Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 30, Num. 2 (2023) 132-140

RESEARCH ARTICLE

Energy Consumption and Performance Evaluation of Multi-Model NoSQL DBMSs

Avaliação de Desempenho e Consumo de Energia de SGBDs NoSQL Multimodelos

Fúlvio Falcão[1], João Moura[1], Gabriel Silva[1], Carlos Araujo[1], Erica Sousa[2], Eduardo Tavares[1]*

Abstract: New applications have required data storage using multiple data models, which are usually known as polyglot persistence applications. Their implementation is often complex, as the system must simultaneously manage and store data in multiple database management systems (DBMS). Over the years, multi-model DBMSs have been conceived, which commonly integrate multiple NoSQL data models into a single system. To demonstrate their feasibility, some researches have evaluated multi-model NoSQL DBMSs in the context of performance, but energy consumption is usually not taken into account. Indeed, energy consumption is an issue that should not be neglected due to the respective cost and environmental sustainability. This paper presents a performance and energy consumption evaluation of multi-model and single-model NoSQL DBMSs, more specifically, ArangoDB (multi-model), OrientDB (multi-model), MongoDB (document) and Redis (key-value). The experiments are based on Yahoo! Cloud Serving Benchmark (YCSB), and results demonstrate energy consumption may vary significantly between the evaluated DBMSs for different commands (e.g., read) and workloads. The proposed evaluation contributes to the state of the art, as storage system designers have additional insights regarding the behavior of multi-model NoSQL DBMSs for distinct workloads and energy usage.
Keywords: performance evaluation — NoSQL databases — data storage systems — energy consumption

Resumo: Novas aplicações computacionais têm adotado armazenamento de dados usando múltiplos modelos de dados, e estas são usualmente conhecidas como aplicações de persistência poliglota. As respectivas implementações são complexas, pois o sistema precisa gerenciar e armazenar dados simultaneamente em múltiplos sistemas de gerenciamento de banco de dados (SGBDs). Ao longo dos anos, SGBDs multimodelos foram concebidos, os quais contemplam múltiplos modelos de dados NoSQL em um único sistema. Algumas pesquisas avaliam SGBDs NoSQL multimodelos no contexto de desempenho, mas o consumo de energia não é levado em consideração usualmente. Consumo de energia é uma métrica que não pode ser desconsiderada devido ao respectivo custo e impacto ambiental. Este artigo apresenta uma avaliação de desempenho e consumo de energia de SGBDs NoSQL com único modelo e multimodelos, mais especificamente, ArangoDB (multimodelo), OrientDB (multimodelo), MongoDB (documento), e Redis (chave-valor). Os experimentos utilizam o YCSB benchmark, e os resultados demonstram que o consumo de energia pode variar significativamente entre os SGBDs avaliados para diferentes comandos (e.g., leitura) e cargas de trabalho. Este trabalho contribui para o estado da arte, pois projetistas de sistemas de armazenamento terão perspectivas adicionais sobre o comportamento de SGBDS NoSQL multimodelos para diferentes cargas de trabalhos e consumo de energia.
Palavras-Chave: avaliação de desempenho — banco de dados NoSQL — sistemas de armazenamento de dados — consumo de energia

[1] *Centro de Informática, Universidade Federal de Pernambuco, Brazil*
[2] *Universidade Federal Rural de Pernambuco, Brazil*
***Corresponding author**: eagt@cin.ufpe.br
DOI: http://dx.doi.org/10.22456/2175-2745.136568 • **Received:** 03/11/2023 • **Accepted:** 07/02/2024

1. Introduction

The evolution of computing applications, such as social networks and Internet of Things, has resulted in the generation of a large amount of structured, semi-structured and unstructured data [1]. Using data mining tools, useful information can be extracted for decision-making, but this is a challenging task. Indeed, one of the challenges for the database (DB) research

community in recent years has been the development of technologies to manage this large amount of heterogeneous data [2].

Relational DB technology may be unsuitable for dealing with highly heterogeneous data due to the high cost of consistency control. Such an issue encouraged the development of new solutions, more specifically, NoSQL database management systems (DBMS) [3]. This technology deals with distributed storage, eventual consistency and distinct data models (e.g., key-value, column, document, graph).

As a response to the need for simultaneously managing multiple data formats, multi-model DBMSs have been conceived [4]. In this case, a single DBMS stores, indexes and queries distinct data models, and such a feature is commonly termed polyglot persistence [3]. Multi-model DBMS significantly simplifies the development of applications requiring polyglot persistence, but system performance may be impacted. As a consequence, some researches have evaluated multi-model NoSQL DBMSs in the context of performance, but energy consumption is usually not taken into account.

Energy consumption is an issue that should not be neglected due to the respective cost and environmental sustainability. By 2050, data centers may consume three times more energy than the total power generation in Japan [5]. Data storage is also a prominent part of data centers, and the respective subsystems may be responsible for more than 27% of energy consumption [6].

