MongoDB and Hadoop

Zhihan Wang
ECS
University of Southampton
United Kingdom
zw3u18@soton.ac.uk

ABSTRACT

Relational databases as traditional data management systems were commonly used to store structured data with a declarative query language. However, with the volume and complexity of data increasing rapidly, relational databases cannot meet the demand anymore. NoSQL databases are introduced to organize big data which are not only structured but also unstructured or semi-structured. This paper compares the differences between these two kinds of databases with regard to their pros and cons. Also, different NoSQL architectures and choices of database technologies based on the given email data are discussed in this paper. To process the email dataset, a framework named MapReduce using parallel algorithm in distributed Hadoop environment is implemented. This paper provides an overview of this process and a detailed comparison based on the attributes of the parallel and distributed computation, and centralized solutions respectively.

CCS CONCEPTS

• **Information system→ Databases**; *key-value pairs, relational database, database query languages, non-relational database, database architecture*

• **Theory of computation→ Algorithms**; *parallel algorithms, distributed algorithms, MapReduce algorithms*

KEYWORDS

Relational, NoSQL, database, MongoDB, Hadoop, MapReduce, parallel, distributed, centralized

1 Introduction

Database is a set of data organized in the computer storage and accessed or manipulated by means of specialized software[1]. To create and manage databases, the database management system is implemented, in which end users can create, retrieve, update and manage data. Two main types of databases used today are relational databases and NoSQL databases.

Relational databases only organize the structured data, whereas NoSQL databases could manage structured, semi-structured and unstructured data[2]. Each database has its own costs and benefits.

Since data were organized in the databases, to acquire the information behind these data, computations would be implemented on them. Some computations like summaries of the number of specific words occurred, or finding maxima/minima of a set of numbers, are conceptually straightforward[9]. However, the input data are always large and the computations have to be distributed across a number of servers to reduce the computing time[3]. The MapReduce process helps to distribute data to different computers, map key-value pairs, and merge values together to produce much simpler and clearer outputs. Hadoop, an open source collection of frameworks, provides an environment where MapReduce could be used to process huge data. It is popular as it has a lot of good properties such as free to use and supporting a wide range of programming languages.

This paper mainly consists of two parts, databases and the MapReduce process, with detailed descriptions, cost-benefit analysis and comparisons based on different data management technologies.

2 Relational databases and NoSQL databases

2.1 General

Relational databases are based on relational models, in which data are fairly structured and clearly defined. They are supported by a declarative language, structured query language (SQL),an English-like language and the most commonly used[4]. So relational databases are also named as SQL databases. In a relational database, relational data are organized into a series of tables, consisting of rows identified by keys and columns by corresponding values. Relational databases could guarantee four principles, atomicity, consistency, isolation, and durability (ACID). With the rigid structure, data are clear and easy to understand or maintain. But also owing to this inflexible structure which increases the complexity of data processing, SQL databases are not suitable for complex data such as texts, images or videos.

In reality, data structures change frequently and new type of data could be incorporated at any time; therefore, it is not possible that these complex data structures could be figured out in advance all the time. SQL which is designed only for structured data is difficult to work with unstructured ones as well. Besides, large data always need to be divided to process, but it is complicated to join results using tables[5]. New types of storage are required.

NoSQL databases are more flexible as they are schema-free. SQL and relational models are not necessarily used here. So NOSQL databases could store semi-structured or unstructured data which account for more than 80% data collected today. However, without strict structure restrictions, this type of database loses capacity to guarantee ACID principle[2]. BASE stands for basically available, soft state, eventually consistency model. It does not require consistency. Instead of ACID, NoSQL databases only can follow BASE principle.

2.2 Main differences

As referred before, each database has its own attributes and applied circumstances. Comparisons on fundamental differences between NoSQL databases and relational databases are as follow:

1. *Type of data*: Data in relational databases are totally structured but could be not only structured but semi-structured or unstructured in NoSQL databases.
2. *Schema*: Schemata stay the same in relational databases all the time, but change frequently and accordingly uncertain in NoSQL databases.
3. *Fields*: Relational databases work well in fields like inventory, or funnel analysis using relational models as these data generally are structured, whereas NoSQL databases would be better in more complex problems such as text mining, or language processing[4].
4. *Language*: Most relational databases use SQL[6], whereas most NoSQL databases not only support SQL but many other different query languages with their own unique APIs[7].
5. *CAP-Theorem*: Relational databases are good at Partition Tolerance and Consistency, compared to NoSQL databases performing better in Partition Tolerance and Availability.
6. *Servers*: The data in a centralized relational database generally stored in one server as they are related and difficult to divide. NoSQL databases, on the other hand, distribute data to multiple servers. They support parallel and distributed computation[5][7].
7. *License*: Most NoSQL databases are open sources, but relational databases are not.

