JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 29 (2023) 360–365
Received: 29 June 2023
Accepted: 24 September 2023

A performance analysis of a cloud database on mobile devices

Badanie wydajności chmurowej bazy danych na urządzeniach mobilnych

Sylwester Kot*, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The article presents a performance analysis of Firebase cloud database. Two services, namely Realtime Database and Cloud Firestore, are examined, and their query speed are compared to those of the local SQLite database. Basic CRUD operations were examined, taking into account the number of records in the database, the size of individual records and the complexity of the database structure. Upon completion of the research, it was concluded that Realtime Database outperforms Cloud Firestore and cloud databases are slower than the local database when it comes to operations on a single record. However, when working with a larger volume of data, cloud database can achieve better results than SQLite. The accuracy of the outcome is also influenced by the stability of the network connection and the distance from the cloud server.

Keywords: performance; cloud database; Firebase; mobile device

Streszczenie

Artykuł dotyczy badania wydajności chmurowej bazy danych Firebase. Badane są dwie usługi: Realtime Database oraz Cloud Firestore, których prędkość zapytań jest porównywana do prędkości zapytań lokalnej bazy danych SQLite. Zbadane zostały podstawowe operacje CRUD z uwzględnieniem ilości rekordów w bazie danych, rozmiaru pojedynczego rekordu oraz rozbudowaniem struktury bazy. Po zakończeniu badań stwierdzono, że baza Realtime Database jest wydajniejsza od bazy Cloud Firestore oraz chmurowe bazy danych są wolniejsze od lokalnej bazy w przypadku operacji na pojedynczym rekordzie. Jednocześnie przy pracy na większej ilości danych chmurowe bazy danych potrafią osiągać lepsze rezultaty niż SQLite. Wpływ na dokładny wynik ma też stabilność łącza oraz odległość od serwera chmury.

Słowa kluczowe: wydajność; chmurowa baza danych; Firebase; urządzenie mobilne

*Corresponding author

Email address: sylwester.kot@pollub.edu.pl (S. Kot)

1. Introduction

With the development of the market for mobile business applications and applications designed for private clients, the issue of data storage arises. Application developers need to consider scalability and the type of data storage in their databases. The accessibility and storage method are also important factors. Thanks to the ubiquity of the Internet, one of the solutions that has emerged is the cloud-based database. It eliminates the need for local memory for storing information and potential hardware limitations. However, it requires continuous internet access, which may limit the speed of database operations.

Initially, all mobile applications utilized the only available database, SQLite. However, due to technological advancements and the efforts of tech giants such as Amazon and Microsoft, there are now numerous database offerings designed for mobile devices on the market. These databases come in both relational and NoSQL forms, differing in schema structure. Currently, one of the most popular cloud solutions for mobile devices is Firebase, a service created by Google. It offers two database solutions: Realtime Database and Firestore Database. This article presents a performance evaluation of these tools based on the speed of database operations, taking into account the number of records, database schema complexity, and individual record size. The results are then compared with a corresponding local database created in SQLite.

2. Literature review

The article "Cloud database as a service" by W. Al. Shehri [1] discusses cloud-based databases as the future standard for information storage, highlighting their scalability and hardware fault tolerance. It also presents parameters to consider when selecting an appropriate cloud database service, such as data size, portability, transaction capability, availability, and security.

In the article "Comparison of NoSQL and SQL Databases in the Cloud" by D. Hammes, H. Medero, and H. Mitchell [2], relational and non-relational databases are compared as cloud services. The CAP theorem is introduced as a means of identifying the main weaknesses of any database system. According to this theorem, any database implementation must choose two out of three properties: consistency, availability, and partition tolerance. Relational databases prioritize availability and consistency, while NoSQL databases lean towards consistency and partition tolerance. Performance tests were also conducted using Postgres and MongoDB databases, with the results favoring

Postgres as the more performant database. However, further research is deemed necessary to confirm these findings.

The article "A performance comparison of SQL and NoSQL Databases in the Cloud" [3] focuses on comparing a larger number of non-relational databases and examining their performance for key-value data. The results indicate that while non-relational databases are optimized for key-value data, not all services outperform the reference relational database. The results vary depending on the type of database operation and the number of operations performed. The authors determined that Couchbase and MongoDB are the fastest in read, write, and delete operations.

The article "The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test" [4] compares the Firebase database with MySQL using the Wilcoxon signed-rank test for paired observations to determine the optimal database for a mobile application intended for daily nutritional needs for young children. The tests were conducted for all CRUD operations and using the database structure used in the application. The results indicate that Firebase is a more efficient database.