This paper presents a performance and energy consumption evaluation of multi-model and single-model NoSQL DBMSs, more specifically, ArangoDB (multi-model), OrientDB (multi-model), MongoDB (document) and Redis (key-value). The experiments are based on Yahoo! Cloud Serving Benchmark (YCSB), in which insertion, read and update commands are evaluated for each adopted DBMS. A measurement framework is also presented, namely, measurement server, which includes hardware components based on Arduino and a software tool to estimate energy consumption of computing systems.

Our evaluation provides a prominent contribution to the state of the art, as storage system designers have additional insights regarding the behavior of multi-model NoSQL DBMSs in the context of energy utilization and performance. The results may help in decision-making regarding operational costs and capacity planning.

The rest of this paper is organized as follows. Section 2 summarizes related work and Section 3 presents an overview of NoSQL DBMS for polyglot persistence. Section 4 describes the methodology and Section 5 presents experimental results. Finally, Section 6 concludes this work.

2. Related Work

Over the last years, some works have evaluated the performance of NoSQL DBMSs, and few researches have considered multi-model database systems. Besides, energy consumption is usually not taken into account.

Swaminathan and Elmasri [7] assessed the performance of 3 prominent NoSQL DBMSs: MongoDB (document), Cassandra (column) and HBase (column). YCBS benchmark is adopted, and results indicate Cassandra outperformed other DBMSs in most workloads. In [8], the authors carry out a performance evaluation with OrientDB (multi-model), Neo4j (graph) and MongoDB. A cluster with 3 nodes is adopted, and distinct workloads are executed. The focus is on query execution, and OrientDB provided close results to Neo4J and MongoDB, taking into account the respective data models. Rohmat [9] et al. adopted MongoDB, ArangoDB (multi-model) and CouchDB (document) for assessing response time considering create, read, update and delete (CRUD) operations. Such a work utilized YCSB benchmark, and MongoDB obtained the lowest mean time for read, update and delete operations. ArangoDB had the shortest response time for insertion.

Gomes et al. [10] carried out performance and energy consumption evaluation of three NoSQL DBMSs: MongoDB, Cassandra and Redis (key-value). The experiment adopted YCSB tool, and read, write and update operations were taken into account, assuming a range of 1,000 to 100,000 operations. Results indicate no single DBMS that outperforms its counterparts for all benchmarks. In [11], the authors considered a comparative experiment, which evaluated Redis, MongoDB and Cassandra. The workload is based on YCSB assuming mixed (e.g., read and write) and single (e.g., read) commands. Overall, results point out Redis provides the shortest execution time, but MongoDB has better performance for insertion.

Martins et al. [12] compared seven NoSQL DBMSs: MongoDB, Cassandra, HBase, OrientDB, Voldemort (key-value), Memcached (key-value) and Redis. Voldemort, Memcache, and Redis obtained the best performance due to in-memory operations. OrientDB and MongoDB showed the slowest performance. In [13], the same authors extended their previous work [12], demonstrating Cassandra performs better for the adopted benchmarks, but such DBMS is very dependent on machine resources. On the other hand, MongoDB provided better results in a computer with constrained hardware.

In [14], the authors compared the performance of Redis and SSDB (key-value) DBMSs in a single node. The adopted workload mimics web applications, and, in general, SSDB had better throughput. In [15], HBase and Cassandra are evaluated on a virtual environment deployed on Openstack plataform. The focus is on execution time. In general, Cassandra has a higher fluctuating latency during read operations, and such DBMS also has larger runtime than HBase. Nothing is stated about energy consumption. Barros et al. [16] conducted a performance and energy consumption assessment of Cassandra DBMS in a distributed environment. YCSB benchmark is adopted, and the number of servers (nodes) is also a factor in the experiments. Results indicate the ideal number of nodes (balancing energy utilization and response time) is related to the data size.

In [17], the authors assessed the performance and energy

consumption of relational and NoSQL DBMSs, focusing on query tuning. The proposed evaluation describes important insights about energy savings, but multi-model NoSQL DBMSs are not taken into account. Additionally, such a work does not provide details concerning the mechanism for estimating energy consumption. Silva and Lima [18] carried out a performance evaluation of relational and NoSQL DBMSs on single-board computers. Although the authors emphasize the importance of low-power clusters, energy consumption is not considered in the experiments. Kaur et al. [19] analyzed the performance of polyglot persistence using distinct relational and NoSQL DBMSs. In that work, the authors do not describe the approach for measuring energy consumption, and multi-model DBMSs are not utilized.

Although all presented works deal with NoSQL and performance evaluation, few researches adopt multi-model DBMSs or energy consumption assessment. Our paper compares prominent multi-model and single-model NoSQL DBMSs, taking into account a representative benchmark and energy consumption. A measurement platform is also presented, which allows estimating the energy utilization of computing systems.

