doi: 10.17234/INFUTURE.2015.27 Review

Big Data in Data Warehouses

Tomislav Šubić, Patrizia Poščić, Danijela Jakšić
Department of Informatics, University of Rijeka
Radmile Matejčić 2, Rijeka, Croatia
tsubic@student.uniri.hr, patrizia@inf.uniri.hr, dsubotic@inf.uniri.hr

Summary

The amount of data grows every day. Along with that, new challenges and needs appear. We wonder how to collect and process this enormous amount of data and to get the information we need for business management. Tools and technologies for that task should be scalable, distributed and with high bandwidth so we can quickly get to the information we want. As a solution impose NoSQL databases. But databases without a high quality system for processing and analysis of data are not very helpful. Some solutions that have started to address these problems appeared, but there is still room for development. Some of these solutions are Apache Hadoop and Hive, which are used every day in big companies like Facebook and Google. This paper provides insight into existing technologies that implement such solutions, which have their advantages and disadvantages, and we will try to predict how this story will develop in the future.

Keywords: big data, data warehouse, NoSQL, Hadoop, Hive

Introduction

The information revolution that is now taking place leaves a great impact on all types of businesses and business organizations. With all the tools available it is becoming easier to collect data of the operations of an organization, the user needs etc. and based on the analysis of this data to make better business decisions. The first step was made in 1970 by E. Cobb, when he introduced the relational database model, which remains the standard to this day for the storage and processing of data in information systems. However, 40 years later, customer needs are changing, and in spite of all the adjustments and optimizations of the relational model there are new technologies that are held behind by this model. For better analysis it is necessary to collect as many as possible high quality data from all aspects of the business. A new trend called Big Data appears; this represents the large amounts of data that usually come from sources such as social networks, publicly available information from various agencies, subscription information, news, documents, e-mails etc. (and include structured or unstructured text), as well as digital devices and sensors, location data from smartphones, weather etc. Most companies are not familiar with collecting and processing large amounts of data from these sources. So much of this data remains lost and unstructured, and thus aren't included into reports and business insights which help in making better business decisions. We can say that most business organisations do not realize the true value of this data. The companies that are trying to collect enormous amounts of data have problems, because they do not fit into the relational model. Unstructured data is the data that has no predefined data model, and the data which does not fit into existing models. Such data is accumulating very fast; these are videos, photos and generally all possible output files of various programs. Furthermore, new technologies such as Web 2.0 applications used by large numbers of users depend on the storage and processing of large amounts of data. Such a large database needs to be distributed, scalable and fast. All this presents great challenges for the relational model and the tools it implements, such as SQL (Stonebraker et al., 2007). Because of these reasons, a big number of organizations such as Facebook, Amazon, Yahoo and Google are using non relational solutions (for specific tasks); popularly known as NoSQL databases. This term first occurred in 1998, to become real competition to the relational model around 2009. Because of the open model, a lot of different companies develop their own non relational solutions, and most of them are available to everyone because they are open source. Of course, apart from raw data storage, there is a more important part, which takes care of the data processing and analysis. Today, we aim to business intelligence systems and their major component is the data warehouse. The technologies that have been developed to solve this problem are sharing some common characteristics associated with the scalability, flexibility and high availability. MapReduce technology together with the so-called Hadoop Distributed File System (HDFS) and the HBase database, as part of the Apache Hadoop project is a new and innovative approach to the processing and analysis of such unstructured data. In this paper we deal with unstructured data, the possibilities of Apache Hadoop technology and the integration of non relational databases into a data warehouse. We will clarify some main concepts, present some existing solutions, and try to show whether Hadoop is a suitable replacement for existing data warehouses. In the second chapter we will speak generally about data warehouses and technologies that are used, in the third chapter we look at non relational databases and give an insight into the different types, further in the fourth chapter we deal with the current Big Data technologies, while in the fifth chapter we discuss data warehouses that implement these technologies. Finally, the sixth chapter provides a description and overview of the Hive data warehouse that uses the described technology.

