UNIVERSITAS IPWIJA
SK Kemendikbudristek RI No. 627/E/O/2022
Jl. H. Baping No.17 Kel. Susukan, Kec. Ciracas
Jakarta Timur. 13750 Telp. 021-22819921
E-mail : contact@ipwija.ac.id https://ipwija.ac.id
UNIVERSITAS IPWIJA

No. : 108/IPWIJA.LP2M/PT-00/2023
Perihal : Edaran Penelitian Dosen
Lampiran : -

Kepada Yth.
Bapak/Ibu Dosen Tetap
Universitas IPWIJA

Dengan hormat,

Sehubungan dengan dimulainya semester Ganjil Tahun Akademik 2023/2024, perlu diingat kembali tentang salah satu kewajiban Tri Dharma Perguruan Tinggi Dosen yaitu melaksanakan penelitian. Berkenaan dengan hal itu maka disampaikan:

- Terima kasih kepada Bapak/Ibu Dosen Peneliti yang telah merespon Surat Edaran Kepala LP2M No.043/IPWIJA.LP2M/PT-00/2023 tanggal 6 Maret 2023 tentang Kegiatan Bidang Penelitian dengan aktif berperan dalam berbagai pertemuan ilmiah, melaksanakan penelitian dan mempublikasikan hasil penelitian di berbagai jurnal ilmiah.
- Dosen yang telah menyelesaikan laporan penelitian dan mempublikasikannya pada semester Genap Tahun Akademik 2022/2023 diharapkan mengajukan usulan penelitian baru kepada LP2M.
- Dosen yang telah menyelesaikan tahap akhir penelitian diharapkan dapat segera membuat laporan hasil penelitian dan mempublikasikannya di semester Ganjil Tahun Akademik 2023/2024.
- Pada semester Ganjil Tahun Akademik 2023/2024, Dosen diharapkan aktif mengikuti berbagai kegiatan yang berkaitan dengan penelitian seperti: pertemuan ilmiah, sharing knowledge, diseminasi, pelatihan, seminar, proceeding, publikasi dan lain sebagainya.
- Agar penelitian dosen sesuai dengan Rencana Strategis penelitian institusi maka diharapkan kerjasama pada Dosen dengan jalan senantiasa berkoordinasi dengan LP2M, Prodi dan setiap elemen di UNIVERSITAS IPWIJA.

Demikian edaran ini disampaikan dan terima kasih.

Jakarta, 4 September 2023

Dr. Ir. Titing Widyastuti, M.M.
Kepala LP2M Universitas IPWIJA

Tembusan : Rektor Universitas
Wakil Rektor 1
Wakil Rektor 2

Hybrid Round-Robin Load Balancing with Indexing for Fast Collection of Personal Data at Big Data Webservices

1st Cakra Adipura Wicaksana
Department of Informatics
University of Sultan Ageng Tirtayasa
Cilegon, Indonesia
cakraadipura@untirta.ac.id

2nd Hanif Anggit Wicaksono
Department of Electrical Engineering
University of Sultan Ageng Tirtayasa
Cilegon, Indonesia
hanifanggit@gmail.com

3rd Supriyanto
Department of Electrical Engineering
University of Sultan Ageng Tirtayasa
Cilegon, Indonesia
supriyanto@untirta.ac.id

4th Hafiyyan Putra Pratama
Department of Telecommunication System
Indonesia University of Education
Purwakarta, Indonesia
hafiyyan@upi.edu

5th Mohammad Fatkhurrokhman
Department of Electrical Engineering
Vocational Education
University of Sultan Ageng Tirtayasa
Serang, Indonesia
fatkhur0404@untirta.ac.id

6th Ainatul Radhiah
Department of Software Engineering
University of IPWIJA
East Jakarta, Indonesia
ainatulradhiah@ipwija.ac.id

