Formosa Journal of Computer and Information Science (FJCIS)
Vol. 5, No. 1 2026: 49-62

Microservices-Based Open-Source Video Conference Deployment for Optimized Online Learning Infrastructure

Davy Putra Ananda[1], Muhammad Fadhil Ramadhan Wicassono[2], Farhah Safrila Diva[3], Abdullah Rasyid[4], Juwita Istiqomah Trahira[5], Neny Rosmawarni[6*]
Universitas Pembangunan Nasional "Veteran" Jakarta
Corresponding Author: Neny Rosmawarni nenyrosmawarni@upnvj.ac.id

A R T I C L E I N F O

Keywords: OpenMeetings, Docker, WSL2, Virtualization, Server Performance

Received : 20, November
Revised : 22, January
Accepted: 24, March

A B S T R A C T

The rapid advancement of information technology has fundamentally shifted the interaction paradigm in education from conventional methods to hybrid learning models. In this context, the availability of stable, real-time communication platforms has become crucial for maintaining the effectiveness of knowledge transfer. This study evaluates the implementation of Apache OpenMeetings v9.0.0 using Docker and WSL2 to provide efficient video conferencing. Using an experimental methodology, system performance was monitored during active sessions. Results show high resource efficiency with a stable CPU utilization of 4.94% and memory usage of 1.339 GiB. The system achieved a rapid startup velocity of 11.1 seconds, proving that containerization offers optimal isolation with minimal overhead. The study concludes that this architecture provides a lightweight, portable, and cost-effective solution for independent communication infrastructure in educational institutions.

DOI prefik: https://doi.org/10.55927/fjcis.v5i1.16470
ISSN-E: 2830-3040
https://journal.formosapublisher.org/index.php/fjcis

INTRODUCTION

The rapid advancement of information technology has fundamentally shifted the interaction paradigm in education from conventional methods to hybrid learning models. In this context, the availability of stable, real-time communication platforms has become crucial for maintaining the effectiveness of knowledge transfer. However, institutions often face challenges when relying on third-party video conferencing providers, including high subscription costs and sensitive research data privacy concerns. Consequently, developing an independent, open-source video conferencing infrastructure is a strategic alternative.

Apache OpenMeetings is a prominent open-source platform offering a comprehensive suite of features, including video conferencing, whiteboards, and meeting room management. Despite its extensive capabilities, traditional multimedia server installations frequently encounter complexities due to system dependency conflicts and inconsistent environment configurations. Containerization technology through Docker offers a solution by encapsulating the application and its entire dependency stack into a single, isolated container unit. The utilization of Docker ensures that the application runs consistently across various infrastructures without conflicting with other services on the host system.

In a Windows-based operating system environment, Docker integration is further optimized with the presence of Windows Subsystem for Linux 2 (WSL2). WSL2 provides a native Linux kernel that allows containers to operate with performance levels nearing a native Linux environment while maintaining the flexibility of management through the Windows interface. This combination is expected to provide better system resource efficiency compared to traditional virtualization methods based on Virtual Machines (VMs), which tend to be resource-intensive.

This study aims to analyze the implementation and performance of the latest version of Apache OpenMeetings (v9.0.0) running on a Docker and WSL2 architecture. The primary focus of this research lies in evaluating system resource utilization, specifically monitoring Central Processing Unit (CPU) utilization and Random Access Memory (RAM) usage in real-time. The results of this study are expected to serve as a technical reference for institutions in building a lightweight, stable, and portable independent communication infrastructure.

LITERATURE REVIEW

Online Learning

Online learning, also referred to as e-learning, is a mode of education that leverages internet technology and digital devices as the primary medium for teaching and learning. Brika et al. (2022) define e-learning as an educational approach that has been widely validated during the COVID-19 pandemic, noting a significant surge in related research publications from 2020 onward. Its core advantages lie in the flexibility of time and location, broader accessibility, and the capacity to accommodate large numbers of learners simultaneously without physical space constraints.