3. NoSQL DBMS and Polyglot Persistence

NoSQL is an abbreviation for "Not Only SQL" [20], which commonly contemplates DBMSs that do not fully comply with ACID (Atomicity, Consistency, Isolation and Durability) transactions and the relational data model. High performance and high availability are remarkable features of NoSQL DBMSs, and these attributes are usually achieved by avoiding the ACID properties.

A NoSQL DBMS adopts the BASE properties [21], which stands for Basically Available, Soft State and Eventually Consistent. In other words, the database is available all the time (basically available), inconsistency can be tolerated for a time period (soft state), and, eventually, the database will be consistent (eventual consistency).

NoSQL DBMSs commonly adopt alternative data models, such as column, key-value, document and graph [3]. Nowadays, several applications simultaneously require the storage of distinct data models, which is usually referred to as polyglot persistence. There are two common approaches for implementing polyglot data storage. One mechanism adopts separate data management systems, such that each DBMS implements a specific data model. However, the application must deal with additional complexity, since, for instance, each DBMS may have different communication interfaces (e.g., query language). On the other hand, multi-model DBMSs provide several data models simultaneously, in a single management system (polyglot persistence), allowing great flexibility. Besides, multi-model DBMSs avoid data duplication, and a user only needs to be concerned with one DBMS supplier.

3.1 Multi-model DBMS

Multi-model DBMSs commonly adopt a data processing layer [3], which is responsible for converting a data model to an internal representation (hidden from users). In this work, we adopt ArangoDB and OrientDB (see Section 4) as the multi-model DBMSs in the experiments.

3.1.1 OrientDB

OrientDB is an open-source multi-model database manager written in Java, in which the internal data representation is based on graphs. For better performance, database records are directly connected on a graph, and a SQL layer is also available for easier migration from the relational model.

OrientDB implements index-free adjacency, in which data manipulation is performed more efficiently due to the adopted mechanism for connecting graph nodes. The reader is referred to [22] for more details.

3.1.2 ArangoDB

ArangoDB is an open-source database management system written in C++, and the internal data representation is based on documents [23]. A database is composed of collections, which may contain several documents storing application data (using JSON format).

ArangoDB has a storage mechanism, namely, Shape, which allows the sharing of common document structures. When a document is created for the first time, its shape (structure) is saved. For new documents with the same structure, pointers are created to the shape, instead of creating a complete document. As a consequence, data storage is reduced.

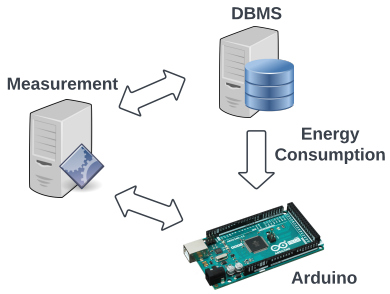


Figure 1. Measurement Framework

4. Methodology

This work adopts an evaluation methodology based on design of experiments (DoE) [24], in which a factorial design l^k with r replications is adopted. Two factors ($k = 2$) are considered: (i) *DBMS* – ArangoDB-DOC, ArangoDB-KEY, OrientDB-DOC, OrientDB-KEY, MongoDB, Redis; and (ii) *Command* (*Comm*) - insert, read, update. These DBMSs have been chosen, as they are the most popular regarding NoSQL [25]. Consequently, document (DOC) and key-value (KEY) are the common data models, as they are adopted by MongoDB and Redis, respectively. Such data models have also been

Table 1. ET-4091 and Arduino - Energy Consumption

	ET-4091 (J)	Arduino (J)
Video	1821.70	1822.85
Sysbench CPU	516.78	509.95
Hibernating	287.98	309.15
Sysbench File	398.49	402.65
Sysbench Memory	495.70	483.58
MongoDb	1125.57	1015.73
ArangoDb	574.66	520.24

considered for ArangoDB and OrientDB, as they are multi-model DBMSs and are often utilized for polyglot persistence. ArangoDB-DOC indicates the utilization of document model for ArangoDB, and ArangoDB-KEY means the utilization of key-value model. The same notation is utilized for OrientDB.

The experiments take into account 3 different workloads (1,000 operations, 5,000 operations, 10,000 operations) generated by YCSB benchmark. The metrics of interest are energy consumption (J) and execution time (s). YCSB is an open-source benchmark suite for evaluating computer applications, and this software is commonly adopted to evaluate the performance of NoSQL DBMS [26]. In addition, 100 replications ($r = 100$) are considered to obtain mean values (with approximate normal distribution) and to reduce the impact of measurement noises [24]. In this work, the results are analyzed using ANOVA [24].

The following section describes the proposed measurement framework.

4.1 Measurement Framework

We conceived a framework (Figure 1), namely, measurement server [27], which collects data related to energy consumption and execution time of a computing system. The framework is based on the Arduino platform [28], which adopts ACS217-20A [29] and ZMPT101B [30] sensors for collecting, respectively, current and voltage values. Real power [31] can then be obtained, and, using the execution time, energy consumption is estimated.