2023 1st International Conference on Advanced Engineering and Technologies (ICONNIC) | 979-8-3503-0648-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICONNIC59854.2023.10467647

***Abstract* – Having the necessary skills to differentiate between front-end and back-end services is crucial. In today's world, there is a growing emphasis on developing web services, a type of software that provides services that other software programs can access via the Web. However, this increased demand from users can lead to system inefficiencies, resulting in sluggishness and bottlenecks. This study aims to assess and shed light on the load-balancing capabilities of the round-robin approach to address these issues. The load-balancing server duty is divided up across numerous Web servers running the Apache Web Server using this method, while the database server in use is MySQL. In studies with 80 users and 30 trials without round robin, the average reaction time from 10 trials was 6394 ms; the average minimum response time was 940 ms; the average maximum response time was 16349 ms; and the error was 0.00%. As a closing remark, utilizing the Round Robin method and indexing will reduce response time and produce a constant number when compared to not utilizing them.**

Keyword - Web API, Round Robin, Load Balancing, and webservices

I. INTRODUCTION

The level of server activity from internet users is very high, of course, this will have an impact on information providers. The performance of Web servers and databases as media content providers is always expected to meet all the needs of users. This impact is certainly not desired by several agencies whose all activities are dependent on computer networks [1]. Therefore, these agencies do not hesitate to allocate funds to purchase special server devices with high capabilities. If not taken seriously, this could result in the servers being overloaded with requests from users. This is because the demand from the user is greater than the server's ability to provide services.

Everyday service requests from users are always increasing [2]. This is of course related to the increasing number of devices that can use internet facilities such as computers, laptops, netbooks, smartphones, tablets, and other devices. Websites with high data traffic can cause heavy workloads on the server side, which in turn will result in decreased server performance and even overall system failure.

In the realm of contemporary digital interactions, web APIs serve as a critical conduit for efficient data exchange between users and web services. However, the issue of sluggish response times for HTTP requests made through web APIs presents a persistent and consequential challenge. This phenomenon, characterized by extended intervals of data retrieval or webpage loading, can arise from a multitude of factors, including network congestion, server limitations, resource-intensive client-side computations, or suboptimal coding practices [3]. In the context of web APIs, the implications of inadequate HTTP response speed extend beyond mere inconvenience, encompassing reduced user satisfaction and diminished interactivity and potentially compromising the overall success of web-based platforms heavily reliant on seamless API-driven interactions. Addressing the intricacies of delayed HTTP responses in the realm of web APIs is of paramount importance. By uncovering and mitigating the root causes of this issue, not only can the performance and effectiveness of web API-based services be enhanced, but the overall digital experience of users can also be elevated, fostering a more engaging and productive online environment.

In this study mentioned here [4], certes (Client Response Time Estimated by the Server), a server-based online mechanism for web servers to assess client perceived response time as if measured at the client, is presented. However, certes does not illustrate integration with databases. In research [5][6], it tried to build a huge database that collected data from Twitter. Because of the limitations of storing data, there is a bottleneck in acquiring the data. Consequently, the system takes too much time to retrieve. A very detailed explanation was also presented in [7], More interestingly, the research used six servers as an example, and surprisingly, the system was loaded with plenty of requests concurrently. However, there is no explanation or combination in conjunction with the database. Comparisons between the round-robin method and the least method are also demonstrated in [8]. The problem is still in the same case; more specifically, it presented a well-crafted system that only balanced the incoming requests to the server and made no reference to the database. A study in [9], more interestingly, also compares two methods between the

Round-Robin and the least-connection algorithm. The experimental results reveal that the least-connection algorithm is superior to round-robin. However, since there is a drawback to round-robin, it needs to be improved properly. The previously mentioned ponder was centered on adjusting the approaching requests, not on the database. Eventually, how to plan a brief procedure to bargain with database frameworks but moreover to make outstandingly imperative influence.

