On the State of NoSQL Benchmarks

Vincent Reniers, Dimitri Van Landuyt,
Ansar Rafique and Wouter Joosen
imec-DistriNet-KU Leuven
Celestijnenlaan 200A, Leuven, Belgium
firstname.lastname@cs.kuleuven.be

ABSTRACT

The proliferation of Big Data systems and namely NoSQL databases has resulted in a tremendous heterogeneity in its offerings. It has become increasingly difficult to compare and select the most optimal NoSQL storage technology. Current benchmark efforts, such as the Yahoo! Cloud Serving Benchmark (YCSB), evaluate simple read and write operations on a primary key. However, while YCSB has become the de-facto benchmark solution for practitioners and NoSQL vendors, it is lacking in capabilities to extensively evaluate specific NoSQL solutions.

In this paper, we present a systematic survey of current NoSQL benchmarks, in which we identify a clear gap in benchmarking more advanced workloads (e.g. nested document search) for features specific to NoSQL database families (e.g. document stores). Secondly, based on our survey, we discuss the strengths and weaknesses of the different benchmark design approaches, and argue in favor of a benchmark suite that targets specific families of NoSQL databases yet still allows overall comparison of databases in terms of their commonalities.

CCS Concepts

•**Information systems → Database performance evaluation;** *Database utilities and tools;*

Keywords

NoSQL benchmarks; YCSB; performance benchmarks

1. INTRODUCTION

Databases store and process increasingly large data sets generated from various online and offline sources. Traditionally, applications are supported by relational database management systems (RDBMSs) which show key limitations in horizontal scalability and elasticity [12, 3, 14]. Large Internet companies such as Amazon, Facebook and LinkedIn have identified these limitations and developed in-house storage alternatives such as DynamoDB and Cassandra [3, 17, 12]. Today, we know such databases as NoSQL, or "not only" SQL databases. NoSQL databases relax the strict ACID properties of RDBMSs in favor of horizontal and elastic scalability. Although NoSQL is an umbrella term for a wide variety of databases, these databases can be categorised according to the supported data model and four main broad subcategories are commonly defined: document stores, graph stores, column stores and key-value stores [8, 6, 14]. Each database is aimed at a data model (ranging from structured, to semi-structured to unstructured data) [8, 27]), and offers a corresponding set of functionality. For example, MongoDB is a document-based database which provides storage for JSON data and supports search on the nested structure. Cassandra is a column-family store for structured data with fast write performance. A plethora of NoSQL technologies has emerged as currently over 200 NoSQL technologies are in existence [21].

However, due to the sheer heterogeneity in database API's, terminology, and data models, it has become increasingly difficult to compare these technologies in terms of their features, database functionality, and especially performance. The Yahoo! Cloud Serving Benchmark (YCSB) [8] has emerged as the most-widely used NoSQL benchmark for comparing NoSQL databases on create, read, update, delete (CRUD) and scan operations.

However, application requirements exceed such simple operations, and some NoSQL databases do offer more advanced support, for example traversing relations in graph trees, applying aggregation functions and text search on nested structures. In this paper, we present our survey of existing NoSQL benchmark efforts taking into account: (i) their supported database technologies, (ii) data models, and (iii) the supported workload types. Based on our assessment, we discuss different design strategies and argue in favor of benchmark suites that encompass several specific application domains from graph stores to document stores, while still allowing for a general comparison between NoSQL databases in terms of common functionality.

The contributions of the paper are twofold: (i) we provide a survey of the current state in NoSQL benchmarks, and (ii) from this perspective, we discuss strategies towards an encompassing NoSQL benchmark suite and the challenges in data model and workload generation.

The remainder of this paper is structured as follows. First, Section 2 provides the background on NoSQL databases. Then, Section 3 discusses the assessment criteria used in our benchmark survey, which is then presented in Section 4. Sec-

ICPE '17 Companion, April 22-26, 2017, L'Aquila, Italy
© 2017 ACM. ISBN 978-1-4503-4899-7/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3053600.3053622

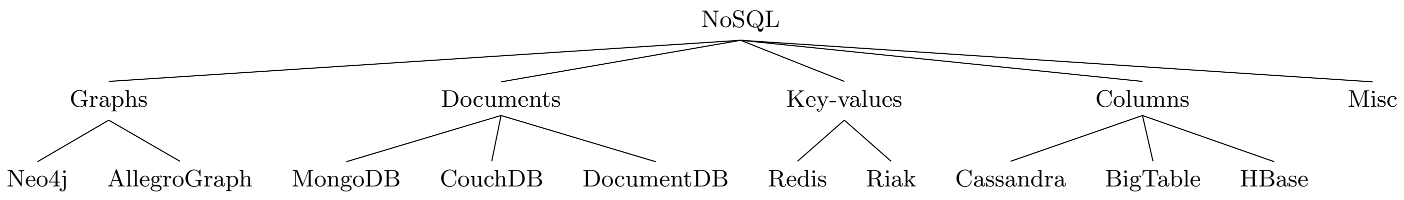

Figure 1: NoSQL databases categorised according to the supported data model

tion 5 presents a general discussion on aspects of improved benchmark design. Finally, Section 6 discusses related work and Section 7 concludes the paper.

2. NOSQL DATABASES

Traditional relational database systems (RDBMSs) are "one-size-fits-all" storage solutions, in the sense that their main aim is to provide general-purpose storage solutions [20], applicable to a wide variety of applications. These database systems however have been shown to inadequately deal with the challenges inherent to the domain of Big Data, which can be summarized as the 4V's: Volume, Veracity, Variety and Velocity [15]. In essence, RDBMSs fall short when a tremendous volume of data of large variety has to be processed at a fast velocity due to limitations in scalability and flexibility.

Contrary to RDBMS, NoSQL databases relax the strict ACID properties (*Atomicity*, *Consistency*, *Isolation* and *Durability* [27]), and adhere to the BASE acronym instead, meaning *Basically Available*, *Soft-state* and *Eventually consistent* [24, 6]. As such, NoSQL databases favor horizontal and elastic scalability over strong consistency. Horizontal scalability allows the database to increase in size while maintaining consistent performance. In addition, nodes can be added or removed to ensure elasticity, as to cope with for example peak loads.

In dealing with the variety of data, NoSQL databases support simplified and flexible data models, often schemaless, meaning that the data in each record does not necessarily have to adhere to pre-defined database schemas. However, NoSQL databases represent specialized solutions, tailored to specific use cases, and a categorisation of NoSQL databases is often made according to the supported data model. As depicted in Figure 1, four major categories exist: document stores, column-family stores for structured data, graph stores and key-value stores [6]. A consequence of this degree of specialization increased fragmentation: the query languages and API's of have become highly heterogeneous and technology- or database-specific. The general functionality of a NoSQL database is however determined by its data model. Graph stores will for example provide functionality for traversing graphs. While, document databases can store nested JSON files with the possibility for search and nested querying.

The next section determines some of the desired properties for a NoSQL benchmark, which are then applied in our assessment of the current state of the art.

3. BENCHMARK PROPERTIES

Considering the wide variety in database systems, use cases and data models that are associated to the NoSQL technology, we have defined three properties for a relevant and widely applicable NoSQL benchmark namely:

- **P1. Database support and applicability** How widely applicable is the benchmark? Is the benchmark focused on a specific family of NoSQL databases (e.g. wide-column stores, graph stores), multiple families, or to any NoSQL database? Does the benchmark remain relevant for specific application scenarios (e.g. document search)?
- **P2. Data models** Considering the large variety of data in Big Data applications and categories of data models in NoSQL, the benchmark has to generate various data sets that are aimed at testing the supported data models: graphs, regular key-value pairs, and even relational or connected and nested data. The generated data should be tailored for the purpose of evaluating the database and its model-associated functionality: e.g., by systematically varying the number of edges between nodes, varying the level of nesting depth in a JSON document, and the type of data used. In the ideal case, the user can supply the benchmark with sample data from a production environment, after which similar data is generated to allow relevant benchmarks for specific application scenarios. This property takes into account which data models are supported in the data generation step of the benchmark.
- **P3. Workloads** Closely tied to P2 are the workloads which can be executed on the inserted data set. When no support is provided for e.g. graph generation, no graph traversing workloads will be possible either. The workloads should be able to fully exploit the capabilities of each NoSQL database, while allowing comparison on a subset of mutually supported functionality between a certain set of NoSQL databases. For example, the user provides a workload description of its search queries and applies this workload to its data modelled as a graph, and as a document, to identify which database and data model can potentially be faster for search. This property takes into account which types of workloads are supported in the benchmark.

The next section presents our analysis of the current state of the art in NoSQL benchmarks in terms of the aforementioned properties.

4. THE STATE OF NOSQL BENCHMARKS

In the early 80's no standardized benchmarks for comparing databases existed [10, 1]. At that time mainly database-specific benchmarks were self developed by database vendors. A similar observation is made for NoSQL, namely

Redis, Aerospike, LevelDB, MarkLogic and EnterpriseDB all provide implementations of self-defined benchmarks comparing their respective offerings to competitors. The results from such benchmarks are often hard to understand, apply to other scenarios and induce comparison bias.

Nowadays, standardised organisations such as the Transaction Processing Council (TPC) and the Standard Performance Evaluation Corporation (SPEC) release standardised benchmarks according to strict guidelines and review processes. Unfortunately, as of yet none of these organisations provide benchmarks for NoSQL databases. However, TPC does provide benchmarks for Big Data technologies such as TPC-DS, TPCx-HS for Hadoop or TPCx-BB for Hive and Spark with the possibility for structured, semi-structured and unstructured data generation [29].

Despite the absence of a standardised NoSQL benchmark, several benchmark initiatives are presented in Table 1. Most NoSQL benchmarks are focused on a specific family of NoSQL storage systems (e.g. column, document stores). Most notably is the Yahoo! Cloud Serving Benchmark (YCSB), which has become the de-facto standard for cross-family comparison of NoSQL databases [8].

In the remainder of this section we compare benchmark initiatives per NoSQL category, starting with key-value benchmarks and YCSB. From this section we (i) identify gaps in specific family benchmarks, and (ii) determine whether or not cross-family benchmarks are in existence.

Key-value store benchmarks

When looking at the NoSQL categories in Figure 1, the commonality is that regardless of the specific data model, each database provides storage of a key-value pair. The value however varies as a collection of columns, a node with edges, a document or blob. To some degree, these databases can all be classified as key-value stores. Although generally key-value stores provide much more simplistic functionality such as get and put on ID.

The **Yahoo! Cloud Serving Benchmark (YCSB)** works on this very assumption. YCSB is able to execute workloads on read, write, update and scan operations on a primary key for various NoSQL databases such as Cassandra, MongoDB and HBase. A database interface layer is implemented for each database, which executes the intended (e.g. read, write) operation in the database's native query language. Data is generated consisting of synthetic strings with one string serving as the primary key. The workload executes the operation on a primary key with the associated fields.

Another key-value store benchmark KVZone [13] evaluates BerkeleyDB, Tokyo Cabinet, SQLite and Alphard. In contrast to YCSB, it has barely gained traction, while YCSB has seen wide-spread adoption. However, YCSB's workload and data generation is very simple and lacks capabilities to extensively evaluate the full database's capabilities.

Similar observations have been made and gave rise to several extensions by researchers in the shape of YCSB+T, YCSB++ and XGDBench [11, 28, 9]. YCSB+T is an extension which aims to evaluate the transactional overhead of database operations [11]. Although, not many NoSQL databases provide support for transactions and are limited to e.g. atomicity on single documents. YCSB++ is aimed at column stores, while XGDBench focuses on extensions for graph stores.

We discuss these and similar benchmarking initiatives per NoSQL category in the next sections.

Document store benchmarks

Similarly to YCSB is a workload generator called NoWog, however it allows user-specified workload descriptions on the basis of a generic grammar. The workload descriptions in an abstract grammer are then translated to database-specific operations, which results in CRUD operations executed on a set of supported database through mapping [22]. However, the functionality is limited to CRUD operations.

In [18] by Lungu et al. a simple benchmark tool was implemented with a small data generation component and read, write operations for MongoDB and MySQL.

The authors in [19] present SSB+, an extension on a decision support benchmark for relational databases, namely Star Schema Benchmark (SSB), and allow for the generated tables to be stored in a document and column-oriented databases. In addition, the OLAP SQL queries are translated to the respective NoSQL database. However, the process explained first generates normalized data and then denormalizes it to fit to the NoSQL paradigm. In addition a relational database benchmark is used as the starting point, which may not fully exploit the full capabilities of a NoSQL database.

Column store benchmarks

YCSB++ [28] is an extension on YCSB for benchmarking advanced features of scalable table stores (i.e. column-family stores) such as Apache HBase and Accumulo. The benchmark provides distributed synchronization between multiple benchmark clients and is able to measure eventual consistency, bulk loading, and the effect of optimizations for batch writing such as table pre-splitting. In addition, the benchmark can also measure the performance overhead of additional features such as access control and collects monitoring information on resource metrics at each cluster node.

The work by Pirzadesh et al. [23] is also similar to YCSB however focuses more on extensive range queries and conducts evaluations for Cassandra, HBase and Voldemort. A small component of YCSB is even used to create zipfian workload distributions.

Graph store benchmarks

Graph trees are commonly stored in-memory as the workload is characterised by random data access, for which persistent hard disks would incur a serious overhead [7]. However, to identify links between data in a distributed setting, techniques such as clustering have to be applied to reduce cross-node communication. Beis. et al [4] present a benchmark on the problem of community detection for Neo4j, OrientDB and Titan. Four workloads are provided on: massive insertion, single insertion, query workloads (graph traversal) and identifying communities in large networks through clustering.

Ciglan et al. [7] present a benchmark which evaluates graph traversal over graph databases, an implementation is provided for five graph databases. In [16] the authors present GDB, a distributed graph database benchmarking framework on operations such as exploring a node's neighborhood and finding the shortest path. XGDBench is an extension to YCSB for benchmarking graph stores by adding a graph data model and workload generator [9].

Table 1: NoSQL benchmarks and the capabilities in data set generation and workloads.

Applicability (P1)	Benchmarks	Data sets (P2)	Workloads (P3)
Key-value stores	YCSB [8], KVZone [13]	Key-value pairs of synthetic strings	CRUD + Range queries
Document stores	NoWog[22], Lungu et al.[18], SSB+[19]	Nested documents, limited generation options	Generic workload descriptions, CRUD, limited document queries
Column stores	YCSB++[28], Pirzadesh et al.[23], SSB+[19]	Structured data in tables with e.g. column families, limited generation options	CRUD, bulk loading, consistency evaluation, extensive range queries, limited table queries
Graph stores	Beis et al.[4], Ciglan et al.[7], GDB[16], XGDBench[9], LinkBench [2]	Multiple synthetic graph generators with e.g. varying clustering coefficients an edge connectivity preferences	Node and edge storage and retrieval, graph traversal, neighborhood discovery, clustering algorithms, ...

LinkBench is a database benchmark based on social graph data from Facebook which mimicks similar access patterns and operations [2]. It has seen evaluation for MySQL and HBase, however no NoSQL graph stores.

Various synthetic graph data generators are in existence for these benchmarks, which allow to generate graphs on properties of: e.g. clustering coefficient, diameter, and following e.g. a power-law degree distribution [7].

Conclusion.

From Table 1, we observe that benchmarking efforts for document and column-oriented stores are lacking in capabilities for workload and data generation. Graph store benchmarks show very diverse workloads (in e.g. cluster, graph traversal) and several data generators. Secondly, benchmark efforts that are tailored at measuring capabilities across NoSQL families are sporadic. As such, it is increasingly difficult to compare and facilitate the selection of a NoSQL technology regardless of a specific family.

5. TOWARDS A NOSQL BENCHMARK SUITE

In our ongoing research we require such a cross-family NoSQL benchmark to decide the location of data placement across various heterogeneous NoSQL databases on the basis of performance. However, such decision-making is currently very difficult due to the lack of cross-family NoSQL benchmarks. A case is also presented in LinkBench [2] by Facebook, which evaluates MySQL and HBase for deciding the best candidate for storing a social network. In addition, such an evaluation can potentially be extended with support for graph stores.

Currently, deciding upon a NoSQL technology regardless of the family is difficult, as cross-family benchmarks are lacking, with YCSB being the most prominent with solely support for CRUD operations and the generation of synthetic string data.

Therefore, in this section we outline our vision towards the creation of a more full-fledged benchmark suite, allowing cross-family comparison (Section 5.3), based on (i) our earlier experiences of extending the YCSB with search support [25] (Section 5.1) and (ii) our in-depth analysis of different suitable benchmark design approaches (Section 5.2).

5.1 Lessons learned from our own YCSB query benchmark

In previous research [25], the need arose to evaluate MongoDB's document query language. As evident from Section 4, no benchmarks are available which extensively evaluate MongoDB's query language, except for YCSB's simple operations on a primary key. In previous research we compared MongoDB's query language to MongoDB with SQL access facilitated through frameworks.

In facilitating a MongoDB query benchmark we adopted YCSB, since it first allowed comparison on simple CRUD metrics and allows for consistent generation of synthetic strings and operation calls according to various distributions. However, for search queries it lacked in data set generation on other values than strings, and connected structures. Secondly, there is no support for defining complex query operations.

We extended YCSB with a data generator generating names, birth dates, personnel numbers and other values. In terms of workload, we implemented a predefined set of read queries in SQL and MongoDB QL. However, this extension in YCSB quickly reached its limits in terms of extensibility on the workload generation (varying queries) and data generation, which kept repeating certain values after a while.

From this prototype we quickly realized the importance of **P2** (data models) and **P3** (workloads). First, data sets should be flexible enough to fully evaluate the specific capabilities of each database (graph connectivity, document nesting depth, relations, etc.). When no support is provided for e.g. graph generation, no graph traversing workloads will be possible either.

In addition for **P3**, the workloads should be able to fully exploit the capabilities of each NoSQL database, while allowing comparison on a subset of mutually supported functionality between a certain set of NoSQL databases, despite potentially originating from different NoSQL families. The limitation in functionality should be varied according to the selected databases for comparison, and should be as extensive as possible given the overlap in functionality.

5.2 Benchmark design strategies

Based on our survey of benchmarks, we clearly observe two different approaches to establishing a NoSQL benchmark, which differ in the dimension of property **P1**,

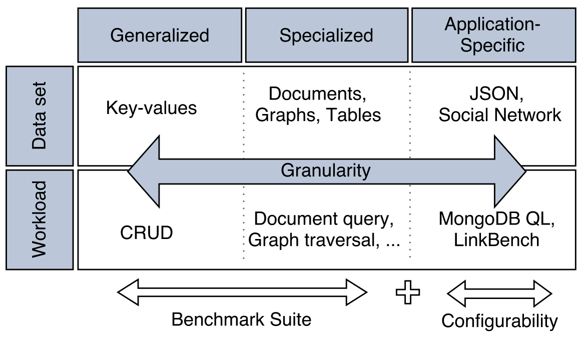

Figure 2: Data model and workload from a generalistic approach, to specialized and application-specific approaches

i.e. database support and applicability:

Generalistic approach In a *generalistic approach*, a broad set of NoSQL databases can be compared in terms of the common functionality between all NoSQL databases. As a result, performance benchmarks that are designed in such an approach are broadly applicable, since the generic data model and generic workloads can relatively easily be applied to a wide variety of databases. A key downside however is that these benchmarks do not assess the more specialized database functionality or more sophisticated workloads and as a result, may not yield informative or realistic benchmark results (low relevance). The YCSB benchmark is a prime example of this approach.

Specialized approach Specialized benchmarks focus on specific NoSQL data models, and may involve application-specific workloads that only make sense in specific NoSQL technologies. Examples benchmarks are XDGBench [9], focused on graph stores; YCSB++ [28], focused on column-family databases; TCP-DS [29], a MapReduce benchmark, etc. When the workloads and technologies tested in these benchmarks correspond to those of the technology adopter, these benchmarks are highly relevant and representative. However, comparison across DB families becomes impossible: these benchmarks are not suited for a broader search in the NoSQL space.

Integrated benchmark suite As depicted in Figure 2, the design approaches outlined above represent two extremes in the spectrum between applicability and relevance. We envision a third alternative, which is a fully encompassing NoSQL benchmark suite that supports combinations of generalistic and specialized comparisons. One the one hand, a broader set of NoSQL databases can be compared on a subset of common database functionality and in terms of generalistic workloads. In extension, an integrated benchmark suite also includes NoSQL-technology specific data models and workload generators. Thirdly, the user of the benchmark can configure the benchmark for application-specific workloads by providing workload and data descriptions simulating concrete use cases (e.g. representing a social network with graph traversal operations).

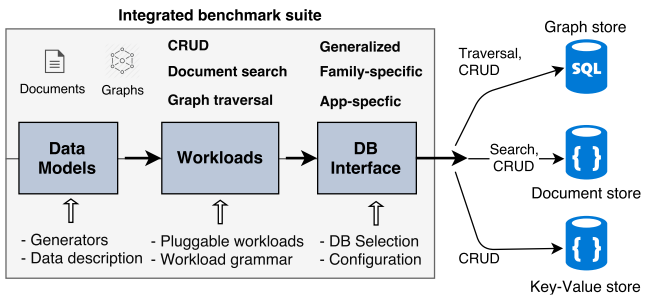

Figure 3: Benchmark suite

5.3 High-level design

Figure 3 showcases the high-level design of an integrated NoSQL benchmark suite. The benchmark supports a wide diversity in NoSQL databases and families according to **P1**. We elaborate on the three main components of data generation, workload generation and database interface.

The `Data Generator` generates various data from graph trees, nested documents to relational tables and simple key-value pairs. Attribute-level descriptions can be provided by the user to determine the selected data type and properties of e.g. node connectivity. The choice can be made to create a generator which acts upon a sample set of data or a predefined model similarly to the TCP benchmarks which modulate business and warehouse queries (OLTP and OLAP) on a predefined relational schema. BigDataBench provides in this regard an extension to a data generator which acts on sample data [1]. Pluggable support for existing generators should be a given as to integrate the numerous benchmarking efforts in a cohesive framework.

Secondly, the `Workload Generator` applies a multitude of workloads ranging from simple CRUD operations on a primary key, to graph traversals, node neighborhood queries, text searches, and from document retrievals to column updates. The workloads should be diverse and configurable enough to fully evaluate the selected database's their capabilities. To avoid conflicts, a specific compatible subset of functionality should be applied according to the set of targeted databases for evaluation. This subset should preferably be the maximum overlap in functionality theoretically possible, given the selected subset of NoSQL databases, while remaining fair and reproducible.

Finally, the `DB Interface` provides an interface for numerous workload operations. Each database implements this generic interface and maps the functionality to its native query language. Similarly, conversions should be applied for the generic data sets to the native data model. For example, a table may be provided which will have to be mapped to e.g. a document or graph. In essence this is a mapping component for the data sets and workload operations.

While the aforementioned components are the key components, the main goal of the framework should be to provide pluggability for existing workload and data generators as to combine existing efforts in this domain and further innovation.

6. RELATED WORK

Apart from the related NoSQL benchmarks discussed in Section 4, benchmarking efforts in the closely related domain of Big Data Analytics are of relevance as well. Han et al. [26] provide a survey on the current state of the art for Big Data Benchmarks. For such systems, various benchmarks have come into existence and specifically in the space of Big Data Analytics. Benchmarking this functionality is possible with TCP or other efforts by academia and industry such as BigDataBench. BigDataBench is a benchmark suite aimed at five different application domains of: search engine, social networks, e-commerce, multimedia analytics, and bioinformatics [1, 5]. In dealing with the heterogeneity of the supported database or more specifically Big Data Analytic platforms, the workloads are sometimes only applicable to a subset of the platforms. This is a trade-off which allows to properly evaluate each platform's individual capabilities.

In terms of data model, the suite comes with an extensive data model generator for structured, semi-structured and unstructured data and can even use sample data as a starting point. From this context, there is a lot to learn from this benchmark suite towards a benchmark for NoSQL storage systems.

7. CONCLUSION

The majority of benchmarks are complementary initiatives that are aimed at specific NoSQL technologies or technology families, concrete data models or specific application workloads. We identified that such efforts have seen the largest development in graph stores concerning the generation of synthetic graphs and various workloads. However, document and column-store benchmarks are specifically still lacking in these capabilities.

We conclude that cross-family NoSQL benchmark initiatives are currently limited to YCSB, although the question of storing graph data in a document store, RDBMs or graph store can be a relevant question. However, the tools and means to evaluate and aid in this decision making are currently non-existent.

We argue that there is a current lack of, and thus a strong potential for better integration of NoSQL performance benchmarking efforts, and we outline our vision and initial ideas on the design of such a benchmark suite. In future work, we will further develop and refine these ideas, leading to the development of an open and extensible, technology-aware collection of NoSQL-family-specific data generators, benchmark workloads, and in the longer run, benchmark efforts that are better suited for assessing the performance in the context of NoSQL and the database characteristics that set them apart.

8. ACKNOWLEDGMENTS

This research is partially funded by the Research Fund KU Leuven (project GOA/14/003 - ADDIS), the strategic basic research (SBO) project DeCoMAdS, and the imec-ICON SEClosed project.

9. REFERENCES

[1] Ahmad Ghazal et al. BigBench: towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. ACM, 2013.

[2] T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. LinkBench: a database benchmark based on the Facebook social graph. In *ACM SIGMOD '13*. ACM, 2013.

[3] C. Băzăr, C. S. Iosif, et al. The Transition from RDBMS to NoSQL. A Comparative Analysis of Three Popular Non-Relational Solutions: Cassandra, MongoDB and Couchbase. *Database Systems Journal*, 5(2):49–59, 2014.

[4] S. Beis, S. Papadopoulos, and Y. Kompatsiaris. Benchmarking graph databases on the problem of community detection. In *New Trends in Database and Information Systems II*. Springer, 2015.

[5] BigDataBench. Bigdatabench. http://prof.ict.ac.cn/BigDataBench/.

[6] R. Cattell. Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[7] M. Ciglan, A. Averbuch, and L. Hluchy. Benchmarking traversal operations over graph databases. In *Data Engineering Workshops (ICDEW)*. IEEE, 2012.

[8] 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*. ACM, 2010.

[9] M. Dayarathna and T. Suzumura. XGDBench: A benchmarking platform for graph stores in exascale clouds. In *Cloud Computing Technology and Science (CloudCom)*. IEEE, 2012.

[10] D. J. DeWitt. The Wisconsin Benchmark: Past, Present, and Future., 1993.

[11] A. Dey, A. Fekete, R. Nambiar, and U. Röhm. YCSB+T: Benchmarking web-scale transactional databases. In *ICDEW '14*. IEEE, 2014.

[12] Giuseppe DeCandia et al. Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007.

[13] S. Gokhale, N. Agrawal, S. Noonan, and C. Ungureanu. Kvzone and the search for a write-optimized key-value store. In *HotStorage*, 2010.

[14] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz. Data management in cloud environments: Nosql and newsql data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):1, 2013.

[15] IBM. IBM: The FOUR V's of Big Data. http://www-01.ibm.com/software/data/bigdata/.

[16] S. Jouili and V. Vansteenberghe. An empirical comparison of graph databases. In *Social Computing (SocialCom)*. IEEE, 2013.

[17] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[18] I. Lungu, B. G. Tudorica, et al. The Development of a Benchmark Tool for NoSQL Databases. *Database Systems Journal BOARD*, 13, 2013.

[19] Max Chevalier et al. Benchmark for OLAP on NoSQL technologies. In *RCIS '15*. IEEE, 2015.

[20] Michael Stonebraker et al. The End of an Architectural Era: (It's Time for a Complete Rewrite). VLDB '07. VLDB Endowment, 2007.

[21] NoSQL databases. NoSQL databases. http://www.nosql-database.org.

[22] Parinaz Ameri et al. NoWog: A Workload Generator for Database Performance Benchmarking. In *(DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2016.

[23] Pouria Pirzadeh et al. Performance evaluation of range queries in key value stores. *Journal of Grid Computing*, 10(1):109–132, 2012.

[24] D. Pritchett. BASE: An Acid Alternative. *Queue*, 6(3):48–55, 2008.

[25] V. Reniers, A. Rafique, D. Van Landuyt, and W. Joosen. Object-NoSQL Database Mappers: a benchmark study on the performance overhead. *Journal of Internet Services and Applications*, 8(1):1, 2017.

[26] Rui Han et al. Benchmarking big data systems: State-of-the-art and future directions. *arXiv preprint arXiv:1506.01494*, 2015.

[27] M. Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53(4):10–11, 2010.

[28] Swapnil Patil et al. YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011.

[29] TPC. Transaction Processing Performance Council. http://www.tpc.org/.