There are many differences between these two most common databases based on their unique attributes and features. Therefore, choices of databases should be based on the specific application.

3 NoSQL databases

3.1 Different architecture

Unlike a relational database only with a 'table' structure, there are four main NoSQL architectures based on different data types[5].

1. *Key-value*: This is the simplest NoSQL architecture and also the mother of other architectures. It organizes data using hash tables with rows and columns. Each row is a unique item consisting of a key/pointer and corresponding values. The values could be numbers, texts, and some other complex structures.
   **Pros and cons*: The unique key access could be used to easily read and write data. In a model which requires a higher speed to perform data, key-value architecture would be a good choice. However, when some data updates are required, this kind of store lose flexibility because of its strict structure. Besides, it is also not suitable for the strongly related data. And it only supports Java Script language and JSON file, which is a big limitation.
2. *Document-based*: Similar to the key-value databases, in a document store, the document could be queried using its unique key. The difference is that in each document there are more than one unique key corresponding their values, and each key with its value could be regarded as a record in the document. Document-based databases store data using collections of documents instead of tables. The schemata are not fixed, so the records types in each document could be different. This type of database can store unstructured or semi-structured data like texts. Another point worth mentioning is that a document could be embedded as a record in the set of documents to achieve denormalization.
   **Pros and cons*: It could organize complex data because of its flexible schema. Handling query is also simple using code[6]. It supports almost all query languages except SQL, and all data formats like JSON, BSON, XML, and RDF. But transactions with complex operations are not available because the modification could only be done automatically within a single document and should be done manually in multiple documents.
3. *Column-based*: Data are grouped in columns from a column family. Data are read and wrote using columns rather than rows. In other words, data are aggregated and retrieved by attributes, not by key names.
   **Pros and cons*: Data with the same attribute stored together are easily analyzed in some popular analytics tools like R and Spark[7]. It would be better for fast access because a query only needs to locate the column entry instead of going across all items from each row in a row-oriented database. However, with this nature, it would be very slow when data need to be extracted from different columns. To make a transaction in a specific item is difficult as well. In addition, Column-based database only supports JSON format and SQL, which is limited.
4. *Graph-based*: Data are be represented using nodes and edges. Nodes represent entities and edges represent relationships between items.
   **Pros and cons*: There are no tables, no collections, and totally format-free in a graph-based database. It is more like a native graph with a high scalability and index-free adjacency for data access. So data can easily be transformed from one model to another[4]. The relationship between data would be clear as well. But transaction is hard to implement and queries such as sum or max could not be efficient. Besides, there are a limited number of commonly used query languages and analytics methods for a graph-based database.

3.2 Choices for the Enron email dataset

There are around half a million email items including user names, id, datetime, threads, subjects, and contents of messages in the Enron email dataset. The dataset is provided in a JSON file. Data in it are not well cleaned and the structure of each email items is not uniformed. Besides, some features in the items are numerical, and some are text. As the data are large and the

schemata are uncertain, it would be better to handle Enron email dataset using a NoSQL database.

In Enron, data need to be extracted based on each item. Each email item can be regarded as a document, and the whole dataset is a collection of documents. Each document is retrieved using a unique key which could be the user id or something else from the dataset. The relationships between documents are weak. In each document, there are many key-value pairs as records, but some of them are with empty or null values. Some records are embedded documents. In addition, the length of each document is not identical.

In conclusion, a document-based NoSQL architecture would be the best choice.

4 Database technology

In task 1, Enron email data are processed with MongoDB, a non-relational database technology. There are some reasons why MongoDB is chosen there.

Firstly, the reason why using a non-relational database is referred in 3.2. A non-relational database provides cheap storage, flexible schemata and distributed architectures for massive amounts of structured, semi-structured and unstructured data. It is much more suitable for Enron dataset than a relational database, as email data are large and complex. In addition, NoSQL databases are generally faster than SQL databases to instantiate database buckets and query[8].

There are a wide range of NoSQL database technologies such as MongoDB, CouchDB, Hadoop, and Neo4J. These technologies based on different data storing styles specialize on specific aspects. All of them have NoSQL properties and also have their own features. MongoDB database is document-based, and documents in it are stored as JSON, XML, or BSON formats. It focuses on four characteristics, flexibility, strength, speed and ease of use. MongoDB database has a core API in JavaScript and a series of other APIs like PyMongo, which could be used in different languages like Python[12]. It also grows in popularity.

The Enron data is in JSON format and denormalization widely exists in the documents. Besides, the dataset in the task needs to be handled in Python. So MongoDB would be an available choice.

The performance of operations on the database should also be considered when to make a choice from different database technologies. The comparisons are shown as below[8].