One solution to overcome this problem is to apply load-balancing techniques [10][11][12][13]. A load balancing approach is a strategy for evenly distributing the traffic load across more connection lines so that traffic can flow more efficiently, response times can be shortened, and overload on one connection line can be avoided. When a server already has several users and has reached its capacity limit, load balancing is used [14]. One of the load-balancing tools is Nginx [15]. One of the operating system's process scheduling techniques is round robin. The purpose of a round robin method, also known as a cyclic executive, is to split the time allotted for each procedure into equal sections and to do so in a cyclical fashion. Round robin scheduling is easy to implement, and free of starvation [16]. Round robin is designed for a time-sharing system [17]. Software systems that can be quickly constructed by combining a variety of pre-existing web APIs from diverse sources include microservice services.

II. PROPOSED STRATEGY

One technology that has not been utilized is the use of a database system that can ensure the integrity of the data sent and can be used to authenticate/authorize two different applications. Using authentication can help increase the security of the data. Then optimizing virtual servers that previously used a single server and then optimized using multi servers. It is advantageous to employ numerous servers to facilitate data movement between them and to maximize the use of resource delivery programs to prevent server overload. In this study, we try to utilize the combination of the round-robin algorithm and the modification of the database to balance the request systems.

A. Database System

The database design used in this study using a MySQL database consists of a table with six fields, including *ID, name, phone number, city, address, and province.* The MySQL database belongs to the XAMPP application in this regard. The utilization of MySQL is based on the research in [18] that stated MySQL can still be used for a huge amount of data and the research in [19] for a plan for keeping track of a lot of data (perhaps tens of millions of rows).

- Table Structure

Row ids as primary keys have a big int type format with a total length of 20. Then for name, phone number, city, address, and province, they must have different lengths. For example, in this experiment, the name must be 35 characters long, the phone number must be 15 characters long, the address must be long text, and the province must be 20 characters long. Finally, all of them must have a varchar-type format. It really should be advised that the characters should not use 255 at all because it will slow down the database. The database designed has real data imported from Microsoft Excel in.csv format totaling ***59,283*** rows, real person data from all over Indonesia, which aims to be a benchmark for calculating response time, min. response time, max. response time, and errors.[20].

- Additional indexes in tables

An index is an object in MySQL that contains ordered data from the values in one or more fields in a table. Just like the table of contents in a book, the index is mainly used to speed up the search for a data set with certain conditions involving a combination of fields that have been defined in an index. Without an index, searching for data will usually take a long time, especially if the data is already on a very large scale. The use of an index was inspired by the proposed method in [21] that uses a three-dimensional or signature-based structure as well as in [22].

B. Load balancing system

This research uses network system design by applying load balancing algorithm as a mechanism in server selection. Block diagram of the Load balancing system in Figure 1.

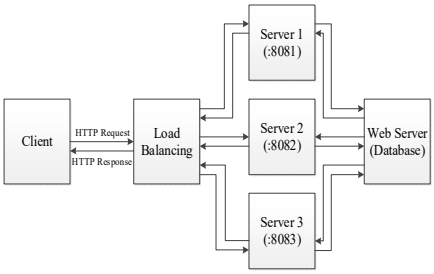

Figure 1 Flowchart Load balance System

Figure 1 is an experimental scenario using 3 virtual servers connected to load balancing. Each virtual server is started via Command Prompt. The load balancing process in general, which is the sharing of the load on the server in dividing traffic automatically. At the beginning of the process, the client makes a request to the server. Requests from clients will enter the Load balancing controller and check the server. Then the controller will select the server according to the load balancing algorithm. The client request will be forwarded to the server queue. The server will respond and send data to the client. The load balancing system will continue to run until the request from the client has stopped.

- Algorithm Round Robin Configuration

Previously discussed system design. Where in the design of the system there are steps in device configuration. Device configuration is the first step in analyzing the load balancing algorithm. Load balancing algorithm settings are performed on the server controller Load balancing. Configuration is done in the default file in the etc/nginx/sites-available directory. The modification and combination are also inspired by [23],[24], and [25].

