PERTANIKA PROCEEDINGS

Journal homepage: http://www.pertanika.upm.edu.my/

Performance Analysis of an OAuth 2.0-Based Authentication and Authorization System Using a Redis In-memory Database

Akbar Mahmudin, Hendriyana* and Mochamad Iqbal Ardimansyah

Department of Software Engineering, University Pendidikan Indonesia, 229 Dr. Setiabudi Street, Bandung 40514, West Java, Indonesia

ABSTRACT

The OAuth 2.0 framework is a protocol that allows third-party applications to gain authorized access to user resources. However, inefficient token management and storage can adversely affect the workload of OAuth 2.0 systems, especially when system traffic is high, such as performance degradation, slow response time and security vulnerabilities. Therefore, token management and storage are essential in the OAuth 2.0 ecosystem to ensure user security and convenience. This research aims to analyze the impact of Redis in-memory database implementation on token management on OAuth 2.0 system performance. In addition, the research focuses on the results of load testing conducted on the OAuth 2.0 system, referring to software quality based on performance efficiency in accordance with ISO 25010, including aspects of time response, throughput and resource utilization. Load testing simulates the authentication and authorization process using the OAuth 2.0 system to observe the effect after the implementation of Redis in token management at user load levels of 100, 300, and 500. The results show that Redis has an influence on the performance of the OAuth 2.0 system on response time, throughput, and memory usage. On the aspect of CPU usage, it does not show any influence, even after Redis is implemented. The advantages of Redis can be attributed to the memory-based storage that enables faster read-write operations, especially for single data with low latency.

Keywords: Authentication, authorization, IMDB, OAuth, Redis

ARTICLE INFO

Article history:
Received: 30 April 2025
Published: 17 July 2025

DOI: https://doi.org/10.47836/pp.1.3.007

E-mail addresses:
akbarmahmudin@upi.edu (Akbar Mahmudin)
hendriyana@upi.edu (Hendriyana)
iqbalardimansyah@upi.edu (Mochamad Iqbal Ardimansyah)
* Corresponding author

INTRODUCTION

OAuth 2.0 is one of the most popular authentication and authorization protocols as it allows third-party applications to gain legitimate access to user resources without the need to reveal user passwords (Kiani, 2020). The Authorization Code Grant offers enhanced security benefits due to the fact that the authorization code is transmitted exclusively over a secure channel between the client and the OAuth server (Fett et al., 2016). Inefficient token storage can result in performance degradation, longer response times, and even potential security vulnerabilities (Safaryan et al., 2020). Therefore, there is a need for a database system that can efficiently store and manage tokens.

One type of database with fast read performance is an in-memory database, where all information is stored in main memory (Kausar et al., 2022; Safaryan et al., 2020). When it comes to storing data that is stored and retrieved from volatile memory, Redis has better performance than Memcached in terms of data persistence and handling heavy traffic loads (Alami et al., 2018). In addition, Redis has the best read performance of MongoDB and Cassandra (Kausar et al., 2022; Kabakus & Kara, 2017). This is due to Redis's ability to access data with very low latency. With faster and more efficient request response, Redis has the potential to improve user experience in the OAuth 2.0 ecosystem in system performance.

PROBLEM STATEMENT

The research implements the Authorization Code Grant flow using the Redis in-memory database as a token database, with the aim of analyzing the performance of the OAuth 2.0-based authentication and authorization system. This will entail an in-depth examination of the system's response time, throughput, and utilization (Li et al., 2017; Wang & Wu, 2019; Zulfa et al., 2020). The objective of this study is to conduct a comprehensive examination of the operational efficacy of the Redis system (Tkachenko & Lukianiuk, 2021). The research reference for subsequent investigations that may undertake a comparative analysis of Redis with other IMDBs.

RESEARCH QUESTIONS

On the server side or backend, using a microservices architecture shown in Figure 1.

