Journal of Applied Science and Engineering Innovation Vol.2 No.2 2015
ISSN (Print): 2331-9062
ISSN (Online): 2331-9070

Survey on NoSQL Database Technology

Changlin He[1]
[1] *Information Technology Service Center of Hexi University, Gansu Zhangye 734000, China*

Abstract: As one of the products of the big data era, NoSQL data base is used to solve the problem of the storage and process of the data. Intending to give references to NoSQL data base user, this paper discusses the theoretical basis and classification of NoSQL based on the introduction of the emergence and the development of database from Relational to NoSQL and the analysis of its limitations of the relational database in the very era.

Keywords NoSQL; Big data; Database; Storage systems; CAP; BASE

INTRODUCTION

Anybody and anything will leave some digital marks in the context of the information society, which are also called the data. Rapid progress in the information society generates a large amount of data, the massive increase in the data volume generates a large amount of information, the information spawns new applications, and the new applications lead to new data. It is in this way that the information and the application develop iteratively, enabling the data to snowball. According to the statistical study that IDC Digital World conducted in 2012[1], the data size around the world has reached the order of magnitude of ZB in 2010, compared to 130EB in 2005, meaning that the data volume has increased by 10 folds within 5 years. This implies that at this increment speed, the data size around the world will reach up to 40ZB in 2020. According to the ZDNET annual technical report *Data Center 2013: Hardware Reconfiguration and Software Definition*[2], China created over 0.8ZB of data in total for 2013, which is two times as much as that for 2012 and equal to the total amount generated by the world in 2009. It is estimated that China will create over 8.5ZB of data in 2020, 10 times that for 2013. The quick increase in data volume heralds the arrival of big data era. So it is urgent to ascertain how to store and manage the massive amount of data in the context of big data era.

OVERVIEW OF NOSQL

There has been no strict definition of NoSQL so far. It is a general term for the open-source, distributed, and non-relational database techniques. NoSQL originates from the first version of Berkeley DB in 1991. Berkeley DB is a Hash database that stores key/data pairs and fitted for embedded applications that have simple data types but require extremely fast inserting and reading speed[3]. NoSQL began to flourish in 2007. Engineers from Google and Amazon published many papers on BigTable and Dynamo databases, describing the ideas on the novel database they have adopted. BigTable proposed the column storage model, demonstrating that the persistent storage of data can be extended to thousands of nodes[4]. Dynamo proposed to achieve greater availability and extendibility via eventual consistency[5]. The distributed cache system Memcadied proved that distributed internal storage of data can achieve greater performance[6]. In fact, BigTable from Google, Dynamo from Amazon and Memcached are the ancestors of all NoSQL databases. Inspired by their ideas, many companies and organizations have developed their own open-source NoSQL databases. For example, Hypertable is an open-source implementation of BigTable.

There are over 100 types of NoSQL databases around the world[7], and the NoSQL databases have flourished in China since 2009.

In spite of the considerable momentum for NoSQL growth, it is not long since its development and its database techniques need to be further refined in practical applications, because there are both successful and failed NoSQL applications. In April, 2010, Twitter decided to suspend the plan to store feed by using Cassandra as a substitute for MySQL, on the ground that Cassandra has insufficient cases and experiences of parallel access to mass data and that it has to endure a period of time before the new produce stabilizes[8]. Having adopted MongoDB, the website of foursquare was down for 11 hours due to data fragmentation and poor monitoring. The NoSQL database products are still flawed; its architectures as well as the operation and maintenance abilities need to be improved in its infancy, implying that there is still a long way to go before it can rival the relational database.

THEORETICAL PRINCIPLES OF NOSQL

The CAP theory, BASE theory and eventual consistency are regarded as the three theoretical foundations for NoSQL. As for hardware costs, the five-minute rule theatrically enables internal storage

Corresponding Author: Changlin He, Information Technology Service Center, Hexi University, Zhangye 734000, China.

of data. These principles are derived from long-term practical applications.

CAP theory

The CAP theory was first proposed by Eric Brewer in 2000[9]. The correctness of the CAP theory has been proved by Seth Gilbert and Nancy Lynch in 2002. The CAP theory states that it is impossible for a distributed system to provide consistency, availability and partition-wise tolerance at the same time, and that it can only meet two of the three requirements at most simultaneously, as shown in Figure 1.

