



REST Architecture Optimization in Cloud Computing Ecosystem to Support E-Learning Platform

Faisal Faisal, Gede Putra Kusuma

Abstract: This study will present an application design process in the style of Representational State Transfer (REST) architecture to support the E-Learning platform in the cloud computing ecosystem. An application optimization process will be presented to provide E-Learning applications for schools, faculties or universities that in most cases need manual deployment and require more time for server provisioning. This process is optimized by providing application solutions that can provide speed of provisioning. The core system used Kubernetes containerization technology to provide scalability of growing E-Learning tenants. Evaluation of the core system architecture uses the Architecture Trade-off Analysis Method (ATAM) to evaluate aspect of performance and scalability as quality attributes. From the experimental results, the process of making new tenants for schools requires an average time of around 173.4 seconds. This meets the expectations of the set time limit of 5 minutes. The results of stress tests for 250 concurrent users show that the system has availability above 98%. Thus, education stakeholders such as schools and universities, no longer need to provide expensive e-learning infrastructure in the form of hardware or manpower to deploy the e-learning application on premise. In the future, this solution will provide a scalable E-Learning system that can spread at scale on the cloud computing ecosystem and support a Software as a Service solution in educational technology.

Keywords: Cloud Computing, E-Learning, Moodle, Kubernetes, REST.

I. INTRODUCTION

Penetration of the number of internet users in Indonesia is growing rapidly. Based on statistics the number of internet users in Indonesia is the fifth largest in the world with a total of 171.26 million or around 63.5% of the total population [1]. With the internet, the new era of information needs is becoming wider and our learning needs are increasing, so that types of distance learning such as Digital Learning or E-Learning that can be accessed through mobile devices such

as laptops, tablets and mobile phones are needed to achieve learning anytime and anywhere.

Distance learning and online education will have a strategic role for the expansion and equitable distribution of education that is expected for all people in remote areas of Indonesia to have the same quality of education. This learning method will also become more flexible, well-distributed, on time, and on demand. Increasing access to education in Indonesia must keep abreast of the times which have now entered the era of the industrial revolution 4.0 [2].

Nowadays web technology is quite rapidly developing where web service is used as a technology base for services. Representational State Transfer (REST) is an SOA design for hypermedia or distributed systems [3]. REST is an architectural style for creating web-based SOA and is often called RESTful web service. This has become the industry standard in large-scale SOA-based software architecture [4].

The application of REST has been adopted by one of the currently popular Learning Management System (LMS) solutions, Moodle (Modular Object-Oriented Dynamic Learning Environment). Statistically, Moodle has been implemented on 92,971 registered sites in 229 countries which consists of 17,982,765 subject matter with a total user of 148,620,029[5].

The use of Representational State Transfer (REST) as an architectural style to integrate services and applications brings several benefits, but also poses challenges and risks. The use of Architecture Trade-off Analysis Method (ATAM) can help evaluators evaluate REST-based architecture in order to identify trade-offs and risks to overcome the requirements of quality attributes such as security, reliability and performance [6].

Related to the above studies, we identified research gaps related to how to optimize REST-based architecture in the cloud computing ecosystem to support the E-Learning platform, especially with Moodle LMS. This research will use the SOA principles to centralize Moodle LMS architecture design in cloud computing ecosystems. REST design pattern[3], REST constraints, SOA design patterns[7] are utilized to design this architecture. Architecture Trade-off Analysis Method(ATAM) will be carried out to evaluate trade-offs and risks in the selection of architectural designs [8]. The evaluation process is scenario-based and focuses on software quality attributes [6].

Some aspects that will be the focus of the optimization include:



Revised Manuscript Received on December 30, 2019.

* Correspondence Author

Faisal Faisal, Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480.

Gede Putra Kusuma*, Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an [open access](#) article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

1. Performance aspect, how to handle the number of users accessing LMS simultaneously (concurrent). How to maintain a reliable system for handling learning activities such as Quiz, Homework, Forums and others.
2. Scalability aspect, how to prepare all changes related to the size of the existing cloud system. Provision of cloud infrastructure that is over-provisioning (excessive resource but low utilization) which has an effect on waste of costs, or under-provisioning (high utilization exceeds the specified resource) which has an effect on user access being slow.

