Foundation of Data Science Coursework2 Report

Hang Su 30005019
University of Southampton

ABSTRACT

This report explained five questions about non-relational database and parallel and distributed computation such as application of Hadoop. For database, this article compares the difference between relational and non-relational database and takes MySQL and MongoDB as examples to clarify these differences. Then it focus on NoSQL, trying to figure out advantages and disadvantages of some kinds of NoSQL architectures, also relates this to the data used in task 1. For distributed and paralleled computation, this report compares it with traditional centralised solution in costs and benefits of industries aspects. Meanwhile, it gives some scenarios in which Hadoop can increase the effectiveness of productivity. Specifically, this report gives a general description of the MapReduce process which is related to the task 2 in this coursework.

CCS CONCEPTS

• **Information systems** → Database design and models; Graph-based database models; • **Computing methodologies** → **Distributed computing methodologies**; **Parallel computing methodologies**.

KEYWORDS

Relational database, NoSQL, Hadoop, paralleled and distributed computation, centralised computation, data science

1 INTRODUCTION

In recent years, with the expansion of the volume of data, traditional data storage and data process technologies cannot fulfill the requirement of rapid development. About data store, many of the new systems are referred to as "NoSQL". As for data processing, the parallel and distributed computation or systems are popular these days. In this report, it tries to give the significance of using these two new technologies through comparing them with traditional relational database and centralised computation respectively, it also gives some example scenarios of applying them.

2 DIFFERENCE BETWEEN RELATIONAL AND NON-RELATIONAL DATABASE

Relational databases are traditional database architecture, they have been used for several years, and they have been used among all kinds of industries. Meanwhile, relational databases are suitable for industries who have huge production scale and standard process like financial industry, banking, etc. "Relational databases are designed to run on a single machine, so for this we need a large machine to scale."[16]. Thus for a company who has a huge volume of data to store, it is necessary to arrange a machine with large storage capacity. It is a big cost for any company. In general, relational databases have standard structured tables, users can do complex queries through multiplying several tables, however, this leads to a result that it is difficult to expand the structure of the tables. Meanwhile, the SQL environment follows the rule ACID(Atomicity, Consistency, Isolation and Durability), which will cause some troubles in NoSQL conversely[16].

Talking about NoSQL, "it can handle unstructured data such as documents, e-mail, multimedia and social media efficiently"[7]. This means that unlike the relational database, the data needed to be stored in the database must have features deigned previously. What's more, through comparing the experience of using DynamoDB, MongoDB and MySQL, it shows a clear difference that NoSQL database used to store very simple data type like JSON, a kind of key-value pairs data structure". A key feature of NoSQL is 'shared nothing' horizontal scaling - replicating and partitioning data over many servers."[4]. However, when using relational database like MySQL, there are high coupling between data tables, users can search data using operations like UNION and JOIN. Besides, there exists some advanced function like VIEW, ASSERTION and TRIGGER in SQL, which non-relational databases do not have. Furthermore, the data structure suitable for NoSQL is totally different, according to an article belongs to Vicknair et al., the data suitable for NoSQL system having tables with lots of columns, but some of them may be used only by a few rows, and there are large quantities of data have many-to-many relationships, but the largest difference is that NoSQL data management system needs to change it's data scheme frequently[18]. About rule, NoSQL databases obey BASE(Basically, Available, Soft state and Eventual consistency), which is the reverse of ACID, and the main focus of BASE is permanent availability[16].

It is also worth to mention that there have a number of drawbacks compared with the traditional relational database. For example, NoSQL databases are time-consuming for complex operations though they have high speed for simple tasks. The other drawback is that they lack native support for consistency[12].

The most important difference between relational and non-relational database is the CAP theorem[1]:

(1) Consistency: consistency in a database means that whenever data is written, the latest version of the data will be provided to whoever reads it.
(2) Availability: this in database means users will always get responses from the database whenever they query it for information.
(3) Partition tolerance: when part of the database is inaccessible, it still can be read from and written to.

Only NoSQL database can be divided into a number of databases on the basis of CAP theorem.

