GooDBye: a Good Graph Database Benchmark - an Industry Experience

Piotr Matyjaszczyk
Poznan University of Technology
Poland
piotrmk1@gmail.com

Przemyslaw Rosowski
Poznan University of Technology
Poland
przemyslaw.rosowski@student.put.poznan.pl

Robert Wrembel
Poznan University of Technology
Poland
robert.wrembel@cs.put.poznan.pl

ABSTRACT

This paper reports a use-case developed for an international IT company, whose one of multiple branches is located in Poland. In order to deploy a graph database in their IT architecture, the company needed an assessment of some of the most popular graph database management systems to select one that fits their needs. Despite the fact that multiple graph database benchmarks have been proposed so far, they do not cover all use-cases required by industry. This problem was faced by the company. A specific structure of graphs used by the company and specific queries, initiated developing a new graph benchmark, tailored to their needs. With this respect, the benchmark that we developed complements the existing benchmarks with 5 real use-cases. Based on the benchmark, 5 open-source graph database management systems were evaluated experimentally. In this paper we present the benchmark and the experimental results.

1 INTRODUCTION

Among multiple database technologies [26], for a few years graph databases (GDBs) have gained their popularity for storing and processing interconnected Big Data. In the time of writing this paper there existed 29 recognized graph database management systems (GDBMSs), cf., [9], offering different functionality, query languages, and performance.

When it comes to selecting a GDB to suit an efficient storage of given graphs and efficient processing, a company professional has to either implement multiple 'proofs of a concept' or rely on existing evaluations of various databases. Typically, important assessment metrics include: (1) performance and (2) scalability w.r.t. a graph size and (3) scalability w.r.t. a number of nodes in a cluster.

In practice, assessing performance of IT architectures and particular software products is done by a benchmark. There exist multiple dedicated benchmarks for given domains of application. In the area of information systems and databases, the industry accepted and used benchmarks are developed by the Transaction Processing Council. There also exist dedicated benchmarks for non-relational databases and clouds, cf. Section 2.

Although there exist multiple benchmarks designed for graph databases, the motivation for our work came as a real need from industry, i.e., a large international IT company (whose name cannot to be revealed), having one of its multiple divisions located in Poland. The company stores large data volumes on various configurations of their software and network infrastructures. These data by virtue are interconnected and naturally form large graphs. Currently, these graphs are stored in flat files but in the future, they will be imported into a proprietary GDB and analyzed there. For this reason, a fundamental issue was to choose a GDBMS that would be the most suitable for particular 'graph shapes' and queries needed by the company. The assessment criteria included: (1) performance characteristics w.r.t. variable number of nodes in a cluster as well as (2) functionality and user experience.

The specific structures of graphs produced by the company and specific queries have not matched what was offered by the existing GDB benchmarks. These facts motivated the development of the *GoodBye benchmark*, presented in this paper. Designing *GooDBye* was inspired by [19] and it complements the existing graph database benchmarks mentioned before. The benchmark contributes real business use-cases.

The paper is structured as follows. Section 2 overviews benchmarks developed by research and industrial communities. Section 3 presents the benchmark that we developed. Section 4 outlines our test environment. Section 5 discusses the experimental evaluation of GDBs and their results. Finally, Section 6 summarizes the paper.

2 RELATED WORK

The performance of a database management system is typically assessed by means of benchmarks. Each domain of database application incurs its own benchmark. A benchmark is characterized by a given schema (structure of data) and different workload characteristics (query and data manipulation), and often by performance measures. Database benchmarking over years have received a substantial attention from the industry and research communities. Nowadays, the standard industry approved set of benchmarks for testing relational databases is being offered by the Transaction Processing Council (TPC) [31]. They support 2 main classes of benchmarks, namely: (1) TPC-C and TPC-E - for testing the performance of databases applied to on-line transaction processing, (2) TPC-H - for testing the performance of databases applied to decision support systems. Special benchmarks were proposed for testing the performance of data warehouses (e.g., [8, 15, 20, 27]).

[17] overviews the existing cloud benchmarks with a focus on cloud database performance testing. The author argues about adopting TPC benchmarks to a cloud architecture. [14] proposes a DBaaS benchmark with typical OLTP, DSS, and mixed workloads. [7] compares a traditional open-source RDBMS and HBase a distributed cloud database. [25] and [4] show the performance results of relational database systems running on top of virtual machines. [30] presents a high-level overview of TPC-V, a benchmark designed for database workloads running in virtualized environments.

