Open Access

Open Journal of Databases (OJDB)
Volume 1, Issue 2, 2014

www.ronpub.com/journals/ojdb
ISSN 2199-3459

Which NoSQL Database? A Performance Overview

Veronika Abramova [A], Jorge Bernardino [A, B], Pedro Furtado [B]

[A] Polytechnic Institute of Coimbra - ISEC / CISUC, Coimbra, Portugal, veronika@student.dei.uc.pt
[B] University of Coimbra – DEI / CISUC, Coimbra, Portugal, pnf@dei.uc.pt, jorge@isec.pt

ABSTRACT

NoSQL data stores are widely used to store and retrieve possibly large amounts of data, typically in a key-value format. There are many NoSQL types with different performances, and thus it is important to compare them in terms of performance and verify how the performance is related to the database type. In this paper, we evaluate five most popular NoSQL databases: Cassandra, HBase, MongoDB, OrientDB and Redis. We compare those databases in terms of query performance, based on reads and updates, taking into consideration the typical workloads, as represented by the Yahoo! Cloud Serving Benchmark. This comparison allows users to choose the most appropriate database according to the specific mechanisms and application needs.

TYPE OF PAPER AND KEYWORDS

Short communication: *NoSQL databases, performance evaluation, execution time, benchmark, YCSB*

1 INTRODUCTION

Nowadays, databases are considered a vital part of the organizations and are used all over the globe. Relational databases allow data storage, extraction and manipulation using a standard SQL language. Until now, relational databases were an optimal enterprise choice. However, with the constant growth of stored and analyzed data, relational databases exhibit a variety of limitations, e.g. the limitations of scalability and storage, and efficiency losing of query due to the large volumes of data, and the storage and management of larger databases become challenging.

In order to overcome these limitations, a new database model was developed with a set of new features, known as NoSQL databases [1]. Non-relational databases emerged as a breakthrough technology, and can be used sole or as complement to the relational database. NoSQL increases the performance of relational databases by a set of new characteristics and advantages. In comparison to relational databases, NoSQL databases are more flexible and horizontally scalable [2]. They are capable of taking advantage of new clusters and nodes transparently, without requiring additional database management or manual distribution of information. Since database administration may be a difficult task with such amounts of data, NoSQL databases are projected to automatically manage and distribute data, recover from faults and repair the whole system automatically [3].

When NoSQL technology started to emerge, NoSQL databases were known and characterized by the lack of consistency of its stored data. For the companies and systems, where strong consistency was essential, the lack of consistency could be a big limitation. With the increase of popularity of non-relational databases, such features and system

characteristics started to evolve. Currently, there are over 150 NoSQL databases with diverse features and optimizations [1], and a number of NoSQL databases provide all new features and advantages while keeping data consistent or even eventually consistent, depending on the system needs [4]. For example, MongoDB[1], DynamoDB[2] and SimpleDB[3] supports strong and eventual consistency, and CouchDB[4] provides the feature of the eventual consistency. Furthermore, in order to increase execution speed of querying, non-relational databases began to use volatile memory. Since I/O data access is slower, mapping database or its parts into volatile memory increases performance and reduces the overall execution time of querying.

Yet, although the use of non-relational databases has increased over past years, their capabilities have not been fully disclosed. In order to choose a database that would be more appropriate for a specific business, it is important to understand its main characteristics. Similar to relational databases, each NoSQL database provides different mechanisms to store and retrieve data, which directly affects performance. Each non-relational database has also different optimizations, resulting in different data loading time and execution times for reads or updates. The performed evaluation allows us to compare different types of NoSQL databases, and test the execution times of *read* and *update* operations.