However, the gap between relational and non-relational databases is becoming more and more vague. The meaning of the term NoSQL is widely acknowledged as "Not Only SQL" rather than "NO SQL" [7]. At the same time, some RDBMS(Relational Data Base Management System) start to support storing data like XML files and images. There are also some functions similar to SQL in pipeline stages and pipeline operators when doing task 1. Finally, relational databases and non-relational databases are designed for different type of data, both of them can play well in certain industry area and can improve the effectiveness of development.

3 PROS AND CONS OF USING DIFFERENT NOSQL ARCHITECTURE

In the article called **A Comparative Study: MongoDB vs. MySQL** belongs to Cornelia et al.[7], it describes four architectures used by NoSQL systems nowadays, they are as follows:

(1) Key-Value stores - Such as Riak and Voldemort. This is a kind of simple data store similar to the popular memcached distributed in-memory cache. Users only can access the data by a single key-value index, but at the same time, it provides some additional functions, like replication, versioning, locking and sorting. And key-value stores have relatively higher query speed and are often used as a high-speed in-memory cache for applications[8]. The most severe drawback is that it does not provide secondary indices or keys[4] and there is no way to query based on the content of the value[14].

(2) Document stores - Such as MongoDB and CouchDB. Document-oriented databases do not have predefined data schema since they are flexible in the type of content. The data model of them consists of objects with a whole selection of attributes. Besides, this kind of system support secondary indexes and multiple types of documents per database[4]. On the other hand, taking MongoDB as an example, the data stored in any collection can be easily dropped by the programmer, which raises the probability of accidents.

(3) Column-oriented - This architecture using tables as the data model, but does not support table association. "Although column-oriented database has not subverted the traditional stored by row, but in architecture with data compression, shared-nothing, massively parallel processing, column-oriented databases can maintain high-performance of data analysis and business intelligence processing."[8]. However, because the row and column design are critical[14], thus before storing the data, you have to split the record into columns and then compress it into the existing table structure, which will consume large quantities of labour.

(4) Graph-oriented - such as Neo4j. In this store, data is stored in a series of nodes, relationships and properties[14]. When querying large graph or connected data, studies have shown these technologies have much better performance than traditional relational databases. "Graphs are often represented graphically and are powerful for visualising and analysing data." In general, graphs are easy understanding for human beings. Thanks to the data structures and query tools they provided have closely relationship to our real world, graph-oriented NoSQL often can provide excellent performance[3]. But graph-oriented databases have poor scalability when graphs do not fit into RAM. Moreover, most of them need to use specialised query languages[14].

After checking several pieces of email data, it is clear that some of them do not have specific fields, especially in "headers" field, and it contains lots of unstructured data like the main contents of the emails. Moreover, in this task, we need to search data according to the keywords in the content rather than the key, which can not be done in simple key-value NoSQL databases. Thus it is suitable to use the document-oriented database in these email data. Besides, as for the key-value model, data value must be indexed by the key and store does not know anything about them. And programmer only allowed to access the data by the key, which limits the advanced query functions that can be done.

4 WHY CHOOSING NON-RELATIONAL DATABASE AND MONGODB IN TASK1

In this coursework, each item of the email data formed like the following:

```
{'_id':
ObjectId('4f16fc98d1e2d3237100435a'),
'body': '...',
'filename': '76.',
'headers': {'1': '7bit',
              'Content-Type': 'text/plain;',
              'Date': '...',
              'From': 'eric.bass@enron.com',
              'Message-ID': '...',
              'Mime-Version': '1.0',
              'Subject': 'check it out',
              'To': '...'
              'X-FileName': 'ebass.nsf',
              'X-Folder': 'Eric_Bass_Dec2000',
              'X-From': 'Eric Bass',
              'X-Origin': 'Bass-E',
              'X-To': 'Shanna',
              'X-bcc': '',
              'X-cc': ''},
'mailbox': 'bass-e',
'new_date': datetime.datetime(...),
'subFolder': 'sent'}
```

As we can see, each email data is a JSON, a kind of semi-structured data, thus these data are not suitable for traditional relational databases. If we want to use relational database, we need to design every field in 'headers' as a column, which will cost a lot of time and this will change the structure of the raw data. And JSON's format is key-value pairs, thus key-value store and document-oriented database are better choices. Furthermore, the document stores support querying collections based on multiple attribute value constraints. As for column-oriented stores, they generally use tables as data model but do not support table association, which makes it hard to solve some of the questions in this coursework. According to these, it can be said that document-oriented database is more suitable than key-value store and column-oriented store in this task.

