ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/388355441

Comparative Analysis of Database Technologies for IoT Environments

Conference Paper · April 2024

CITATIONS
0

READS
54

2 authors, including:

Ahmet Özmen
Ağrı İbrahim Çeçen University
19 PUBLICATIONS **114** CITATIONS

SEE PROFILE

All content following this page was uploaded by Ahmet Özmen on 24 January 2025.

The user has requested enhancement of the downloaded file.

Comparative Analysis of Database Technologies for IoT Environments

A. Ozmen[1], M. Ertugrul[2]

[1]*Department of Electronics and Automation, Vocational High School, Agri Ibrahim Cecen University, Agri, Turkey, ORCID: https://orcid.org/0000-0002-3631-4883*
[2]*Department of Metallurgical and Materials Engineering, Karadeniz Technical University, Trabzon, Turkey, ORCID: https://orcid.org/0000-0003-1921-7704*

Abstract

The Internet of Things (IoT) refers to systems composed of interconnected devices capable of exchanging data. These systems offer services for data collection and analysis, making both data collection and storage critically important in IoT environments. With the widespread adoption of IoT systems, the demand for data storage solutions has increased. Traditionally, Structured Query Language (SQL) based databases have been favored due to their simplicity, robustness, flexibility, scalability, and performance. However, with the rise of Big Data, NoSQL databases have emerged, providing alternatives to traditional relational databases. NoSQL databases are non-relational, schema-less, do not require joins, are easily replicable, and are horizontally scalable. In IoT systems, data is typically stored in cloud-based systems, but there are scenarios where offline data storage becomes necessary. Thus, the choice of database in IoT systems becomes a critical factor depending on the use scenarios. This study is based on experiments conducted using the ESP32 microcontroller to compare the performance differences between SQL and NoSQL databases utilized in IoT-based systems. During these experiments, data insertion and retrieval operations were measured using Google's NoSQL-based Firebase database and the local SQLite database. Our findings indicate that Firebase performs faster in data insertion operations compared to SQLite. In data retrieval operations, especially with smaller data sets, the SQLite database on the SPI Flash File Storage (SPIFFS) file system exhibited better performance. Additionally, it was observed that the Little File System (LittleFS) file system could handle larger data volumes more effectively compared to SPIFFS.

Keywords: IoT, NoSQL, IoT database, SQLite, ESP32 microcontroller

Introduction

The Internet of Things (IoT) represents a technological revolution enabling communication between various devices and applications via the internet. Offering advanced data analytics and real-time processing capabilities, IoT paves the way for innovations across a wide range of fields from industry to healthcare services. This expanding ecosystem generates large-scale data streams, introducing significant challenges in the management of database systems[1].

While IoT devices commonly store their data in cloud systems, in scenarios lacking internet access, with weak communication, or where highly secure data collection is necessary, data is stored directly on embedded systems. This necessitates the establishment of databases within these embedded systems. In this context, SQLite emerges as a free solution that facilitates the creation of SQL-based databases in embedded systems[2,3]. Moreover, the emergence of large datasets has paved the way for the development of NoSQL databases, which differ from traditional relational database systems by offering schema-less structures[2,4–6]. The capability of NoSQL databases to handle unstructured data (such as texts, images, videos, emails) [7] distinguishes these technologies from traditional SQL systems.

In their study, Rautmare et al. (2016) conducted a comparative analysis of SQL and NoSQL databases within the context of a small-scale IoT sprinkler system application, investigating whether NoSQL demonstrated superior performance to SQL across various scenarios. They utilized MongoDB, a NoSQL database, to measure the time expended in executing 'Select' and 'Insert' queries against varying numbers of records and threads. Their findings indicated that MongoDB occasionally required shorter response times compared to MySQL; however, MySQL responses were more consistent when compared with those from MongoDB [4]. Reetishwaree et al. (2020) Utilizing sensor data like temperature/humidity and water level, they monitored a facility environment in real-time. The results revealed that while NoSQL databases performed 'Update' and 'Delete' operations more swiftly, SQL databases excelled in 'Select' operations[1]. Mladenova et al. (2022) presented a study and performance comparison of SQL and NoSQL databases concerning data from IoT devices and sensors. They tested 'Insert' and 'Select' operations under various technical variables. Their research concluded that MySQL was the preferred choice when multiple 'Select' queries were needed, whereas MongoDB was more suitable for scenarios with fewer 'Insert' queries[5].

Although extensive comparisons between these two types of databases exist in the literature[1,4,5,8] direct comparisons using performance data from low-capacity embedded systems like the ESP32 on SQLite-based implementations have not been conducted. This study includes comparisons of data insertion, search, and error-free record times in IoT contexts using the ESP32 microcontroller, comparing SQLite and Firebase. The findings presented provide valuable insights into how these database systems can be optimized within IoT devices.