We tested five popular NoSQL databases: Cassandra[5], HBase[6], MongoDB[7], OrientDB[8] and Redis[9], and evaluate their execution speeds for different types of requests. Although there are a variety of solutions available for different types of NoSQL databases, those five NoSQL databases are most popular solutions for respective type. Some of the other popular NoSQL solutions are BigTable [19] (used as reference for HBase and Cloudata[10]), DynamoDB, Couchbase[11] Server, etc. During evaluation we used a benchmark with a typical range of workloads, Yahoo! Cloud Serving Benchmark [5], which provides execution of *get* and *put* operations, allowing to better understand the performance of a specific database: e.g. if it is faster for *reads* or *inserts*. The analysis and comparison of the results allowed us to verify how the different features and optimizations influence the performance of these databases.

In our previous work [6, 7], we evaluated the scalability of Cassandra and compared Cassandra and MongoDB. Differently, in this paper, we compared a higher number of the databases, and this work allowed us to understand which NoSQL database performs better in terms of operation types.

The remainder of this paper is organized as follows. The next section describes NoSQL databases. Section 3 presents the setup used for the evaluation of the NoSQL databases. The experimental evaluation is performed in the section 4. The section 5 discusses related work. Finally, in section 6 we present our conclusions and suggest future work.

2 NoSQL Databases

NoSQL databases are based on BASE (Basically Available, Soft State, and Eventually Consistent) principle that is characterized by high availability of data, while sacrificing its consistency [8, 9, 11]. On the other hand, relational databases are represented by ACID (Atomic, Consistent, Isolated, and Durable) principle where all the transactions committed are correct and do not corrupt database, and data is consistent [8]. Both principles come from the CAP theorem - Consistency, Availability, and Partition Tolerance [12]. According to this theorem, when it comes to working with distributed systems, only two of the three guarantees (C, A or P) can be achieved, so we need to choose the most important. When the consistency of data is crucial, relational databases should be used. When comparing these two models, it may be considered that BASE is more flexible than ACID. When data is distributed across multiple servers, the consistency becomes hard to achieve. NoSQL databases can be divided into four categories according to different optimizations [13]:

- Key-value store. In this type of databases all the data is stored as a pair of key and value. This structure is also known as “hash table”, where data retrieval is usually performed by using key to access value.
- Document Store. Such databases are designed to manage data stored in documents that use different format standards, such as, XML [15] or JSON [11]. This type of storage is more complex in comparison to storage used by Key-value Stores.

[1] MongoDB: http://www.mongodb.org/
[2] DynamoDB, http://aws.amazon.com/dynamodb/
[3] SimpleDB: http://aws.amazon.com/simpledb/
[4] CuchDB: http://couchdb.apache.org
[5] Cassandra: http://cassandra.apache.org/
[6] Hbase: http://hbase.apache.org/
[7] MongoDB: http://www.mongodb.org/
[8] OrientDB: ttp://www.orientdb.org/
[9] Redis: http://redis.io/
[10] Cloudata: http://www.cloudata.org/
[11] Couchbase: http://www.couchbase.com/

- Column Family. Similar to RDBMS (Relational Database Management System), in this model all the data is stored as a set of rows and columns. Columns are grouped according to the relationship of data. When the data stored in some columns are often retrieved together, these columns are arranged in one group.
- Graph Database. The best use of these databases is when stored information can be represented in the form of a graph with interlinked elements, for example, social networking, road maps or transport routes.

Hence, Key-value Store databases would be more appropriate for the management of stocks and products, and data analysis in real time, due to the fact that these databases have good retrieving speed – retrieving values given specific keys - when the greatest amount of data can be mapped into memory. Document Store databases are a good choice when working with large amounts of documents that can be stored into structured files, such as text documents, emails or XML and CMS and CRM systems. Column Family databases should be used when the number of write operations exceeds reads, and this occurs, for example, during system logging. Finally, graph databases are more appropriate for working with connected data, for example, to analyze social connections among a set of individuals, road maps and transport systems.

In summary, NoSQL databases are built to easily scale across a large number of servers (by sharding/horizontal partitioning of data items), and to be fault tolerant (through replication, write-ahead logging, and data repair mechanisms). Furthermore, NoSQL supports achieving high write throughput (by employing memory caches and append-only storage semantics), low read latencies (through caching and smart storage data models), and flexibility (with schema-less design and denormalization). In addition, the different systems offer different approaches to issues such as consistency, replication strategies, data types, and data models.

