Applied Information System and Management (AISM)
Volume 7, (1) 2024, p. 29-36
P-ISSN: 2621-2536; E-ISSN: 2621-2544; DOI: 10.15408/aism.v7i1.34871

Improving Moodle Performance Using HAProxy and MariaDB Galera Cluster

Johan Ericka Wahyu Prakasa[1*], Ajib Hanani[2], Fajar Rohman Hariri[3], Shoffin Nahwa Utama[4]

Abstract—**Moodle is a widely used Learning Management System in various educational institutions worldwide. However, frequent reports on internet forums indicate performance degradation when massive simultaneous users access Moodle. One of the most resource-intensive components supporting Moodle is the database, as all user-accessed data is stored in it. This study aims to optimize Moodle's performance through distributed databases. Distributing the database into multiple database servers allows the database load to be distributed across all the database servers, resulting in an overall improvement in Moodle performance. This study compares the performance of Moodle installed on a single server with that installed on multiple database servers. Various testing parameters are employed to get valid results. Namely, course read, course write, and database performance, utilizing the server performance plugin available in Moodle. This research reveals a performance improvement of 384% in course read, 193% in course write, and 260% in the Moodle database in the multi-server scenario compared to the single-server scenario. This result validates that the database is the most crucial part of Moodle.**

***Index Terms*— Distributed databases, HAProxy, MariaDB galera cluster, moodle, optimization, performance.**

I. Introduction

With the shift toward online learning in response to the COVID-19 epidemic, the Learning Management System (LMS) has experienced substantial growth in use over the past few years [1], [2]. A LMS is a sophisticated software application meticulously designed to streamline and enhance the distribution, administration, and monitoring of educational resources and information. This comprehensive tool empowers educators by allowing them to create and efficiently manage online courses, engage in interactive sessions with their students, and closely monitor the progress and performance of each student, all from the convenience of a centralized location. The LMS acts as a dynamic hub that combines various educational elements, including course materials, assessments, and communication tools, offering a holistic platform for educators to deliver impactful learning experiences.

The significance of LMS became even more apparent during the unprecedented challenges posed by the COVID-19 pandemic. LMS emerged as a pivotal technology that played a critical role in ensuring the uninterrupted continuity of learning. Its adaptive features allowed educational institutions to transition from traditional in-person teaching to effective online delivery swiftly. In this context, the LMS served as a technological solution and became a cornerstone in fostering remote learning environments, enabling educators and students to navigate the complexities of virtual education seamlessly.

The LMS landscape presents various platforms, each characterized by different licensing models, from free and open source to free-to-use and freemium structures, where advanced features entail payment. Learning Management Systems with free and open-source licenses necessitate users to provide their server infrastructure and computer networks, commonly called on-premise infrastructure. Installation and configuration are undertaken independently, although there is support from the LMS user community. Under this concept, user institutions take full responsibility for the whole operational aspects of the LMS, from IT infrastructures to LMS management. This approach grants institutions greater autonomy and control over the LMS, allowing them to tailor the system to their needs. However, it also requires technical expertise and resources to effectively manage and maintain the on-premise infrastructure. Moodle, Canvas, and OpenEDX are LMS platforms operating under open-source licenses, affording users the freedom to utilize them without cost and adapt them further to suit the specific requirements of their institutions.

Users who cannot provide the necessary supporting infrastructure for a learning management system (LMS), such as servers and adequate computer networks, can make use of one of the many free learning management system (LMS) providers, such as Google Classroom, Edmodo, or Schoology [3]. These LMS systems run under a free license, enabling users to access them without being charged a fee; however, users are often prevented from modifying the main LMS functionality. Users can leverage the LMS features without being required to

Received: 19 September 2023; Revised: 2 November 2023; Accepted: 27 November 2023

*Corresponding author
[1]J. E. W. Prakasa, Teknik Informatika Department, UIN Maulana Malik Ibrahim Malang Indonesia (e-mail: johan@uin-malang.ac.id).
[2]A. Hanani, Teknik Informatika Department, UIN Maulana Malik Ibrahim Malang Indonesia (e-mail: ajib@uin-malang.ac.id).
[3]F. R. Hariri, Teknik Informatika Department, UIN Maulana Malik Ibrahim Malang Indonesia (e-mail: fajar@ti.uin-malang.ac.id).
[4]S. N. Utama, Teknik Informatika Department, UIN Maulana Malik Ibrahim Malang Indonesia (e-mail: shoffin@uin-malang.ac.id).

handle the underlying technical infrastructure when the service model known as Software as a Service (SaaS) is utilized.

For more advanced users, there are several LMS platforms with freemium licenses. LMS platforms with this license type offer basic LMS functionalities for free, albeit with certain limitations, and provide specific features under a paid model. By utilizing this licensing model, users are relieved of the need to manage LMS-supporting infrastructure and employ expert staff for LMS configuration. Users can immediately leverage the free basic LMS features and make payments only when they require advanced features. In contrast, Blackboard and Classe365 are LMS platforms that operate under a freemium license. The variety of LMS licenses provides educators with numerous options to optimize their teaching processes without requiring intricate technical skills to manage the LMS.