Testing will be conducted on the backend or server side of the OAuth 2.0 system (Auth Server) using load testing to test each system performance metric, namely response time, throughput, CPU usage and memory usage. The description of each metric used in this study is as follows:

1. Average response time (M1) required by the OAuth 2.0 system to respond to a user or task (Equation 1).

$$X = \sum_{i=1\ ke\ n} \frac{A_i}{n} \qquad [1]$$

Ai = time required by the OAuth 2.0 system, n = number of responses

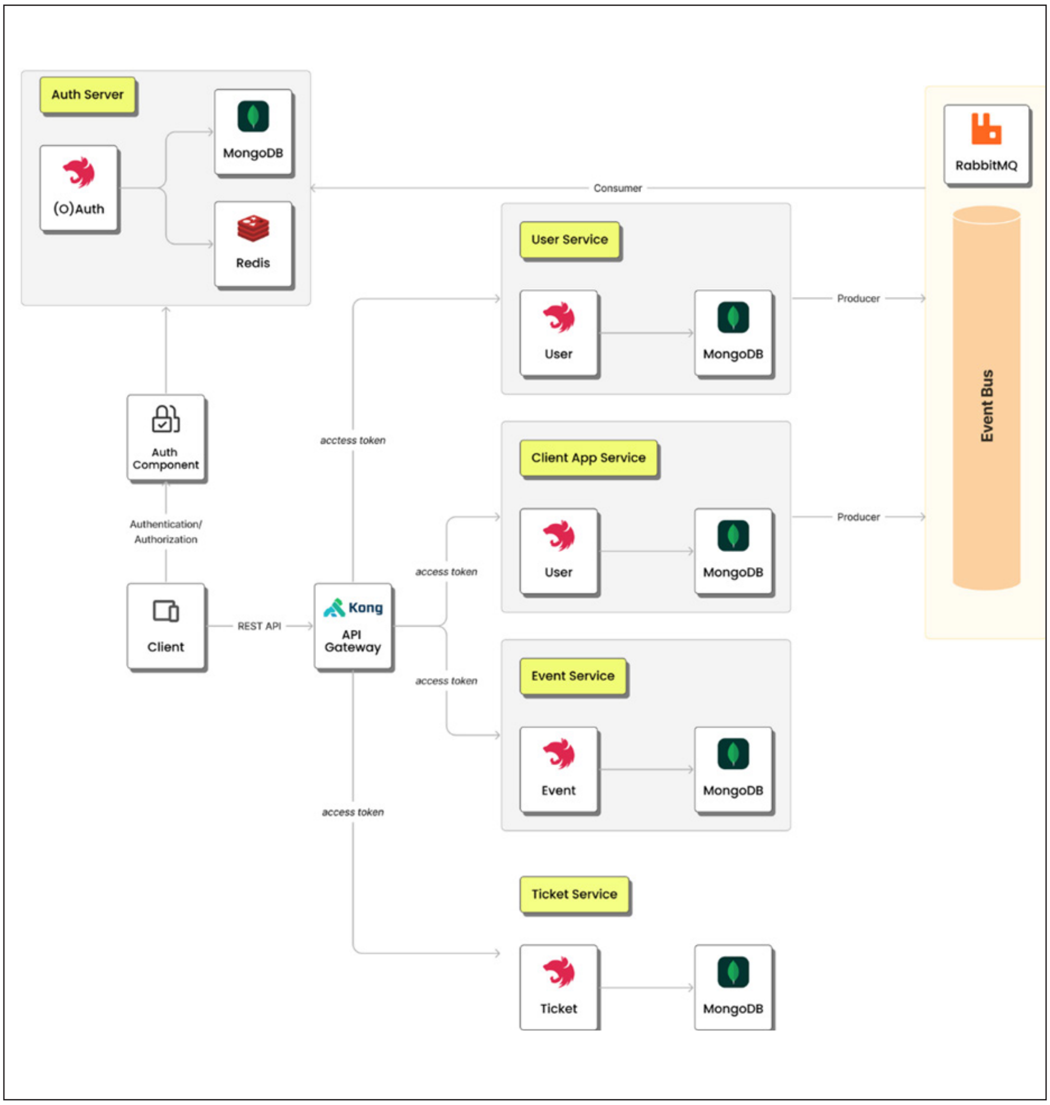


Figure 1. System architecture

2. Average throughput (M2) that the OAuth 2.0 system can process in a unit of time (seconds) (Equation 2).

$$X = \sum_{i=1\ ke\ n} \frac{A_i/B_i}{n} \qquad [2]$$

Ai = number of commands successfully processed, *Bi* = time to execute a command

3. Average CPU usage (M3) when executing commands within a certain period (Equation 3).

$$X = \sum_{i=1\ ke\ n} \frac{A_i / B_i}{n} \qquad [3]$$

Ai = CPU usage size to execute a command, *Bi* = operation time to execute a command