Online learning, also referred to as e-learning, is a mode of education that leverages internet technology and digital devices as the primary medium for teaching and learning. Brika et al. (2022) define e-learning as an educational approach that has been widely validated during the COVID-19 pandemic, noting a significant surge in related research publications from 2020 onward. Its core advantages lie in the flexibility of time and location, broader accessibility, and the capacity to accommodate large numbers of learners simultaneously without physical space constraints.

Massive Open Online Course (MOOC)

A Massive Open Online Course (MOOC) is an online education model that enables unlimited and open participation via the internet. Alturkistani et al. (2020) conducted a systematic review of MOOC evaluation methods and found that the most commonly used approaches include learner satisfaction surveys, activity log data analysis, and learning outcome assessments. Leading MOOC platforms such as Coursera and edX have partnered with hundreds of universities worldwide, collectively serving tens of millions of learners and offering credentials recognized by industry.

Despite their considerable potential in democratizing education, MOOCs continue to face the persistent challenge of low course completion rates. Billsberry and Alony (2024), drawing on a comprehensive bibliometric and systematic review of MOOC research from 2009 to 2022, identified four primary research foci: student-focused, design-focused, context and impact, and instructor-focused studies. Their findings suggest that learner engagement is a critical determinant of MOOC success, influenced by both internal factors such as motivation and self-regulation, and external factors such as technical support and course design quality.

Client-Server Communication in Networks

Client-server architecture is the most widely adopted network computing model in modern information systems, including online learning platforms. Nyabuto (2024) describes client-server architecture as a software model in which the client sends a request over a network, the server receives and processes it, and then returns an appropriate response, thereby enabling multiple users to access shared resources concurrently. Communication between client and server typically relies on the TCP/IP protocol suite, which serves as the foundational framework of the modern internet.

In the context of video conferencing applications such as Apache OpenMeetings, client-server communication involves additional protocols designed to support real-time multimedia data transmission, including WebRTC for peer-to-peer audio and video directly in the browser, and RTMP for media streaming between server and client. Security in this communication layer is enforced through Transport Layer Security (TLS), which ensures encryption of data in transit, thereby protecting user privacy and maintaining data integrity in compliance with applicable data protection regulations.

Apache OpenMeetings

Apache OpenMeetings is an open-source web conferencing application that provides video conferencing, interactive whiteboard, document collaboration, screen sharing, and user management features. According to the official Apache OpenMeetings website, the platform uses the Kurento Media Server for audio and video streaming and fully supports WebRTC, enabling real-time communication directly in the browser without requiring any additional client-side plugins. As an official project under the Apache Software Foundation, it is distributed under the Apache License 2.0, making it a cost-effective solution for educational institutions.

OpenMeetings also provides a REST and SOAP API that enables seamless integration with major Learning Management Systems such as Moodle, Drupal, and SugarCRM. A study by Zahid and Falgárt (2022) demonstrated that the integration of Moodle with Apache OpenMeetings at a naval academy successfully established an effective distance learning environment, where remote participants received complete audio, video, and documentary information comparable to physical in-room attendance. This evidence highlights OpenMeetings as a viable open-source alternative to proprietary platforms, particularly for institutions that prioritize data sovereignty and privacy.

Docker

Docker is an open-source platform that implements operating system-level virtualization, often referred to as containerization. It allows developers to package applications and their entire dependency stack into a single, isolated unit called a container. Unlike traditional virtual machines that require a full guest operating system, Docker containers share the host's OS kernel, which significantly reduces resource overhead and ensures high portability across different computing environments.

