Fault Tolerance Strategy to Increase Moodle Service Reliability

Akhmad Zaini \*, Heri Santoso, Muhammad Priyono T.S.

Departement of Informatics Engineering, Universitas PGRI Kanjuruhan Malang, Malang 65148, East Java, Indonesia

E-mail: zaini@unikama.ac.id

**Abstract.** Moodle is one of online Learning Management System that can be downloaded and installed on our server for free. Moodle is suitable solution for universities to organize online learning. Changing all learning activities from offline to online during a pandemic situation, especially for universities, requires Moodle services to be reliably available. The more students it means the more traffic should be handled by Moodle server. Moodle consists of system, user’s data and a database. To increase Moodle reliability, we can rely on these components. We can use virtualization technology to simplify migration system and recovery process. Distributed file system technology is a best solution to store Moodle user data because it can be replicated into several nodes and support for high availability. Finally, the last component of Moodle is a database, we can replicate the database into several machine to increase availability. At the top of every Moodle components, we need cluster control to make sure all of Moodle components can be accessed. The results show that with the right fault tolerance strategy, all requests can be responded to 100% without error. In addition, this strategy can also increase the response time 12 times faster when compared to a single server.

1. Introduction

Moodle is an open source learning processing system that is widely used in tertiary institutions [1]. Moodle is perfectly designed to support multiple learning models, whether ground, blended or hybrid learning [2]. In addition to endorsing multiple learning models, it is also very conceivable to use Moodle as a MOOC (Massive Open Online Course) platform [3]. The fact that Moodle is licensed as open source making it the preferred platform, particularly in developing countries. While open source licensed, Moodle includes various features that support different learning activities, from file sharing, directories, discussion forums, chat, assignments, wikis, blogs, glossaries, checklists, quizzes, as well as several other features [4]. The development of Moodle as one of the top three most commonly used LMSs has also led to improved accuracy and convenience of access by introducing a voice recognition feature by leveraging Amazon Alexa services [5].

To replace various human activities particularly learning activities in pandemic situations, online media has become an efficient choice. Moodle can therefore be used during a pandemic as an alternative of learning support system [6]. The Moodle framework is built on a web server where all students have access to it. In any case, the server as the cornerstone serving all requests for learning content must always be accessible. Most of the change in learning activities from offline to online induces complete reliance on server availability. If there are issues with the server service, it is certain that all learning activities will be interrupted. Of course, efforts must be made to improve the reliability of server availability to reduce or eliminate the risk.

Various studies have been conducted by scientists in studying and perfecting Moodle, but these studies generally cover a lot about the features and functionality of Moodle [1],[3],[5],[4] and [7]. The enhancement of service efficiency on Moodle servers has not been reviewed by several reports. The Moodle application usually consists of several components, specifically:

* 1. Core System

Core system or system component is a collection of php files stored on a web server. This component contains the core code, subsystems and plugins. This system component is the part that is directly accessed by Moodle users. Most of the Moodle development activities are carried out on this component, therefore the version of the system component can indicate which version of Moodle is being used.

* 1. Assets/Moodle Data

The behaviour of Moodle, which is used by multiple users (teachers and students), triggers the need to distinguish the system location from the learning asset location. In the form of pdf files, presentations, images and videos used while learning, learning assets can be used. It is expected that Moodle maintenance such as updating can be carried out easily by separating the assets and the scheme without impacting the learning assets that have previously been stored. Moodle assets or data are usually stored in a separate folder from the system but can still be accessed by the web server. In addition, asset separation from the system is also carried out to ensure system security, considering that the asset folder usually contains user uploads that have the potential for malware uploads.

* 1. Database

The database section contains all Moodle system data. Moodle uses client-server DBMS as database. At least the database components must be installed separately from the Moodle system to improve the efficiency of this component. It is expected that it would enhance the efficiency of the Moodle system components by splitting the database and Moodle system into separate machines.

Virtualization is a solution that might be used to separate an operating system accordingly it becomes easier to build and operate the operating system [8]. Virtualization technology might generally be split into two parts, specifically kernel-level virtualization and hardware-level virtualization [9]. Kernel-level virtualization such as Linux containers can be used as an effective method for isolating web applications, therefore this form of virtualization is appropriate when used for storing the Moodle system.

LXD and Docker are the most used Linux containers. LXD has advantages in terms of transfer rate and access speed when compared to Docker [10]. Although docker performs better when applied to high packet counts, in the context of an LMS a user will download more content than make requests. Indeed, this requires a stable transfer rate from the server aspect. Apart from being superior in terms of transfer rate and access speed, LXD also allows customization of networking, resources, cpu, memory and disk [11]. Therefore, in this discussion, LXD would be the choice of container that we will use.

