Features and Application of Modern NoSQL Databases

Krasimir Slavyanov [1], Todor Beshlikov [1]

[1] "Vasil Levski" National Military University, 1 "Karel Shkorpiul" str., Shumen, Bulgaria,

Abstract. At present, there are sufficiently serious alternatives to relational systems in the face of RDBMS. Each NoSQL database has evolved around a different need and is being developed to satisfy it in close contact with its users. Despite the sharing of common principles, the individual NoSQL databases are technologically and architecturally diverse. Knowing them is important for making the right choice of what database to use for a new or growing software project.

Keywords: non-relational database, real-time web, big data.

1. NOSQL – RELATIONAL AND NONRELATIONAL SYSTEMS.

The Non-Structured Query Language or NoSQL provides a data storage and retrieval mechanism that uses a freely coordinated model as opposed to a more commonly used relational database. The benefits of this approach include clear design, horizontal scaling and fine control of available information.

The non-referenced database is the most well-optimized repository containing key-value information. Its purpose is to facilitate recovery and adding processes to optimize performance in unintentional system latency and excessive data input. This database has a significant role in real-time web and large data applications.

Non-relational systems are called "Not only SQL" in Translation "Not Just Database" to emphasize that they actually allow the use of search languages different but similar to those in SQL. NoSQL is a denial of the relational model, not so much of SQL as a language. For this, RDBMS (Relational Database Management Systems) need to be briefly reviewed to help clarify the cause of the emergence of NoSQL from their drawbacks (Fowler & Sadalage, 2014).

The main obstacle to the relational model is problems such as flexible scalability, engagement of applications using the relational model with additional logic, cache problems, adaptability and alteration options as well as disruption of the relational model as an optimization method.

It is these disadvantages that are the reason for the emergence of NoSQL. NoSQL is interpreted as "not just SQL" and in no way rejects the relational model, rather it offers a different look at the data storage methods and solves the most common problems in the relational model.

The basic construction in the relational model is the relationship. The relationship is a two-dimensional table in which data is placed and is based on the mathematical concept of relationship. Attributes serve as column names.

The name of the relationship and its set of attributes form the relationship scheme. The multiple of all relationship schemas in a database is the database schema.

If A and B are sets, the relation is a subset of the work of AxB (Serra, 2015). The schema of a particular database built with the relational model is constructed by applying normalization rules. Normal forms aim to store data from anomalies when adding, updating, and reading information.

2. STORAGE AND DATA PRESENTATION MODELS

Document model. Each key is associated with a complex data structure known as a document. Documents contain one or more fields, each field containing a value (eg string, date, array, etc.). Instead of assigning records to multiple columns and tables in relational databases, here every record and its associated data are stored together in a single document. This simplifies access to data and reduces the need for related and complex transactions. Each document may contain different fields, which is very useful in modeling unstructured data. It also facilitates the development of applications, for example by adding new fields. An example of such a database is Lotus Domino (Bankes & Hatter, 1999).

Graphic model. It uses column structures with nodes, shoulders, and properties to represent the data. In particular, data is modeled as a network of links between particular elements. This is especially useful for social networks.

Key-value model. These are the simplest NoSQL databases. Everything in the database is stored as a defining name (or "key") along with its value. The value is invisible to the system - data can only be searched for by the key. This model may be useful for unstructured data.

The CAP Theorem. The relational model is the most widespread at the moment and millions of applications work with it. However, as has been pointed out in highlighting its shortcomings, problems are increasingly encountered mainly with the increase in data and load. When it comes to distributed systems - clustering of the DB, the concept of the CAP theorem is introduced (Figure 1). The CAP theorem states that every distributed BS system is based on three main pillars - data consistency, data availability, scalability. The CAP theorem proves that only two of these three pillars can be used to create such a system. We can have a system with high consistency and expandability, a system with high data availability and expandability, or a system with high consistency and high availability but cannot be expanded.

3. CUSTOMER CONSISTENCY.

Client consistency has to deal with how the individual processes communicating with the system see the updates. If Process A makes an update, we have the following types of consistency:

- Strong consistency. Upon completion of the upgrade, each customer will see the updated value;
- Low consistency. The system does not guarantee that prospective customers will receive the revalued value unless certain conditions are met. The period until the conditions are fulfilled is a period of inconsistency;
- Possible consistency. This is a form of weak consistency whereby the system ensures that if no new updates are made to the data, eventually all access points will return the correct value.

Unrelated databases can not necessarily give full assurance of ACID (Atomicity, Consistency, Isolation, Durability). Normally, consistency is guaranteed or transactions are limited to one information object. This means that if a sufficiently long period of time is given during which no changes are sent, updates may be expected to spread to the system.

