DOI: 10.2507/32nd.daaam.proceedings.101

COMPARING HDFS – GREENPLUM DATA LOADING OPTIONS

Alexander Suleykin, Anna Bobkova, Peter Panfilov & Ilya Chumakov

This Publication has to be referred as: Suleykin, A[lexander]; Bobkova, A[nna]; Panfilov, P[eter] & Chumakov, I[lya] (2021). Comparing HDFS – Greenplum Data Loading Options, Proceedings of the 32nd DAAAM International Symposium, pp.0724-0731, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-33-4, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/32nd.daaam.proceedings.101

Abstract

In the last five years, many companies around the world have been successfully implemented Apache Hadoop as a main Data Lake storage for all data presented in the organization. At the same time, the adoption of other Open-Source technologies has been also increasing over years, such as classical MPP-based systems for Analytical workloads. Thus, the issue of efficient and fast data integration of Apache Hadoop and other organizational data storage systems becomes highly important for enterprises, where business and decision makers require the minimum delay of heterogeneous data exchange between Hadoop and other storages. In this paper, we compare different options for loading data from Apache Hadoop, representing the Data Lake of organization, into Open-Source MPP Greenplum database with the role of classical data warehouse for analytical workloads, and choose the best one. Also, we identify potential risks of using different data loading methods.

Keywords: Big Data; Hadoop Distributed File System; Greenplum; Massively Parallel Processing

1. Introduction

1.1. Hadoop HDFS as a main Data Lake storage

As it is defined in the IBM Redbook, Data Lake represents a set of centralized repositories, that contains vast amounts of structured and unstructured data, described in metadata, and organized into identifiable datasets [1]. One of the well-known distributed file systems for projects with a large amount of data is Hadoop Distributed File System (HDFS) [2], [3]. It is needed for storing large files and allows streaming access to data that are distributed in blocks of computing cluster that may consist of various hardware. HDFS is written in Java for Hadoop framework. There are factors that make HDFS a number one choice for data storage of an organization. HDFS can improve data processing and integrate different systems into a single Data Lake. Another factor is the fact that Hadoop is 10 to 100 times less expensive to deploy than conventional data warehousing. Today, companies in all industries starting to extract and place data for analytics into a single Hadoop-based repository, e.g., Google, Yahoo, Facebook, Netflix [4]. As is known from a study conducted by BARC (Business Application Research Center) in 2016, the main benefits of Hadoop are better heterogeneous data analysis – 59%, increased customer understanding - 53% and greater flexibility – 47% [5].

HDFS as the main Data Lake storage is the choice for most organizations with large amounts of unstructured data. Now, most of the medical data about patients, their anamnesis, the results of clinical trials are presented in an unstructured form, and their transformation and reduction to a structure takes a lot of time from medical organizations. There are several studies on this topic that support the point that Hadoop is a suitable solution for analytics of medical data [6], [7], [8]. As for other industries, namely in the Russian market, most often companies in the IT market also give their preference to building Data Lake with data storage in Hadoop, e.g., Kaspersky Lab, Aliexpress.ru, OZON, X5 Retail Group.

1.2. MPP databases

Massively parallel processing (MPP) databases in comparison to traditional database solutions make data reading from nodes and data processing on average much faster. The distinctive feature of MPP database architecture is physical separation of the memory of compute nodes combined into a cluster. Each node of MPP database of a cluster works only with its own hard disks by parallelizing data reading and writing operations. After each node finishes its computations and sort the results, it sends its portion of data to other nodes for getting data from them as well. During operation of receiving data each node selects only necessary records that are needed for computations. Other data is filtered out and does not take up space in the RAM. Commercial data warehouses are also in demand among MPP databases, such as Teradata, Oracle Exadata, Vertica, Netezza, etc. Since the comparison of databases is not in the scope of this paper, the Greenplum MPP-database will be considered in more detail. Greenplum is an MPP analytics database that adopts a 'shared-nothing' architecture with multiple cooperating processors. Greenplum manages the storage and processing of large amounts of data by balancing the load across multiple servers to create an array of individual databases that work together to present a single image of the database [9].

