BigDimETL with NoSQL Database

Hana Mallek[1*], Faiza Ghozzi[1], Olivier Teste[2], Faiez Gargouri[1]

[1]*MIRACL Laboratory, Institute of Computer Science and Multimedia Sfax, BP 1030, Tunisia*
[2]*Université de Toulouse, IRIT 5505, 118 Route de Narbonne, 31062 Toulouse, France*

Abstract

In the last decade, we have witnessed an explosion of data volume available on the Web. This is due to the rapid technological advances with the availability of smart devices and social networks such as Twitter, Facebook, Instagram, etc. Hence, the concept of Big Data was created to face this constant increase. In this context, many domains should take in consideration this growth of data, especially, the Business Intelligence (BI) domain. Where, it is full of important knowledge that is crucial for effective decision making. However, new problems and challenges have appeared for the Decision Support System that must be addressed. Accordingly, the purpose of this paper is to adapt Extract-Transform-Load (ETL) processes with Big Data technologies, in order to support decision-making and knowledge discovery. In this paper, we propose a new approach called Big Dimensional ETL (BigDimETL) dealing with ETL development process and taking into account the Multidimensional structure. In addition, in order to accelerate data handling we used the MapReduce paradigm and Hbase as a distributed storage mechanism that provides data warehousing capabilities. Experimental results show that our ETL operation adaptation can perform well especially with Join operation.

Keywords:ETL, Hbase, BigData, Twitter, Join operation

* Corresponding author. *E-mail address:* mallekhana@gmail.com

1. Introduction

Decisional Support System architecture [1] is composed of three important phases: Data Warehouse (DW) building, exportation and ETL processes which are responsible for Extracting, Transforming and Loading data into a multidimensional DW. Hence, the Extraction phase covers all tasks to collect the required data. While the

10.1016/j.procs.2018.08.014

Transformation phase is the execution of a series of operations to transform the extracted data into standard formats and Loading phase processes extracted and transformed data into the DW target storage. Thus, a DW is a data collection designed according to a dimensional modelling which is dedicated to On Line Analytical Processing (OLAP) and Business Intelligence (BI) applications to deliver useful data for support business decisions. According to [1], a DW is implemented as a relational databases which is structured as a star schema. In fact this schema has for objective to observe facts through measures, according to the dimensions that represent the analyses axes.

Nowadays, with the broad range of data available on the web, very interesting business goals could be achieved with the collecting and performing analytics of social networks data such as Twitter, Facebook, Instagram, etc. For instance, Twitter is one of the most popular social media with more than half billion of users [2]. According to IBM[1*], 400 million tweets are sent per day by about 200 million monthly active users. Thus, the increasing use of social media produces the "Big Data" notion which refers to massive data sets of unstructured data. In fact, this huge amount of data is often defined by the 3Vs characteristics, i.e. Volume (size of data), velocity (the speed of generating, capturing and sharing of data), and variety (structured and unstructured data sources). In such a context, data become difficult to capture, store, manage and analyse via typical technologies and databases. Accordingly, it is a critical importance to process the useful information in an efficient manner from the massive volume of data. So, the emergence of Big Data creates big challenges in the Business intelligence (BI) and data warehousing domain, where RDMSs are not suitable for distributed databases as argued in [3]. Furthermore, typical ETL processes of data integration cannot support this big evolution of data. In fact, ETL is composed of several operations such as (Select, Conversion, Concatenation, Join, etc.) executed sequentially. After each operation Data Storage Area (DSA) stores extracted data at regular intervals such as daily, weekly or monthly. Therefore, to handle these issues, we propose a multidimensional big data architecture based on ETL operations called BigDimETL. In fact, this paper is a continuity of our work presented in [4] where we gave a brief description of BigDimETL solution. Therefore, in this work, we present a set of modifications and enhancements to our solution to be more efficient in the Big Data context. Also, we demonstrate the importance to treat data after vertical partitioning in order to build the adopted multidimensional structure.

Our approach is based on new technologies appeared with big data domain such as Hadoop [5], which is an open source framework for handling unstructured data using a parallel processing technique called MapReduce [6]. In addition, we use Hbase as a column-oriented data store instead of classical relational database or plat files (CSV, XML, etc.). It is worth to mention that most of these new breeds of new technologies support the scalability and the performance for integration process.

