Big Data Analysis and Query Optimization Improve HadoopDB Performance

Cherif A. A. BISSIRIOU
Systems Engineering Laboratory,
ADSI Research Group,
National School of Applied Sciences (ENSA)
Ibn Tofail University
cbissiriou@gmail.com

Habiba CHAOUI
Systems Engineering Laboratory,
ADSI Research Group,
National School of Applied Sciences (ENSA)
Ibn Tofail University
mejhed90@gmail.com

ABSTRACT
High performance and scalability are two essentials requirements for data analytics systems as the amount of data being collected, stored and processed continue to grow rapidly. In this paper, we propose a new approach based on HadoopDB. Our main goal is to improve HadoopDB performance by adding some components. To achieve this, we incorporate a fast and space-efficient data placement structure in MapReduce-based Warehouse systems and another SQL-to-MapReduce translator. We also replace the initial Database implemented in HadoopDB with other column oriented Database. In addition we add security mechanism to protect MapReduce processing integrity.

Categories and Subject Descriptors
C.4.3 [**Performance of Systems**]: design study, performance attributes, query processing.

General Terms
Performance, Design, Reliability, Experimentation, Security.

Keywords
Big Data, Query Execution, Optimization, MapReduce, Hadoop

1. INTRODUCTION
Nowadays, we are witnessing an explosion of large amounts of data being collected, stored and processed with the advent of Internet of Things and social web technologies. Efficient processing of this "Big Data" as it has been called becomes a major issue to benefit from the added value it can bring. In this context, MapReduce framework [8] has become the leading framework for performing scalable big data analytics and big data mining.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SEM '14, September 04 - 05 2014, Leipzig, AA, Germany
Copyright 2014 ACM 978-1-4503-2927-9/14/09…$15.00.
http://dx.doi.org/10.1145/2660517.2660529

[11], impressive ease-of-use experience (Pavlo et al. 2009), as well as Google's, Yahoo!'s, and Facebook's wide usage and evangelization of this technology. The initial applications of the MapReduce framework included Web indexing and text analytics. While the MapReduce framework is especially designed to satisfy scalability, it has often been criticized for its poor performance. Consequently, in recent years some researches and commercials activities have focused on integrating MapReduce and relational database technology together. There are two approaches to do this:

- Starting with a parallel database system and adding some MapReduce features [10]. Aster [9] and Greenplum [10] are two commercial systems that integrate the MapReduce framework. These systems show how the MapReduce operator and other relational operators are combined for superior performance.
- Starting with MapReduce and adding database system technology [12]. The Pig project [13] and the open source Hive project [14] are the MapReduce-based Warehouse systems. They define an SQL-like high-level language for processing large-scale analytics workloads. The queries expressed in these languages are transformed into a set of MapReduce jobs which are submitted to Hadoop. HadoopDB [1] follows the second approach. HadoopDB is a hybrid system that takes the best features from both technologies; approaches parallel databases in performance and efficiency, and still yields the scalability, fault tolerance, and flexibility of MapReduce-based systems. HadoopDB [1] show how the performance of MapReduce framework can be significantly improved by using techniques inherited from Parallel Database Management Systems. However HadoopDB performance is lower in query optimization that need in a large and high production environment. Our work aim is to improve HadoopDB performance by incorporating new query optimization techniques such as the new translator YSmart [2] and the data placement structure RCFile [3], which have been developed for the MapReduce-based Warehouse Systems. We also replace the initial Database implemented in HadoopDB with other column oriented Database called MonetDB [4]. We found that, the design of MapReduce-based platform don't embed a security layer. Thus, as our system is an open system, we add a security layer in our system, which in our point of view is an essential property in such big data analytics platform. Our secure implementation is based on the work done at SecureMR [15]. We use the same idea in this work to implement security mechanism to ensure MapReduce integrity in our system.

This paper is organized as follows: Section 2 describes the underlying approaches we use in our new approach to improve HadoopDB performance. The details of our new approach are presented in Section 3. Finally, a conclusion will be given with prospects for future works.

2. BACKGROUND

2.1 HadoopDB

The main idea behind HadoopDB is to create a single system by connecting multiple independent single-node databases deployed across a cluster as shown in figure 1:

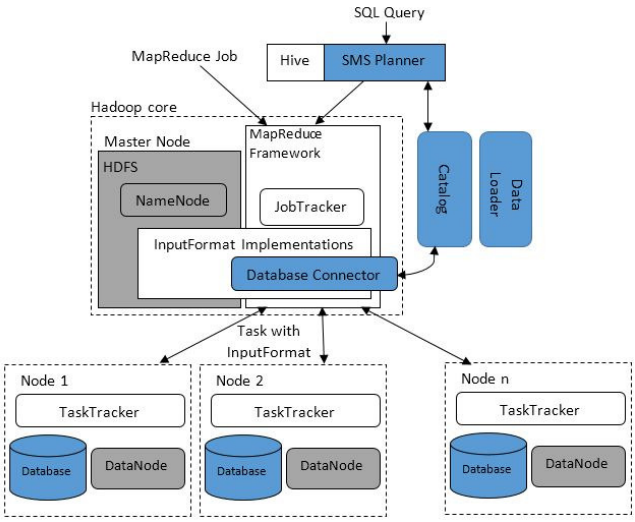

Figure 1. Initial HadoopDB Architecture

Queries are parallelized using Hadoop, which serves as a coordination layer. To achieve high efficiency, performance sensitive parts of query processing are pushed into underlying database systems. HadoopDB is able to effectively transform any single-node database system into a shared-nothing parallel database, while inheriting the job tracking, runtime scheduling and fault tolerance of MapReduce.

This system operates as follows. First, SQL queries are translated into MapReduce jobs using a modified version of Hive. Then, each MapReduce job connects to the underlying databases, executes as much of the query as possible there, and returns a set of key-value pairs to the MapReduce framework for further processing. The main components of HadoopDB include:

- **Database Connector**: that allows Hadoop jobs to access multiple database systems by executing SQL queries via a JDBC interface.
- **Data Loader**: that hash-partitions and splits data into smaller chunks and coordinates their parallel load into the database systems.
- **Catalog**: This contains both metadata about the location of database chunks stored in the cluster and statistics about the data.
- **SQL to MapReduce to SQL (SMS) Planner**: This allows queries to be submitted in HadoopDB. The SMS planner extends Hive by intercepts the normal Hive flow.

2.2 RCFile

Record Columnar File (RCFile) [3], aim is to satisfy all the requirements of large scale data analysis in systems like MapReduce. RCFile adopts the concept of "first horizontally partition, then vertically partition". RCFile increases the basic operation unit and organizes records into row groups. Thus, each HDFS block may contain one or more row groups, each in turn containing multiple records. With this design RCFile achieves a minimal cost of record reconstruction since, given the relational schema, all record attributes can be guaranteed to be together in the same row-group in the same node. Then, each row group is internally organized like a column store. This makes it possible to skip unnecessary column reads at query execution time, since RCFile can read from the underlying storage only the columns required by the executed query. Both these design choices enable fast query processing times for RCFile. Secondly, RCFile achieves efficient storage space utilization, by compressing each column separately, like a column-store.

2.3 YSmart

YSmart [2] is a correlation aware SQL-to-MapReduce translator, which is built on top of the Hadoop. For a given SQL query and related table schemas, YSmart can automatically translate the query into a series of MapReduce jobs. YSmart supports three types of intra-query correlations defined based on the key/value pair model of the MapReduce framework. After automatically detecting such correlations in a query, YSmart applies a set of rules to generate optimized MapReduce jobs, which are managed by the Common MapReduce Framework (CMF). This design provides significant query performance improvement by reducing redundant computations, unnecessary disk accesses, and network overhead. Compared to other SQL-to-MapReduce translators, YSmart has been proved to have the following advantages:

- **High Performance**: The MapReduce programs generated by YSmart are optimized.
- **High Extensibility**: YSmart is easy to modify and extend. It is designed with the goal of extensibility.
- **High Flexibility**: YSmart can run in two different modes: translation-mode and execution-mode.

2.4 MonetDB

MonetDB [4] is an innovative open-source database system. MonetDB has some advantages, such as efficient data access patterns, flexibility with changing workloads, reduced storage needs, and run-time query optimization. It is a column-store database system. This approach minimizes the data flow from disk through memory into the CPU caches since only the columns relevant for processing have to be fetched from disk. This can be especially favorable in data analysis applications that need to efficiently retrieve and process large portions of stored data. It was designed to provide high performance on complex queries against large databases, such as combining tables with hundreds of columns and multi-million rows.

2.5 SecureMR

SecureMR [15] is a practical service integrity assurance framework for MapReduce. SecureMR consists of five security components, which provide a set of practical security mechanisms that not only ensure MapReduce service integrity as well as to prevent replay and Denial of Service (DoS) attacks, but also preserve the simplicity, and scalability of MapReduce. SecureMR provides a decentralized replication-based integrity verification scheme for ensuring the integrity of MapReduce in open systems, while imposing low performance overhead. The security

components in SecureMR can be easily integrated into existing MapReduce implementations. The design of SecureMR follows two aspects: (1) architecture design; (2) communication design and is based assumption that the master is a trusted entity. The architecture design of SecureMR, comprises five security components: Secure Manager, Secure Scheduler, Secure Task Executor, Secure Committer and Secure Verifier. Each component has specific task and is distributed across the all cluster.