The article "A comparison of NoSQL and SQL Databases over the Hadoop and Spark Cloud Platforms using Machine Learning Algorithms" [5] utilizes machine learning algorithms to create a NoSQL database from a relational database. The authors then evaluate the performance of the algorithms on both databases using the k-means method and random forest. The results indicate the superiority of the non-relational database in terms of operation speed, ranging from 26% to 54%.

The author of the article "A Performance Comparison of SQLite and Firebase Databases from a Practical Perspective" [6] compares two officially supported types of databases on the Android system: SQLite and Firebase, citing results from previous articles. The comparison involves basic database operations such as data insertion, retrieval of data and specific records, updating, and deletion. The study was conducted on a simple database model containing one entity with two properties: ID and text. The obtained results favor SQLite as the more efficient database in every operation except data deletion. It is also noted that Firebase performs better when sharing database resources with a larger number of users or when limited by local disk space.

In the article "On the Performance of Cloud-Based mHealth Applications: A Methodology on Measuring Service Response Time and a Case Study" [7] the Firebase database is being used in a performance test based on a prototype medical application. The study is being conducted on both Android and iOS platforms. Results show that the average response time for Android devices is slightly higher than that for iOS devices, which may be influenced by buffering and differences in the Firebase API design. It is worth noting that the response time is not dependent on the smartphone's battery mode. Additionally, in the case of a Wi-Fi connection, the average response time is lower by at least a factor of two as in the case of an LTE connection, indicating Wi-Fi as a more efficient connection for retrieving large data chunks.

The authors of the article "Monitoring the performance of cloud real-time databases: A firebase case study" [8] are investigating the performance of the Firebase database using Firebase Console and Google Cloud Monitoring tools. It was observed that Firebase Console provides detailed information regarding the database, while in cases of availability or latency issues, consideration should be given to using Google Cloud Monitoring, which automatically collects data on Firebase services. Additionally, it was determined that for the free version of Firebase, the maximum number of concurrent connections to the database is limited to 100.

3. Research method

Two forms of data storage provided by the Firebase service were subjected to the study: Realtime Database and Cloud Firestore, along with a local SQLite database implemented using the Room library. Three different database schemas were used: a simple key-value type, complex database with multiple objects in relationships, and a database with a single record of size 1KB. The complex database with relationships is an order model consisting of a product list and details related to payment and customer. In the SQLite database, the relationships were established using foreign keys, while in the Realtime Database, a reference to object IDs was added, and in Cloud Firestore, three collections were created: payment, order and customer.

For the purpose of this research two mobile apps were made: one for cloud databases and one for local database. To achieve optimal performance for the Cloud Firestore database batched writes were used for higher amount of data. For the Realtime Database transactions were not increasing the performance of operations so standard methods were used.

Cloud servers used in the research were located in Belgium and Western Europe. A Wi-Fi connection with a bandwidth of 30Mb/15Mb was used to connect to the servers.

The study was conducted using two smartphones: Xiaomi Redmi Note 8 Pro and Samsung A8 with the specifications shown in Table 1.

Table 1: Specifications of devices used in research

Parameter	Xiaomi Redmi Note 8 Pro	Samsung A8 (2018)
CPU	Mediatek Hello G90T 8x2.05 GHz	Samsung Exynos 7885 2.2 GHz
RAM	6 GB	4 GB
internal memory	64 GB	32 GB
operating system	Android 11	Android 9

3.1. Research scenarios

All test scenarios were conducted on 10, 100, 500, 1000, 2000 and 10000 records as well as on the three different database models mentioned before. Each test was performed 10 times, and the average execution time was calculated. After each test, the database and application memory were cleared and generated again for the next test. Prior to the research, the smartphone cache was cleared and all background applications were shut down.

The following operations were examined:

- creation of a database and populating it with data,
- inserting single record,
- editing single record,
- reading single record and entire database,
- deleting single record and entire database.

Additionally, the impact of database size on the performed operations was investigated.

4. Results

Due to the limits of free tier for Cloud Firestore, which is 20000 write and delete operations per day, scenarios using data sample of 10000 records were performed in the following days and tests for other databases were repeated to get the accurate results.

4.1. Inserting data

First scenario tested was about generating database and populating it with data.