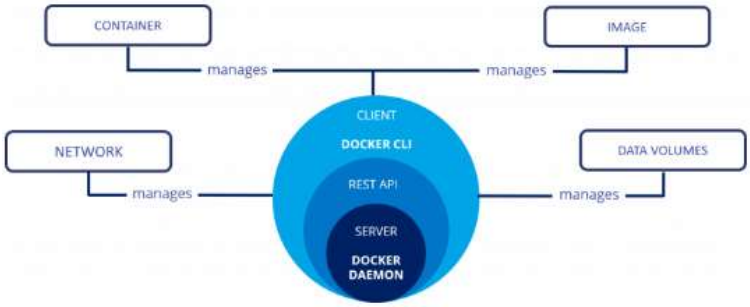

Figure 1. Docker

Source: www.docs.docker.com

In the context of multimedia servers, Docker ensures that complex dependencies such as Java runtimes and media servers are managed consistently. The core architecture involves the Docker Daemon (Server) which interacts with the Client through a REST API to manage various objects including containers, images, networks, and data volumes as illustrated in Figure 1. Furthermore, the deployment of container-based architectures allows for enhanced scalability and rapid deployment, enabling developers to maintain system stability even under varying network loads (Pohan, 2020).

Windows Subsystem for Linux 2 (WSL2)

Windows Subsystem for Linux 2 (WSL2) is a compatibility layer developed by Microsoft that enables a full Linux kernel to run natively on the Windows operating system. Unlike WSL1, which relies on system call translation, WSL2 uses a lightweight virtual machine architecture with a real Linux kernel, resulting in improved compatibility and performance, particularly for I/O-intensive and container-based workloads (Microsoft, 2023).

WSL2 integrates seamlessly with Docker, allowing containers to run with near-native Linux performance while maintaining the convenience of a Windows environment. This integration eliminates the need for traditional virtual machines and reduces system overhead (Docker Inc., 2020).

Additionally, WSL2 enhances file system performance, supports modern Linux features such as groups and namespaces, and enables efficient resource utilization through dynamic allocation. These capabilities make it well-suited for deploying real-time multimedia applications, including video conferencing systems that rely on stable backend processing (Microsoft, 2023).

Microservices Architecture

Microservices architecture is an approach that structures applications as a collection of small, independent services that communicate through lightweight APIs. According to National Institute of Standards and Technology (NIST), this model enables each service to be developed, deployed, and scaled independently (NIST, 2022).

Compared to monolithic architecture, microservices provide better scalability, flexibility, and fault isolation. The use of containerization technologies such as Docker further enhances this approach by ensuring consistent environments and efficient resource utilization (Pahl, 2015). This architecture is particularly suitable for real-time applications such as video conferencing systems, where different components can operate independently to improve system performance and reliability.

METHODOLOGY

The research methodology is designed to provide a systematic framework for developing and testing the containerized video conferencing system. This study adopts the Network Design Life Cycle (NDLC) approach, which is specifically tailored for network-oriented projects that require high levels of availability and stability. The iterative nature of this framework allows for continuous refinement during the technical deployment phase to ensure optimal server performance. The specific stages of the methodology utilized in this research are visualized in Figure 2.

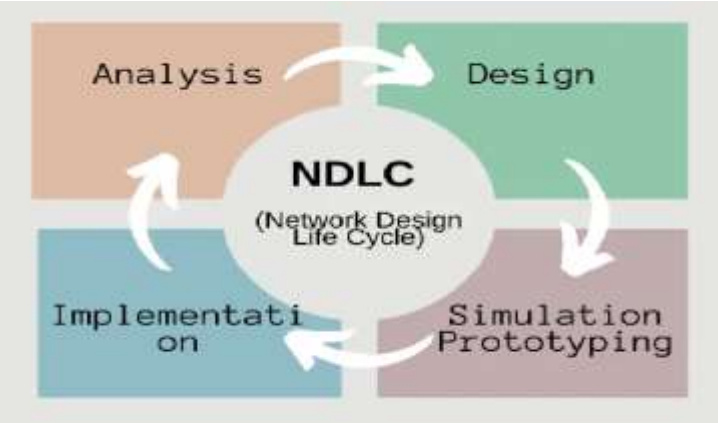

Figure 2. NDLC Methodology (Faldi et al., 2023)

As illustrated in Figure 2, the NDLC framework consists of four primary stages that guide the research from initial requirement gathering to the final deployment. The process begins with the Analysis phase to identify infrastructure needs, followed by the Design phase to architect the Docker environment. Subsequently, the Simulation Prototyping and Implementation phases are executed to validate the system's responsiveness and resource efficiency. By following this structured lifecycle, the research ensures that the video conferencing server remains scalable and secure throughout its operational cycle.