Material and Method

In our research, we utilized the widely used ESP32 microcontroller in IoT systems. Two systems were created for the experiment, with the first system (D1) used to examine data "INSERT" and "SELECT" parameters. In the D1 system, the ESP32 was operated with three algorithm versions. The reason for applying different algorithms is the availability of two different File Systems (FS) that we can use with ESP32, namely SPIFFS and LittleFS. The performance evaluation of the SQLite database in our study was separately analyzed based on these two FS. For comparison, the NoSQL-based Firebase database produced by Google was used. The C++ code was written to utilize randomly generated temperature, humidity, and time information for 10 and 100 records for 10 sensors. The difference between the system time at the start of the " INSERT" and " SELECT" processes (T1) and the system time at the completion of the processes, T2, was calculated in microseconds for all records, and the average was computed[9].

Table 1. Materials used in IoT system

Device/Sensor	Usage Feature
ESP32	Microcontroller
DS18B20	Temperature Measurement
DHT11	Humidity Measurement
DS3231	Real-Time Clock (RTC)

The second system (D2) was designed to showcase the performance of faltering records due to communication issues in IoT communication networks. As seen in Table 1, the IoT system comprises 2 DS18B20 sensors, 1 DHT11 humidity sensor, DS3231 RTC sensors, and an ESP32. A total of 10 devices are present in the created IoT network, and the system has been recording temperature, humidity, and time data to the Firebase database at a rate of 144 data points per day for 6 months.

Results and Discussion

In our study, a total of three analyses were conducted using two IoT systems. The performance indicators obtained from the "INSERT" operation performed with the D1 system are presented in Figure 1A. The data obtained indicate that for the "INSERT" operation, 100 records took 109961 µs to be stored in the SQLite database set up on the LittleFS system (LittleFS + SQLite), 56031 µs on the SPIFFS system (SPIFFS + SQLite), and 23971 µs when saved on Firebase for 100 records. Similarly, for 1000 records, it took 324706 µs on LittleFS + SQLite and 138964 µs on SPIFFS + SQLite. Firebase was excluded from this analysis due to its hourly limit of 100 data points (which applies to free usage).

According to these results, the "INSERT" operation is much faster with the NoSQL database Firebase compared to the SQL database. Additionally, it was concluded that SPIFFS is faster than LittleFS.

The second analysis conducted with the D1 system is the "SELECT" operation. The results obtained from this operation are presented in Figure 1B and Table2. For the "SELECT" operation from a dataset containing 100 records, it took 165142 µs on LittleFS + SQLite, 78951 µs on SPIFFS + SQLite, and 419907 µs on Firebase. This analysis involved a "GET" operation from the Firebase database. For 1000 records, it took 791866 µs on LittleFS + SQLite and 783602 µs on SPIFFS + SQLite. While the processing time for "SELECT" operations was close between LittleFS and SPIFFS systems with large datasets, SPIFFS was found to be faster with smaller datasets. Moreover, cloud-based systems take significantly longer to retrieve data compared to local systems. Possible reasons for this could be attributed to security measures in cloud systems and the impact of internet speed on latency.