The benefits to be gained from this research are answering challenges related to performance and scalability aspects of the Moodle LMS system so that it can be used as a reference for implementing a multi-tenant E-Learning system to serve relevant stakeholders such as universities, schools, course institutions and companies.

II. RELATED WORKS AND THEORIES

A. E-Learning & Cloud Computing

Research related to cloud computing-based E-Learning systems has been conducted for the development of E-Learning through a mobile learning model with cloud computing[9][10]. The cloud model consists of several service components hosting an E-Learning system in the cloud, including:

- Data storage: used to store data related to learning material in the cloud
- Memory management: memory management is very important for every distribution server in the cloud to support the computing process
- Process layer: this layer is the core of computing in the cloud ecosystem where each virtual machine in the cloud needs a processor to process every request made by the user on the client-side
- Security: services that run in the cloud such as E-Learning applications require security-related concerns, for example, regarding authentication and authorization. Also security-related applications such as how to avoid SQL injection attacks
- Firewalls: firewall setting in the cloud is very important especially related to security for the server ecosystem for example to avoid Denial of Service (DDoS) attacks
- Network access: at this layer the cloud requires network access settings such as load balancer so that the access process from the client-side is smooth and uninterrupted even though it is accessed by thousands and even millions of users.

In the client model, computational offloading on the cloud is sent to mobile devices on the user / client-side. Users can choose the desired learning topic, which can be in the form of learning topics in the form of text, images and videos. The process is self-help / self-service and the user can download the material directly to his mobile device.

The next e-learning model is a model called High Performance Computing (HPC) for Mobile Distance Learning [11]. This model presents a mobile learning system whose infrastructure is inside the university's private cloud. At the lower level, there is a cluster network interconnection that connects many computer nodes (slaves) to serve the computing processing of a large number of users where these nodes are connected to the master computer in the form of servers that are active or passive. This server utilizes Storage

Area Networks for application storage and educational content data. This active and passive server is connected with a load balancer that functions to divide the burden of mobile learning access traffic from users so that there is no delay on the user's side. In addition, this load balancer regulates the mobile learning system in the event of a failure on one of the servers. Thus, there will be no disruption to the service.

As for the application side, this mobile learning model specifically uses the open-source Learning Management System (LMS) application, moodle.org, which provides an API for connectivity of E-Learning content between universities. In this model, the entire system above is called High Performance Computing (HPC). High-performance Computing (HPC) is actually the terminology of using computer clusters or super computers and parallel processing techniques to solve complex computing problems. HPC is specifically used for research activities through computer modeling, simulation and analysis.

HPC is connected to the university's Local Area Network (LAN) and to the internet. From this e-learning model, users from outside the university can access other university mobile learning content connected via the Moodle platform API.

The trends related to containerization (Docker) technology have made it easier for educational stakeholders to create cloud-based data science teaching media. The approach is to use sophisticated cyber infrastructure to teach multi-institutional bootcamps throughout the day in machine learning lectures held at the University of California at San Francisco (UCSF). This makes it easier for instructors to prepare course infrastructure without the need for extraordinary technical knowledge[12].

B. E-Learning and SOA

Research has been conducted which aims to broaden the core idea behind the Virtual Learning Environment (VLE) tool, which is Moodle, which dominates academic institutions. The contribution to be addressed is to build VLE with a web service concept approach, namely SOA and related techniques. The basic contribution of the proposed study is to show that VLE can be made available as a service that can be published, discovered and arranged as perceived in the SOC (Service-Oriented Computing) paradigm [13].

Other research attempts to develop further architectural designs for virtual resources in computer science learning by using SOA proposes the development of Moodle LMS architecture design using SOA principles [14]. In this journal the researcher tries to utilize the web service features available in LMS for connectivity with external applications such as virtual lab / virtual compilers. With the use of SOA, this can overcome the limitations of LMS in supporting practicum activities in the field of computer science.

The next literature review is related to the development of mobile application architecture for education using SOA [15]. In this research, the background is related to the challenges of the education world which so far have used the E-Learning platform but have not touched the mobile ecosystem so that students are more interactive in learning. Educational institutions are still constrained by the security aspects related to the implementation of the E-Learning system on mobile devices.

The formulation of this research problem is how to develop a mobile application system with the principle of SOA that can be used safely and connected to an existing E-Learning system.

