameNode management node, files' addresses are recorded in a tree-like warehouse system, as well as in metadata files and directories (Fig. 2).

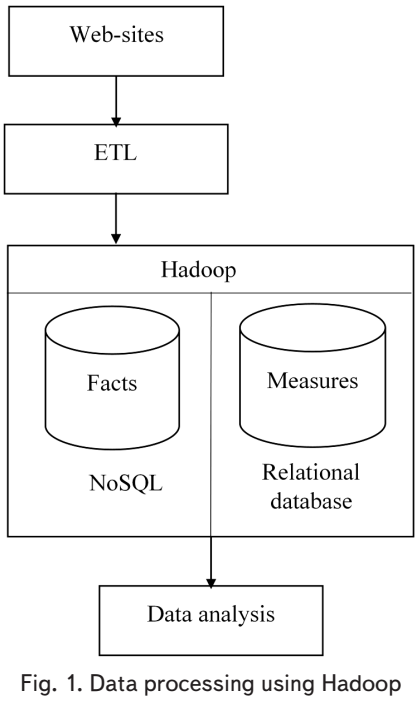

Fig. 1. Data processing using Hadoop

NameNode node also performs a reading operation. First, the nearest unit with necessary data is read, after which the data are selected from it. Several applications can run

with these data simultaneously, which makes it possible to increase the scalability of the system.

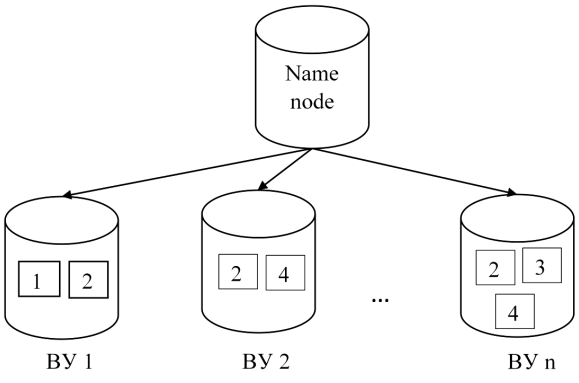

Fig. 2. The structure of HDFS file system

Distributed computations in Hadoop are constructed in the MapReduce (MR) framework. Two stages of the computational process are implemented in it:

– at stage 1, required data (Map) are selected;

– at stage 2, they are used for calculations (Reduce).

Since data are placed at different units at different servers in the cluster, the operations, conducted at stage 1, are executed in parallel. This organization reduces the total query execution time (Fig. 3) and, thus, improves the service quality in a distributed system.

The Map stage creates the "key-value" pairs, which are shuffled by matching key values and, if necessary, and sorted [43]. This stage can be simultaneously implemented on a large number of computers (nodes), including some intermediate computations in order to reduce the amount of transmitted data.

The feature of the given technology is that it includes a chain of multiple MR processes. With an increase in the number of elements in the chain and the number of computers in a cluster, there occurs a problem, associated with scalability of the architecture. To solve it, framework YARN [44], designed to manage cluster resources and parallel execution of tasks, is used. In this case, the program provides isolation of these tasks, which is an analogue of transactions in relational databases.

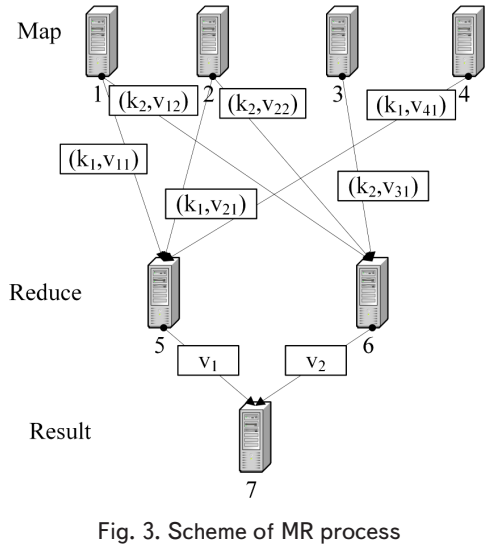

Fig. 3. Scheme of MR process

The distributed application, running with YARN, should have a dedicated management class, it is responsible for synchronization of tasks. In general form, the data processing scheme using YARN and MR can be seen in Fig. 4.

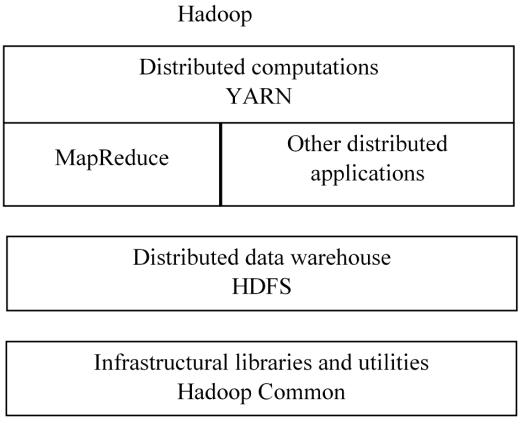

Fig. 4. Structural scheme of Hadoop MapReduce

Described scheme proved to be effective to address the problem of scalability: its feature is not the vertical scalability by increasing the capacity of nodes, included in a cluster, but the horizontal. In it, elasticity is realized by connecting additional clusters (computers). A similar problem also arises when using NoSQL-type databases [45], which is solved through sharding and replication procedures. In Hadoop, sharding is compared with data replication, as well as splitting files into units. In the absence of transactions, reliability (fault tolerance) of computational processes is ensured.

Thus, Hadoop MR together with YARN planner ensures scalability and fault tolerance of the distributed system due to the procedure of automatic saving of several copies of all data on different cluster nodes. In case of disconnection of a node during data processing, tasks are forwarded to other nodes in the cluster.

5. Analysis of performance of SQL and Hadoop connectors

Data transfer at Hadoop is determined by the following factors:

– data volumes;

– performance of input/output devices;

– performance of a computer network;

– magnitude of migration parameter;

– Hadoop cluster performance (selected platform for installation, configuration, and operation).

When moving Big Data, it seems necessary to test and adjust separate parts of data streams to enhance the system performance, allowing reduction of the cost of their transmission. It also requires a well-grounded choice and installation of software components for the selected architecture (processing mode) of the Hadoop cluster to ensure the required performance of the system.

Below, there is an example of results of testing performance of data migration from MySQL tables to Hadoop using Sqoop (Table 1) [46].

