What If Mixing Technologies for Big Data Mining and Queries Optimization

Dhouha Jemal[1(✉)] and Rim Faiz[2]

[1] LARODEC, ISG Tunis, University of Tunis, Tunis, Tunisia
dh.jemal@gmail.com
[2] LARODEC, IHEC Carthage, University of Carthage, Carthage, Tunisia
rim.faiz@ihec.rnu.tn

Abstract. Data is growing at an alarming speed in both volume and structure. The data volume and the multitude of sources have an exponential number of technical and application challenges. The classic tools of data management became unsuitable for processing and unable to offer effective tools to deal with the data explosion. Hence, the imposition of the Big Data in our technological landscape offers new solutions for data processing. In this work, we propose a model that integrates a Big Data solution and a classic DBMS, in a goal of queries optimization. Then, we valid the proposed optimized model through experiments showing the gain of the execution cost saved up.

Keywords: Big data · Optimization · Mapreduce · Information retrieval · Integration · OLAP · Cost · Performance

1 Introduction

The data is currently in the middle of themes and its amount is increasing dramatically. It's not easy to measure the total volume of data generation and processing. Data generation is estimated of 2.5 trillion bytes of data every day[1]. This impressive figure masks even more important evolutions. First, unstructured data will grow faster than structured data. Moreover, beyond the storage, challenges will focus on the capacity to process the data and make it available to users. As described in [16], with the worldwide volume of data which does not stop growing, the classical tools for data management have become unsuitable for processing. Therefore a need of defining and developing new technologies and solutions for access to the massive quantities of data in order to extract meaningful information and knowledge.

In this context, Big Data is a popular phenomenon which aims to provide an alternative to traditional solutions database and analysis. It leads to not only a technology revolution but also a business revolution. It is not just about storage of and access to data, Big Data solutions aim to analyze data in order to make sense of that data and exploiting its value. Thus, MapReduce [3] is presented as one of the most efficient

[1] http://www-01.ibm.com/software/fr/data/bigdata/

DOI: 10.1007/978-3-319-24306-1_60

Big Data solutions. As presented in [4], it is emerging as an important programming model for large-scale data-parallel applications.

Data is the most precious asset of companies and can be mainspring of competitiveness and innovation. As presented in [15], many organizations noticed that the data they own and how they use it can make them different than others. This explains that most organizations try to collect and process as much data as possible in order to find innovative ways to differentiate themselves from competitors. IDC[2] estimates that enterprise data doubles every 18 months. Otherwise, business intelligence and analytics are important in dealing with the magnitude and impact of data driven problems and solutions. According to the IDC[3] Research, the worldwide market for business analytics software grew 8.2% to reach $37.7 billion in 2013. That is why organizations need to be able to rapidly respond to market needs and changes, and it has become essential to have efficient and effective decision making processes with right data to make the decision the most adapted at a given moment. Although, according to Salesforce research, 89% of business leaders believe Big Data will revolutionize business operations in the same way the Internet did, and 83% have pursued Big Data projects in order to seize a competitive edge[4]. The question is how organizations should prepare for these developments in the big data ecosystem, what technology to use for data analysis in such an environment.

Conscious of the need to a powerful and optimized tools to verify and analyze data in order to support the decision-making process, the aim of this work is to optimize queries to meet new organizations needs by determining the most efficient way to execute a given query. We propose to integrate two main categories of data management systems: classic Database management systems (DBMS) and NoSQL DBMSs. The idea is to integrate the ORDBMS [17] PostgreSQL [6] as the most notable research project in the field of object relational database systems, and MapReduce to perform OLAP queries in a goal of minimizing Input/Output costs in terms of the amount of data to manipulate, reading and writing throughout the execution process. The main idea of integration leans on the comparison of query execution costs by both paradigms with the aim of minimizing the Input/Output costs. We valid the proposed approach through experiments showing the significant gain of the cost saved up compared to executing queries independently on MapReduce and PostgreSQL.

The remainder of this paper is organized as follows. In section 2, we give an overview of related work addressing data management systems integration search. In section 3, we present our proposed approach for optimizing the online analytical processing (OLAP) queries Input/Output execution cost, then in section 4 we discuss the obtained results. Finally, section 5 concludes this paper and outlines our future work.