The NoSQL databases evaluated in this paper are from the following categories:

- Cassandra and HBase: Column Family databases.
- MongoDB and OrientDB: Document Store databases.
- Redis: Key-value Store database.

In the next section we will describe the experimental setup, which are used during evaluation of the databases, and specify the benchmark and versions of the databases that were tested.

3 EXPERIMENTAL SETUP

For the experimental analysis, we used the YCSB - Yahoo! Cloud Serving Benchmark [5], which allows us to evaluate and compare the performance of NoSQL databases. This benchmark consists of two components: a data generator and a set of performance tests consisting, in a simplistic way, of *read* and *insert* operations. Each of the test scenarios is called workload and is defined by a set of features, including a percentage of read and update operations, total number of operations, and number of records used. The benchmark package provides a set of default workloads that may be executed and are defined by *read*, *update*, *scan* and *insert* percentages. Default workloads are: A (50% read and 50% update), B (95% read and 5% update), C (100% read), D (95% read and 5% insert), E (95% scan and 5% insert) and F (50% read and 50% read-modify-write). Our focus is on comparing execution speed of *get* and *put* operations, which are most used operations. Therefore, we only executed workloads A, C and an additional workload H, defined by us, which is 100% update. Table 1 shows the executed workloads and the respective operations.

Table 1: Executed Workloads

Workload	% Read	% Update
A	50	50
C	100	0
H	0	100

In order to evaluate the databases, we randomly generated 600,000 records, each with 10 fields of 100 bytes over the key registry identification, resulting in roughly 1kb total per record. The execution of workloads was made using 1000 operations, and this means that there were 1000 requests to the database under test, while varying the number of stored records and operations. There are other benchmarks available, such as, TPC-H or SSB, which could be used to evaluate database performance. We used YCSB because in a simplistic way NoSQL databases have only two operations: *get* and *put*, whereas TPC benchmarks are more suited for evaluation of SQL databases while executing decision support queries over non-synthetic data.

All the tests were executed on a Virtual Machine Ubuntu Server 32bit with 2GB RAM available, hosted on a computer with Windows 7 and a total of 4GB RAM. In this study, Graph databases have not been evaluated. Because as stated in [16], Graph databases

should not be evaluated according to the scenarios used in the analysis of the other types of NoSQL databases (Key-value Store; Document Store; Column Family), with requests formed by *read* and *update* operations. Usage of links between records requires a different approach, so there are specific benchmarks developed to evaluate the performance of Graph databases, such as, XGDBench [17].

During the experimental evaluation, we tested the following NoSQL databases, which are most used ones:

- Cassandra: Column Family database, version 1.2.1 (http://cassandra.apache.org/).
- HBase: Column Family database, version 0.94.10 (http://hbase.apache.org/).
- MongoDB: Document Store database, version 2.4.6 (http://www.mongodb.org/).

1. OrientDB: Document Store database, version 1.5 (http://www.orientdb.org/).

- Redis: Key-value Store database, version 2.6.14 (http://redis.io/).

4 EXPERIMENTAL EVALUATION

In the following subsections we present and analyze the execution times based on only *reads* and only *updates*, and both operations at the same time. We executed YCSB workloads A, C and H.

4.1 Evaluation over Workload A

Figure 1 show the results, in seconds, obtained while executing workload A that consists of 50% reads and 50% updates, over 600.000 records.