The results (Table 2) indicate that for simple and complex data models SQLite is more efficient and has better average execution time regardless of number of records inserted into database. However, when the size of single record is about 1KB, cloud databases are better as more records are being added. Realtime Database manages to beat local database from 100 records onwards and Cloud Firestore manages to get faster time for 1000 records.

Table 2: Execution time of database generation

		Realtime Database	Cloud Firestore	SQLite
Number of records	Database model	Average execution time (ms)		
10	Key-value	63	1302	14.5
	Complex	98.5	1232	36
	Large	506.5	908	203
100	Key-value	111.5	3059.5	20
	Complex	370.5	9851	84.5
	Large	2145.5	5819	3515.5
500	Key-value	155.5	969.5	35
	Complex	1067	4551.4	307
	Large	9932.5	29235.5	21680
1000	Key-value	304.5	1775	43.5
	Complex	1624	9297.5	478
	Large	21574.5	36126	43436.5
2000	Key-value	420.5	3266.5	69.5
	Complex	3086	46962	941.5
	Large	14422	-	86111
10000	Key-value	1405	15018.5	400
	Complex	14852	-	3189
	Large	-	-	493641

Unfortunately, for record count above 2000 in case of Cloud Firestore and 10000 in case of Realtime Database there was an OutOfMemory exception thrown by the server side of database, which prevented getting the results for these conditions.

Next examined operation was about inserting single record to already existing database. This led to the following results (Table 3).

Table 3: Execution time of inserting single record

Database model	Database	Average execution time (ms)
Key-value	Realtime Database	54
	Cloud Firestore	98
	SQLite	5
Complex	Realtime Database	53
	Cloud Firestore	95
	SQLite	11
Large size	Realtime Database	103
	Cloud Firestore	181
	SQLite	20

Realtime Database manages to get faster execution time than Cloud Firestore for inserting single record in each database model scenario. The difference between them is nearly twofold. It can also be observed that the complex model does not cause a decrease in performance while a larger object size results in twice the execution time. On the other hand, the local database experiences a twofold increase in time for complex model and a fourfold increase for a 1KB-sized record.

It is worth noting that the number of records in database does not affect the execution time and the average execution time is composed of every tested case.

4.2. Editing data

The second scenario involves editing a single record in already existing database. In case of a key-value model, a value is changed. In a complex model the order status is changed and for the large set the big size field was changed. The results are as follows (Table 4).

Table 4: Execution time of editing data

Database model	Database	Average execution time (ms)
Key-value	Realtime Database	52
	Cloud Firestore	93
	SQLite	3
Complex	Realtime Database	53
	Cloud Firestore	77
	SQLite	5
Large size	Realtime Database	98
	Cloud Firestore	128
	SQLite	3

When it comes to editing data, execution times are similar to the inserting single record time. Realtime Database achieves differences at maximum of 5 milliseconds for the large record, while other models stay at 53-54 milliseconds. Cloud Firestore manages to get faster execution time in every model and the biggest difference is seen in the 1KB-sized record where the difference is over 50 milliseconds. However, both of the cloud databases have worse performance compared to local database, which has execution time below 5 milliseconds.

Just as with previous scenario, the number of records in database does not affect the execution time of editing a single record.

4.3. Reading data

The next scenario of the study examines database read operations. Both reading a single record and retrieving the entire database are evaluated.

When it comes to reading a single record (Table 5), Realtime Database stays at similar execution time no matter the database model. Even on the large-size model the difference between the best time is less than 10 milliseconds. Cloud Firestore performance is worse than the other cloud database, especially when it comes to simple key-value model where the average time exceeds 100 milliseconds. However, as with previous single-record operations, both databases have worse performance than local database.

Table 5: Execution time of reading single record

Database model	Database	Average execution time (ms)
Key-value	Realtime Database	56
	Cloud Firestore	122
	SQLite	4
Complex	Realtime Database	53
	Cloud Firestore	92
	SQLite	4
Large size	Realtime Database	62
	Cloud Firestore	94
	SQLite	4

Table 6: Execution time of reading whole database

		Realtime Database	Cloud Firestore	SQLite
Number of records	Database model	Average execution time (ms)		
10	Key-value	48.5	196	4.5
	Complex	69.5	169	28
	Large	158	496	3
100	Key-value	86	302	7.5
	Complex	160	149.5	39
	Large	680	2048	7
500	Key-value	101.5	276.5	8.5
	Complex	386.5	106.5	192
	Large	3077.5	10619.5	8.5
1000	Key-value	116	614.5	9.5
	Complex	556.5	171	236.5
	Large	5961	-	11.5
2000	Key-value	161.5	930.5	15
	Complex	1193.5	404	494.5
	Large	5223	-	19.5
10000	Key-value	416	5684.5	185
	Complex	3953.5	186.5	2290.5
	Large	-	-	84.5