Moodle is one of the LMS platforms utilized by educational institutions worldwide [4]. This is because it is licensed under a free and open-source model, enabling further development. Moodle also has a robust community assistance system, making it more straightforward for new users to learn their way around the platform. Moodle is one of the most adaptable LMS platforms because it has many community-created plugins. These plugins make Moodle one of the few LMSs that can accommodate nearly unlimited development options. Moodle provides tools for tracking student progress, providing feedback, and integrating multimedia resources such as videos, audio, and interactive content [5].

According to findings from the earlier study, the database's performance considerably impacts Moodle's overall performance [6]. The Moodle database is where the settings for the system, as well as the user information and course materials, are stored. Consequently, the Moodle platform's performance and speed might be substantially hindered by any lags in the database or other problems that arise with it. The performance of the database is susceptible to being influenced by some factors, such as the size and complexity of the Moodle site, the number of users who access the site, and the amount and complexity of the queries made on the database. When Moodle is visited simultaneously by many users, it will send many data queries to the database. The processing capability of the database server determines where these queries will be placed in the queue to be processed. The amount of time needed to complete each query differs based on the data the user is looking for. When a query takes a considerable amount of time to execute, the total time the other inquiries in the queue must wait increases. If the amount of time spent in the queue exceeds the maximum amount of time the server is allowed to execute queries, the query will be terminated. If there are numerous aborted queries, it will lead to a situation where no data is presented on the user's screen (the white screen of death), substantially decreasing the quality of the user experience provided by Moodle.

In the Moodle online forums, people frequently complain that the performance of Moodle suffers when it is simultaneously accessible by many users. The performance of the database is essential to overall Moodle performance since Moodle stores activity generated by users within the database. When database optimization is accomplished but the performance issue continues, upgrading the Moodle server is the only method to overcome performance degradation and restore optimal functionality. There are two different models for scaling or upgrading a server: horizontal and vertical scaling.

The process of increasing the capacity of a server by adding more resources to the server is known as vertical scaling. Vertical scaling, often called "scaling up," involves increasing the power of a single server. This can include adding more processing power, memory, or storage to accommodate growing demands. While vertical scaling is a straightforward approach, it has limitations. There's only so much you can upgrade a single server before encountering hardware restrictions.

Additionally, if the server fails, it can result in significant downtime, making it a less resilient option. On the other hand, horizontal scaling, or "scaling out," distributes the load across multiple servers [7]. When compared to vertical scaling, horizontal scaling offers a more significant number of advantages, some of which include load balance and failover protection. In addition, automatic server cloning and load balancing configuration are available by default with cloud computing services.

Load balancing is a critical aspect of horizontal scaling. It involves the efficient distribution of incoming network traffic or workload across multiple servers. This prevents any server from becoming overwhelmed and ensures that resources are utilized optimally. Various algorithms, such as Round Robin, Least Connection, or IP Hash, are used in load balancing to determine how to distribute incoming requests. Failover protection is another significant advantage of horizontal scaling. In a multi-server environment, the others can still handle the load if one server fails. This enhances the system's overall reliability and minimizes downtime.

By utilizing these characteristics, the server can automatically clone itself and configure load balance to distribute traffic when required. Moodle's learning management system (LMS) can retain its performance even when subjected to tremendous demand because of its automatic scaling features [8].

In addition to enhancing server performance, horizontal scaling ensures high server availability, meaning an application will continue functioning on other servers if one server goes down [9]. Utilizing a load balancer server is a must to distribute query requests across multiple database servers. A load balancer server distributes the workload evenly among multiple servers. Numerous server load-balancing software solutions are widely utilized, including NginX, HAProxy, and Zevenet [10], [11]. Various dynamic techniques are available for distributing server loads to distribute the workload effectively. These techniques are Round Robin, Least Connection, IP Hash, Generic Hash, Least Time, and random methods. The round-robin algorithm is designed to evenly distribute the server workload among all cluster members evenly, ensuring an equal share of the workload for each server. However, this algorithm may be less than optimal when the servers have varying specifications, as it can lead to some servers becoming overloaded due to the cyclic load distribution. In the Least-connection Algorithm, server selection is determined by the server with the fewest active connections [12]. The rationale behind this approach is to maintain optimal server performance for serving new users. However, this algorithm is limited, as it solely considers the number of user connections to the server and does not consider the server's workload. It is conceivable that even with a small number of users, a server may be engaged in resource-intensive tasks, such as simultaneously generating randomized quizzes, which can impact its performance.