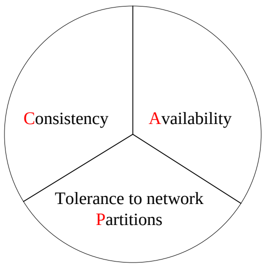

Figure 1. The CAP theory

If a distributed system has high requirements on consistency and partition-wise tolerance, then the user must process failed read and write due to system unavailability. If the user chooses high availability and partition-wise tolerance, then consistency will be a very challenging problem and even dirty read may occur. The traditional relational database system chooses high availability and consistency by default, resulting in both poor horizontal scalability and little room for improvement. NoSQL provides a solution to achieve a trade-off between the three choices to meet diverse requirements.

CAP is not perfect. In practical applications, the system designers should balance between the three factors. For example, the relational database provides high consistency and availability but its scalability is poor. Some databases (e.g. Big Table, HBase, MongoDB, Redis) are characterized by high consistency and scalability but the availability is dissatisfactory. Some databases (e.g. Dynamo and Cassandra) are good at scalability and availability but poor at consistency. Therefore, it may be necessary for applications to combine several databases. For some non-real-time data that is insensitive to consistency, it can choose the system like Cassandra; for the data highly sensitive to consistency, the relational database is still the chief choice.

BASE

The real meaning of the BASE model[10] is Basically Available, which mainly has three aspects: Basically Available, Soft-State and Eventual Consistency. In practical applications, the user usually has high expectations on transactions. For example, the data cannot be lost, the transaction cannot be inconsistent, and the system must be available. But the system that can meet all user requirements completely does not exist. The user has to grasp an understanding of the most essential system requirements before choosing a proper database.

Consider the blog data. It suffices to limit inconsistency to the order of magnitude of minutes, because occasional inability to write has no serious impacts on the application. But any problem with the read operation is unaffordable. Loss of some old data will not severely affect the application. Therefore, the database that enables read-write separation, new-old data separation and horizontal scalability is recommended.

Eventual Consistency

Eventual consistency usually refers to relaxed process, tight outcome and eventual consistency [11]. It is sometimes called soft consistency. The concept of consistency has long been present in the computer science. A stand-alone computer can improve its multi-core concurrency performance via different consistency protocols. In the cluster environment, it can improve NoSQL's scalability. Consistency has many types and it can be mainly categorized into strong consistency, causal consistency, monotone read consistency and monotone write consistency.

NOSQL CLASSIFICATION

The data model of the database determines the logical organization of the data. Its query model specifies how to retrieve and update the data. The currently available NoSQL databases can be classified into the column storage-oriented type, the key/value storage type, the document storage-oriented type and the graphic storage type.

Column-oriented orderly storage

With explosive increase in data size, the column-oriented NoSQL database is receiving more and more attention. The traditional relation databases store and process the data with line as the unit. Hence, the traditional relation databases are also called the line-oriented database. The column-oriented database processes and stores the data with column as the unit.

The column-oriented database is so scalable that the increase in data size will not result in decreased processing speed. So it is well-suited for processing huge amount of data. For the line-oriented database, the column-oriented database can be used for the storage of the batch program to update the massive amount of data. However, the idea of the column-oriented database is different from that of the traditional database, making it difficult to apply it to practice. The typical examples of the column-oriented databases include Cassandra, HBase and HyperTable.

Key/value storage

The hash table or the associative array is the simplest type of data structure that can contain the key/value pair. This data structure is so efficient that the time complexity of its data access is O(1), thereby gaining popularity among users. The key of the key/value pair is unique among the set and easy to find, so it facilitates data access. The key/value storage-based NoSQL database is the most common NoSQL database. It provides very fast processing speed, but its access to data can only be performed by querying about consistency of the key. According to the way of data storage, this type of databases can be classified into the temporary, permanent and hybrid categories.

The temporary database can speed up data storage, read and write by storing the data into the internal storage. But once the system collapses, the data will be lost. Furthermore, in this type of database, the data is stored into the internal storage, so it cannot process data whose size exceeds the internal storage capacity. Unlike the temporary category, the permanent database stores the data into the hard disk. Its performance is degraded as it needs to perform I/O operations on the hard disk while processing the data. But its greatest advantage is that the data will not be lost. The hybrid database combines the advantages of the temporary and permanent key/value storage. It first stores the data into the internal storage, and then writes the input into the hard disk when the specified conditions are met. This ensures the data processing speed of the temporary database, and guarantees data persistency by writing the data into the hard disk. The typical examples of the key/value-based NoSQL database systems include Redis, LevelDB, Tokyo Cabinet, and Berkeley DB.