Data Warehousing

The main purpose of SQL and relational databases is to store and access data from a variety of transactions that occur in everyday business such as orders, invoices, salaries etc., execute queries on this data and generate reports. Further development of business systems, development of strategic planning and management which occurred because of the globalization of a competitive market, but also the development of hardware and software technology greatly relied on the relational data model and the analytical processing of such data. In recent years we notice the intensive development of Web 2.0 applications. These are applications such as massive online stores, social networks etc., which must process large amounts of data to millions of users simultaneously in real time. Limitations occur when handling large amounts of unstructured data in traditional data warehouses. Structured data requires a specific scheme to match the relational model. This of course is not the case in unstructured data that does not have a specific scheme. Although SQL supports unstructured text search by keywords, it is limited to just text (Jain et al., 2007). All that led to the development of a completely new model of data warehouses based on the new data models. These models are called NoSQL. The main idea of systems, which are trying to take advantage of the Hadoop technology, is the integration and construction of interfaces with existing data warehouses, or possibly the construction of new types of data warehouses.

Non relational database models

Over the years the term NoSQL turned into something that represents non relational, distributed, horizontally scalable databases. In addition, the most widely used ones are open source. Their main task is to ensure quick retrieval of existing data, as well as storing new data. Since OLAP and other business intelligence systems are critical to many business organisations, the question is how do NoSQL databases fit in such systems? There are several major models that that meet the above criteria and have become the standard (Grolinger et al., 2013):

- column oriented
- document oriented
- key-value database
- graph database.

In this paper, we primarily deal with *column* and *document oriented* databases because they are commonly used in the fields that we discuss in this paper.

Column oriented databases

Unlike the relational model, which stores data in rows (tuples), column oriented databases store data by column, which means that each row has only one data type. Thus, the data from the database is consistently written to the computer storage, without the fragmentation which occurs in implementation by rows. Other benefits include easier retrieval of data; all data fetching usually occurs within one table (no need to merge data from multiple tables), that goes for iterative processing too. Some examples of such databases are Apache Cassandra, Google BigTable and Hbase. Often is the wrong terminology used to describe NoSQL column oriented non relational databases, namely there is a relational model that distributes data by columns, and is called column store.

Document oriented databases

Databases which simply store data in the form of documents are considered to be the main form of non relational databases. They support data storage in documents of any format. The data itself within the document is structured using XML, JSON or something similar. Today, due to the simple notation, most commonly is used JSON. Popular databases that implement this model are MongoDB, CourchAB and OrientDB.
We will discuss technologies based on non relational databases. Hadoop and Hive both support a variety of NoSql databases.

Big data technologies

Data Vault and Anchor modelling

We cannot talk about data warehouses with a large amount of data without mentioning these two models. Data Vault is a hybrid model that combines the best characteristics of the third normal form and the star schema developed specifically for EDW-a (Enterprise Data Warehousing). Its main concepts are so-called Hub, Link and Satellite entities. All data is stored in the Satellite entities, the keys are stored in the Hubs, while the mutual relation keys are specified in Links (Linstedt, 2011). Anchor modelling is a database modelling technique that is based around the claim that the data warehouse environment is constantly changing and that big changes outside the model should lead to small changes in the model. The aim of using this type of modelling is to achieve a highly decompressed implementation that can deal with the growth of data warehouses. These are highly normalized (6NF) models which are characterized by the ability to store changes in the business environment (Krneta et al., 2014). Still, these are relational models and in this paper we discuss non relational solutions, we will refer to them much.

MapReduce

Developed by Google as a "programming model with the implementation for processing and generating large data sets (Dean and Ghemawat). It is a method which greatly simplifies the distribution of tasks across multiple nodes. Each node processes the data stored on that node (Kumar et al.). It consists of two parts, the first function that is responsible for the so-called mapping of key-value coming from the input data in some other suitable key-value pairs:

$$Map((K1,V1) \Rightarrow (K2,V2))$$

The second, so-called reduce function reduces the value so that the values are first sorted and grouped according to a given key; it gives the final output:

$$Reduce((K2, list(V2)) \Rightarrow list(K3, V3))$$

MapReduce is popular for many reasons:

- automatic parallelization and distribution
- fault tolerant
- easy to use and develop
- programs are written in Java.

