NoSQL databases: a software engineering perspective

João Ricardo Lourenço[1] , Veronika Abramova[1], Marco Vieira[1], Bruno Cabral[1], Jorge Bernardino[1,2]

[1] CISUC – Centre of Informatics and Systems of the University of Coimbra
FCTUC – University of Coimbra, 3030-290 Coimbra, Portugal
[2] ISEC – Superior Institute of Engineering of Coimbra
Polytechnic Institute of Coimbra, 3030-190 Coimbra, Portugal
jorl17.8@gmail.com, veronika@student.dei.uc.pt, mvieira@dei.uc.pt, bcabral@dei.uc.pt, jorge@isec.pt

Abstract. For over forty years, relational databases have been the leading model for data storage, retrieval and management. However, due to increasing needs for scalability and performance, alternative systems have started being developed, namely NoSQL technology. With increased interest in NoSQL technology, as well as more use case scenarios, over the last few years these databases have been more frequently evaluated and compared. It is necessary to find if all the possibilities and characteristics of non-relational technology have been disclosed. While most papers perform mostly performance evaluation using standard benchmarks, it is nevertheless important to notice that real world scenarios, with real enterprise data, do not function solely based on performance. In this paper, we have gathered a concise and up-to-date comparison of NoSQL engines, their most beneficial use case scenarios from the software engineer viewpoint, their advantages and drawbacks by surveying the currently available literature.

Keywords: NoSQL databases, Key-Value, Document Store, Columnar, Graph, Cassandra, MongoDB, Couchbase, Software engineering, Quality attributes

1 Introduction

Relational databases have been the stronghold of modern computing applications, providing guarantees with its ACID properties (Atomicity, Consistency, Isolation, Durability) that have been tested time and time again. However, recent demands with regards to scalability [1, 2, 3], in particular due to Big Data [4], have led to the creation of numerous NoSQL databases with different strong and weak points.

The term NoSQL was first used in 1988 to name a relational database that did not have a SQL interface [5]. It was then brought back in 2009 as the name for an event which highlighted new non-relational databases such as BigTable and Dynamo [6] and has since been used without an "official" definition. Generally speaking, a NoSQL database is one that uses a different approach to data storage when compared to relational database management systems [7, 8]. These approaches often abandon the full support of ACID properties as a tradeoff for their increased performance and horizontal scalability [7, 1]. Brewer coined the term BASE for these systems — they

are Basically Available, have a Soft state (during which they are not yet consistent), and are Eventually consistent, as opposed to ACID systems [9]. This BASE model forfeits the essential ACID properties of consistency and isolation in order to favor "availability, graceful degradation, and performance" [9]. While originally the term stood for "No SQL", it has recently been restated as "Not Only SQL" [1, 7, 10] to highlight that these systems rarely fully drop the relational model. Thus, in spite of being a recurrent theme in literature, NoSQL is a very broad term, encompassing very distinct database systems.

There are hundreds of readily available NoSQL databases, and each has different use case scenarios [11]. They are usually divided in four categories [9, 12, 2], according to their data model and storage: Key-Value Stores, Document Stores, Extensible Record Stores, Wide Column Stores or Column Families and Graph databases. There have been attempts at classifying these NoSQL systems according to their best use-case scenarios, but there has not yet been a comprehensive revision of the literature from the point of view of software engineering. We try to fill this gap by looking at several software engineering quality attributes and evaluating which NoSQL system best fits.

The remainder of this paper is structured as follows. In section 2, we perform a short review of the literature and performance evaluation surrounding NoSQL systems. In Section 3, we analyze the different quality attributes and identify the currently available best NoSQL solutions. Finally, section 4 presents future work possibilities and final conclusions.

2 State of the Art

The idea of NoSQL was re-introduced in 2009 during an event about distributed databases. At that time, it is important to differentiate papers published by Stonebreaker and Cattell [12, 13, 14]. Stonebreaker defends that the two possible main reasons to move to NoSQL databases are resumed to: performance and flexibility. Performance is mainly focused on sharing and management of distributed data, while flexibility of data corresponds to the semi-structured or unstructured data that may arise from the web. Already in 2011 the main non-relational technologies were already known and described accordingly to their functioning. Hecht and Jablonski [11] described the main characteristics offered by different NoSQL solutions such as Voldemort, Redis, Riak, MongoDB, CouchDB, Cassandra, HBase, etc. Konstantinou et al. [15] performed a study based on the elasticity of non-relational solutions and compared HBase, Cassandra and Riak during execution of read and update operations. The authors concluded that HBase provides high elasticity and fast reads while Cassandra is capable of delivering fast inserts (writes). On the other hand, Riak did not show good scaling and high performance increase, regardless of the operation type.