4. Average memory usage (M4) when executing a command in a certain period (Equation 4).

$$X = \sum_{i=1\ ke\ n} \frac{A_i / B_i}{n} \qquad [4]$$

Ai = memory usage size to execute a command, *Bi* = memory usage time to execute a command

The tests were conducted by incrementally increasing the load based on the number of virtual users, i.e., 100, 300, and 500 users simultaneously within 300 seconds (Suryawana & Muliantaraa, 2024). Based on previous research, this number can identify and address potential bottlenecks and performance issues without overloading the system in the early stages of testing (Tkachenko & Lukianiuk, 2021). Figure 2 is a scenario that simulates the user authentication and authorization process on a third-party application.

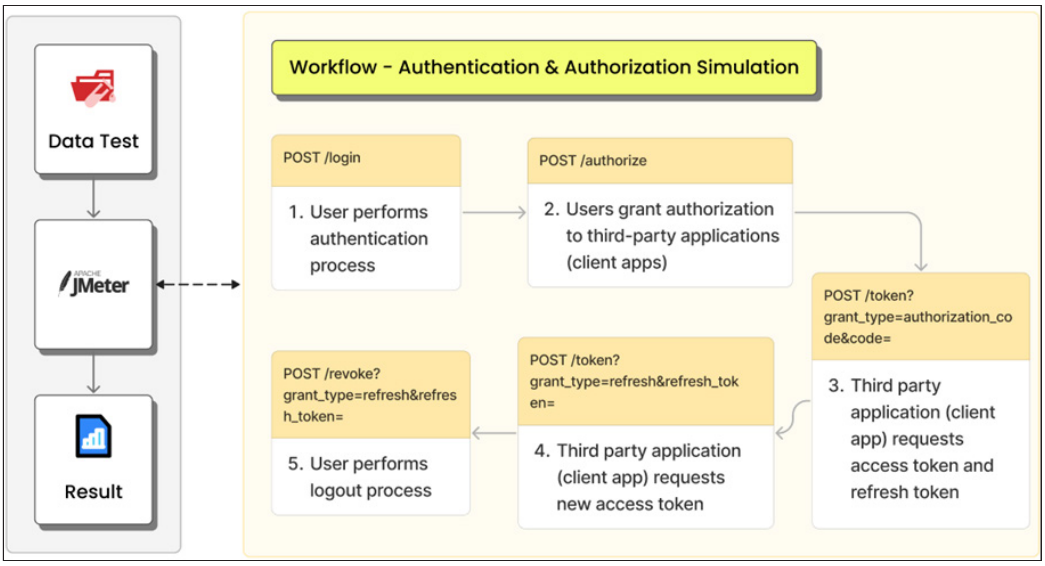

Data Test
JMeter
Result
Workflow - Authentication & Authorization Simulation
POST /login
1. User performs authentication process
POST /authorize
2. Users grant authorization to third-party applications (client apps)
POST /token?grant_type=authorization_code&code=
3. Third party application (client app) requests access token and refresh token
POST /token?grant_type=refresh&refresh_token=
4. Third party application (client app) requests new access token
POST /revoke?grant_type=refresh&refresh_token=
5. User performs logout process

Figure 2. Testing scenario

The findings of the research conducted on load testing the OAuth 2.0 system under two different conditions to assess the impact of implementing Redis in the OAuth 2.0 system on response time, throughput, CPU usage and memory usage. The condition before implementing Redis is named pre-Redis, and after implementing Redis is named post-Redis.