The remainder of this paper is organized as follows: section 2 presents related work, pointing the main limitations identified in the state-of-the-art. Section 3 cites a brief description of our architecture as well as several key concepts. Section 4 describes our solution to adapt ETL operations to the Big Data context through Join Algorithm. Section 5 shows the experimental results to illustrate the BigDimETL operations performance, while section 6 concludes with some remarks and guidelines for future work.

2. Related work

The aim of this section is to present some existing approaches that deal with data integration. Hence, in literature, ETL processes gained widespread attention especially on modelling its processes. Hence, in the last 15 years, these processes have evolved to maturity by overcoming many limitations by giving many benefits on data warehousing performance [7][8]. Nowadays, with the explosion growth of data several technologies have appeared for data processing like Hadoop, data storage like HBase[9] and query processing Hive[11], Pig[12]. Nevertheless, some challenges remain still open. Mainly, they are related to the management of ETL processes, unstructured data, etc. Thus, it is very crucial to use these technologies to support Big Data requirements, especially to respect the 3Vs characteristics, which must be considered while implementing ETL processes. In fact, previous works have shown

[1]http://www.ibmbigdatahub.com/infographic/four-vs-big-data

the power and the gain in time for processing and storing large volumes of data using Hadoop framework. This latter has one of the most widely used methods for process data called MapReduce. In [13] and [14] we find that MapReduce used to add a parallelism to the several operations of integration to minimize the time consumption for handling data. In fact, in DW context, authors in [15] and [16] use MapReduce paradigm to add the parallelism concept to ETL processes in a physical level of integration. Authors in [15], propose a framework called P-ETL that presents a parallel ETL through the MR paradigm. Other works use MapReduce with a distributed database to meet the reliability and scaling needs, instead of relational database such as MongoDB and Hbase for structured and semi-structured data. Using these storage technologies can improve several interests, where [17] presents a review of different NoSQL databases and the importance of these databases for enterprises to improve the scalability and the high availability. In this case, several approaches concentrate to moving from relational to NoSQL database through column-oriented database especially Hbase. [18] Proposes a new solution for heuristic transformation to transform relational database into Hbase. Similarly, [19][20] and [21] use Hbase without considering ETL processes specificity on big Data context. Also [3] shows how Hbase is very performed to improve OLAP querying in DW context through the multidimensional structure. However, ETL was absent in their work. Authors in [22] demonstrate that the performance of OLAP queries can be improved if the data warehouse is perfectly selected and integrated. Otherwise, [23] ignore ETL, but it loads data on BigTable which is considered as a DW.
For operations processing such as (Select, Join, etc.), we find some works that adapt classical SQL query on Big data context through adding MapReduce paradigm like [24] with JackHare framework also Hadoop++ with [14], and [25] it ensures the performance of MapReduce with Join operations.
In [26], authors use MapReduce and Hive system to define a Framework called CloudETL that supports the representation of DW Star schema. [27] Uses Hive and Hadoop to support DW in a cloud computing environment and to build OLAP cubes. This work employs the parallelism strategy while integrating data, without considering the MDS, which is considered as the paramount stage for further analysis operations.
In summary, we could say that MapReduce is appropriate to serve long running queries in batch mode over raw data (ETL). In fact, these previous studies are very interesting in the data warehousing context. Although, these works share some similarities with ours, but the partitioning, operations processing of ETL and temporary storage aren't covered. As a conclusion, our goal is to reduce the cost of DW implementation and produce relevant information for decision support. Among the possible solutions, the MapReduce paradigm proves a powerful solution for parallel processing of massive data. Thus, we propose to handle the extraction and the transformation phase according to a multidimensional structure from the first step using vertical partitioning and Hbase as a distributed data storage.

3. Overview of BigDimETL with NoSQL database

Our proposed architecture BigDimETL presented in fig.1 represents ETL processes with MapReduce paradigm through NoSQL Column Oriented database as distributed data storage. In the following sub-sessions, we define the main concepts needed for our solution:

3.1. Column-oriented NoSQL databases

As mentioned previously, Column-oriented databases can be a good alternative to RDBMS. Data in Column-oriented structure is represented as a table and grouped by columns. Each set of columns called Column Families.
Hence, our Table (T) is represented vertically as a list of Rows R_i:T= $\{R_1,\ldots,R_n\}$, where i $\in$[1..n].
Each row Ri is composed of several elements, where R_i=$\{RK_i, CF_1,\ldots,CFp, V_i, TimeS_i\}$. In this case, each R_i is identified by a Row Key (RK_i) and at least one Column Family (CF) and a version (V_i) to keep track of changes by storing different versions of row which are identified by a timestamps ($TimeS_i$). In this paper, the versioned principal is not treated. Each Column Family CF_m is defined as: CF_m=$\{C_1, \ldots,C_k\}$.Where each CF_m groups an arbitrary number of columns C_j where j $\in$[1..k]. In this case, each CF can gather columns that have the same categories of attributes which called also Column Qualifier (CQ) where the access to data within a CF is done through it