[2] http://www.infoworld.com/article/2608297/infrastructure-storage/how-to-survive-the-data-explosion.html

[3] http://www.sas.com/content/dam/SAS/en_us/doc/analystreport/idc-ba-apa-vendor-shares-excerpt-103115.pdf

[4] http://www.forbes.com/sites/louiscolumbus/2015/05/25/roundup-of-analytics-big-data-business-intelligence-forecasts-and-market-estimates-2015/

2 Related Work

Both MapReduce and Parallel DBMS provide a means to process large volumes of data. However, with exponential generation, sources of data changes producing new types of data and content that gave rise to new challenges in the data treatment and analysis. Researchers asked as to whether the parallel DBMS paradigm can scale to meet needs and demands. Several existing studies have compared MapReduce and Parallel DBMS. A strong interest towards this field is arising in the literature actually.

A good amount of literature has been dedicated to provide a broad comparison of the two technologies. In [7], McClean et al. provide a highlevel comparison between MapReduce and Parallel DBMS. The authors provide a selection of criteria that can be used to choose between MapReduce and Parallel DBMS for a particular enterprise application and to find the best choice for different scenarios. However, in [8], Nance et al. offer a summary of the different arguments for a mix of both traditional RDBMS and NoSQL systems, as they are often designed for and solve different problems. Pavlo et al., to evaluate both parallel DBMS and the MapReduce model, conduct the experiments in [12]. This work It has been noted that the MapReduce model lacks of computation time. However, it showed the superior performance of MapReduce model relative to parallel database systems in scalability and fault tolerance.

In many research works, MapReduce has been presented as a complement to the DBMS technology. Stonebraker et al. in [5], compare MapReduce with the parallel DBMS. This work argues that the parallel DBMS showed clear advantages in efficient query processing and high-level query language and interface, whereas MapReduce excelled in ETL and analytics for “read only” semi-structured data sets. In this work, MapReduce is considered as a complement to the DBMS technology rather than competitor with it, since databases are not designed to be good at ETL tasks. Also, in [9] Gruska and Martin have been working to integrate strengths of the two technologies and avoid weaknesses of them. Researchers consider the two systems RDBMSs and MapReduce as complimentary and not competitors. This work aims to present different types of integration solutions between RDBMS and MapReduce, including loosely and hightly integration solutions.

Many research works aim to use the two approaches together. Yui and Kojima propose in [10] propose a purely relational approach that removes the scalability limitation of previous approaches based on user-defined aggregates. A database-Hadoop hybrid approach to scalable machine learning is presented, where batch-learning is performed on the Hadoop platform, while incremental-learning is performed on PostgreSQL. HadoopDB, presented in [11] by Abouzeid et al., attempted to bridge the gap between the two technologies and is adopted for illustrating the performance of MapReduce on processing relational data stored in database systems. It benefits from DB indexes by leveraging DBMS as a storage in each node.

Many research work are conducted to compare MapReduce and database systems. Some presented the use of DBMSs and MapReduce as complementary paradigms, others as competing technologies.

Several studies underlined the performance limitations of the MapReduce model, in terms of computation time, and explained this lack by the fact that the model was not originally designed to perform structured data analysis. Other works discuss the case of many data analysis tasks, where the algorithm is complex to define. The application requires multiple passes over the data, and operations are dependents and complementary where the output from one is the input to the next. For these complex applications, MapReduce provides a good alternative since it allows the user to build complex computations on the data, without the limitation and the difficulty of the SQL language to build such algorithms.

In the other hand, several research studies have been conducted between MapReduce and RDBMSs in the goal of integration, but all these works evaluate the model having as a metric the computation time and efficiency. No attempt was carried out to analyse the I/O execution cost.

3 MapReduce-PostgreSQL Integration Model

Organizations look to choose the technology to be used to analyze the huge quantity of data, and look for best practices to deal with data volume and diversity of structures for best performance and optimize costs. The question is about the selection criteria to be considered in processing the data while meeting the objectives of the organization.

Data processing needs are changing with the ever increasing amounts of both structured and unstructured data. While the processing of structured data typically relies on the well developed field of relational database management systems (RDBMSs), MapReduce is a programming model developed to cope with processing immense amounts of unstructured data.