The measurement server synchronizes the start and end of a sample (i.e., replication), collects the respective execution time, and obtains the energy consumption based on the data from the Arduino platform. Our framework provides statistical techniques based on parametric methods, non-parametric techniques, and extreme value theory for estimating extreme or mean values concerning execution time and energy consumption.

The proposed framework also allows to define the sampling rate and provides a simple protocol for making easier the synchronization with a target system. Besides, Arduino was chosen due to the respective open-source platform, low cost, and several hardware as well as software components are available for assisting in the construction of digital systems. Similar equipment could be utilized to measure energy consumption. However, some platforms have constraints on the sampling rate, do not provide the documentation for the communication protocol, and may have a high cost for acquisition.

The conceived Arduino-based system was validated using MINIPA ET-4091 clamp meter [32], which is also compatible with the measurement server. The main idea is a comparison between the values obtained with the proposed platform and data collected from another device (i.e., ET-4091). Representative works have utilized a similar approach [33, 34, 35]. The focus is on energy consumption. As the framework is responsible for starting and waiting for the conclusion of a workload, the execution time is not affected by the adopted power meter.

The validation contemplates 7 experiments, in which energy consumption was measured on ET-4091 and the Arduino platform: (i) a video reproduction; (ii) the execution of 3 sysbench workloads (CPU, File, Memory); (iii) and a YCBS workload on MongoDB and ArangoDB. The adopted computer is a Core 2 Duo CPU T5450 1.66GHz, 8GB of RAM, running Ubuntu 20.04.4 LTS (Linux) with EXT4 as the file system . All operating system (OS) services have been kept to a minimum to not affect data collection.

For each measuring device and experiment, 100 samples were collected, and the respective mean value (Table 1) was estimated using bootstrap [24]. Such sample size is related to the central limit theorem, and the adopted statistical evaluation is based on the normal distribution [24]. Paired T-test [24] was carried out to assess the mean value of the differences of all matched pairs. In this test, the null hypothesis denotes the mean difference is equal to 0. A 95% confidence interval was obtained using the values in Table 1: [-19.3;64.1]. Since 0 is contained in the interval, there is no statistical evidence to indicate both platforms provide different results.

5. Experimental Results

This section presents experimental results, which utilize the design of experiments described in Section 4. As explained, YCSB benchmark is taken into account considering workloads with 1,000, 5,000 and 10,000 operations. We have kept the default configuration for each DBMS and YCSB, such that each inserted record has 1KB (default value in YCSB). The metrics of interest are execution time and energy consumption. The computer server is the same as the computer adopted in the validation.

As follows, the results are presented for each workload using ANOVA technique [24]. Next, correlation is discussed followed by general remarks.

5.1 Workload results

Table 2 and Table 3 present the results for ANOVA analysis (significance level $\alpha = 0.05$) for execution time and energy consumption. Figure 2 and Figure 3 depict the results considering the mean values and the respective 95% confidence intervals (on the top of each bar).

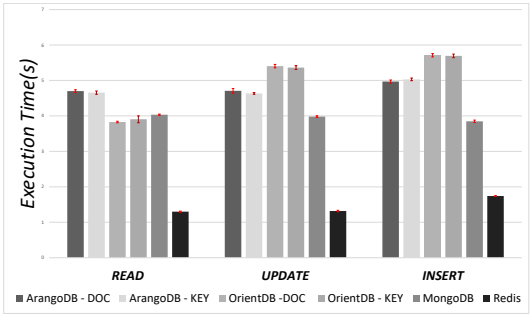
Table 2 shows all factors and their interactions (*source*) significantly impact execution time (e.g., $p - value < 0.001$). Depending on the workload (*work.*), the factor may have a

Table 2. ANOVA: Execution Time

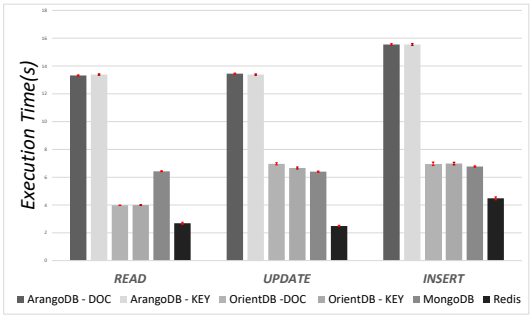
work. 1,000				work. 5,000				work. 10,000			
source	var.%	F stat.	p-value	source	var.%	F stat.	p-value	source	var.%	F stat.	p-value
DBMS	84.78	11093.85	<0.001	DBMS	93.63	57432.097	<0.001	DBMS	72.23	76308.63	<0.001
Comm	5.41	1770.82	<0.001	Comm	3.76	5770.39	<0.001	Comm	11.25	29722.74	<0.001
DBMS*Comm	7.08	463.27	<0.001	DBMS*Comm	2.03	622.25	<0.001	DBMS*Comm	16.18	8544.13	<0.001
Error	2.72			Error	0.58			Error	0.34		

Table 3. ANOVA: Energy Consumption

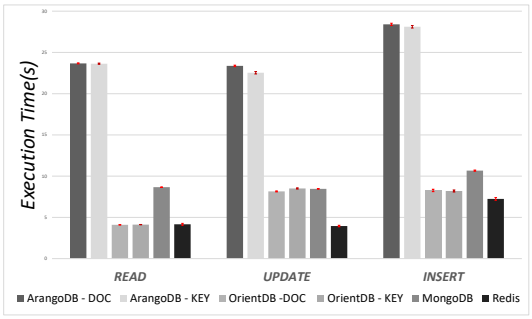
work. 1,000				work. 5,000				work. 10,000			
source	var.%	F stat.	p-value	source	var.%	F stat.	p-value	source	var.%	F stat.	p-value
DBMS	94.16	43393.38	<0.001	DBMS	78.24	28247.77	<0.001	DBMS	69.36	28863.97	<0.001
Comm	2.84	3275.64	<0.001	Comm	14.38	12980.28	<0.001	Comm	17.55	18254.09	<0.001
DBMS*Comm	2.22	511.66	<0.001	DBMS*Comm	6.39	1154.27	<0.001	DBMS*Comm	12.23	2545.24	<0.001
Error	0.77			Error	0.99			Error	0.86		



(a) 1,000 workload

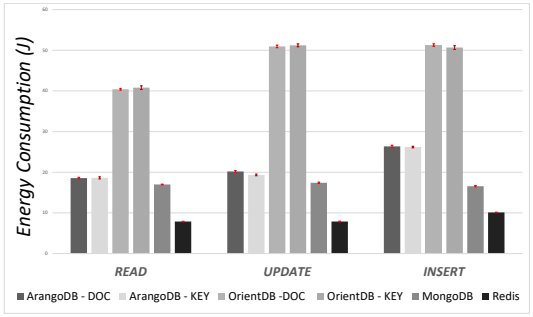


(b) 5,000 workload

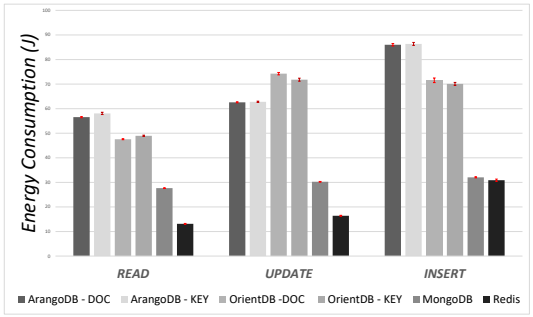


(c) 10,000 workload

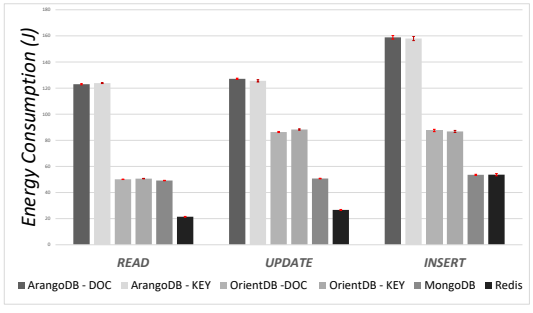
Figure 2. Mean Execution Time with 95% Confidence Interval



(a) 1,000 workload



(b) 5,000 workload



(c) 10,000 workload

Figure 3. Mean Energy Consumption with 95% Confidence Interval

distinct influence on the metric (*var.%*), as some DBMSs are influenced by the amount of operations carried out. *DBMS* has the most significant impact on the results, and the measurement noises (*Error*) are very small. As follows, the explanation is based on the assessment obtained with Tukey's procedure [24] (a post-ANOVA test). In general, the data model does not significantly influence the performance of ArangoDB and OrientDB. Only when the difference is representative, the explanation explicitly states the impact in order to avoid a cumbersome presentation.

Figure 2 (a) shows the average execution times for 1,000 operations. For all commands, the difference between all DBMSs is statistically significant. Redis is the fastest, and its execution time is almost three times less than its counterparts for reading operations. For this workload, OrientDB is better than ArangoDB and MongoDB only for reading data. Regarding update and insert, MongoDB is only behind Redis. In the context of energy consumption (Figure 3 (a)), OrientDB has the highest consumption for all commands, even for read operations. Redis provides the lowest consumption, and it can be seven times lower than OrientDB.

Considering 5,000 operations (Figure 2 (b)) ArangoDB seems to be less scalable for all commands. For instance, concerning read, ArangoDB was 500% slower than Redis, which continues to outperform all DBMSs. MongoDB is the second slowest for read commands, and OrientDB indicates a good scalability for increasing workloads. Besides, there is no statistical difference between OrientDB and MongoDB for insertion, and MongoDB is only 3.88% faster for update. Figure 3 (b) depicts the energy consumption values for such a workload. The values indicate the workload variation impacts the energy consumption differently for each DBMS. Although OrientDB has an insertion time similar to MongoDB, the energy consumption is 132.6% higher. Additionally, OrientDB is faster than ArangoDB for update, but consumes 14% more energy. Redis provides the best energy savings for all commands.