Any consistency in MongoDB. MongoDB defines several types of consistency:

Single Writer Eventual Consistency - eventual consistency with one source of change. With 1 master and multiple slaves, this is a situation where the customer can read obsolete information or get misdirected changes.

Monotonic Read Consistency - a possible consistency that cannot be changed in the wrong order.

Read-your-own-writes Consistency - the possible consistency in which the customer accesses their own updates. In a web app, this may mean the user, not the client. Using a load balancer can easily be compromised.

Depending on the type of compromise with the consistency that may be allowed in the application, the MongoDB configuration must be ensured.

4. MAP / REDUCE.

This is a method of retrieving information. Includes 2 functions that apply to a map and reduce list. The map function returns a processed result of 0, 1 or more records for each entry in the list, while reducing aggregates the list in 1 value. Map / reduce can work distributed, in many processes, simultaneously on many machines, breaks the information into pieces, summarizes them, and summarizes them again.

Map / Reduce is a method for displaying views and processed, aggregated information, on a regular basis, through one-time writing and multiple reading later.

It is supported by almost all NoSQL databases. Typical map and reduce features are JavaScript, but some databases support multiple languages. Regardless of the language in which they are written, map functions contain emit - a new entry is added to the result. The reason for not using return is that for 1 incoming entry we may have several outgoing ones.

Both document and column databases use map / reduce. However, some key / value pairs, such as Redis, do not offer this option.

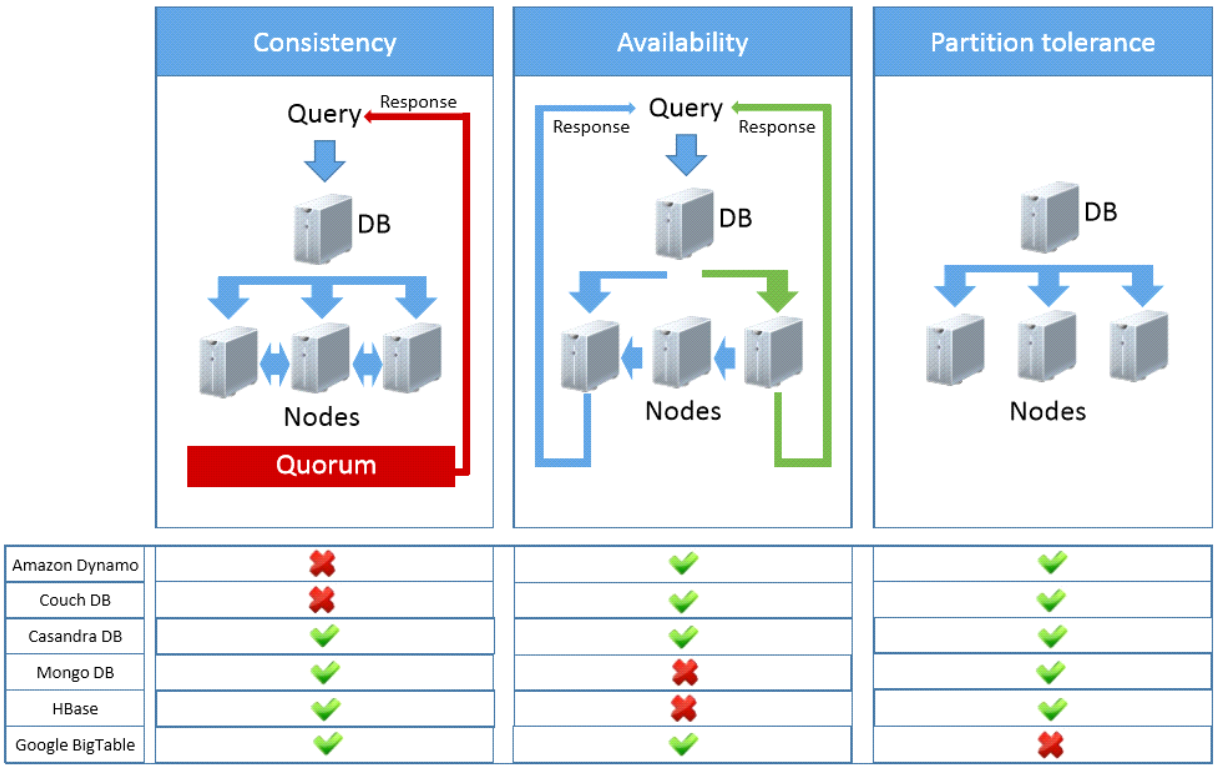

Fig.1 Concept of the CAP theorem

5. NETWORK FILE SYSTEMS.

NoSQL databases have extremely diverse approaches to data storage. Some databases, such as Google Cloud BigTable, use dedicated file systems to allocate, others run in a complete software package - like HBase + Hadoop - a database and a separate storage engine. In the third file system it does not matter because the data base is shared by the NoSQL database itself (MongoDB, CouchDB, Cassandra, Riak, etc.)

Many NoSQL database technologies have excellent integrated caching capabilities, keeping the data used in system memory as

often as possible and eliminating the need for a separate caching layer that needs to be maintained.

The different vendors of relational databases have taken different approaches to realizing caching:

MySQL caches the most recent requests to the server, making sure that the query is no longer available in the cache after submitting the request. If the response is returned directly from the cache, if the request is not cached running on the server, the returned result is cached and the client receives the response. MSSQL / Oracle - MSSQL does not cache the most recently submitted queries, and the most recently read blocks of memory, this is called the data page. When a request is made to the server, it first checks for a cache result, if it is missing, the request is sent to the server. Prior to execution on the server, the query falls into the so-called Query plans, which takes care of optimizing queries to increase performance, then the query is executed and the result is returned to the client. In MSSQL, this becomes transparent from the point of view of administrators and they have no control over the process. In Oracle, things are similar in the way that the controller has control over the process, can determine the size of the buffers, the cache, and so on.

One of the main cache problems is so. cold start. When we have a server cached with months of large volumes of data and he reboots he loses his cache. Upon restoration of the system, each request submitted to the server begins to cache again. With large volumes of data, this results in a huge peak load on the server. This is where the application finds the first NoSQL database. They take care to export the cache and store it in optimized BD so after the recovery of the machine the cache is recovered from the BD and the losses in it are insignificant. Examples of such databases are MemCache and Redis.

Ability to adapt and change - this flaw is detected by Twitter, then WordPress is testing it and confirming it. Twitter initially pledges to MySQL as a database for its platform. After increasing the volume of data and reaching several TV channels, it is found that the process of modifying the scheme is either impossible or extremely slow. For this reason, Twitter decided to migrate from MySQL to NoSQL - Cassandra. In the migration process, however, they face the effect that they can not add additional columns to the tables, because adding each new column leads to an increase in large data. In NoSQL DB this problem does not exist because of the fact that they do not have a schematic diagram, the changes are dynamic and not structured.

Violation of the relational model as a method of optimization - this phenomenon also faces Twitter. They notice that for large volumes of data, SELECT type queries go relatively fast, but queries like JOIN are several dozen times slower. To quickly find an effective solution they begin to duplicate the same information in multiple tables as the idea was to run SELECT on only one table to increase productivity (Zaitsev, 2010).

Productivity increases at the expense of duplicate data, which violates the normal form of data and the relational model as a whole. Twitter has experienced extremely severe data deformation, as data stored in the system has been several thousand times more than what is actually needed.

6. ORACLE NOSQL DATABASE

Oracle is the first vendor to offer a complete and integrated solution to address the full spectrum of enterprise big data requirements. Oracle's big data strategy is centered on the idea that you can extend your current enterprise information architecture to incorporate big data. New big data technologies, such as Hadoop and Oracle NoSQL database, run alongside your Oracle data warehouse to deliver business value and address your big data requirements.

Oracle NoSQL Database is a distributed, highly scalable, key-value database based on Oracle Berkeley DB. It delivers a general purpose, enterprise class key value store

adding an intelligent driver on top of distributed Berkeley DB. This intelligent driver keeps track of the underlying storage topology, shards the data and knows where data can be placed with the lowest latency. Unlike competitive solutions, Oracle NoSQL Database is easy to install, configure and manage, supports a broad set of workloads, and delivers enterprise-class reliability backed by enterpriseclass Oracle support (Kyte & Kuhn, 2014; www.oracle.com).

The primary use cases for Oracle NoSQL Database are low latency data capture and fast querying of that data, typically by key lookup. Oracle NoSQL Database comes with an easy to use Java API and a management framework. The product is available in both an open source community edition and in a priced enterprise edition for large distributed data centers. The former version is installed as part of the Big Data Appliance integrated software.

7. HADOOP ARCHITECTURE

Hadoop is an open-source Apache project started in 2005 by engineers at Yahoo, based on Google's earlier research papers. Hadoop then consisted of a distributed file system, called HDFS, and a data processing and execution model called MapReduce. The base Apache Hadoop framework consists of the following core modules:

Hadoop Common: The common utilities that support the other Hadoop modules; Hadoop Distributed File System (HDFS): A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster; Hadoop YARN: A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications; Hadoop MapReduce: A programming model for large scale data processing.

In addition to these base modules, the term 'Hadoop' has evolved to also include a dozens of other independent tools and projects that can be installed on top of or alongside Hadoop to simplify access and processing of data stored in the Hadoop cluster:

Ambari: GUI for managing and monitoring Hadoop clusters; Hive: Data warehouse infrastructure providing SQL-like access to data; Pig: Scripting language for accessing and transforming data; Sqoop: Managing data movement between relational databases and Hadoop; Flume: Service for collecting data from log files into HDFS; Mahout: Machine learning library; Tez: Data-flow programming framework, built on YARN, for batch processing and interactive queries. Used increasingly to replace MapReduce for Hive and Pig jobs; Spark: In-memory cluster computing framework used for fast batch processing, event streaming and interactive queries. Another potential successor to MapReduce, but not tied to Hadoop. Spark is able to use almost any filesystem or database for persistence. Zookeeper: A high-performance coordination service for distributed applications (Khudairi, 2011).

Applications submit work to Hadoop as jobs. Jobs are submitted to a Master Node in the Hadoop cluster, to a centralized process called the JobTracker. One notable aspect of Hadoop's design is that processing is moved to the data rather than data being moved to the processing. Accordingly, the JobTracker compiles jobs into parallel tasks that are distributed across the copies of data stored in HDFS.

The JobTracker maintains the state of tasks and coordinates the result of the job from across the nodes in the cluster.

Hadoop determines how best to distribute work across resources in the cluster, and how to deal with potential failures in system components should they arise. A natural property of the system is that work tends to be uniformly distributed – Hadoop maintains multiple copies of the data on different nodes, and each copy of the data requests work to perform based on its own availability to perform tasks. Copies with more capacity tend to request more work to perform.

8. MONGO DB

Traditional Relational Database Management Systems (RDBMS) are modeled around schemas and tables to organize and structure data in a combination of columns and rows. Most current systems are RDBMS, and it is probably going to stay that way for the foreseeable future. For many companies, RDBMS solutions are suitable, but not necessarily appropriate for every use case. These systems often run into bottlenecks with scalability and data replication when handling large amounts of data/data sets (http://www.slideshare.net/mongodb/replica-sets; www.mongodb.com).

As a document-oriented database management system, MongoDB stores data in collections, in which different data fields can be queried once, versus multiple queries required by RDBMS' that allocate data across multiple tables in columns and rows. The data is stored as Binary JSON (BSON), and is readily available for ad-hoc queries, indexing, replication, and MapReduced aggregation. Database Sharding can also be applied to allow distribution across multiple systems for horizontal scalability as needed. MongoDB is written in C++, and can be deployed on a Windows or Linux machine, but especially considering MongoDB for real-time low-latency projects, Linux is an ideal choice for the sake of efficiency. A primary difference between MongoDB and Hadoop is that MongDB is actually a database, while Hadoop is a collection of different software components that create a data processing framework.

9. COMMENTS AND CONCLUSIONS

NoSQL databases are designed to solve specific problems, with the motivation of the creators of different databases being different and as a result each of them is optimal for use in different cases. RDBMS are universal, but their versatility comes at a high price - in hardware, electricity and licenses. This motivates the development of NoSQL databases on which many of the world's largest Web services operate.

Oracle and MS SQL Server are not an alternative to large-scale projects due to their high cost, exceeding $ 50,000 per server. MySQL is used for busy projects like Google Adsense, but in order to get it running, it usually comes down to a key / value store and uses the poor of MyISAM storage engine capabilities.

NoSQL DB make it possible to have Web services that could not otherwise exist. Hadoop and Cassandra move Facebook, Twitter, separate Yahoo services. CouchDB stands behind bbc.co.uk. SourceForge and Foursquare use MongoDB. Yahoo's main services are Sherpa, Google - BigTable, Amazon - Dynamo, and all of them also use other NoSQL solutions in certain cases.

The moment the largest services in the world have gone from RDBMS to NoSQL may have already occurred.

In the practical part, it has been demonstrated that despite their different architecture, NoSQL databases are usable and capable of delivering a significant improvement in performance - whether they are used for a whole project or for some part of it. There is something to be done about libraries, a wealth of functionality and ease of work, but the obstacles are overwhelming and the tasks can be accomplished.

REFERENCES

Bankes, T., D. Hatter, 1999. Lotus Notes and Domino Essential Reference

Fowler, M., P. J. Sadalage, 2014. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence

Khudairi, S., 2011. The Apache Software Foundation announces Apache™ Cassandra™

Kyte, T., D. Kuhn, 2014.Expert Oracle Database Architecture, 3rd Edition

Serra, J., 2015. Relational databases vs Non-relational databases

Zaitsev, P., 2010. Storing MySQL Binary logs on NFS Volume