Benchmarking of other types of databases, like XML (e.g., [23, 29]), RDF-based (e.g., [16]), NoSQL, and graph, received less interest from the research and technology communities in the past. However, with the widespread of Big Data technologies,

testing performance of various NoSQL data storage systems became a very important research and technological issue. In this context, [6] proposed Yahoo! Cloud Serving Benchmark (YCSB) to compare different key-value and cloud storage systems. [28] proposed a set of BigTable oriented extensions known as YCSB++.

In the area of GDBs, several benchmarks have been proposed so far. [3] advocated for using a large parameterized weighted, directed multigraph and irregular memory access patterns. In [10] the authors discussed characteristics of graphs to be included in a benchmark, characteristics of queries that are important in graph analysis applications, and an evaluation workbench. In the same spirit, problems of benchmarking GDBs were discussed in [5]. The authors explained how graph databases are constructed, where and how they can be used, as well as how benchmarks should be constructed. Their most important conclusions were that: (1) an increase in the size of a graph in most graph databases leads only to a linear increase of an execution time for highly centralized queries, (2) the same cannot be said for distributed queries, and (3) an important factor controlling throughput of highly distributed queries is the size of memory cache, and whether an entire graph structure can be fit in memory.

[1] described the so-called SynthBenchmark, which is included in the Spark GraphX library. It also offers a small graph generator. [2, 13] outlined a Java-based benchmark for testing social networks. Its data were stored in MySQL. The benchmark allowed to generate a graph of 1 billion of nodes, with its statistical properties similar to the one of Facebook.

[12, 22] proposed the Social Network Benchmark, focusing on graph generation and 3 different workloads, i.e., interactive, Business Intelligence, and graph algorithms.

[19] suggested and implemented a benchmark for a GDBMS working in a distributed environment. The authors attempted to create a holistic benchmark and - using the Tinkerpop stack - run it on a series of the most popular graph databases at that time, including Neo4j, OrientDB, TitanDB, and DEX. [11] evaluated the performance of four GDBs, i.e, Neo4j, Jena, HypergraphDB, and DEX with respect to a graph size, using typical graph operations. [24] focused on benchmarking 12 GDBs, i.e., Neo4j, OrientDB, InfoGrid, TitanDB, FlockDB, ArangoDB, InfiniteGraph, AllegroGraph, DEX, GraphBase, HyperGraphDB, Bagel, Hama, Giraph, PEGASUS, Faunus, NetworkX, Gephi, MTGL, Boost, uRiKA, and STINGER. This work tests performance of the majority of the GDBs but only in a centralized environment.

[18, 21] described a benchmark developed in the co-operation with 4 IT corporations and 4 universities. The benchmark consists of six algorithms: Breadth-first Search, PageRank, Weakly Connected Components, Community Detection using Label Propagation, Local Clustering Coefficient, and Single-source Shortest Paths. The data part includes real and synthetic datasets.

3 OUR APPROACH: GOODBYE - A GOOD GRAPH DATABASE BECHMARK

The GooDBye benchmark includes: (1) a parameterized graph data generator, (2) a graph database, and (3) queries that are to be run on it. In order to use the benchmark, a user needs to:

(1) run a data generator,
(2) decide which GDBMS is to be tested, and install it on a cluster,
(3) transform data generated by the benchmark into a form readable by the selected GDBMS,
(4) load the data into a GDB, using its proprietary tool,
(5) turn off database's caching mechanisms, as the same subset of queries will need to be repeated multiple times,
(6) run queries on the GDB.

3.1 Graph data

A graph used in the benchmark is directed and cyclic, with a maximum cycle length of 2. The graph reflects typical software and hardware configurations in a large company. A node represents one of the three following data entities:

- a package - it is composed of objects; a package can be transformed into another package; all packages have the same structure (fields);
- an object - it is composed of fields; an object can be transformed into another object, similarly as a package;
- a field - a field can be transformed into another field, similarly as an object, all fields have the same simple elementary datatype.

An arc represents:

- a data transformation - packages can be transformed into other packages, objects into other objects, and fields into other fields; each transformation (identified by its ID) is represented at all three levels of data entities;
- a data composition - each package contains one or more objects, and each object contains one or more fields.