1. Time Response Aspect (M1)
   The average response time of the OAuth 2.0 system shows a significant performance improvement, with a load of 100 showing an increase of 9.18%, a load of 300 showing an increase of 6.41%, and a load of 500 showing an increase of 4.06%. Figure 3 below provides a visualization of the performance improvement of both OAuth 2.0 system conditions.
2. Throughput Aspect (M2)
   The OAuth 2.0 system shows significant performance improvement, with load 100 showing an increase of 7.75%, load 300 showing an increase of 5.10%, and load 500 showing an increase of 3.14%. Figure 4 visualizes throughput aspect observations, demonstrating that the post-Redis system condition is capable of handling a greater number of requests per second.
3. CPU Usage Aspect (M3)
   The performance improvement is not as significant, with load 100 only showing a 1.54% improvement, load 300 showing a lower improvement of 0.16%, and load 500 showing a 1.67% improvement based on the average CPU usage. Figure 5 provides

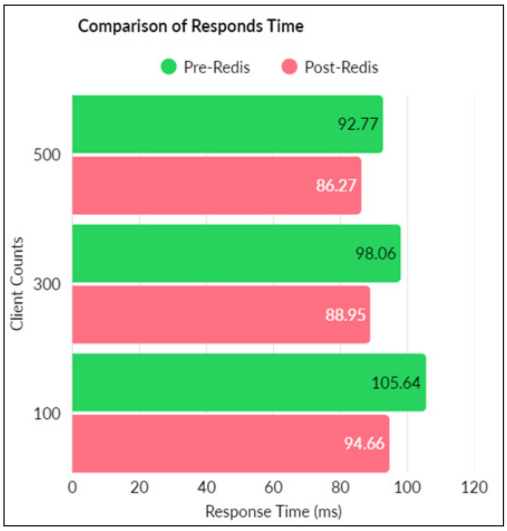


Figure 3. Bar chart of scenario M1 for each load

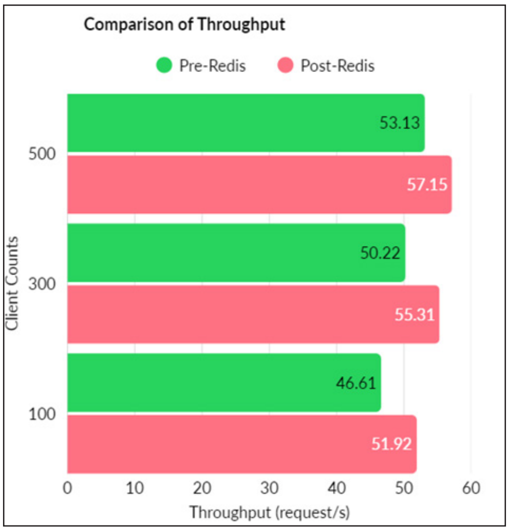
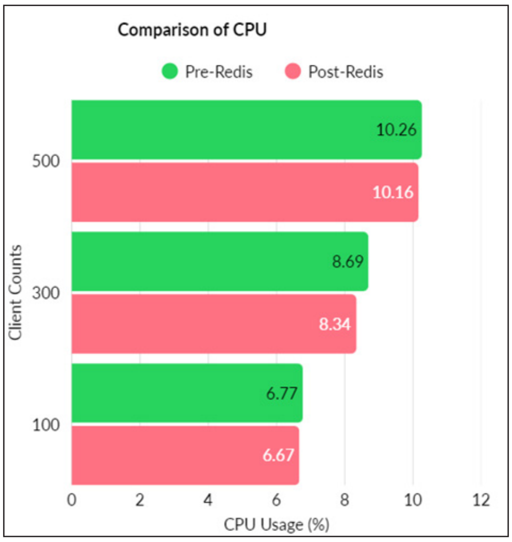


Figure 4. Bar chart of scenario M2 for each load



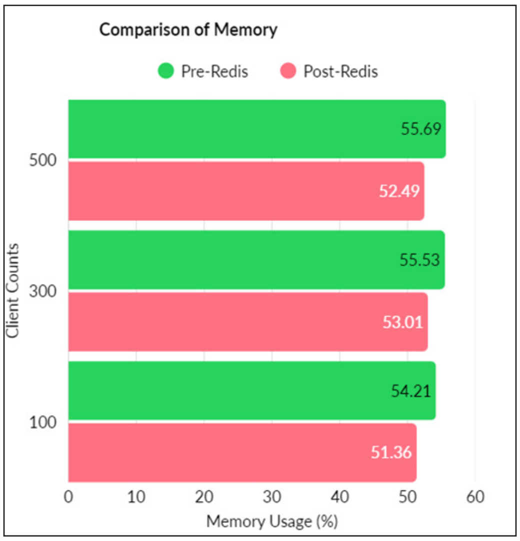


Figure 5. Bar chart of scenario M3 for each load

Figure 6. Bar chart of scenario m4 for each load

a visualization of the CPU usage of both system conditions at each load. CPU usage aspects reveal a consistent and gradual increase in CPU usage over a period, with a slight decrease over time, in both system conditions.