Document-oriented data storage system

The document-oriented database is not a document management system. The developers who are new to NoSQL usually confuse the document-oriented database with the document/content management system. The term *document* in the document-oriented database refers to the loose set of key/value pairs, usually JSON, rather than the traditional documents and tables. The document-oriented database regards the document as a whole, instead of partitioning the document into many key/value pairs. This enables the documents of different structures to be put into the same set. The document-oriented database supports document index, including not only primary identifiers but also document properties. The typical examples of the document-oriented databases include MongoDB, CouchDB and Terrastore.

Graphic data management system

Compared with the above three types, the graphic database and the SML data storage can also be regarded as the NoSQL database. But among the NoSQL products, the graphic database is somewhat special. Its success is mainly attributed to its special data model: unlike other models with good scalability and performance, it stores the data in the form of the graph. The graphic database can store nodes and edges of the graphs and set the weights of the edges and nodes. It also provides some graphic algorithms and supports graphic query.

Like the document-oriented NoSQL, the bottom layer of the graphic database has different implementations. The bottom layers of some graphic databases are the key/value storage, while others are the document storage. The typical examples of the graphic database are Neo4J, FlockDB, InfoGrid, and HyperGraphDbs.

NoSQL Database

NoSQL database of many kinds, and each are not identical, table 1 lists some NoSQL.

Table 1. Some NoSQL Database

Name	Manufacturers	Language	Platform	License	No definitionSchema	Extensible
Column-oriented orderly storage						
HBase	Hadoop	Java	Java	Apache 2.0	Y	Y
Azure Tables	Microsoft	.Net	Azure	SaaS	Y	Y
Cassandra	Apache	Java	Java	Apache 2.0	Y	Y
Hypertable	Open source	C++	Linux/Mac	GPLv2	Y	Y
SimpleDB	Amazon	Erlang	EC2	SaaS	Y	Y
Document-oriented data storage system						
MongoDB	Open source	C++	Linux/Mac/Windows	Friendly AGPL	Y	Y
CouchDB	Open source	Erlang	Linux	Apahce 2.0	Y	N
Terrastore	Open source	Java	Java	Apache 2.0	Y	Y
Key/value storage						
Redis	Open source	C	Linux	BSD	Y	Y
LevelDB	Open source	C	Linux	BSD	Y	Y
Tokyo Cabinet/Tyrant	Open source	C	Linux/Windows	LGPL	Y	N
Berkeley DB	Open source	C	Linux/Mac/Windows	Sleepycat License	Y	Y
MemchachedDB	Open source	C	Linux/Mac/Windows	BSD	Y	Y
Graphic data management system						
Neo4J	Neo Technologies	Java	Java	AGPL/Commercial	Y	N
InfoGrid	NetMesh Inc	Java	Java	AGPL/Commercial	N	N
HyperGraphDB	Kobrix	Java	Java	LGPL	N	N

NoSQL Framework

Generally, the framework of the NoSQL database can be partitioned into four layers: interface layer, data logic model layer, data distribution layer and data persistence layer. Each layer consists of different interfaces and functional modules, as shown in Figure 2.

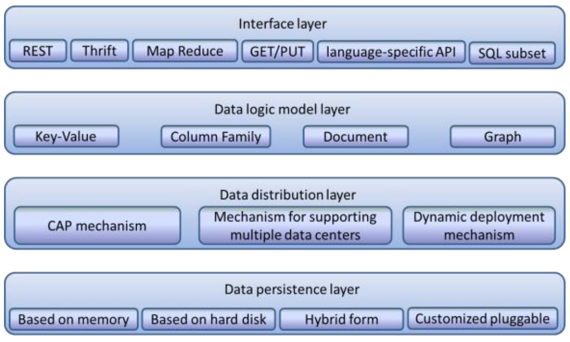

Figure 2. Framework of NoSQL

Interface layer

The interface layer is mainly responsible for providing proper and convenient data access interface for upper applications. The interfaces provided are far more than those from the traditional relation databases. Major interfaces provided by this layer include: (a) the common REST (Representational State Transfer), which is adopted by HBase and CouchDB; (b) the RPC protocol Thrift from Facebook, which is integrated into HBase and Cassandra; (c) the MapReduce suited for processing the large amount of data, like HBase, CouchDB and MongoDB; (d) the Memcached-like Get/Put method, which is adopted by Voldemort; (e) the language-specific API, like JAVA, with MongoDB being very good in this regard; and (f) the SQL subset. Although "Join" is a taboo in NoSQL, it is a good idea to help users by providing a basic SQL subset.