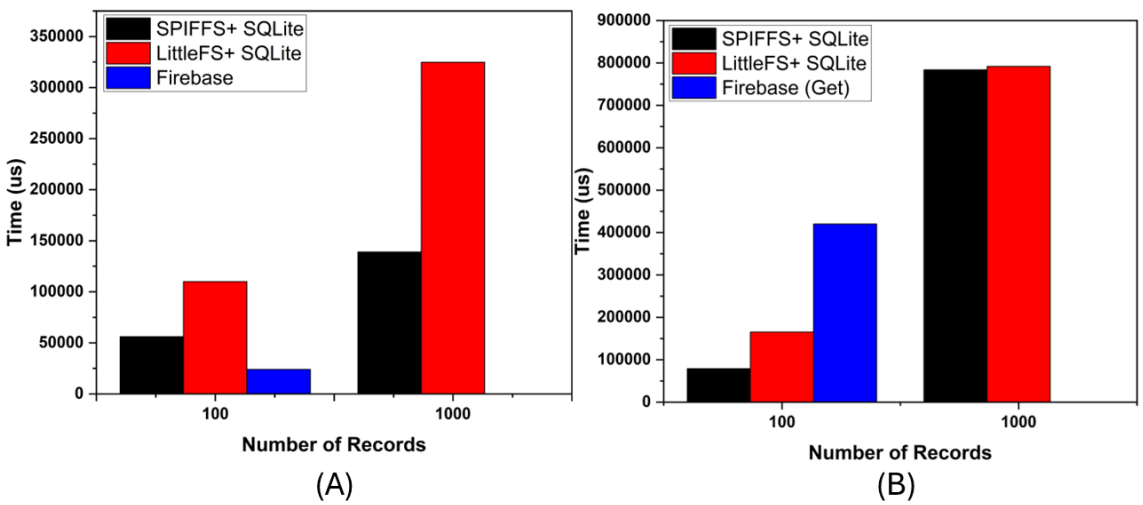

Figure 1. A) Latency in 'INSERT' Operations, Latency in 'SELECT' Operations

An additional investigation was conducted with the D1 system to determine the error-free record limit for LittleFS + SQLite and SPIFFS + SQLite. It was found that SPIFFS + SQLite becomes unresponsive after an average of 2000 records, while LittleFS + SQLite worked smoothly up to 10000 records. Since the system was configured for a maximum of 10000 records, the limit of LittleFS + SQLite was not reached. To conduct a similar analysis for Firebase, the D2 system was set up. With the D2 system, a Firebase data loss analysis was conducted, revealing a 2.46% data loss rate in 259200 records. This level of data loss can be attributed to communication and internet interruptions in the IoT system.

Table 2. Latency Times by Operation

Database	Number of Data	Operation Type	
		INSERT (µs)	SELECT (µs)
LittleFS+ SQLite	100	109961	165142
	1000	324706	791866
SPIFFS + SQLite	100	56031	78951
	1000	138964	783602
Firebase	100	23971	419907

Conclusions

This study presents a comprehensive comparison of SQL and NoSQL database systems within the context of IoT implementations, particularly focusing on embedded systems. The findings from our research using the ESP32 microcontroller highlight significant performance differences between the two types of databases in handling IoT data. Our experiments reveal that Firebase, a NoSQL database, demonstrates considerable speed advantages in "INSERT" operations compared to SQLite operating on both LittleFS and SPIFFS file systems. This suggests that for applications where rapid data entry is critical, and the system constraints allow, Firebase presents a more efficient option. In contrast, the "SELECT" operations show that SQLite on SPIFFS performs more efficiently than on LittleFS, especially with smaller datasets. This indicates that for systems where frequent data retrieval is necessary, SQLite on SPIFFS might offer better performance. Additionally, SPIFFS + SQLite systems become unresponsive at a lower data volume compared to LittleFS + SQLite, suggesting that LittleFS may be more suitable for applications requiring larger databases.

References

[1] S. Reetishwaree, V. Hurbungs, 2020. Evaluating the performance of SQL and NoSQL databases in an IoT environment, in: 2020 3rd Int. Conf. Emerg. Trends Electr. Electron. Commun. Eng., IEEE, (2020) 229–234.

[2] M. Slabinoha, S. Melnychuk, I. Manuliak, B. Pashkovskyi, 2022. Comparative analysis of embedded databases performance on single board computer systems using Python, in: 2022 IEEE 17th Int. Conf. Comput. Sci. Inf. Technol., IEEE, (2022) 222–225. https://doi.org/10.1109/CSIT56902.2022.10000475.

[3] G. Rani, T. Sharma, A. Sharma, 2023. Future Database Technologies for Big Data Analytics, in: 2023 Int. Conf. Intell. Syst. Commun. IoT Secur., IEEE, (2023) 349–354. https://doi.org/10.1109/ICISCoIS56541.2023.10100525.
[4] S. Rautmare, D.M. Bhalerao, 2016. MySQL and NoSQL database comparison for IoT application, in: 2016 IEEE Int. Conf. Adv. Comput. Appl., IEEE, (2016) 235–238. https://doi.org/10.1109/ICACA.2016.7887957.
[5] T. Mladenova, I. Valova, 2022. Performance Study of MySQL and MongoDB for IoT Data Processing and Storage, in: 2022 Int. Conf. Autom. Informatics, IEEE, (2022) 60–63. https://doi.org/10.1109/ICAI55857.2022.9960134.
[6] A. B. Moniruzzaman, S.A. Hossain, 2013. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison, International Journal of Database Theory and Application. 6 (2013) 43–45.
[7] A. Al-Sakran, H. Qattous, M. Hijjawi, 2018. A proposed performance evaluation of NoSQL databases in the field of IoT, in: 2018 8th Int. Conf. Comput. Sci. Inf. Technol., IEEE, (2018) 32–37. https://doi.org/10.1109/CSIT.2018.8486199.
[8] M. Muniswamaiah, T. Agerwala, C.C. Tappert, 2020. Performance of databases in IoT applications, in: 2020 7th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/2020 6th IEEE Int. Conf. Edge Comput. Scalable Cloud, IEEE, (2020) 190–192. https://doi.org/10.1109/CSCloud-EdgeCom49738.2020.00041.
[9] V. Pupezescu, M.-C. Pupezescu, L.-A. Perisoara, 2022. Optimizations of Database Management Systems for Real Time IoT Edge Applications, in: 2022 23rd Int. Carpathian Control Conf., IEEE, (2022) pp. 171–176.

View publication stats