Table 1

Testing the performance of export from MySQL to Hadoop using Sqoop

General capacity of table	7 GB	
Memory capacity of AWS RDS instance	600 MB	
Configuration of Hadoop	1 Name Node, 2 Data Nodes	
Unit's capacity in Hadoop	256 MB	
Capacity of random access memory of AWS instance (m1.large)	7,5 GB	
Test		
Command	Mappers: 2, Reducers: 1, mapred. child.java.opts : 550 MB	17 min.
	Mappers: 4, Reducers: 1, mapred. child.java.opts : 550 MB; sqoop -m 8 .	12 min
	Mappers: 4, Reducers: 1, mapred. child.java.opts: 550 MB, sqoop -direct	5.54 min
	Mappers: 4, Reducers: 1, mapred. child.java.opts : 550 MB, sqoop -direct, -compression	4.37 min

Thus, for tables with the specified size, depending on the features and settings of the system and the used command types, it is possible to achieve improvement of data migration performance in the range of 140–390 % (1.4–4 times) by means of the Sqoop connector. Other possible settings to improve performance of the Sqoop connector are presented in paper [47].

Below, there is an example of results of testing the performance of data migration from MySQL tables to Hadoop using Hive (Tables 2, 3) [48].

Table 2

Testing performance of data export from MySQL to Hadoop using Hive

General capacity of the table	200 MB–10 GB
Memory capacity	8 GB
Configuration Hadoop	1 Name Node (8 2,4 GHz Xeon), 32 Data Nodes (Intel Core 2 Duo 3 GHz)
Unit capacity in Hadoop	128 MB
Capacity of random access memory	Name Node – 8 GB, Data Node – 2 GB
Computer network	Gigabit Ethernet

Table 3

Comparative analysis of query execution time for different data volume

Number of entries	Volume, MB	MySQL, s	Map-Reduce, s	Hive, s
500	235	4.20	81.14	535.1
1,000	475	13.83	82.55	543.64
2,500	1,000	85.42	84.41	548.45
5,000	2,000	392.42	83.42	553.44
10,000	5,000	1,518.18	88.14	557.51
15,000	7,000	1,390.25	86.85	581.5
20,000	9,000	2,367.81	88.90	582.7

Thus, with an increasing volume of transferred files starting with 5 GB for the configuration in the form of 1 node-master and 32 working nodes, there is a steady tendency of a decrease in query execution time with the use of the Hive connector in the range of 270–400 % (by 2.7–4 times). Without the use of additional settings for the table capacity of 7 GB, similar to Sqoop connector, the use of Hive allows a decrease in query execution time up to 9.7 min, which corresponds to results of test 2 for Sqoop.

It can be concluded from the shown results that an increase in performance of the information data processing system in conjunction with distributed computations depends on the configuration of the cluster and its selected operation mode, the capacity of data to export as well as the limitations (requirements) in relation to promptness of query execution and memory. They can be assigned in the form of metrics – absolute or relative indicators, reflecting distinctive features (characteristics) of the considered technology.

6. Technology of connector PolyBase SQL Server – Hadoop

The considered solutions of the interface with Hadoop are extended by the component SQL Server 2016 – PolyBase [49], which uses T-SQL.

It is necessary to separate the following from the operations with PolyBase:

1) queries from SQL Server to Hadoop;

2) export/import from SQL Server to Hadoop.

The latter two are the ETL-operations for warehouse organization: the first one is used to analyze the data, stored in a hybrid warehouse, the other one is used to send data to data marts, analysis and reporting.

Polybase is the functional system of the parallel database system (Parallel Data Warehouse, PDW), including parallel query optimizer and the mechanism for their implementation. The system uses MapReduce as an "auxiliary" mechanism for data processing queries, downloaded from HDFS (Fig. 5, 6).

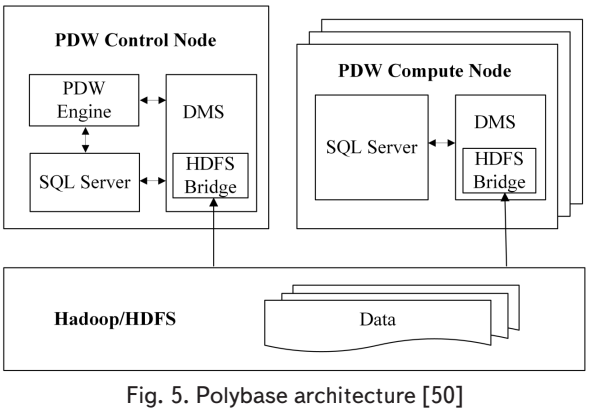

Fig. 5. Polybase architecture [50]

The paradigm "query separation" was implemented in Polybase for scalable query processing [50, 51]: for structured data – in relational tables and for unstructured data – in HDFS.

The place of PolyBase in data processing together with Hadoop is shown in Fig. 7.

The main advantage of PolyBase is that it makes it possible to apply all analytical Microsoft tools for the data,

stored in Hadoop. We should separate simpler ones for an end user – Excel and PowerBI with PowerPivot and Power Query, and professional tools, such as Integration Services, AnalysisServices, Reporting Services and Machine Learning Services, oriented to developers.

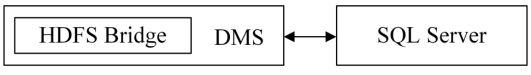

Fig. 6. HDFS: instance of a bridge in computational node of PDW [51]

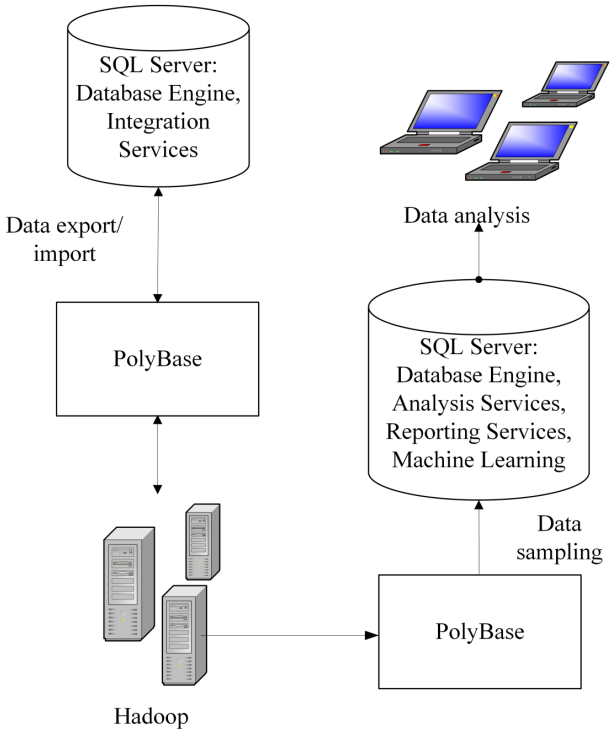

Fig. 7. Schematic of data processing using PolyBase-Hadoop

7. Software and hardware environment

The developed system was implemented on PC with HDD 1 TB, RAM – 16 GB, processor Intel® Core™ i5-7600, x86-64, 4 core×3,5 GHz, SmartCache L3 6 MB, Intel® Turbo Boost 2.0.