The data generator is parameterized and can produce graphs described by different statistics. For the benchmark application presented in this paper, the graph had the following statistics:

- the number of vertices: 911034, which represented 500 packages;
- the number of arcs: 3229158,
- the average number of objects in a package: 100 (binomial distribution n=8000, p=0.0125),
- the number of object categories (types): 2; 30% of objects belong to category A, 70% belong to category B,
- the average number of fields of objects in category A: 30 (binomial distribution n=1500, p=0.02),
- the average number of fields of objects in category B: 8 (binominal distribution n=400, p=0.02),
- the average number of incoming fields transformation arcs: 2.5 (binominal n=80, p=0.03125),
- 4% of arcs form single-arc cycles,
- 2% of arcs form two-arc cycles.

3.2 Queries

Eight queries were defined and implemented in the benchmark, as required by the company. Queries Q1-Q5 (described below) were demanded by the company. Q1-Q3 aim at checking how long it takes for a GDB to find neighbor vertices, as well as navigating via incoming and outcoming arcs. Q4 and Q5 check how fast a GDB finds nodes, having given in- and out-going arcs, to calculate an impact of changes on transformations. Q6-Q8 are typical queries that are defined in other benchmarks.

- **Q1** - it finds and returns all vertices that transform to node of type A, i.e., nodes that have an outgoing arc of type *Transformation* that is an incoming arc to A.
- **Q2** - it finds and returns all nodes that have an incoming arc of type *Transformation*, whose source is A.
- **Q3** - it counts all the vertices that are connected to a node by the *Transformation* arc lading to the node of type A and

computes the percentage of these nodes over the number of all vertices in the graph.
- **Q4** - it counts all direct neighbors of nodes connected by a given transformation type and returns the percentage of the entire graph they comprise.
- **Q5** - it counts all direct nodes connected by a given transformation type, including nodes adjacent to nodes of type A.
- **Q6** - it counts the number of incoming and outgoing arcs of every single node in the graph, and returns a total count for each of them. This models a degree calculation for the entire graphs.
- **Q7** - it returns all nodes in the database that have their attribute equal to given number.
- **Q8** - it computes the shortest path between two nodes.

4 TEST ENVIRONMENT

4.1 GBDMSs under test

The company imposed the following requirements on a GDBMS:

- to be run on either an open-source or academic licence,
- to be used in practice by industry (listed on the *DB-Engines* website [9]),
- to support at least 3 of the ACID properties,
- to be capable of running in a cluster of workstations.

Based on the aforementioned criteria, out of 29 available GDBMSs, the following were selected for evaluation: ArangoDB, TitanDB, JanusGraph, OrientDB, and Spark GraphX. One system we heavily considered using was Neo4j. According to *DB-Engines* ranking, at the time of writing this paper, it was the most popular GDBMS. Unluckily, we were unable to obtain an academic license of its full, 'enterprise' edition, providing among others distributed storage and processing. As such, we have decided not to test it, rather than unfairly assess its toy version.

4.2 Benchmark setup

The GDBMSs were installed in a micro-cluster composed of 9 physical machines. Each node was run by Ubuntu and had the following parameters: (1) 8GB RAM, (2) 457GB HDD, (3) Intel Core2 Quad CPU Q9650 3.00GHz, (4) graphic card: GT215. The machines were physically fully interconnected with each other, enabling direct communication whenever required. The logical connections depended on the database system used.

Depending on an experiment, there were 1, 3, 5, or 9 nodes used at any given time. Such cluster sizes allowed to use with 1, 2, 4, and 8 worker nodes, and 1 access/coordinator node. Data were partitioned as equally as possible between nodes, using data distribution mechanisms provided by each GDMS.

5 PERFORMANCE EVALUATION OF SELECTED GRAPH DATABASES

The goal of the experiments was to evaluate response time of the 8 queries outlined in Section 3.2, for the 5 GDBMSs under test. Each query was run twelve times on the same dataset, on 1, 3, 5, and 9 nodes. The highest and lowest measurements were discarded. An average value and standard error were calculated for the remaining measurements. Due to huge differences in performance between the tested GBDMSs, a logarithmic scale was used in charts. Below we discuss the obtained performance characteristics.