RESEARCH RESULT

Analysis

To assess the viability and long-term sustainability of the containerized communication platform, a comprehensive SWOT analysis is presented in Table 1.

Table 1. SWOT Analysis of OpenMeetings v9.0.0 Implementation

Category	Analysis Description
Strengths	1. Docker containers eliminate dependency conflicts across different hardware. 2. Offers full collaborative tools (whiteboard, recording, screen sharing) as an open-source solution.
Weaknesses	1. Performance is strictly bound to the host's physical RAM and CPU capacity during high-concurrency sessions. 2. Requires specialized knowledge in Docker CLI, WSL2 configuration, and network port mapping.
Opportunities	1. Enables institutions to eliminate high subscription fees for third-party proprietary platforms. 2. The architecture is ready for future integration with orchestration tools like Kubernetes for larger deployments.
Threats	1. As a self-hosted system, security relies entirely on the institution's internal firewall and network management.

2. Requires periodic updates to maintain compatibility with evolving WebRTC and Docker standards.

Design (Architectural Topology)

The design phase establishes the structural foundation of the video conferencing infrastructure. The architecture is designed as a multi-layered system that bridges the Windows host environment with a high-performance Linux backend to support real-time multimedia processing. The architectural topology consists of several key layers as follows in Figure 3.

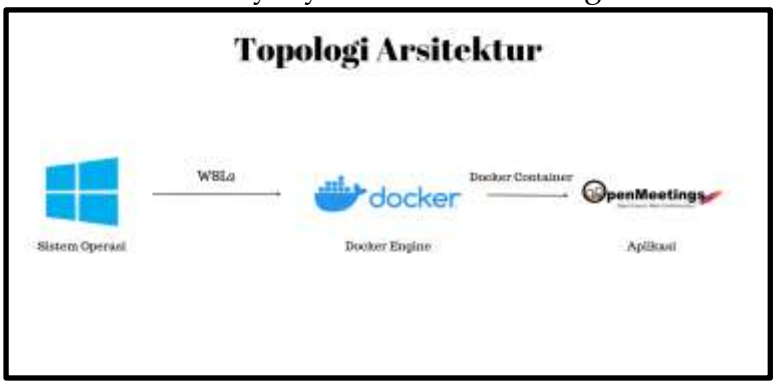

Figure 3. Architectural Topology of OpenMeetings

As illustrated in Figure 3, the architecture follows a four-layer integration process. It begins with the Windows Operating System as the host, which utilizes WSL2 to provide a high-performance Linux kernel. The Docker Engine then runs on this kernel to manage the container environment, where the OpenMeetings Application is deployed. This layered design ensures system isolation, making the infrastructure portable and easy to maintain without affecting the host's primary configuration.

Simulation/Prototyping (Protocol Configuration)

The simulation phase serves as a technical validation stage to ensure that the designed architecture and communication protocols are functional within the local environment. This phase involves several critical configuration steps:

a. Docker Environment Initialization by activating the Docker service on the host system to prepare the container management environment for image deployment.
b. Port Mapping Configuration to link the container's internal services with the host's network interface via port 5443 for secure HTTPS access.
c. WSL2 Backend Verification ensuring native Linux kernel support for the Kurento Media Server and its required directory structures.
d. Startup Sequence Monitoring using command line logs to identify the server startup status for Apache Tomcat and multimedia services.
e. Local Connectivity Simulation through localhost or host IP to confirm the functional state of the login and registration interfaces.
f. Network Isolation Validation confirming the OpenMeetings container remains isolated from the host operating system while maintaining stable multi-device connectivity.

