ETL in the Big Data Era

Luis Miguel Monteiro Silva

Instituto Superior Tcnico, Portugal
{luis.m.silva@tecnico.ulisboa.pt}
https://tecnico.ulisboa.pt

Abstract. The amount of data that exists nowadays is growing faster, since the companies produce more and more data, in increasingly higher speeds. This makes it necessary to have tools which are able to cope with this increased volume of data. This paper presents a preliminary study to evaluate ascertain under what conditions we can benefit from the use of Big Datas technologies for an Extract, Transform, Load process (ETL).

Keywords: Big Data, Hadoop, ETL, Mapreduce

1 Introduction

We live in a world where the volume of data is growing in an exponential way [2]. The increase of the number of physical devices connected to the internet, mobile devices in particular, the wide use of social networks and the big necessity of decision support systems, has as consequence the increase of volume of data at high speeds, as well heterogeneity of types of data (videos, text, music, etc.). This volume, variety and speed of data is called Big Data [2]. In comparison with traditional data sets, Big Data generally includes a large amount of data, structured and non-structured [2].

Big Data brings a series of new opportunities regarding the extraction and transformation of data into information and also brings new challenges, like managing and organizing these data sets in an effective way. New computation paradigms were developed to deal with these volumes of data, with MapReduce being the most popular. MapReduce paradigm is a programming model for efficient distributed computation. It provides a parallel programing model and several associated implementations in order to process large amounts of data.

The focus of this paper is framed in the context of Extract, Transform and Load (ETL) processes, that are responsible for transforming data, originated from multiple data sources into useful and meaningful information for the purposes of business analysis. With the increase of the volume of data ETL processes have must process increasingly more data with several formats. These represents a problem to the traditional ETL process, because they may not be able to deal with Big Data and non-structured data in an efficient way. The MapReduce paradigm can make the ETL process faster, allowing the transformations to benefit from its processing model that distributes the data and the processing itself across several nodes from a cluster of machines.

1.1 Objectives

The objective of this work is to verify in which contexts it is more useful to use Big Data technologies in an ETL process. The goal consists in analyze in which circumstances it is suitable to use a centralized environment or a distributed environment for the execution of an ETL process. In order to fulfill this goal, we conducted the following experimental validations:

- We implemented an analytical model example for a given volume of data;
- We design an ETL process to fill the analytical model with data from a data source;
- We implemented the ETL process using MapReduce with the data integration and analysis tool Pentaho Data Integration[1];
- We implemented the same ETL process with the data analysis platform Pig[2];
- We developed the same process using Pentaho Data Integration but without MapReduce;
- We compared the three developed implementations taking into account criteria such as performance, ease of development and features offered.

1.2 Outline

This paper is organized as follows: The Section 2 presents concepts that we consider necessary to the understanding of this document. On Section 3 related works are analyzed. On Section 4 we describe the implementations of the ETL process.On Section 5 we present experiment results. Finally, Section 6 presents the main conclusions of this project.

2 Fundamental Concepts

In the following section we present concepts that we consider necessary to the understanding of this document.

2.1 Big Data

In general, Big Data refers to data sets that cannot be acquired, managed, and processed by traditional software/hardware and Information Technology (IT) tools within a certain time limit [2]. However, there are several definitions for Big Data, among which we can find the following:

- In 2010, in the context of the project Apache Hadoop, Big Data has been defined as "data sets that cannot be captured, managed and processed by ordinary computers within an acceptable time limit" [2];

[1] `http://www.pentaho.com/product/data-integration`
[2] `http://pig.apache.org/`

– Based on the above definition, McKinsey and Company in May 2011, announced Big Data as the next frontier for innovation, competition and productivity, defining Big Data as "data sets that cannot be acquired, stored and managed by classic database software" [2].

The increase in data in the era of Big Data, brings challenges in the acquisition, storage, management and analysis of data. Some of the main challenges of Big Data are: (i) Integration, Aggregation and Data Representation; (ii) Privacy; (iii) Cooperation; (iv) Timeliness.

