Energy Consumption Evaluation of NoSQL DBMSs

Carlos Gomes[1], **Eduardo Tavares**[1], **Meuse Nogueira de O. Junior** [2]

[1]Centro de Informática, Universidade Federal de Pernambuco, (CIn,UFPE)

[2]Instituto Federal de Pernambuco (IFPE)

{cga,eagt}@cin.ufpe.br, meusejunior@recife.ifpe.edu.br

Abstract. *Over the years, NoSQL Database Management Systems (DBMS) have been adopted as an alternative to the constraints of relational/SQL DBMSs. In order to demonstrate their feasibility, works have evaluated NoSQL DBMSs regarding some performance metrics, but energy consumption has not been assessed. Indeed, energy consumption is an issue that should not be neglected due to the rise of energy costs and environmental sustainability. This paper presents a peformance and energy consumption evaluation of NoSQL DBMSs, more specifically, Cassandra (column), MongoDB (document-oriented), Redis (key-value). Experiments are based on YCSB benchmark, and results demonstrate energy consumption can vary significantly among the assessed DBMSs for different commands (e.g., read) and workloads.*

1. Introduction

For building and maintain a data center, high investments are involved, and energy consumption plays a remarkable role. According to reports [NRDC 2015], data centers consumed 91 billions KiloWatts in United States during 2013 at a cost of $ 9 billions USD, and the consumption is estimated to be 140 billions KiloWatts by 2020 (at a cost of $ 13.7 billions USD). Additionally, governments around the world have tried to regulate energy demand for data centers, and, thus, designs based on energy saving have been crucial for modern data centers [NRDC 2015].

Data storage is an important subsystem in data centers, and it has significantly evolved and increased the capacity due to advent of new paradigms, such as cloud computing. Consequently, Big Data concept has emerged, which requires new tools to deal with a large volume of complex data [Demchenko et al. 2014, Ji et al. 2012].

No relational database management systems (DBMS), termed as NoSQL, have been proposed as a solution to deal with the constraints (e.g., performance) of conventional data management tools, such as relational/SQL DBMSs. Nowadays, there are over 150 NoSQL DBMSs [Kuznetsov and Poskonin 2014], which adopt distinct data models, such as key-value. The abundance and diversity of NoSQL DBMSs have motivated researches to evaluate and compare these DBMSs regarding latency and throughput. However, to the best of our current knowledge, energy consumption is usually neglected, despite its huge importance for modern data centers.

As previously stated, energy consumption is an important non-functional requirement, and the selection of an appropriate NoSQL DBMS may allow cost reduction and less environmental impact. Besides, system reliability is impacted by energy consumption due to the influence on operational temperature [Zhang et al. 2014]. Reducing

consumption in data center provides benefits in terms of heat dissipation and, thus, system reliability.

This paper presents an energy consumption evaluation of representative NoSQL DBMSs with distinct data models, more specifically, Redis (key-value), MongoDB (document-oriented) and Cassandra (column). Experiments are based on Yahoo! Cloud Serving Benchmark (YCSB), in which insert, read, and delete commands are evaluated for each adopted DBMS. Besides, we also provide performance metric for these operations and their correlation to energy consumption. A measurement framework, namely, EMeter, is also presented, which contemplates hardware components and a software tool to estimate energy consumption and execution time.

The remainder of this paper is organized as follows. Section 2 summarizes related works and Section 3 presents an overview of NoSQL DBMS. Section 4 describes the methodology and the measuring framework, and Section 5 presents the results. Finally, Section 6 presents closing remarks and future works.

2. Related Works

According to Intel Labs report [Minas and Ellison 2009], in a server, CPU is the main energy consumer (followed by RAM memory), which indicates processes that require intense CPU usage and memory access, like those utilized by DBMSs, lead to high energy consumption. However, few works have evaluated the performance and energy consumption of DBMSs. Commonly, researchers assess throughput and latency assuming distributed applications and high-capacity hardware - memory and processing power - for evaluating performance under high workloads [Seriatos et al. 2016, Floratou et al. 2012, Neves and Bernardino 2015]. It is difficult to find works related to energy consumption for DBMSs, besides, researches assessing energy consumption in NoSQL DBMS are scarce.