5.1 Results

The response time (elapsed) for Q1-Q8 was measured in milliseconds. Below we present the results and discuss the obtained performance characteristics.

Q1 - transformation sources for a given node. The response times for Q1 are shown in Figure 1. As we can observe, ArangoDB clearly outperforms all the other GDBMSs. GraphX is the only system for which query response time decreases with the increasing number of nodes. The performance of TitanDB and JanusGraph degrades with the increasing number of nodes - queries run on the 9-machine cluster take about twenty times more than when running on a single node.

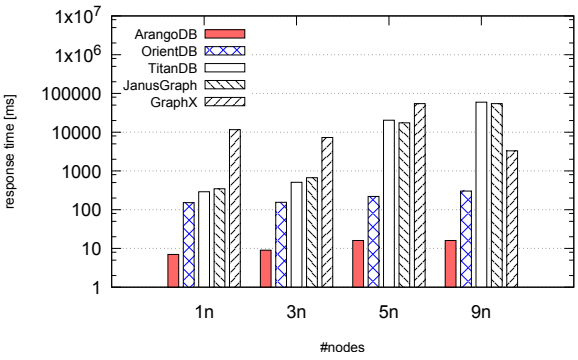

Figure 1: Execution time of Q1

Q2 - listing transformation destinations for a given node. The response times of this query are shown in Figure 2. In this test, ArangoDB, once again outperforms all the other GDBMSs, although the degradation of its performance when increasing the size of the cluster is more noticeable. GraphX execution times decrease by a factor of three, when a cluster size increases to 9, resulting in better execution times than TitanDB or JanusGraph, but still worse than OrientDB.

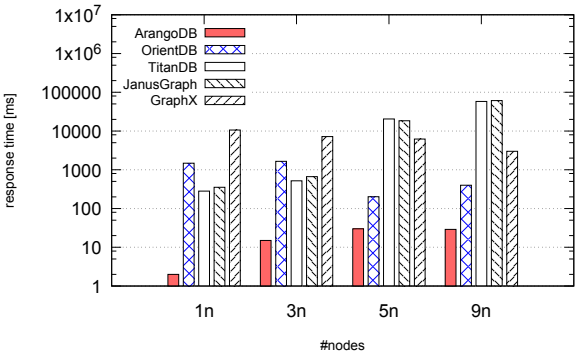

Figure 2: Execution time of Q2

Q3 - measuring the impact of changes in a node. The response times of this query are shown in Figure 3. From the chart we can notice that ArangoDB has a clear lead over its competitors yet again, with its executions taking thousands times less than of the other GDBMSs. GraphX slowly approaches ArangoDB execution times as the cluster size increases. OrientDB achieves better results than TitanDB and JanusGraph on clusters of greater size.

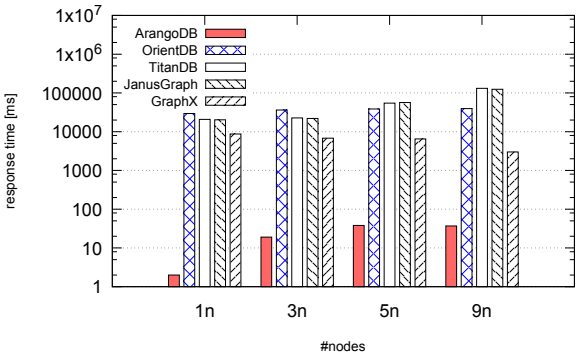

Figure 3: Execution time of Q3

Q4 - measuring the impact of changes in transformation. As Q4 query is very similar to Q3, the execution times shown in Figure 4 have characteristics similar to those shown in Figure 3, i.e., ArangoDB achieves the best results, with GraphX slowly decreasing ArangoDB lead as the cluster grows.

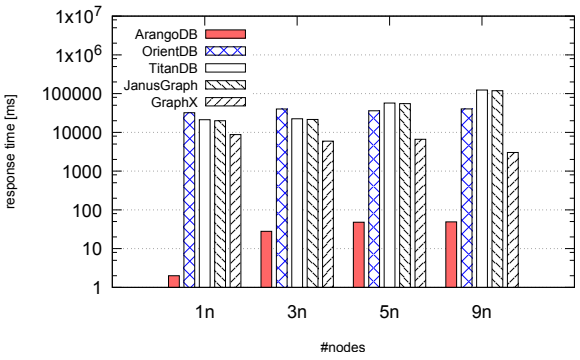

Figure 4: Execution time of Q4

Q5 - measuring the impact of changes in topology. The execution times from this experiment are shown in Figure 5. Once again, ArangoDB is in the lead. GraphX execution times decrease as the cluster size increases. OrientDB performs worse that TitanDB and JanusGraph for a cluster size up to 3 and performs better when the cluster grows.