Starting from 2012 and up to today, NoSQL databases have been more and more evaluated and compared to RDBMSs. Performance evaluation executed by [16] compared Cassandra, MongoDB and PostgreSQL. It was concluded that MongoDB is capable of providing high throughput but mainly when it is used as a single server

instance. The best choice for the large distributed sensor system was considered Cassandra. Floratou et al. [4] used the recent standard Yahoo! Cloud Serving benchmark [17] and TPC-H to compare the performance of MongoDB and MS SQL Server as well as evaluating Hive [18]. As a conclusion, Floratou et al. state that NoSQL technology has room to improve and should be continuously updated. Ashram and Anderson [7] state that usage of non-relational technology creates additional engineering on the programmers' side. Parker et al. also chose MongoDB and compared its performance with MS SQL Server while using only one server instance [19]. According to the presented results, while performing inserts, updates and selects, the authors concluded that MongoDB is faster but MS SQL outperforms MongoDB while working with the more complex queries instead of key-value access.

In [10] a survey of some of the most popular NoSQL solutions is described. The authors state some of the advantages and main uses accordingly to the NoSQL database type. As another evaluation, [20] performed their tests using real medical scenarios suing MongoDB and CouchDB. In [21], the Yahoo! Cloud Serving Benchmark is used with a middleware layer that allows translating SQL queries into NoSQL commands. They tested Cassandra and MongoDB with and without the middleware layer.

3 Software Quality Attributes and NoSQL Systems

As we have seen, over the past years, a variety of studies regarding NoSQL technology have been made. Those results give us a performance perspective on the execution speed of different types of requests. However, performance is not a solo aspect that must be considered when moving a working system from relational storage to non-relational. Software engineering defines a broad set of quality attributes which may be desired as non-functional requirements used to evaluate the performance of a system. Among these, there are some nearly ubiquitous attributes of which at least one certainly applies to any large software project. We looked at these attributes and the current NoSQL literature to find the best NoSQL solution for each of them. In particular, we look at availability, reliability, scalability, durability, operation performance (read and write) and recovery and stabilization time. These attributes cover a broad set of distinct software engineering scenarios and have been the target (even if indirectly) of some studies, rendering them ideal picks for this work.

3.1 Availability

Availability concerns what percentage of time a system is operating correctly [1]. NoSQL technology is inherently bound to provide availability more easily than SQL systems. In fact, given the existence of Brewer's CAP theorem, and the undeniable presence of failures in real-world networks, NoSQL databases oppose most relational databases by favoring availability instead of consistency. Thus, one can assert that the higher the availability of a NoSQL system, the less likely it is that it provides high

consistency guarantees. Several NoSQL databases provide ways to tune the tradeoff between consistency and availability, including Dynamo [22], Cassandra, CouchDB and MongoDB [9].

Apache CouchDB uses a shared-nothing clustering approach, allowing all replica nodes to continue working even if they are disconnected, thus being a good candidate for systems where high availability is needed [9]. It is worth noting, however, that this database periodically requires a compaction step which may hinder system performance, but which does not affect the availability of its nodes under normal operation [3].

In 2013, [23] tested several NoSQL Databases (Aerospike, Cassandra, Couchbase and MongoDB) concerning its failover characteristics. Their results showed that Aerospike had the lowest downtime, followed by Cassandra, with MongoDB having the least favorable downtime. One should note that the results shown in the paper are limited to RAM-only datasets and hence might not be the best source for real-world scenarios. MongoDB's results are also not surprising, as even though it allows for fine-tuning (to adjust the consistency-availability tradeoffs), several tests have shown that it is not the best choice for a highly available system, in particular due to overhead when nodes are rejoining the system (see, for instance, [1], [9] and our reliability section). Lastly, [5] tested several NoSQL databases on the Cloud and noted that Riak could not provide high-availability under very high loads.