C. Representational State Transfer (REST)

REST is becoming popular as an architectural solution in building web services. Research has been conducted related to evaluating the differences between REST and SOAP as web services [16]. REST service applications have increased based on a survey conducted in 2007-2010. The REST application grew in popularity in 2011 and shows that REST is the right architecture for the web. REST also faces challenges in security, design standards, and solving problems for companies. REST started to gain popularity since 2008 and turned into a stronger and holistic framework [17].

REST has been described as six REST constraints, each reflecting one or many software quality attributes [3]. It has a symbol and can be described as follows:

$$\text{REST} = (C - S, S, \$, U, L, \text{CoD}) \quad (1)$$

- Client-Server (C-S)

Client-server architecture implements separation of concerns through the role of client and server with specific responsibilities that interact with each other. The server provides services to the client, and the client provides the user interface to access services. The client-server architecture allows REST applications to be highly scalable and allows client and server development to occur independently. The client provides the user with a simple and fast interface without affecting the server, while the server can manipulate larger data sets because it is freed from having to carry out client responsibilities.

- Stateless (S)

In the REST architecture, interactions must be stateless. This means that the server does not store information about the client's current state or previous requests made by the client. The server only observes that the client was there when the request was made. All information needed for the server to understand and respond to requests comes through requests, and requests are contained with each other. This improves web service performance, because the server does not have to remember the client's current status in the system. But the trade-off is that this imposes significant limits on the way clients and servers communicate. Each time the client sends a request to the server, it must provide and store information about the current state.

- Cache (\$)

This constraint means that the client can store a local copy (cache) of the server's response to be used later, depending on what information the server adds to the response to label it as cacheable or non-cacheable. This can help improve performance by reducing the number of requests for the same resource and helps ensure that the client does not store excessive or useless data.

- Uniform Interface (U)

The point is there must be a uniform interface for communication between client and server. This constraint has a certain impact. The first is that there are special methods that can be understood. REST uses common HTTP

methods namely GET, PUT, POST, and DELETE, to communicate the different actions that clients want to do on resources. The second is that resources must be identified in the request with certain Uniform Resource Identifiers (URIs) that will represent uniform resources. Responses have specific headings, and resources are written in three specific ways: XML, JSON, or simple text.

- Layered System (L)

REST is a layered system. REST can consist of several layers of software or hardware architecture that can be called by the client and server. These layers can be used to improve performance, translate messages, and manage traffic.



Fig. 1. Layered system in REST architectural style

This helps improve REST Web service reusability because layers can be added and removed based on the services needed by the client.

- Code-on-Demand (CoD)

Code-on-Demand (CoD) is the only optional constraint in REST. This allows the client to increase its flexibility because it is actually the server that decides how certain things are done. For example, with Code-On-Demand, clients can download JavaScript, Java Applets or even Flash applications to encrypt communications so that the server is not aware of any routine / encryption keys used in this process. However, using CoD reduces visibility, which is why this constraint is optional. Also, not every API requires this kind of flexibility. Interoperability also decreases because the code must be compatible with the target consumer. Security is also a concern because it can be infiltrated with malicious code.

These constraints make REST a flexible and high-performance architectural style for building service-oriented systems based on web standards. REST provides benefits including high scalability, reusability, and loose coupling that enables it to meet the needs of modern applications with millions of users.

D. SOA Design Pattern

Design patterns are proven design solutions to common problems in software design. Problems are documented in a standard format and in a consistent manner[7]. Design patterns are used to design architectures based on problem cases because they provide field-tested solutions so that design patterns can speed up the development process. In the SOA context, there are many design pattern categories that discuss different aspects of SOA-based systems including: service messaging patterns, service implementation patterns, service security patterns, composition implementation patterns, and so on. Erl has established eighty-five design pattern profiles for SOA. There are also seven new design patterns inspired by REST to solve problems using REST capabilities [18].

The following are examples of illustrations related to design patterns in REST architecture, namely the Uniform Contract type pattern.