2.2 Hadoop

Hadoop is a framework that allows you to perform distributed processing of large data sets in a cluster of machines using programming models such as MapReduce. Hadoop is commonly defined as being constituted by MapReduce and the Hadoop Distributed File System (HDFS), a distributed file system that manages the storage of data on machines that makes up the cluster where Hadoop is installed.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.

2.3 MapReduce

MapReduce is a programming model used for processing large data sets. Users specify one map function that processes a key/value pair and generates an intermediate set of key/value pairs and a reduce function that groups together all intermediate values associated with the same intermediate key and returns a smaller set of values [4].

Programs written in this paradigm are automatically parallelized and run on a cluster of computers. MapReduce is responsible for partitioning the input data, scheduling program execution using a set of machines, handling the failures of the machines and manage communication between machines.

Programming Model. The MapReduce programming model is given a set of data as input, and converts it into another data set, where the individual elements are transformed into tuples (key/value pairs). As the name indicates, the user of this programming model expresses the processing into two distinct functions: Map and Reduce. The map function receives a data set as input and produces a set of intermediate key/value pairs. The MapReduce then groups all intermediate values associated with a given I key and passes this key and its values to the reduce function[4]. The reduce function accepts an intermediate I Key and a set of values associated with it, i.e., it receives tuples that have been

generated by the map as input. Then combines these data to form a possibly smaller set of values. Typically, only zero or one output value is produced per reduce invocation of the function. Intermediate values are provided to the reduce function via an iterator allowing handling lists of values that are difficult to keep in memory [4].

2.4 Extract Transform and Load

Extract Transform and Load (ETL) is a process responsible for running the extract, transform and load data from one or more source data systems to a target data system.

Extraction. Extraction is the operation that consists in obtaining data from a source system for later use in a target system. The design and creation of an extraction process is often one of the most time consuming tasks of the ETL. Systems where the source data resides, can be very complex and have little documentation, which ultimately makes it difficult to determine what data needs to be extracted [3].

Transformation. The transformation phase applies a set of rules to transform the extracted data in order to load them into a destination repository. The data transformation is not just a simple mapping of columns and tables to the correct location on the destination repository. It is often necessary to change the data in order to match the target system and corresponding integrity restrictions.

Loading. After performing all the tasks that make up the transformation phase, the data is ready to be loaded to the destination data store.

3 Related Work

In the following section we present some technologies and related works that are related to the objective of this paper.

3.1 Apache Hadoop Project Tools

Apache Pig. Apache Pig is a platform for analyzing large data sets that provides an engine for execution of parallel data flows on Hadoop[5]. The Pig is executed over the Hadoop, it uses the file system provided by Hadoop, HDFS, and MapReduce. This platform includes its own language, Pig Latin, to specify data flow. The Pig Latin language includes operators for many of the traditional data operations (join, sorting, filtering, etc.), but gives also the possibility for users to develop their own functions, in languages like Java, Phython, JavaScript or Ruby for reading, processing and writing of data. Scripts written in Latin Pig by users are compiled by Pig and converted to one or more MapReduce programs which are then executed by Hadoop.

Apache Hive. Apache Hive is a software platform for data warehouse that facilitates query and management off large data sets stored in a distributed way. The Hive has a mechanism that allows structuring data and make queries about them using similar language to SQL called HiveQL. This language also allows MapReduce programmers to use their map and reduce functions when it is inconvenient or inefficient to use HiveQL. The HiveQL language its similar to SQL, which allows users familiar with SQL to query the data. Moreover, this language allows programmers who are familiar with the MapReduce framework to use its reduce or map functions in order to make a type of analysis that may not be supported by the capabilities of HiveQL language. The Hive has two components, HCatalog and WebHCat. HCatalog is a table and a storage management layer for Hadoop that gives users of different data processing tools (including Pig and MapReduce) an easier way to read and write data to the Hive environment.

3.2 ETL Frameworks

CloudETL. CloudETL (Liu et al., 2012) is a framework for ETL using Hadoop to parallelize ETL flows and process data in Hive. The user defines the ETL process through high-level buildings and transformations and do not need to worry about technical details related to MapReduce. The CloudETL framework supports different data warehousing schemas such as star schemas and slowly changing dimensions [7]. This framework can be seen as a layer that sits on top of the Hive and aims to make it easier and faster to create scalable ETL processes to load data into Hive data warehouses. The CloudETL allows ETL developers to easily convert an ETL program in MapReduce programs on Hadoop using high-level instructions, i.e. instructions with a high degree of abstraction, in a Java program and without having to deal with MapReduce details [7].