- Apache JMeter Configuration

JMeter performance testing includes load tests and web application stress tests. Adding Thread Group: This node is used for group services to be tested; the number of threads is

200, the ramp-up period is 1, and the loop count is 1. For example, having Student and Lecturer services. HTTP Header Manager, this node is used to add name and value, with the name column filled with Content-Type and Value application-Json. Next add the HTTP Request node. At this node, the researcher will determine the Web service that will be tested. For example, the Web service has an Auth JWT service, which is to configure authentication to the database using JSON Extractor and Person to accept authentication from Auth JWT using JSR223 Preprocessor. This service will be added as part of the HTTP Request.

- Apache JMeter Design Testing

This time, Apache JMeter is being used for the test. Open-source software made entirely of Java, the Apache JMeter Application is used to load functional behavior tests and track performance. Figure 2 below depicts the Apache JMeter performance flow.

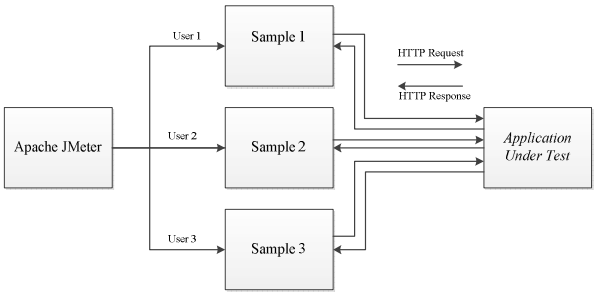

Figure 2 Flowchart Apache JMeter Design

Tests are also carried out to determine the average value for each given parameter. Granting requests is used to load the server at one time. The results of each experiment will be compared and analyzed. So that it will get a performance comparison between the Round Robin algorithm and not using the Round Robin algorithm, and using Indexes or not using Indexes which can be concluded at the end of the study. Tests on the Response time parameters above were carried out 6 times, having 5 categories, namely:

a. 40 requests per 30 seconds with several 40 connections.
b. 60 requests per 30 seconds with several 60 connections.
c. 80 requests per 30 seconds with several 80 connections.
d. 90 requests per 30 seconds with several 90 connections.
e. 100 requests per 30 seconds with several 100 connections.

III. Results and Discussion

The test data will be collected from the summary report results which are exported to Microsoft Excel on the JMeter tool to get the performance results of the Round Robin method. This test is conducted to determine the performance of the Web server, the parameter used is response time. Response time is the time determining the average result of an HTTP request.

A. Testing of Indexes without Round Robin

- Example 40user/30s

The results of testing with indexes without the Round Robin method with a sample of 40 users/30s and running 10 trials yield that the average response time in an experiment is 1177 milliseconds, the minimum response time is 640 milliseconds, and the maximum response time is 5304 milliseconds. Accordingly, the error rate is still 0%.

- Example 60user/30s

With a sample size of 60 users/30s, 10 trials, and testing with indexes without the Round Robin approach, the findings show that the average response time in an experiment is 1839 milliseconds, the shortest response time is 662 milliseconds, and the longest response time is 4680 milliseconds. The mistake rate is therefore still 0%.

- Example 80 users/30s

An experiment's average response time is 5929 milliseconds, the smallest response time is 745 milliseconds, and the longest response time is 10932 milliseconds, according to the results of testing using indexes without using the Round Robin approach on a sample size of 80 users/30s over the course of 10 trials. Therefore, the error rate remains at 0%.

- Example 90 users/30s

After conducting 10 trials with a sample size of 90 users/30s, it was observed that when indexes were not used in conjunction with the Round Robin approach, the average response time for the experiment was found to be 11844 milliseconds. The shortest recorded response time was 187 milliseconds, while the longest response time reached up to 22020 milliseconds. As a result, the rate of error stands at 11.11%.