3. Our Proposal

Our approach as shown in figure 2, is based on the technics mentioned in the previous section. We aim to take the best features of these approaches, combine them to form a coherent and optimized entity. Our system as HadoopDB, must ensure the following properties, which are essential for querying analysis of large volumes of data optimally and efficiently: (1) High performance (fast data loading, faster query processing, efficient use of storage space); (2) Fault Tolerance and Scalability; (3) Suitable for a heterogeneous environment; (4) Flexible query interface. In addition, we ensure that our system is open source.

Firstly, instead of using the PostgreSQL database on each DataNode as in HadoopDB, we choose MonetDB database [4]. According to their analysis, the authors of HadoopDB in their initial article stated that the PostgreSQL databases affect HadoopDB performance. Thus, in the second version of HadoopDB, called Hadapt [6], which is a commercial version of HadoopDB, the authors have changed PostgreSQL databases from another database named "VectorWise/X100 Database". The VectorWise database is a single-node DBMS based on the MonetDB/X100. As we can notice, MonetDB is more efficient than others column-store database systems based on PostgreSQL such as CitusDB [18].

Secondly, we embed RCFile file system format in HadoopDB. The data placement structure is one of the critical factors that affect the MapReduce-based platform performance. This factor can reduce mainly two types of incremental costs in MapReduce Framework: the network and storage costs for massive accesses. Four requirements for the data placement structure are needed in MapReduce environment: (1) fast data loading; (2) fast query processing; (3) high and efficient use of storage space; (4) strong adaptability to any model of dynamic workload. Several solutions have been proposed for data placement structure in a MapReduce environment. RCFile is the only approach that provides the best technical data placement structure in MapReduce. RCFile fully meets these requirements mentioned above compared to other data placement technics. RCFile provides a high performance, an important property of our system. In comparison with other structures, RCFile outperforms all of them in two different workloads: TCP-H workload and Facebook workload. With her performance as it has been demonstrated in their paper, we embed RCFile in our new system to get better performance.

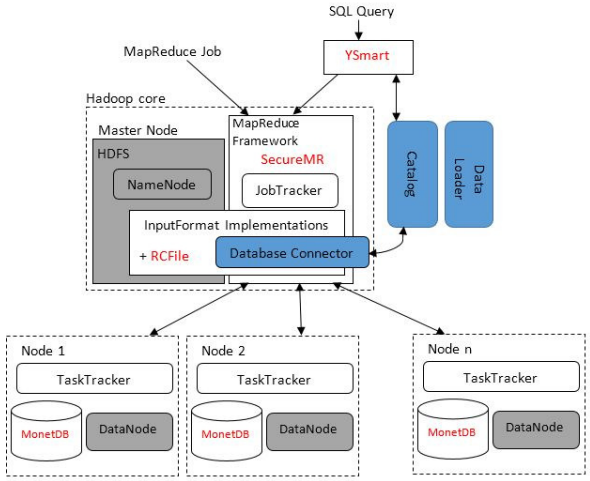

Figure 2. New HadoopDB Architecture

Thirdly, we substitute SMS Planner by YSmart. HadoopDB enabling to process SQL queries with SMS Planner. The SMS Planner is an extension based on Hive [14]. Hive accepts queries expressed in a SQL-like language called HiveQL and executes them against data stored in the Hadoop Distributed File System (HDFS). When translating a query expressed by such a language into MapReduce programs, HIVE takes a one operation to one job approach. However, auto-generated MapReduce jobs for many queries are often extremely inefficient compared to hand-coded programs by experienced programmers. HIVE cannot generate high-performance MapReduce programs. Other limitation of Hive is its data storage layer, deployed on top of a distributed file system, Hive is unable to use hash-partitioning on a join key for the colocation of related tables, a typical strategy that parallel databases exploit to minimize data transaction across nodes. The SMS Planner modifies Hive in order to generate less job than Hive. However the SMS Planner performance is lower due to inherit from Hive. Recently other translators have been proposed, including AQUA [7] and YSMART [2], in order to generate efficient MapReduce jobs. AQUA is a query optimization method designed for MapReduce-based MPP systems. AQUA performs query optimization in two phases. In the first phase, the optimizer partitions the tables into join groups. Each join group is processed by a single MapReduce job. In the second phase, the optimizer searches for the best plan to generate the final results by combining the join group. We carried out a comparative study between AQUA and YSmart based on their initial paper. This comparison show that YSmart performance is better than AQUA performance. YSmart also outperforms all other translators (Hive, Pig). Thus we substitute YSmart instead of the SMS Planner.