Database management system Greenplum is one of MPP-databases which is based on PostgreSQL for managing large-scale analytical data warehouses. It is considered that Greenplum cluster is a number of instances of PostgreSQL object-relational database with modification that work together as one Database management system. The programming code of PostgreSQL for Greenplum was modified for working with parallel processing, especially for parallelization of SQL queries. Greenplum uses ideas of the concept 'Shared nothing' when cluster nodes that interacts with each other for performing computing operations do not share resources. In this case, each of them has its own memory, operating system, and hard disk. It allows efficiency to parallel the load on performing analytical queries to multi-terabyte data stores.

Thus, having an idea of what Hadoop HDFS and Greenplum are, we can outline the purpose of this study. Now, there are a lot of tools for loading data. The paper presents the results of experiments with three instruments, such as data loading with python, data loading with Greenplum-Spark Connector, and data loading with PXF HDFS Connector. The result of this work is considered the most optimal tool for loading data from HDFS to Greenplum.

2. Integrating Hadoop HDFS and Greenplum

Greenplum flexibly complements Hadoop. It does not only allow using HDFS for data storage but also provides extensive opportunities for separate storage of partitions. For instance, some partitions can be stored on SSD for quick access. Other partitions with rarely used data on HDD because it is cheaper. Third partitions can be stored on HDFS. One of the features of the Greenplum DBMS is the usage of HDFS for creating external tables which avoids significant data loading time [13], [14]. The database includes the ability to provide gphdfs protocol privileges to the owner of external tables that access files on HDFS. Moreover, there is a software-based Hadoop connector that provides high-performance parallel import and export of compressed and uncompressed data from Hadoop clusters.

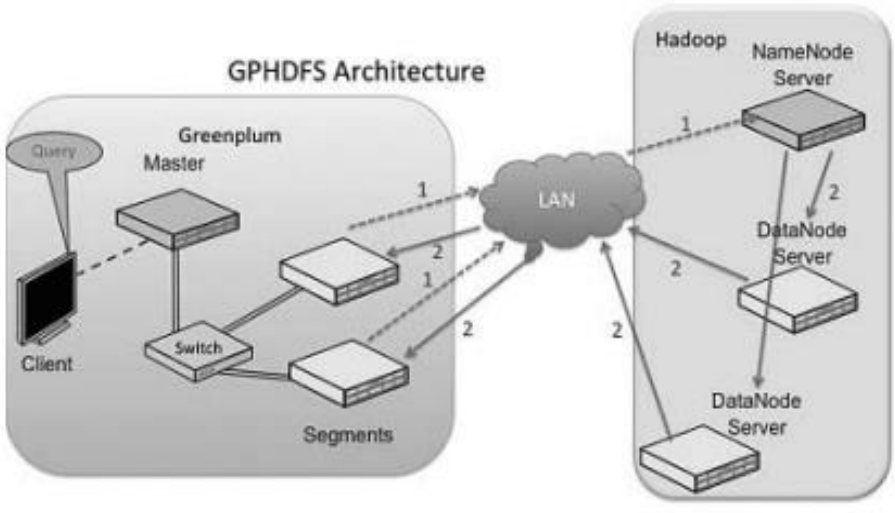

Fig. 1. Process of transferring data files from a Hadoop cluster to the Greenplum database

Custom format data in Hadoop can be converted to GPDB format using MapReduce and then imported into it for additional optimization of resource consumption during loading. This provides a much more efficient and flexible data exchange between Hadoop and the Greenplum DBMS [10], [11]. Greenplum supports parallel processing of SQL and MapReduce with data volumes ranging from hundreds of gigabytes, tens to hundreds of terabytes, to several petabytes. Fig. 1 illustrates the process of transferring data files from a Hadoop cluster to the Greenplum database upon reading request initiated by it. Reading request is sent to NameNode that controls access to the requested file data blocks. NameNode sends DataNode instructions to provide blocks of files to the Greenplum database. Greenplum has the framework PXF that allows exchanging data with third-party heterogeneous systems, including HDFS. The interaction of PXF with HDFS is explained in Fig. 2. Greenplum database is represented by a master server (MasterHost) and several segment servers (Segment Hosts). PXF framework consists of the Greenplum database protocol, client library for integrated data stores and Java service.