The Moodle database maintains all of the data, such as course materials, data on assignments along with grades, data on exams along with grades, and so on. As a result, the database that is included with Moodle is a crucial component. Every action that Moodle users take is connected in some way to the database since that is where all of the data is stored. This is true whether they are retrieving data from the database or adding data to it. These user operations are executed using a language specific to databases and known as a query. When compared to information systems that are not LMSs, Moodle's queries are more complex since they require data from many different tables. As a result, they take a more extended amount of time to process [13].

In practical applications, using Moodle to facilitate the learning process often leads to numerous users accessing the system concurrently. This influx of users generates a queue of queries awaiting processing in the database. The processing time duration depends on the complexity of the data requested by users; more complex queries will need extended processing periods. As the queue of processes is unstoppable, there's a critical need to prevent system overload. The database server has an automatic termination mechanism to address system overload, designed explicitly for queries characterized by extended processing durations or prolonged wait times in the queue.

When seen from the perspective of the end-user, this operating method has a significant impact on the experience they have. The user interface might not work as well as it should, the responsiveness of the Moodle system might deteriorate, and there might be more errors in the system. For instance, when an instructor requests information about student grades or a student submits an assignment, Moodle executes queries to retrieve or modify the corresponding data in the database. However, when the queries involve multiple tables, it can lead to delays during periods of high user engagement. Because of the large number of concurrent user requests, the system might have trouble executing queries in a timely manner, which could delay the process of either retrieving or storing data. This delay in responding can significantly damage the overall user experience, causing dissatisfaction and preventing smooth learning.

Enhancing the performance of Moodle goes beyond optimizing the web server, particularly considering the critical role of the database, which stands as one of the most frequently accessed components. Recognizing this, some databases come equipped with advanced features like replication or multi-server capabilities to address the demands of high-concurrency environments. The MySQL database, for instance, provides a robust replication feature. This functionality allows for installing the MySQL database across multiple servers, offering various configuration options such as master-master or master-slave setups. In a master-master configuration, each server can function as both a master and a slave, facilitating bidirectional data replication. On the other hand, the master-slave configuration designates one server as the master, handling write operations, while others act as slaves, replicating data from the master. These configurations contribute to improved performance by distributing the database workload, ensuring redundancy, and enhancing fault tolerance [14].

Galera Cluster is a feature of MariaDB that allows users to establish a distributed database across multiple machines. Data will be synced between database servers using the Galera Cluster function in MariaDB, similar to MySQL's master-master feature. By utilizing this method, a highly available server environment will be produced. This indicates that the given services will be accessible at all times, even if there are issues with the server [15]. The previous database technology used Master-slave database techniques to handle large database transactions. This technique makes database replication to multiple servers where the slave database will get updates from the master database but not vice versa [16]. The program is set to read data from the slave database while writing/updating data to the master database to reduce database load. However, Galera Cluster can make database replication with master-master status. With this technique, data can be written to any database server; then the database server will synchronize. Thus, a high-availability environment will be created where, when a database server experiences problems, the system will continue running using another server [17].

II. Related Work

While research on utilizing Moodle as a Learning Management System is standard, relatively limited research is available on optimizing Moodle's performance when accessed by many concurrent users.

Efforts to reduce the load on Moodle when accessed by many concurrent users have previously been made through rule-based approaches [18]. This study proposes implementing several rules to mitigate the workload on Moodle. These rules include introducing time intervals between the start of lectures for different faculties, avoiding the simultaneous use of examination features by numerous users, and implementing other regulations to prevent concurrent access to Moodle during the same time. Furthermore, this research recommends several strategies to alleviate the burden on Moodle, such as implementing flexible exam scheduling to allow students to take exams asynchronously, pre-generating randomized questions to reduce Moodle's need for real-time question randomization, transitioning to asynchronous learning modes, and conducting classroom management activities like class backups, enrollment of students, and various management tasks during semester breaks when there is no Moodle-based learning activity. The implementation of these recommendations has resulted in a reduction in the Moodle workload of up to 20%. Removing the old Moodle system after upgrading to a newer version has led to approximately 10% to 15% storage space savings. It is also essential to monitor the plugins used in the learning process, allowing for the removal of unused plugins.

The load distribution technique among multiple servers to enhance Moodle performance has been previously implemented [19]. In this research, Moodle utilizes three separate servers: the web server, the application server, and the database server. Performance measurement is conducted by comparing the server performance across various scenarios. The scenarios used include (1s, 1s, 1s), (1s, 1s, 2s), (1s, 1s, 1h), and (2s, 2s, 2s). Each parameter represents the specifications for the web server, application server, and database server. "s" signifies the standard server specifications (2-core vCPU, 2GB RAM, 20GB storage), while "h" represents the high server specifications (4-core vCPU, 4GB RAM, 40GB storage). The testing used the Apache Bench and Sysbench tools to measure server performance in each scenario. This research indicates that the best performance was achieved in the scenario (1s, 1s, 1h), where the database server had higher specifications than the web server and application server. This research shows that the database server is the component that handles the majority of the workload in the Moodle system. Enhancing the hardware specifications of the database server has a positive impact on the overall Moodle performance.

