A comparison between several NoSQL databases with comments and notes

Bogdan George Tudorica, Cristian Bucur
Department for Economical Mathematics and Economical Informatics
Petroleum-Gas University of Ploiesti
Ploiesti, Romania
tudorica_bogdan@yahoo.com

Abstract—**This paper is trying to comment on the various NoSQL (Not only Structured Query Language) systems and to make a comparison (using multiple criteria) between them. The NoSQL databases were created as a mean to offer high performance (both in terms of speed and size) and high availability at the price of loosing the ACID (Atomic, Consistent, Isolated, Durable) trait of the traditional databases in exchange with keeping a weaker BASE (Basic Availability, Soft state, Eventual consistency) feature. Remains to be seen which of the multiple solutions created since the official appearance of the NoSQL concept (which was defined in 1998 and reintroduced in 2009, around which moment several NoSQL solutions emerged; at the present moment there are known over 120 such solutions) are really delivering on these promises of higher performance (although several of them are already used with very good results).**

Keywords-component; database; NoSQL; performance; comparison

I. INTRODUCTION

The concept described by the term NoSQL (meaning a database system which is distributed, may not require fixed table schemas, usually avoids join operations, typically scales horizontally, does not expose a SQL interface and may be open source [1] – some are even using the term with the meaning of a completely non relational system) is also referred by the more academic sources as a form of structured storage [4][10][11][12] (although the terms may not be equivalent; the relational databases also comply by the official definition of the structured storage term and they are somehow opposite to the NoSQL term).

One can not simply label the terms RDBMS and NoSQL as being the exact opposite. There do even exist some middleware appliances (such as CloudTPS for Google’s BigTable and Amazon’s SimpleDB [17]) or various solutions (such as Percolator for Google’s BigTable [14] and an unnamed prototype system for Google’s Hbase [7]) which are adding full ACID features to some NoSQL systems.

It is certain that the NoSQL databases are one of the byproducts of the Web 2.0 era – they were really used only at the time when the designers of web services with very large number of users discovered that the traditional relational database management systems (RDBMS) are fit either for small but frequent read/write transactions or for large batch transactions with rare write accesses, and not for heavy read/write workloads (which is often the case for these large scale web services – we mean Google, Amazon, Facebook, Yahoo and such).

It seems that at least some of the major RDBMS producers are learning something from this evolution (e.g. Microsoft introduced some NoSQL type features such as snapshot isolation, although used at a single table level, into its newer RDBMS product labeled Azure; Oracle 11g is also containing a similar facility called Oracle Streams, but this one is limited in the same way as the MS product, this time to a single instance [7]).

II. WHAT DO WE COMPARE

In order to be able to compare a set of NoSQL solutions the first step should be to select / classify some products which are fulfilling similar purposes or have similar qualities / features.

For the moment there is no official taxonomy for this kind of software although several attempts do exist.

First one is provided by Stefan Edlich on his page [8] and it is providing the following categories:

A. Core NoSQL Systems, most of them created as component systems for Web 2.0 services, with the following subtypes:

- Wide Column Store / Column Families (Hadoop / HBase, Cassandra, Hypertable, Cloudata, Amazon SimpleDB, SciDB),
- Document Store (CouchDB, MongoDB, Terrastore, ThruDB, OrientDB, RavenDB, Citrusleaf, SisoDB, CloudKit, Perservere, Jackrabbit),
- Key Value / Tuple Store (Azure Table Storage, MEMBASE, Riak, Redis, Chordless, GenieDB, Scalaris, Tokyo Cabinet / Tyrant, GT.M, Keyspace, Berkeley DB, MemcacheDB, HamsterDB, Faircom C-Tree, Mnesia, LightCloud, Pincaster, Hibari, Scality),
- Eventually Consistent Key Value Store (Amazon Dynamo, Voldemort, Dynomite, KAI, SubRecord, Mo8onDb, Dovetaildb),

- Graph Databases (Neo4J, Infinite Graph, Sones, InfoGrid, HyperGraphDB, Trinity, AllegroGraph, Bigdata, DEX, OpenLink Virtuoso, VertexDB, FlockDB, Java Universal Network / Graph Framework, Sesame, Filament, OWLim, NetworkX, iGraph),