However, when handling this email data, it is difficult to recognise the relationship between sender and receiver. According to the features of graph-oriented databases explained in section 3, they can store relationships and properties, thus it is reasonable to say that the graph-oriented database is better than document store in this task.

Talking about MongoDB, some of the fundamental features of MongoDB are shown as followings:

(1) Document database: A record in MongoDB is a document, a data structure composed of field and value pairs. Specifically, the value of fields can be other documents and arrays[2]. What's more, MongoDB stores data in BSON format, a binary JSON-like data format, which is quite suitable for data structure showed above. About BSON, it supports boolean, integer, float, date, string and binary types, this fulfills the condition of every email data in task 3.
(2) High performance: it provides high performance data persistence[2]. In this task, it requires to consider the efficiency of the operations. MongoDB supports embedded data models, which reduces I/O activity on database system, and query through indexes means faster query speed, thus MongoDB is suitable for this task.

In summary, I chose MongoDB based on reasons as followings: first, it is a document NoSQL database, which is more suitable than key-value store and column-oriented store when handling e-mail data in this task. But graph-oriented database may can get the result more efficient. Meanwhile, it has some advantages compared with other document database system. take CouchDB as an example, it does not support a non-procedural language. And the aggregate pipeline is a convenient function for querying and doing some statistic with the query results, which releases the programmer from explicit utilization of indices.

5 HADOOP

5.1 Scenarios

For individual users, Hadoop may be just a new technology which can make search engine or ad-serving easier. But it is widely used by a whole selection of companies in various scenarios, followings are 3 use cases:

(1) Online travel. Tourism activities are sharply increasing these days, thus online travel companies need to provide traveling plans or products based on the analysis results of real-time information. Almost 80 percent of online travel companies booked Hadoop distribution to power up themselves. Especially, they need to build a recommendation system, thus MapReduce Hadoop can help them to reduce the cost when processing the big data[6].
(2) E-commerce. E-commerce companies need to use Hadoop to achieve trend analysis to estimate future event to help them to improve their management. In the e-commerce market, every order has lots of data, thus it is a challenging task to process or analyse the trend in huge amount of data or to extract meaningful information. In this situation, any centralised computation architecture cannot achieve this task. Conversely, Apache Hadoop and Hive can provide data summarization, querying and analysis using parallel and distributed computation architecture[15].
(3) Image processing. There are millions of images shared by people on Snapchat, Instagram and Facebook. Or it can be said that a large of image data has been accumulating. Thus the requirement of using Hadoop MapReduce framework to divide this massive task into several sub-tasks. Especially when trying to do operations such as face detection, image classification, this cloud computing environment can do these tasks with low costs as well as increased performance[17].

5.2 Steps of MapReduce process

The typical stages of MapReduce are illustrated in figure 1. Generally, a Hadoop MapReduce task mainly consist of two user-defined functions: *map* and *reduce*. In the map stage, the main task is to read the corresponding input data block and then split them into key-value pairs. After that entering the shuffle phase, reduce task firstly copy the intermediated data from map node to reduce node, and then sort and merge them in memory. Finally, the reduce task executes after the shuffle phase and generates the final outputs. What's more, shuffle phase is implemented as part of Reduce task in Hadoop[11]. Moreover, some Hadoop tutorials say that the shuffle stage is the most important part and it contains partition, sort, combine, copy and merge these 5 sub-stages.

