The Indonesian Journal of Computer Science
http://www.ijcs.net/
Volume 14, Issue 3, June 2025
https://doi.org/10.33022/ijcs.v14i3.4617

Optimizing the Performance of the PSHS CARC Knowledge Hub: A Mixed-Method Evaluation of a Moodle-Based LMS

Graceson D. Cuyasen[1]
gdcuyasen@carc.pshs.edu.ph[1]
[1] University of the Cordilleras, Baguio City, Philippines

Article Information

Received : 6 Jan 2025
Revised : 6 Jun 2025
Accepted : 16 Jun 2025

Keywords

performance optimization, mixed-method evaluation, server performance metrics, MOODLE, learning management system

Abstract

This study focuses on optimizing the performance of the Philippine Science High School – Cordillera Administrative Region Campus Knowledge Hub (PSHS-CARC KHub), a Moodle-based Learning Management System (LMS), by addressing key performance issues such as slow response times. A mixed-method approach, including a comprehensive literature review and experimental testing, was used to identify effective strategies for hardware, software, and front-end optimization. The study examined the impact of server configurations, memory upgrades, and Apache module optimizations on system performance. Results indicate that hardware optimizations (such as SSD deployment), software improvements (including database indexing and caching), and front-end enhancements (such as minimizing HTTP requests and optimizing images) led to measurable improvements in system scalability and responsiveness. While performance tests showed reduced response times and stable throughput, occasional delays for high-latency transactions were observed. These findings provide actionable insights for optimizing Moodle-based LMS platforms, improving both user experience and system efficiency.

A. Introduction

The PSHS-CARC Knowledge Hub, a Moodle-based Learning Management System (LMS), faces critical performance challenges, such as server crashes, slow response times, and high resource consumption. These issues disrupt the teaching and learning process, compromise system stability, and negatively impact the user experience. As an essential platform for delivering STEM education, addressing these challenges is crucial to ensure reliability, minimize downtime, and optimize resource efficiency.

The importance of Learning Management Systems (LMSs) has grown significantly within Science, Technology, Engineering, and Mathematics (STEM) programs and courses over the past decade, driven by better access to the internet and advancements in online teaching and learning technologies. Many educational institutions have successfully implemented LMSs and continue to explore their effectiveness across different platforms. Recent research in STEM education indicates that various LMSs and their associated tools enhance student engagement, motivation, and collaboration, performance, retention, and critical thinking. Furthermore, LMSs enable STEM educators to monitor learning outcomes, predict student performance (facilitating early identification of at-risk students), and use this information to adapt and refine their teaching practices. The future of STEM education can be further enhanced through innovative LMSs and technology-driven learning resources, including but not limited to online laboratories, online tutorials, and virtual reality applications. A recent systematic review of research trends in STEM education suggests that "learning environments," which include LMSs, are a key area poised for continued evolution[1].

Modular Object-Oriented Dynamic Learning Environment (MOODLE) is an open-source online learning management system recognized for its user-friendliness, intuitive interface, and extensive features. It is widely utilized by schools, universities, and businesses seeking to provide distance education. Moodle supports educational and communication functions to establish an online learning environment, enabling the creation of interactive courses and fostering a network of interactions among educators, learners, and learning resources. Due to its numerous advantages, Moodle has become a go-to platform in the field of online education [2].

Apache JMeter can be used to load test Moodle by generating heavy server loads to assess performance under different conditions. Moodle includes a JMeter test plan generator, allowing setup for local benchmarking with adjustable user loads and configurations. It’s essential to perform these tests on a non-production environment to avoid disruptions. The setup involves preparing servers resources, Java installation, and specific Moodle configurations, with customizable test sizes based on user loads.[3]

This study aims to identify and implement suitable optimization techniques, such as caching and load balancing, to maximize the use of key resources (CPU, RAM, database efficiency). Despite the existing research on Moodle optimization, there remains a significant gap in addressing these technical performance issues tailored specifically to the unique high-demand educational setup of PSHS-CARC. This research is conducted exclusively to bridge that gap, focusing on the specific

needs and challenges faced by PSHS-CARC's educational environment. Moreover, existing research has predominantly been conducted on laboratory servers rather than on actual servers hosting deployed, active Moodle environments. This limits their practical applicability, as they often fail to capture the real-world complexities of an active system. Additionally, many studies lack a mixed-method approach that combines quantitative performance metrics with qualitative feedback from users and administrators, further reducing the relevance of their findings to practical implementation.