Efforts to enhance Moodle's performance using cloud computing technology have also been undertaken previously. This research involved migrating Moodle from the university's local server to the Microsoft Azure cloud computing platform [20]. The study utilized a Virtual Machine Scale Set (VMSS) with automatic scaling features. The specifications for the VMSS included two vCPU cores, 7GB RAM, and 128GB SSD storage with the Ubuntu 18.04 LTS operating system. The researcher set up a load balancer to distribute traffic evenly across all servers. For the database server, AzureDB for MySQL was used with server specifications of 4 CPU cores and 20GB of RAM. All servers were placed in the same West Europe Azure Region to minimize data transfer between distant servers, which could result in high latency. The testing was conducted using a quiz feature with 20 multiple-choice questions, displayed with five questions per page. The quiz duration was set to 20 minutes and was simultaneously taken by 300 users. This research indicates that cloud computing technology can significantly improve Moodle's performance compared to on-premise server hosting. No issues were encountered in the cloud computing testing, which contrasts with the on-premise server testing, where problems were consistently experienced. From the server specifications used in cloud computing, it is evident that the database server had higher specifications than the web server. This design choice is made to efficiently handle requests from the web server, which can be replicated based on the number of users accessing the system.

Quality of Service is one of the most crucial factors in information systems. There is research on Moodle's Quality of Service area from the network point of view [21]. This research focuses on improving Moodle's Quality of Service from a network perspective using Software Defined Network (SDN). SDN is a relatively new technology in networks used by cloud computing. Using SDN, networks on cloud computing can be managed virtually using a programming language. This research tries to prioritize network traffic for Moodle over another type of traffic in the network. Network prioritization is done by creating a special plugin called SDN4Moodle. This plugin will communicate with SDN to prioritize traffic for Moodle based on Moodle's response to user activity. The result shows that SDN4Moodle can prioritize network traffic for Moodle and increase Moodle's Quality of Service.

Moodle was essential in academic activities as a Learning Management System, especially during the COVID-19 pandemic. That is why Moodle needs high availability and performance [22]. This research focuses on making Moodle accessible most of the time by using clustering technology. This research uses redundant servers for the web server, database server, and file server under the coordination of a load balancer. The load balancer will distribute traffic to the web, database, and file servers. Since the hardware specification of the servers is identical, the load balancer uses a round-robin algorithm to distribute traffic evenly to all the servers. To keep data synchronized among database servers, GaleraDB was employed. File replications are done by rsync To keep files synchronized among file servers. This setup will distribute the load to servers, resulting in a performance boost since server load is distributed to many servers. This setup also provide High Availability to the Moodle server in case some server is down due to technical issue; user request will served by another server in the pool. Users will not suffer system errors even if there is any technical issue in server infrastructure, and increase user experience.

Effort to improve Moodle's performance has been elaborated using various technique. Using many servers to distribute user loads is a proven method, but this will increase IT budgeting, mainly to upgrade the server's infrastructure. It's known from previous research that Moodle's performance relies on the database the most. So, this research will focus on improving Moodle's performance by distributing the database server alone rather than all the servers needed by Moodle (including web servers and file servers). This technique will prevent an explosion of budgeting on IT infrastructure but still can increase Moodle's performance.

This research aims to improve the performance of Moodle in an on-premises environment by distributing the database workload across multiple servers. Database load is distributed using HAProxy, a load balancer that can distribute the workload across several servers. HAProxy is a popular open-source load balancer that distributes incoming traffic across multiple servers to improve application performance, scalability, and reliability. There are several load-balancing methods that HAProxy uses to distribute traffic, and each has its advantages and use cases [23]. This research uses the least connection method as the HAProxy load balance method.

This research utilizes the Galera Cluster feature in MariaDB to maintain data consistency among database servers. Galera Cluster is a database management system ensuring that all servers' data remains consistent [24]. Galera Cluster supports

Applied Information System and Management (AISM)
Volume 7, (1) 2024, p. 29-36
P-ISSN: 2621-2536; E-ISSN: 2621-2544; DOI: 10.15408/aism.v7i1.34871

multi-master replication, meaning all the database servers can serve all users' requestsInsert, Update & Delete) [25]. Galera Cluster will automatically synchronize data updates among database servers.

By using HAProxy and Galera Cluster in MariaDB, this research is expected to improve Moodle's performance by speeding up access to the database and minimizing the time required to send data between servers. Additionally, by distributing the database workload across multiple servers, this research can also improve the scalability of Moodle, allowing the platform to handle a more significant number of users.

III. Research Method

This study employs a comparative research design to evaluate the performance of the Moodle learning management system in two distinct server configurations: a single server environment and a multiple servers (cluster) environment. Data collection involves the deployment of multiple Moodle instances, each configured to simulate real-world usage scenarios. Database-related performance metrics are systematically measured, including reading course performance, writing course performance, and database performance. Simulated usage scenarios representing various levels of user activity are created to assess the impact of server configuration on Moodle's performance.