For this purpose, we suggest a model to integrate the RDBMS PostgreSQL and the MapReduce framework as one of the Big Data solutions in order to optimize the OLAP queries Input/Output execution cost. We aim to integrate classic data management systems with Big Data solutions in order to give businesses the capability to better analyze data with a goal of transforming it into useful information as well as minimizing costs.

3.1 Inspection

The basic idea behind our approach is based on the cost model to approve execution and selectivity of solutions based on the estimated cost of execution. To support the decision making process for analyzing data and extracting useful knowledge while minimizing costs, we propose to compare the estimates of the costs of running a query on Hadoop MapReduce compared to PostgreSQL to choose the least costly technology. For a better control, we will proceed to a thorough query analysis.

The detailed analysis of the queries execution costs showed a gap mattering between both paradigms. Hence the idea of the thorough analysis of the execution process of each query and the implied cost. To better control the cost difference between costs of Hadoop MapReduce versus PostgreSQL on each step of the query's execution process, we propose to dissect each query to a set of operations that demonstrates the process of executing the query. In this way, we can check the impact of the execution of each operation of a query on the overall cost and we can control the total cost of the query by controlling the partial cost of each operation in the information retrieval process.

In this context, we are inspired from a work done in [13] to provide a detailed execution plan for OLAP queries. This execution plan zooms on the sequence of steps of the process of executing a query. It allows detailing the various operations of the process highlighting the order of succession and dependence.

In addition to dissect the implementation process, the execution plan details for each operation the amount of data by the accuracy of the number of records involved and the dependence implemented in the succession of phases. These parameters will be needed to calculate the cost involved in each operation. After distinguishing the different operations of the query, the next step is to calculate the unit cost of each operation. As part of our approach we aim to dissect each query and focus on each separate operation. That way we can control the different stages of the execution process of each query with the aim of calculate the cost implied by each operation as well as its influence on the total cost of the query. Therefore we can control the cost of each query to support the decision making process and the selectivity of the proposed solutions based on the criterion of cost minimization.

3.2 Queries Cost Estimation

Having identified all operations performed during the query execution process the next step is then to calculate the cost implied in each operation independently, in both paradigms PostreSQL and MapReduce with the aim of controlling the estimated costs difference according to the operations as well as the total cost of query execution. At this stage we consider each operation independently to calculate an estimate of its cost execution on PostreSQL on one hand then on MapReduce on the other hand.

Cost estimation on MapReduce. In a MapReduce system, a query star join between F (fact table) and n dimension tables, runs a number of phases, each phase corresponds to a MapReduce job. So, for MapReduce paradigm we have first to extract the number of jobs that will be run for executing the query. The MapReduce job number depends on the number of joint and the presence or absence of aggregation and sorting data. There are three cases of figure: The request contains only n successive joint operations between F and n dimension tables; Join operations are followed by a process of grouping and aggregation on the results of the joint; Sort is applied to the results. In this context, we propose to rely on the equation (1) presented below, and inspired from [14]. This equation allows to determine the number of MapReduce jobs implied in the execution of a given OLAP query. This number can be estimated by the following formula:

$$Nbrjobq = n + x \qquad (1)$$

The “n” refers to the number of dimension tables. The x can be equal to: 0, if the query involves only join operations; 1, if the query contains grouping operations and aggregation; 2, if the results are sorted. After identifying all the jobs of query execution, the next step is to calculate the Input/Output cost implicated in each job. In this stage, we relied on the mentioned work presented in [BOK 14] to extract the amount of data and the number of records involved in each operation. These two parameters will be used in a MapReduce cost model that we implemented to approve our proposed approach, in order to calculate the Input/Output cost of each job.

Cost estimation on PostgreSQL. PostgreSQl provides the possibility of itemize each operation by an incremental value of the Input/Output cost implied in each step.

In PostgreSQL platform, the command “explain” shows the execution plan of a statement. This command displays the execution plan that the PostgreSQL planner generates for the supplied statement. Besides the succession of the executed operations, the most critical part of the display is the estimated statement execution cost (measured in cost units that are arbitrary, but conventionally mean disk page fetches). It includes information on the estimated start-up and total cost of each plan node, as well as the estimated number of rows. Actually two numbers are shown: the start- up cost before the first row can be returned, and the total cost to return all the rows. Therefore for each operation, the cost should be the difference between these two values.