- Example 100 users/30s

The experiment's average response time was 15627 milliseconds, the shortest response time was 216 milliseconds, and the longest response time was 36133 milliseconds, according to the results of testing using indexes without the Round Robin approach on a sample size of 100 users/30s over the course of 10 trials. As a result, the error rate is 9.61 percent.

Of the five index tests without round robin with five categories, namely 40 users, 60 users, 80 users, 90 users, and 100 users, the average of 40 users, as seen from the graph above, is very stable when doing 10 consecutive trials; the average of 60 users, as seen from the graph above, is quite stable; the average of 80 users looks quite stable; and for 90 users and 100 users, the average response time graph is close to and not much different; the graph shows very stable.

The value of all experiments with indexes without round robin has a high value but has a stable graph on average because it adds the indexing method. The minimum response time and maximum response time are also quite low, stable, and not far from the average of each parameter. Indexing itself is a data structure object that does not depend on the database table structure. Each index consists of a column value and a pointer (or ROWID) to the row containing that value. The pointer directly points to the correct row in the table. In this study, the table "name" was chosen as the ROWID, thus avoiding the occurrence of a full table scan.

B. Testing of Indexes with Round Robin

- Example 40 users/30s

Excitingly, when employing the Round Robin method with a group of 40 users/30s, we obtained the subsequent outcomes: the mean response time during an experiment measures 681 milliseconds, while the shortest and longest response times account for 640 and 799 milliseconds respectively. Excitingly, there has been no increase in the error rate whatsoever.

- Example 60 users/30s

The Round Robin method was used with a group of 40 users/30s, and excitingly, we were able to achieve the following results: the mean response time during an experiment is 1036 milliseconds, while the shortest and longest response times are 774 and 1541 milliseconds, respectively. Interestingly, there hasn't even been a slight rise in the error rate.

- Example 80 users/30s

We were able to get the following results using the Round Robin approach with a group of 40 users/30s: the mean response time for an experiment is 5459 milliseconds, while the shortest and longest response times are 811 and 12232 milliseconds, respectively. Strangely, the error rate hasn't even slightly increased.

- Example 90 users/30s

The Circular Robin strategy was utilized with a bunch of 40 users/30s, and excitingly, we were able to realize the taking after comes about: the cruel reaction time amid a try is 11951 milliseconds, whereas the most limited and longest reaction times are 218 and 29797 milliseconds, individually. In interests, there is a rate of error at 5.11%.

- Example 100 users/30s

When indexes were employed in conjunction with the Round Robin technique, the experiment's average reaction time was discovered to be 13922 milliseconds after 10 trials with a sample size of 90 users/30s. The quickest response time ever was 178 milliseconds, whereas the slowest was as long as 25977 milliseconds. The rate of error is consequently 6.02%.

The scenario testing is done in a Round Robin format with 5 groups of users: 40, 60, 80, 90, and 100. The average of 40 users as determined by the studies is quite trustworthy when performing 10 consecutive experiments. From 60 users is rather stable, the average of 80 users shows a very stable, and for 90 users and 100 users looks incredibly stable compared to the previous test, the error value of 90 users and 100 users is comparatively small. There is one thing that we must notice: the error rate only appears when the user is equal to or more than 90.

Indicates that the request and response have a high success rate. Because the data distribution request employs load balancing round robin, which lowers the value, and this test also uses indexing, the value of all trials with indexes with round robin is marginally lower than the index test without round robin. The minimum and maximum response times are likewise reasonably low, consistent, and close to the mean of each parameter. The data structure object for indexing is independent of the database table structure. A column value and a pointer (or row ID) to the row holding that value are both included in each index. The appropriate row in the table is directly indicated by the pointer. To avoid a full table search, the "name" of the table was used as the ROW ID in this investigation.