In [Abramova and Bernardino 2013], the authors assessed the execution time of two NoSQL DBMSs, Cassandra and MongoDB, taking into account distinct workloads generated by YCSB benchmark. Results demonstrate Cassandra provided an improved scalability in comparison to MongoDB. [Abubakar et al. 2014] evaluated MongoDB, ElastickSearch, OrientDB and Redis DBMSs considering distinct workloads for read, insert and update, but energy consumption is not taken into account. [Seriatos et al. 2016] carried out a performance evaluation comprising HBase, MongoDB and Cassandra in a cloud adopting YCSB benchmark, considering different scenarios in which different parameters of each DBMS was assessed.

[Neves and Bernardino 2015] evaluated the performance of Voldemort, a NoSQL DBMS, using YCSB benchmark considering an enviromnent with 1, 3 and 6 nodes. The work concluded Voldemort DBMS does not scale considerably, and additional nodes do not have a significant impact on performance. [Cai et al. 2013] evaluated Hbase DBMS using also YCSB benchmark, and the work shows issues related to scalability and concurrent client requests. [Li et al. 2014] assessed energy consumption in NoSQL DBMSs regarding the idle state on cluster configurations. The analysis was performed collecting the idle time of each node, which may not reflect the dynamics and behavioral variations of accessing a DBMS.

Different from previous works, this papers provides an energy consumption eva-

luation and comparison of representative NoSQL DBMSs, also providing a correlation between performance and energy consumption.

3. NoSQL DBMS

Databases comprise data repositories and a common model for manipulating data. To access these repositories, DBMSs are adopted, which are set of mechanisms for providing data storage and manipulation [Abramova and Bernardino 2013]. Over the decades, several DBMSs have been proposed.

The term NoSQL was adopted for the first time in 1998 [Kuznetsov and Poskonin 2014] for a small DBMS that did not adopt the relational model. Nowadays, the term also encompasses DBMSs, which do not fully comply to ACID (Atomicity, Consistency, Isolation, Durability) properties and/or the relational model [Kuznetsov and Poskonin 2014]. Besides, the popularity of NoSQL DBMSs is related to the advent of Big Data, as huge data volume needs to be quickly manipulated, and traditional data models or ACID transactions may considerably affect system performance.

In this work, we evaluate 3 popular NoSQL DBMSs (Redis, MongoDB, Cassandra) with representative data models (key-value, document-oriented, column) (Figure 1), and they are described as follows.

3.0.1. Column

Column model (Figure 1 (a)) is the closest to the relational model, in which data are structured using the following column types [Abramova and Bernardino 2013]:

- **column** represents a set of data structured as key-value;
- **super column** provides a group of columns;
- **column family** represents a data set composed of super columns, which resemble the relational model.

Cassandra [Planet 2015] is a popular column DBMS, which was initially developed by Facebook using Dynamo DBMS (Amazon) and BigTable (Google) as a basis. Additionally, Cassandra provides a set of functionalities to facilitate a deployment onto distributed architectures, and it adopts a query language with a syntax similar to SQL.

3.0.2. Document-oriented

Document-oriented model (Figure 1 (b)) represents data using a document format, usually, based on well-known standards, such as XML and JSON. Each document may have several fields, which may include arrays and even other documents. A document has a unique key, which is the usual mechanism for accessing documents. Besides, documents can be grouped [Abramova and Bernardino 2013].

MongoDB[Mongo 2015] is a representative document-oriented DBMS, which adopts JSON for data storage . To avoid performance issues, MongoDB extensively adopts main memory. Documents are manipulated using conventional operations, such

as read, insertion, exclusion, update. To search for documents, some mechanisms are available based on projection (adopted by YCSB benchmark) and iteration. Besides, during an update, a document can be totally or partially modified.

3.0.3. Key-Value

This data model stores all data using a structure based on key-value (Figure 1 (c)). Each key is unique and data access is performed relating keys to hash values in order to access the desired value. The value can be a primitive type or complex data structure, such as a tree.

In this work, we adopt Redis [Kuznetsov and Poskonin 2014], which is a open-source DBMS that supports key-value model. Besides, Redis supports atomicity and isolation, and data is kept on main memory. To allow persistent storage, Redis provides different mechanisms, in which the default mechanism periodically stores snapshots obtained from the main memory.