To address these gaps, this study employs a mixed-method approach that includes a comprehensive literature review, analysis of server metrics, and qualitative feedback from system users and administrators. The key contributions of this study include developing targeted recommendations for database tuning, caching, and load balancing, which significantly enhance the scalability, stability, and reliability of the LMS. The findings provide a framework that educational institutions can use to deploy more efficient and adaptable LMS solutions to meet increasing user demands and technological advancements.

Building on the challenges and objectives outlined, this research seeks to address specific aspects of optimizing the PSHS-CARC Knowledge Hub. It is guided by three key questions: What optimization strategies best enhance the performance of the PSHS-CARC Knowledge Hub? How do hardware, software, and instructional design improvements collectively contribute to the scalability and efficiency of Moodle in educational environments like the PSHS-CARC Knowledge Hub? What practical strategies can be implemented to reduce server load and maintain system reliability, particularly in managing resource-intensive content and plugins? By addressing these questions, the study aims to develop targeted solutions that ensure the stability, scalability, and efficiency of the system in a high-demand educational context.

B. Methodology

The methodology for this study is structured across four key phases, each designed to address specific aspects of performance optimization for the PSHS CARC Knowledge Hub, a Moodle-based Learning Management Systems (LMS). These phases include a combination of literature review, case studies, and experimental testing to identify and evaluate effective strategies for enhancing LMS performance. By employing a mixed-method approach, the study integrates both qualitative insights and quantitative performance data, providing a comprehensive framework to optimize resource utilization, improve response times, and ensure scalability.

The Preliminary Research and Literature Review phase was to compile existing optimization techniques used in Moodle-based Learning Management Systems (LMS) to determine which methods were effective across different environments. The research involved a literature review of current studies on Moodle performance and a benchmark analysis comparing global tools and techniques for LMS optimization. The outcome was a list of strategies which could be tested to improve Moodle's efficiency.

The Case Studies and Interviews phase aimed to gather qualitative data through case studies on PSHS campuses using Moodle-based systems to identify

real-world challenges and applied optimization solutions. The research involved selecting three (3) campuses with varying infrastructure levels and conducting semi-structured interviews with the LMS Coordinators. These interviews explored performance challenges, system configurations, and optimization practices. The outcome was a set of insights into the strengths and weaknesses of current optimization approaches in real-world settings, which informed subsequent experimental phases.

The Experimental Research phase aimed to evaluate and implement various optimization techniques for the PSHS CARC Knowledge Hub, a Moodle-based Learning Management System (LMS) actively deployed at the time. Optimization strategies included expanding Random Access Memory (RAM) capacity, and software and backend optimization. Performance testing was conducted using JMeter to simulate user loads and measure key metrics, including response time and throughputs. Data was collected before and after implementing optimizations to compare performance improvements under different user loads. The outcome was a set of quantitative results identifying the most effective optimization techniques under specific conditions, providing insights for enhancing system performance in similar educational contexts.

The host system is powered by an Intel® Xeon® E-2286G processor running at 4.00 GHz with 12 physical cores. It includes an integrated Matrox G200eW3 Graphics Controller, 40 GB of RAM, and an 8 TB SATA HDD spinning at 7,200 RPM. The server operates on the Xen Cloud Platform - Next Generation (XCP-ng) version 8.2.1, an open-source virtualization platform based on the Xen hypervisor. XCP-ng provides enterprise-grade features such as live migration and snapshots, making it an efficient and cost-effective choice for data center environments [16].

The PSHS-CARC KHub LMS was deployed on a virtual machine (VM) hosted on this platform, configured with a 4-core virtual CPU, 4 GB of RAM, and 100 GB of storage.

The Data Analysis and Synthesis phase integrated findings from the previous phases to develop a comprehensive understanding of the optimization techniques for Moodle-based Learning Management Systems (LMS). This phase focused on analyzing qualitative data from Phases 1 and 2 through thematic analysis and interpreting quantitative data from Phase 3 through simple observation of performance metrics.