Fourthly, we add a security layer in our new system. Our security implementation is based on the same idea in SecureMR [15]. Originally, the design of MapReduce framework don't embed a security mechanism. In Hadoop implementation, this fact can allow any user to impersonate any other user and allow a malicious network user to impersonate cluster services. The Hadoop File System lax authorization can allowed anyone to write data and any data to be read. An arbitrary java code could be submitted to Job Trackers to be executed as the Job Tracker user account [16]. In open systems, a data processing integrity is major issue since the nodes may come from different domains that are

not always trustworthy [15]. In this context, Yahoo!'s developers propose a new security design to improve the state of Hadoop security [17]. They use the Simple Authentication and Security Layer (SASL) with Kerberos, to authenticate users. Their new Hadoop security design makes use of Delegation Tokens, Job Tokens and Block Access Tokens. Each of these tokens is similar in structure and based on HMAC-SHA1. However their new design poses a number of issues. Mainly this design does not provide integrity mechanism. We also noticed that, their design introduced some overheads costs that affect Hadoop performance. We then use the idea provide in SecureMR to get integrity mechanism in our system. The security components in SecureMR can be easily integrated into existing MapReduce implementations like HadoopDB and still preserve the simplicity, applicability and scalability of MapReduce. The evaluation analysis show that the performance overhead introduced by SecureMR is below 10s under different scenarios. We first integrate the five security components in HadoopDB like SecureMR. Secure Scheduler and Secure Manager are integrate into the NameNode. Secure Task Executor, Secure Committer and Secure Verifier are integrate into DataNode on each node. When single-node databases systems generate a key-value pairs, which are transformed into MapReduce jobs, our system follows the same communication design between the five components, as is describe in SecureMR to ensure integrity mechanism. As it has been proven in their paper, our system provides also these two properties: no false alarm and non-repudiation.

4. Conclusion

Perform complex queries with high efficiency and high performance is major requirement for big data analytics tasks. It's now evident to put together MapReduce framework and parallel database system for better performance. In this paper we show that we can improve HadoopDB performance by adding new MapReduce-based warehouse technics, which are designed and developed to give high performance in big data production environments. Both RCFile and YSmart are currently used in Facebook's production environment. We noticed that, actually Hadoop-based platform don't embed a security layer. To prevent security threat, we integrate a security layer in our approach. Our security implementation is based on SecureMR, which provide service integrity assurance for MapReduce. Integrity assurance for MapReduce processing is in our point of view, an essential security criterion in such big data analytics platform. An experimental evaluation is in progress to verify the effectiveness of our new system. In our future research we plan to test others column-store database systems in our proposal.

REFERENCES

[1] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., and Rasin, A. (2009). HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc. VLDB Endow. 2, 922–933.

[2] R. Lee, T. Luo, Y. Huai, F. Wang, Y. He, and X. Zhang. (2011). Ysmart: Yet another sql-to-mapreduce translator. In Proceedings of the Distributed Computing Systems (ICDCS), 31st International Conference on, 2011, pp. 25–36.

[3] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu. (2011). RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In Data Engineering (ICDE), 2011 IEEE 27th International Conference on, 2011, pp. 1199–1208.

[4] P. A. Boncz, M. Zukowski, and N. Nes. (2005). MonetDB/X100: Hyper-Pipelining Query Execution. In CIDR, 2005, vol. 5, pp. 225–237.

[5] Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., and Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, (ACM), pp. 165–178.

[6] Bajda-Pawlikowski, K., Abadi, D.J., Silberschatz, A., and Paulson, E. (2011). Efficient processing of data warehousing queries in a split execution environment. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, (ACM), pp. 1165–1176.

[7] Wu, S., Li, F., Mehrotra, S., and Ooi, B.C. (2011). Query optimization for massively parallel data processing. In Proceedings of the 2nd ACM Symposium on Cloud Computing, (ACM), p. 12.

[8] J. Dean and S. Ghemawat. (2008). Mapreduce: simplified data processing on large clusters. In Commun. ACM, 51(1):107–113, Jan. 2008.

[9] http://www.aster.com.

[10] http://www.greenplum.com.

[11] http://hadoop.apache.org.

[12] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. (2009). MAD skills: new analysis practices for big data. In PVLDB, 2(2):1481-1492, 2009.

[13] Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. (2008). Pig latin: a not-so-foreign language for data processing. In SIGMOD, 2008.

[14] http://hadoop.apache.org/hive.

[15] Wei, W., Du, J., Yu, T., and Gu, X. (2009). Securemr: A service integrity assurance framework for mapreduce. In Computer Security Applications Conference, 2009. ACSAC'09. Annual, (IEEE), pp. 73–82.

[16] Andrew Becherer (2010). Hadoop Security Design. Just Add Kerberos? Really? In BlackHat-USA 2010.

[17] Devaraj Das, Owen O'Malley, Sanjay Radia and Kan Zhang. (2010). Adding Security to Apache Hadoop. Hortonworks Technical Report.

[18] https://www.monetdb.org/content/citusdb-postgresql-column-store-vs-monetdb-tpc-h-shootout