IV. CONCLUSION

From the research that has been done, several conclusions were obtained as follows :

1. Overall, the response speed of each request depends on the specifications of the server, the network being passed. These two factors are interdependent and cannot be separated. If the client request has a fast network, but the server being accessed has low specifications, the response made by the server to the client is less responsive.
2. The proposed combination between the round-robin and indexes that have successfully reduced the time response is more stable than a single server and does not add indexes that have higher values and irregular statistics.
3. The number of rows of data needs to be improved since it will lead to more valid data since it could be compared with the other load balance methods.

V. REFERENCES

[1] F. Ardianto, B. Alfaresi, and A. Darmadi, "LOAD BALANCING DESIGN OF TWO MICROTICS-BASED INTERNET SERVICE PROVIDER (ISP)," *JURNAL SURYA ENERGY*, vol. 3, no. 1, p. 198, Aug. 2018, doi: 10.32502/jse.v3i1.1232.

[2] R. Dani and F. Suryawan, "Design and Testing of Load Balancing and Failover Using NginX," *Khazanah Informatika : Jurnal Ilmu Komputer dan Informatika*, vol. 3, no. 1, pp. 43–50, Jun. 2017, doi: 10.23917/khif.v3i1.2939.

[3] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler, "Waiting times in quality of experience for web based services," in *2012 Fourth International Workshop on Quality of Multimedia Experience*, IEEE, Jul. 2012, pp. 86–96. doi: 10.1109/QoMEX.2012.6263888.

[4] D. P. Olshefski, J. Nieh, and D. Agrawal, "Inferring client response time at the web server," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 160–171, Jun. 2002, doi: 10.1145/511399.511355.

[5] C. A. Wicaksana, M. Fatkhurrokhman, H. P. Pratama, R. Tryawan, Alimuddin, and R. Febriani, "Twitter Sentiment Analysis in Indonesian Language using Naive Bayes Classification Method," in *2022 International Conference on Informatics Electrical and Electronics (ICIEE)*, Yogyakarta: IEEE, Oct. 2022, pp. 1–6. doi: 10.1109/ICIEE55596.2022.10010002.

[6] C. A. Wicaksana and W. Martiningsih, "Deformation of 3D Object of Human Body Internal Organs Using Finite Element Method Approach Accelerated by GPU," in *Proceedings of the International Joint Conference on Science and Engineering (IJCSE 2020)*, Paris, France: Atlantis Press, 2020. doi: 10.2991/aer.k.201124.003.

[7] F. Apriliansyah, I. Fitri, and A. Iskandar, "Implementation of Load Balancing on Web Servers Using Nginx," *Journal of Information Technology and Management*, vol. 6, no. 1, Apr. 2020, doi: 10.26905/jtmi.v6i1.3792.

[8] Dwi Budi Santoso, Jeffri Alfa Razaq, and Fatkhul Amin Amin, "Monitoring Unisbank E-Learning Server Load Balancing Using the Least Method," *Journal of Electronics and Computers*, vol. 16, no. 1, pp. 11–19, Jul. 2023, doi: doi.org/10.51903/elkom.v16i1.932.

[9] T. Wira Harjanti, H. Setiyani, and J. Trianto, "Load Balancing Analysis Using Round-Robin and Least-Connection Algorithms for Server Service Response Time," *Applied Technology and Computing Science Journal*, vol. 5, no. 2, pp. 40–49, Dec. 2022, doi: 10.33086/atcsj.v5i2.3743.

[10] X. Wang, Q. Sun, and J. Liang, "JSON-LD Based Web API Semantic Annotation Considering Distributed Knowledge," in *IEEE Access*, 2020, pp. 197203–197221. doi: 10.1109/ACCESS.2020.3034937.

[11] Z. Cui *et al.*, "Closer: Scalable load balancing mechanism for cloud datacenters," *China Communications*, vol. 18, no. 4, pp. 198–212, Apr. 2021, doi: 10.23919/JCC.2021.04.015.