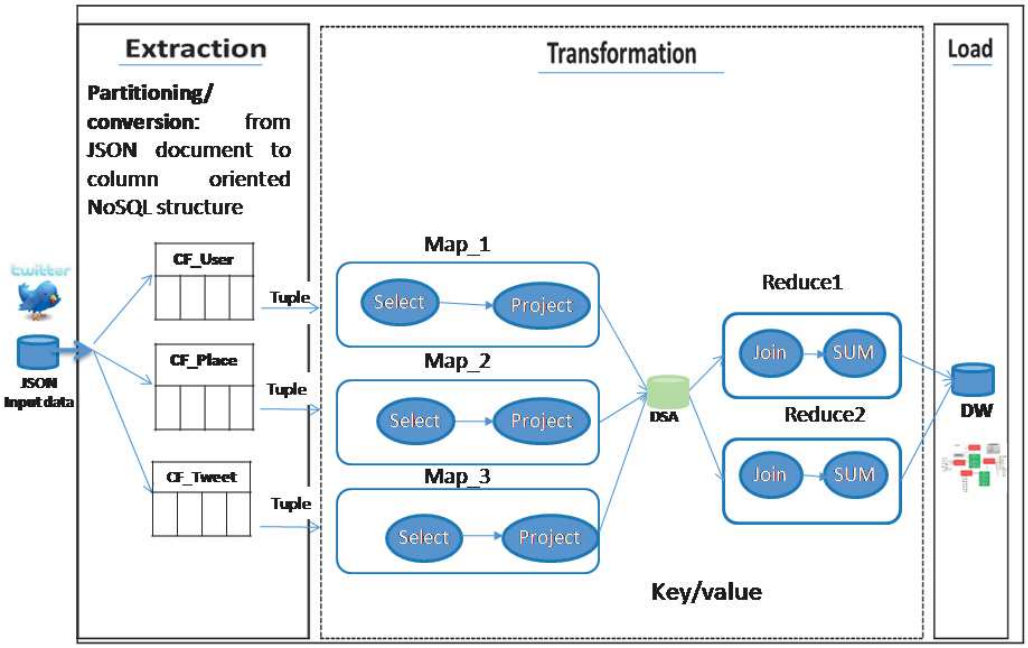

Fig.1 BigDimETL with NoSQL database

3.2. Multidimensional structure

The multidimensional structure of a DW is represented as star schema. This schema is composed of a Fact (F) table to present the subject of analysis. Where, F is linked to associate Dimensions (D_j) to represent several dimensions of analysis, where $j \in [1..n]$.So, the formal structure of F and D is as follows: $F = \{Fk_1,..,Fk_k, m_1,\ldots,m_L\}$; $D=\{att_1,\ldots,att_n\}$. Each F_i, where $i \in [1..k]$ is composed of diverse calculated numerical measures m_j where $j \in [1..L]$ and related to Dimensions through foreign key (Fk). Each dimension D_j regroups a set of attributes, which is defined as $D_j=\{att_1,\ldots,att_n\}$.This structure is important for OLAP Analysis operations. Hence, we can't overlook the importance of the multidimensional structure.

4. Adapted ETL processes through MapReduce paradigm

The MapReduce paradigm is a framework developed by Google in 2004 under the Hadoop ecosystem. It is described as a programming model used for a parallel processing of massive data sets. The input data must be divided into several horizontal partitions according to the default size (64 MB). Each split is browsed separately by several jobs to perform transformation tasks through Map and reduce functions. Where these functions are characterized by its simplicity, which is one of the attractive qualities of MapReduce [28]. The first function Map takes key/value pairs from splits as input. The resulted key/value pairs play the role of input value for the second function Reduce.
Formally, we can describe MapReduce mechanism as follows:

- List of Jobs $J= \{J_1,..,J_n\}$, where each job is composed of Map Functions (MF) and Reduce Functions (RF).
- In this case, $J_i=[MF, RF]$ and could be presented as $J= \{(MF_1, RF_1),..,(MF_k, RF_k)\}$.
- MapReduce considers input data to MF as Key/value pairs (k_i,v_i)

Then, the RF regroups all output data in a list ($List\_V_i$) with several values attached to a specific key k_i : (k_i, $List\_V_i$)

ETL processes are described as three principal phases which are Extraction, Transformation and Loading. These processes are considered as the more crucial steps for DW implementation [7].

In this paper, we will describe several steps of adaptation of these processes using the principal of MapReduce paradigm.

4.1 Extraction phase

This phase is responsible for the extraction and preparation of the input data. In our context, data is extracted from Twitter social media. This latter is extracted as JSON (Java Script Object Notation) format which is considered as a complex structure and can't be the optimal solution to be treated with ETL processes.
In order to use our Input data in a distributed way with MapReduce paradigm, our data will be partitioned horizontally automatically according to the size of split (64MB). Furthermore, our idea is based on regrouping data into axes of analyses CF. It gives the importance to add a vertical partitioning to our input data. It is worth to mention that vertical partitioning is very significant, to create the multidimensional structure of DW. Where, each CF corresponds to a specific D_i or F. In this case, each job J_i takes a specific CF and each CF is full of D_i attributes att_i. The extraction phase is based on the conversion of the input data from object oriented, into column oriented structure. In this case, all objects (O_i) of JSON file are converted into a CF and their CQ are the list of sub attributes of object O_{attj}.

- Rules of transformation: O_i -> CF and O_{attj} ->CQ

For example, the User object is considered as a Complex Attribute (CA) of a tweet and represents as attributes user_name, user_id, etc. Also, the Direct Attribute (DA) is concerned the tweet itself so it is stored under the same FC called "Tweet". In this case, the fact here is the "Tweet" CF and all other objects represent a dimensions D_i.
Example. Let's consider the example of tweet (Tw) on a JSON file, which is composed of DA and CA. We can consider a simple multidimensional structure where a tweet is the Fact table, and User, Place, Time and Date, are their dimensions [29] .
In our example of twitter data, we have CF_User which represents all information about user such as (Name, Id, Language, etc.), CF_Place which represents all information about place (place_type, place_Id).

4.2 Transformation phase

The transformation phase is responsible for handling the extracted data. Typically, as defined by [7] and [4] this phase can be composed of several critical operations (Select σ, Project π, Union U, Join ⋈ , etc.). Which are classified into two families: Elementary Operation (EO) and Complementary Operations (CO). While EO requires as input one operation (Select, Project operations). EO={π, σ} ; and CO requires two or more operations as input (Union, Intersection, and Join). CO= {U, ∩,⋈ }.
Our approach is based on MapReduce paradigm to add the parallelism aspect for classical ETL operations in order to minimize the time consuming of data processing. Therefore, we need to adapt ETL operations through MF and RF. In the following, we focus on Select, Project and Join operations:

- **Select, Project operations**

The major objectives of ETL are cleaning and regrouping data which are done using select and project operations. In [4], we talked about Select from CSV (Comma Separated Values) file where the information is represented as a text separated by a comma. But the conversion process can have time loss where NoSQL database is able to structure the JSON data as Columns oriented. Hence, this latter makes our input data as fact and dimension in CF. The select operation reflects, in general the restriction of data where we have the Where clause. The basic syntax of select statement in relational algebra is as follows: $\sigma\ (p)_{CF}$,where p is the selection condition or a predicate. This predicate is applied independently to each individual tuple t in a CF (vertical partition).
Example: σ(User_Name="Alex")$_{Twitter\_User}$. For the project operation, it is responsible to extract which columns are important for the resulting query. The project operation can be presented as follows: $\pi_{(col1, col2, ..)}$, where col_1, col_2 are which columns to be selected.

- **Join operation**

The join operation is treated in several works, notably with MapReduce to add the parallelization aspect such as in [24] which deals with cross join using HBase. On the decisional business domain and during data warehousing

processes, joining operation with ETL attends big lack with big volume of data. Accordingly, we need to join data extracted from several selected results. When we have a complex query, we need as input several tables, so each temporary result from each table will be buffered on a temporary table called DSA (Data Storage Area). This latter, is very important to take it into consideration in our solution to minimize the volume of handled data. For more details, as shown in Fig.1, our solution is based on NoSQL database which is used on several storage steps of BigDimETL: as input, output and temporary table. However, after the select operation, we should join resulted data through a unique key. Where each row R_i from NoSQL Temporary table (Ntt) has to be joined with rows R_j that have the same join key (JK) value. Our objective here is to join data through NoSQL column oriented database using MapReduce paradigm. Hence, we need to redefine our main Map and Reduce functions. In general, the Map function needs as input a set of distinct pairs key K_i and value V_i. Our solution is reasoning about different join implementation in MapReduce in the literature. Thus, we can't overlook the performance of the existing works [24] and [25]. To do the Join operation, we need the select operation in which on the predicate clause, we should indicate the condition of Keys equality. For example to select each user with its tweets, we have as expression $\sigma(CF\_User.User.Id=CF\_Tweet.user.Id)_{CF\_User, CF\_Tweet}$. In this case, we need implicitly the project operation to point into CF_User, and CF_tweet and the predicate "CF_User.User.Id=CF_Tweet.user.Id" is responsible to indicate the join condition. Our general algorithm that aims to call several ETL operations is presented in the following:

Algorithm 1. ETL operations processing

Input: req= input user query, NumReducer= number of reducers
1: Parsing query
2: List_CF ← Identify which CF will be projected
3: List_CQ ← Identify which CQ will be selected
4: List_C ← Identify list of conditions on predicate clause
5: Connect to NoSQL Table T
6: Subdivide vertically Table T on several T_CF according to CF
7: Call ProjectJob(T_CF, NumReducer)
8: Create Ntt T_CF1+T_CF2 to join
9: Call JoinJob(T_CF1+T_CF2, NumReducer)

In this algorithm, the two procedures "ProjectJob and JoinJob" are called respectively on line 7 and 9. The first one aims at selecting the required CF by calling MF and RF [4]. While, JoinJob procedure is responsible for verifying the equality of JK of two tables T_CF1 and T_CF2 and creates an Ntt which concatenate these two tables.
The MF and RF algorithm of JoinJob procedure is described as follows:

Algorithm 2. Map of Join function

Input T_CF1, T_CF2
1: Tab_key_1← all RK of T_CF1
2: Tab_Key_2← all RK of T_CF2
3: if (Tab_key_1[i]= Tab_Key_2[j]){
4: CommonKey ←Tab_key_1[i]
5: }
6: add common key and its values to Ntt
7: emit (CommonKey, values of different CQ related to the CommonKey)

Algorithm 3. Reduce of Join function

Input Ntt = NoSQL Temporary Table
1: List_V← selected rows from Ntt
2: foreach cell in List_V {
3: values ← RK.getCF(), RK.getCQ(), RK.getValue()
4: }
5: emit(RK, values)
6: **end.**

Ihe **Map function** in Algorithm.2, we can attribute the row key RK as a key for each table T_CF1, T_CF2. If there is a correspondence between these two tables "commonkey" variable will be created and considered as a key value. The rest of the commonkey row is affected as a value. These temporary results should be saved on an Ntt.
For **Reduce function** in Algorithm.3**,** we have as input the Ntt which is affected to a List_V, then we use the "Values" variable to integrate each CF, CQ and with their values into Column Oriented Database. At the end the output result is the pairs of values (RK, values)

5. Experiments and Evaluation

Performance Metric. Evaluating the effectiveness of databases is mainly measured through the response time of executing an input query as a standard performance metric for comparing query engines. Thus, in our context, the response time is defined as the time between submitting a query to the querying engine and getting the query response.
Experimental Environment Description. All experiments have been performed on OSIRIM platform which relies on the distribution Hadoop Hortonworks 2.3. It consists of 640 core computing clusters distributed over 17 computing servers with 28 Nvidia GTX 1080 TI cards (3584 cuda cores each) distributed over 7 servers. As database management system (DBMS), we choose Hbase as a distributed column oriented database with version of1.4.1. We compare the performance of our method with Oracle SQL DBMS for highlighting well the major improvements of exploiting our method.
Queries. Three practical data retrieval queries are used demonstrating the effectiveness of our proposed method, varying in their complexity at joining tables' level. The formal descriptions of the queries are described in Table 1, while the main purposes of having such queries are illustrated as follows:

- **Query 1:** aims at retrieving information from three different tables so that a join operation is required to have the final aggregated results. It targets to retrieve meta-data about user name, text of tweet, and name of place that the user has been checked-in. This query is the most complex one in our experiment since it performs join among three tables.
- **Query 2:** less complex than the query 1 so that we exclude the place table of tweets from the query and thus the join operation will be performed between two tables only.
- **Query 3:** is the simplest one compared to other queries so that it is responsible to retrieve tweet records from single table only without performing any kind of join operations between other tables.

Twitter Data. In filling the databases, we have exploited our team public Twitter data crawled through using streaming APIs provided by Twitter for developers. The data are stored in JSON format split among equal size files of tweets where each file consists around 4k tweets. The volume of Twitter data considered in the experiments is about 250k tweets.

5.2 Experiment Results

Before introducing the response time results of the three queries described above, we first recall some key-points that can facilitate the interpretation of results. A DBMS is a piece of software that runs on a computer as well as it is subject to the same limitations of other software. DBMS can only process as much information as its hardware is capable of handling. The optimal way to make a query run faster is through reducing the number of calculations that the DBMS (and therefore hardware) must perform. Indeed, high-level factors could affect the number of calculations that DBMS needs, and consequently the runtime of queries:

Table 1 Statements of ETL queries

Queries statements
Q_1: SELECT Twitter_User.User.Name, Twitter_Tweet.Tweet.Text, Twitter_Places.Place.Name WHERE Twitter_User.User.Id =Twitter_Tweet.User1.Id AND Twitter_Places.Place.Id = Twitter_Tweet.Place1.Id
Q_2: SELECT Twitter_User.User.Name, Twitter_Tweet.Tweet.Text WHERE Twitter_User.User.Id =Twitter_Tweet.User1.Id

Q_3: SELECT Twitter_User.User.Name

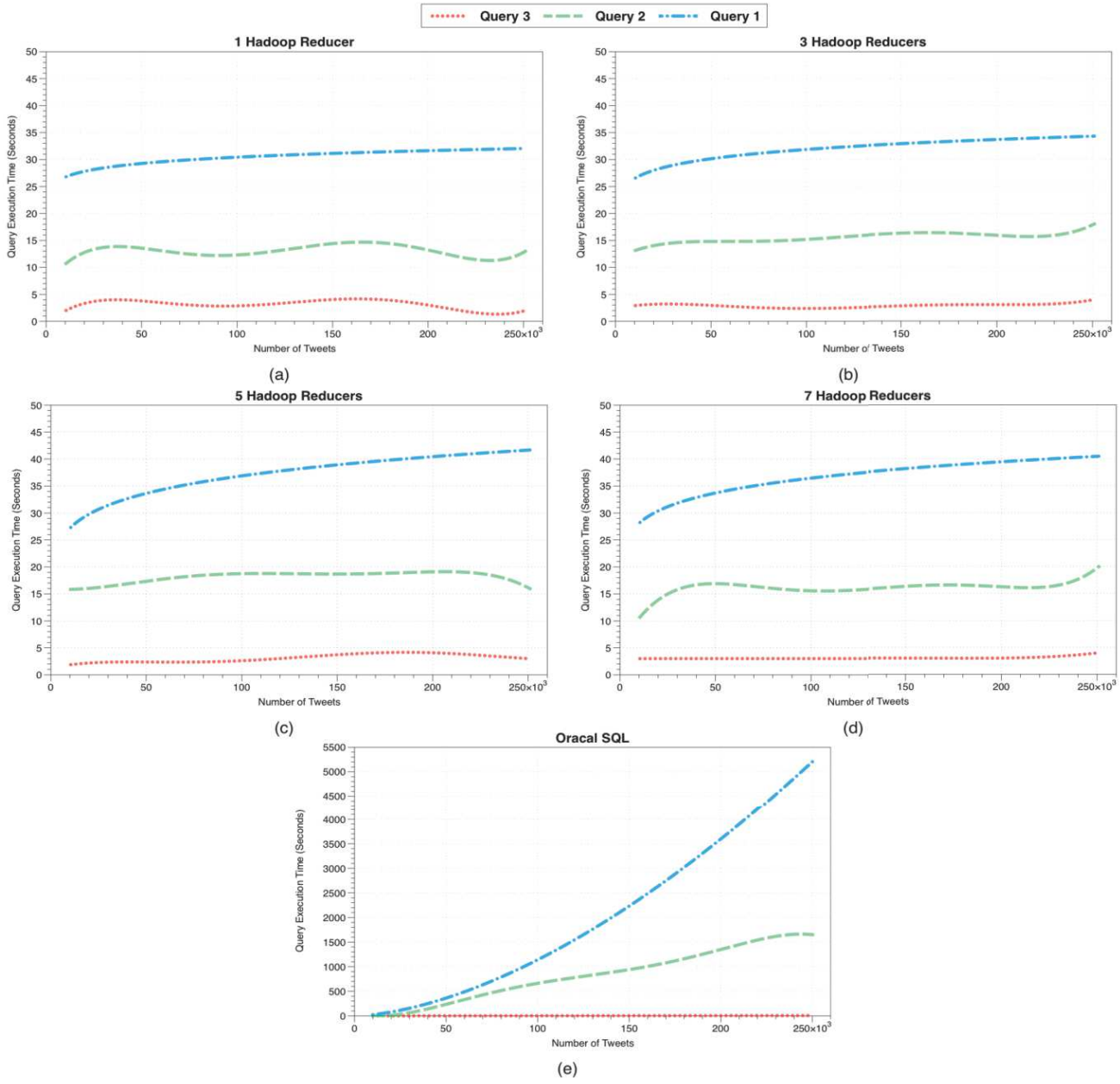

Fig 2 Execution time performance of the three queries reported for our proposed method at different number of reducers and Oracel SQL DBMSs.

- **Table Size:** Using a query hitting one or more tables with millions of rows or more has a direct relation with the increasing of response time.
- **Joins:** Having a query joins tables in a way that substantially increases the row count of the result set, the execution time of the query will increase and thus the response time.
- **Aggregations**: Combining multiple rows to produce a result requires more computation than simply retrieving those rows.
- **Concurrent Queries:** Running more queries on a database in a concurrent way increases the processing time and thus the execution time of each query submitted. Indeed, the execution time of queries will become worse when the running queries are resource-intensive ones that fulfill some of the above factors.

With the description of the DMBS performance based factors, we study the impact of table size factor through increasing the number of tweets in the databases and then measuring the response time of executing the three queries. Thus, for this purpose, we have studied this factor at different number of tweets, including {10k,30k,70k,100k,150k,200k,250k}. The effect of join factor is mainly dependent on the nature of queries that the user design and consequently the introduced three queries perform join among three tables as in Query 1 and two tables as in Query 2, while Query 3 does not perform any kind of join. The aggregations factor is not clear in Oracle SQL DBMS, while in our proposed method it corresponds to the number of Hadoop reducers which aim to aggregate the results that are produced by mappers. Therefore, we study the impact of reducers at different values, including {1,3,5,7}. Figure 2 shows the experimental results of our proposed method at different values of Hadoop reducers and Oracle SQL as a baseline method. It is obvious that the impact of join operation is the most significant factor compared to table size and data aggregation factors. More precisely, our method has a strong correlation with the number of joined tables in the query, while the effect of increasing table size (i.e., number of tweets) is not obvious on query time execution. For instance, in our method, Query 3 has an average and unvaried execution time around 3 seconds regardless the size of tables (data volume), while Query 2 and Query 1 have almost stable and fixed execution query time approximated to 15 and 35 seconds, respectively. As unexpected behavior related to the number of reducers, the experiments show that the number of reducers has a direct proportion with query execution time so that running more reducers leads to increase query response time. Indeed, the most reasonable interpretation for this behavior is related to management of available resources by the operating system, meaning that creating more reducers will be time overhead. On the other hand, the results of Oracle SQL have exponential query execution time behavior with respect to a combination factors of number of joined tables and table size (number of tweets). For instance, the execution of Query when the number of tweets about 250k in the database needs about 5,000 seconds (83 mins), while 1,500 seconds (25 mins) seconds for Query 2. Such results are not comparable with our method at all, making the proposed solution of using Hbase as DBMS with Hadoop an acceptable BigDimETL one to deal with big data problems.

6. Conclusion

In this paper, we have proposed an adapted Join algorithm for ETL processes in our solution BigDimETL. Our approach leads to integrate Big Data with conserving the multidimensional structure of DW through vertical partitioning. In this article, BigDimETL is based on executing several queries using MapReduce paradigm for processing the unstructured data in NoSQL database. On the experiments, our solution is developed based on Hadoop and Hbase to store Twitter data in a multidimensional structure (fact, dimensions). The experimental results show the performance of our proposed join algorithm. As future work, we intend to investigate the apparent advantage in the performance of our proposed solution compared to Pig and Hive.

References

[1] Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, 2nd ed.: John Wiley & Sons, Inc., 2002.

[2] Alfredo Cuzzocrea, Carmen De Maio, Giuseppe Fenza, Vincenzo Loia, and Mimmo Parente, "OLAP analysis of multidimensional tweet streams for supporting advanced analytics," in

[3] Mohamed Boussahoua, Omar Boussaid, and Fadila Bentayeb, "Logical Schema for Data Warehouse on Column-Oriented NoSQL Databases," in *DEXA*, vol. 10439, 2017, pp. 247-256.