Results of reading the entire database (Table 6) presents that Cloud Firestore excels in reading many data from the complex structure. The average execution time for all evaluated database size samples is less than 500 milliseconds. This is especially good for the larger number of records, where execution time for both Realtime Database and SQLite exceeds 2 seconds, whereas Cloud Firestore manages to get similar performance regardless of number of records to read. However, when it comes to simple key-value model and for the large-sized records Cloud Firestore performs the worst out of the three. Realtime Database shows that it can compete with local database on the complex and key-value model at larger set of data, however it falls short on 1KB-sized records. It can also be seen that SQLite struggles with complex database model as the execution time is much higher than for other database models.

Just as in the database generation scenario, for records above 1000 for Cloud Firestore and above 10000 for Realtime Database there was an OutOfMemory exception which made it impossible to gather the time of the operation.

It is worth mentioning that the average execution time for cloud databases decreased every time the read operation was repeated on the same database. This is caused by the device caching the data in memory so it doesn’t have to load whole data from the cloud server. The decrease is significant, however in this scenario only the first read from database was measured.

4.4. Deleting data

The last performed test scenario relates to single record and whole database deletion. Results of the tests can be seen below (Table 7, Table 8):

Table 7: Execution time of deleting single record

Database model	Database	Average execution time (ms)
Key-value	Realtime Database	56
	Cloud Firestore	90
	SQLite	4
Complex	Realtime Database	48
	Cloud Firestore	78
	SQLite	10
Large size	Realtime Database	52
	Cloud Firestore	99
	SQLite	3

Results for deleting single record shows that Realtime Database deletes data at around the same time no matter the database model. Cloud Firestore performs better at deleting a complex model record, which is about 20 milliseconds faster than other models. However, as in all previous one-record operations, local database performs much better providing nearly-instant execution time.

As for the database deletion operation the results vary depending on the database (Table 8). Realtime Database seems to be unaffected by both number of records and size of single record. The average execution time for key-value model increases slightly with number of records, on the other hand it decreases for large-sized record the more records are in the database. The performance is slightly worse for complex database model but all tests indicate that execution time is between 63-118 milliseconds, which is better than Cloud Firestore. The latter database has execution time of over a second for all of the database models above 500 records. The best performing model is key-value, following large-sized records at small quantity, but being outperformed by complex model at records above 500. What is worth mentioning, Realtime Database has better performance in deleting database consisting of larger size data than SQLite for higher amount of records deleted. It is best seen on the 10000 record test as the execution time for local database reaches over 10 minutes compared to only 63 milliseconds for Realtime Database.

Table 8: Execution time of deleting whole database

		Realtime Database	Cloud Firestore	SQLite
Number of records	Database model	Average execution time (ms)		
10	Key-value	70	541	2.5
	Complex	98.5	1354	26
	Large	59.5	495.5	15.5
100	Key-value	73.5	3060.5	3
	Complex	97.5	11253.5	17.5
	Large	118	3937	411.5
500	Key-value	76	905.9	5
	Complex	107.5	4713.4	12
	Large	71.5	6343.1	7649
1000	Key-value	90	1480	6
	Complex	103	8406.5	13.5
	Large	65.5	13397.5	20235
2000	Key-value	82.5	2702	6
	Complex	118.5	20833	19.5
	Large	78	-	148684
10000	Key-value	92.5	11755.5	13
	Complex	118.5	-	107.5
	Large	63	-	>100000

Looking at stability of operations, Realtime Database managed to successfully complete all delete operations regardless of number of records in database. Cloud Firestore however failed to delete the data for large records in number above 2000 and for the complex data model in the 10000 record test due to OutOfMemory exception which happened for the most of the multiple-records tests.

5. Conclusions

The article presents performance tests of selected cloud databases in CRUD operations.

The results obtained in all conducted tests indicate that Realtime Database outperforms Cloud Firestore in terms of efficiency. The only area where Cloud Firestore is favored is reading from a complex database with multiple records. In all other cases, the differences in execution time of operations are more than twice as large.