Data is collected using Moodle's performance measurement plugins. Writing course performance is measured to understand Moodle's performance upon creating new courses and their content. Reading course performance is measured to understand Moodle's performance upon presenting the course and its content to the user. Database performance is measured to understand the overall database performance used by Moodle in the response to user requests.

In the single-server scenario, Moodle is installed on a server that supports web, database, and file servers, with all the supporting services installed on the same server, enabling server resource sharing between the services. Figure 1 provides a visual representation of the topology of the single-server scenario.

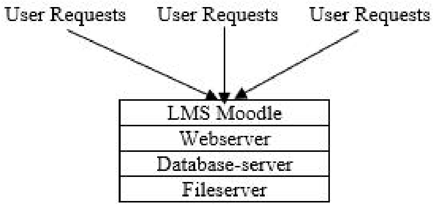

Fig. 1. Moodle LMS on a single server.

On the other hand, the multi-server scenario involves installing Moodle LMS on one server providing web server and file server services. In contrast, the database server is installed on a separate server using MariaDB's Galera Cluster feature. A load balancer is installed on the webserver to evenly distribute the load of accessing the database server. This approach ensures that all database servers receive an equal workload, preventing the overloading of any single server. Figure 2 provides a visual representation of the topology of this scenario.

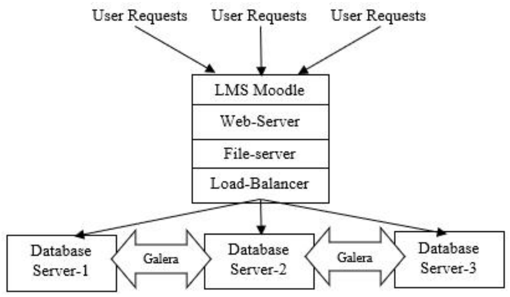

Fig. 2. Moodle LMS on multi-database server.

IV. Result

We conducted 300 tests for each scenario to explore different server loads, yielding average outcomes summarized in Tables 1 and 2. Our examination focused on the performance of reading courses, writing courses, and the database, employing the Moodle LMS server performance function. The testing criteria comprehensively addressed these three dimensions. The values presented in the tables depict the duration of each process in seconds. This meticulous testing approach allows us to scrutinize and compare the efficiency of the Moodle system under various conditions, providing valuable insights into its responsiveness and robustness in handling distinct workloads. The outcomes in the tables encapsulate the quantitative representation of these evaluations, offering a comprehensive view of the system's temporal dynamics during diverse operations.

Table 1.
Performance Test Results for The Single-Server Scenario

Exp.	Read	Write	DB	Exp.	Read	Write	DB
1	0.415	0.162	0.212	16	0.414	0.147	0.147
2	0.411	0.165	0.236	17	0.412	0.152	0.212
3	0.418	0.136	0.202	18	0.438	0.150	0.213
4	0.452	0.172	0.212	19	0.424	0.222	0.276
5	0.426	0.137	0.211	20	0.411	0.411	0.219
6	0.407	0.149	0.217	21	0.416	0.140	0.204
Exp.	Read	Write	DB	Exp.	Read	Write	DB
7	0.425	0.160	0.425	22	0.403	0.125	0.212
8	0.413	0.141	0.212	23	0.501	0.181	0.229
9	0.403	0.152	0.213	24	0.431	0.431	0.215
10	0.469	0.157	0.225	25	0.400	0.159	0.208
11	0.420	0.157	0.212	26	0.407	0.140	0.208
12	0.425	0.154	0.221	27	0.421	0.136	0.212
13	0.532	0.144	0.228	28	0.404	0.404	0.210
14	0.416	0.130	0.212	29	0.426	0.153	0.211
15	0.428	0.134	0.220	30	0.425	0.150	0.211

From Table 1, it can be observed that the average time to display the requested course content is approximately 0.4 seconds. Although this appears relatively fast in terms of time, a processing queue can form when numerous users request the display of course content simultaneously (for instance, during a class schedule). If this queue becomes excessively long, it can result in the occurrence of the "white screen of death." The "write" parameter in the table represents the time it takes for Moodle to save data when there is input from users. For example, when users submit assignments or complete quizzes, the results must be saved. The duration of the writing process varies between 0.1 to 0.4 seconds, as shown in Table 1. This variation is influenced by various factors, such as the length of the processing queue that Moodle needs to handle and the simultaneous data writing by multiple users, especially during activities like quiz submissions that conclude simultaneously. These factors contribute to queuing of written commands, which, in turn, results in longer writing times. Meanwhile, the "database" parameter in Table 1 represents the time needed by the database to handle user requests, both for reading (read) and writing (write) data. In a single-server scenario with only one database server, all activities are serviced by a single database, potentially leading to a processing queue with an average processing time of 0.2 seconds.