Implementation

The implementation phase represents the realization of the architectural design into a functional video conferencing environment. The results of the deployment are analyzed through the following technical verification stages

Figure 4. Desktop Setup

The initial stage of implementation involves the deployment of the OpenMeetings image within the Docker Desktop environment. As illustrated in Figure 4, the system successfully initializes the containerized environment on the Windows host via the WSL2 backend. The execution logs confirm that the Apache Tomcat web server and the Kurento Media Server have reached an operational state within 11.1 seconds. This visual evidence verifies that all internal dependencies, including Java runtimes and media processing modules, are correctly configured and isolated from the host operating system.

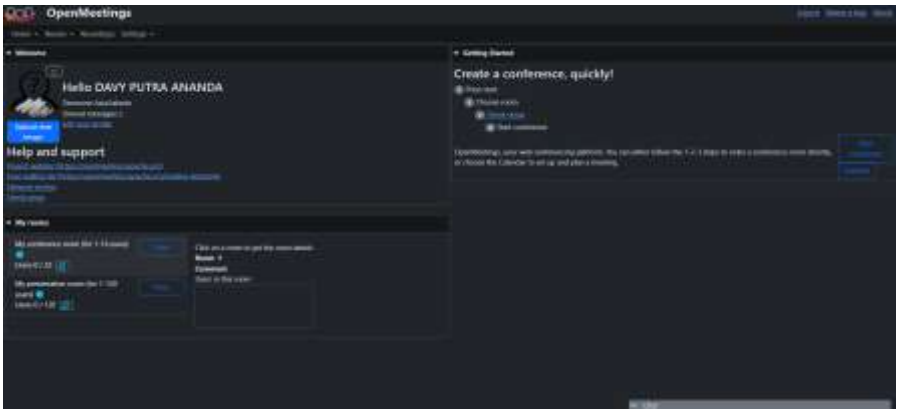

Figure 5. Dashboard OpenMeetings

Following a successful container startup, the accessibility of the administrative interface is verified through a web browser. Figure 5 displays the OpenMeetings main dashboard, confirming that the system is reachable via the mapped port 5443 using the HTTPS protocol. The interface allows for real-time user management and room configuration, demonstrating that the database and web modules are fully synchronized. The presence of the personalized user profile in the dashboard further validates the successful integration of the administrative credential system.