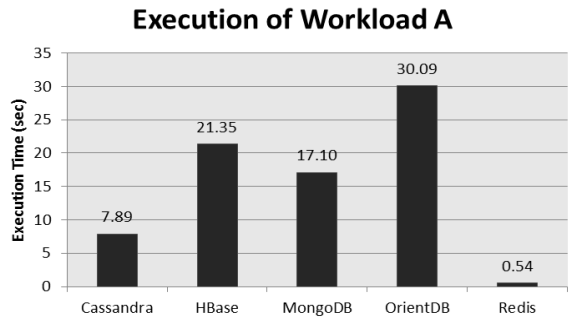

Figure 1: Execution time of workload A (50% reads and 50% updates over 600.000 records)

When analyzing the results of execution of workload A, a good performance is achieved by the Key-value Store database, Redis. This database highly uses volatile memory for data storage and retrieval, which allows lower execution time of requests. Among the tested databases of Column Family type, Cassandra exhibited a performance of 7.89 seconds, 2.70 times faster than HBase. The worst performance was presented by the Document Store database OrientDB (30.09 seconds), with an execution time 1.75 times higher compared to another Document Store database MongoDB. The worst execution time of OrientDB is due to the fact that records have to be read from disk, which is much slower in comparison to the volatile memory.

4.2 Evaluation over Workload C

Figure 2 shows the results obtained while executing workload C that consists of execution of 1000 read operations over 600.000 records.

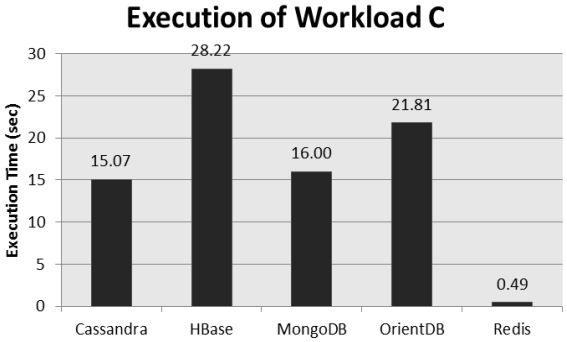

Figure 2: Execution time of workload C (100% reads over 600.000 records)

The results of the execution of workload C indicate that the Document Store databases, HBase and OrientDB, showed the slow execution time during read operations. HBase presented the worst result, and it is 1.86 times lower compared to the Column Family database Cassandra. Given a large number of records, HBase showed more difficulty during execution of *reads*. In HBase, parts of the same record may be stored in different disk files, and this results in an increased execution time. HBase is optimized for the execution of updates, but for reads, as we will see later on, HBbase shows a good performance over the workload H with 100% updates.

The second worst outcome was shown by OrientDB, which stores data records in disk and does not load data into memory. Redis had good execution time: it kept records in memory and thus showed minimal execution time for read operations. Database

Redis is projected to fast record retrieval using key due to mapping data in memory.

4.3 Evaluation over Workload H

Figure 3 shows the results of the execution of the workload H, which is 1000 updates over 600.000 records.

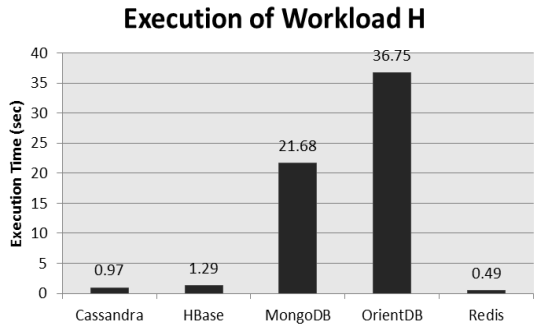

Figure 3: Execution time of workload H (100% update over 600.000 records)

With the execution of the workload H with 1000 updates, we observed better optimization of some databases for the execution of writes (in a simplistic way one update is one write). Two Column family databases, Cassandra and HBase, are optimized for performing updates: they load records as much as possible into memory, and thus the number of operations performed on the disk is reduced and the performance is increased.

Among the evaluated Document Store databases, OrientDB had the highest execution time with a total of 36.75 seconds, thus having a performance 1.69 times lower compared to the performance shown by MongoDB. The cause for this difference in execution time is the distinction of the storage type used by these databases: The OrientDB keeps records on disk rather than loading data into memory. The poor results of MongoDB are due to the use of locking mechanisms to perform update operations, and this increases execution time. Key-value Store databases are in-memory databases: they use volatile memory to map records, and thus database performance is increased significantly.