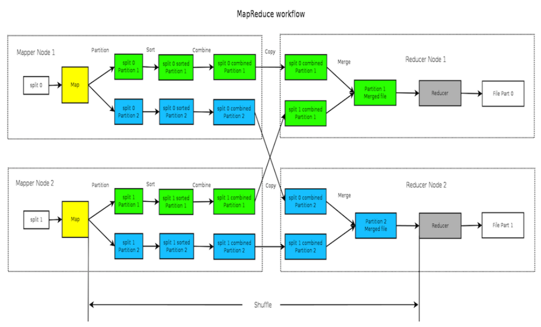
MapReduce workflow
Mapper Node 1
Partition
Sort
Combine
Copy
split 0
Map
split 0 Partition 1
split 0 sorted Partition 1
split 0 combined Partition 1
split 0 Partition 2
split 0 sorted Partition 2
split 0 combined Partition 2
Reducer Node 1
Merge
split 0 combined Partition 1
split 1 combined Partition 1
Partition 1 Merged file
Reducer
File Part 0
Mapper Node 2
split 1
split 1 Partition 1
split 1 sorted Partition 1
split 1 combined Partition 1
split 1 Partition 2
split 1 sorted Partition 2
split 1 combined Partition 2
Reducer Node 2
split 0 combined Partition 2
split 1 combined Partition 2
Partition 2 Merged file
File Part 1
Shuffle

Figure 1: Shuffle

6 PARALLEL AND DISTRIBUTED COMPUTATION VS. CENTRALISED COMPUTATION IN DATA SCIENCE

6.1 Distributed computation

6.1.1 *Advantages.*

(1) High scalability - Take Hadoop as an example, it is extremely scalable. For example, Hadoop was the first tool considered may fix scalability issue in Nutch. Besides, HDFS, the major components of Hadoop, is optimised for high throughput[9].
(2) HDFS has the ability to replicate files for a specified number of time. Meanwhile, it has a high tolerance of software and hardware because it can replicate the data blocks with failures on nodes automatically[9].
(3) Low latency - Distributed systems allow the traffic to hit the closest node since each user can have a node based on their location, this mechanism will provide better performance and lower latency.
(4) Cost effectiveness - Distributed system are much more cost effectiveness than large centralised systems. Even though it will cost more in the beginning, but a distributed system

made of many minicomputers can be more cost effective than a large super machine when the resources reach a certain point.

(5) Efficiency for big data tasks - When using a distributed system to solve a huge question with massive data, programmers can split this task into smaller pieces and assign them to multiple computers, they can solve these pieces in parallel. This greatly reduces demand time.
(6) Flexible - Distributed computation such as MapReduce does not have any dependency on data model and schema in general. Thus programmers can handle irregular or unstructured data model easily[10].

6.1.2 *Disadvantages.*

(1) Higher initial cost - It will cost much more funds and resource to deploy a distributed system. And it needs more advanced technologies to deploy it. Besides, distributed systems are more complex than centralised systems, thus they are more difficult to maintain.
(2) Security concerns - In a distributed system, the data is separated into several different localisation, thus it is hard to ensure the security of any sub-computer, which leads to potentially dangerous for users. For Hadoop and MapReduce, it is disabled by default owing to its extreme complexity though it offers a security model. And Hadoop does not provide storage or network level encryption, this leads to serious concern in government sectors[9].
(3) Low efficiency for small files - It has been proved that HDFS is inefficient when dealing with small files and it has no transparent compression[9]. Moreover, like MapReduce, the distributed computing operations are not always optimised for I/O efficiency because of the reason that they set fault-tolerance and scalability as their primary goals[10].

6.2 Centralised Computation

"'Centralised computing' refers to a model of computing used to conceptualise the relationship between end user and computers."[5]

6.2.1 *Advantages.*

(1) Ease of management and control - There are only a few(perhaps one) computers to manage, and the computers are placed only one location. Because of this, Data management becomes much easier, also, fast backups and efficient error management can easily achieved[13].
(2) Enhance security - Users can hold and manage their information within only one or few computers in a certain location, it is relatively more likely to ensure the physical and logical securing of the computing environment.
(3) High utilisation. A whole selection of the work associated with the business running on the central server means that the machine will work in most of the time. That means making the best economic use of the investment in the computing equipment[5].