To deploy software of the Hadoop system, we used: guest OS: virtual machine Virtual Box 5.2 (Oracle), RAM – 4 GB, 2 core ×3,5 GHz; Hadoop 2.6 (Hortonworks HDP 2.2), OS – CentosOS 7; autonomous mode (at a single machine (server)) (I/O files at a single machine (server)).

To deploy Polybase component, MS SQL Server 2017 Enterprise for OS Windows 10 was used.

8. Development of methodological support for deployment of Hadoop-PolyBase

Stage 1

Deployment and setting of Hadoop.

The most common are distributives from companies Cloudera, Hortonworks, MapR, IBM and Pivotal [52]. For CLOUD-platforms, Elastic MapReduce (Amazon Web Services) and HDInsight (Microsoft Azure) are used.

It should be noted that currently PolyBase is installed on platforms HDP and CDH [52].

Stage 2

Installment and configuration of SQL Server 2016 (and above – Enterprise or Developer).

Stage 3

Installment of PolyBase (performed at configuring SQL Server 2016, 2017).

Stage 4

Connection of PolyBase to Hadoop as a data source.

It includes two stages: configuring the server and creation of external data system (EDS). A change in configuration is implemented by sp_configure and RECONFIGURE.

Stage 5

Connection of YARN and MR to Hadoop .

The first component is connected by assigning the key value to configuration yarn.application.classpath of same name property of SQL Server, the second one – by setting configuration parameter mapreduce.application.class¬path (file yarn.site.xml).

Stage 6

Authentification in Hadoop.

SSL protocol is used for authentification and transfer of encrypted data between the nodes. Configuration authenticate (authenticity check) is selected by default. To provide a high protection level, it is necessary to assign the value integrity in property hadoop.rpc.protection, and the highest protection level – the value privacy (confidentiality).

Stage 7

Scalability of PolyBase.

It is provided by connection of PolyBase instances, included in one domain – a scalable group [53]. In domain, one instance is a managing node (Software – SQL Server 2016, PolyBase Engine, PolyBase DMS) (MN), the rest are computational (software – SQL Server 2016, PolyBase DMS)) (CN) (Fig. 8).

It is implemented by the following algorithm.

Step 1

MN of SQL Server receives a query.

Step 2

MN converts a query into implementation plan (DSQL plan) – a sequence of SQL procedures (DSQL), performed on instances of DB on CN, and DMS operations (Data Movement Service) to transmit data between CN.

Step 3

The PolyBase kernel in it parallelizes the query implementation.

Stage 8

Creation of objects.

To use PolyBase, it is necessary to create the external table (ET), format of external file (EF) and EDS.

The EDS object determines Hadoop location and, probably, accounting information for EDS authentication. For example, data source named HadoopCluster1 on the cluster, connected to port 8050 of the computer with IP-address 10.10.10.10, is created by the following script (Fig. 9).

In the EF, it is necessary to indicate the file type (Parquet, Hive ORC, Hive RCFile and Delimited Text) in the data source, determined earlier and for a text file to set: the field end attribute, line spacer, date format, file compression method, etc. For example, if Hadoop contains Parquet-files, compressed by Gzip method, it is necessary to set value FORMAT_TYPE = PARQUET in format description.

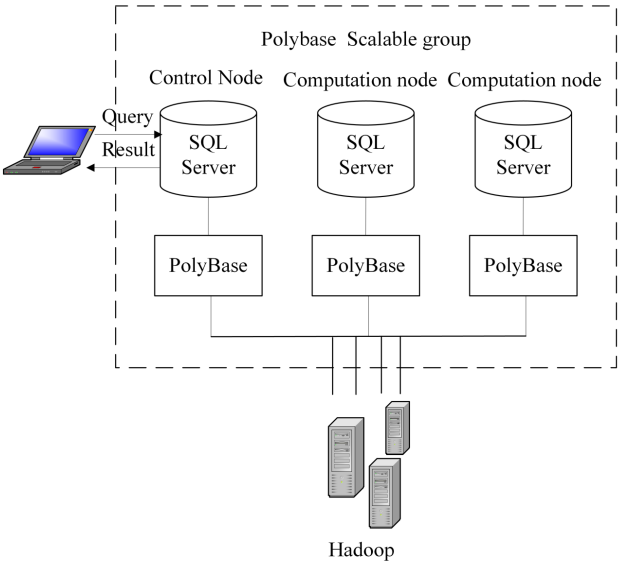

Fig. 8. PolyBase Scalable group

```
CREATE EXTERNAL DATA SOURCE HadoopCluster1
WITH (
TYPE = HADOOP,
LOCATION = 'hdfs://10.10.10.10:8050'
)
```

Fig. 9. Creation of an external data source

The ET object displays an object of data warehouse from Hadoop in PolyBase: tables of relational and document-oriented database and their representation.

For work with ET in a correct way, it is necessary to take into account in its description that the number of its columns and their types must exactly match the correspondent parameters of the table scheme within Hadoop. It is possible to postpone giving messages by setting in the reject_options parameter an allowable number of "failed" lines or their percentage. This approach becomes relevant in the case where the data contained in the Hadoop cluster are contained in the document-oriented database that uses flexible scheme of collections.

Thus, for example, to retrieve sales data from the fact Sales table that is stored in Hadoop (data source – HadoopCluster1) and has the format ParquetGzip1, the ET of HDP_factSales is created by the script (Fig. 10).

```
CREATE EXTERNAL TABLE HDP_factSales
(
   [DateKey] INT,
   [ManufacturerKey] INT,
   [ProductKey] INT,
   [Quantity] SMALLINT,
   [Cost] MONEY
)
WITH (
   LOCATION = '/apps/hive/DW/factSales',
   DATA_SOURCE = HadoopCluster1,
   FILE_FORMAT = ParquetGzip1,
   REJECT_TYPE = value,
   REJECT_VALUE=0
)
```

Fig. 10. Creation of external file format

It should be noted that ET is a reference to the data stored in Hadoop for performing Insert, Select, Join operations, etc. Thus, in its functional features it is close to representation in a relational database, which increases the possibility of a user working with data.

