NoSQL Database: A Scalable, Availability, High Performance Storage for Big Data

Yu Huang and Tiejian Luo

University of Chinese Academy of Sciences, Beijing, China
huangyu111@mails.ucas.ac.cn,
tjluo@ucas.ac.cn

Abstract. This paper is focused on NoSQL databases which designed to handle the problem of storing large amounts of data. NoSQL database sacrifice some consistency to achieve horizontal scalability and high-performance. We aim at giving a systematic overview of NoSQL, introducing concepts, techniques and categories. For each category we give several typical NoSQL databases and discuss them in detail. Finally we analysis the challenges of NoSQL databases and give some predictions.

Keywords: NoSQL database, CAP theorem, BASE theorem, category, horizontal data distribution, weak consistency.

1 Introduction

Internet service has changed in fundamental ways over the last 25 years, from the first on-line web services which only offered static pages of news and messages to todays various, big data web and applications. According to a research from IBM, everyday we create 2.5 quintillion bytes of data, more than 2 billion users are active in social media sites [17]. YouTube.com receives over 3 billion hits daily, and the e-commerce platform in Amaze deals millions of transactions a day. The Internet presents an unprecedented scale and demand with headache to the designers: how to design an Internet service which must be responsive, robust, always available, low-latency and high-performance of supporting thousands of millions of users demand.

With the booming user number and demands of social content web sites, a modern Web service must be available for potentially larger user populations, supporting millions of concurrent users without downtime, adjusting easily to application capacity by changing the number of application servers. In addition, a good building framework of web service must reduce the cost of system and promote the resource usage. To meet the new challenges, web services should have four properties which the designer must consider: concurrency, sustain service, low latency and scalability [14].

Concurrency of load is fundamental to the web services, a good architecture of Internet services could deal with unexpected abruptness promptly. Continuous service is also very important. Users will feel uncomfortable if they find the sites are often in downtime. Access latency usually is limited by the bandwidth,

Q. Zu et al. (Eds.): ICPCA/SWS 2013, LNCS 8351, pp. 172–183, 2014.

but an optical design could improve the throughput to minimize the latency. To support more users for a web service, system needs more commodity web servers, and we expect the system is scalable and the system cost linearly with increases in users.

Unfortunately, facing with new challenges, the traditional database technology, Relational Database Management System (RDBMS) cannot keep up to the demands, a new contender has risen to challenge the supremacy of relational database. NoSQL (mostly interpreted as non-relational database) is a broad class of database management systems which do not follow the relational data model, which generally do not use SQL for data manipulation. This class of database seeks to make breakthroughs in the rigidity of relational model, using various models (include key/value stores, document-stores, graph databases), which can store data without first requiring one to define a database schema. In contrast, relational database requests information being defined in relationship before data can be stored. According to *Rick Cattell* in [6], NoSQL databases generally have six key features:

- the ability to horizontally scale "simple operation" throughput over many servers,
- the ability to replicate and to partition data over many servers,
- the ability to call interface or protocol simply and loosely,
- a weaker concurrency model than RDBSM,
- use distributed indexes and RAM for data storage efficiently,
- the ability to dynamically add new attributes to data records.

Obviously, NoSQL databases are far more better than relational databases when working with huge quantity data, and they are also well match for the needs of modern web services.

This paper analyzes the benefits and detriments of taking NoSQL databases instead of relational databases. We also compare the characters and performance of several current available NoSQL databases in detail. Finally, we summarize observations about NoSQL databases and give some predictions about the tendency of it and traditional RDBMS. The rest of paper is structured as follows: Section 2 we introduce basic concepts and characteristics of NoSQL and RDBMS, and a taxonomy for NoSQL database . Section 3 we presents a taxonomy for NoSQL techniques and discuss it in detail. Section 4 mentions related platforms that use NoSQL techniques, compares several main trend NoSQL database implementations and analyzes the limitation of these systems. In section 5 we summarize observations about NoSQL databases and give some predictions of the NoSQL and RDBSM.