C. Result and Discussion

This section delves into the findings of this research, presenting a comprehensive analysis of both preliminary research and relevant literature. By examining various optimization techniques, we aim to highlight the most effective strategies for enhancing Moodle's performance. The discussion is structured to address key areas of hardware and software optimization, providing insights into their impact on user experience and system efficiency.

This section presents the results of preliminary research and a review of relevant literature, establishing a foundational understanding of Moodle optimization and identifying existing gaps in the field. The solutions identified through the literature review encompass Hardware Optimization, Software and

Backend Optimization, Front-End Optimization, and Instructional Design Optimization.

For Hardware Optimization, implementing solid-state drives (SSDs) for Moodle servers and databases can significantly reduce user response times. A study demonstrated that deploying Moodle on SSDs led to the most substantial decrease in end-user response time compared to other optimization techniques[4]. For this research however, implementing solid state drives is impossible as access to the server hardware is limited.

Software and Backend Optimization includes regular updates and maintenance of Moodle are essential to ensure optimal performance. Approaches such as efficient database indexing, caching mechanisms, and load balancing have been recognized as effective strategies for improving Moodle's responsiveness and scalability[5].

Improving the user interface and experience is vital for user engagement. Techniques like minimizing HTTP requests, optimizing images, and leveraging browser caching can enhance Moodle's front-end performance, leading to faster page loads and a more responsive interface [4].

This section presents insights gathered from case studies and interviews, highlighting real-world examples and experiences that illustrate the effectiveness of various Moodle optimization strategies.

Minimizing the use of Moodle plugins, particularly those that require significant system resources, such as real-time analytics, video conferencing, advanced reporting tools, or interactive content tools like H5P.

To enhance server performance, campuses offering KHub services exclusively through a local network power down the server daily. Since there is no demand for server usage during the night, the server is shut down at the end of each day after dismissal and restarted the following morning. This approach helps reduce energy consumption, minimize hardware wear, and maintain overall server efficiency.

Utilizing third-party websites to host resource-intensive content can significantly enhance system performance. For example, uploading videos to video streaming platforms and sharing only the links, or using cloud storage services for large files and providing links to those files, reduces the load on the local server. This approach not only improves overall system performance but also ensures better accessibility and user experience.

Conducting regular server clean-ups is essential for maintaining optimal performance. Automating this process whenever feasible ensures consistency and efficiency, reducing the likelihood of performance issues and minimizing manual intervention.

This section presents findings from experimental research, showcasing controlled studies that demonstrate the impact and effectiveness of various Moodle optimization strategies. Testing was conducted using JMeter. The test cycle involved logging in, viewing the course outline, and logging out. A total of 500 dummy accounts were created to simulate simultaneous logins to Moodle.

The initial test focused on evaluating Apache's mpm_prefork, mpm_worker, and mpm_event modules to determine which performs better. The mpm_prefork module uses multiple single-threaded processes to handle requests, ensuring

stability and compatibility with non-thread-safe libraries. The mpm_worker module combines multi-threading with multiple processes, offering improved scalability and efficiency. The mpm_event module extends the worker model by optimizing for keep-alive connections, making it more suitable for high-traffic environments. mpm_prefork and mpm_worker exhibit identical performance in most response time metrics, with mpm_prefork having a slight edge due to a lower error rate. mpm_event offers slightly higher throughput but at the cost of increased response times (especially at higher percentiles) and a marginally higher failure rate. If reliability and consistent performance are the priorities, mpm_prefork is the best option. However, if higher throughput is more critical, mpm_event is worth considering despite its trade-offs.