For example, the output of data on sales of goods by months is implemented by the script (Fig. 11).

```
SELECT d.Year*100+d.MonthNumberOfYear as YearMonth, SUM(s.Cost) as
MonthCost
  FROM DimDate as d INNER JOIN HDP_factSales as s
    ON d.DateKey = s.DateKey
  WHERE s.Cost > 200
  Group By (d.Year*100+d.MonthNumberOfYear)
```

Fig. 11. Script of data output

This script used connection ET – HDP_factSales with the local table DimDate with data, grouped by months, as well as data filtering on field Cost ET. To speed up the latter two operations, the MR tasks are enabled.

The import operation into a local table will be demonstrated using the example of transfer of sales data for 2018 from the ET of HDP_factSales to the local table factSales2018. It is implemented by the script (Fig. 12).

```
SELECT *
  INTO factSales2018
  FROM  HDP_factSales
  WHERE HDP_factSales.DateKey BETWEEN 20180101 AND 20181231
```

Fig. 12. Script of import operation into a local table

For archiving outdated data for 2016, the export from the factSales table in HDP_factSales2016 was performed by the script (Fig. 13).

```
-- Creation of the external table
CREATE EXTERNAL TABLE [HDP_factSales2016]
(
   [DateKey] INT,
   [ManufacturerKey] INT,
   [ProductKey] INT,
   [Quantity] SMALLINT,
   [Cost] MONEY
)
WITH (
   LOCATION='/OldData/2016/SalesData',
   DATA_SOURCE = HadoopCluster2,
   FILE_FORMAT = TextFileFormat,
   REJECT_TYPE = VALUE,
   REJECT_VALUE = 0
);
-- Data export
INSERT INTO HDP_factSales2016
SELECT *
   FROM factSales
   WHERE factSales.DateKey BETWEEN 20160101 AND 20161231
```

Fig. 13. Script of data export

The productivity of the developed system can be improved based on the T-SQL language that allows:

– reading data from Hadoop;

– combining them with the data, stored in database of SQL Server;

– importing and exporting data between the local database and Hadoop warehouse.

The examples of developed requests (Fig. 9–11) that implement the core operations of SQL Server and Hadoop interactions are represented above and further used when solving test tasks. The syntax of basic requests is represented in Microsoft documentation [54–56].

Activity diagram in UML modeling language of implementation of the developed methodical support is shown in Fig. 14.

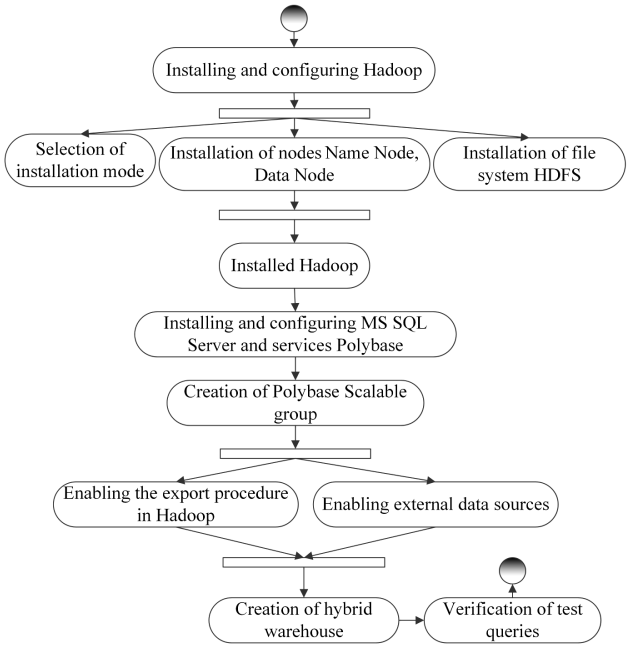

Fig. 14. UML-activity diagram of PolyBase-Hadoop deployment

The methodological support, considered in the framework of the existing methodologies for continuous deployment and integration of complex systems, enables formation of a structure of business process of analytical processing of Big Data of different types in the face of operation of high-performance cluster with developed interface tools. This makes it possible to reduce the correspondent cost to customize the software and hardware part of the studied system and thus to enhance performance of the data processing system in general.

9. Experimental studies into estimating the effectiveness of Big Data processing using PolyBase-Hadoop

It should be noted that the results of the experimental studies, obtained from the use of the studied technology were not adequately displayed in modern literature due to the fact that this technology was presented by Microsoft in mid-2017. In this regard, the following results are actually some of the first research into this technology [50, 51].

This section describes and analyses experimentally obtained estimates of effectiveness of application of the PolyBase component and the Hadoop cluster for applications with big data of different types.

The studies were conducted according to the following methodical scheme.

1. Transferring data from Hadoop into a temporary table in SQL Server for their subsequent processing in PolyBase. Connection of Hadoop nodes was carried out by the predicate pushdown mode (option in SELECT proposal [51]). When predicate pushdown is forced into the script, we add (Fig. 15), otherwise (Fig. 16)

```
OPTION (FORCE EXTERNALPUSHDOWN)
```

Fig. 15. Script of forcing predicate pushdown

```
OPTION (DISABLE EXTERNALPUSHDOWN)
```

Fig. 16. Script of disabling predicate pushdown

2. Test datasets

Files with information about the customers of one of the companies were selected for testing. Data for research were generated randomly through the site [57]. The number of entries in each of the used files is shown in Table 4.

Table 4

Number of entries and memory capacity of occupied test files

File name	Number of entries	Memory capacity, MB
Customers_1	50,000	30
Customers_2	200,000	80
Customers_3	500,000	200
Customers_4	1,000,000	400
Customers_5	1,300,000	520
Customers_6	1,500,000	600

3. Deployment of platform for data placement.

These files were places in Hadoop Hortonworks Data Platform 2.2 for the OS CentOS. The work with database was implemented in SQL Server Management Studio 2017.

4. Connection.

To connect SQL Server to Hadoop, EDS was created by the script, in which for correct operation of the predicate pushdown option, parameter RESOURCE_MANAGER_LOCATION, setting the address of Hadoop resource manager, was forced [43].