2 Basic Concepts and Categories

2.1 Basic Concepts

Carlo Strozzi [15]first used the term NoSQL in 1998 as a name for his open source relational database without SQL interface, and in 2009 *Eric Evans* [8]reintroduced

this term with an event discussing open source distributed databases. Traditional database technology is a transactional processing characterized by *ACID* properties, those are atomicity (A), consistency (C), isolation (I) and durability (D). In practice, relational databases always have been fully and rigidly ACID compliant. By giving up parts of ACID, one would achieve much higher performance and scalability.

There is computer science theorem–Eric Brewer's CAP theorem [5] which including consistency (C), availability (A) and partitioning tolerance (P), that quantifies the inevitable trade-off. Consistency means all nodes see the same data in the same time; Availability means every operation terminates in an intended response; Partition tolerance means the system continues to operation even parts of it inaccessible. And according to the CAP theorem, a distributed system cannot satisfy all three of these properties simultaneously, but only two out of them. NoSQL databases generally give up consistency and use BASE, that is Basic Availability, Soft-state and Eventual consistency, as an alternative to ACID [12]. We could summarize the BASE properties in the following way: an application works on basically all the time (Basically Available), does not have to be consistent all the time (Soft-state) but will be in some state eventually(Eventual consistency).

2.2 Data Store Categories

There are many NoSQL databases, and some organizations such as Google and Amazon also invented NoSQL database technologies for their own necessities. Until now, NoSQL databases can be categorized according to their data store model into the following classes:

- Column stores: records are stored extensible that can be partitioned vertically and horizontally across nodes, such as Googles Bigtable, Amazon SimpleDB, Hadoop HBase.
- Distributed hash table key-value stores: in key-value systems, records are stored as values, which can be structured or completely unstructured, and a key uniquely identifies a value, such as Amazon Dynamo, Dynomite, Voldemort.
- Document-oriented stores: records in these systems look like semi-structured documents equipped by indexes, and systems use a query mechanism to find records, such as CouchDB, MongoDB and OrientDB.
- Graph database: these systems use graph structures with nodes, edges and properties to represent and store data, they are index-free since every element contains a direct pointer to its adjacent element. Neo4j, OrientDB and InfiniteGraph are some typical graph databases.

Popescu summarizes serval categories and gives a comparison of different NoSQL databases as table 1.

Table 1. Classifications C Categorization and Comparison [11]

	Performance	**Scalability**	**Flexibility**	**Complexity**	**Functionality**
Key-Value Stores	high	high	high	none	variable (none)
Column stores	high	high	moderate	low	minimal
Document stores	high	variable (high)	high	low	variable (low)
Graph databases	variable	variable	high	high	graph theory
Relational databases	variable	variable	low	moderate	relational algebra

2.3 General Characteristic

Different NoSQL database systems may have various techniques in detail, while they shared some common characteristic in general.

- No need schema: we can add data into a NoSQL database without defining a rigid schema. Moreover, the format of the data in NoSQL databases can be changed at anytime, and we need not to worry about any harm to databases. This brings tremendous flexibility to systems, and the web services can easily manage their users data [2].
- Distributed fault-tolerant architecture: data in NoSQL systems is stored in distributed servers, and the systems automatically spreads data across servers without requiring to change applications. Servers can be added to NoSQL system for scaling out, and error servers can be tolerated [2].
- Integrated caching: caching in relational databases usually must be deployed in separate infrastructures, it is inconvenient for developer to manage them and slow down the I/O speeds. Advanced NoSQL database technologies integrate the caching into system memory to reduce latency and increase sustained data throughout [2].

Next sections, we will cover four NoSQL categories of data stores, and compare these systems with each other in detail.

3 Column Stores

As discussed above, the basic data model of column store is that records could be partitioned vertically and horizontally, that is, data model is splitting both rows and columns over multiple nodes. Column stores databases allow information to be stored in multiple locations on disk, while data in row-stores usually is stored in a line row on disk. Even more, the same attributes of data are allowed to be stored together in column-stores, such as all of customers name could be stored together, all of Emails together, etc. For example, a database table contains information about customers (UserID, name, DateOfBirth, Email), in row-stores data are stored as table 2, compare to figure 3 which demonstrate data in column-stores.