[12] K. S. Chaudhury, S. Pattnaik, H. S. Moharana, and S. Pradhan, "Static Load Balancing Algorithms in Cloud Computing: Challenges and Solutions," 2020, pp. 259–265. doi: 10.1007/978-981-15-2475-2_24.

[13] A. Abdulrahim, S. E. Abdullahi, and J. B. Sahalu, "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm," *Int J Comput Appl*, vol. 90, no. 4, pp. 27–33, Mar. 2014, doi: 10.5120/15563-4277.

[14] Arif Maulana Komaruddin, Della Maerlin Sipitorini, and Pian Rispian, "Load Balancing with the Round Robin Method for Sharing Web Server Workloads," *Jurnal Siliwangi Sains Teknologi*, vol. 5, no. 2, pp. 47–50, 2019, doi: 10.37058/jssainstek.v5i2.1184.

[15] J. Wang and Z. Kai, "Performance Analysis and Optimization of Nginx-based Web Server," in *Journal of Physics: Conference Series*, Jun. 2021, p. 012033. doi: 10.1088/1742-6596/1955/1/012033.

[16] Y. Arta, "Application of the Round Robin Method in Multihoming Networks in Computer Clusters," *IT JOURNAL RESEARCH AND DEVELOPMENT*, vol. 1, no. 2, pp. 26–35, Aug. 2017, doi: 10.25299/itjrd.2017.vol1(2).677.

[17] H. Nasser and T. Witono, "ANALYSIS OF ROUND ROBIN, LEAST CONNECTION, AND RATIO ALGORITHMS IN LOAD BALANCING USING OPNET MODELER," *Jurnal Informatika*, vol. 12, no. 1, Jun. 2016, doi: 10.21460/inf.2016.121.455.

[18] R. Čerešňák and M. Kvet, "Comparison of query performance in relational a non-relation databases," in *Transportation Research Procedia*, High Tatras, May 2019, pp. 170–177. doi: 10.1016/j.trpro.2019.07.027.

[19] R. S. Kraleva, V. S. Kralev, N. Sinyagina, P. Koprinkova-Hristova, and N. Bocheva, "Design and Analysis of a Relational Database for Behavioral Experiments Data Processing," *International Journal of Online Engineering (iJOE)*, vol. 14, no. 02, p. 117, Feb. 2018, doi: 10.3991/ijoe.v14i02.7988.

[20] Fathansyah, *Database*, Third Edition. Bandung: Informatika Bandung, 2018. Accessed: Aug. 27, 2023. [Online]. Available: www.biobses.com

[21] P.-L. Suei, Y.-F. Lu, R.-J. Liao, and S.-W. Lo, "A signature-based Grid index design for main-memory RFID database applications," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1205–1212, May 2012, doi: 10.1016/j.jss.2012.01.026.

[22] M. Hayati and A. Setyanto, "Index Effect on Data Manipulation Toward Database Performance," *J Phys Conf Ser*, vol. 1140, p. 012036, Dec. 2018, doi: 10.1088/1742-6596/1140/1/012036.

[23] H. GIBET TANI and C. EL AMRANI, "Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data," *Journal of Data Mining & Digital Humanities*, Jan. 2018, doi: 10.46298/jdmdh.3104.

[24] A. K. Gupta, P. Mathur, C. M. Travieso-Gonzalez, M. Garg, and D. Goyal, "ORR: Optimized Round Robin CPU Scheduling Algorithm," in *Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence*, New York, NY, USA: ACM, Aug. 2021, pp. 296–304. doi: 10.1145/3484824.3484917.

[25] A. Fiad, Z. M. Maaza, and H. Bendoukha, "Improved Version of Round Robin Scheduling Algorithm Based on Analytic Model," *International Journal of Networked and Distributed Computing*, vol. 8, no. 4, p. 195, 2020, doi: 10.2991/ijndc.k.200804.001.