5. Creation of the ET.

ET was created for each file in the Hadoop. For example, for the file Customers_1, its scheme is represented by the following script (Fig. 17).

```
CREATE EXTERNAL TABLE [dbo].[Customer_1]
(
        Id int NOT NULL,
        FirstName nvarchar(100),
        LastName nvarchar(100),
        Gender nvarchar(100),
        Zip int,
        State nvarchar(100),
        City nvarchar(100),
        Birthday date
)
WITH (DATA_SOURCE = [HDP2],
        LOCATION = N'/user/hadoop/customers/customers_1',
        FILE_FORMAT = [CSV],
        REJECT_TYPE = VALUE,
        REJECT_VALUE = 1
)
```

Fig. 17. Script of creation of external table

6. Generation of queries.

Conducted detailed analysis of functions to improve the performance of data export operations in Hadoop showed that it is necessary to select the predicate pushdown. The main purpose of this function is to reduce the number of lines to be moved, to decrease the number of columns to be moved and to use subsets of expressions and operators to create and optimize the external table. A query can have some predicates that can be transferred to a Hadoop cluster.

For experiments, we used a query, the result of implementation of which is all customers born after 1980. For the ET, THE query was implemented in two modes: with forced predicate pushdown option (Fig. 18) and disabled predicate pushdown option (Fig. 19) scripts, respectively:

```
SELECT * FROM [dbo].[Customer_{N}]
    WHERE YEAR(Birthday) > 1980
    OPTION (FORCE EXTERNALPUSHDOWN);
```

Fig. 18. Script of forcing the predicate pushdown mode

```
SELECT * FROM [dbo].[Customer_{N}]
  WHERE YEAR(Birthday)>1980
  OPTION (DISABLE EXTERNALPUSHDOWN).
```

Fig. 19. Script of disabling predicate pushdown mode

7. Performance metrics.

The following were used as performance metrics:

absolute: query execution time, capacity of memory, required to create temporary tables;

relative: the ratio of query execution time to capacity of transferred tables, the ratio of capacity of temporary tables to capacity of transferred tables.

To carry out the experimental study and to obtain quantitative estimates of performance improvement, the following queries were implemented.

To determine the memory, required for temporary tables, it is necessary to use the query to the systemic database tempdb. It should be executed together with the main queries due to the fact that upon its sampling completion all temporary tables, associated with it, are removed from the database.

Quantitative estimations of query execution time and required memory of temporary tables for used test files are shown in Table 5.

Table 5

Results of queries execution

File name	Execution time, s		Memory capacity, MB	
	DISABLE	FORCE	DISABLE	FORCE
Customers_1	1	3	2.6	1.5
Customers_2	3	3	10.0	6.0
Customers_3	7	4	25.6	11.0
Customers_4	15	6	52.0	23.0
Customers_5	18	8	59.0	25.0
Customers_6	21	10	64.0	28.0

The results of research show that at smaller data volume, query execution time using the pushdown option is somewhat less than in the control sample. However, with an increase in the amount of processed data, queries with forced predicate pushdown option are executed in considerably less time than without it (Fig. 20).

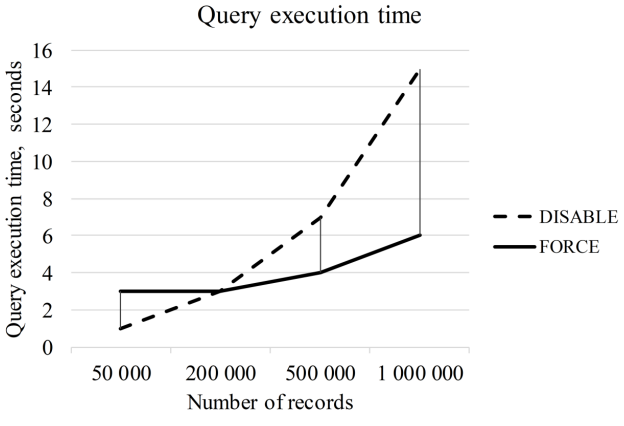

Fig. 20. Dependence of query execution time on the number of entries with the use of the predicate pushdown option and without it

The action of the forced predicate pushdown option has a significant impact on the magnitude of volume of used memory to create temporary tables, because only the data that meet selection criteria return to SQL. It should be noted that in this case, less memory to store such data in a temporary table is required, which is demonstrated in Fig. 21.

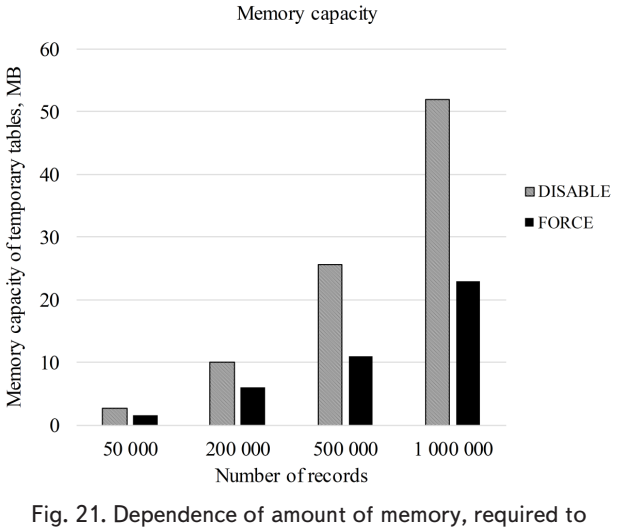

Fig. 21. Dependence of amount of memory, required to execute records, on the number of records with the use of predicate pushdown option and without it

For quantitative estimation of performance of the studied technology with the forced predicate pushdown function, new metrics were additionally proposed – the ratio of query execution time to memory capacity of moved tables (Table 6, Fig. 22) and the ratio of memory of created temporary tables to capacity of moved tables (Table 7, Fig. 23).

