NoSQL Databases

(Paper for Advanced Seminar "Computer Engineering")

Matthias Hauck*
* Email: matthias.hauck@stud.uni-heidelberg.de

***Abstract*—This paper will give a short overview about NoSQL databases. Therefor a short overview about the traditional approach as well as the new requirements for NoSQL database will be given. Furthermore the different types of NoSQL databases and basic technologies they use will be presented. Finally some critic on NoSQL will be mentioned.**

I. Introduction

In the last decade new classes of applications arise for example web applications or applications for mobile devices. These applications have new requirements for there backend storage that are different from the requirements of previous generations.
Mobile devices have in many cases not a guaranteed permanent connection to a central server. Therefore these applications have often a local replication of the data that have to be synchronized with the central storage. Another problem applications on simple mobile devices has to struggle are there limited resources. Java ME for example has therefor a simple record store as persistent storage.
Web applications and other BigData applications have problems on a completely other scale. Web applications need databases that scale. Applications like a web search or a service for short messages like twitter can have data of an enormous size that still grows. In addition the database have to be very fast to handle requests from millions of users. This requirements can typically only be fulfilled by a cluster.
The availability of a web applications has to be very high. If an application is not available users can probability go to a competitor. Web applications need to evolve in order not to lose user, too. But even in the case of necessary changes on the database schema, a downtime is not acceptable.
In some cases the availability of an application is even more important than consistency. If a recommendation service of a web shop has not the latest updates of the actions of a user, it does not matter. In general the value of consistency be very differently depending on the application.
For web application another important requirement is that there costs are low, because they are often financed by ads. Because of this many companies try to avoid expensive high end hardware and use standard hardware in a cluster. To achieve high reliability replication is used. They also use often open source software to avoid expensive license fees. To eliminate administration costs many administration tasks, like to integrate a new computer in a cluster, has to be automated.
The main source of the following paper is the paper NoSQL Databases[1] by Walter Kriha(Stuttgart Media University).

II. SQL RDBMS

To understand NoSQL it is first important to understand SQL databases. Traditional SQL databases are RDBMS(Relational Database Managements Systems) and support therefor the relational data model. In this model data is organized in tuples that are grouped in a relation. In the databases these relations are realized as tables. Different tupels can be combined by a join. The fix relation schema behind the relations is typically normalized in such a way that it is redundancy free.
The roots of RDBMS are in the 1970th. At this time IBM created the very exemplary database "System R" that introduced the query language SEQUEL that address relational Databases. SEQUEL is the direct ancestor of SQL. Today RDBMS are very mature.
For a very long time computer were very expensive, but administrators were relatively cheap. Therefore the RDBMS databases have not to scale across a huge number of computers and in addition there was no real need to reduce administration costs. Relational databases have a good performance, because of the strict schema they use. Often they are row oriented, because this simplifies the very common append operations.
RDBMS are typically focused on the use as a part of business applications or applications that are used by public authorities. These applications are for example financial services like the transfer of money or the storage of water well status data. Very often these applications need online transaction processing(OLTP)[1].
What all of these applications have in common, is that there data is very valuable. These data have to be well protected and the databases have to be robust. This is achieved through the ACID(Atomicity, Consistency, Isolation, and Durability) consistency model.

III. NoSQL Databases

NoSQL databases try to address different problems of classic RDBMS databases. There are very different types of NoSQL databases with a lot of different feature sets that solve different problems. But there are things that many NoSQL databases have in common or that is at least very similar:
NoSQL database omit a strict relational data model. This is typically what makes a NoSQL database to a NoSQL database. They try to group data that are used together instead. In

[1]Another important class of applications need online analytic processing(OLAP). These applications need typically special databases.

addition the schemes are flexible.
As a query interface NoSQL databases have an interface that is not SQL based. Some of these databases have a query language that is similar to SQL, because this should make the use of this interface easier. For complex queries many NoSQL databases have an internal support for MapReduce or an interface to an external MapReduce suit like Hadoop.
The query interface is often very simple. Expensive operation like transactions across multiple entries and joins are often not possible. Many NoSQL databases are distributed. They have to handle the data replication and distribution. Some databases even do not hide details of this.
Another thing that many NoSQL databases have in common is that they have been developed by a company that run large web applications. Many of these companies have published them as Open Source.