Data logic model layer

The data logic model layer is mainly responsible for describing logical representation of the data. NoSQL has greater flexibility in logical representation than the relational database. The main representations of this layer encompass (a) the most common Key-Value form, which is monotonous but has advantage in scalability; (b) the column family which supports more complicated data than the Key-Value form but is inferior in scalability; (c) the document form, which originates from Lotus Notes, a famous collaboration software from IBM. It is very similar to Key-Value in fact, and the main difference is that the Value only stores document data and that the document form performs well in terms of complicated data support and scalability. (d) The graph form, which is not used widely. It is customized for the graphic data, such as the relation in the SNS applications.

Data distribution layer

The data distribution layer is mainly responsible for defining the distribution of data. Its difference with the relational database is that the NoSQL database has many choices of mechanisms. These mechanisms include: (a) the CAP mechanism for horizontal expansion, which is supported by HBase, MongoDB and Cassandra; (b) the mechanism for supporting multiple data centers, ensuring that the NOSQL database can operate stably across different data centers, like Cassandra; and (c) the dynamic deployment mechanism, allowing a node to be added or deleted dynamically and smoothly in a production cluster.

Data persistence layer

The data persistence layer is mainly responsible for defining the forms of data storage. These forms encompass: (a) the internal storage form, which is fastest but may suffer data loss. Redis is a database that supports this form. (b) The hard disk form, which provides good data persistence but is inferior to the internal storage form in terms of the speed. MongoDB adopts this form. (c) The hybrid form, which combines the advantages of the previous two patterns, providing good speed without the possibility of data loss. So it is regarded as the most appropriate scheme. This form is adopted by HBase and Cassandra. (d) Customized pluggable form, which is famous for flexibility.

Although the framework of NoSQL can be partitioned into four layers, it does not mean that each product can only select one property at each layer. Instead, it is capable of choosing multiple properties. For example, HBase supports REST, Thrift and Map Reduce at the interface layer. At the data distribution layer, Cassandra supports CAP and multiple data centers.

Conclusions

NoSQL has received much attention from the industry and academia because it can store a huge amount of data, is cost-effective, flexible and scalable. This paper first discusses the necessity of NoSQL, completely surveys the development of NoSQL databases, describes the three theoretical foundations and classifications of NoSQL, and finally analyzes the framework of NoSQL. NoSQL databases may bring an end to the traditional relation database in the future. But it is not long since NoSQL is born. Instead of substituting for the traditional databases altogether, NoSQL is only suited for some applications. In some sense, it is just a supplement to the traditional database. For specific system applications, users should specify system requirements and then choose the right database. Therefore, there is still a long way to go to improve NoSQL.

ACKNOWLEDGMENT

This work is supported by Teaching research project of Hexi University at Zhangye (No. HXXYJY-2014-017) and Gansu province college graduate tutor project of the education department of Gansu province(No. 1210-02).

REFERENCES

John Gantz and David Reinsel. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East[R]. IDC iView, Sponsored by EMC, December 2012.

Guangbin Zhang， Jun Pan， Zhiqiang Zeng. Data Center 2013：Hardware reconfiguration and software defined [R]. ZDNET，Beijing，2014.1.

Vijaykumr S, Saravanakumar S. Implementation of NoSQL for robotics[C]. Emerging Trends in Robotics and Communication Technologies(INTERACT), 2010 International Conference on: 195-200.

Fay C, Jeffery D, Sanjay G, Wilson H, Deborah W, Michael B, Tushar C, Andrew F, Robert G. BigTable: a distributed storage system for structured data[J]. ACM Trans. Comput.Syst.,2008(26).

Giuseppe D, Deniz H, Madan J, Gunavardhan K, Avinash L, Alex P, Swaminathan S, Peter V, Werner V. Dynamo: Amazon's highly available key-value store[J]. ACM, 2007, 205-220.

Petrovic J. Using Memcached for data distribution in industrial environment[C]. Syatems,2008,ICONS 08,IEEE Third International Conference on:368-372.

Jose J, Subramoni H, Miao L, Minjia Z, Jian H, Wasiur M. Memcached design on high performance RDMA capable interconnects[C]. Parallel Processing(ICPP), 2011 IEEE International Conference on:743-752.

Becker M.Y, Sewell P. Cassandra:flexible trust management, applied to electronic health records[C]. Computer Security Foundations Workshop,2004 Proceedings. 17th IEEE: 139-154.

Julian Browne. Brewer’s CAP Theorem. 2009. http://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

John D. Cook. ACID versus BASE for database transactions. 2009. http://www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/

Werner Vogels. Eventually Consistent[J]. Queue,2008,6(6):14-19.

Baoyao Zhou, Wei Liu and Chenggong Fan. Big Data—— Strategic technology and Practice [M]. Beijing：Publishing House of Electronics Industry，2013:88.