Thus, there is no obvious candidate for a highly available system, but there are several competing solutions, in particular when coupled with systems such as Memcached [12]. The specific architecture (number of replicas, consistency options, etc.) employed will play a major role, as pointed by several authors [25, 24, 23]. Furthermore, the popular MongoDB and Riak databases seem less likely to be good picks for this use case scenario.

3.2 Scalability

Scalability concerns a system's ability to deal with increasing workloads [1]. In the context of databases, it may be defined as the change in performance when new nodes are added, or hardware is improved [26]. NoSQL databases have been developed specifically to target scenarios where scalability is very important. These systems rely on horizontal and "elastic" scalability, by adding more nodes to a system instead of upgrading hardware [9, 13, 14]. The term "elastic" refers to elasticity, which is a characterization of the way a cluster reacts to the addition or removal of nodes [26].

In [27] the authors compared Cassandra and HBase, improving upon previous work. They concluded that both databases scale linearly with different read and write performances. They also provided a more in-depth analysis at Cassandra's scalability, noticing how performing horizontal scalability with this platform leads to less performance hassles than performing vertical scalability.

In [26], the authors measure the elasticity and scalability of Cassandra, HBase and MongoDB. They showed surprise by identifying "superlinear speedups for clusters of size 24" when using Cassandra, stating that "it is almost as if Cassandra uses better algorithms for large cluster sizes". For clusters of sizes 6 and 12, their results show HBase the fastest competitor with stable performance. Regarding elasticity, they

found that HBase gives the best results, stabilizing the database significantly faster than Cassandra and MongoDB. Regarding the different scalability capabilities of the databases themselves, Cassandra, HBase and Riak all support the addition of machines during live operation.

Further studies regarding scalability are needed in literature. It is clear that NoSQL databases are scalable, but the question of which scale the most, or with the best performance, is still left unanswered. Nevertheless, we can conclude that popular choices for highly scalable systems are Cassandra and HBase. One must also take notice that scalability will be influenced by the particular choice of configuration parameters.

3.3 Durability

Durability refers to the requirement that data be valid and committed to disk after a successful transaction [1]. As we have previously covered, NoSQL databases act on the premise that consistency doesn't need to be fully enforced in the real world, preferring to sacrifice it in adjustable ways for achieving higher availability and partition tolerance. This impacts durability, as if a system suffers from consistency problems, its durability will also be at risk, leading to potential data loss [28].

In [28], the authors test Aerospike, Couchbase, Cassandra and MongoDB in a series of tests regarding durability and performance tradeoffs. Their results featured Aerospike as the fastest performing database by a factor of 5-10 when the databases were set to synchronous replication. However, most scenarios do not rely on synchronous replication, but rather asynchronous (meaning that changes aren't instantly propagated among nodes). In that sense, the same authors, which in [23] studied the same databases in the context of failover characteristics, show that MongoDB loses less data upon node failure when asynchronous replication is used. Cassandra comes as forerunner to MongoDB by about a factor of 100, and Aerospike and Couchbase both lose very large amounts of data. In [1], MongoDB is found to have issues with data loss when compared to CouchDB, in particular during recovery after a crash. In the same paper, the authors highlight that CouchDB's immutable append only B+ Tree ensures that files are always in a valid state. CouchDB's durability is also noticed and justified by the authors of [12]. It should be noted that document-based systems such as MongoDB and CouchDB usually use a single-versioning system, which is designed specifically to target durability [29].

In conclusion, as with other properties, the durability of NoSQL systems can be fine-tuned according to the needs. However, databases based on immutability, such as CouchDB, are good picks for a system with good durability due to their inherent properties [1]. Furthermore, single-version databases, such as MongoDB, should also be the focus of those interested in durability advantages.

3.4 Reliability

Reliability concerns the system's probability of operating without failures for a given period of time [29]. The higher the reliability, the less likely it is that the system fails.

Recently, Domaschka et al., in [29], have proposed taxonomy for describing distributed databases with regards to their reliability and availability. Since reliability is significantly harder to define than availability (as it depends on the context of the application requirements), the authors suggest that software architects consider the following two questions: "(1) How are concurrent writes to the same item resolved?; (2) What is the consistency experienced by clients?". With these in mind, and by using their taxonomy, we can see that systems which use single-version techniques, such as Redis, Couchbase, MongoDB and Neo4j, all perform online write conflict resolution detection, being good picks for a reliable system in the sense that they answer question (1) with reliable options. Regarding question (2), MongoDB, CouchDB, Neo4J, Cassandra and HBase all provide strong consistency guarantees. Thus, in order to achieve strong consistency guarantees and good concurrent write conflict resolution, as proposed by the authors, one should look at systems which have both these characteristics – MongoDB and Neo4j.