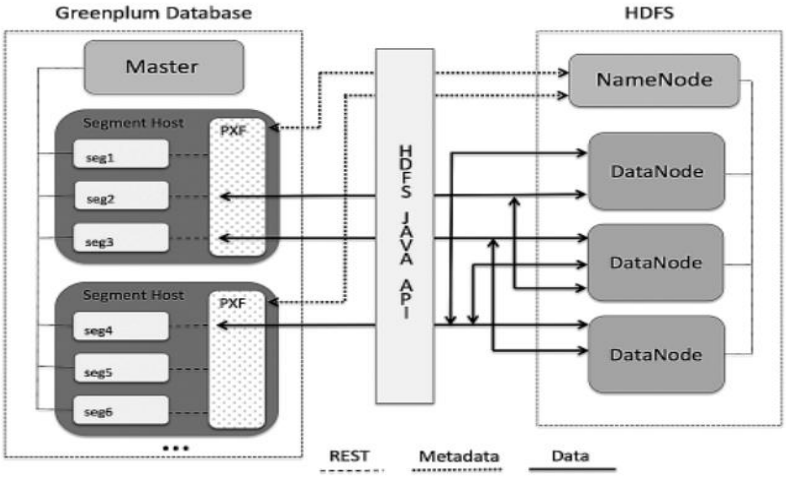

Fig. 2. The interaction of PXF with HDFS

PXF framework provides connectors for Hadoop, Hive, and HBase datastores that define profiles that support Binary and TEXT, Avro, JSON, RCFile, Parquet, SequenceFile and ORC. PXF connectors include the clients of the datastores in use on each segment host of the Greenplum database [6]. PXF extension implements a protocol called pxf is used to create an external table that references data in the integrated storage. The command to create an external table is used after the PXF extension is registered in the databases to access external data and privileges for the pxf protocol are assigned to those roles that need access. When executing a request for data in HDFS, the Master Host sends it to all segment servers. Each segment instance connects with a PXF running on its segment host and when PXF receives a request from segment instance it calls the Java HDFS API to retrieve the metadata information from the NameNode for the HDFS file and provides it to the segment instance. PXF is effective for integration with Greenplum since it excludes a network component during data exchange, has great flexibility and the ability to connect third-party systems with programming its own connector.

3. Experiments with Greenplum data loading

Name	Role	CPU, amount	RAM
demo-1	segment	6	32 Gb
demo-2	segment	6	32 Gb
demo-3	segment	6	32 Gb
demo-4	segment	6	32 Gb
demo-5	segment	6	32 Gb
demo-6	segment	6	32 Gb
demo-m	master	2	12 Gb
demo-s	master standby	2	12 Gb

Table 1. Experimental setup: Greenplum database configuration

In our experiments, we have tested three different data loading methods into Greenplum - using pure python, Apache Spark and PXF. Greenplum configuration in the test environment is shown in table 1: All hosts used SAS disks from shared storage. Machines did not change it configuration during all stages of testing.

3.1. Greenplum data loading with python

Load testing was carried out using a self-written python script. The dataset consists of an array of random strings 18 * 100,000 in size. The script was run with insert Single_insert (row insertion), Execute_many, Execute_batch, Execute values methods. To communicate with the database, the psycopg2 python library was used. The dataset was handled by Pandas and NumPy. Below the results are presented of different insert types with python libraries into Greenplum system (Fig. 3, table 2):