Table 1 compares time for adding key-value pairs into the bucket and table 2 summarizes the time for deleting. From two tables, MongoDB and Couchbase are much faster in writing and deleting.

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	61	75	84	387	2693	23354
RavenDB	570	898	1213	6939	71343	740450
CouchDB	90	374	616	6211	67216	932038
Cassandra	117	160	212	1200	9801	88197
Hypertable	55	90	184	1035	10938	114872
Couchbase	60	76	63	142	936	8492
MS SQL Express	30	94	129	1790	15588	216479

Table 1: Time for writing (ms)

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	4	15	29	235	2115	18688
RavenDB	90	499	809	8342	87562	799409
CouchDB	71	260	597	5945	67952	705684
Cassandra	33	95	130	1061	9230	83694
Hypertable	19	63	110	1001	10324	130858
Couchbase	6	12	14	81	805	7634
MS SQL Express	11	32	57	360	3571	32741

Table 2: Time for deleting (ms)

Database	Number of keys to fetch					
	10	50	100	1000	10000	100000
MongoDB	4	4	5	19	98	702
RavenDB	101	113	115	116	136	591
CouchDB	67	196	19	173	1063	9512
Cassandra	47	50	55	76	237	709
Hypertable	3	3	3	5	25	159
MS SQL Express	4	4	4	4	11	76

Table 3: Time for fetching(ms)

Table 3 provides a comparison about time for fetching the whole keys. All the databases in the table are quick to fetch keys except CouchDB[8]. Note that Couchbase which performs the best in deleting and writing is not in the table as it does not have an API to support fetching.

In conclusion, according to the performance of operations in different databases, MongoDB could get a relatively higher score overall. Therefore, based on these comparisons and features of the Enron dataset, MongoDB is chosen to process email data.

5 Hadoop

As said before, today's data are always in large size, from different sources, and coming in real-time. They are so complex that traditional storage and analytical processes are no longer suitable.

Although the storage capacities of hard drives has increased massively over the years, the access speed has not kept up[11]. Therefore, processing data in only one terminal is not available anymore. A new method which could handle data in a parallel and distributed style should be applied.

Hadoop is a collection of frameworks, which also could be named as an eco-system. It encompass a variety of components to store, analyse and maintain data, like HDFS (Hadoop distributed file system), MapReduce, Yarn and so on. The storage in it is provided by HFDS and analysis could be achieved by a MapReduce process. It can store and distribute large datasets across multiple cheap servers and operate them in parallel[9]. Therefore, Hadoop helps to save costs in the hardware and also reduce time spent on reading and writing files.

5.1 Hadoop implementation

Hadoop is widely used in some commercial circumstances because of its convenient and great properties. For instance, Hadoop helps the websites to respond quickly in some specific period of time when the number of coming demands are rapidly increasing. This is achieved by distributing coming demands from visitors to multiple servers. Another example is that, to understand the customers better, British Airways launched a program named 'Know Me' using a MapReduce process.

The detailed scenarios where Hadoop is used are as follow[11]:

- *Data staging*: ETL stands for extracting, transforming, and loading data. Hadoop could take care of ETL processing by using MapReduce. MapReduce loads data into HDFS and then transforms them into the data warehouse[4]. The unprocessed and already transformed data are stored together, which reduces the need of ETL tools. Furthermore, without data pre-processing, the potential value of the raw data would be kept and the costs spent on it are saved. It is easy to manage, reprocess and analyze data anytime using MapReduce.
- *Data processing*: For some unstructured data and semi-structured data, traditional data processing methods using data warehouses are not available anymore since a large number of data warehouses are built on specialized infrastructure and the data there could only

be scaled vertically. So it is not easy to update data or add capacity[7], especially for today's data, from multiple sources and always in different types, numbers, texts, images or videos. But in Hadoop, these data could be easily organized, updated and then sent to warehouses.

- *Data achieving*: Using Hadoop to store large volume of data would be very cost-effective. Instead of destroying some old but still important data to save storage, using Hadoop to store these data is in limited costs. Besides, it is faster and more convenient to derive meaning from unstructured data such as the email contents in Enron dataset using MapReduce in Hadoop.

5.2 MapReduce process

MapReduce is a programming model which could be implemented in the Hadoop eco-system as a framework. In a MapReduce process, input data could be in a free structure. Data are broken into blocks and then distributed across multiple servers. The main concept of MapReduce process is 'divide and conquer', which is achieved by two main phases, Map phase and Reduce phase[11].

Firstly, the input data need to be split. Then, in the Map phase, a map function is executed to map the key and value based on the split data. The next process is completed in the reduce phase which could be separated into two steps.