In the multi-server scenario, there was an improvement in performance across all parameters (course read, course write, and database) compared to the single-server scenario. The average test results for the multi-server scenario are presented in Table 2. For the "course read" parameter, the average testing result yielded a value of 0.1 seconds. Similarly, the average value obtained for the "course write" parameter was 0.09 seconds, and for the "database" parameter, the average value was also 0.09 seconds.

Table 2.
Performance Test Results for The Multiple-Server Scenario

Exp.	Read	Write	DB	Exp.	Read	Write	DB
1	0.119	0.071	0.105	16	0.109	0.073	0.097
2	0.108	0.086	0.095	17	0.110	0.068	0.101
3	0.108	0.081	0.097	18	0.108	0.087	0.104
4	0.110	0.073	0.096	19	0.108	0.081	0.097
5	0.119	0.074	0.098	20	0.110	0.075	0.098
6	0.108	0.082	0.097	21	0.106	0.086	0.097
7	0.109	0.072	0.096	22	0.109	0.088	0.100
8	0.109	0.073	0.098	23	0.109	0.090	0.097
9	0.115	0.082	0.096	24	0.116	0.075	0.098
10	0.107	0.072	0.096	25	0.117	0.071	0.105
11	0.111	0.075	0.097	26	0.117	0.071	0.105
12	0.111	0.080	0.099	27	0.111	0.076	0.097
13	0.108	0.081	0.098	28	0.109	0.086	0.096
14	0.111	0.080	0.104	29	0.115	0.081	0.113
15	0.110	0.082	0.102	30	0.115	0.081	0.113

The results indicated that using multiple servers for databases can improve the performance of Moodle LMS compared to a single-server configuration. When you average the values from Table 1 and Table 2 and compare each parameter, the results can be visualized in the graph below. Figure 3 illustrates a graph of the average times generated by the "course read" parameter. On average, the "course read" time in the single-server scenario is 0.42 seconds, while in the multi-server scenario, it's 0.11 seconds. This results in a performance improvement of 384% in Moodle's multi-server scenario compared to the single-server scenario.

The average value for the "course write" parameter in the single-server scenario is 0.15 seconds, while in the multi-server scenario, it averages 0.07 seconds, as depicted in Fig. 4. In this scenario, multi-servers can improve course write performance by 193%.

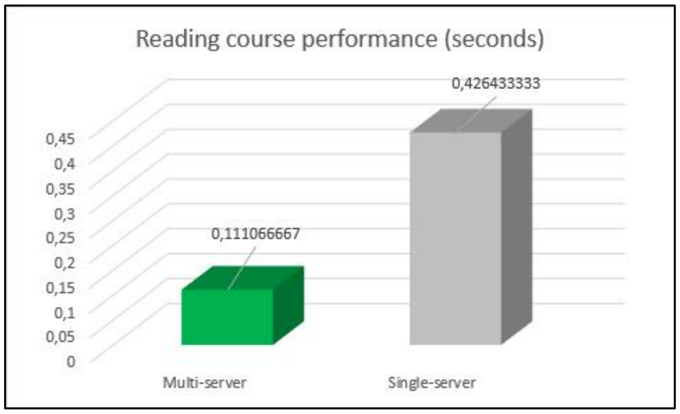

Fig. 3. Reading course performance comparison.

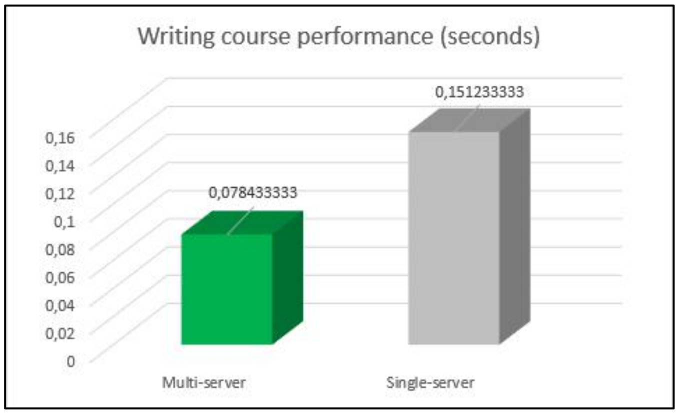

Fig. 4. Writing course performance comparison.

The last parameter being compared is database performance. In the single-server scenario, the average database performance is 0.25 seconds, whereas in the multi-server scenario, it averages 0.09 seconds, as shown in Fig. 5. Database performance improvement up to 260% in the multi-server scenario compared to the single-server scenario. Therefore, when these parameters are combined, the Moodle performance in the single-server scenario results in a time of 0.27 seconds, while in the multi-server scenario, it's 0.09 seconds. This demonstrates an improvement in performance in Moodle of 279% when compared to the single-server scenario, as shown in Fig. 6.