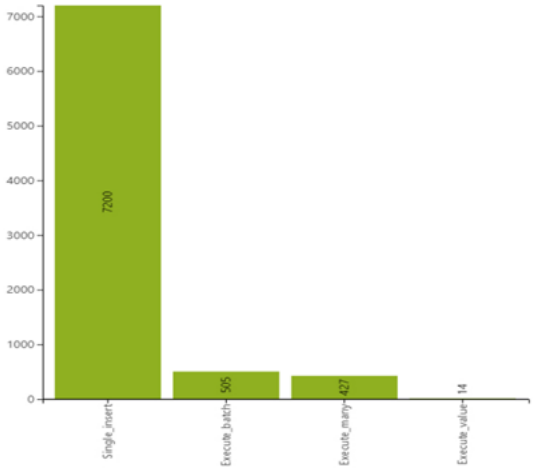

Fig. 3. Different Insert types in Greenplum using python, figure

Insert Type	Time, sec	Speed, rows/sec
Single Insert	7200	17
Execute Batch	505	198
Execute Many	427	234
Execute Value	14	7145

Table 2. Different Insert types in Greenplum using python, table

3.2. Greenplum data loading with Greenplum-Spark Connector

The source file in Apache Parquet format (with Snappy compression) includes 18 columns with numeric and text arbitrary data, all records are unique. The average row size of a test dataset in Greenplum is 933 bytes. The number of rows in the test dataset is presented in the corresponding column in the totals table. Based on the source file, a Spark DataFrame is formed with the number of sections corresponding to the number of Greenplum segments to achieve maximum parallelism (distribution keys also match). Filling of internal Greenplum tables is performed using the Greenplum-Spark Connector from the generated DataFrame. ZSTD compression is used for internal Greenplum tables.

Dataset Size, rows	Data Loading Time, sec	Deserialization Time, sec	Data Loading Speed, rows/sec
10 000 000	324	168	20325
20 000 000	900	396	15432
50 000 000	2100	1080	15723

Table 3. Apache Spark data loads, performance comparison

The test evaluation of cluster load was performed using Grafana – an open-source tool for metrics visualization. We measured the most important metrics when performed load tests which are the CPU load and RAM usage. Every node in Greenplum cluster is shown in different color. The following Figures show the CPU data load (Fig. 4 – 6) and RAM memory load (Fig. 7 – 9) visualizations with 10 mln., 20 mln., and 50 mln. rows inserts with Apache Spark Connector, respectively.

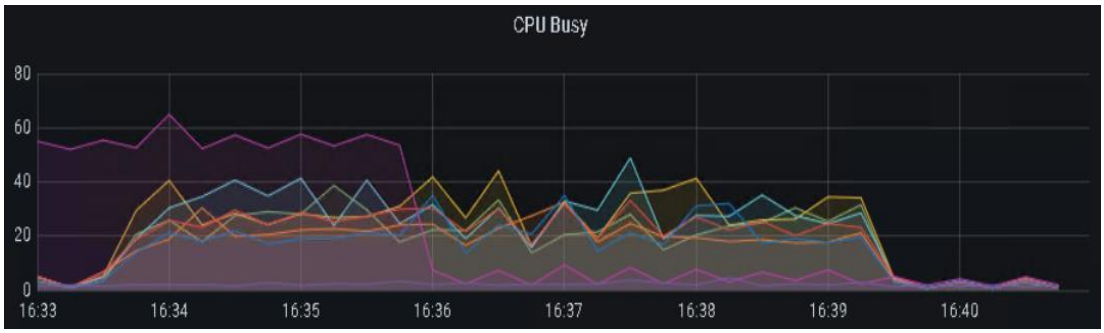

Fig. 4. CPU load with Apache Spark Connector and 10 000 000 records loading

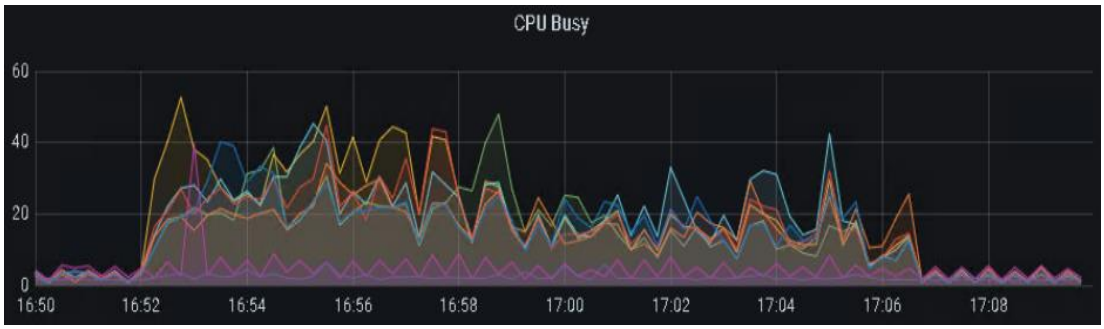

Fig. 5. CPU load with Apache Spark Connector and 20 000 000 records loading

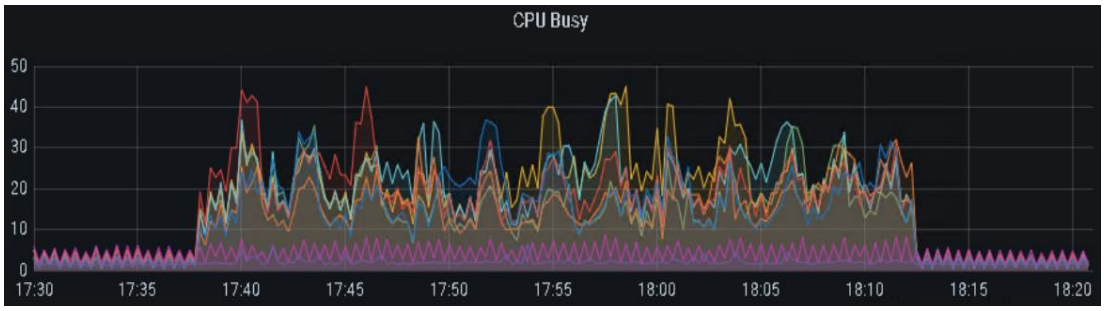

Fig. 6. CPU load with Apache Spark Connector and 50 000 000 records loading

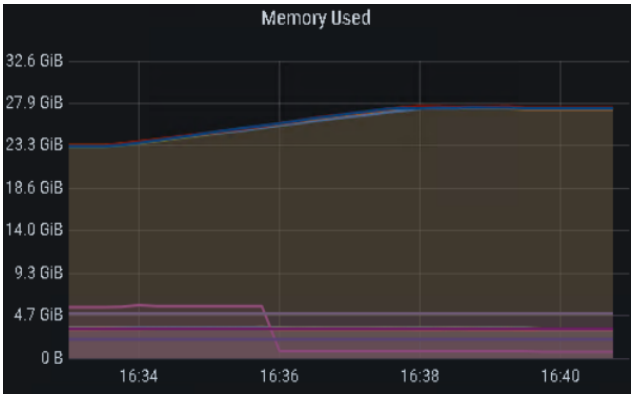

Fig. 7. RAM used with Apache Spark Connector and 10 000 000 records loading

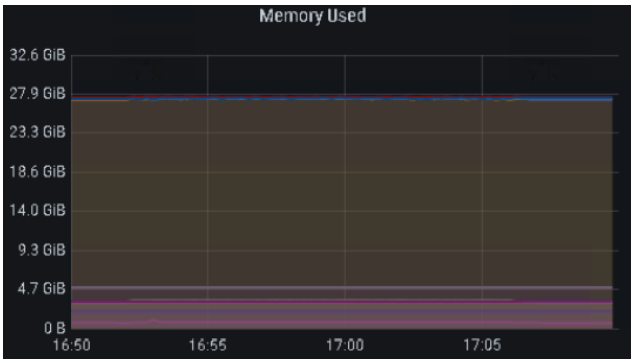

Fig. 8. RAM used with Apache Spark Connector and 20 000 000 records loading

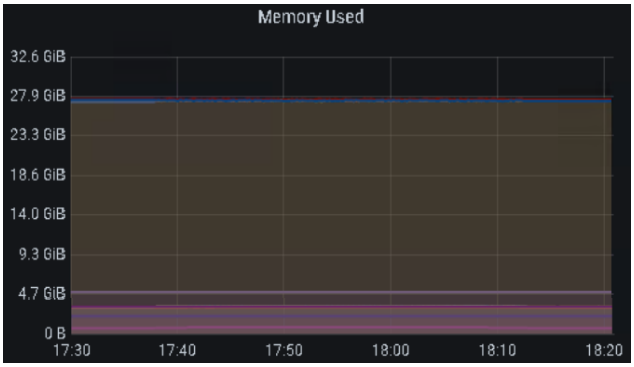

Fig. 9. RAM used with Apache Spark Connector and 50 000 000 records loading

3.3. Greenplum data loading with PXF HDFS Connector

The structure of the dataset is the same as using Greenplum-Spark Connector test set. Based on the source file, the external Greenplum tables are populated using PXF HDFS Connector, and the data is transferred from them to the internal Greenplum tables. For internal tables, ZSTD compression is used. Table 4 depicts the results for different number of records with PXF Connector:

Number of Rows in Dataset, rows	**Loading Time, sec**	**Loading Speed, rows/sec**
10 000 000	144	69444
20 000 000	279	71685
50 000 000	759	65876

Table 4. PXF data loadings, performance comparison

The following Figures show the CPU data load (Fig. 10 and 11) and RAM memory load (Fig. 12 and 13) visualizations with 10 mln. and 20 mln. rows inserts with PXF Connector, respectively.