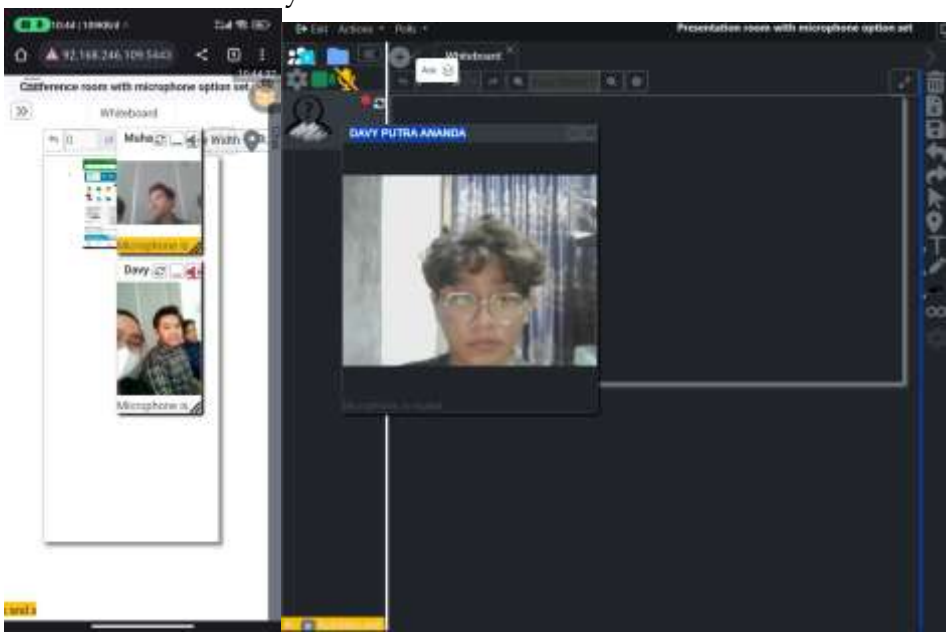

Figure 6. Video Conference

The successful establishment of a video conferencing session as illustrated in Figure 6 demonstrates the system's capability to handle real-time multimedia synchronization across heterogeneous client platforms. The implementation of the Kurento Media Server as the backend engine facilitates stable audio and video transmission between the desktop-based server interface and mobile client endpoints without significant latency. This functional state confirms that the WebRTC protocols are correctly configured within the Docker container to manage high-bandwidth data streams while maintaining session integrity. Furthermore, the visual evidence in Figure 6 proves that the responsive design of the OpenMeetings v9.0.0 interface allows for seamless collaborative interactions including the use of participant video slots and real-time chat modules. The ability to maintain this connectivity across different network interfaces validates the effectiveness of the port mapping strategy and the overall reliability of the containerized architecture for remote educational applications.

DISCUSSION

Hardware and Physical Component Analysis

The physical infrastructure used to host the Apache OpenMeetings v9.0.0 server consists of a mobile workstation environment with specific hardware capabilities to ensure stable container operations. The host system is powered by an 11th Gen Intel Core i5-1135G7 processor with a base frequency of 2.40 GHz and a turbo boost capability up to 2.42 GHz. For memory management, the system is equipped with 8.00 GB of installed RAM (with 7.73 GB usable capacity), providing the necessary headroom for running the WSL2 Linux kernel and Docker Desktop simultaneously.

The software environment operates on a 64-bit Windows 11 Home Single Language operating system, version 23H2, which fully supports the advanced virtualization features required by Docker. Detailed hardware and system environment specifications extracted from the host machine are summarized in Table 2.

Table 2. Physical and System Environment Specifications

Category	Analysis Description
Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Installed RAM	8 GB
System Type	64-bit Operating System, x64-based processor
Operating System	Windows 11 Home Single Language 23H2
Virtualization	Docker Desktop with WSL2 Backend
Application Version	Apache OpenMeetings v9.0.0

Performance Testing Results

The evaluation of system performance during the operation of Apache OpenMeetings was captured using the docker stats utility. Based on the hardware specifications mentioned above, the testing results during an active session are summarized in Table 3.

Table 3. Performance Testing Results

Metric Parameter	Observed Value
CPU Utilization	4.94%
Memory Usage (RAM)	1.339 GiB
Memory Percentage	17.56%
Network Input (Net I/O)	20.7 MB
Network Output (Net I/O)	14.2 MB
Block I/O (Disk)	528 MB / 28.3 MB
Active Processes (PIDs)	223

The empirical data captured in Table 3 demonstrates that the containerized deployment of Apache OpenMeetings v9.0.0 provides a highly optimized communication environment with minimal resource overhead. The CPU utilization of 4.94% indicates that the 11th Gen Intel Core i5 processor manages real-time multimedia encoding and decoding tasks with significant ease, leaving substantial processing headroom for other concurrent background services. Furthermore, the memory usage of 1.339 GiB, which constitutes only 17.56% of

the host's total RAM, proves that the system is suitable for implementation on mid-range hardware typically found in academic laboratories

Analysis of the network throughput reveals a dynamic data exchange, where the 20.7 MB input and 14.2 MB output reflect the server's role in processing high-definition WebRTC streams from connected clients. The Block I/O metrics, recording 528 MB of data reading, further evidence active system logging and temporary file management within the WSL2 virtual hard disk, ensuring session persistence without affecting the host's primary storage performance. With a stable process count of 223 PIDs, the isolated Docker environment successfully maintains all necessary backend modules, including Apache Tomcat and Kurento Media Server, while ensuring that the host operating system remains secure and unaffected by the application's internal dependencies.