Moodle assets could be stored directly on the hard drive or distributed storage as well. If the host where the Moodle device is damaged, stored Moodle assets on that device is directly at risk of causing failure. In this case, since the system and assets are on the same computer, the administrator would not be able to perform recuperation. Moodle assets should be placed in distributed storage and stored redudantly to prevent this risk. The replication and redundancies required to support fault tolerance [12] are not provided by all distributed storage. GlusterFS is a distributed cloud storage option that supports replication and has a decentralized architecture, so it is probably to be used to store assets from Moodle. GlusterFS is also a general distributed file system, in addition to being decentralized, where data can be used directly through a simple mounting point [13].

Moodle service data center conditions of course vary. Unforeseen matters, such as network disruptions, number of visitors and machine failures have an impact on service availability. By implementing the right fault tolerance strategy, it is expected to improve the quality of Moodle's service, both in terms of response success rate and response time.

1. Method

Figure 1 shows that fault tolerance strategy is constructed behind a load balancer technology. HAProxy is a load balancer often used to manage requests from HTTP [14]. The load balancer is responsible for receiving the client's requests and then forwarding them to the actual web server. The web server is packaged in a Linux container, it consists of moodle system. Several web server containers are installed redundantly, therefore HAProxy could automatically redirect requests to other active containers when one container goes down.

|  |
| --- |
|  |
| **Figure 1.** Fault Tolerance Strategy  |

Each container is provided with a shared folder mounted to a Gluster distributed volume. There are already several replicated bricks behind Gluster distributed volume. If the asset on a brick fails, this replication feature could prevent loss of Moodle properties.

Testing was performed by deactivating one of the nodes on the web server container or brick Gluster volume. It proceeds to re-access the Moodle page after both the container and brick are offline to see if the service is still usable even though one of the nodes has failed. Measurements are often made to assess the degree to which output improvements could occur after a fault tolerance technique has been applied, in addition to checking failure conditions. Apache JMeter [15] was used to run all the measurements..

1. Discussion
	1. Testing Setup

The Moodle System is composed of three containers. A dual core CPU and 2 GB of RAM were included in each container. This is an option as the authors aimed at incorporating Moodle in a resource-limited environment. It is expected that the use of limited resources reflects the situation of devices that are commonly used in developed countries [16], where not all educational institutions in developing countries are able to provide high-specification servers.

We are using Apache JMeter to measure the performance of this strategy, because it has been widely used to measure the performance of applications and web frameworks[17]. Moodle's provides standard Apache JMeter test plan scenario that reflect common tasks that are frequently performed by users, including: logging in, opening pages for classes, opening forums, downloading files, and logging out. Another benefit of using the Moodle test plan standard is that the test plan has been configured to resemble the behaviours of human requests, where requests are not necessarily submitted to the server continuously. Indeed, when the users open a tab, often they also perform minor delay reading or perhaps writing replays or posts to a forum.

* 1. Service Failure Test

|  |
| --- |
|  |
| **Figure 2.** Request Summary |

The failure test was carried out by running 30 threads, where each thread contained various request activities starting from logging in, accessing courses, forums to logging out. When the whole thread was running, we deliberately turned off two containers containing Moodle system. In Figure 2, we can see that even though there are two servers that were turned off, the request could still run with a 100 percent of success rate (without errors).

This studies employed the Apdex score to identify how long the time each type of request could be received by the user [18]. The Apdex score uses a Toleration Threshold parameter of 500 ms and a Frustation Threshold of 1500 ms. Figure 3 shows that delay when turning off the Moodle system server causes an increase in response time, thus affecting the Apdex score. However, when viewed as a whole, the majority of page requests were in the good category and above with a cumulative Apdex score of 0.85.

|  |
| --- |
|  |
| **Figure 3.** Apdex Score Overview |

* 1. Additional Benefits

|  |
| --- |
|  |
| **Figure 4.** Response Time Comparation |

If Moodle is only served by one single server, this has the potential to increase request failures, because when the server is experiencing an increase in concurrency, of course there are no other server backups that are able to back up the service, resulting in the server unable to reply to user requests. In Moodle service with one single server, there are still 0.11% failures, while by utilizing Load Balancer, 100% of requests from users can be served. In addition, the overall response time has also increased which can be seen in Figure 4.

|  |  |
| --- | --- |
| Single server | Multiple server |
| **Figure 5.** Response Time Distribution Comparation |

Using a single server results in an increased response time distribution range. Figure 5 shows that single server has longest response time about 12000 ms, while the longest response time using a load balancer could only reach 2000 ms.