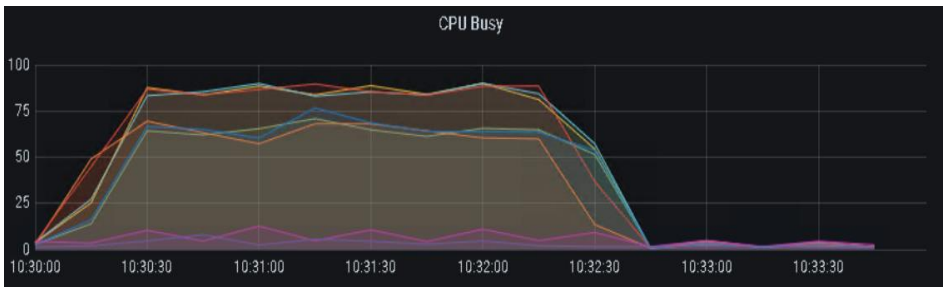

Fig. 10. CPU load with PXF Connector and 10 000 000 records loading

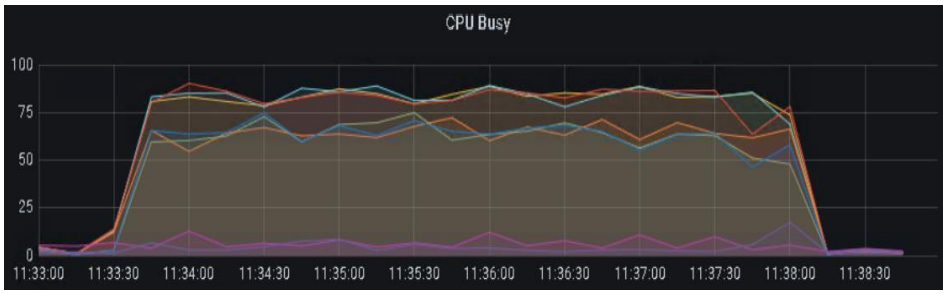

Fig. 11. CPU load with PXF Connector and 20 000 000 records loading

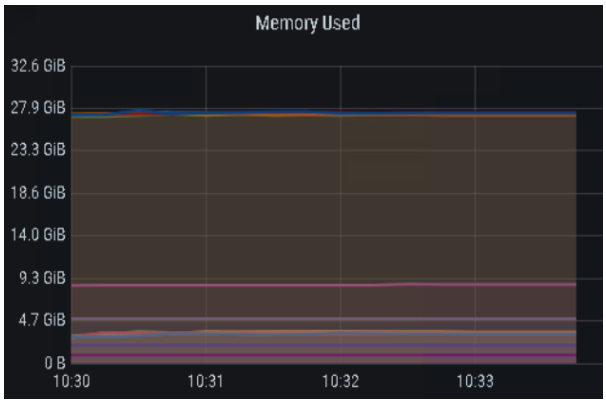

Fig. 12. RAM used with PXF Connector and 10 000 000 records loading

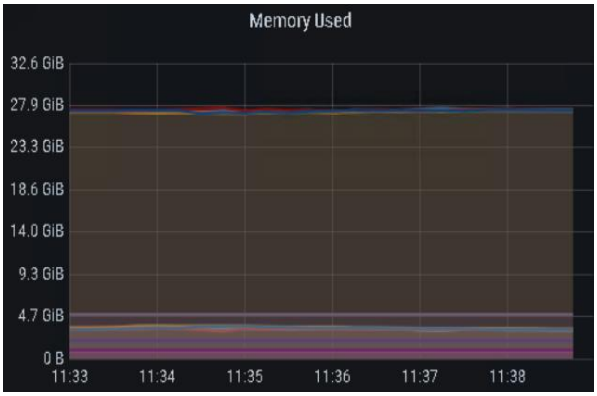

Fig. 13. RAM used with PXF Connector and 20 000 000 records loading

With similar settings, loading data through PXF is faster, including due to higher equipment utilization. No degradation of productivity was observed with increasing volumes. When using the PXF protocol, the load is evenly distributed among the Greenplum segment nodes. Master node usage is minimal. Combined with the high speed of the PXF protocol and the absence of degradation, this option is suitable for loading large amounts of data into Greenplum.

4. Discussion

The use of direct inserts (SQL inserts) through the Master node is immediately excluded, since has an obvious problem loading the master, which is used to parse requests and check permissions. Greenplum-Spark Connector provides loading that bypass the master and possibly has the potential for horizontal scaling. Formally, this is enough to complete the task, but there are nuances. Among the obvious drawbacks, it has a mandatory step of loading data into a Spark DataFrame, which creates additional overhead. To download data through this connector, the Spark competency (Java / Scala) is required, which increases the TCO of the solution and entails additional difficulties in replicating updates. PXF HDFS Connector also provides data loading that bypass the master node, that allows for the solution to scale horizontally. When installed, it seamlessly integrates with the Hadoop cluster. During data access, automatic load balancing occurs on the Greenplum side, which allows for abstracting from the Hadoop cluster configuration. To work with data through this connector, it is enough to know only the declarative part of SQL, and no knowledge of other languages is required.

5. Conclusion

In this paper, we have presented the practical study of the usage, architectural features, and main benefits of Apache Hadoop HDFS-based data infrastructure in many organizations. As a popular alternative solution and enhancement to data infrastructure, we have also discussed architecture, implementations, and advantages of the Open-Source MPP-based database Greenplum usage in organizations, as well as different ways of how data exchanges between HDFS and Greenplum can be arranged. Finally, we have tested different HDFS-Greenplum integration options based on Master node row insert, Spark-Greenplum connector and PXF usage. As a result of test implementation, the best option for Hadoop-Greenplum data exchange has been proposed and different risks for the development and production exploitation were considered. Returning to the question of which tool is the most optimal for loading data from HDFS to Greenplum, our experiments have shown that it is PXF HDFS Connector. But it is important to note that with the correct application of any of these tools, the desired result can be achieved. Therefore, as one of the ways for further research, the settings of these tools for solving various kinds of tasks can be analyzed.

6. Acknowledgements

The reported experimental research was partially supported by RFBR grant No 20-07-00958 and the HSE University's Project Group Competition grant.