Comparative Analysis with Proprietary Platforms

The implementation of Apache OpenMeetings v9.0.0 via Docker offers several distinct advantages over mainstream proprietary platforms such as Zoom and Google Meet. Unlike commercial services that require monthly subscription fees for extended features, OpenMeetings provides a full suite of collaborative tools including multi-whiteboards and session recording under an open-source license. From a technical standpoint, the containerized architecture allows for localized data hosting which ensures higher data sovereignty and privacy compared to cloud-based solutions that store academic data on third-party servers.

Furthermore, the resource efficiency measured in this study proves that OpenMeetings is highly competitive in terms of infrastructure requirements. While proprietary applications often demand high-bandwidth and significant background processing power, the observed 4.94% CPU utilization and 1.339 GiB RAM usage demonstrate that this system can be deployed on existing campus hardware without specialized upgrades. This localized deployment also eliminates the dependency on external service availability, providing the institution with a stable and independent communication environment that is fully customizable to specific educational needs. The strategic and technical differences between the implemented open-source system and mainstream proprietary platforms are summarized in Table 4.

Table 4. Comparison between OpenMeetings and Proprietary Platforms

Feature	Apache OpenMeetings (Docker)	Proprietary (Zoom/Google Meet)
Licensing Cost	Free (Open Source)	Subscription Based
Data Privacy	Localized (Data Sovereignty)	Third-Party Cloud Servers
Customization	Fully Customizable Source Code	Limited to Provider Features
Deployment	Self-Hosted Container	Service-as-a-Software (SaaS)
Dependency	Independent of External Providers	High Dependency on Provider Uptime

Based on the comparative data presented in Table 4, it is evident that Apache OpenMeetings serves as a powerful and strategic alternative to proprietary platforms, particularly for academic institutions prioritizing data sovereignty and cost-efficiency. While mainstream applications offer convenience through cloud-based SaaS models, the containerized OpenMeetings infrastructure provides unparalleled control over the source code and internal security configurations. The ability to maintain such a feature-rich environment with minimal resource consumption requiring only 4.94% CPU and 1.339 GiB RAM validates that the transition from commercial subscriptions to independent open-source solutions is both technically viable and economically beneficial for long-term educational operations.

CONCLUSIONS AND RECOMMENDATIONS

Based on the implementation and performance evaluation, this research concludes that the deployment of Apache OpenMeetings v9.0.0 utilizing Docker and WSL2 provides a highly optimized and stable infrastructure for online learning. The empirical results demonstrate exceptional resource efficiency, with the system maintaining a low CPU utilization of 4.94% and a memory footprint of 1.339 GiB, which constitutes only 15.67% of the allocated resources. Furthermore, the rapid startup velocity of 11.1 seconds confirms that this microservices-based approach offers superior agility for rapid deployment compared to traditional virtualization.

The integration of containerization successfully addresses the challenges of dependency conflicts and high infrastructure costs typically associated with proprietary platforms. By leveraging an open-source framework, educational institutions can achieve full data sovereignty and eliminate recurring licensing fees while maintaining high-quality real-time multimedia synchronization. Ultimately, this architecture serves as a strategic and cost-effective technological reference for developing independent, portable, and secure communication platforms within academic environments.

Suggestions for further development to enhance this system are as follows:

1. Scalability Expansion by implementing container orchestration using Kubernetes or Docker Swarm to handle larger concurrent user loads across multiple server nodes.
2. Feature Enhancement through the integration of external recording storage and additional collaborative plugins to expand the system's educational functionality.
3. Advanced Security Hardening by implementing external SSL certificates and advanced firewall configurations to ensure safe public network accessibility.
4. Network Infrastructure Stability ensuring high-bandwidth and low-latency internet connectivity to maintain the quality of real-time multimedia streams.