The CloudETL framework uses Hadoop as ETL execution platform, Hive as a data warehouse system and has the following components: application programming interfaces (APIs) used by the users ETL programs, a set of elements to make changes in data called ETL transformers and a program manager that controls the execution of programs submitted in Hadoop.

ETMLR. ETLMR [6] is a programming framework for the ETL that uses MapReduce to achieve scalability. The ETLMR framework has native support for specific schemes for data warehousing, such as star schemas, snowflakes, and slowly changing dimensions. The fact that it supports these specific schemes makes it possible for users to build ETL streams based on MapReduce using few lines of code [6].

This framework provides specific high-level ETL buildings, on top of facts and dimensions tables, both in star schemas and snowflakes schemes. A user can implement parallel ETL programs using these buildings without having to be aware of the details of the parallel execution of ETL processes. This feature facilitates the programming work since the user only has to make a configuration

file with a few lines of code to declare objects of facts and dimensions and the necessary transformation functions [6]. The ETLMR uses a programming framework based on Python to make programming easier, called pygrametl [8].

3.3 Pentaho Data Integration

The Pentaho Data Integration (PDI) is an ETL tool that enables users to access, prepare, analyze and extract useful information from both traditional data and Big Data. The Pentaho aims to simplify the management of large volumes of data entering companies, at increasingly higher speeds and varieties, regardless of the type of data and number of data sources. The PDI has a graphical interface that offers users an alternative to having to code programs in SQL and MapReduce to process the data.

4 ETL Process Implementations

In the following section we present the three implementations of the ETL process.

4.1 Pentaho Data Integration (PDI) using conventional ETL operators.

The calculation of the analytical indicators is organized into two jobs, one for direct indicators and one for calculated indicators. Both jobs have a set of transformations that calculate one or more indicators. In this first implementation of the ETL process indicators are obtained by running SQL queries. In this sense, the indicators that share the same "where" clause are calculated in the same transformation.

The transformation begins to run the operator responsible for executing the query that calculates the indicator. In order to store the analysis window in the respective fact table it is run a set of searches to the analysis dimensions to obtain the identifier of each dimension. This implementation of the process will get the input data in a MySQL database and stores the results in files on the local file system of the computer.

4.2 Pentaho Data Integration (PDI) using MapReduce operators

As MapReduce is used to perform the calculation of the indicators, the source data have to be transferred in the form of files to the Hadoop distributed file systems, the HDFS.

In this implementation, when using MapReduce, indicators that meet the same selection criteria are calculated in the same MapReduce program. The two implementations of the ETL process that uses PDI, although they are implemented in the PDI, are quite different. As the two implementations have used the same program (PDI), it could be thought that converting one to another would be a simple process, but it is not the case. Because one of the implementations

belong to a centralized environment and the other to a distributed environment, the PDI components that were used in both cases were the adequate ones for the two types of environment. Therefore, to convert an implementation in to another it involves the redo of the process to change the operators used.

4.3 MapReduce using Pig scripting language

The calculation of the indicators is divided into two scripts, one for direct indicators and another for the calculated indicators. As in the previous implementation, the input data must be transferred to the HDFS so that they can be used by Pig.

After loading the information, it starts the indicators calculation. To calculate the indicators, it is performed a GROUP BY by the analysis dimensions and is then applied the desired operation, whether being a summation, a counting, maximum determination, etc. In addition, some indicators require that a particular data filtering is made. The example of Number of diverted flights illustrates an example of calculating an indicator.

```
Desviados1 = FILTER dados_voos_clean BY Diverted==1;
Desviados2 = group Desviados1 by (Year, Month, DayofMonth,
UniqueCarrier, Dest, Origin);
(10) Desviados3 = foreach Desviados2 generate flatten($0),
COUNT($1) as numvoosdesviados;
```

(Example Number of diverted flights)

Unlike the implementation of the ETL process using PDI with MapReduce, it was not necessary to divide the processing of data by two MapReduce phases, since the Pig is in charge to make this division. Thus the user only has to worry about the operations that are applied to the data, leaving the Pig to the management of the operations performed in the mapper and the reducer.