V. Conclusion

The study aimed to optimize Moodle performance by distributing the database into multiple servers. HAProxy handled load distribution with the least-connection algorithm, and the Galera Cluster feature on MariaDB does data synchronization. The Moodle performance plugin used course reading, course writing, and database performance to compare results.

P-ISSN: 2621-2536; E-ISSN: 2621-2544; DOI: 10.15408/aism.v7i1.34871

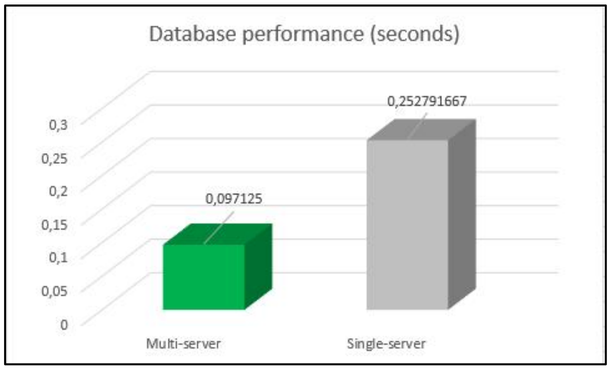

Fig. 5. Database performance comparison.

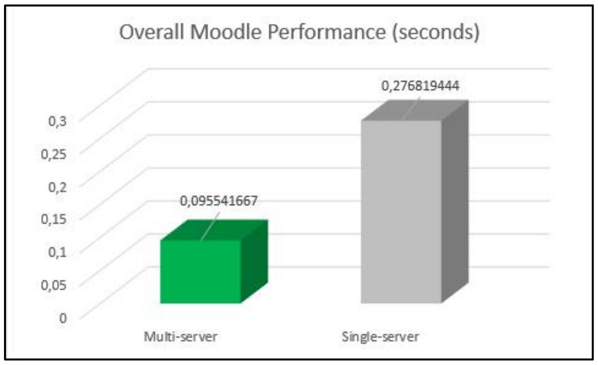

Fig. 6. Overall Moodle performance comparison.

The result shows that distributing the database into multiple servers gives an advantage over single-server, especially in peak access time where Moodle is accessed by massive users simultaneously. Reading course performance is increased by 384% faster in multiple-server scenarios compared to single-server scenarios. Writing course performance is increased by 193% faster in multiple-server scenarios compared to single-server scenarios. The essential thing in this research is performance improvement on the database up to 260% faster than single-server scenarios. Moodle performance is better in multiple-server scenarios, up to 297% faster than in single-server scenarios. This research proves that the database is the most critical part of Moodle. Distributing databases into multi-servers will improve overall Moodle performance drastically.

Acknowledgment

This research was supported by LITAPDIMAS grant number 1466A/LP2M/PPK/OT.01.3/04/2022 under LP2M UIN Maulana Malik Ibrahim Malang.

References

[1] S. B. Dias, S. J. Hadjileontiadou, J. Diniz, and L. J. Hadjileontiadis, “DeepLMS: a deep learning predictive model for supporting online learning in the Covid-19 era,” *Sci Rep*, vol. 10, no. 1, Dec. 2020, doi: 10.1038/s41598-020-76740-9.

[2] S. A. Raza, W. Qazi, K. A. Khan, and J. Salam, “Social isolation and acceptance of the learning management system (LMS) in the time of COVID-19 pandemic: An Expansion of the UTAUT Model,” *Journal of Educational Computing Research*, vol. 59, no. 2, pp. 183–208, Apr. 2021, doi: 10.1177/0735633120960421.

[3] A. H. Mujianto, C. Mashuri, G. S. Permadi, and R. Wiratsongko, “Analisa pemanfaatan learning management system schoology menggunakan HOT fit model terhadap pembelajaran di masa pandemi covid 19,” *Applied Information System and Management (AISM)*, vol. 5, no. 1, pp. 45–52, Apr. 2022, doi: 10.15408/aism.v5i1.24767.

[4] T. Y. Aikina and L. M. Bolsunovskaya, “Moodle-based learning: motivating and demotivating factors,” *International Journal of Emerging Technologies in Learning*, vol. 15, no. 2, pp. 239–248, 2020, doi: 10.3991/ijet.v15i02.11297.

[5] A. S. Mustafa and N. Ali, “The adoption and use of moodle in online learning: a systematic review,” *Information Sciences Letters*, vol. 12, no. 1, pp. 341–351, Jan. 2023, doi: 10.18576/isl/120129.

[6] K. Wiechork and A. S. Charão, “Investigating the performance of moodle database queries in cloud environments,” in *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems*, SciTePress, 2020, pp. 269–275, doi: 10.5220/0009792202690275.

[7] A. H. Ali and M. Z. Abdullah, “A survey on vertical and horizontal scaling platforms for big data analytics,” *International Journal of Integrated Engineering*, vol. 11, no. 6, pp. 138–150, 2019, doi: 10.30880/ijie.2019.11.06.015.