B. Soft NoSQL Systems, most of them being older or newer systems which are not related to any Web 2.0 service but are sharing the traits being described as NoSQL characteristics (A/N: some of them are having strong ACID / relational capabilities and, from this reason, they may be misplaced in a list of NoSQL systems; further analysis may be needed on this subject), with the following subtypes:

- Object Databases (db4o, Versant, Objectivity, Gemstone, Progress, Starcounter, Perst, ZODB, NEO, PicoLisp, Sterling, StupidDB, KiokuDB, Durus),
- Grid & Cloud Database Solutions (GigaSpaces, Queplix, Hazelcast, Joafip, GridGain, Infinispan, Coherence, eXtremeScale),
- XML Databases (Mark Logic Server, EMC Documentum xDB, Tamino, eXist, Sedna, BaseX, Xindice, Qizx, Berkeley DB XML),
- Multivalue Databases (U2, OpenInsight, OpenQM, Globals),
- other NoSQL related databases (IBM Lotus/Domino, Intersystems Cache, eXtremeDB, ISIS Family, Prevayler, Yserial).

Another taxonomy is provided by an unknown author on an wiki page [23] and provides the following categories of NoSQL databases:

- Document store (Apache Jackrabbit, Apache CouchDB, Lotus Notes, MongoDB, MarkLogic Server, eXist, SimpleDB, Terrastore),
- Graph (AllegroGraph, Neo4j, DEX, FlockDB),
- Key-value store, with the following subtypes: Eventually - consistent key - value store (Cassandra, Dynamo, Hibari, Project Voldemort, Riak), Hierarchical key-value store (GT.M), Hosted services (Freebase), Key-value cache in RAM (Citrusleaf database, memcached, Oracle Coherence, Redis, Tuple space, Velocity), Key-value stores implementing the Paxos algorithm (Keyspace), Key-value stores on disk (BigTable, CDB, Citrusleaf database, Dynomite, Keyspace, membase, MemcacheDB, Redis, Tokyo Cabinet, TreapDB, Tuple space, MongoDB), Multivalue databases (Extensible Storage Engine - ESE/NT, OpenQM, Revelation Software's OpenInsight, Rocket U2), Object database (db4o, GemStone/S, InterSystems Caché, JADE, Objectivity/DB, ObjectStore, Versant Object Database, ZODB), Ordered key-value store (Berkeley DB, IBM Informix C-ISAM, MemcacheDB, NMDB), Tabular (BigTable, Hbase, Hypertable, Mnesia), Tuple store (Apache River).

As it is not in authors' intention to provide a NoSQL taxonomy in this paper, we will not tread further on the reasons the two sources used for their results.

It is easy for one to see that the two taxonomies, although seemingly using the same reason (the manner of implementation) are providing different results (products which are in the same category in one taxonomy are listed in separate categories in the other one, the categories labels and divisions are different).

For this reason we decided to use as grouping criteria, instead of a single property, an ad-hoc set composed of: main intended usage, manner of implementation, ease of obtaining and testing. We only searched for open-source solutions, having roughly the same number of "users" (we mean implementations in use), and with more or less the same size for the average and the largest installation and, if possible, with the same intended use.

As such, from the multitude of NoSQL solutions available we restricted our research to a single type of NoSQL databases (meaning "the Wide Column Store / Column Families" subtype from the first taxonomy which is roughly equivalent with the "Key-value store" type from the second taxonomy) and from this set we took two of the products which have larger use at the present moment. The result was that we took into consideration for this study only Hbase and Cassandra (which, besides the qualities given earlier are also products from the same family and based on the same framework – Hadoop).

As some description of the selected solutions maybe in order, here it is:

"The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures."[20]

"HBase is an open-source, distributed, versioned, column-oriented store modeled after Google' Bigtable: A Distributed Storage System for Structured by Chang et al. Just as Bigtable leverages the distributed data storage provided by the Google File System, HBase provides Bigtable-like capabilities on top of Hadoop."[21]

"The Apache Cassandra Project develops a highly scalable second-generation distributed database, bringing together Dynamo's fully distributed design and Bigtable's ColumnFamily-based data model."[19]

As a reference element we also took MySQL (also open-source, but full relational/SQL able) to see what is lost and what is gained by using a NoSQL solution instead of a "classic" one.