Table- I: Uniform Contract CRUD

CRUD	REST	
CREATE	P OST	Create resource
READ	GET	Retrieve current state of resource
UPDATE	PUT	Initialize or update the state of a resource on the given URI
DELETE	DELETE	Removing a resource, after which the URI is no longer valid

In detail the difference is as follows:

- GET is a read-only process. This can be repeated without affecting the state of the resource and can be cached. Can read many times with the same results. An HTTP GET should never be used to change data.
- POST is a read-write process and can change resource status and cause side effects on the
- PUT is an operation used to update the resource state. If the PUT operation is performed N times, the first request will update the resource, then the rest i.e. the N-1 request will only overwrite the same resource state again and again which effectively does not change anything (idempotent).
- DELETE is an operation to delete a resource. When N has a similar DELETE request, the first request will delete the resource and the response is 200 (OK) or 204 (no content). Other N-1 requests will return 404 (not found). Obviously, the response is different from the first request, but there is no change in status for any resource on the server-side (idempotent) because the original resource has been deleted.

E. Architecture Trade-off Analysis Method (ATAM)

The purpose of ATAM is to assess the consequences of architectural decisions based on quality attribute requirements. As an example ATAM is used to evaluate remote temperature sensors[19]. This framework helps determine the useful characteristics of each architectural option. ATAM helps determine each location of the architectural trade-off point, and it makes us understand the limits of each option. This information is useful for making action plans for evaluating, starting new iterations of methods, and modifying architectures based on evaluations. ATAM was made to make possible and rational choices between software architecture options. Not only that, it also tried to improve the quality of architecture in each method iteration.

To evaluate quality attributes and understand the exchange between architectures, scenario-based tests must be carried out. A single testing scenario must be able to reflect on what software quality attributes must be achieved. Other research organizes the ATAM scenario report into a general scenario consisting of qualities that are linked to achieve. By mapping certain scenarios into general scenarios, it can show the quality attributes that are of concern in every software development project[20].

The use of ATAM is also used to evaluate software architectures for avionics system product lines [8]. This

experiment came to the conclusion that ATAM can increase stakeholder awareness. Architectural evaluations carried out before the code is developed can resolve risks that arise before they are too expensive to fix. Evaluating architecture before or when a system is developed can also be effective in dealing with future disasters. It also helps evaluate software architecture engineers in making software.

Other studies present a qualitative analysis of the security aspects of Web-based applications that utilize Service Oriented Architecture (SOA)[21]. Architectural solutions that address security requirements are examined and compared with other quality attributes that are relevant to web-based systems. More specifically, a trade-off analysis based on ATAM was conducted to show the correlation between security and the quality of other systems associated with successful SOA selection. The optimal architectural solution not only meets the security requirements of a web-based system but also meets other quality attributes such as performance, availability, usability, modification capabilities, etc.

III. METHODOLOGY

The initial architecture of the Moodle LMS system that is generally applied in the field is monolithic, that is, all the frameworks of both the resource front end, backend, database and file storage system are all on one server. This poses a challenge for educational stakeholders where they have to provide infrastructure investment in the beginning in the form of expensive and non-scalable servers. This research seeks to solve these challenges namely how to provide an E-Learning system that is affordable, scalable and has fast deployment time to be applied in many different educational institutions. So the main target is to create an E-Learning system that has a Software as a Service (SaaS) model in the cloud computing ecosystem with good performance and scalability. Scalability is a major concern in choosing the right architecture for this E-learning system. From the problems above, there is an urgency to research new architectures that can be measured for use in these educational technologies. Architecture needs to be evaluated to achieve certain standards in scalability and other related quality attributes. Evaluation output is used to improve architecture Fig.2 illustrates the research activities to be undertaken to evaluate and optimize the Moodle REST LMS architecture in the cloud computing ecosystem. The Literature review was discussed in the previous chapter. This chapter discusses the design process for early architecture deployments in Moodle LMS in the cloud computing ecosystem, REST constraints, and SOA design patterns.