- *Shuffling*: The outputs of map task are transferred to the machine, in which they could be merged and passed to the reduce function.
- *Reducing*: The outputs from shuffling are aggregated in the reduce function. And then the keys in key-value pairs are combined with the aggregated value to form the final results. Lastly, HDFS stores the final results.

6 Distributed and centralised

Parallel and distributed computation uses multiple servers to achieve one goal[9]. Servers are independent but cooperate with each other using a shared memory or commuting with each other.

In the centralised system, on the other hand, only a single computer or server is used to store data and achieve all calculations. These two type data solutions have their own pros and cons respectively.

6.1 Centralised processing

As the name referred, everything in a centralised processing is centralised. This kind of style has its own main benefits:

1. *Easy to manage and control*: There is only one terminal or location with only few computers in a centralised system, which means instructions could be much easier to be located and then debugged.
2. *Reduced cost*: Since there are only a limited number of computers, fewer people are required to manage servers, which reduces costs on managed people.
3. *High utilization*: The fewer servers in a centralised system are always busy with much work. The computing equipment with an ample utilization is more economic.
4. *Security*: As all of processing is controlled in the central location, it would be much easier to monitor the single center to guarantee security.

However, like every paper with two sides, there are also some costs existed.

1. *High performance requirement*: A centralised system totally depends on the central computer. So if it crashes, efficiency of the whole system goes down[9]. So a high requirement comes to the central computer.
2. *Slow*: Every computation has to be completed in the central computer, which results in an extremely low speed when to process larger data.
3. *Increased cost*: A high demand to the equipment corresponds to an increased cost.
4. *Low fault resistance*: Since all of commands are ran in a single location, even a tiny mistake could cause a total paralysis in the system.
5. *Inflexible*: Because of the tight control, every slight change has to be completed in the terminal. It would be very slow and inconvenient.

6.2 Parallel and distributed computation(PDC)

Instead of only depending on a central server, in the PDC, a task is distributed to multiple nodes to achieve. The advantages of this kind of solution are as below[9]:

1. *Improved availability*: A single failure could only affect one or few nodes. For other end users, the nodes could still work as before.
2. *Faster*: Parallel and distributed computation reduces computing time making the system more efficient.
3. *Increased flexibility*:
   - Introducing changes is not required to go back to the terminal anymore. Instead, implement instructions on the corresponding nodes.
   - Instead of upgrading the whole system in the central computer, add additional capacities by adding additional distributed computers.

On the other hand, properties bringing benefits can bring costs as well.

1. *Increased cost*: More computers to manage increases the investment spent on the managed people and the large number of computers.
2. *Increased fault rates*: More people and servers involved would bring more unpredictable errors.
3. *Low utilization*: More nodes are only used for a single purpose, which makes the system not economic.

7 Conclusion

Each database or technology has its own properties and is not perfect. NoSQL and Hadoop with MapReduce are more suitable for big and complex data but not excellent for everything**.** The choices of database architecture and technologies should be based on the specific application.

REFERENCES

[1] Macmillan. Beynon-Davies P. (2004) Database, Dbms and Data Model. In: Database Systems. Palgrave, London, 30-47.

[2] Han, Jing, et al. "Survey on NoSQL database." Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011.

[3] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[4] Nenad Jukic, Susan Vrbsky, and Svetlozar Nestorov. 2016. Database Systems: Introduction to Databases and Data Warehouses (1st ed.). Prospect Press.

[5] Leavitt, Neal. "Will NoSQL databases live up to their promise?" Computer43.2(2010). White, Tom. Hadoop: The definitive guide. "O'Reilly Media Inc.",2012.

[6] Melton J. (1998) Database Language SQL. In: Bernus P., Mertins K., Schmidt G. (eds) Handbook on Architectures of Information Systems. International Handbooks on Information Systems. Springer, Berlin, Heidelberg.

[7] Sharma, Vatika, and Meenu Dave. "SQL and NoSQL databases." International Journal of Advanced Research in Computer Science and Software Engineering 2.8 (2012).

[8] Li, Yishan, and Sathiamoorthy Manoharan. "A performance comparison of SQL and NoSQL databases." Communications, computers and signal processing (PACRIM), 2013 IEEE pacific rim conference on. IEEE, 2013

[9] Ghosh, Sukumar. "Distributed Systems: An Algorithmic Approach, (Chapman Hall/CRC Computer and Information Science Series)." (2007).

[10] Van der Veen, Jan Sipke, Bram Van der Waaij, and Robert J. Meijer. "Sensor data storage performance: SQL or NoSQL, physical or virtual." Cloud computing(CLOUD), 2012 IEEE 5th international conference on. IEEE, 2012.

[11] White, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[12] Banker, Kyle. MongoDB in action. Manning Publications Co., 2011.

I am aware of the requirements of good academic practice, and the potential penalties for any breaches.