4.4 Overall Evaluation

Over previous subsections we presented results obtained over different workloads and data loading. In order to show more clearly the overall performance of these evaluated databases regardless of the type of performed operations, Figure 4 is generated. This figure shows the total execution time, values in seconds, for each of the tested databases. These values were obtained by summing the execution times of all workloads (A + C + H), and sorted in ascending order, from lowest execution time to highest.

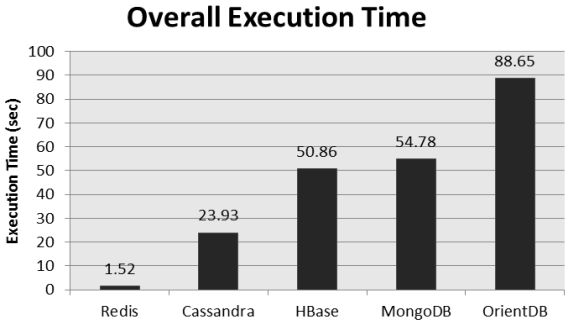

Figure 4: Overall execution time of workloads A+C+H

The overall results show that the in-memory database, Redis, had the best performance. Redis is a Key-value Store database and is highly optimized for performing *get* and *put* operations due to mapping data into RAM. It is well-known that in-memory databases are more efficient in query processing, but quantitative accuracy still lacks. One of our contributions in this study is presenting the quantitative results of the execution speed of the in-memory NoSQL database.

Cassandra and HBase, as Column Family databases, showed good update performance, since they are optimized for update operations. Nevertheless, from the overall evaluation results, those databases were more than 15 times slower than the Key-value Store database, Redis. Finally, Document Store databases had the worst execution times, and OrientDB is the database with the lowest overall performance. OrientDB was 1.61 times slower than MongoDB and had 58.32 times lower performance in comparison with Redis.

5 RELATED WORK

The concept of NoSQL was first used in 1998 by Carlo Strozzi to refer to an open source database that does not use SQL interface [18]. Strozzi prefers to refer to NoSQL as "noseequel" or "Norel" (non-relational), since it is the main difference between this technology and relational model. Its origin can also be related to the creation of Google's BigTable model [19]. This database system, BigTable, is used for storage of projects developed by Google, for example, Google Earth. Amazon subsequently developed its own system, Dynamo [20]. These projects allowed taking a step

towards the evolution of NoSQL. However, the term reemerged only in 2009, at a meeting in San Francisco organized by Johan Oskarsson [21]. The name for the meeting, "NoSQL meetup", was given by Eric Evans and from there on NoSQL became a buzzword.

Over the last years, NoSQL databases have been tested and studied, and their performance has been evaluated. There is a variety of papers, such as [22, 23, 24], which given overall analysis and presented theoretical approaches to describing characteristics and mechanisms of NoSQL databases. However, due to increased interests in non-relational technology, NoSQL databases have been analyzed not only from application perspective, but as enterprise ready and advantageous databases. Therefore, the research of their performance, characteristics and used mechanisms, has been increased. Some of those studies, such as [5], evaluate advantages of use of NoSQL technology by analyzing the throughput and the advantages that are brought by scalability of NoSQL databases. Different from these previous contributions, which evaluate throughput, we compare and analyzed the performance in terms of execution time of widely used NoSQL databases. Although in-memory databases are obviously more efficient than disk-based databases, but the efficiency and performance of NoSQL databases have not been compared quantitatively. Our work in this paper gives the quantitative results.

6 CONCLUSIONS AND FUTURE WORK

The popularity of NoSQL databases has increased as massive amounts of data are being collected and processed today. These databases bring a number of advantages, compared to relational databases, especially for large volumes of data, that are non-structured or semi structured. There are different types of NoSQL databases and each has its own set of features and characteristics, and these lead to the performance difference. The performance is an important factor for deciding which database will be used for enterprises and applications. Therefore, it is necessary to compare and analyze the execution time of difference NoSQL databases, and provide a performance reference.