5 Validation

In this section we validate the hypothesis behind the present work, the verification of in which contexts it is more useful to use Big Data technologies in an ETL process.

5.1 Infrastructure Used

As support infrastructure to the completion of the work, the Hadoop cluster belonging to the Rede de Novas Licenciaturas (RNL) of the Instituto Superior Tcnico was used. The cluster is running in the laboratories workstations using Debian 7 64bit. Depending on the time of day and activity of laboratories, there are available between 40-360 cores, with about 2GB of RAM per core. Hadoop is available in 90 nodes of the cluster. Altogether there is a possible maximum of 360 slots distributed with the same weight for Map and Reduce. Cluster computers have the following characteristics:

- Group I: 60 computers
  - Intel Core i5 760 2.80ghz (Quad-core) Processor
  - 8GB (8192MB) RAM
  - Debian 7.0/Windows 7 Operative System
  - 500 GB HDD
- Group II: 30 computers
  - IIntel(R) Core(TM) i5-3570 CPU @ 3.40GHz (Quad-core) processor
  - 8GB (8192MB) RAM
  - Debian 7.0/Windows 7 Operative System

Hadoop and Pig versions used were respectively Hadoop Version 2.2.0 and Pig Version 0.14.0. For tests with Pentaho without MapReduce was used a computer from Group I. and Pentaho Data Integration version 5.4.0.1-130.

5.2 Description of the data used

For this paper we used public data with details on the departure and arrival of commercial flights performed in the United States. As mentioned in Chapter 4, the data is divided into three files, information on flights, information on airlines and finally information on airports.

The test data set contains approximately 120 million records in total, with a occupied space of 1.6 gigabytes when compressed and 12 gigabytes when not compressed.

5.3 Experiments

The tests conducted with the three implementations of the ETL process were divided into two parts. The difference between the two sets of experiments is the number of used process implementations and the use of a data cleaning component aimed to replace data not available (NA) for a null value. In each test were calculated a set of direct indicators and a set of calculated indicators.

ETL process without the data cleansing component. In a first experiment, we measured the execution time of ETL processes implemented involving processing, grouping and calculation of the indicators, and discarding the time of storing the input data in the database and the HDFS. In order to test the scalability of the three processes the execution time was measured for different amounts of data. The results for the direct indicators are shown in Figure 1, were is illustrated the runtime obtained for different volumes of information.

Analyzing the results, we can see that the two processes that benefit from the distributed environment (Pig and PDI with MapReduce) have better results as it progresses scaling the data. For small data volumes (5 and 10 million records) the two processes that use the PDI shows better results than the Pig. We conclude that, for data volumes of this order, the use of a non-distributed environment has great advantages over a centralized environment, since the process that uses

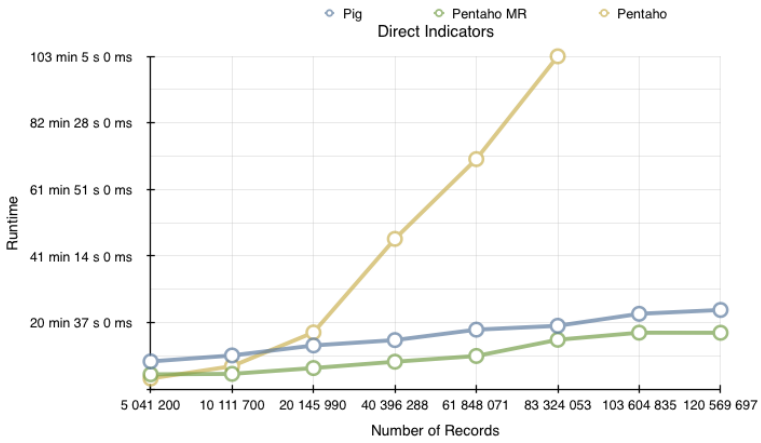

Fig. 1. First Experience Direct Indicators

the PDI without MapReduce can have good runtimes. The breaking point occurs when the volume of data reaches the home of 20 million records. From here, as is easily noticeable, the use of a distributed environment exhibits the best results.