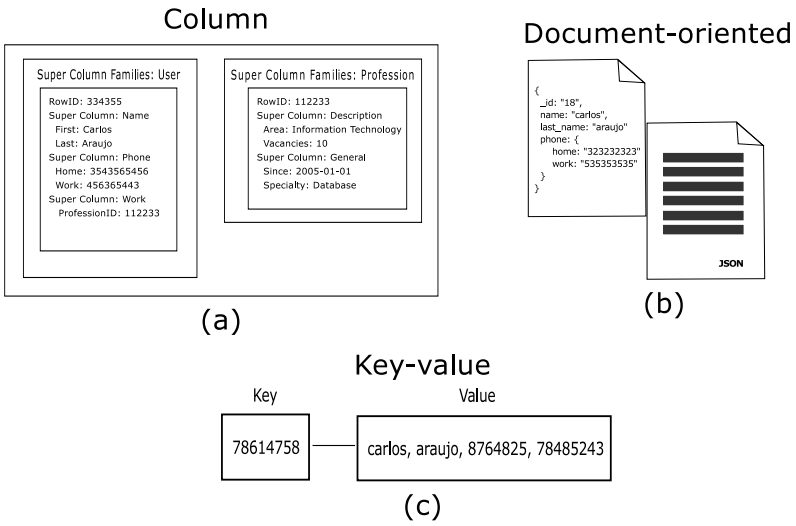


Figura 1. NoSQL Data Models

4. Methodology

The methodology is based on design of experiments [Douglas C. Montgomery 2013], in which a l^k factorial design with r replications is adopted. We have considered 2 factors ($k = 2$) with 3 levels ($l = 3$): (i) DBMS - Cassandra, MongoDB, Redis; and (ii) command - insert, read, update. Besides, 3 different workloads are considered (1,000 operations, 10,000 operations, 100,000 operations) generated by YCSB benchmark [Cooper et al. 2010], and the metrics of interest are energy consumption and execution time for each workload (e.g., 1,000 operations). The workload could be an explict factor for the proposed evaluation, but it would be the major source of variation in result analysis, and the overall measurement noise (i.e., random errors) would hinder a finer comparison.

YCSB is an open-source benchmark suite for evaluating computer applications, being often adopted to compare the performance of NoSQL DBMSs. Indeed, YCSB benchmark has been the standard for evaluating the performance of general DBMS. Additionally, 75 replications ($r = 75$) are taken into account to obtain mean values (with an approximate normal distribution) and to reduce the impact of measurement noises. A replication is a workload execution (e.g., a single execution of 1,000 operations). In this work, results are analyzed using ANOVA [Douglas C. Montgomery 2013].

The adopted DBMS server is a Core 2 Duo CPU T5450 1.66GHz with of 2GB RAM memory, running Debian 7.8 (Linux) with EXT4 as system file. All operating system (O.S.) services were kept to a minimum to not impact data collection, and no application process (except DBMS software) is executing. Regarding DBMS softwares, Cassandra 2.0.15, MongoDB 2.0.6 and Redis 2.4.14 were adopted with the default configuration.

As follows, the conceived measurement framework is described.

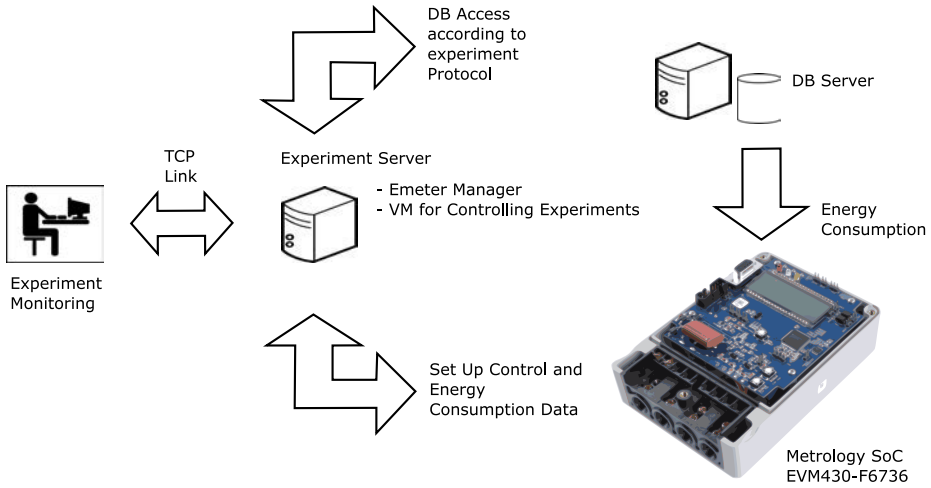


Figura 2. Measurement Framework

4.1. Measurement framework

We conceived a measurement framework (Figure 2), namely, Emeter, to allow the collection of data related to energy consumption and execution time in the DBMS server. The framework contemplates hardware components and a software tool to store and visualize the collected metrics.