The second test was designed to evaluate the performance of different system configurations by comparing the impact of varying RAM sizes on response time, throughput, and error rates. The configurations tested were all based on a 4-core CPU, with RAM sizes of 4 GB, 8 GB, and 16 GB. Each configuration was subjected to 14,500 samples, measuring key performance indicators such as average response time, minimum and maximum response times, as well as throughput and error rates. The objective was to determine how increasing RAM affects system performance, particularly in terms of responsiveness and transaction handling, while also analyzing error occurrences. The data reveals that the 4 Core, 4 GB RAM configuration exhibits the highest average response time of 304.10 ms, while maintaining zero errors and a throughput of 27.97 transactions per second. Upgrading to 8 GB RAM slightly improves the performance, reducing the average response time to 257.86 ms, with minimal errors (0.01%) and a marginal increase in throughput to 28.06 transactions per second. Further increasing the RAM to 16 GB brings the average response time down to 250.18 ms, but this configuration experiences the lowest throughput of 27.81 transactions per second, with no errors. Overall, while the increase in RAM leads to reduced response times, the changes in throughput and error rates are minimal. In addition, observations through Xen Orchestra's stats monitoring revealed that the server did not fully utilize the 8 GB and 16 GB memory, even when high loads were simulated on Moodle.

Further optimization tests were conducted by adjusting key Apache configuration values. StartServers (5) determines the number of worker processes started initially to handle requests, while MinSpareServers (5) and MaxSpareServers (10) control the minimum and maximum number of idle worker processes to efficiently manage traffic spikes without wasting resources. MaxRequestWorkers (150) limits the maximum number of simultaneous requests to prevent server overload, and MaxConnectionsPerChild (1000) ensures each worker handles up to 1000 requests before being replaced, preventing resource leaks and maintaining performance. These adjustments aim to improve resource utilization and response times under heavy workloads.

After optimizations, the system showed improved average (down by 6 ms) and median (down by 42 ms) response times, with no errors and stable throughput (27.79 vs. 27.81 transactions per second). However, the maximum response time and the 90th, 95th, and 99th percentiles increased, indicating that while most requests became faster, the slowest requests took significantly longer

to process. The optimizations improved typical performance but led to occasional delays for certain requests, suggesting that while the system's overall responsiveness improved, some high-latency transactions now experience longer processing times. Further investigation may be needed to address these increased latencies in the higher percentiles.

D. Conclusion

In conclusion, this research highlights the importance of various optimization techniques in enhancing the performance of Moodle-based systems. Through a comprehensive review of hardware, software, and front-end optimization strategies, combined with experimental testing, the study emphasizes the potential improvements in user experience and system efficiency. Key findings suggest that hardware optimizations like solid-state drives, regular software updates, and efficient server configurations can significantly reduce response times and increase scalability. Furthermore, front-end optimizations, such as minimizing HTTP requests and optimizing images, also play a crucial role in boosting user engagement and system responsiveness. While the experimental results show improvements in response times and throughput, the research also points to areas requiring further investigation, such as minimizing latencies in higher percentiles. Overall, implementing these optimization strategies can substantially improve the performance and user experience of Moodle systems.

E. References

[1] S. H. P. W. Gamage, J. R. Ayres, and M. B. Behrend, “A systematic review on trends in using Moodle for teaching and learning,” Int. J. STEM Educ., vol. 9, no. 1, p. 9, Jan. 2022, doi: 10.1186/s40594-021-00323-x.

[2] Y. Zhang, A. Ghandour, and V. Shestak, “Using Learning Analytics to Predict Students Performance in Moodle LMS,” Int. J. Emerg. Technol. Learn. IJET, vol. 15, p. 102, Oct. 2020, doi: 10.3991/ijet.v15i20.15915.

[3] A. Sharif, “How to Benchmark Performance of Moodle,” Severalnines. Accessed: Nov. 09, 2024. [Online]. Available: https://severalnines.com/blog/how-to-benchmark-performance-moodle/

[4] P. Manchanda, “Analysis of Optimization Techniques to Improve User Response Time of Web Applications and Their Implementation for MOODLE,” Dec. 25, 2013, arXiv: arXiv:1309.7173. Accessed: Nov. 09, 2024. [Online]. Available: http://arxiv.org/abs/1309.7173

[5] “The Adoption and Use of Moodle in Online Learning: A Systematic Review,” Inf. Sci. Lett., vol. 12, no. 1, pp. 341–351, Jan. 2023, doi: 10.18576/isl/120129.

[6] S. H. P. W. Gamage, J. R. Ayres, M. B. Behrend, and E. J. Smith, “Optimising Moodle quizzes for online assessments,” Int. J. STEM Educ., vol. 6, no. 1, p. 27, Aug. 2019, doi: 10.1186/s40594-019-0181-4.