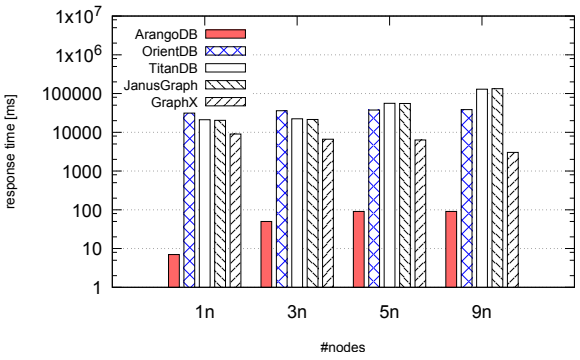

Figure 5: Execution time of Q5

Q6 - computing the degree of each node. The execution times from this experiment are shown in Figure 6. For this query, GraphiX offers the best performance, regardless of the cluster size. In a 3-, 5-, and 9-node cluster ArangoDB performs the worst. The performance of OrientDB remains unchanged regardless of the cluster size.

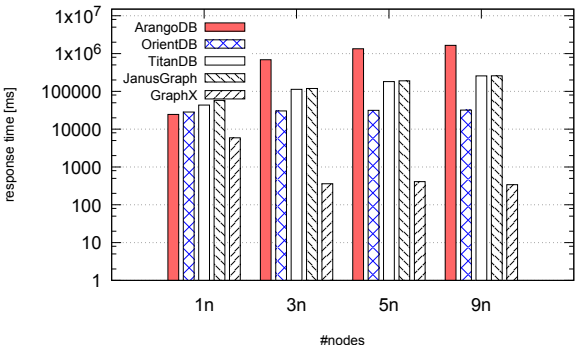

Figure 6: Execution time of Q6

Q7 - filtering query. The results of this evaluation are shown in Figure 7. On a single node, average execution times of the same query on ArangoDB and GraphX differ only by forty-five milliseconds, but as the cluster size increases GraphX gains noticeable lead over all the other GDBMSs.

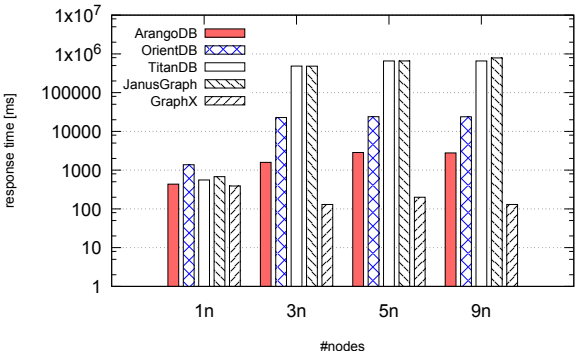

Figure 7: Execution time of Q7

Q8 - finding the shortest path between nodes. This experiment was run on ArangoDB, OrientDB, TitanDB, and JanusGraph. The reason for eliminating GraphX was caused by the implementation of the shortest path algorithm in GraphX. Rather than simply finding the shortest path between two nodes, it finds all the shortest paths from all the nodes to the target one, only then allowing users to select specific paths from a generated RDD. This heavily influences execution times of such queries. The first stage (computation of the shortest paths) takes minutes rather than milliseconds, and the second (retrieval of specific paths) takes a few milliseconds, making the results fairly incomparable to other GDBMSs. Figure 8 reveals that ArangoDB handles this query in the least amount of time. On a single node, OrientDB performs worse than TitanDB or JanusGraph, and performs better on 3, 5, and 9 nodes.