As for single-record operations, the performance of cloud databases is similar for all of the conducted operations. In both Realtime Database and Cloud Firestore there is slight increase in execution time for large-sized record database model, expect for reading a single record. However, compared to local database, the difference is visible as SQLite performs single-record operations nearly instantly, when cloud databases have respond time within 50-150 milliseconds. When comparing the results of cloud databases with the local one, it is important to consider the fact that in cloud databases, the execution time of operations includes both the data processing time and the server response time, which is doubled due to the query and response time. On the other hand, local databases do not have such limitations, which allows them to execute certain queries instantly, especially for smaller amounts of data.

When it comes to working with large number of data cloud databases manage to outperform local database in generating and deleting database for objects of 1KB size. The result depends on the database, as Realtime Database performs better in simple database model, whereas Cloud Firestore is optimized for complex model and performs exceptionally well in reading database for that type of data. However, it works worse with a simple key-value data model. Another thing to consider is the size limit of operations, as in the research, especially for number of records exceeding 2000, server happened to throw the OutOfMemory exception which make it unable to process big amount of data at the same time, which is not happening in SQLite. For the Cloud Firebase the maximum number of operations for a single transaction can’t exceed 500 [9] and for the Realtime Database the limits are only for maximum size of a single response, which is 256MB and write rate at 1000 writes/second [10].

When assessing the performance of cloud databases, it is essential to take into account the conditions that exist in mobile devices. Significant variations in operation times can be attributed to factors such as network stability, connection quality, distance from the cloud server, or server latency. Moreover, when operating with a larger number of records, there is a chance of reaching the concurrent data transmission limit, resulting in exceptions within the application. However, when utilizing SQLite database with the Room library and the MVVC model implementation, such issues do not occur, and queries for a larger number of records can take over 10 minutes. Additionally, Firebase services impose limitations on data size, as observed in Realtime Database, or daily limits on read, write, and delete operations, as seen in Cloud Firestore. Utilizing paid tiers would allow examining how operation times change with a larger number of records, a scenario commonly encountered in advanced IT systems where data volume can reach several million entries.

Summarizing all the points presented above, Realtime Database performs better in basic operations than Cloud Firestore, however Cloud Firestore is more optimized for complex data structure. Both of the cloud databases perform worse than local one in single-record operations but can outperform it if working at large number of data. The user has to bear in mind the limits of a single call according to the documentation.

References

[1] W. Al Shehri, Cloud Database Database as a Service, International Journal of Database Management Systems 5(2) (2013) 1-12.

[2] D. Hammes, H. Medero, H. Mitchell, Comparison of NoSQL and SQL Databases in the Cloud, SAIS 2014 Proceedings (2014) 12-20.

[3] Y. Li, S. Manoharan, A performance comparison of SQL and NoSQL databases, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM, Victoria, BC, Canada, August 27-August 29, 2013, IEEE (2013) 15-19.

[4] M. Ohyver, J. V. Moniaga, I. Sungkawa, B. E. Subagyo, I. A. Chandra, The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test, Procedia Computer Science 157 (2019) 396-405.

[5] C. H. Lee, Z. W. Shih, A Comparison of NoSQL and SQL Databases over the Hadoop and Spark Cloud Platforms using Machine Learning Algorithms, 2018 IEEE International Conference on Consumer Electronics-Taiwan, ICCE-TW, Taichung, Taiwan, May 19-May 21, 2018, IEEE (2018) 1-2.

[6] A. T. KABAKUŞ, A Performance Comparison of SQLite and Firebase Databases from A Practical Perspective, Düzce Üniversitesi Bilim ve Teknoloji Dergisi 7(1) (2019) 314-325.

[7] D. Inupakutika, G. Rodriguez, D. Akopian, P. Lama, P. Chalela, A. G. Ramirez, On the Performance of Cloud-Based mHealth Applications: A Methodology on Measuring Service Response Time and a Case Study, IEEE Access 10 (2022) 53208-53224.

[8] M. F. Younis, Z. S. Alwan, Monitoring the performance of cloud real-time databases: A firebase case study, 2023 Al-Sadiq International Conference on Communication and Information Technology, AICCIT, Al-Muthana, Iraq, July 4-July 6, 2023, IEEE (2023) 240-245.

[9] Usage and limits | Firestore | Firebase, https://firebase.google.com/docs/firestore/quotas?hl=en [15.06.2023]

[10] Realtime Database Limits | Firebase Realtime Database, https://firebase.google.com/docs/database/usage/limits?hl=en [15.06.2023]