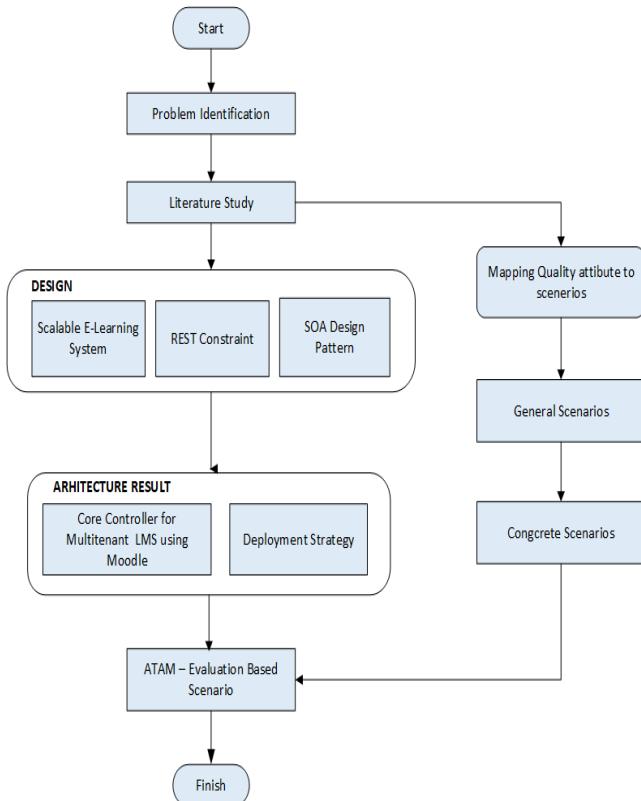


Fig. 2. Research Framework

This research focuses on the design of REST architecture in Moodle LMS in the cloud computing ecosystem. The process is in the form of designing, evaluating, and modifying architectural strategies to deal with the challenges of how to provide an E-Learning system that is affordable, scalable and has fast deployment time to be implemented in many different educational institutions.

To design the initial architecture, Moodle LMS implementation will be used in the cloud computing ecosystem on the Google Cloud Platform (GCP). Then the problem is formulated and mapped to REST inspired SOA design pattern [18]. This study also designed architecture by following the REST constraints [3]. The result of the design is the initial E-Learning system architecture and how the deployment strategy is. The ATAM evaluation used in this study is specific to the REST architecture[6][22]. This evaluation uses scenarios related to quality attributes obtained from the literature review. This is used to examine REST standards, REST design questions, ATAM scenario-based testing, and trade-offs in architectural decisions.

IV. PROPOSED METHOD

The initial proposed architectural design is multi-tenant architecture. The RESTful API (backend) server can serve many tenants, which means many schools in several different institutions. Web services will be centered to serve all E-Learning System clients. Fig. 3 will provide an initial review of the proposed architectural design.

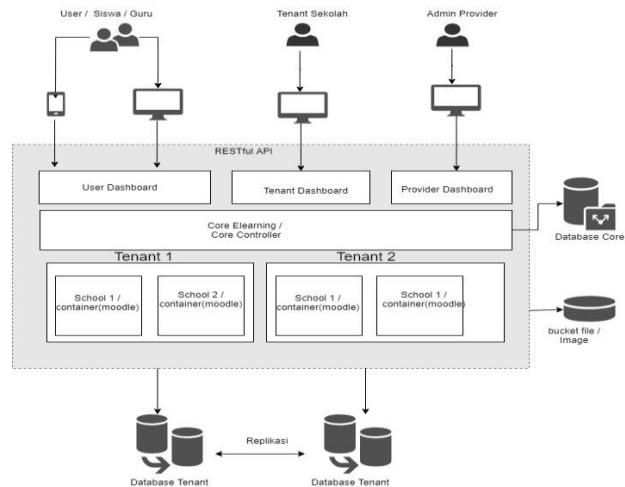


Fig. 3. Multi-tenant E-Learning Architecture design

Fig.3 illustrates clearly how the initial architecture of the LMS architecture to handle multitenancy. Users such as teachers and students will access the E-Learning System through web and mobile applications. There is also an application for the dashboard of partners or tenants who can monitor Moodle LMS containers that are deployed at each institution or foundation. The design of this multi-tenant E-Learning system will apply the MVC method (Model, View, Controller) which contains two main parts namely core and tenant. The core controller will handle shared business logic (models) that are used together like logins and user management.

The core controller determines the user who is a tenant in each request by using its own database, which mostly contains general user and tenant data. The core controller is responsible for managing requests and forwarding them to tenant controllers, models, and databases. For the user login and authorization process to the core controller, this web service will use a special RESTful API for logging in.

The main concern in this initial architecture was to keep the web service engine stateless. Stateless means the web service engine may not maintain or save the state of the client. Stateless is the first approach to dealing with scalability, and token authorization is one of them. Stateless is also one of the main REST foundations. The deployment strategy that will be used in this study is illustrated in Fig.4.