In conclusion, in spite of reliability being an important quality attribute, we have found that there is little focus in current literature about this topic, and, therefore, are limited in our answers to this research question.

3.5 Operation Performance

When it comes to the performance and execution of different types of operations, NoSQL databases are divided mostly into two categories: read and write optimized. That means that, in part, regardless of the system type and records, the database has an optimization that is granted by the used mechanisms used for storage, organization and retrieval of data. For example, Cassandra is known for being very highly optimized during execution of writes (inserts) and is not able to show the same performance during reads. Overall, Column Family and Key-Value databases use more memory to store their data, some of those being completely in-memory (and, hence, completely unsuited for other attributes such as durability). Document Stores, on the other hand, are considered being more read optimized. This behavior resembles that of relational databases, where data loading and organization is slower, with the advantage of better preparing the system for future reads. We should also consider that databases such as MongoDB and Couchbase are considered more enterprise solutions with a set of mechanisms and functionalities besides traditional key-value retrieval, which is mostly used not only by Key-Value stores but also by Column Families databases.

3.6 Recovery Time and Stabilization Time

Besides availability, there are other failover characteristics which determine the behavior of a system and might impact system stability. In the study made in [23], which we have already covered, the authors measure the time it takes for several NoSQL systems to recover from a node failure – the recovery time –, as well as the time it takes for the system to stabilize when that node rejoins the cluster – the stabilization time. They find that MongoDB has the best recovery time, followed by

Aerospike (when in synchronous change propagation mode), with Couchbase having values an order of magnitude slower and Cassandra two orders of magnitude slower. Regarding the time to stabilize on node up, all systems perform well (< 1ms) with the exception of the exact systems which we have just mentioned. MongoDB takes a long 31 seconds to recover to stabilize on node reentry, and Aerospike in synchronous mode takes 3 seconds. These results tend to indicate that MongoDB and Aerospike are good picks if one is looking for good recovery times, but that these choices should be taken with care, such that when a node reenters the system, it does not affect its stability. Overall, the topic of failover is highly dependent of configuration and desired properties.

3.7 Summary Table for current popular databases

By gathering all the information we presented in the previous sections, we tried to establish a comprehensive summary table that indicates which database best suits each quality attribute. Each column in the table represents a NoSQL database, and each row one of the studied software engineering quality attributes. A "+" is assigned when a particular database is oriented towards a quality attribute. If it is particularly geared for it, or comparatively better, we add more symbols ("++"). For instance, Cassandra is write-performance oriented, even more than Couchbase. Similarly, we assigned a "-" (or "--", to allow for a comparative analysis) when a database is not an ideal pick, according to our literature revision. In cases where we were unsure what was the correct answer, we used the question mark symbol "?".

We used the criteria described in each of the previous sections to quantify the databases. Regarding availability, the downtime was used as a primary measure, together with relevant studies [23, 5]. With respect to scalability, we looked at each database's elasticity, its increase in performance due to horizontal scaling, and the ease of on-line scalability (for instance, project Voldemort does not allow for on-line addition of nodes [9]). Durability relates to the use of single or multi version control schemes, as well as consistency guarantees. Reliability is graded according to the taxonomy presented in [29] and by looking at synchronous propagation modes (databases which do not support them tend to be less reliable). For read and write performance, we considered recent studies [23] and the fine-tuning of each database. For recovery time and stabilization time, highly related to availability, we based our classification on the popular results shown in [23]. We looked at the most commonly cited databases or those in the most relevant papers.

Table 1. Summary table of different quality attributes studied for popular databases.