Table 6

Ratio of query execution time to table capacity

File name	Mode	
	DISABLE, s	FORCE, s
Customers_1	0.03	0.1
Customers_2	0.0375	0.0375
Customers_3	0.035	0.02
Customers_4	0.0375	0.015
Customers_5	0.034	0.0153
Customers_6	0.035	0.016

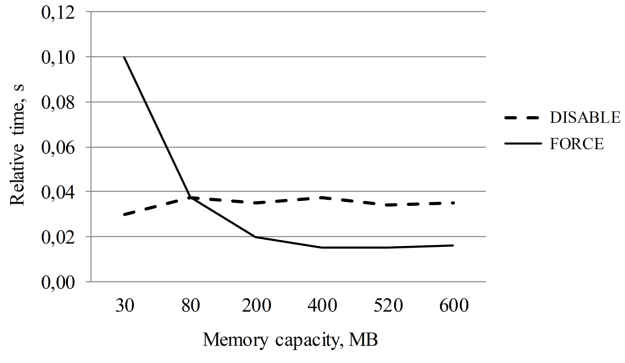

Fig. 22. Dependence of ratio of query execution time to tables capacity on tables capacity using predicate pushdown option and without it

Table 7

Ratio of temporary tables capacity to exported tables capacity

File name	Mode	
	DISABLE	FORCE
Customers_1	0.086	0.05
Customers_2	0.125	0.075
Customers_3	0.128	0.055
Customers_4	0.13	0.057
Customers_5	0.113	0.048
Customers_6	0.106	0.046

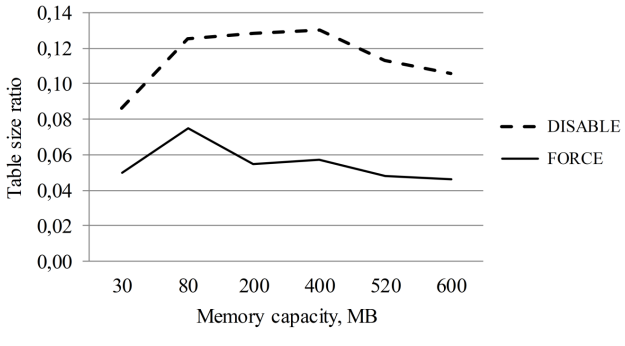

Fig. 23. Dependence of ratio of temporary tables capacity to capacities of tables using the predicate pushdown option and without it

Results of conducted experiments with the presented test data allow us to draw the following conclusions.

1. Productivity of calculations in Hadoop can be improved using the predicate pushdown option.

2. It is not expedient to disable calculations on the Hadoop cluster when processing the data of insignificant capacity (for presented test examples – less than 80 MB).

3. Forcing calculations on the Hadoop cluster is effective if an insignificant part of the entire table gets in a query result. This is due to the fact that when calculations on Hadoop are disabled, SQL Server first copies all data into temporary tables (which requires a lot of memory), and then performs filtering. In this case, the use of the predicate pushdown can significantly decrease the absolute and relative indicators of the use of resource memory (system-oriented indicator) and query execution time (providing service quality – customer-oriented indicator) (Fig. 20–23).

4. Application of the predicate pushdown option allows decreasing the cost of storing files, as in the case of the use of a cluster, data warehouse cost at its nodes is significantly less than the cost of larger files in SQL Server. Thus, in this case it is possible to use, for example, the relative indicator of total cost of storing files in nodes of the Hadoop cluster to total cost of storing files in SQL server as an additional metrics – in this case, the economic indicator of operation of the information system.

10. Discussion of results of research into performance of Polybase connector with Hadoop cluster when working with Big Data of different types

The conducted studies showed that the use of PolyBase technology in conjunction with cluster technologies to manage Big Data makes it possible to use effectively operative and analytical data simultaneously on one platform, addressing the problem within the Translytical Data Platforms. The use of PolyBase as a component of SQL Server allows increasing reliability of the entire system of data processing. In this case, the data, obtained from external sources in SQL Server, are previously stored in a relational database, then are transmitted to be stored to the file system HDFS with the use of the external table PolyBase. Despite the fact that this procedure somewhat decreases data transfer rate, effectiveness of the system as a whole increases significantly due to the use of the following procedures:

1) a log is maintained when CRUD operations are performed in a relational database, which provides additional tools of data recovery, both at soft and hard failures, due to Protocol "Write Ahead Log (WAL)";

2) processing events, related to data storage, is fully transactional. Therefore, implementation of Consistency and Durability properties ensures an increase in reliability of filling a data warehouse.

The given examples of using the PolyBase indicate quite high efficiency of the explored technology for organization of a hybrid warehouse. However, they are model and require subsequent experimental research involving practical problems in business analysis, communication, IoT and other technologies, working with Big Data at the level of an enterprise and a corporation.

The proposed methodological support of integration of the Hadoop cluster and the PolyBase component allows increasing efficiency of business processes of organizations when using modern IT-technologies within the framework of modern methodology of deployment and operation of complex systems DevOps for Big Data processing.

The work presents only basic queries. They are not enough to fully address the broad range of applied problems of data processing. Thus, it can be considered a restriction of this study. However, it should be noted that the studied technology is new (2017) and that is why it is virtually unexplored in modern scientific literature. An exception could be, for example, papers [50, 51].

In this work, for installing and configuring the Hadoop cluster, one uses the mode on a single machine, which is quite specific in itself but allows getting good qualitative and quantitative results using limited computing resources. They can serve as a basis for predictive analysis of the application of this technology in cloud platforms and server solutions with great performance. That is why the obtained results are a novelty, because under predetermined conditions, the performance of a new technology, almost not studied before, is evaluated.

The paper presents experimental studies on interaction of SQL Server and Hadoop as a unified system. The data, stored in HDFS, are processed by means of Hadoop and are transferred to SQL Server for a subsequent analysis. To do this, SQL Server 2017 has a special component In-Database Machine Learning Services. In this study, we obtained quantitative system- and customer-oriented estimates of enhancing performance of the system based on the experimental study of enabling predicate pushdown option in the MapReduce query to retrieve the lines from HDFS.

Enabling the PolyBase component for companies that have already installed the DBMS SQL Server is free of charge. Due to the high prevalence of DBMS, SQL Server nowadays is a big enough market segment of users. The main competitor of SQL Server is Oracle having similar operating functionality [58]. If a company or an enterprise does not use these DBMS and considers the license acquisition and installation of one of them, the probability of selecting SQL Server is higher due to the pricing policy of Microsoft Corp: a license for the acquisition of SQL Server costs 10 times less than that by Oracle [59–61]. Otherwise, the Oracle Big Data Connectors can be chosen as a connector [62]. The price of the license of this component for a single processor is US $2,000.00.

11. Conclusions

1. An analysis of the technology of Big Data processing in distributed information systems using different organization was performed. It was shown that the main direction for development of hybrid data warehouses is bridges between data of different types – SQL and NoSQL.

2. The review of the existing connectors for interaction with Hadoop cluster resources was carried out. Examples of quantitative estimates of application of Sqoop and Hive for different configurations of the cluster were given. It was shown, that it is possible to provide the required performance of the system considering many factors influencing it. It was concluded on the need to develop qualitative methodological support within the methodology of continuous deployment and integration of components of complex systems, specifically, to install and configure the cluster in different modes of its operation when processing different Big Data types.