III. A QUALITATIVE POINT OF VIEW

One can compare some items based on qualitative or quantitative criteria. As such we will start by comparing what features are available for the NoSQL databases taken into account. The features we searched for are:

- Persistence (1)
- Replication (2)
- High Availability (3)
- Transactions (4)
- Rack-locality awareness (5)
- Implementation Language (6)
- Influences / sponsors (7)
- License type (8)

The results are given in the following table. One can see that the three products offer the same features, the only differences being the ones related to transactions, implementation language and license type (although the other features are not implemented or working in the same way). The dual licensing solution available now for MySQL is a result of the series of acquisitions from the last few years (Sun bought MySQL, Oracle bought Sun).

TABLE I. A COMPARATIVE TABLE WITH THE FEATURES OF THE THREE SELECTED PRODUCTS

Feat.	Cassandra	HBase	MySQL
1	yes	yes	yes (using a different type of connection than the typical one)
2	yes	yes	yes
3	distributed	distributed	distributed, available with MySQL Cluster
4	eventually consistent	locally (row-level) consistent	consistent (full ACID actually)
5	yes (inherited from Hadoop)	yes (inherited from Hadoop)	yes (with MySQL Cluster)
6	Java	Java	ANSI C / ANSI C++
7	Dynamo and BigTable, Facebook/Digg/ Rackspace	BigTable	Oracle
8	Apache 2.0	Apache 2.0	GPL+FLOSS / proprietary

IV. A QUANTITATIVE POINT OF VIEW

For quantitative evaluation criteria we used two different sets, one related to size and one related to performance.

A. *Common instalations size measurements*

The information used for size related criteria are mainly taken from [19], [22] but also form various sources. There will be no values given for MySQL as the NoSQL products are specially designed for large size databases so there is no point in comparing them with MySQL (it is common knowledge that the largest MySQL installations cannot be larger than, let's say, 1 million records of average size without memory caching and extended sharding; over that limit information retrieval is becoming too slow to be useful in any situation [15]).

There is no official measurement unit for the size of a DB installation but we can take several factors into account:

- Number of records / rows /documents stored: [22] is giving values of 6 to 450 million records for different installations of HBase, most of them being in the range of 6 to 25 million records; various sources are giving sizes of 2 to 150 million records for diverse installations of Cassandra;
- Number of nodes in an installation: [22] is giving values of 5 to 110 nodes for Hbase, most of them being in the range of 6 to 20 nodes; 4 to 150 nodes for Cassandra with most installations in the span of 5 to 25 nodes;
- Total size of the installations: less documented; some instances are showing maximal sizes for current installations of 140 TB for Hbase and 150 TB for Cassandra.

B. *Performance measurements*

Most of the data from the following paragraphs, included in the figures is obtained from [2] which is describing a laboratory based benchmark which uses YCSB (Yahoo! Cloud Serving Benchmark) as a measurement tool (more on YCSB can be found at [25]). The benchmark was run on 120 million records of small size (1kB), 6 node, and 0.12 TB equivalent installations of the three products.

1) Performance in a write intensive environment (the number of writes is equal to the one of reads)

The performance achieved can be seen in Figure 1 and 2.

Figure 1. Read latency in a write intensive environment (source: [2])

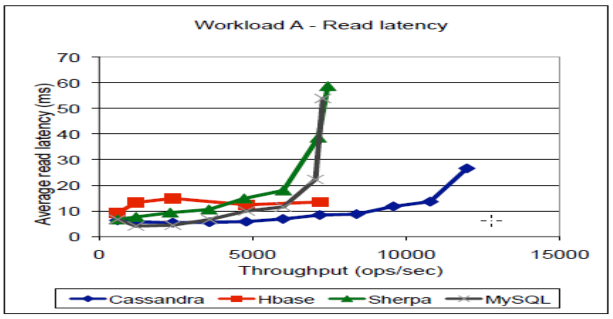
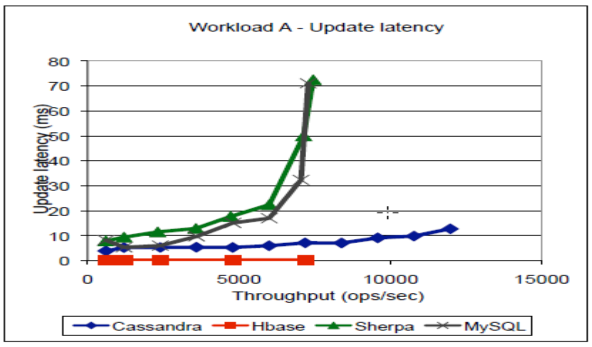


Figure 2. Write latency in a write intensive environment (source: [2])



The latency for both reading and writing in Figures 1 and 2 is given as a dependency of number of operations per second.