1. Conclusion

The implementation of the fault tolerance strategy has the prospect of improving the reliability of Moodle services. Even though the server experiences a decrease in time when responding to user requests, this does not affect all requests. Delay will occur when one or more servers experience disconnection or failure, the load balancer takes time to redirect to another active server and repeat the request process from the old server. In addition to ensuring that all requests can be served properly, the fault tolerance strategy also has the opportunity to increase the Apdex score because the response time distribution range is not too far if you only use one server.

**References**

[1] Yawson D E and Yamoah F A 2020 Understanding satisfaction essentials of E-learning in higher education: A multi-generational cohort perspective *Heliyon* **6** e05519

[2] Badia A, Martín D and Gómez M 2019 Teachers’ Perceptions of the Use of Moodle Activities and Their Learning Impact in Secondary Education *Technol. Knowl. Learn.* **24** 483–99

[3] García-Martín J and García-Sánchez J-N 2020 The effectiveness of four instructional approaches used in a MOOC promoting personal skills for success in life *Rev. Psicodidáctica (English ed.)* **25** 36–44

[4] Fernando W 2020 Moodle quizzes and their usability for formative assessment of academic writing *Assess. Writ.* **46** 100485

[5] Melton M and Fenwick J 2019 Alexa Skill Voice Interface for the Moodle Learning Management System *J. Comput. Sci. Coll.* **35** 26–35

[6] Aucejo E M, French J, Araya M P U and Zafar B 2020 The Impact of COVID-19 on Student Experiences and Expectations: Evidence from a Survey. *J. Public Econ.* 104271

[7] De Medio C, Limongelli C, Sciarrone F and Temperini M 2020 MoodleREC: A recommendation system for creating courses using the moodle e-learning platform *Comput. Human Behav.* **104** 106168

[8] Zhang Q, Liu L, Pu C, Dou Q, Wu L and Zhou W 2018 A Comparative Study of Containers and Virtual Machines in Big Data Environment *IEEE Int. Conf. Cloud Comput. CLOUD* **2018**-**July** 178–85

[9] Fayos-Jordan R, Felici-Castell S, Segura-Garcia J, Lopez-Ballester J and Cobos M 2020 Performance comparison of container orchestration platforms with low cost devices in the fog, assisting Internet of Things applications *J. Netw. Comput. Appl.* **169** 102788

[10] Auliya Y A, Nurdinsyah Y and Wulandari D A R 2019 Performance Comparison of Docker and LXD with ApacheBench *J. Phys. Conf. Ser.* **1211**

[11] Flauzac O, Mauhourat F and Nolot F 2020 A review of native container security for running applications *Procedia Comput. Sci.* **175** 157–64

[12] Acquaviva L, Bellavista P, Corradi A, Foschini L, Gioia L, Carlo P and Picone M 2018 Cloud Distributed File Systems : a Benchmark of *2018 IEEE Glob. Commun. Conf.* 1–6

[13] Sim H, Khan A, Vazhkudai S S, Lim S H, Butt A R and Kim Y 2020 An Integrated Indexing and Search Service for Distributed File Systems *IEEE Trans. Parallel Distrib. Syst.* **31** 2375–91

[14] Zhao Y, Li S, Hu S, Wang H, Yao S, Shao H and Abdelzaher T 2016 An experimental evaluation of datacenter workloads on low-power embedded micro servers *Proc. VLDB Endow.* **9** 696–707

[15] McMillan C, Poshyvanyk D, Grechanik M, Xie Q and Fu C 2013 Portfolio: Searching for relevant functions and their usages in millions of lines of code *ACM Trans. Softw. Eng. Methodol.* **22**

[16] Ali M A, Alam K, Taylor B and Rafiq S 2020 Does ICT maturity catalyse economic development? Evidence from a panel data estimation approach in OECD countries *Econ. Anal. Policy* **68** 163–74

[17] Sunardi A and Suharjito 2019 MVC architecture: A comparative study between laravel framework and slim framework in freelancer project monitoring system web based *Procedia Comput. Sci.* **157** 134–41

[18] Kusuma M, Widyawan and Ferdiana R 2017 Performance comparison of caching strategy on wordpress multisite *Proceeding - 2017 3rd Int. Conf. Sci. Technol. ICST 2017* 176–81