The framework adopts a experiment server, in which Emeter software executes, and it communicates with a EVM430-F6736 hardware [Texas-Instruments 2015]. The latter is a specialized device that collects instantenous current and voltage for estimating instantaneous eletric power. Energy consumption is calculated by using a numerical integration using the instantenous powers and the time interval for executing a workload. EVM430-F6736's firmware was modified, such that Emeter's software can communicate and collect the required data via a serial interface. For managing and controlling, Emeter has a GUI (Graphical Unit Interface), in which real-time consumption and execution

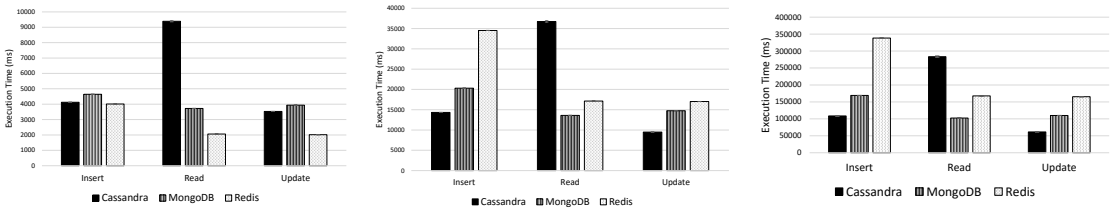
information can be visualized. The collected data is stored on a database or a csv file. Additionally, EMeter supports remote control by TCP/IP communication, in the sense that all experiments can be externally controlled.

The experiment server also executes YCSB benchmark, which triggers the workload on the DBMS server. The time interval for a workload is adopted for estimating execution time and energy consumption, and each execution is kept independent in order to avoid interfence between replications (e.g., cache).

5. Experimental results

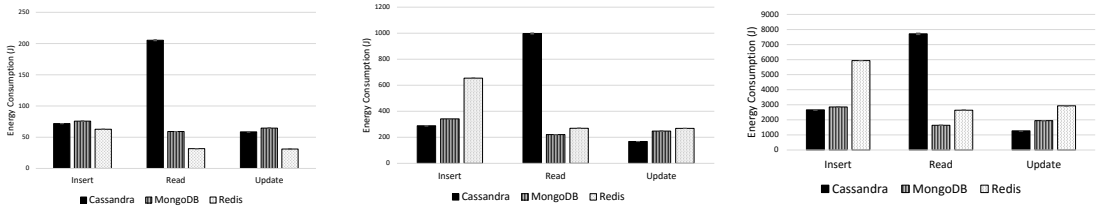
We have adopted a design of experiment, which is explained in Section 4. YCSB benchmark is taken into account considering workloads with 1,000, 10,000 and 100,000 operations. For each DBMS, the default configuration is adopted, and the metrics of interest are execution time and energy consumption. Besides, each inserted record in the database has 1KB (standard value defined by YCSB), and 10 threads is adopted for mimicking a real environment with concurrent accesses. As follows, the results are presented for each workload using ANOVA analysis [Douglas C. Montgomery 2013]. Next, correlation is discussed for both metrics followed by general remarks.