4. Memory Usage Aspect (M4)
   Figure 6 is a visual representation of the comparison between the two system conditions. This system condition exhibits superiority across various types of loads, including the highest loads. This aspect does not require further discussion, given that Redis is an in-memory database that stores all data in RAM.

CONCLUSION

A major challenge faced in the OAuth 2.0 ecosystem is the efficient storage and management of tokens. This research is motivated by the flexibility of the OAuth 2.0 standard and the performance reliability of the in-memory database Redis, which has the potential to access token data with very low latency and with faster and more efficient request response. The results show that the implementation of Redis in token management in OAuth 2.0-based authentication and authorization systems has an effect on response time, throughput, and memory usage. However, there is no significant effect on the CPU usage aspect after the Redis implementation. When viewed from the largest load, the percentage of performance improvement in the aspect of response time increased by 4.06%, throughput increased by 3.14%, CPU usage increased by 1.67%, and memory usage increased by 2.87%. The superiority of Redis in almost all aspects can be attributed to the memory-based storage that enables faster read/write operations, especially for single data with low latency.

ACKNOWLEDGEMENT

This research was supported by the Universitas Pendidikan Indonesia under research grant No. 385/UN40.LP/PT.01.03/2024. We also acknowledge the Software Engineering Study Program for providing access to laboratory facilities during this study.

REFERENCES

Alami, A. E., Bahaj, M., & Khourdifi, Y. (2018). Supply of a key value database redis in-memory by data from a relational database. In *2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)* (pp. 46-51). IEEE. https://doi.org/10.1109/MELCON.2018.8379066

Fett, D., Küsters, R., & Schmitz, G. (2016). A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1204-1215). ACMDL https://doi.org/10.1145/2976749.2978385

Kabakus, A. T., & Kara, R. (2017). A performance evaluation of in-memory databases. *Journal of King Saud University - Computer and Information Sciences*, *29*(4), 520–525. https://doi.org/10.1016/j.jksuci.2016.06.007

Kausar, M. A., Nasar, M., & Soosaimanickam, A. (2022). A study of performance and comparison of nosql databases: Mongodb, cassandra, and redis using ycsb. *Indian Journal of Science and Technology*, *15*(31), 1532–1540. https://doi.org/10.17485/IJST/v15i31.1352

Kiani, K. (2020). *How to secure your oauth implementation*. Citeseer. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dbd29ff47bb6b600e24250d9e4a34d985e044537

Li, S., Jiang, H., & Shi, M. (2017). Redis-based web server cluster session maintaining technology. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)* (pp. 3065-3069). IEEE. https://doi.org/10.1109/FSKD.2017.8393274

Safaryan, O., Pinevich, E., Roshchina, E., Cherckesova, L., & Kolennikova, N. (2020). Information system development for restricting access to software tool built on microservice architecture. *E3S Web of Conferences*, *224*, Article 01041. https://doi.org/10.1051/e3sconf/202022401041

Suryawana, A. I., & Muliantaraa, A. (2024). Database performance optimization using lazy loading with Redis on online marketplace website. *Jurnal Elektronik Ilmu Komputer Udayana P-ISSN*, *2301*, Article 5373.

Tkachenko, V., & Lukianiuk, S. (2021). Analysis of the use of the Redis in the distributed order processing system in the restaurant network. *Technology Audit and Production Reserves*, *5*(2), 39-43. https://doi.org/10.15587/2706-5448.2021.238460

Wang, J., & Wu, J. (2019). Research on performance automation testing technology based on JMeter. In *2019 International Conference on Robots & Intelligent System (ICRIS)* (pp. 55-58). IEEE. https://doi.org/10.1109/ICRIS.2019.00023

Zulfa, M. I., Fadli, A., & Wardhana, A. W. (2020). Application caching strategy based on in-memory using Redis server to accelerate relational data access. *Jurnal Teknologi dan Sistem Komputer*, *8*(2), 157–163. https://doi.org/10.14710/jtsiskom.8.2.2020.157-163