Figure 2 (c) shows execution times for the workload of 10,000 operations. The behavior resembles the 5,000 workload, in the sense that ArangoDB provides the slowest performance. For read, OrientDB is as fast as Redis, and such DBMS is only 13.17% slower than Redis for insert. MongoDB is slower for all operations in comparison to OrientDB, except for update and OrientDB-KEY. OrientDB-DOC and OrientDB-KEY have a small difference ($0.35s$), in which the former carries out the update quicker. Concerning ArangoDB-DOC, ArangoDB-KEY and update, there is a representative difference, which is about $1s$ for this workload. In relation to energy consumption, Figure 3 (c) presents the mean values. Although OrientDB has a remarkable execution time for a larger workload, the energy consumption for update and insertion is higher than MongoDB and Redis.

5.2 Correlation

For all DBMSs, there is a strong (linear) correlation (Figure 4) between execution time and energy consumption, which is indicated by the coefficient of determination (R^2) [24]. The values are 0.985, 0.986, 0.911, 0.902, 0.943 and 0.927 for ArangoDB-DOC (Figure 4 (a)), ArangoDB-KEY (Figure 4 (b)), OrientDB-DOC (Figure 4 (c)), OrientDB-KEY (Figure 4 (d)), MongoDB (Figure 4 (e)) and Redis (Figure 4 (f)), respectively.

Figure 4 also depicts the linear equation for each DBMS. The slope (i.e., the first derivative) provides interesting information about the influence of the benchmark on energy consumption, but this value should not be the single mechanism for comparison with other database systems. ArangoDB - DOC has the lowest rate J/s (5.744), but such a system takes more time to perform the workloads. Thus, the energy consumption is considerably high considering all the operations in a request.

As presented in Figure 2 and 3, Redis has a prominent performance with reduced energy consumption. Although this DBMS has a significant rate (7.292), the execution times are the lowest among all database systems. For instance, a direct comparison with only the slopes would indicate ArangoDB is more energy efficient than Redis, which contradicts the presented results. Using linear correlation, the execution times cannot be neglected.

5.3 Remarks

It is important to note that all experiments were performed adopting the default configuration for each DBMS, and configuration tuning is outside the scope of this work. However, tuning may provide different results from the values presented in this work. In addition, we only adopted a single computer to reduce potential interference from other issues associated with a cluster environment on the results. However, in a cluster environment, the results may also differ.

Regarding multi-model DBMS, in general, ArangoDB has the largest execution times for the adopted benchmark, but energy consumption is not high for small workloads. On the other hand, OrientDB has the highest energy consumption when few operations are executed, but, for a larger set of operations, energy utilization does not considerably increase.

OrientDB adopts more main memory than other DBMSs for small workloads, which may explain the highest energy consumption in this context. The initialization of internal data structures may impact the consumption for few operations, but such an approach compensates for the scalability of dealing with more requests (e.g., 10,000 workload). Concerning ArangoDB, the data processing layer may be responsible for the increasing consumption and execution times regarding the workload variation.

Redis performed remarkably well in the experiments, in the sense that such a DBMS provided the smallest execution times and energy consumption. Nevertheless, results indicate, with larger requests, OrientDB and MongoDB approximate