Table 2. The data stored in row-stores

UserID	Name	DateOfBirth	Email
1	Jim	1992-04-23	Jim@msn.com
2	Tom	1986-05-17	Tom@msn.com

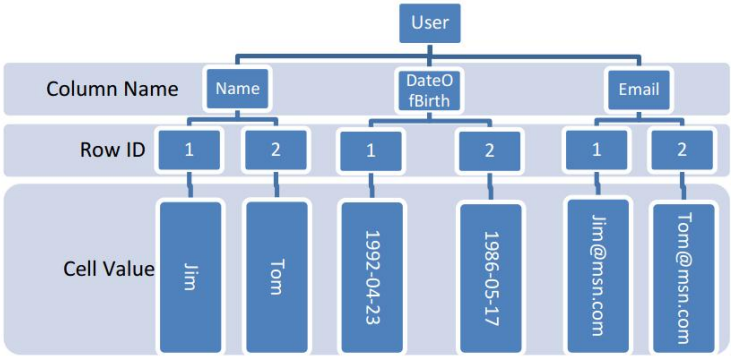


Fig. 1. The data stored in column-store

3.1 HBase

HBase is an open source, non-relational database model developed as part of Aapche Software Foundations Hadoop project. It runs on the top of the Hadoop Distributed File System(HDFS), and written in Java. It provides a fault-tolerant way for storing large data like Bigtable by using Hadoop MapReduce framework. HBase is a high performance, column-oriented, scalable distributed storage system, and HBase tables can be used as both input and output for MapReduce running in Hadoop. There are three keys in HBase table: row, time-stamp and column. The row key is the primary key, data in the table sorted by the row key. The timestamp presents the time of each operation, it shows the version of data in table. The column is used to storage the attributes of data, and supports for dynamic expansion. We can create a table without defining the number of column firstly. A typical example of HBase table as table 3.

4 Key-Value Stores

The key-value storage of databases may be the simplest data model in these models, it allow developers to store information without considering the schema any more. Information in key-value stores are not structured or semi-structured, they are treated as associated array of entries, which consists of a unique key and associated actual data. The unique key is to identify the entry, and the actual data is the value, they compose the key-value pair. The most advantage of key-value store is that it supports hundreds of thousands of concurrent queries, and

Table 3. A table in HBase

Row key	Row key	Column	
		Value	URL
1	2012-10-3,11:00	10001	url=www.ccc.com
	2012-9-28,17:00	10000	url=www.bbb.com
	2012-9-25,14:00	9999	url=www.aaa.com
2	2012-9-20,17:00	9998	url=www.222.com
	2012-8-5,14:00	9997	url=www.111.com

the retrievals are very fast [10]. Traditional relational databases only support for hundreds of concurrent queries at one time. However, the biggest limitation of key-value store is that it only permits to index data using the unique key, and complex conditional retrieval is not accessible [13].

Every version of data change is associated one vector clock, when data version conflicts happen, Dynamo reconcile the conflict according to the vector clock, earlier vector clock will be abandoned. In Dynamo, there are a set of configurable value to achieve desired performance, availability and durability. This set of parameters include N, R and W. N is present the number of replicas per partition determines the durability of each object. R is the minimum number of nodes that must participate in a successful read operation. W is the minimum number of nodes in a successful write operation.

4.1 Cassandra

Cassandra is a highly scalable, eventually consistent, structured key-value store. Cassandra combines the distributed systems technologies from Dynamo and the data model form Google's BigTable. The combination makes Cassandra available for providing a column-based data model richer than typical key-value systems. To some extent, Cassandra is a hybrid between a key-value and a column-orient database. Cassandra offers robust support for clusters, with asynchronous masterless replication allowing low latency operations for all clients. Cassandra could support multiple client API in various languages like Python, Ruby, PHP, but it lacks a transactional support. Cassandra was initially developed at Facebook, later it was released as open source project and became an Apache Incubator project.

4.2 Voldemort

Voldemort is a distributed data store used in LinkedIn, a famous social web site. It is a key-value storage system based on consistent hashing like Dynamo. Voldemort cluster can contain multiple nodes, and a physical node can run multiple virtual nodes like Dynamo. Voldemort also uses vector clocks for guaranteeing consistency. Voldemort supports various storage formats such as BerkeleyDB, Java Edition and MySQL. The main difference between Voldemort and Dynamo

is that Voldemort is a pluggable architecture. Every module in Voldemort shares the same code interface, and each module has different function. Developers can easily change and group these modules according to application needs.

5 Document-Oriented Store

Document-oriented databases could store and manage semi-structured data and information without tables. In these databases, schemas are not necessary and attributes are allowed to add or remove according to the needs of tuples. That means different tuples could contain different number of attributes without wasting space by creating empty fields for all tuples. Even if used in a structured way, there are no constraints or schemas limiting the database to conform to a preset structure [10]. Document is the central concept of document-oriented database. All of document-oriented databases assume that data are encapsulated in some standard formats or encodings as XML, YAML, JSON and BSON. No rigid schema makes it easy to create dynamic and flexible data, while the cost is the reduction in performance and safety. In document-oriented database documents are addressed by a unique key, usually is a simple string as a URL or path. Users can use the key to retrieve data from database.

5.1 SimpleDB

Amazon SimpleDB is a highly available and flexible non-relational database which is unbound by the rigid schema of relational database. SimpleDB could provide high availability and flexibility, manage distributed replicas of user data [14]. It allows developers to add new attributes that only apply to certain records rather than rebuilding table or indices. Even more, it allows a cell in a column to contain more than one value, because values are in individual cells. For example, the table 4 demonstrates a simple record in SimpleDB. If we want to add a new attribute Email to the table 4, we could directly create a new column in the table rather than rebuilding our schema or indices, even some value is empty as table 5.

Table 4. Before adding a new attribute into simpleDB

UserID	Name	DateOfBirth	State
1	Jim	1992-04-23	NY
2	Tom	1986-05-17	MA

Table 5. Add a new attribute into simpleDB

UserID	Name	DateOfBirth	State	Email
1	Jim	1992-04-23	NY	Jim@msn.com
2	Tom	1986-05-17	MA	

However, the SimpleDB must be accessed through the Amazon Web service platform, there are many limits to its usage and store size. Items are limited to 256 attributes, and every attribute are limited to 1024 bytes. The size of each domain item limits to 10GB and a query time limits to no more than 5 seconds [3].

5.2 MongoDB

MongoDB is a high performance, scalable, document-oriented database system. It stores structured data in blocks of JSON(JavaScript Object Notation) format, which called the BSON format. This format makes the integration of data easier and faster in applications. It is also a schema-free database like other NoSQL database system, which makes it easy to set up and scale out. The query language in MongoDB is very similar to SQL query language, it support complex query and ad-hoc query. MongoDB supports master-slave replication, load balancing and horizontally scaling.

5.3 CouchDB

CouchDB is the most famous open source non-relational database. It uses JSON to store data and defines document-oriented storage model. It does not store data and relationships in tables. Data is encapsulated in document format, and the database in CouchDB is regarded as a collection of independent documents. There is no unified schema in CouchDB, document maintains its own data and self-contained schema. CouchDB uses a mechanism of Multi-Version Concurrency Control (MVCC) to avoid locking the database file during writing new data into database. This method improves the input speed obviously, because conflicts are left to application to resolve. Another main feature of CouchDB is that it support ACID properties which is different from other document-oriented NoSQL database.

6 Graph Database

Graph database is designed to represent complex networks such as social network, public transport links. It is a type of database that uses graph structures to represent data. There are three core concepts in graph databases: nodes, relationships between nodes and key/value pairs. The nodes present the entities, relationships show the relationship between entities and the key/value pairs which attached to nodes and relationships show the properties. Compared with relational database, graph databases are often faster for associative data sets, and map more directly to the structure of object-orient applications. Graph databases can scale more naturally to large data sets without requiring expensive operations. We can take Figure 2 to show the graph database used in social networks:

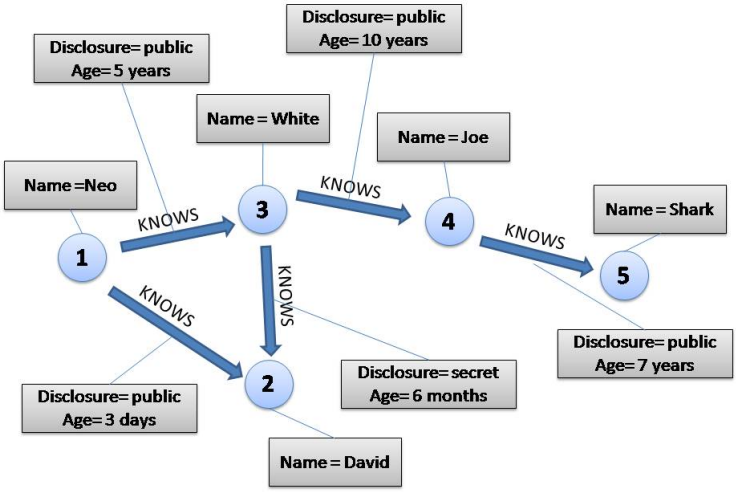
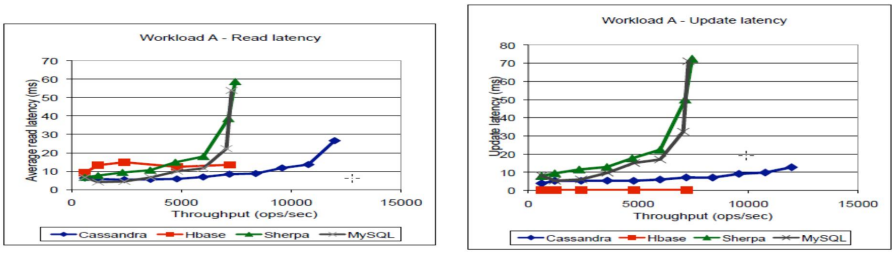


Fig. 2. The data store in graph database

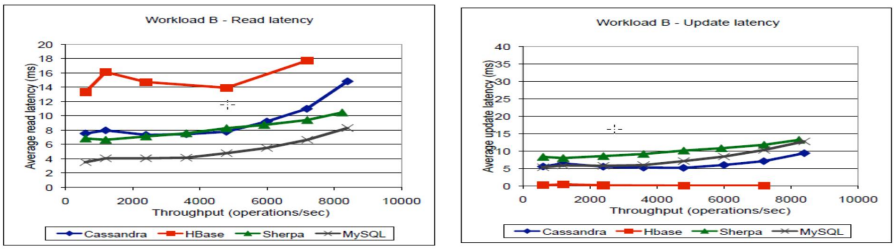
6.1 Neo4j

Neo4j is a high-performance, non-relational graph database, implemented in Java. According to Neo4j web site, Neo4j is an embedded, disk-based, fully transactional Java persistence engine that engine that stores data structured in graphs rather than in tables [1]. Three concepts are core in Neo4j: nodes(vertices), relationship(edges) and property(key/value pairs).



(a) Read latency **(b)** Update latency

Fig. 3. Performance in write intensive environment



(a) Read latency **(b)** Update latency

Fig. 4. Performance in read intensive environment

7 Performance Comparison of Several DB

The information used for comparison are mainly obtained from [7], included in the figures which are describing a laboratory based benchmark which uses YCSB (Yahoo! Cloud Serving Benchmark) [7] as a measurement tool. The benchmark was run on 120 million records of small size (1kB), 6 nodes.

7.1 Performance in Write Intensive Environment

Figure 3a and Figure 3b respectively show the read latency and the update latency in a write intensive environment [16]. We can see that over approximately 7000 read or update operations per second both MySQL and Sherpa (MySQL's variation) are becoming sharp, the latency time ascend very quickly and become unaffordable for real life application. The low latency time of HBase greatly attributes to its column-store, and it also uses memory to read and write which avoids directly read data from disk.

7.2 Performance in Read Intensive Environment

Figure 4a and Figure 4b respectively show the read latency and the update latency in a read intensive environment [16]. In read intensive environment, MySQL and its variation Sherpa show better results, keeping the pace with the NoSQL products. HBase still obtains a very good update performance, which perhaps attributes to write data into memory instead of disk.