	MongoDB	CouchDB	Cassandra	HBase	Voldemort	Aerospike	Couchbase
Availability	-	+	+	-	+	+	+
Scalability	-	-	+	+	-	+	+
Durability	+	+	+	+	+	-	-
Reliability	++	+	+	+	?	-	-
Write-Performance	-	-	++	+	++	+	+
Read-Performance	++	++	-	-	-	+	-
Recovery Time	++	?	--	?	?	++	+
Stabilization Time	--	?	+	?	?	--	+

By analyzing Table 1, we can see that MongoDB is the database that mostly resembles the classical relational use case scenario – it is better suited for reliable, durable and read-oriented use cases. It is somewhat lacking in terms of availability, scalability and write-performance, and it is very hindered by its stabilization time (which is also one of the reasons for its low availability). CouchDB provides similar technology to MongoDB, but is better suited for situations where availability is needed. Cassandra is a multi-purpose database (in particular due to its configurable consistency properties) which mostly lacks read performance (since it is tuned for write-heavy workloads). HBase has similar capabilities but is unable to cope with high loads, limiting its availability in these scenarios. Furthermore, it is not as efficient during write operations. We lack some information on Voldemort, but find it to be a poor pick for scalable solutions or situations where reads are very important – it is, however, a good pick for durable, write-heavy scenarios. Aerospike suffers from data loss issues, affecting its durability, and it also has issues with stabilization time (in particular in synchronous mode). Lastly, Couchbase provides good availability capabilities (coupled with good recovery and stabilization times), making it a good candidate for situations where failover is bound to happen. We should highlight that there are more quality attributes that should be focused on, which we intend to do in future work, and, thus, that this table does not intend to show that "one database is better than another", but, rather, than some database is better for a particular use case scenario where these attributes are needed.

5 Conclusions and Future Work

In this paper we described the main characteristics and types of NoSQL technology while approaching different aspects that highly contribute to the use of those systems. We also presented the state of the art of non-relational technology by describing some of the most relevant studies and performance tests and their conclusions, after surveying a vast number of publications since NoSQL's birth.

We concluded that although there have been a variety of studies and evaluations of NoSQL technology, there is still not enough information to verify how suited each non-relational database is in a specific scenario or system. Moreover, each working system differs from another and all the necessary functionalities and mechanisms highly affect the database choice. Sometimes there is no possibility of clearly stating the best database solution. Furthermore, we tried to find the best databases on a quality attribute perspective, an approach still not found in current literature. In the future, we expect that NoSQL databases will be more used in real enterprise systems, allowing for more information and user experience available to conclude the most appropriate use of NoSQL according to each quality attribute and further improve this initial approach.

As we have seen, NoSQL is still an in-development field, with many questions and a shortage of definite answers. Its technology is ever-increasing and ever-changing, rendering even recent benchmarks and performance evaluations obsolete. There is

also a lack of studies which focus on use-case oriented scenarios or software engineering quality attributes. All of these reasons make it difficult to find the best pick for each of the quality attributes we chose in this work, as well as others. The summary table we presented makes it clear that there is a current need for a broad study of quality attributes in order to better understand the NoSQL ecosystem, and it would be interesting to conduct research in this domain. When more studies and with more consistent results have been performed, a more thorough survey of the literature can be done, and with clearer, more concise results.

Acknowledgments

This research would not have been made possible without support and funding of the FEED – Free Energy Data and iCIS – Intelligent Computing in the Internet Services (CENTRO-07 - ST24 – FEDER – 002003) projects, to which we are extremely grateful.

References

1. K. Orend: Analysis and Classification of NoSQL Datbases and Evaluation of their Ability to Replace an Object-relational Persistance Layer. Master thesis, Technical University of Munich, Munich (2010)
2. Lavitt, N.: Will NoSQL databases live up to their promise? Computer, Vol. 43 No. 2, pp. 12-14. (2010)
3. Chang, F.; Dean, J., Ghemawat, S.; Hsieh, W. C.; Wallach, D. A.; Burrows, M.; Gruber, R.: Bigtable: A Distributed Storage System for Structured Data. OSDI (2006)
4. Floratou, A.; Teletia, N.; DeWitt, D.J.; Patel, J. M;Zhang, D.: Can the elephants handle the NoSQL onslaught? Proc. VLDB Endow. 5, 12, 1712-1723. (2012)
5. Lith, A.; Mattson, J.: Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data, p. 70. Göteborg: Department of Computer Science and Engineering, Chalmers University of Technology (2010)
6. Sadalage, J. S.; Fowler, M.: NoSQL Distilled. Addison-Wesley, Upper Saddle River, NJ (2013)
7. Schram, A. and Anderson, K. M. MySQL to NoSQL: data modeling challenges in supporting scalability. In: Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity (SPLASH '12). ACM, New York, NY, USA, 191-202. (2012)
8. NoSQL, http://nosql-database.org/
9. Strauch, C.: NoSQL Databases. Lecture “Selected Topics on Software-Technology Ultra-Large Scale Sites” (unpublished)
10. S. D. Kuznetsov, A. V. Poskonin. NoSQL data management systems. Programming and Computer Software, November 2014, Volume 40, Issue 6, pp 323-332 (2014)
11. Hecht, R., Jablonski, S. NoSQL evaluation: A use case oriented survey. Cloud and Service Computing (CSC), 2011 International Conference. 2011, 336 – 341 (2011)
12. Cattell, R. Scalable SQL and NoSQL data stores. SIGMOD Rec. 39, 4, 12-27. DOI=10.1145/1978915.1978919 (2011)

13. Stonebraker, M. Stonebraker on NoSQL and Enterprises. Communications of)the ACM, Vol. 54 No. 8, Pages 10-11 (2011)
14. Stonebraker, M. SQL databases v. NoSQL databases. Commun. ACM 53, 4, 10-11. DOI=10.1145/1721654.1721659 (2010)
15. Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D. and Koziris, N. On the elasticity of NoSQL databases over cloud management platforms. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM '11), Bettina Berendt, Arjen de Vries, Wenfei Fan, Craig Macdonald, Iadh Ounis, and Ian Ruthven (Eds.). ACM, New York, NY, USA, 2385-2388. (2011)
16. Van der Veen, J.S.; TNO, Groningen, Netherlands ; van der Waaij, B. ; Meijer, R.J. Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual. Cloud Computing (CLOUD), 2012 IEEE 5th International Conference. June 2012, 431 – 438 (2012)
17. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. "Benchmarking cloud serving systems with YCSB". In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10). ACM, New York, NY, USA, 143-154 (2010)
18. Brown, R. A. 2009. Hadoop at home: large-scale computing at a small college. SIGCSE Bull. 41, 1, 106-110. DOI=10.1145/1539024.1508904 (2009)
19. Parker, Z., Poe, S. and Vrbsky, S. V. Comparing NoSQL MongoDB to an SQL DB. In Proceedings of the 51st ACM Southeast Conference (ACMSE '13). ACM, New York, NY, USA, Article 5 , 6 pages. (2013)
20. Silva, L. A., Beroud, L., Costa, C., Oliveira, J. L. Medical imaging archiving: A comparison between several NoSQL solutions. Biomedical and Health Informatics (BHI), IEEE-EMBS International Conference. 1-4 June 2014, 65 – 68 (2014)
21. Rith, J., Lehmayr, P. S., Meyer-Wegener, K. Speaking in tongues: SQL access to NoSQL systems. SAC 2014: 855-857 (2014)
22. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: Amazon's highly available key-value store, In: *SOSP,* pp. 205–220 (2007)
23. Nelubin, D.; Engber, B.: NoSQL Failover Characteristics: Aerospike, Cassandra, Couchbase, MongoDB. Thumbtack Technology (2013)
24. Tsuyuzaki, K.;Onizuka, M.: NoSQL Database Characteristics and Benchmark System. NTT Technical Review, Vol 10 (2012)
25. Beyer, F.;Koschel, A.; Schulz, C.; Schäfer, M.: Testing the Suitability of Cassandra for Cloud Computing Environments. 2nd Intl. Conf. on Cloud Computing, Grids, and Virtualization, pp. 86-91 (2011)
26. Kuhlenkamp, J.; Markus, K.; Oliver, R.: Benchmarking Scalability and Elasticity of Distributed Database Systems. Proceedings of the VLDB Endowment 7.13 (2014)
27. Dory, T.; Van Roy, P.; Tran, N. L.: Comparative elasticity and scalability measurements of cloud databases. Proc of the 2nd ACM symposium on cloud computing (SoCC) (2011)
28. Nelubin, D.; Engber, B.: Ultra-High Performance NoSQL Benchmarking, Analyzing Durability and Performance Tradeoffs. Thumbtack Technology (2013)
29. Domaschka, J.; Hauser, C., B.; Erb, B.: Reliability and Availability Properties of Distributed Database Systems. Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International. (2014)