3. The features of organization of data storage in the distributed system Hadoop were explored. Connection of the Task Scheduler and parallelization technologies in Hadoop makes it possible to increase the productivity of data processing systems. It was shown that existence of an interface (a bridge) between SQL data and clustered HDFS warehouse improves efficiency of working with many different data types using existing analytical tools of information processing products by Microsoft Corp.

4. An analysis of the technological platform of the PolyBase component was performed and it showed the prospects of its application in conjunction with high-performance computations. This is determined by means of horizontal scaling, the T-SQL language and support of data entry in large units, ensuring efficiency of Big Data processing in the Hadoop cluster.

5. The methodical provision of integrated deployment and configuring of the Hadoop cluster and PolyBase on the virtual HDP platform, using the operation mode “at a single machine” (pseudodistributed mode), was developed. Examples of creating external objects and examples of implementation of various types of queries for sharing these technologies were explored in detail.

6. Experimental studies were conducted for quantitative assessment and analysis of the developed absolute and relative metrics of performance of the examined system of Big Data processing. The obtained results showed that the use of the optional solutions of PolyBase and Hadoop on databases of large capacities (from 200,000 to 1,000,000 entries of the volume from 30 to 600 MB) makes it possible to decrease memory capacity, used for temporary tables, by 2 times, query execution time by 2.5 times, and memory usage ratio – from 10 % to 4 % (by 2.5 times) for the maximum capacity of the table.

7. An analysis of the developed relative metrics showed that a relative indicator of query execution time for Sqoop totaled 0.0366 and for Hive – 0.0582. Thus, they are somewhat worse than the result, obtained for test examples for Polybase – 0.015 –0.016 (Table 6).

8. Further studies could address the optimization of planning and improvement of efficiency of the MapReduce framework and databases replications. In addition, assessment of performance of homogeneous and heterogeneous Hadoop cluster are required when working at local resources with the tools of VDI (virtual desktop infrastructure), as well as on cloud platforms Azure, AWS, and Google Compute Engine.

References

1. Big Data 2.0 Processing Systems: Taxonomy and Open Challenges / Bajaber F., Elshawi R., Batarfi O., Altalhi A., Barnawi A., Sakr S. // Journal of Grid Computing. 2016. Vol. 14, Issue 3. P. 379–405. doi: https://doi.org/10.1007/s10723-016-9371-1
2. Big Data Taxonomy. BIG DATA WORKING GROUP. 2014. 33 p. URL: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Taxonomy.pdf
3. Fedko V. V., Tarasov O. V., Losiev M. Yu. Orhanizatsiya baz danykh ta znan. Kharkiv: Vyd. KhNEU, 2013. 200 p.
4. Fedko V. V., Tarasov O. V., Losiev M. Yu. Suchasni zasoby dostupu do danykh. Kharkiv: Vyd. KhNEU im. S. Kuznetsia, 2014. 328 p.

5. Priyanka, AmitPal A Review of NoSQL Databases, Types and Comparison with Relational Database // International Journal of Engineering Science and Computing. 2016. Vol. 6, Issue 5. P. 4963–4966.
6. Mohamed M. A., Altrafi O. G., Ismail M. O. Relational vs. NoSQL Databases: A Survey // International Journal of Computer and Information Technology. 2014. Vol. 03, Issue 03. P. 598–602.
7. A Survey and Comparison of Relational and Non-Relational Database / Jatana N., Puri S., Ahuja M., Kathuria I., Gosain D. // International Journal of Engineering Research & Technology. 2012. Vol. 1, Issue 6. P. 1–5.
8. Abdullah A., Zhuge Q. From Relational Databases to NoSQL Databases: Performance Evaluation // Research Journal of Applied Sciences, Engineering and Technology. 2015. Vol. 11, Issue 4. P. 434–439. doi: https://doi.org/10.19026/rjaset.11.1799
9. AH Al Hinai. A Performance Comparison of SQL and NoSQL Databases for Large Scale Analysis of Persistent Logs. Uppsala University, 2016. 111 p.
10. Evaluation of relational and NoSQL database architectures to manage genomic annotations / Schulz W. L., Nelson B. G., Felker D. K., Durant T. J. S., Torres R. // Journal of Biomedical Informatics. 2016. Vol. 64. P. 288–295. doi: https://doi.org/10.1016/j.jbi.2016.10.015
11. SQL and data analysis. Some implications for data analysits and higher education. Data Scientist Master's Program. URL: https://www.simplilearn.com/big-data-and-analytics/senior-data-scientist-masters-program-training
12. Elshawi R., Sakr S. Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service. URL: https://arxiv.org/pdf/1709.07493.pdf
13. An information modeling framework for bridge monitoring / Jeonga S., Houb R., Lynchc J. P., Sohnc H., Law K. H. // Advances in Engineering Software. 2017. Vol. 114. P. 11–31.
14. Scaling Social Science with Apache Hadoop. URL: http://blog.cloudera.com/blog/2010/04/scaling-social-science-with-hadoop/
15. Liu X. An Analysis of Relational Database and NoSQL Database on an Ecommerce Platform Master of Science in Computer Science. University of Dublin, Trinity College, 2015. 81 p.
16. Ferreira L. Bridging the gap between SQL and NoSQL. Universidade do Monho, 2012. 97 p. URL: http://mei.di.uminho.pt/sites/default/files/dissertacoes/eeum_di_dissertacao_pg15533.pdf
17. Roijackers J. Bridging SQL and NoSQL. Eindhoven University of Technology Department of Mathematics and Computer Science Master's thesis, 2012. 100 p.
18. Oluwafemi E., Sahalu B., Abdullahi S. E. TripleFetchQL: A Platform for Integrating Relational and NoSQL Databases // International Journal of Applied Information Systems. 2016. Vol. 10, Issue 5. P. 54. doi: https://doi.org/10.5120/ijais2016451513
19. Roijackers J., Fletcher G. H. L. On Bridging Relational and Document-Centric Data Stores // Lecture Notes in Computer Science. 2013. P. 135–148. doi: https://doi.org/10.1007/978-3-642-39467-6_14
20. Data adapter for querying and transformation between SQL and NoSQL database / Liao Y.-T., Zhou J., Lu C.-H., Chen S.-C., Hsu C.-H., Chen W. et. al. // Future Generation Computer Systems. 2016. Vol. 65. P. 111–121. doi: https://doi.org/10.1016/j.future.2016.02.002
21. Kuderu N., Kumari V. Relational Database to NoSQL Conversion by Schema Migration and Mapping // International Journal of Computer Engineering in Research Trends. 2016. Vol. 3, Issue 9. P. 506. doi: https://doi.org/10.22362/ijcert/2016/v3/i9/48900
22. Maislos A. Hybrid Databases: Combining Relational and NoSQL. URL: https://www.stratoscale.com/blog/dbaas/hybrid-databases-combining-relational-nosql/
23. Blessing E. J., Asagba P. O. Hybrid Database System for Big Data Storage and Management // International Journal of Computer Science, Engineering and Applications. 2017. Vol. 7, Issue 3/4. P. 15–27. doi: https://doi.org/10.5121/ijcsea.2017.7402
24. Yuhanna N., Gualtieri M. The Forrester Wave™: Translytical Data Platforms, Q4 2017. URL: https://www.forrester.com/report/The+Forrester+Wave+Translytical+Data+Platforms+Q4+2017/-/E-RES134282
25. DB-Engines Ranking. URL: https://db-engines.com/en/ranking
26. Magic Quadrant for Operational Database Management Systems. URL: https://www.gartner.com/doc/3823563/magic-quadrant-operational-database-management
27. DevOps for Hadoop. URL: http://agilealmdevops.com/2017/10/22/devops-for-hadoop/
28. Özcan F., Tian Y., Tözün P. Hybrid Transactional/Analytical Processing // Proceedings of the 2017 ACM International Conference on Management of Data – SIGMOD '17. 2017. doi: https://doi.org/10.1145/3035918.3054784
29. Apache Sqoop. URL: http://sqoop.apache.org
30. Hadapt Inc. URL: http://www.hadapt.com
31. The Apache Hive. URL: https://hive.apache.org
32. Apache Pig: Overview. URL: https://www.tutorialspoint.com/apache_pig/apache_pig_overview.htm
33. The Apache Software Foundation. URL: https://hbase.apache.org
34. Apache Hbase. URL: https://hortonworks.com/apache/hbase
35. CASSANDRA. URL: http://cassandra.apache.org
36. The Cassandra Query Language (CQL). URL: http://cassandra.apache.org/doc/latest/cql
37. MongoDB Connector for Hadoop. URL: https://docs.mongodb.com/ecosystem/tools/hadoop
38. Data adapter for querying and transformation between SQL and NoSQL database / Liao Y.-T., Zhou J., Lu C.-H., Chen S.-C., Hsu C.-H., Chen W. et. al. // Future Generation Computer Systems. 2016. Vol. 65. P. 111–121. doi: https://doi.org/10.1016/j.future.2016.02.002