3.3 Analysis

The analysis of the results of the estimated costs independently for each operation showed the high cost of the first join operation executed on PostgreSQL, and a noticeable difference for the Hadoop MapReduce paradigm. This can be explained by the fact that in the case of data warehouses, the fact table is still the largest table in terms of number of tuples, which explains the high cost of its analysis. The figure 1 illustrates the process of an OLAP query in our proposed smart model. Based on the previous ascertainment, we propose performing the first joint operation that integrates fact table on MapReduce framework which proves competence for heavy processing to be performed on a large volume of data. In this way we try to minimize the cost of the query execution by minimizing the cost of the most expensive operation. Then the output of the first sub process will be the input of the subsequent operations. Thus, other operations required by the query such as aggregation, sorting, in addition to other join operations will be passed on PostgreSQL.

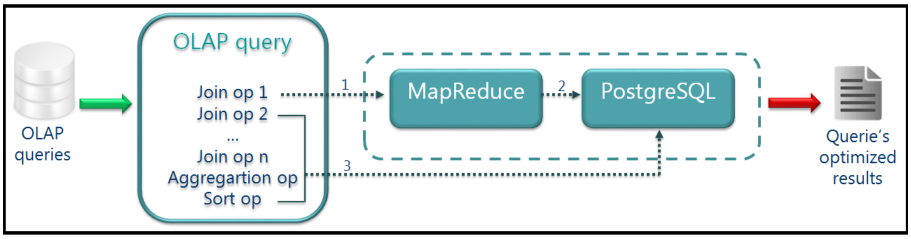

Fig. 1. MapReduce-DBMS smart model

4 Experimental Results

In order to validate our proposed approach, we present in this section the results of experiments conducted to evaluate the proposed model performance as well as gain cost compared with the cost required by each platform independently.

4.1 Experimental Environment

The experiments involve two DBMSs: an ORBMS PostgreSQL and a NoSQL DBMS Hadoop MapReduce. To test and compare our theoretical expectations with the real values, we set up a cluster consisting of one node. For all the experiments, we use the version 9.3 of PostgreSQL. For Hadoop, we used the version 2.0.0 with a single node as worker node hosting DataNode, and as the master node hosting NameNode.

We worked on a workload of 30 queries OLAP. The training data consisted of a data warehouse of 100GB of data with a fact table and 4 dimension tables.

4.2 Discussion

Our approach proposes an hybrid model between ORDBMS and Hadoop MapReduce, based on the comparison of Input/Output costs on the both paradigms. The results of the application of the proposed approach are presented in the table 1. It shows the total Input/Output cost of running the workload on Hadoop MapReduce (Tot_cost_MR), the total Input/Output cost of running the workload on ORDBMS PostgreSQL (Tot_cost_PG), the total Input/Output cost of the workload by applying our proposed approach (Tot_cost_intg), the total Input/Output cost of the workload by choosing for each operation of each query the lowest cost between Hadoop MapReduce and PostgreSQL (Best_Cost_MR_PG).

Table 1. Proposed approach results

Tot_cost_MR	Tot_cost_PG	Tot_cost_intg	Best_Cost_MR_PG
505579305,9	786297800,8	369473436,7	365360413

Observing the values presented in table 1 illustrate the difference of the cost saved up by the application of the proposed approach. The total cost of running all the workload under the proposed approach is only slightly over predicted compared to the cost estimated cost to run the workload by choosing for each operation of each query the lowest cost between the two proposed paradigm (Hadoop MapReduce and PostgreSQL).

The presented values shows the gain of the Input/Output cost by running the workload independently on Hadoop MapReduce and on postgreSQL. In addition, we can compare the Input/Output cost of the workload obtained by applying our proposed approach to the Input/Output the cost obtained if we choose the lower cost for each operation contained in the execution plan of each query.

The table illustrates the notable difference of the Input/Output cost of running the workload by applying the proposed approach compared to the Input/Output running cost on the two paradigms PostgreSQL and Hadoop MapReduce separately.

The gain estimated of the Input/Output cost saved up by running the workload thanks to applying the proposed approach is of 27% compared to Hadoop MapReduce and 53% compared to PostgreSQL. This percentage gain proves the performance of our proposed approach. In addition, the cost returned by our proposed approach is 98.8% of the optimal cost obtained in the case of choosing for each operation of each query the lowest cost between Hadoop MapReduce and PostgreSQL.

5 Conclusion

Given the exploding data problem, the world of databases has evolved which aimed to escape the limitations of data processing and analysis. There has been a significant amount of work during the last two decades related to the needs of new supporting technologies for data processing and knowledge management, challenged by the rise of data generation and data structure diversity.

Both technologies, MapReduce model and relational database systems present strengths and weaknesses. To provide an effective and efficient data management and analysis system, the author believes that Combining MapReduce and RDBMS technologies has the potential to create very powerful systems. In this paper we have proposed a smart model integrating the MapReduce model and a RDBMS, in a goal of optimization of OLAP queries I/O execution cost.

We expect future releases to enhance performance. We plan to deal with OLAP queries with many imbrications and to highlight their influence on the cost.

References

1. Sagiroglu, S., Sinanc, D.: Big data: a review. In: Collaboration Technologies and Systems (CTS) 2013 International Conference on IEEE, pp. 42–47 (2013)
2. Narasimhan, R., Bhuvaneshwari T.: Big Data - A Brief Study. International Journal of Scientic and Engineering Research **5**(9) (2014)
3. Dean, J., Ghemawats, S.: Mapreduce : Simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
4. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving MapReduce performance in heterogeneous environments. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation OSDI, vol. 8, no 4, pp. 29–42 (2008)
5. Stonebraker, M., Abadi, D., Dewitt, D., Madden, S., Paulsone, E., Pavlo, A., Rasin, A.: Mapreduce and parallel dbmss: friends or foes? Communications of the ACM **53**(1), 64–71 (2010)
6. Douglas, K., Douglas, S.: PostgreSQL: A comprehensive guide to building, programming, and administring PostreSQL databases, 1st edn. Sams Publishing (2003)
7. McClean, A., Conceicao, R., O'halloran, M.: A comparison of MapReduce and parallel database management systems. In: ICONS 2013 The Eighth International Conference on Systems, pp. 64–68 (2013)

8. Nance, C., Losser, T., Iype, R., Harmon, G.: NOSQL vs RDBMS – why there is room for both. In: Proceedings of the Southern Association for Information Systems Conference Savannah USA, pp. 111–116 (2013)
9. Gruska, N., Martin, P.: Integrating MapReduce and RDBMSs. In: Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research, IBM Corp., pp. 212–223 (2010)
10. Yui, M., Kojima, I.: A database-hadoop hybrid approach to scalable machine learning. In: IEEE International Congress on Big Data 2013, pp. 1–8. IEEE (2013)
11. Abouzeid, A., Pawlikowski, K., Abadi, D., Silberschatz, A., Rasin, A.: Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads. In: Proceedings of the VLDB Endowment, pp. 922–933 (2009)
12. Pavlo, A., Rasin, A., Madden, S., Stonebraker, M., Dewitt, D., Paulson, E., Shrinivas, L., Abadi, D.: A comparison of approaches to large scale data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 165–178 (2009)
13. Boukorca, A., Faget, Z., Bellatreche, L.: What-if physical design for multiple query plan generation. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014, Part I. LNCS, vol. 8644, pp. 492–506. Springer, Heidelberg (2014)
14. Brighen, A.: Conception de bases de données volumineuses sur le cloud. In: Doctoral dissertation, Université Abderrahmane Mira de Béjaia (2012)
15. Demirkan, H., Delen, D.: Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. Decision Support Systems **55**(1), 412–421 (2013)
16. Ordonez, C.: Can we analyze big data inside a DBMS? In: Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP, pp. 85–92. ACM (2013)
17. Brown, P.G.: Object-Relational Database Development: A Plumber's Guide. Prentice Hall PTR, USA (2000)
18. Chandrasekar, S., Dakshinamurthy, R., Seshakumar, P.G., Prabavathy, B., Babu, C.: A novel indexing scheme for efficient handling of small files in hadoop distributed file system. In: 2013 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–8 (2013)