Unfortunately, the MapReduce programming framework is Google's proprietary software. However, there are equally good open source alternatives that work on the same principle. The best known and most widely used is Apache Hadoop. This implementation relies on HDFS in processing and storing large amounts of data. Map and reduce platforms such as this play a key role in data warehouses and business intelligence systems, because they enable us to do complex data processing such as aggregation and filtering very effectively.

Comparison of Hadoop and traditional ETL tools

Data analysis built on top of Hadoop in some cases, can get instant access to a large amount of high quality data without the need for long ETL processes (Duda, 2012). Hadoop can implement ETL processes by writing custom programs, which sometimes is not really effective and cannot always replace the traditional ETL tools from the existing data warehouses. ETL tools contain a lot of built-in capabilities suitable for cleaning, coordination and transformation of data. Such tasks need to be manually written in Hadoop. Their disadvantage is they are not particularly scalable and encounter problems when attempting to process several hundred gigabytes of data. It is recommended to use ETL tools for small amounts of data, because it saves data warehouse development time, while with the large amount of data it is better to develop custom programs in Hadoop. There are also hybrid solutions such as IBM InfoSphere DataStage that uses Hadoop for processing unstructured data and then forwards them to ETL tools for further processing (IBM, 2014).

Hadoop

Hadoop is an open source programming framework which supports processing of large amounts of data in a distributed computing environment (Kumar et al.). It is part of the Apache Project and is funded by the Apache Software Foundation. Hadoop includes an interface that allows parallel storage and processing of data. It runs MapReduce programs in a parallel environment of thousands of nodes. It is capable of collecting data from various sources, and can be further

customized to fit the type of data. It is worth noting that Hadoop is a kind of MAD (Magnetic, Agile, Deep) type system (Cohen et al., 2009).
These types of systems are distinguished by their characteristics:

- magnetic – attracts all types of data from various data sources
- agile – able to adapt processing and analysis to possible changes in the data structure
- deep – capable of high scalability in-depth analysis of large amounts of data.

Because of these characteristics as its biggest advantage stands out the ability of in-depth analysis of unstructured data collected from various sources much better and faster than analytics tools based on SQL (Cuzzocrea et al., 2011).

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a scalable distributed file system that provides instant access to application data. Computer clusters with HDFS work on the master and worker nodes principle (Date et al., 1986). The master node is responsible for management of the entire file system, metadata of all files and directories in the file system and the locations of all the blocks of data stored in the worker nodes. Worker nodes store and return information when requested by the main node.

Data Warehouses with non relational databases

One of the definitions of business intelligence gave E. Turban et al., "An umbrella term that encompasses tools, architectures, databases, data warehouses, performance management, methodologies, and so forth, all of which are integrated into a unified software suite” (Turban et al., 2010). During the execution of ETL processes the data is loaded from ERP, CRM and other systems, as well from the database, and if necessary, from different spreadsheets or other formats. After the data is loaded, we can do OLAP analysis, use tools for data mining or just do custom reports (Duda, 2012). There are several models by which data warehouses are built; ROLAP, MOLAP and their hybrid HOLAP. However, current systems and data warehouses do not have all the necessary tools and information necessary to make business decisions in today's dynamic and complex economic environment. Large amounts of data (Big Data) simply cannot be collected and processed by traditional systems and existing OLAP servers that are based on the relational model. Because of their limitations and new needs, solutions that are based on the non relational model are developed.
Map and reduce platforms that we described earlier together with some of the open NoSQL databases that support Hadoop (such as HBase) can be an alternative or supplement for traditional data warehouses. An example of such a solution is Apache Hive. Although is supports an interface similar to SQL state-

ments, queries are translated into map-reduce jobs, which are then performed on the Hadoop platform (Thusoo et al., 2010).