39. High Performance Connectors for Load and Access of Data from Hadoop to Oracle Database. URL: http://www.oracle.com/technetwork/bdc/hadoop-loader/connectors-hdfs-wp-1674035.pdf
40. Greenplum. URL: https://greenplum.org
41. Asterdata. URL: http://www.asterdata.com
42. Shvachko K. Apache Hadoop. The Scalability Update // Login: The usenix Magazine. 2011. Vol. 36, Issue 3. P. 7–13.
43. Shuffling and Sorting in Hadoop MapReduce. URL: https://data-flair.training/blogs/shuffling-and-sorting-in-hadoop/
44. Kawa A. Introduction to YARN. URL: https://www.ibm.com/developerworks/library/bd-yarn-intro/index.html
45. Banker, K. MongoDB in Action. Manning Publications, 2012. 287 p.
46. Performance Tuning Data Load into Hadoop with Sqoop. URL: http://www.xmsxmx.com/performance-tuning-data-load-into-hadoop-with-sqoop/
47. Sqoop Performance Tuning Guidelines. URL: https://kb.informatica.com/h2l/HowTo%20Library/1/0930-SqoopPerformanceTuningGuidelines-H2L.pdf
48. Hadoop and Hive as scalable alternatives to RDBMS: a case study. URL: https://scholarworks.boisestate.edu/cgi/viewcontent.cgi?referer=https://www.google.com.ua/&httpsredir=1&article=1001&context=cs_gradproj
49. PolyBase Queries. URL: https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-queries
50. Indexing HDFS data in PDW / Gankidi V. R., Teletia N., Patel J. M., Halverson A., DeWitt D. J. // Proceedings of the VLDB Endowment. 2014. Vol. 7, Issue 13. P. 1520–1528. doi: https://doi.org/10.14778/2733004.2733023
51. Split query processing in polybase / DeWitt D. J., Halverson A., Nehme R., Shankar S., Aguilar-Saborit J., Avanes A. et. al. // Proceedings of the 2013 international conference on Management of data – SIGMOD '13. doi: https://doi.org/10.1145/2463676.2463709
52. Cloudera vs Hortonworks vs MapR: Comparing Hadoop Distributions. URL: https://www.experfy.com/blog/cloudera-vs-hortonworks-comparing-hadoop-distributions
53. PolyBase scale-out groups. URL: https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-scale-out-groups?view=sql-server-2017
54. CREATE EXTERNAL DATA SOURCE (Transact-SQL). URL: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql?view=sql-server-2017
55. CREATE EXTERNAL FILE FORMAT (Transact-SQL). URL: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-file-format-transact-sql?view=sql-server-2017
56. CREATE EXTERNAL TABLE (Transact-SQL). URL: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-table-transact-sql?view=sql-server-2017
57. Generate Test CSV Data. URL: http://www.convertcsv.com/generate-test-data.htm
58. System Properties Comparison Microsoft SQL Server vs. Oracle. URL: https://db-engines.com/en/system/Microsoft+SQL+Server%3BOracle
59. Please, Please Stop Complaining about SQL Server Licensing Costs and Complexity. URL: https://joeydantoni.com/2016/08/18/please-please-stop-complaining-about-sql-server-licensing-costs-and-complexity/
60. Ramel D. Microsoft Takes Aim at Oracle with SQL Server 2016. URL: https://rcpmag.com/articles/2016/03/10/microsoft-vs-oracle-at-data-driven.aspx
61. «Za tu zhe funkcional'nost', kotoruyu daet SQL Server, Oracle prosit v 10 raz bol'she», – Konstantin Taranov o SQL Server. URL: https://habr.com/company/pgdayrussia/blog/329842/
62. Oracle Big Data Connectors. URL: https://shop.oracle.com/apex/product?p1=OracleBigDataConnectors&p2=&p3=&p4=&p5=&intcmp=ocom_big_data_connectors