In Figure 9 we present total execution times of a workload composed of queries Q1, Q2, ..., Q7, for ArangoDB, OrientDB, TitanDB, JanusGraph, and GraphX in a cluster composed of 1, 3, 5, and 9 machines. As we can observe, ArangoDB, TitanDB,

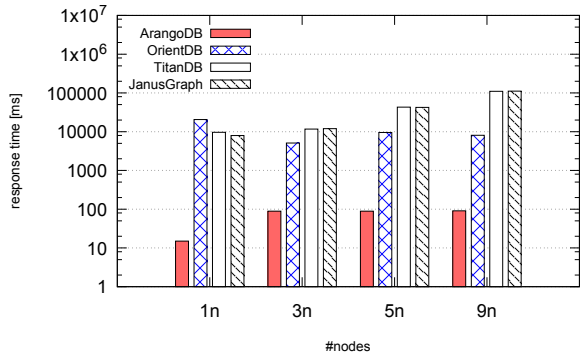

Figure 8: Execution time of Q8

and JanusGraph do not offer scaling out, as the total execution time grows with the increase of the number of machines. On the contrary, OrientDB and GraphX offer rather constant execution time w.r.t. the number of machines.

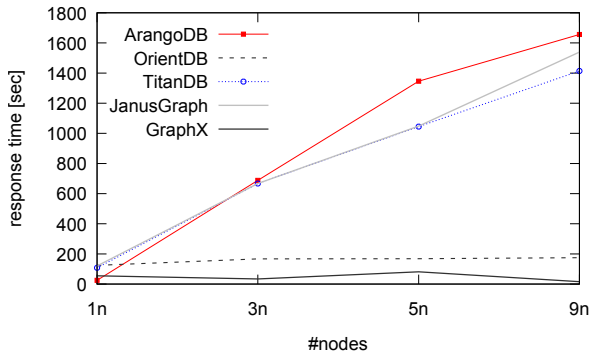

Figure 9: Total execution time of a workload composed of queries Q1 - Q7

5.2 Significance tests

From the presented charts we can observe that on the average, ArangoDB and GraphX offer the best performance. ArangoDB offers the best performance for all queries but Q6. Graphx achieves various results, with being a clear winner in Q6, but performing worse than ArangoDB for all other queries. Thus, we need to check whether it is statistically significant that:

- ArangoDB achieves better results for Q1-Q5 and Q7-Q8 than GraphX,
- ArangoDB achieves worse performance than GraphX for Q6,
- GraphX achieves better performance than OrientDB for Q6, since OrientDB is more efficient than ArangoDB in executing Q6.

To this end, we applied T-Student tests with p=0.01. The results of the significance tests are included in Table 1. The p-values for the significance of the results between GraphX and ArangoDB are represented by rows with queries Q1-Q5 and Q7, whereas p-values for the significance between GraphX and OrientDB are represented by row with Q6. Each row includes p-values for the experiments on 1, 3, 5, and 9 nodes.

As we can observe, the p-values are much lower than the assumed p-value of 0.01. This means, that the difference in execution times between ArangoDB, OrientDB, and GraphX are statistically significant for all 8 queries but Q7 on 1 node. It means that our conclusions are valid for Q1-Q8 except Q7 on 1 node.

Table 1: p-values for testing statistical significance of execution times between (1) GraphX and ArangoDB (Q1-Q5 and Q7) as well as between (2) GraphX and OrientDB (Q6)

Query	1 node	3 nodes	5 nodes	9 nodes
Q1	0.0000000000	0.0000000001	0.0000000003	0.0000000048
Q2	0.0000000054	0.0000000014	0.0000000000	0.0000000000
Q3	0.0000000423	0.0000000006	0.0000000000	0.0000000000
Q4	0.0000000740	0.0000000011	0.0000000000	0.0000000000
Q5	0.0000003403	0.0000000108	0.0000000000	0.0000000000
Q6	0.0000000000	0.0000000000	0.0000000000	0.0000000000
Q7	0.1674163506	0.0000000000	0.0000000000	0.0000000000

5.3 Functionality assessment

In this section we present our assessment of some features of the GDBMSs, related to user experience, grading each of them on a scale from 1 to 5 (1 being the lowest and 5 - the highest). The following features were assessed: (1) ease of installing and setting up the GDBMS, (2) ease of using the GDBMS (how complicated is its query language, whether it provides access to graph data from other languages), (3) support for multiple OS, and (4) visualization capabilities. The assessment results are shown in Table 2.