The two figures are indicating that:

- Over approximately 7000 read or write operations per second both MySQL and its variation called Sherpa are becoming unresponsive – the latency time is becoming too great for a real life application;
- The write performance of Hbase is greatly improved by the fact that it's committing to memory (and not directly to disk as the other products). [2] is indicating that the write performance of Cassandra, Sherpa and MySQL can also be improved by using a log disk.

2) Performance in a read intensive environment (the read operations are accounting for 95% of the total number of operations)

Studying Figures 3 and 4, one can see that:

- In a read intensive environment, MySQL and its Sherpa variation are offering better results, keeping the pace with the NoSQL products (although, taken into account that the benchmark database was not of a real large size, we do not think that this trend will look the same for larger installations);
- A particular figure is given again by Hbase which is obtaining a very good write performance by committing to memory.

Figure 3. Read latency in a read intensive environment (source: [2])

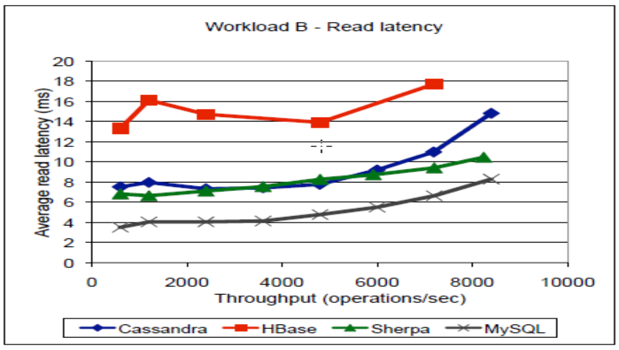
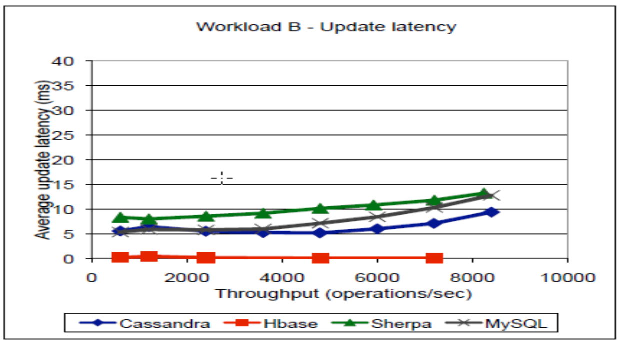


Figure 4. Write latency in a read intensive environment (source: [2])



V. CONCLUSIONS

Although the SQL and the NoSQL databases are having some shared features their behaviors are not similar in given instances. This is suggesting that they cannot be used interchangeable for solving any type of problem but one shall rather choose between the two types of databases for a given instance.

REFERENCES

[1] Agrawal, Rakesh et al., "The Claremont report on database research", http://doi.acm.org/10.1145/1462571.1462573, SIGMOD Record (ACM) 37 (3): 9–19. ISSN 0163-5808,

[2] Cooper, Brian F., "Yahoo! Cloud Serving Benchmark", http://research.yahoo.com/files/ycsb-v4.pdf, (unpublished)

[3] Bucur, Cristian; Tudorica, Bogdan George, "Solutions for working with large data volumes in web applications", The Proceedings of the IE 2011 „Education, Research & Business Technologies" International Conference, 5-7 May 2011, (in press),

[4] Chang, Fay, et al., "Bigtable: A Distributed Storage System for Structured Data", http://labs.google.com/papers/bigtable-osdi06.pdf, Google, (unpublished),

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

[6] Cooper, Brian F.; Silberstein, Adam; Tam, Erwin; Ramakrishnan, Raghu; Sears, Russell, "Yahoo! cloud serving benchmark", http://research.yahoo.com/files/ycsb.pdf, ACM Symposium on Cloud Computing, ACM, Indianapolis, IN, USA (2010),