8 Challenges and Predictions

As a new breed of database systems, the advantages of the NoSQL database has generated a lot of enthusiasm, but there are some challenges that it should be faced with. Maturity is the first problem. RDBMS have been used for a long time,

for most internet companies, the maturity of the RDBMS is reassuring because of its stability and security. However, NoSQL database technology is very young, it needs more test at wide range for attaining enterprises' trust. Secondly, enterprises expect their database systems to get credible technique support timely. Most of RDBMS vendors could provide a high level of enterprise support, in contrast, most NoSQL systems are open source projects, and cannot provide credible technique support like Oracle, Microsoft or IBM [9]. Finally, there are millions of developers throughout the world who are familiar with RDBMS concepts and SQL programming, in contrast, NoSQL is a new technology, the situation is that few NoSQL developers have the same rich experiments in NoSQL database as it in RDBMS. If NoSQL database want to replace RDBMS, there is still a long way to go [9]. Faced with so many opportunities and challenges, what is the future of NoSQL database ? Here are some predictions of NoSQL database over next few years:

- In order to gain scalability, availability and other advantages, globally-ACID transactions will be abandoned by many developers [6]. More and more electronic businesses will adopt NoSQL technology to meet the booming growth of users' demands.
- According to [4], enterprises are forming their own big data teams, and pay more attention and budget on managing, storing and analyzing big data, we could predict that in future new data schemas will be introduced in NoSQL, and more novel NoSQL technology will appear.
- New relational RDBMS will compete with NoSQL database and take a share of the market, if they could handle scalability, query transaction and high throughput [6].

9 Conclusions

The aim of this paper was to give a thorough overview and introduction to the NoSQL database. We have presented various NoSQL databases, which show the some differences as compared with traditional databases. According to the variances of data model, NoSQL databases are classified into four categories. We discuss each kind of NoSQL in detail, and give examples to illustrate their differences. We also analyse the challenges of NoSQL would meet and give some prediction about its future. Although NoSQL database technology shows outstanding performance in dealing with large data, horizontal scalability and high throughput, it hardly competes with relational databases that represent huge investments and mainly reliability and matured technology.

In the future, our research will focus on hybrid database systems which will combine NoSQL with relational database technology. Other databases, preserving ACID and fault-tolerant for cloud computing, are also worthy for wide attention.

References

1. Neo4j., http://neo4j.org
2. Nosql database technology, http://www.couchbase.com
3. Simpledb, http://aws.amazon.com/cn/simpledb/
4. Bantleman, J.: Rainstor makes top big data predictions for 2013, http://rainstor.com/rainstor-top-big-data-predictions-2013/
5. Brewer, E.: Cap twelve years later: How the "rules" have changed. Computer 45(2), 23–29 (2012)
6. Cattell, R.: Scalable sql and nosql data stores. ACM SIGMOD Record 39(4), 12–27 (2011)
7. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with ycsb. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154. ACM (2010)
8. Evans, E.: Nosql: Whats in a name? (October 2009), Blog post of October 30, 2009
9. Harrison, G.: 10 things you should know about nosql databases (2010), http://www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772
10. Lith, A., Mattsson, J.: Investigating storage solutions for large data. Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden (2010)
11. Popescu, A.: Presentation: Nosql at codemash – an interesting nosql categorization, http://nosql.mypopescu.com/post/396337069/presentation-nosql-codemash-an-interesting-nosql
12. Pritchett, D.: Base: An acid alternative. Queue 6(3), 48–55 (2008)
13. Seeger, M., S Ultra-Large-Sites.: Key-value stores: a practical overview. Computer Science and Media (2009)
14. Stanier, C.: Introducing nosql into the database curriculum. In: 10th International Workshop on the Teaching, Learning and Assessment of Databases, p. 61 (2012)
15. Strozzi, C.: Nosql-a relational database management system (2010), http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page (accessed)
16. Tudorica, B.G., Bucur, C.: A comparison between several nosql databases with comments and notes. In: 2011 10th Roedunet International Conference (RoEduNet), pp. 1–5. IEEE (2011)
17. Zikopoulos, P., Eaton, C., et al.: Understanding big data: Analytics for enterprise class hadoop and streaming data (2011)