ETL process with the data cleaning component. In a second experiment, it was considered only the two process implementations using MapReduce and was introduced in these implementations a data cleansing component that replaces not available data (NA) by a null value. The process that uses the PDI without MapReduce was dropped in this second phase, since it no longer made sense to continue to test this implementation process after analyzing the results of the first tests.

By analyzing the results present in the Figure 2, we can conclude that the Pig had better times than the PDI with MapReduce. PDI used more mappers than the Pig but only one reducer, and it made the reducer stay overwhelmed with all the information coming out of the mappers. On the other hand, the Pig did a better balancing with the mappers and reducers than PDI.

6 Conclusion

This paper presented an experimental study and a set of tests on a ETL process in which different technologies have been addressed with the aim of assessing under what conditions is more advantageous use of Big Data technologies for data processing. The specific tasks considered in this master thesis were:

– Develop an ETL flow on the analytical model using MapReduce with the data integration and analysis tool Pentaho Data Integration;

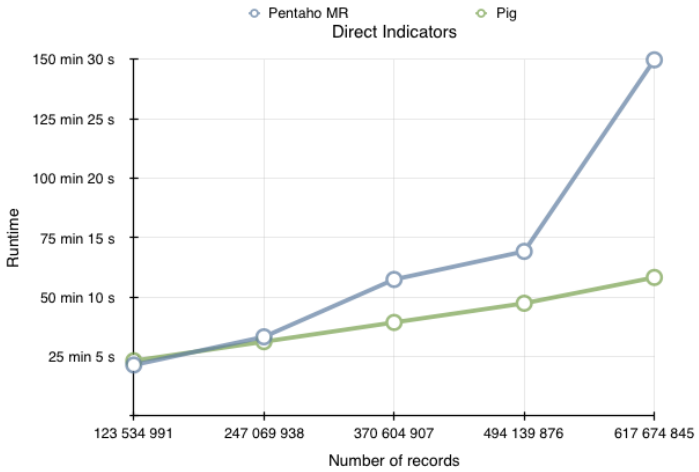

Fig. 2. Second Experience Direct Indicators

- Develop the same ETL flow with the data analysis platform Pig;
- Develop the same flow using Pentaho Data Integration but without MapReduce.

The aim was to realize from what volume of data could take better advantage of the use of technologies that use the MapReduce as data processing engine. Having regard to the type and volume of data that has been used for this thesis there is in fact a break point from which it is impracticable not using Big Data technologies since the processing time begins to be very high.

References

1. AGRAWAL, D., BERNSTEIN, P., BERTINO, E., DAVIDSON, S. and DAYAL, U. (2011). Challenges and Opportunities with Big Data. Cyber Center Technical Reports, Purdue University, Purdue e-Pubs.
2. CHEN, M., MAO, S. and LIU, Y. (2014). Big Data: A Survey. Mobile Netw Appl
3. KIMBALL, R. and CASERTA, J. (2004). The Data Warehouse ETL Toolkit. Wiley Publishing, Inc.
4. DEAN, J. and GHEMAWAT, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Symposium on Operating System Design and Implementation.
5. GATES, A. (2011). Programming Pig. OReilly Media, Inc.
6. LIU, X., THOMSEN, C. and PEDERSEN, T.B. (2011). ETLMR: A Highly Scalable Dimensional ETL Framework based on MapReduce. Proceedings of 13th International Conference on Data Wa-rehousing and Knowledge, Toulouse, France.
7. LIU, X., THOMSEN, C. and PEDERSEN, T.B. (2012). CloudETL: Scalable Dimensional ETL for Hive. Department of Computer Science, Aalborg University, 1DB Technical Report.
8. THOMSEN, C. and PEDERSEN, T.B. (2009). pygrametl: A Powerful Programming Framework for Extract-Transform-Load Programmers. In Proc. of DOLAP.