5.1. Workload results



(a) (b) (c)

Figura 3. Mean Execution Time Values

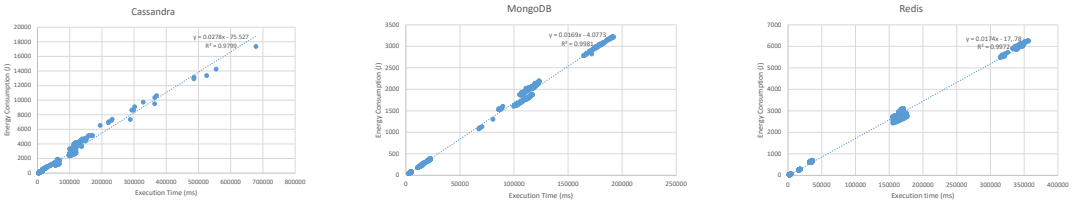


(a) (b) (c)

Figura 4. Mean Energy Consumption Values

Table 1 and Table 2 provides the results for ANOVA analysis (significance level $\alpha = 0.05$) for execution time and energy consumption. Figure 3 and Figure 4 depict the results considering the mean values.

Table 1 depicts all factors and their interactions ($source$) significantly impact the execution time (F statistic -$Fstat.$- and p-$value$). Depending on the workload ($work.$),



(a) (b) (c)

Figura 5. Correlation Energy Consumption X Execution Time

the factor may have a different impact on the metric ($var.\%$), as some DBMSs vary their behaviour for the amount of operations executed. In general, the interaction between DBMS and command ($DBMS*Comm$) is the major source of variation, but, in 100,000 workload, the interaction influence changes due to the behaviour of Redis and Cassandra. Similar results are obtained for energy consumption. However, for 100,000 workload, a minor difference occurs due to a sudden increase of energy consumption by Cassandra (explained in the following paragraphs). Other sources of variation are due to experimental errors (i.e., noise in the measurements), and they are reduced with the increase of workload. Such a situation is related to the time interval required for a workload execution, which decreases, for instance, the influence of other O.S. services. The following explanations are based on the results obtained with Tukey's procedure [Douglas C. Montgomery 2013] (a post-ANOVA test).

Tabela 1. ANOVA: Execution Time

work. 1,000				work. 10,000				work. 100,000			
source	var.%	F stat.	p-value	source	var.%	F stat.	p-value	source	var.%	F stat.	p-value
DBMS	23.15	230.41	<0.001	DBMS	7.97	84.31	<0.001	DBMS	30.80	453.88	<0.001
Comm	9	89.55	<0.001	Comm	16.95	179.28	<0.001	Comm	20.32	299.42	<0.001
DBMS*Comm	30.78	153.16	<0.001	DBMS*Comm	42.74	226.04	<0.001	DBMS*Comm	26.28	193.62	<0.001
Error	37.07			Error	32.33			Error	22.60		

Tabela 2. ANOVA: Energy Consumption

work. 1,000				work. 10,000				work. 100,000			
source	var.%	F stat.	p-value	source	var.%	F stat.	p-value	source	var.%	F stat.	p-value
DBMS	23.24	243.66	<0.001	DBMS	7.54	99.73	<0.001	DBMS	13.28	147.94	<0.001
Comm	9.78	102.55	<0.001	Comm	15.71	207.92	<0.001	Comm	16.62	185.14	<0.001
DBMS*Comm	31.36	164.45	<0.001	DBMS*Comm	51.59	341.34	<0.001	DBMS*Comm	40.19	223.83	<0.001
Error	35.62			Error	25.16			Error	29.90		

Figure 3 (a) depict the mean execution times for running 1,000 operations. Concerning insert command, the difference between values are not statistically signifcant, and, thus, we do not have enough statistical evidence to reject the equality between the execution times. For read commands, the difference is significant, and Cassandra considerably takes more time than other DBMSs. For instance, it is, respectively, 60.33% and 78.04% slower than MongoDB and Redis (which provided the lowest execution time).

Regarding update, Redis also provides the best value for this workload followed by Cassandra. In the context of energy consumption (Figure 4 (a)), the values indicate a correlation with execution time, since similar arguments can be provided. The values for insert command are also close, and Tukey's test indicates the differences between DBMSs are not expressive. Concerning read command, Cassandra provides the highest energy consumption, which is 71.17% larger than MongoDB and 84.57% than Redis for this command. Redis has the least energy consumption for updates, but MongoDB and Cassandra do not have mean values that significantly differ.

Considering 10,000 operations (Figure 3(b)), Redis appears to be less scalable than other DBMSs for insert command. Redis was 58.51% slower than MongoDB and 41.18% than Cassandra. this situation explains the increase of command ($comm.$) as a source of variation in Table 1. Comparing to 1,000 workload, Redis' execution time increased 88.38%, whereas for Cassandra and MongoDB the executime increased 71.22% and 77.13% , respectively. Cassandra provides the best execution time for insert command, but its behaviour repeats for reading requests, which execution time is much larger than Redis and MongoDB. In this context, MongoDB provides the best performance. For update, Cassandra provides good execution time, in which it is almost 2 times faster than MongoDB and Redis. Figure 4 (b) depicts energy consumption values for this workload, which also points to the correlation between execution time and energy consumption. The values indicate workload variation impacts execution time and energy consumption in different proportions. As an example, the execution time of Redis DBMS increased 88.38% for insert command, but energy consumption increased only 90.41%. For Cassandra and MongoDB, the increment were 75.10% and 77.82%, respectively. In relation to read operation, Cassandra, Redis, MongoDB consumed more 79.42%, 88.25% and 73.19%, respectively. Energy consumption for update correspondingly increased 65.04%, 73.91%, 88.41% for Cassandra, MongoDB and Redis.

Figure 3 (c) shows the mean values for execution times concerning 100,000 operations workload. The behaviour resembles 10,000 workload, in which Redis provides the slowest performance, except for update command. The mean time for executing 100,000 insert commands is the poorest performance in all experiments (surpassing the execution time of read commands for Cassandra). The values is 67.96% higher than Cassandra, and 50.06% larger than MongoDB. Also, Cassandras did not scale well for read commands. The values is 63.87% higher than MongoDB and 40.87% than Redis. Concerning update, Cassandra provides the best performance, followed by MongoDB and Redis. Figure 4 (c) provides an interesting behaviour. Although it also provides evidence for correlation between execution time and energy consumption, energy consumption in Cassandra seems higher than other DBMSs. Although Redis' execution time for insert commands are greater than Cassandra's update commands, Cassandra consumed more energy consumption. Such a behaviour indicates the change of variation ($var.\%$) of each factor for both metrics . Particularly, the contribution of command factor raises as well as the noise in the measurement for energy consumption.

5.2. Correlation

Figure 5 depicts the correlation between energy consumption and execution time for each DBMS contemplating all workloads and commands (insert, read and update).

For all DBMS, a strong (linear) correlation is obtained, which is corroborated by the square of correlation coefficient (R^2). The values are 0.9799, 0.9981, and 0.9972 for Cassandra, MongoDB and Redis respectively. Figure 5 (a) depicts the values for Cassandra, in which equation $y = 0.0278x$ - 75.527 represents the correlation. Equation $y = 0.0169x$ - 4.0773 represents the correlation for MongoDB (Figure 5 (b)), and $y = 0.0174x$ - 17.378 for Redis (Figure 5 (c)).

The slope (i.e., the first derivative) provides an interesting information regarding the impact of executing commands on mean energy consumption. For the adopted DBMS, MongoDB is more energy efficient (0.0169 energy consumption/execution time), and Redis is very close (0.0174), despite the issue on 100,000 workload. Cassandra consumes more energy in average, which is represented by the slope 0.0278, which is strongly affected by read commands.

5.3. Remarks

It is important to emphasize that all experiments have been carried adopting the default configuration for each DBMS, and configuration tuning is out of scope for this work. Nevertheless, tuning may provide different results from the values presented in this work. Besides, we have adopted only a single computer to reduce possible interference of other issues associated with a cluster environment on the results. Nevertheless, in a cluster environment, the results may also differ.

Redis had a remarkable performance for 1,000 workload, in the sense that it provided the best performance for this workload concerning read and update commands. Execution time and energy consumption is equivalent to other DBMSs concerning insert command. We believe this workload was not sufficient to activate Redis' snapshot task for persistently saving the new records during a workload execution. However, the snapshot was activated for 10,000 and 100,000 workloads [Abubakar et al. 2014], which affected the performance of insert command and generated the worst execution time in 100,000 workload.

Cassandra performed very well for insert and update commands, and those commands were improved with the increasing of workload. Indeed, for 10,000 and 100,000 workloads, Cassandra provided best results. However, such a DBMS provided worst results for read command in all workloads, and this is the command that consumed more energy. Cassandra adopts several tasks to access persistent data [Cassandra 2015], which include decompress data from a storage device (e.g., hard disk). All samples are independent, and, thus, no data are present in Cassandra's cache during the workload for read command. In [Abramova and Bernardino 2013], the authors observed a similar behaviour for Cassandra.

For all workloads, MongoDB kept a stable performance regarding no abrupt increasing in execution time or energy consumption for the adopted commands. However, it did not provide a dominant performance comparing to other DBMSs. A minor increase in the time for executing insert commands may be related to the journaling mechanism adopted for providing fault-tolerance.

It is important to emphasize that no DBMS provided a dominant behaviour (execution time and energy consumption) in all workloads.

6. Conclusion

Energy consumption is a very important concern, which has gained considerable attention by researchers and practitioners to develop energy efficient platforms and systems. Due to advent of big data, storage systems have also their considerable contribution on energy usage, but such a concern has not been explored on NoSQL DBMSs (although works have evaluated NoSQL DBMSs in the context of throughput and latency).

This paper presented performance and energy consumption evaluation of representative NoSQL DBMSs. Experiments adopted distinct workloads for read, create and update commands using Yahoo! Cloud Serving Benchmark. Although no single DBMS dominates all workloads, MongoDB provided a stable behaviour for the adopted system and default configuration. Cassandra and Redis provided prominent execution time and energy consumption for some scenarios, but they did not performe well for read and insert commands, respectively. As future works, we are planning to evaluate NoSQL DBMSs on a cluster environment.

Referências

Abramova, V. and Bernardino, J. (2013). Nosql databases: Mongodb vs cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pages 14–22.

Abubakar, Y., Adeyi, T. S., and Auta, I. G. (2014). Article: Performance evaluation of nosql systems using ycsb in a resource austere environment. *International Journal of Applied Information Systems*, 7(8):23–27.

Cai, L., Huang, S., Chen, L., and Zheng, Y. (2013). Performance testing of hbase based on the potential cycle. In *Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on*, pages 359–363.

Cassandra, W. (2015). Architecture internals. https://wiki.apache.org/cassandra/ArchitectureInternals. Accessed: 2016-03-24.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154.

Demchenko, Y., De Laat, C., and Membrey, P. (2014). Defining architecture components of the big data ecosystem. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 104–112.

Douglas C. Montgomery, G. C. R. (2013). *Applied Statistics and Probability for Engineers*. Wiley, 6th edition.

Floratou, A., Teletia, N., DeWitt, D. J., Patel, J. M., and Zhang, D. (2012). Can the elephants handle the nosql onslaught? *Proceedings of the VLDB Endowment*, 5(12):1712–1723.

Ji, C., Li, Y., Qiu, W., Jin, Y., Xu, Y., Awada, U., Li, K., and Qu, W. (2012). Big data processing: Big challenges and opportunities. *Journal of Interconnection Networks*, 13.

Kuznetsov, S. D. and Poskonin, A. V. (2014). Nosql data management systems. *Programming and Computer Software*, pages 323–332.

Li, T., Yu, G., Liu, X., and Song, J. (2014). Analyzing the waiting energy consumption of nosql databases. In *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*, pages 277–282. IEEE.

Minas, L. and Ellison, B. (2009). *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*. Intel Press.

Mongo, M. (2015). Bring your giant ideas to life with mongodb. https://www.mongodb.com/what-is-mongodb. Accessed: 2016-03-22.

Neves, R. and Bernardino, J. (2015). Performance and scalability of voldemort nosql. In *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on*, pages 1–6.

NRDC (2015). America's data centers consuming and wasting growing amounts of energy. http://www.nrdc.org/energy/data-center-efficiency-assessment.asp. Accessed: 2016-03-01.

Planet, C. (2015). What is apache cassandra? http://www.planetcassandra.org/what-is-apache-cassandra/. Accessed: 2015-12-10.

Seriatos, G., Kousiouris, G., Menychtas, A., Kyriazis, D., and Varvarigou, T. (2016). *Comparison of database and workload types performance in Cloud environments*, pages 138–150.

Texas-Instruments (2015). Evm430-f6736 - msp430f6736 evm for metering. http://www.ti.com/tool/EVM430-F6736. Acessed: 2016-03-22.

Zhang, H., Shao, S., Xu, H., Zou, H., and Tian, C. (2014). Free cooling of data centers: A review. *Renewable and Sustainable Energy Reviews*, pages 171–182.