A. Types of NoSQL Databases

There are different types of NoSQL-databases. A common method to classify them is by there data model. A very often they are classified as: **key/value store**, **document store**, **graph store** and **column oriented store**.
Key/Value stores are the most simple types of NoSQL databases. These databases store a value that can be access only by the key of the value. These databases are often optimized for high performance and scalability in a cluster. In the context of web applications they store objects of variable size like pictures or serialized objects(e.g. session data). The use of a key/value store and serialization eliminate the need of an expensive object relation mapper which are used to store objects in RDBMS.
Document stores use a more complex data model called document. Their model is not strict and allow a few (dynamic defined) properties per entry. The idea behind this approach is that with a non-strict schema there is no long downtime in the case of necessary schema changes.
In addition values of entries might be complex and often nested documents are supported. Together with the non-strict schema, data that can be used together can be grouped together. There is no need for joins and the data can easily get partitioned.
A **graph store** stores data as nodes and labeled edges. Therefore it is well suited for task were data appears in such a way like in social networks. Typically they need less space and have a higher performance then RDMS that handles graphs as relations. In addition these databases have special algorithms like for tree traversal to handle graphs.
One drawback of these databases is that they can not scale very well. The problem is that graphs are hard to partition, because the graph has to be divided. Relations and nodes that reside on different nodes, but are important for one operation, would cause a lot of network traffic. The later mentioned technique of sharding is because of this often not supported. For more informations about graph databases [2] is a good start.
Column-oriented stores[2] are a type of database that is influenced by Google's BigTable. In the paper [3] by Google BigTable is described as "A Bigtable is a sparse, distributed, persistent multidimensional sorted map." They are suitable for a broad rage of applications[3].
Column-oriented stores use a table to store data. Typically the row ranges of the table are often partitioned for distribution. Each row has a variable number of columns.
The columns of a row are each member of a column-family. Many similar columns of the same row may be member of the same column-family. This allows to optimize the access on the columns, because for an operation on one type of column it is not necessary to load the complete rows. In addition it is often possible to have different version of an entry.

IV. Technologies

To achieve the specific goals like high scalability and high performance NoSQL databases have to solve different problems. This section will give an overview of some important techniques that solve these problems and help to achieve the goals.

A. CAP-Theorem and BASE

A fundamental property of many NoSQL databases is that they can be used in a cluster. A developer that use such a distributed database desires three different guaranties:
Consistency, which means that the data in the cluster look like a single copy; **Availability**, which means that the database response to requests even if some nodes have an outage; (network) **Partition-tolerance**
The bad news is that these three guarantees can not be given all at once. Only two of them can be guaranteed at any time. This finding is known as CAP-Theorem. The CAP-Theorem was formalized 2000 by Eric Brewer and later formal proofed[4] by Seth Gilbert and Nancy Lynch.
Classic distributed RDBMS use ACID and therefor chose consistency and partition-tolerance. As mentioned in the introduction for many applications' availability is much more important than consistency. For such an application a database that choose Partition-tolerance and Availability would be much more useful. Many NoSQL databases have chosen such an approach. They omit a strict consistency model and the relaxed consistency model BASE.
BASE stands for Basically available, Soft-state, Eventual consistency. Basically available means in this context that the database has to run all the time. Soft-state means that the data over all nodes in a cluster have not to be in a strict consistent state all the time. Different variants of the data are possible in the cluster.
Eventual consistency means that the application does not have to be intermediately in a consistent state, but after a finite time. There are different types of eventual consistency. These

[2]The name of this type of database is a bit misleading, because it seems to be identical to a relational column store.
[3]Google use BigTable for example for the Websearch, Google Earth or Google Analytics. For more information see [3].

different types are differed, which updates an application could read. The problem of eventual consistency is that applications must tolerant possible inconsistency.
In his blog Eric Brewer remark 2012[5] that the CAP-Theorem is only relevant in the case of an error, so that there is more than one Partition. Therefor he suggests only to handle these errors explicit. Possible actions are for example either to limit some operations or to store some extra data to recover after the partition. Today not many databases do this.

B. Sharding

One important aspect to ensure scalability is the data partitioning. NoSQL workloads can get so huge that they can easily exceed any single computer, so they have to be distributed across the nodes of a cluster. Such a horizontal distribution is call sharding.
Classical RDBMS can not do sharding or not very well. The problem is that there data is often divided in different tables according to the relational data model. These tables get on the other hand often connected for a query by a join. If the data is on different nodes joins are not possible or would require a lot network traffic. RDBMS systems that allow sharding require often the participation of an administrator.
To reduce administration costs many NoSQL databases allow an automatically sharding of the data. The automatically sharding includes that the database maintains some kind of mechanism to find the data in the cluster. Popular methods to do this are hashing or a lookup server. If there are new computers for the cluster they have also to be integrated automatic, too. The database has therefore to do some kind of membership management.
Another important feature that some sharding mechanisms of NoSQL databases support is automatic load balancing. It monitors the data usage and try to avoid usage hotspots by distributing often used data across the nodes. That load balancing is important shows Twitter[6].
They have used for the sharding of the tweets a static approach based on the time stamp of the tweets, so that tweets with a similar date are stored on the same machine. The problem of this approach is that only the server with the newest tweets have to struggle with the full load, because nobody is interested in old tweets.