[4] Hana Mallek, Faiza Ghozzi, Olivier Teste, and Faiez Gargouri, "BigDimETL: ETL for Multidimensional Big Data," in

[5] Tom White, *Hadoop: The Definitive Guide.*: O'Reilly Media, Inc., 2012.

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, pp. 107-113, 2008.

[7] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos, "Conceptual Modeling for ETL Processes," in *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP*, New York, NY, USA,

2002, pp. 14-21.

[8] Juan Trujillo and Sergio Luj, "A UML Based Approach for Modeling ETL Processes in Data Warehouses," in *22nd International Conference on Conceptual Modeling, 13-16, 2003, Proceedings*, 2003, pp. 307-320.

[9] Lars George, *HBase: The Definitive Guide*, 1st ed.: O'Reilly Media, 2011.

[10] Kristina Chodorow and Michael Dirolf, *MongoDB: The Definitive Guide*, 1st ed.: O'Reilly Media, Inc., 2010.

[11] Ashish Thusoo et al., "Hive: A Warehousing Solution over a Map-reduce Framework," *Proc. VLDB Endow.*, vol. 2, pp. 1626-1629, 2009.

[12] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins, "Pig latin: a not-so-foreign language for data processing," in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1099-1110.

[13] Louai Alarabi, Ahmed Eldawy, Rami Alghamdi, and Mohamed F. Mokbel, "TAREEG: A MapReduce-based System for Extracting Spatial Data from OpenStreetMap," in *Proceedings of the 22Nd ACM SIGSPATIAL*, Dallas, Texas, 2014, pp. 83-92.

[14] Jens Dittrich et al., "Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)," *Proc. VLDB Endow.*, vol. 3, pp. 515-529, 2010.

[15] Mahfoud Bala, Omar Boussa, and Zaia Alimazighi, "P-ETL: Parallel-ETL based on the MapReduce paradigm," in *11th*

[16] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen, "ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce.," *Trans. Large-Scale Data- and Knowledge-Centered Systems*, vol. 8, pp. 1-31, 2013.

[17] Jing Han, E. Haihong, Guan Le, and Jian Du, "Survey on NoSQL database," in *6th International Conference on Pervasive Computing and Applications (ICPCA)*, 2011.

[18] Chongxin Li, "Transforming Relational Database into HBase: A Case Study," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences*, 2010, pp. 683-687.

[19] Khaled Dehdouh, Fadila Bentayeb, Omar Boussaid, and Nadia Kabachi, "Towards an OLAP Environment for Column-Oriented Data Warehouses," in *DaWaK*, 2014, pp. 221-232.

[20] Khaled Dehdouh, "Building OLAP Cubes from Columnar NoSQL Data Warehouses," in

[21] Lucas C. Scabora, Jaqueline Joice Brito, Ricardo Rodrigues Ciferri, and Cristina Dutra Aguiar Ciferri, "Physical Data Warehouse Design on NoSQL Databases - OLAP Query Processing over HBase," in *ICEIS*, 2016, pp. 111-118.

[22] Ladjel Bellatreche, Michel Schneider, Mukesh K. Mohania, and Bharat K. Bhargava, "PartJoin: An Efficient Storage and Query Execution for Data Warehouses.," in *DaWaK*, 2002, pp. 296-306.

[23] Alberto Abello, Jaume Ferrarons, and Oscar Romero, "Building Cubes with MapReduce," in *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*, 2011, pp. 17-24.

[24] Wu-Chun Chung, Hung-Pin Lin, Chen, and al., "JackHare: a framework for SQL to NoSQL translation using MapReduce.," *Autom. Softw. Eng.*, vol. 21, pp. 489-508, 2014.

[25] Spyros Blanas et al., "A comparison of join algorithms for log processing in MaPreduce," in

[26] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen, "CloudETL: scalable dimensional ETL for hive," in *18th International Database Engineering*

[27] Billel Arres, Nadia Kabachi, and Omar Boussaid, "Building OLAP cubes on a Cloud Computing environment with MapReduce," in

[28] Andrew Pavlo et al., "A comparison of approaches to large-scale data analysis," in

[29] Nafees Ur Rehman, Svetlana Mansmann, Andreas Weiler, and Marc H. Scholl, "Building a Data Warehouse for Twitter Stream Exploration," in *ASONAM 2012*, 2012, pp. 1341-1348.