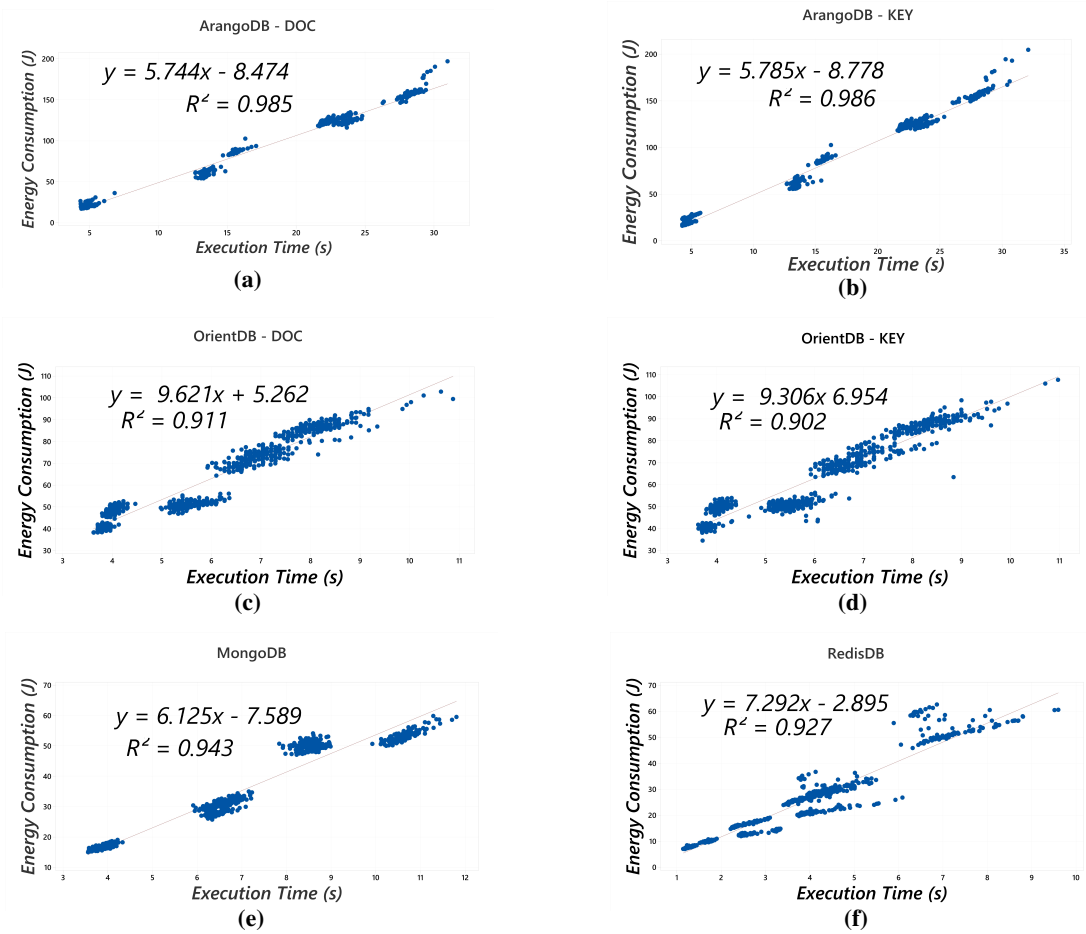


Figure 4. Correlation Energy Consumption and Execution Time: (a) ArangoDB-DOC; (b) ArangoDB-KEY; (c) OrientDB-DOC; (d) OrientDB-KEY; (e) MongoDB; and (f) Redis

Redis performance. MongoDB maintained a stable behavior with no abrupt increase in execution time or energy utilization. However, this DBMS did not provide a dominant performance compared to other database systems.

Regarding the adopted commands, read had the lowest energy consumption (as expected), for instance, due to the internal cache mechanism usually adopted by database management systems. Insert has the highest consumption, since the database needs to be modified. Particularly, ArangoDB consumes almost twice more energy than OrientDB for the 10,000 workload and insert commands.

6. Conclusion

Energy consumption is a significant concern, and storage subsystems remarkably contribute to energy utilization. Over the years, such an issue has gained considerable attention from researchers and practitioners to develop energy-efficient platforms and systems. However, few works have explored the influence of NoSQL DBMSs regarding energy consumption, and multi-model DBMSs are usually neglected.

This paper presented the assessment of performance and energy consumption of representative multi-model and single-model NoSQL DBMSs. The experiments adopted different workloads for reading, creating and updating data using YCBS benchmark. Redis provided the best results, but, considering a larger workload, OrientDB and MongoDB obtained close values regarding execution time and energy utilization.

As a future work, we are planning to evaluate multi-model DBMSs in a cluster environment and additional workloads.

Author contributions

Fúlvio Falcão, Carlos Araujo, Erica Sousa and Eduardo Tavares contributed to the research conception and data analysis. João Moura and Gabriel Silva implemented the measurement platform and carried out data gathering.

References

[1] DAVOUDIAN, A.; CHEN, L.; LIU, M. A survey on nosql stores. *ACM Comput. Surv.*, Association for

Computing Machinery, New York, NY, USA, v. 51, n. 2, apr 2018. ISSN 0360-0300. Disponível em: ⟨https://doi.org/10.1145/3158661⟩.

[2] DHUPIA, B.; RANI, M. U. Research challenges in big data solutions in different applications. In: ____. *Social Network Forensics, Cyber Security, and Machine Learning*. [S.l.]: Springer, 2019. p. 105–116. ISBN 978-981-13-1456-8.

[3] SADALAGE, P. J.; FOWLER, M. *NoSQL Distilled*. [S.l.]: Addison-Wesley Professional, 2012. v. 1. ISBN 978-0321826626.

[4] LU, J.; HOLUBOVá, I. Multi-model databases: A new journey to handle the variety of data. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 52, n. 3, jun 2019. ISSN 0360-0300. Disponível em: ⟨https://doi.org/10.1145/3323214⟩.

[5] IIMURA, N.; AL et. A proposal of storage power control method with data placement in an environment using many hdds. In: *International Conference on Ubiquitous Information Management and Communication*. [S.l.: s.n.], 2015.

[6] KARAKOYUNLU, C.; CHANDY, J. A. Exploiting user metadata for energy-aware node allocation in a cloud storage system. *Journal of Computer and System Sciences*, v. 82, n. 2, p. 282–309, 2016. ISSN 0022-0000. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0022000015001014⟩.