[8] P. Singh, P. Gupta, K. Jyoti, and A. Nayyar, “Research on auto-scaling of web applications in cloud: Survey, trends and future directions,” *Scalable Computing*, vol. 20, no. 2, pp. 399–432, Jun. 2019, doi: 10.12694/scpe.v20i2.1537.

[9] A. Zaini, H. Santoso, and M. P. T. Sulistyanto, “Fault tolerance strategy to increase moodle service reliability,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Apr. 2021. doi: 10.1088/1742-6596/1869/1/012095.

[10] M. Sadikin, R. Yusuf, and D. Arif Rifai, “Load balancing clustering on moodle LMS to overcome performance issue of e-learning system,” *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 17, no. 1, pp. 131–138, 2019, doi: 10.12928/TELKOMNIKA.v17i1.10284.

[11] S. Rajagopalan, “An overview of layer 4 and layer 7 load balancing,” *Lecture Notes on Data Engineering and Communications Technologies*, vol. 66, 2021, doi: 10.1007/978-981-16-0965-7_51.

[12] S. Ebneyousef and A. Shirmarz, “A taxonomy of load balancing algorithms and approaches in fog computing: a survey,” *Cluster Comput*, vol. 26, no. 5, pp. 3187–3208, Oct. 2023, doi: 10.1007/s10586-023-03982-3.

[13] A. H. Fathulloh and H. I. Adauwiyah, “Perbandingan tingkat efisiensi waktu query select pada database interface navicat dan SQLYog di MySQL DBMS,” *Applied Information System and Management (AISM)*, vol. 4, no. 2, pp. 101–105, Oct. 2021, doi: 10.15408/aism.v4i2.18369.

[14] K. P. Bhattarai, K. Visai, R. Ito, K. Sato, and B. P. Gautam, “Monitoring of e-learning system servers using the MariaDB galera cluster,” in *Proceedings - 2019 International Conference on Networking and Network Applications, NaNA 2019*, 2019. doi: 10.1109/NaNA.2019.00058.

[15] R. Shrestha, “High availability & performance of database in the cloud: Traditional master-slave replication versus modern cluster-based solutions,” in *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science*, 2017. doi: 10.5220/0006294604130420.

[16] T. Pohanka and V. Pechanec, “Evaluation of replication mechanisms on selected database systems,” *ISPRS Int J Geoinf*, vol. 9, no. 4, Apr. 2020, doi: 10.3390/ijgi9040249.

[17] S. Widiono, “Experiments and descriptive analysis in the mariadb database cluster system to prepare data availability,” 2019. [Online]. Available: www.codepolitan.com.

[18] G. Slavko and S. Serhiienko, “Optimization of LMS moodle configuration and education technologies on the example of electrical engineering education,” in *2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES)*, IEEE, Sep. 2021, pp. 1–5. doi: 10.1109/MEES52427.2021.9598719.

[19] R. Yusuf and H. Kusniyati, “The analyst model performance multi-tier for increase of efficiency virtual machine in moodle application,”

International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 94–101, Sep. 2020, doi: 10.32628/CSEIT206473.

[20] Z. Zdravev, A. Velinov, and S. Spasov, "Migration of moodle instance to the cloud – case study at Goce Delchev University," in *South East European Journal of Sustainable Development*, A. Pollozhani, Ed., Skopje: Mother Teresa University, Feb. 2021, pp. 99–106.

[21] A. H. da S. Marcondes, C. C. Miers, M. A. Pillon, and G. P. Koslovski, "SDN4Moodle: an SDN-based toolset to enhance qos of moodle platform," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, Jun. 2018, pp. 00627–00632. doi: 10.1109/ISCC.2018.8538523.

[22] F. Cardoso, A. Godinho, J. Rosado, F. Caldeira, and F. Sa, "Proposal of a technological cluster to support eLearning platform," in *2022 31st Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*, IEEE, Jun. 2022, pp. 1–5. doi: 10.1109/EAEEIE54893.2022.9820369.

[23] H. Triangga, I. Faisal, and I. Lubis, "Analisis perbandingan algoritma static round-robin dengan least-connection terhadap efisiensi load balancing pada load balancer haproxy," *InfoTekJar (Jurnal Nasional Informatika dan Teknologi Jaringan)*, vol. 4, no. 1, 2019, doi: 10.30743/infotekjar.v4i1.1688.

[24] A. Setiawan and W. M. Kansha, "Pembuatan sistem database cluster menggunakan aplikasi galera cluster di sekolah vokasi ipb university," *Jurnal Sains Terapan*, vol. 11, no. 2, 2021, doi: 10.29244/jstsv.11.2.49-59.

[25] M. Data, G. Ramadhan, and K. Amron, "Analisis availabilitas dan reliabilitas multi-master database server dengan state snapshot transfers (SST) Jenis Rsync Pada MariaDB Galera Cluster," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 1, 2017, doi: 10.25126/jtiik.201741288.