Hadoop as a data warehouse

Hadoop can be considered as the next step in the development of data warehouses, with a particular focus on the ETL phase. In Hadoop there is no referential integrity, which results in better performance. In comparison, the Data Vault model supports full referential integrity through the entire model, which of course has its advantages (Linstedt, 2011). Partitioning supports the separation of files, even to different computer nodes. We load raw data into Hadoop, data without pre-treatment, which means that they do not have to go through special ETL processes. At first glance a similar idea has a one-layer architecture of traditional data warehouses, but it does not support any form of ETL and in practice it is rarely used. The data is copied to the Hadoop platform, and adaptation (extract and load) rules should be written in the source code of Hadoop implementations. We emphasize that Hadoop is not an ETL tool but a platform that supports the ETL processes (Thusoo et al., 2010). This approach has many advantages such as fast loading of data, distributed processing, automated compression (in some implementations) and many others. Also, one of the advantages is freer access to the data storage moral in comparison to relational. We must adhere to certain rules and we have to use some consistent model such as the XML to make it easier to handle data.

Hadoop and data warehouse architecture

The hadoop platform is used together with some existing data warehousing systems. Processes in the Hadoop platform can use the data warehouse as a data source. MapReduce can issue SQL tasks to the data warehouse. We take advantage of the benefits of both technologies. MapReduce task is just another program, a data warehouse acts as a database (Das and Mohapatro, 2014).

Hive

Hadoop gives us a lot of new features but also has some drawbacks. MapReduce programming interface is low-level and tends to become very abstract and requires developers write custom programs that are usually difficult to maintain and reuse. When it comes to processing large amounts of data, the use of Hadoop technologies can reduce the processing time from one day to a few hours. But the use of Hadoop is not easy for end users. For example, it is necessary to write MapReduce programs for simple inquiries such as getting clean totals or averages of some data. Hadoop is missing some interface that works in a similar way as SQL. For these reasons Hive was developed. Hive is an open source solution built on top of the Hadoop platform (Thusoo et al., 2010). The goal was to develop an interface that can interact with unstructured data in Hadoop, and that brings some familiar concepts of tables, rows and columns and part of the functions and methods that we know from SQL, of course adhering to Hadoops rules with the intention of avoiding noticeable restrictions. It was developed by Facebook where it contained over 700TB of stored data back in 2010 and is still used to provide reports and analytics.

Structure and data mode

Hive structures data in the well-known concepts such as tables, columns, and rows. It supports all major data types; integer, float, doubles and strings, but also more complex types such as folders, lists, and structures. More complex types of support nesting and thus can build more complex types. Also, Hive gives users the ability to upgrade the system to their custom data types (Thusoo et al., 2010). If the data is prepared by another program, or from the relational model, Hive has the ability and tools to incorporate the data to the table, without data transformation, which is a great advantage in case of large data sets.

Hive query language

The language with which are queries built, HiveQL, is very similar to SQL, so users who are familiar with SQL can quickly adapt. Traditional features of SQL as well as various types of joint, carthesian products, grouping, aggregation, creating tables, SELECT command and other useful functions that can be performed on the primitive and complex types make this language similar to SQL. This allows users who are familiar with SQL to a Hive to immediately open a command prompt and start writing queries. Some useful functions for returning metadata tables also present which saves time writing programs or queries. With a single command the data structure can be found out and which are used types of data without further testing various queries or similar methods. There are some limitations with respect to SQL, such as the inability to add new elements to the table or data partition, but if you want to add data, we have to rewrite the entire table or partition. However, this and similar restrictions have not created problems in using (Thusoo et al., 2010). As the data in such systems is retrieved periodically, for example daily or every hour, the data is easily loaded into a new table or data partition. Hive contains plug-ins for analysis using the map-reduce programs which the users write, in the programming language that suits them best. This allows advanced users build much more complex algorithms.

Hive architecture and components

Hive is assembled from a few main components. Megastore is a component that records the system log, and metadata about tables, partitions etc. Driver is a component that takes care of the lifetime of the HiveQL tasks while processing. Query compiler is a component that translates HiveQL inquiries into tasks and subroutines that are suitable for running in Hive. Execution engine is a component that runs the tasks that the compiler gives in the correct order. It cooperates with the lower layers of the architecture. HiveServer is a component which allows you to interact with other applications. Client components such as CLI, web interface and JDBC/ODBC drivers. HiveQL queries are entered through one of the user interfaces like CLI or web interface. The Driver then sends the query to the compiler to check the syntax and semantic accuracy using the same metadata stored the Metastore. The compiler generates a logical sequence of tasks and then optimizes with a simple optimizer based on certain rules. Finally, generate the optimized sequence of tasks in the form of a map-reduce routine. Execution engine running tasks using Hadoop.