ADVANCED RESEARCH

This research is primarily limited to a localized deployment within a workstation environment using a single-node container. While the results demonstrate high efficiency, further investigation is required to analyze the system's performance under heavy concurrent stress from a massive number of users in a real-world MOOC scenario. Future studies should also explore the integration of decentralized media servers to improve streaming resilience and conduct a comprehensive security audit involving penetration testing to evaluate the platform's defense against sophisticated cyber threats in a public network environment.

REFERENCES

Alturkistani, A., Lam, C., Foley, K., Stenfors, T., Blum, E. R., Van Velthoven, M. H., & Meinert, E. (2020). Massive Open Online Course Evaluation Methods: Systematic Review. Journal of Medical Internet Research, 22(4), e13851. https://doi.org/10.2196/13851 https://www.jmir.org/2020/4/e13851.

Billsberry, J., & Alony, I. (2024). The MOOC Post-Mortem: Bibliometric and Systematic Analyses of Research on MOOCs, 2009 to 2022. Journal of Management Education. https://doi.org/10.1177/10525629231190840 https://journals.sagepub.com/doi/10.1177/10525629231190840.

Brika, S. K. M., Chergui, K., Algamdi, A., Musa, A. A., & Zouaghi, R. (2022). E-Learning Research Trends in Higher Education in Light of COVID-19: A Bibliometric Analysis. Frontiers in Psychology, 12, 762819. https://doi.org/10.3389/fpsyg.2021.762819

Caprara, G. V., & Zimbardo, P. G. (2004). Personalizing politics: A congruency model of political preference. American Psychologist. https://doi.org/10.1037/0003-066X.59.7.581.

Diener, E. (2000). Subjective well-being: The science of happiness and a proposal for a national index. American Psychologist. https://doi.org/10.1037/0003-066X.55.1.34.

Docker Inc. (2020). Introducing the Docker Desktop WSL 2 Backend. Diakses dari https://www.docker.com/blog/new-docker-desktop-wsl2-backend.

Doyumgaç, İ., Tanhan, A., & Kiymaz, M. S. (2020). Understanding the most important facilitators and barriers for online education during COVID-19 through online photovoice methodology. International Journal of Higher Education, 10(1), 166–190. https://doi.org/10.5430/ijhe.v10n1p166 https://doi.org/10.5430/ijhe.v10n1p166.

Faldi, F., Romadoni, D., & Sumadi, M. T. (2023). The implementation of network server security system using honeypot. JIKO (Jurnal Informatika dan Komputer), 6(2), 163–170. https://doi.org/10.33387/jiko.v6i2.6385

Haerani, S., Parmitasari, R. D. A., Aponno, E. H., & Aunalal, Z. I. (2019). Moderating effects of age on personality, driving behavior towards driving outcomes. International Journal of Human Rights in Healthcare. https://doi.org/10.1108/IJHRH-08-2017-0040.

Lusardi, A., Mitchell, O. S., & Curto, V. (2010). Financial literacy among the young: Evidence and implications. National Bureau of Economic Research, 358–380. Retrieved from https://www.nber.org/papers/w15352.pdf.

Microservices Architecture Definition. https://www.nist.gov.
Microsoft. (2023). Windows Subsystem for Linux Documentation. Diakses dari https://learn.microsoft.com/en-us/windows/wsl.
National Institute of Standards and Technology (NIST). (2022).
Nyabuto, G. (2024). Client-Server Architecture, a Review. International Journal of Advanced Science and Computer Applications. https://ijasca.org/index.php/ijasca/article/view/48.
Sabri, M. F., & MacDonald, M. (2010). Savings Behavior and Financial Problems among College Students: The Role of Financial Literacy in Malaysia Sabri Cross-cultural Communication. Crosscultural Communication. https://doi.org/10.3968/j.ccc.1923670020100603.009.
Zahid, A., & Falgárt, I. (2022). Utilizing Moodle And Apache Openmeetings In A Learning Management System. Researchgate. Available At: