IoT Data Storage: Relational & Non-Relational Database Management Systems Performance Comparison

Gizem Kiraz
Computer Engineering
Uludag University
Gorukle, Bursa
501631002@ogr.uludag.edu.tr

Cengiz Toğay
Computer Engineering
Uludag University
Gorukle, Bursa
ctogay@ uludag.edu.tr

ABSTRACT

Internet of Things (IoT) becomes recently a popular research topic and market reality. According to several research companies, in 2025, up to 75 billion devices are estimated to connect internet and generate an enormous number of data. This increases in data cause several difficulties such as the storage cost and processing of such large data. In this paper, we have been studied on performance comparison of relational (MySQL) and non-relational (MongoDB) database management systems for storing and processing of this large IoT data. Both types of database management systems have been tested. According to comparison of experimental results, the non-relational database management systems, which we studied and searched, have provided better performance for storing and processing of large data.

Keywords

Internet of Things; MySQL; MongoDB; RDBMS; NRDBMS.

INTRODUCTION

The Internet of Things (IoT) is a self-configuring and adaptive system that consist of sensor networks and smart objects whose aim is to interconnect all devices/sensors in daily life [1]. According to the projections of many organizations and companies, up to 75 billion devices will interconnect using the internet and several challenges and issues that need to be addressed will raise. Therefore, IoT becomes one of the most popular research topic recent years. Moreover, IoT is closely related to big data and cloud technology. Big data is produced by the different types of the applications such as industrial processes, medical devices, embedded control systems, gateways, and GPS sensors etc. This means that an amount of data worldwide increases day by day. In 2016, more than 5.5 million connected devices are inserted every day, and it is expected that number of devices more than 20.8 billion worldwide by 2020 [2]. Sensors are also produced data and they are important for big data growth. The sensor data is the most popular data type between IoT applications.

Big data is a term that describes the large volume of data –both structured and unstructured. The DBMSs basically can be separated into relational and non-relational DBMS. The relational DBMS stores the data rows and columns in tables with a high data consistency. The most commonly used open source relational DBMS is MySQL. Non-relational databases (NOSQL) have arisen as an alternative to relational databases. The aim of the NOSQL is often not to give guaranty the Atomicity, Consistency, Isolation, and Durability (ACID). The NOSQL does not depend on constant table definitions and rigid schemas. Columns or records can be added to the collection at any time without exclusive process. Therefore; the number of records in the columns does not have to equal with each other. Data sets in IoT environment can change after setup of the system, so this environment requires a flexible data storage system. There are four different storage formats in NOSQL namely key-value, columns, document-based and graff-based.

It has been investigated a document-based storage format in NOSQL. In such a system, a record is called document and these documents are usually stored in JSON format [3]. There are various implementation of the NOSQL DBMS such as MongoDB [4] [5], CouchDB [6], HBase [7], Cassandra [8], Amazon SimpleDB [9], and Redis [10]. Since MongoDB is open source and commonly used, it has been chosen MongoDB in this study. There are no database schemas or tables in MongoDB. MongoDB uses “collection” instead of a table, and “document” instead of rows to store data. Furthermore, MongoDB uses two different operations instead of the join operation. These are nesting documents inside each other and to store a reference to the other document rather than nesting entire document.

There are many studies about comparing the performance of databases [11] [12] [13] [14]. These studies vary depending on the data size, the variety of data, the differences in databases used, implementation languages, and subjects of the projects. In the study [15], MongoDB, MySQL, CouchDB, and Redis are compared. It is declared that MongoDB is performing better among the comparative database management systems in terms of the "bulk insert" writing performance. However, MySQL and MongoDB have similar performance results for reading operations. Performance parameters between these DBMSs can be negligible (typically less than 1 second) [15]. However, our test results show that MongoDB has better performance than MySQL in terms of reading and writing as represented in “Results of Experiments” section. MongoDB is utilized for to store GPS sensor data and to communicate with the analysis tools such as Apache Mahout [11]. ACID operations on MongoDB and MySQL DBMSs are also applied to compare them [12] [13]. According to the results, the use of the MongoDB has been encouraged for large data applications, especially for applications of big data [12]. In [14], MongoDB, Raven, CouchDB, Cassandra, HyperTable, CouchBase, and SQL DBMSs are compared in terms of the read, write, delete, and instantiate operations. According to results of this study[14], all keys are needed to fetch, MongoDB has better performance than the others.

The aim of the experiments is a comparison of the relational and non-relational DBMSs for utilization an IoT platform. The system includes IoT devices which publish a tremendous number of sensor data where servers store and process them. Performance of reading and writing tests has been done in both MYSQL and MongoDB in this study. The test results which are calculated after the application runs at least three times are compared to find out where we can store data of IoT considering lowest cost in terms of throughput.

IoT platform is defined which collects, and processes sensor data as seen in Figure 1. The sensor data is produced by devices/clients and collected by *Data Storage Server (DSS)* in server side. The data is stored in a DBMS through insert and update operations. *Data Query Server (DQS)* in the platform provides an interface for processing and reporting by Client Application. The client application sends the data request to the DQS and they are formed as a query for DBMS. The result of the query is delivered to the client application. In our case, enormous data should be stored in the DBMS. Therefore, writing operations are more important than the reading operations. The platform has active-active architecture. Therefore, more than one DSSs can handle the data

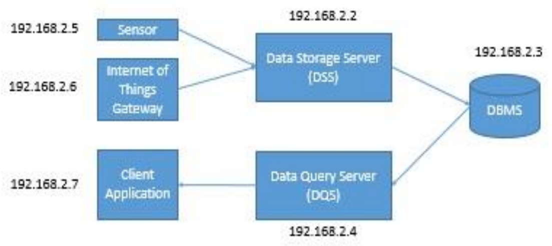

Figure 1.The IoT Platform Architectural Structure

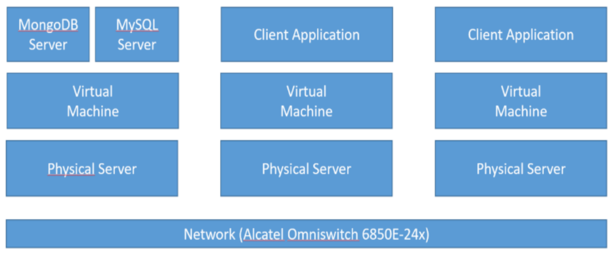

Figure 2.Test Environment

writing transactions. In our platform, our target throughput is forty messages/milliseconds in average for writing. Our target throughput and test results compared with “Results of Experiments” section.

The rest of the article is organized as follows. “Test Environment & Methodology” section consists of the information about the environment of the experiments and methodology. “Results of Experiments” section presents the results and graphics from the experiments. “Conclusions” section summarizes and concludes the experiments and gives a recommendation for future of this study.

TEST ENVIRONMENT & METHODOLOGY

In this paper, research is conducted on the relational and non-relational databases. Server and clients of chosen DBMSs to be used in our experiments are set up separately in virtual machines with Ubuntu 16.04 Server version. Virtual machines are hosted on a physical machine (i7 6700HQ, 16 GB DDR3 RAM, and SSD disc). The virtual machines (4 CPU cores, 4 GB RAM) are executed on the physical servers as depicted in Figure 2. A dedicated network is a setup among the servers. Therefore, it is guaranteed that another network traffic is not disrupted the tests. Both DBMSs are installed on the computer with SSD for the fastest possible read and write speed and they are executed separately during the test scenarios. Multithreaded Java applications for reading and writing operations on DBMSs are implemented.

Experiments’ constraints are the number of machines, number of threads, number of messages, and the size of the string. These constraints are applied for both DBMSs. The application is executed on a virtual machine; therefore, the applications are limited in terms of the CPU core and memory. In this study, since writing operations are more important than reading operations, our tests concentrate on the writing operations. In our tests, data examples are selected as similar to real-time sensor applications.

In this study, two columns are defined including variable-length string type, an integer type. A primary key column is automatically defined in MySQL, but it needs to be defined in MongoDB. The execution time of the tests is calculated in milliseconds. The number of messages is measured dividing the total number of messages into the experiment’s execution time.

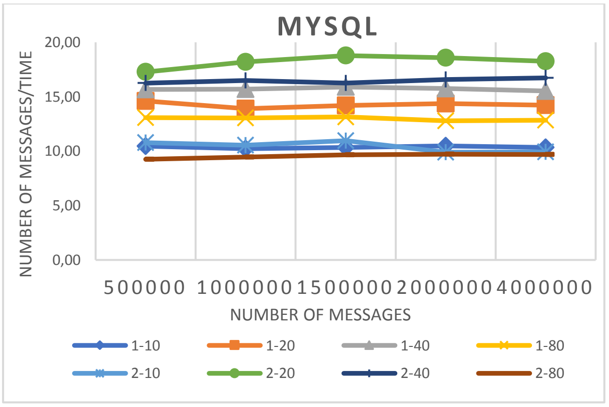

Figure 3.Result Graph of Write Operations on MySQL

RESULTS OF EXPERIMENTS

Insert tests are executed with a multithreaded Java application. The application sends insert SQL request which contains a string (100 characters) and an integer value to DBMSs. The application is executed in computers based on the parameters (number of computer and threads) as depicted in Figure 3 and Figure 4. Best throughput is 18.21 messages/millisecond in average for MySQL such that is succeeded with two computers each has twenty threads as presented in Figure 3. As it can be seen that forty threads for one and two computers utilization have close results. Best throughput will go up when the number of threads increases. However, when the number of threads reaches eighty, throughput value begins to decrease. Therefore, it has been decided that MySQL can manage forty threads for best results.

Similarly, best throughput is 70.95 messages/millisecond in average for MongoDB such that is succeeded with two computers each has forty threads as presented in Figure 4. To eliminate the effects of the thread switching, the third computer is also used for MongoDB. Throughput for utilization of the one, two and three computers is 61.22, 70.95, and 61.65 messages/millisecond, respectively. As it can be seen that two computers' utilization has the best performance. Another result can be obtained from the Figure. 3 and Figure. 4 is a correlation between a number of threads and computers. Best results are obtained for MySQL; forty threads for one computer and twenty threads for two computers and also for MongoDB; eighty threads for one computer, forty threads for two computers, and twenty threads for three computers. It can be seen that MongoDB supports more than sixty threads for best throughput.

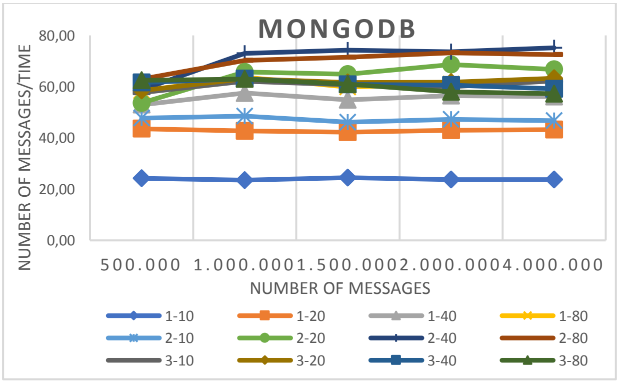

Figure 4.Result Graph of Write Operations on MongoDB

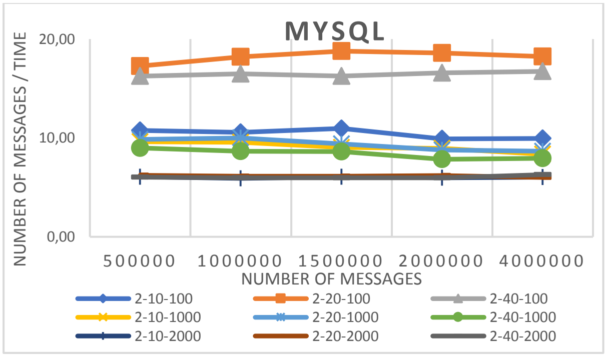

Figure 5.Results of String Length Experiments on MySQL

It is also tested the data with different variable string length as depicted in Figure 5 and Figure 6. Since best results are obtained from the two computers, only two computers cases are tested in these tests. As it can be seen that 18.21, 9.33, and 6.13 messages/millisecond in average are obtained for MySQL with 100, 1000, and 2000 string length respectively as depicted in Figure 5. Similarly, previous results, twenty threads utilization for MySQL has best throughput results. For MongoDB, forty threads utilization has best throughput results; 70.95, 49.04, and 39.85 messages/millisecond in average are obtained with 100, 1000, and string (2000 characters) respectively as depicted in Figure 6. MongoDB DBMS has about four times better results than MYSQL. As it can be seen that length of the message is one of the most important parameters. Such as when the length is doubled, throughput is decreased about twenty percent.

Select tests are also executed with the same multithreaded Java application. The application retrieves data from two DBMSs. The application is executed on a computer and two computers with a different number of the threads as depicted in Figure 7 and Figure 8. MySQL results are not

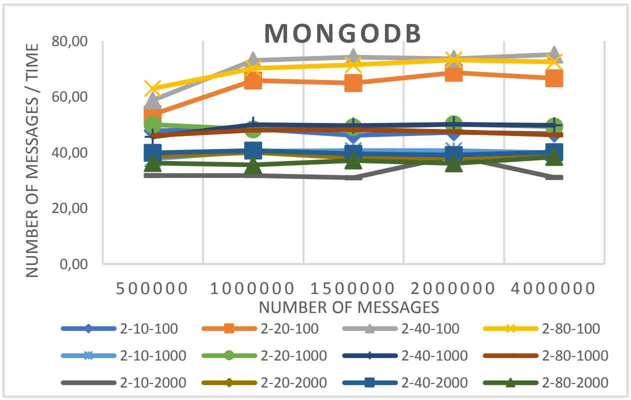

Figure 6.Results of String Length Experiments on MongoDB

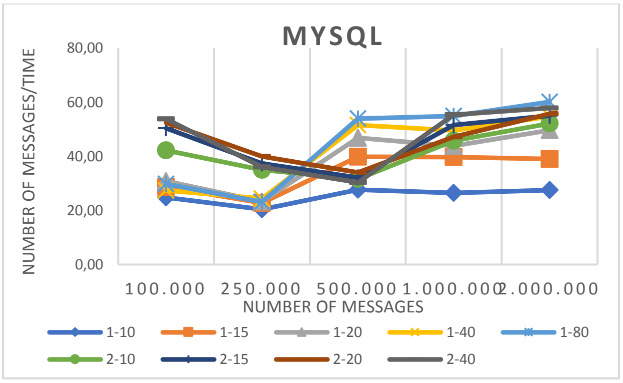

Figure 7.Result Graph of Read Operations on MySQL

stable as MongoDB. Best throughput is 60.09 messages/millisecond and 44.34 in average for MySQL with one computer has eighty threads. In MongoDB, the best result is 68.68 messages/millisecond and 58.61 messages/millisecond in average is succeeded with three computers each has eighty threads as depicted in Figure 8. Furthermore, it has been concluded that MongoDB can manage eighty threads for reading operations.

CONCLUSIONS

Our IoT platform requires that forty messages/milliseconds in average should be written to the chosen DBMS. Otherwise, the number of messages waiting in the queue for writing will increase and it can cause memory problems. In the IoT platform, active-active architecture is applied. Therefore, more than one computer can write to DBMS at the same time. Test results show that MongoDB has better performance than the MYSQL in terms of both writing and reading operations. In the IoT Platform, the message payload is varying between 100 bytes and 200 bytes. For these types of messages, MongoDB has 70.95 messages/milliseconds in average and MySQL has 18.21 messages/milliseconds in average for writing. As it can be seen that only MongoDB satisfy the target expectations 40 messages/milliseconds in average. MongoDB also can satisfy the requirement with a single machine which has 61.22 messages/milliseconds throughput for writing.

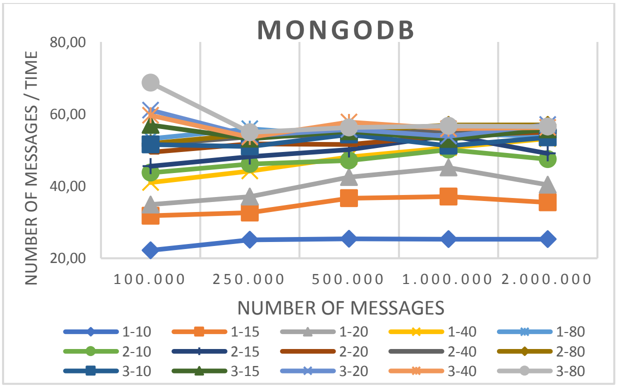

Figure 8.Result Graph of Read Operations on MongoDB

The DQS in the IoT platform applies reading operations on the DBMS. Test results show that MongoDB has better throughput than the MySQL. MongoDB has 55.07 messages/millisecond in average and MySQL has 46.66 messages/millisecond in average for two computers. Therefore, MongoDB is selected as DBMS for writing and reading operations in the IoT Platform.

ACKNOWLEDGEMENT

These results preliminary study of the project proposal applied to TUBITAK 1505 University Industry Collaboration Grant Program and the study is supported by EMKO Electronic A.Ş located in Bursa, Turkey.

REFERENCES

[1] “IoT.” [Online]. Available: https://connectedtechnbiz.wordpress.com/tag/internet-of-things/.

[2] “Gartner Says 6.4 Billion Connected ‘Things’ Will Be in Use in 2016, Up 30 Percent From 2015,” 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317.

[3] “JSON.” [Online]. Available: http://json.org/.

[4] “MongoDB Documentation,” *https://docs.mongodb.com/manual/*. .

[5] K. Chodorow, *Mongo DB: The Definitive Guide*. 2013.

[6] “CouchDB.” [Online]. Available: http://couchdb.apache.org/.

[7] “HBASE.” [Online]. Available: https://hbase.apache.org/.

[8] “Cassandra.” [Online]. Available: http://cassandra.apache.org/.

[9] “Amazon SimpleDB.” [Online]. Available: https://aws.amazon.com/simpledb/.

[10] “Redis.” [Online]. Available: https://redis.io/.

[11] G. Aydin, I. R. Hallac, and B. Karakus, “Architecture and implementation of a scalable sensor data storage and analysis system using cloud computing and big data technologies,” *J. Sensors*, vol. 2015, 2015.

[12] S. Chickerur, A. Goudar, and A. Kinnerkar, “Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications,” *Proc. - 8th Int. Conf. Adv. Softw. Eng. Its Appl. ASEA 2015*, pp. 41–47, 2016.

[13] Z. Parker, S. Poe, and S. V. Vrbsky, “Comparing NoSQL MongoDB to an SQL DB,” *Proc. 51st ACM Southeast Conf. - ACMSE ’13*, p. 1, 2013.

[14] Y. Li and S. Manoharan, “A performance comparison of SQL and NoSQL databases,” *IEEE Pacific RIM Conf. Commun. Comput. Signal Process. - Proc.*, no. August 2013, pp. 15–19, 2013.

[15] P. T. A. Mai, J. K. Nurminen, and M. Di Francesco, “Cloud databases for internet-of-things data,” *Proc. - 2014 IEEE Int. Conf. Internet Things, iThings 2014, 2014 IEEE Int. Conf. Green Comput. Commun. GreenCom 2014 2014 IEEE Int. Conf. Cyber-Physical-Social Comput. CPS 20*, no. iThings, pp. 117–124, 2014.

BIOGRAPHY(S)

Gizem Kiraz

Gizem Kiraz has completed her undergraduate (2016) in Computer Engineering Department from the Pamukkale University (PAÜ). Her postgraduate has started (2017) in Computer Engineering Department from the Uludag University. Currently, she is studying on the Internet of Things.

Cengiz Toğay

Cengiz Togay, Ph.D. is an assistant professor at Uludag University's Computer Engineering Department.

He obtained his undergraduate (1999) and MS (2001) in computer engineering from the Canakkale Onsekiz Mart University and his PhD (2008) in computer engineering department from Middle East Technical University. He has national and international patent applications, papers, articles and projects about Software Engineering, Secure Communications, Smart Cards, and Internet of Things(IoT).