Conclusion

If we want to have a reliable and precise analysis in our data warehouse, or any other system of business intelligence, we need to collect and analyze all available data from the organization itself, but also from its environment, and other data sources. Nowadays, more and more of this data is unstructured and does not fit into the relational model. We are no longer limited to collect certain types of data structures for decision making. In this paper, we explained some of the concepts and technologies that are closely related to the storage, processing and analysis of large amounts of unstructured data. We explained the differences between relational and non relational models. We reviewed the current technologies that use non relational solutions in processing and analyzing large amounts of data. Also, we pointed out some of the flaws and limitations of the relational model. We examined the interface and tools that can come with Hadoop and its integration into the data warehouse. Such a data warehouse can store and process large amounts of data and thereby uses Hadoops MapReduce technology for parallel data processing. One of these is the Hive data warehouse. Although it is still in active development, Facebook uses it daily for processing and analysis of large amounts of sensitive data. We mentioned the Data Vault and Anchor modelling methods that are designed just for the purpose of storing and processing large amounts of data in data warehouses. These models may have come too late to the market, because most of the big companies like Google, Amazon and Facebook have already switched to non relational solutions. Because of the nature of the open source movement and openness of these non relational solutions, they will be the ones to lead the further development of technology in this domain, and that is a big advantage. It is important to note that non relational solutions are not always suitable, and that for certain cases it is still better to use a relational model. We are now at the stage when the non relational databases and technologies that implement them have become a great competition to the relational model and existing, traditional technologies. The development on the integration of Hadoop in existing and new data warehouses is currently active, but in the future we hope to see Hadoop integration with some business intelligence systems, which could eventually lead to some competitive open source solutions.

References

Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., and Welton, C. (2009). MAD skills: new analysis practices for big data. Proc. VLDB Endow. *2*, 1481–1492.

Cuzzocrea, A., Song, I.-Y., and Davis, K.C. (2011). Analytics over large-scale multidimensional data: the big data revolution! In Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP, (ACM), pp. 101–104.

Das, T.K., and Mohapatro, A. (2014). A Study on Big Data Integration with Data Warehouse. Int. J. Comput. Trends Technol. IJCTT–volume *9*.

Date, C.J., Date, C.J., and Date, C.J. (1986). An introduction to database systems (Addison-wesley Reading, Mass.).

Dean, E., and Ghemawat, S. MapRednce: Simplified Data Processing on Large Clusters.

Duda, J. (2012). Business intelligence and NoSQL databases. Inf Syst Manag *1*, 25–37.

Grolinger, K., Higashino, W.A., Tiwari, A., and Capretz, M.A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. J. Cloud Comput. Adv. Syst. Appl. *2*, 22.

IBM (2014). Bolstering data confidence with comprehensive information integration. (IBM Corporation),.

Jain, A., Doan, A., and Gravano, L. (2007). SQL queries over unstructured text databases. In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, (IEEE), pp. 1255–1257.

Krneta, D., Jovanovic, V., and Marjanovic, Z. (2014). A direct approach to physical Data Vault design. Comput. Sci. Inf. Syst. *11*, 569–599.

Kumar, R., Parashar, B.B., Gupta, S., Sharma, Y., and Gupta, N. Apache Hadoop, NoSQL and NewSQL Solutions of Big Data. Int. J. Adv. Found. Res. Sci. Eng. IJAFRSE *1*, 28–36.

Linstedt, D. (2011). Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault (Lexington, KY: CreateSpace Independent Publishing Platform).

Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N., and Helland, P. (2007). The end of an architectural era:(it's time for a complete rewrite). In Proceedings of the 33rd International Conference on Very Large Data Bases, (VLDB Endowment), pp. 1150–1160.

Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., and Murthy, R. (2010). Hive-a petabyte scale data warehouse using hadoop. In Data Engineering (ICDE), 2010 IEEE 26th International Conference on, (IEEE), pp. 996–1005.

Turban, E., Sharda, R., Delen, D., and King, D. (2010). Business Intelligence (Boston: Prentice Hall).