Table 2: Assessing functionality of GDBMSs

	ArangoDB	OrientDB	TitanDB	JanusGraph	GraphX
Ease of setup	5	3	4	4	3
Ease of use	5	3	5	5	2
Portability	5	5	5	5	5
Interface	4	4	2	2	1
Total	19	15	16	16	11

As it can be seen, ArangoDB wins in this regard as well. Its installation is straightforward, setting up a cluster requires nothing but running a few, simple scripts. Its querying language is robust and intuitive, with a focus on sub-querying. It runs on most common operating systems. Its visual interface is decent.

TitanDB and JanusGraph are next in our ranking. Their installation and setting up a cluster require a bit of fiddling, although it does not require all that much skill. The query languages are easy to learn and use. Both of these GDBMSs do not expose any problems of running on any of the popular operating systems. They do require quite a lot of work to set up any kind of visual interface.

OrientDB scores third. Its installation is not difficult, although having a few instances in a cluster is problematic. Its language lacks a few built-ins. It supports numerous OSs. Visual representations of graphs it generates are decent and legible.

GraphX scores last. Ease of use was never the focus for Spark-based tools. Installation and cluster set up is rather easy, but connecting it to a resilient data storage is more difficult. Tutorials for GraphX are almost non-existent, and documentation occasionally leaves a bit to be desired. Since it is Java-based, it has no problems running virtually anywhere. Graphical interface (other than Spark management tool) is nonexistent.

6 SUMMARY AND CONCLUSIONS

In this paper we presented a graph database benchmark developed to meet specific requirements of an international IT company. Even though over 10 graph benchmarks have been proposed in the research literature, none of them reflects the particular structure of the graph or particular queries needed by the IT company. Therefore, the benchmark that we developed can be considered as a complementary to those mentioned in Section 2. It contributes another graph structure used by industry and five queries used by industry.

The benchmark was implemented and used in practice to asses the performance of 5 open-source GBDMSs in a micro-cluster composed variable number of physical nodes (up to 9 nodes were used). The experiments that we run showed that:

- distributing graph data into multiple nodes does not provide scaling out; we observed that: (1) query execution times increased when the size of the cluster increased (the case of ArangoDB, TitanDB, and JanusGraph) or remained approximately constant (the case of OrientDB and GraphX);
- even simple queries can take much longer to execute in a cluster when a GDB needs to cross-check every node for arcs leading to another shard;
- ArangoDB offers the best performance in the majority of tests; it also offers the best functionality from a user perspective;
- GraphX offers the best performance when it comes to massive localized data processing (cf. Figure 6), i.e., it is a good match for certain algorithms such as PageRank, that are highly interested in degrees of nodes.

The performance evaluation can further be extended to test the scalability of GDBMSs w.r.t. graph size and clusters of sizes greater than 9 nodes. To this end, the proposed *GoodBye benchmark* needs to be further extended as well, to generate graphs of parameterized size and multiple statistical properties.

REFERENCES

[1] Apache. [n.d.]. SynthBenchmark. Apache,https://github.com/apache/spark/blob/master/exam-ples/src/main/scala/org/apache/spark/examples/graphx/SynthBenchmark.scala.

[2] Timothy G. Armstrong, Vamsi Ponnekanti, Dhruba Borthakur, and Mark Callaghan. 2013. LinkBench: A Database Benchmark Based on the Facebook Social Graph. In *SIGMOD Int. Conf. on Management of Data*.

[3] D. Bader, J. Feo, J. Gilbert, J. Kepner, D. Koetser, E. Loh, K. Madduri, B. Mann, T. Meuse, and E. Robinson. 2009. HPC Scalable Graph Analysis Benchmark. HPC Graph Analysis, http://www.graphanalysis.org/benchmark/.

[4] Sharada Bose, Priti Mishra, Priya Sethuraman, and H. Reza Taheri. 2009. Benchmarking Database Performance in a Virtual Environment. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*. 167–182.

[5] M. Ciglan, A. Averbuch, and L. Hluchy. 2012. Benchmarking Traversal Operations over Graph Databases. In *Int. Conf. on Data Engineering Workshops*.

[6] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *ACM Symposium on Cloud Computing*. 143–154.

[7] Jean-Daniel Cryans, Alain April, and Alain Abran. 2008. Criteria to Compare Cloud Computing with Current Database Technology. In *Int. Conf. Software Process and Product Measurement*. 114–126.

[8] Jerome Darmont, Fadila Bentayeb, and Omar Boussaid. 2007. Benchmarking Data Warehouses. *Int. Journal of Business Intelligence and Data Mining* 2, 1 (2007).

[9] DB-ENGINES. [n.d.]. DB-Engines Ranking of Graph DBMS. https://db-engines.com/en/ranking/graph+dbms.

[10] David Dominguez-Sal, Norbert Martinez-Bazan, Victor Muntes-Mulero, Pere Baleta, and Josep Lluis Larriba-Pay. 2011. A Discussion on the Design of Graph Database Benchmarks. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*.

[11] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. 2010. Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark. In *Int. Conf. on Web-age Information Management (WAIM)*.

[12] Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter Boncz. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *SIGMOD Int. Conf. on Management of Data*.

[13] Facebook. [n.d.]. LinkBench. GitHub,https://github.com/facebookarchive/linkbench.

[14] Avrilia Floratou, Jignesh M. Patel, Willis Lang, and Alan Halverson. 2011. When Free Is Not Really Free: What Does It Cost to Run a Database Workload in the Cloud?. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*. 163–179.

[15] Florian Funke, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Anisoara Nica, Meikel Poess, and Michael Seibold. 2012. Metrics for Measuring the Performance of the Mixed Workload CH-benCHmark. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*.

[16] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A Benchmark for OWL Knowledge Base Systems. *Web Semantics* 3, 2-3 (2005).

[17] Karl Huppler. 2011. Benchmarking with Your Head in the Cloud. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*. 97–110.

[18] Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat-Pérez, Thomas Manhardto, Hassan Chafio, Mihai Capotă, Narayanan Sundaram, Michael Anderson, Ilie Gabriel Tănase, Yinglong Xia, Lifeng Nai, and Peter Boncz. 2016. LDBC Graphalytics: A Benchmark for Large-scale Graph Analysis on Parallel and Distributed Platforms. *VLDB Endownment* 9, 13 (2016).

[19] S. Jouili and V. Vansteenberghe. 2013. An Empirical Comparison of Graph Databases. In *Int. Conf. on Social Computing*.

[20] Martin L. Kersten, Alfons Kemper, Volker Markl, Anisoara Nica, Meikel Poess, and Kai-Uwe Sattler. 2011. Tractor Pulling on Data Warehouses. In *Int. Workshop on Testing Database Systems*.

[21] LDBCouncil. [n.d.]. LDBC Graphalytics. GitHub,https://github.com/ldbc/ldbc_graphalytics.

[22] LDBCouncil. [n.d.]. Social Network Benchmark. LDBCouncil,http://ldbcouncil.org/developer/snb.

[23] Hadj Mahboubi and Jérôme Darmont. 2011. XWeB: the XML Warehouse Benchmark. *CoRR* (2011).

[24] Robert McColl, David Ediger, Jason Poovey, Dan Campbell, and David A. Bader. 2014. A performance evaluation of open source graph databases. In *Workshop on Parallel Programming for Analytics Applications*.

[25] Umar Farooq Minhas, Jitendra Yadav, Ashraf Aboulnaga, and Kenneth Salem. 2008. Database systems on virtual machines: How much do you lose?. In *Int. Conf. on Data Engineering Workshops (ICDE)*. 35–41.

[26] ODBMS. [n.d.]. Operational Database Management Systems - ODBMS. http://www.odbms.org/.

[27] Patrick O'Neil, Betty O'Neil, and Xuedong Chen. 2009. Star Schema Benchmark. https://www.cs.umb.edu/ poneil/StarSchemaB.PDF.

[28] Swapnil Patil, Milo Polte, Kai Ren, Wittawat Tantisiriroj, Lin Xiao, Julio López, Garth Gibson, Adam Fuchs, and Billie Rinaldi. 2011. YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In *ACM Symposium on Cloud Computing*. 9.

[29] Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu, and Ralph Busse. 2002. XMark: A Benchmark for XML Data Management. In *Int. Conf. on Very Large Data Bases*.

[30] Priya Sethuraman and H. Reza Taheri. 2011. TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments. In *TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC)*. 121–135.

[31] TPC. [n.d.]. Transaction Processing Council Benchmarks. http://www.tpc.org/.