In this paper, we evaluate five most popular non-relational databases from three types: Cassandra and HBase from Column Family databases, MongoDB and OrientDB from Document Store databases, and Redis from Key-value Store database. We use Yahoo! Cloud Serving Benchmark [5], and compare the execution times of these NoSQL databases over different types of workloads. Apart from the experimental evaluation, we also analyze the performance differences from the optimization mechanisms and data store approaches used by these databases.

The databases, which load data into volatile memory, like Redis, exhibited extremely fast response times regardless of workloads, due to the fast speeds of volatile memory compared to the extraction of the files stored on the hard drive. However, such databases depend on the amount of volatile memory, which is a much more expensive storage type compared to the disk.

The database with worst performance was the Document Store database, OrientDB, over different workloads. By analyzing obtained results we discover that this database requires more system capacities compared with the capacities provided by the environment used in the evaluation. Therefore, their performances were limited by the memory management, the operating system and the use of virtual machine environment.

HBase and Cassandra are databases that use a log for storing all performed changes, meanwhile the records are stored in memory for subsequent disk flush. The use of these mechanisms and following sequential writing to disk reduces the amount of disk operations that are characterized by low speed compared to the speed of the volatile memory. Thus, these databases are especially optimized for performing updates, while reads are more time consuming when compared with in-memory databases.

MongoDB is the database that showed largest increase in the execution time directly related to the increase of the number of updates performed. This database uses locking mechanisms, which increase execution time. On the other hand, the reads are not exclusive, so the mapping of records in memory increases performance. OrientDB performance also degraded with the increasing number of update operations.

As an overall analysis, in terms of optimization, NoSQL databases can be divided into two categories, the databases optimized for reads and the databases optimized for updates. Thus, MongoDB, Redis, and OrientDB are databases optimized to perform read operations, while Colum Family databases, Cassandra and HBase, have a better performance during execution of updates.

As future work, we will compare and analyze the performance of NoSQL databases further: we will increase the number of operations performed and run NoSQL databases over multiple servers. This evaluation will allow us to better understand how NoSQL behaves while running in distributed and parallel environments. We also plan to evaluate the performance of Graph databases.

ACKNOWLEDGMENTS

This work is financed by National Funds through FCT Foundation for Science and Technology (Fundação para a Ciência e a Tecnologia) in the framework of PEst-OE/EEI/UI0326/2014 project.

REFERENCES

[1] NoSQL - http://nosql-database.org/.

[2] Stonebraker, M.: SQL databases vs. NoSQL databases. Communications of the ACM, 53(4): 10-11, 2010.

[3] Gajendran, S.: A Survey on NoSQL Databases, http://ping.sg/story/A-Survey-on-NoSQL-Databases---Department-of-Computer-Science, 2012.

[4] Elbushra, M. M. and Lindström, J.: Eventual Consistent Databases: State of the Art. Open Journal of Databases (OJDB), RonPub, 1(1): 26-41, 2014. Online: http://www.ronpub.com/publications/OJDB-v1i1n03_Elbushra.pdf.

[5] Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R.: 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.

[6] Abramova, V. and Bernardino, J.: NoSQL databases: MongoDB vs Cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering* (C3S2E '13). ACM, New York, NY, USA, 14-22, 2013.

[7] Abramova, V., Bernardino, J. and Furtado, P., Testing Cloud Benchmark Scalability with Cassandra, in IEEE 10th World Congress on Services, Anchorage, USA, June 27 -- July 2, 2014.

[8] Pritchett, D.: BASE: An Acid Alternative. ACM Queue, 6(3): 48-55, 2008.

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

[10] S. Groppe, *Data Management and Query Processing in Semantic Web Databases*. Springer, May 2011.

[11] Massimo Carro, NoSQL Databases, CoRR, 2014. http://arxiv.org/abs/1401.2101.