[7] De Sterck, Hans, Zhang, Chen, "Supporting multi-row distributed transactions with global snapshot isolation using bare-bones Hbase", http://www.cs.uwaterloo.ca/~c15zhang/ZhangDeSterckGrid2010.pdf, The 11th ACM/IEEE International Conference on Grid Computing (Grid 2010), Oct 25-29, 2010, Brussels, Belgium

[8] Edlich, Stefan, "NoSQL, your ultimate guide to the non - relational universe!", http://nosql-database.org/, (unpublished)

[9] Eure, Ian, "Looking to the future with Cassandra | Digg about", http://about.digg.com/blog/looking-future-cassandra, About.digg.com. 2009-09-09, (unpublished),

[10] Hamilton, James, "One size does not fit all", http://perspectives.mvdirona.com/CommentView,guid,afe46691-a293-4f9a-8900-5688a597726a.aspx, (unpublished),

[11] Kellerman, Jim, "HBase: structured storage of sparse data for Hadoop" http://blog.rapleaf.com/wp-content/uploads/2007/12/hbase.pdf, (unpublished),

[12] Lakshman, Avinash; Malik, Prashant, "Cassandra, a decentralized structured storage system", http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf, Cornell University, (unpublished),

[13] Lakshman, Avinash; Malik, Prashant, “Cassandra, Structured storage system over a P2P network”, http://static.last.fm/johan/nosql-20090611/cassandra_nosql.pdf, (unpublished),

[14] Peng, Daniel; Dabek, Frank, “Large-scale incremental processing using distributed transactions and notifications”, http://www.google.ca/url?sa=t&source=web&cd=3&ved=0CCQQFjAC&url=http%3A%2F%2Fwww.usenix.org%2Fevents%2Fosdi10%2Ftech%2Ffull_papers%2FPeng.pdf&rct=j&q=Large-scale%20Incremental%20Processing%20Using%20Distributed%20Transactions%20and%20Notifications&ei=eM24TOYnjqedB_mHmLUN&usg=AFQjCNGGm1Xfaml5lq6Aj1R2BlX7WilIuQ&sig2=ZZcPWxhiMVSnY-DmewIFIg&cad=rja, The 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2010), Oct 4–6, 2010, Vancouver, BC, Canada,

[15] Peters, Mike, “How to install Cassandra + Thrift (and why you should care)”, http://www.softwareprojects.com/resources/programming/t-how-to-install-cassandra-+-thrift-and-why-you-shou-1956.html, (unpublished)

[16] Stack, Michael, “HBasics: an introduction to Hadoop Hbase”, http://static.last.fm/johan/huguk-20090414/michael_stack-hbase.pdf, HUGUK, April 14th, 2009,

[17] Wei, Zhou; Pierre, Guillaume; Chi, Chi-Hung, “CloudTPS: scalable transactions for web applications in the cloud”, http://www.globule.org/publi/CSTWAC_ircs53.html, Technical report IR-CS-53, Vrije Universiteit, February 2010, to be published at IEEE Transactions on Services Computing, 2011 (in press),

[18] Wei, Zhou; Pierre, Guillaume; Chi, Chi-Hung, “Consistent join queries in cloud data stores”, http://www.globule.org/publi/CJQCDS_ircs68.html, Technical report IR-CS-68, Vrije Universiteit, January 2011 (unpublished),

[19] ***, “Cassandra”, http://cassandra.apache.org, (unpublished)

[20] ***, “Hadoop”, http://hadoop.apache.org, (unpublished)

[21] ***, ”Hbase”, http://hbase.apache.org, (unpublished)

[22] ***, “Hbase / Powered by”, http://wiki.apache.org/hadoop/Hbase/PoweredBy, (unpublished)

[23] ***, “NoSQL”, http://en.wikipedia.org/wiki/NoSQL, (unpublished)

[24] ***, “The next generation cloud database“, http://www.microsoft.com/windowsazure/sqlazure/database/, (unpublished),

[25] ***, “Yahoo! Cloud Serving Benchmark (YCSB)”, https://github.com/brianfrankcooper/YCSB/wiki, (unpublished)