7. References

[1] Chessell, M., Scheepers, F., Strelchuk, M., van der Starre, R., Dobrin, S., Hernandez, D. (2018). The Journey Continues: From Data Lake to Data-Driven Organization. IBM Corp. Available from: https://www.redbooks.ibm.com/redpapers/pdfs/redp5486.pdf. Accessed: 2021-10-12

[2] Beakta, R. (2015). Big Data and Hadoop: A Review Paper. International Journal of Computer Science & Information te.2. Vol. 2, Issue 2. Available from: https://www.researchgate.net/publication/281403776_Big_Data_And_Hadoop_A_Review_Paper/, Accessed: 2021-10-12

[3] Depardon, B., Le Mahec, G., Séguin, C. (2013). Analysis of Six Distributed File Systems. [Research Report] 2013, pp.44. ffhal-00789086. Available from: https://hal.inria.fr/hal-00789086/file/a_survey_of_dfs.pdf, Accessed: 2021-10-12

[4] Stein, B., Morrison, A. (2014). The enterprise data lake: Better integration and deeper analytics, Available from: http://www.smallake.kr/wp-content/uploads/2017/03/20170313_074222.pdf, Accessed: 2021-10-12

[5] Grosser, T., Bloemen, J., Mack, M. & Vitsenko, J. (2016). Hadoop and Data Lakes: Use Cases, Benefits and Limitations. BARC – Business Application Research Center, a CXP Group Company. Available from: http://barc-research.com/research/hadoop-and-data-lakes/, Accessed: 2021-10-12

[6] Rangarajan, S., Liu, H., Wang, H., Wang, Ch.-L. (2018). Scalable Architecture for Personalized Healthcare Service Recommendation Using Big Data Lake, Available from: https://link.springer.com/chapter/10.1007/978-3-319-76587-7_5, Accessed: 2021-10-12

[7] McPadden, J., Durant, T.J., Bunch, D.R., Coppi, A., Price, N., Rodgerson, K., Torre, Jr C.J., Byron, W., Hsiao, A.L., Krumholz, H.M., Schulz, W.L. (2019). Health Care and Precision Medicine Research: Analysis of a Scalable Data Science Platform, Available from: https://www.jmir.org/2019/4/e13043/, Accessed: 2021-10-12

[8] Soroushmehr, S.M.Reza, Najarian, K. (2016). Transforming big data into computational models for personalized medicine and health care, Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5067150/, Accessed: 2021-10-12

[9] Albertini, O., Bhargov, D., Denissov, A., Guerrero, F., Jayaram, N., Kak, N., Khanna, E., Kislal, O., Kumar, A., McQuillan, F., Owen, L., Raghavan, V., Valdano, D., Zhang, Y. (2020). Image Classification in Greenplum Database Using Deep Learning, Available from: http://greenplum.org.s3.amazonaws.com/wp-content/uploads/2020/05/12170349/Image-Classification-in-Greenplum.pdf, Accessed: 2021-10-12

[10] Bani, F.C.D., Suharjito, Diana, Girsang, A.S. (2018). Implementation of Database Massively Parallel Processing System to Build Scalability on Process Data Warehouse. Available from: https://www.sciencedirect.com/science/article/pii/S1877050918314376, Accessed: 2021-10-12

[11] Greenplum® Database 4.1 Administrator Guide, [online] Available from: media.gpadmin.me/wp-content/uploads/2011/ 05/GP-4100-AdminGuide.pdf, Accessed: 2021-10-12

[12] Kasibhotla, D. (2012). Greenplum and Hadoop HDFS integration, Available from: https://dwarehouse.wordpress.com/2012/ 10/10/greenplum-and-hadoop-hdfs-integration/, Accessed: 2021-10-12

[13] New Functionality in Greenplum Database 4.2. Welcome to Greenplum Database 4.2. Updated: Nov. 23. 2011. Available from: http:// media.gpadmin.me/wp-content/uploads/2012/11/ GPDB_4200_README.pdf, Accessed: 2021-10-12

[14] Using PXF to Read and Write External Data. Greenplum Platform Extension Framework (PXF). Available from: https://gpdb.docs. pivotal. io/550/pxf/using_pxf.html, Accessed: 2021-10-12