6.2.2 *Disadvantages.*

(1) Reduce availability - Once the central server broke down, all end users will lose their computing resource in the past. However, centralised computing facilities take advantage of clustering and other techniques for high availability these days[5].
(2) Perceived bureaucracy - Users only can make changes to central configuration deliberately and most of their manners are restricted because it is necessary to avoid introducing a change that may cause a failure or an outage. Meanwhile, the facility will become inflexible and rigid under this tight control[5].

7 CONCLUSION

In conclusion, relational database, non-relational database, distributed computation and centralised computation are all important technologies in data science. They all play significant roles in different domains. For structured data, using relational database can achieve complex queries. For unstructured or semi-structured data, non-relational databases can store them and allow users to query theme without any pretreatment. What's more, though parallel and distributed computation are widely used within the background of big data, centralised computation and systems still have better performance in dealing with relatively small data files and protecting data security. As a data scientist, it is wise to choose suitable data tools and technologies according to the characters of data needed to be handled.

REFERENCES

[1] Daniel Bartholomew. 2010. SQL vs. NoSQL. *Linux J.* 2010, 195, Article 4 (July 2010). http://dl.acm.org/citation.cfm?id=1883478.1883482
[2] Cansu Birgen, Heinz Preisig, and John Morud. 2014. SQL vs. NoSQL. *Norwegian University of Science and Technology, Scholar article* (2014).
[3] Arnaud Castelltort and Anne Laurent. 2014. NoSQL Graph-based OLAP Analysis. In *KDIR*.
[4] Rick Cattell. 2011. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.* 39, 4 (May 2011), 12–27. https://doi.org/10.1145/1978915.1978919
[5] Angelo F. Corridori. 2012. What is centralized computing. , 3 pages.
[6] Pallavi S. Ghogare and Harmeet Kaur Khanuja. 2015. Automated Travel Itineraries using Hadoop.
[7] Cornelia Győrödi, Robert Győrödi, George Pecherle, and Andrada Olah. 2015. A comparative study: MongoDB vs. MySQL. In *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on.* IEEE, 1–6.
[8] Jing Han, E Haihong, Guan Le, and Jian Du. 2011. Survey on NoSQL database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on.* IEEE, 363–366.
[9] Seyed Nima Khezr and Nima Jafari Navimipour. 2017. MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research. *Journal of Grid Computing* 15, 3 (2017), 295–321.
[10] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, and Bongki Moon. 2012. Parallel data processing with MapReduce: a survey. *AcM sIGMoD Record* 40, 4 (2012), 11–20.
[11] J. Li, X. Lin, X. Cui, and Y. Ye. 2013. Improving the Shuffle of Hadoop MapReduce. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, Vol. 1. 266–273. https://doi.org/10.1109/CloudCom.2013.42
[12] Yishan Li and Sathiamoorthy Manoharan. 2013. A performance comparison of SQL and NoSQL databases. In *Communications, computers and signal processing (PACRIM), 2013 IEEE pacific rim conference on.* IEEE, 15–19.
[13] Robin Jan Maly, Jan Mischke, Pascal Kurtansky, and Burkhard Stiller. 2003. Comparison of centralized (client-server) and decentralized (peer-to-peer) networking. *Semester thesis, ETH Zurich, Zurich, Switzerland* (2003), 1–12.
[14] Dan McCreary and Ann Kelly. 2014. Making sense of NoSQL. *Shelter Island: Manning* (2014), 19–20.
[15] V RamaSatishK. 2016. Trend Analysis of E-Commerce Data using Hadoop Ecosystem.
[16] Vatika Sharma and Meenu Dave. 2012. SQL and NoSQL databases. *International Journal of Advanced Research in Computer Science and Software Engineering* 2, 8 (2012).
[17] Sridhar Vemula and Christopher Crick. 2015. Hadoop Image Processing Framework. *2015 IEEE International Congress on Big Data* (2015), 506–513.

[18] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. 2010. A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective. In *Proceedings of the 48th Annual Southeast Regional Conference (ACM SE '10)*. ACM, New York, NY, USA, Article 42, 6 pages. https://doi.org/10.1145/1900008.1900067