C. Performance optimization

NoSQL databases have often to answer a great amount of queries. Therefore they have to be fast.
Often the performance optimization is done by the remove of overhead. Like mentioned in [7] the main sources of overhead in a database are the communication of the client and the database, logging, locking, latching[4] and the buffer management. The elimination of such a overhead source can dramatic improve the performance. This mean in many cases that a feature has to be removed, but sometimes this is not important.
The communication between client and database can be reduced by pushing server code to the client library. Some NoSQL databases are integrated in a MapReduce framework stack like BigTable in Google's MapReduce or Hbase in Hadoop. This allow often to exploit the locality. Some NoSQL databases have something similar to the stored procedures of RDBMS.
Logging can in many cases simply be omitted, when data is not valuable. In some cases replication across several nodes has a sufficient level of reliability, so there is no need for logging. Locking is not necessary, if there are no transactions over more than one entry. Databases that are completely in the main memory need no buffer management. In addition this type of database benefits from the fact that access to the main memory is much faster than any disk.
In addition there are some other performance optimizations like compression to virtual increase bandwidths and memory space. Some times the optimization is not in the database but rather in the application. An object deserilisation of a binary storage format is often cheaper than an object relation mapper.

[4]Locking of a shared data structure in order to modify it.

V. Critique

There are a few critique points on NoSQL. In this section an overview about some of them will be given. One critic point is that NoSQL is nothing new. There are databases that do similar things as NoSQL database long before the phrase NoSQL established. These are for example Lotus Notes or classical object databases. In addition it is often mentioned that some enterprise databases support different features that are often associated as NoSQL features. Typically these databases are very expensive.
Another point is that NoSQL is not standardized. Many NoSQL databases orient themselves for there query language on SQL to reduce this problem. In addition many applications do not use SQL directly. Instead they use persistence frame works like hibernate. If such a framework supports a database, it is not important which query language the database use. SQL RDBMS are in some cases not standardized, too. Many SQL databases support stored procedures. The language of these procedures is not standardized.
A critique point often by business company's is that there is no commercial support. This point is not right. In the case that the original developer is a web company, it is often right that there is no support through him. But like many other open-source software there is support through other company. Software companies like Oracle(BerkeleyDB) that have developed a NoSQL database offer typically commercial support.
The last point that will be mentioned here is that some concepts like BASE make it hard for programmer to develop software[8]. This is quite and one of the reasons for the development of the next generation of database. Databases like Google F1 use NoSQL features to be very scalable, but use also use a strict consistency model and SQL to support the demands of classical business application. In addition they are a lot easier to program.

VI. Conclusion

With NoSQL databases there is a solution for problems that have to scale. They are optimized for an application area and do there job in this area well. It could be supposed that they get in the next years more mature and further evolve.
But as previous mentioned there will be in the near future a new generation of databases. These New SQL databases will use NoSQL to scale and strong consistency and SQL to support a broader range of applications and to be easy to use. NoSQL databases will not die, because they are relatively simple and can in this way optimized for special applications.

References

[1] W. Kriha, "Nosql databases," in *Selected Topics on Software-Technology Ultra-Large Scale Sites*.

[2] T. Schürmann, "Fünf graphdatenbanken im vergleich," 2012. [Online]. Available: http://www.linux-magazin.de/Ausgaben/2012/04/Graphdaten-banken/%28language%29/ger-DE

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," 2006.

[4] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," 2002.

[5] E. Brewer. (2012) Cap twelve years later: How the "rules" have changed. [Online]. Available: http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

[6] highscalability.com. (2011) How twitter stores 250 million tweets a day using mysql. [Online]. Available: http://highscalability.com/blog/2011/12/19/how-twitter-stores-250-million-tweets-a-day-using-mysql.html

[7] M. Stonebraker. (2009) The "nosql" discussion has nothing to do with sql. [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/50678-the-nosql-discussion-has-nothing-to-do-with-sql/fulltext

[8] J. Shute, M. Oancea, S. Ellner, B. Handy, E. Rollins, B. Samwel, R. Vingralek, C. Whipkey, X. Chen, B. Jegerlehner, K. Littlefield, and P. Tong, "F1 - the fault-tolerant distributed rdbms supporting google's ad business," Google, Tech. Rep., 2012.