[7] SWAMINATHAN, S.; ELMASRI, R. Quantitative analysis of scalable nosql databases. In: *IEEE International Congress on Big Data (BigData Congress)*. [S.l.: s.n.], 2016.

[8] MACAK, M.; AL et. How well a multi-model database performs against its single-model variants: Benchmarking orientdb with neo4j and mongodb. In: *onference on Computer Science and Information Systems*. [S.l.: s.n.], 2020.

[9] GUNAWAN, R. a. a. Performance evaluation of query response time in the document stored nosql database. In: *International Symposium on Electrical and Computer Engineering*. [S.l.: s.n.], 2019.

[10] GOMES, C.; AL et. Energy consumption evaluation of nosql dbmss. In: *WPerformance*. [S.l.: s.n.], 2016.

[11] SEGHIER, B.; KAZAR, O. Performance benchmarking and comparison of nosql databases: Redis vs mongodb vs cassandra using ycsb tool. In: *International Conference on Recent Advances in Mathematics and Informatics*. [S.l.: s.n.], 2021.

[12] MARTINS, P.; ABBASI, M.; SÁ, F. A study over nosql performance. In: *WorldCIST'19 2019: New Knowledge in Information Systems and Technologies*. [S.l.: s.n.], 2019.

[13] MARTINS, P.; AL et. Nosql comparative performance study. In: *WorldCIST 2021: Trends and Applications in Information Systems and Technologies*. [S.l.: s.n.], 2021.

[14] OSE, O.; AL et. Performance benchmarking of key-value store nosql databases. *International Journal of Electrical and Computer Engineering*, v. 8, 2018.

[15] J., P. Hbase or cassandra? a comparative study of nosql database performance. *International Journal of Scientific and Research Publications (IJSRP)*, v. 10, n. 3, 2020.

[16] BARROS, J.; AL et. Integrated analysis of performance and energy consumption in distributed data storage systems. In: *Workshop em Cloud e Aplicações*. [S.l.: s.n.], 2017.

[17] MAHAJAN, D.; AL et. Improving the energy efficiency of relational and nosql databases via query optimizations. *Sustainable Computing: Informatics and Systems*, v. 22, 2019.

[18] SILVA, L.; LIMA, J. An evaluation of relational and nosql distributed databases on a low-power cluster. *The Journal of Supercomputing*, v. 79, 2023.

[19] KAUR, K.; AL et. Energy-efficient polyglot persistence database live migration. *The Journal of Supercomputing*, v. 79, 2023.

[20] DIOGO, M.; CABRAL, B.; BERNARDINO, J. Consistency models of nosql databases. *Future Internet*, v. 11, n. 2, 2019. ISSN 1999-5903. Disponível em: ⟨https://www.mdpi.com/1999-5903/11/2/43⟩.

[21] Ganesh Chandra, D. Base analysis of nosql database. *Future Generation Computer Systems*, v. 52, p. 13–21, 2015.

[22] ORIENT DB v3.2.8 Documentation. ⟨http://orientdb.com/docs/3.0.x/⟩.

[23] ARANGO DB v3.4.11 Documentation. ⟨https://www.arangodb.com/docs/3.4/index.html⟩.

[24] MONTGOMERY, D. C.; RUNGER, G. C. *Applied Statistics and Probability for Engineers*. [S.l.]: John Wiley & Sons, 2013. v. 6. ISBN 978-1118539712.

[25] DB-ENGINES Ranking. ⟨https://db-engines.com/en/rankings⟩.

[26] YAHOO! Cloud Serving Benchmark (YCSB). ⟨https://github.com/brianfrankcooper/YCSB⟩. Accessed on 01 October 2022.

[27] MEASUREMENT Server v0.01. ⟨https://www.cin.ufpe.br/~eagt⟩.

[28] ARDUINO MEGA 2560. ⟨https://docs.arduino.cc/hardware/mega-2560/⟩.

[29] ACS217-20A Datasheet. ⟨https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.pdf⟩.

[30] ZMPT101B Datasheet. ⟨https://innovatorsguru.com/wp-content/uploads/2019/02/ZMPT101B.pdf⟩.

[31] NILSSON, S. A. R. J. W. *Electric Circuits*. [S.l.]: Pearson, 2021. v. 11.

[32] MINIPA ET-4091. ⟨https://www.minipa.com.br/images/Manual/ET-4091-1102-BR.pdf⟩.

[33] PEREIRA, P.; AL et. Stochastic performance model for web server capacity planning in fog computing. *The Journal of Supercomputing*, v. 76, 2020.

[34] YUAN, P.; AL et. Performance modeling and analysis of a hyperledger-based system using gspn. *Computer Communications*, v. 153, p. 117–124, 2020.

[35] GOMES, C.; AL et. Performability evaluation of nosql-based storage systems. *Journal of Systems and Software*, v. 208, 2024.