[12] Browne, J.: Brewer's CAP Theorem, http://www.julianbrowne.com/article/viewer/brewers-cap-theorem, 2009.

[13] Indrawan-Santiago, M.: Database Research: Are We at a Crossroad? Reflection on NoSQL. Network-Based Information Systems (NBiS), 15th International Conference on Network-Based Information Systems, pp.45-51, 2012.

[14] Zhang, H. and Tompa, F.W.: Querying XML documents by dynamic shredding. InPro-ceedings of the 2004 ACM symposium on Document engineering (DocEng '04). ACM, New York, NY, USA, 21-30, 2004.

[15] Crockford, D.: JavaScript: The Good Parts. Sebastopol, CA: O'Reilly Media, 2008.

[16] Armstrong, T., Ponnekanti, V., Dhruba, B., and Callaghan, M.: LinkBench: a database benchmark based on the Facebook social graph. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). ACM, New York, NY, USA, 1185-1196, 2013.

[17] Dayarathna, M. and Suzumura, T.: XGDBench: A benchmarking platform for graph stores in exascale clouds. Cloud Computing Technology and Science (CloudCom), IEEE 4th In-ternational Conference on, Taipei, 363 – 370, 2012.

[18] NoSQL: a non-SQL RDBMS - http://www.strozzi.it.

[19] Chang, F., Jeffrey, D., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems, 26(2), Article 4, 2008.

[20] Decandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W.: Dynamo: amazon's highly available key-value store. In Proceedings of twenty-first ACM SIGOPS Symposium on Operating Systems principles (SOSP '07). ACM, New York, NY, USA, 205-220, 2007.

[21] YDN -- http://developer.yahoo.com/blogs/ydn/posts/2009/06/nosql_meetup/.

[22] Hecht, R. and JABLINSKI, S.: NoSQL Evaluation A Use Case Oriented Survey. Proceedings International Conference on Cloud and Service Computing, pp. 12-14, 2011.

[23] Han, J.: Survey on NOSQL Databases. Proceedings 6th International Conference on Pervasive Computing and Applications, pp. 363-366, 2011.

[24] Leavitt, N.: Will NoSQL Databases Live up to Their Promise?, Computer Magazine, 43(2): 12-14, 2010.

AUTHOR BIOGRAPHIES

Veronika Abramova is currently a researcher at CISUC – Centre for Informatics and Systems of the University of Coimbra. Previously she has received a bachelor degree at Instituto Superior Engenharia de Coimbra (ISEC). She is currently working in an industrial project, with an electricity sector company, focused on transferring part of the company's data to the non-relational storage. She has evaluated and studied different NoSQL databases, focusing on their performance comparison and characteristics. Her main research fields are business intelligence, big data, database knowledge management, NoSQL and SQL databases performance evaluation.

Jorge Bernardino received the PhD degree in computer science from the University of Coimbra in 2002. He is a Coordinator Professor at ISEC (Instituto Superior de Engenharia de Coimbra) of the Polytechnic of Coimbra, Portugal. His main research fields are big data, data warehousing, business intelligence, open source tools, and software engineering, and in these fields he has authored or co-authored dozens of papers in refereed conferences and journals. Jorge Bernardino has served on program committees of many conferences and acted as referee for many international conferences and journals. He was President of ISEC from 2005–2010. Actually, he is serving as General Chair of IDEAS'2014 conference and visiting professor at Carnegie Mellon University (CMU).

Pedro Furtado is Professor at University of Coimbra, Portugal, where he teaches courses in both Computer and Biomedical Engineering. His main research interests are data scalability and Big Data, data mining, service-oriented systems and real-time systems. Lately, his research has focused both on scalable and real-time warehousing, and also on middleware for wireless sensors in industrial and health-care applications. He has published books and more than 100 papers in international conferences and journals, and has several research collaborations with both industry and academia. Besides a PhD in Computer Engineering from U. Coimbra (UC) in 2000, Pedro Furtado also holds an MBA from Universidade Catolica Portuguesa (UCP).