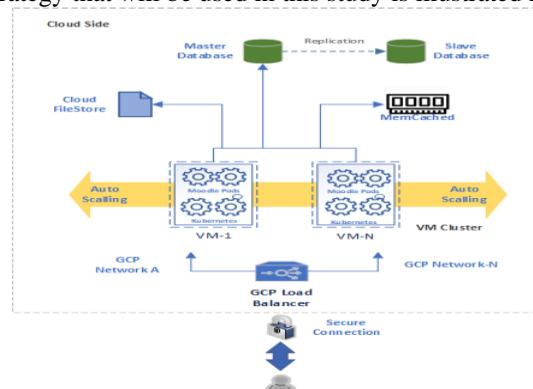


Fig. 4. Overview of E-Learning system deployment strategies in the Cloud

The strategy will be carried out by inserting the Moodle LMS system in a Moodle Pod container containing the Moodle LMS application and web server. This Moodle Pod acts as a web server engine inside a Virtual Machine (VM). To keep these Moodle pods stateless, every storage of supporting files that are dynamic and growing like documents, images and databases are located outside the Pod. In this case, we will use cloud services from GCP for storing files in the form of Cloud FileStore while for the service database named CloudSQL. This Moodle pod is orchestrated by Kubernetes engine [23] which will manage cluster pods in the VM with scale up and scale out features.

The number of concurrent users accessing the E-Learning system can result in the use of CPU and memory resources in the Moodle pod to be overloaded, so to avoid this we can make settings in the Kubernetes engine if the CPU resource in the Pod is above 50% then the Kubernetes will scale up pod by creating a new pod so that it can serve the number of

requests from users accessing the E-Learning system. Conversely, if at the pod level the average CPU and memory usage are below the specified limit for example below 50%, then we can configure it so that the system scales down. This also applies at the VM cluster level, if the RAM and CPU resources in the cluster reach a predetermined limit, then the cluster will automatically scale out (horizontal scaling) by creating a new VM or vice versa scaling down by removing excess VMs. This is the process of what is called auto-scaling so that availability is maintained well and also provides efficiency in costs.

From the proposed architectural design in Fig. 3 and Fig. 4, ATAM evaluation plan will focus on the quality attribute Scalability (S) and Performance (P). Table-II shows evaluation scenario based on ATAM for this REST based application.

Table- II:ATAM General Scenarios of Quality Attributes

Quality Attribut (QA)	General Scenario QA	Concrete Scenarios
Scalability (SC)	SC1 – Tenant LMS sites and API services are made easily without any configuration on the server.	Tenant is created and configured through the admin panel without requiring technical configuration related to the server. After a tenant is created, the system automatically distributes the tenant database, website (client-side logic) and tenant endpoint API (server-side logic).
	SC2 - Server-side logic stores data such as documents, images, videos and other learning data in separate storage outside the web server engine to support horizontal scalability	Learning files data such as documents and videos are stored in the GCP CloudFilestore. The database is located in GCP CloudSQL.
	SC3 – REST applications can easily scale up and scale out horizontally as the number of tenants increases	School-level APIs in the 'A' tenant are visited with a number of users simultaneously in one minute. The number of concurrent users increases with each iteration and stops until it reaches 250 concurrent users. Kubernetes Pod will be duplicated if the total CPU Pod usage reaches 50%. If the total size of the Cluster is not enough to duplicate the Pod, Kubernetes Cluster will scale out to increase computing power.
Performance (P)	P1 – Tenant LMS creation can be done easily, quickly and does not slow down the whole system	Tenant is created and configured through the admin panel without requiring technical configuration on the server. New tenants are made and deployed in less than 5 minutes.
	P2 – The system is visited by a number of tenants at measured time and does not reduce user experiences.	25 concurrent users accessing a number of tenants randomly in one minute. The number of tenants increases with each repetition and stops until it reaches 30 tenants. System availability remains 98% and the average response time is under 2 seconds.
	P3 – The system is visited by a number of concurrent requests at measured times and does not reduce user experiences	Tenant LMS APIs are visited with a number of concurrent users in one minute. The number of concurrent users increases with each iteration and stops until it reaches 250 concurrent users. System availability remains 98% and the average response time is under 2 seconds.

V. RESULTS AND DISCUSSION

In the process of implementing the Multitenant LMS system, the first step that has to be made is the API for the core controller. This API serves as the core to regulate the process of creating a list of tenants along with the process of providing LMS for each existing tenant.

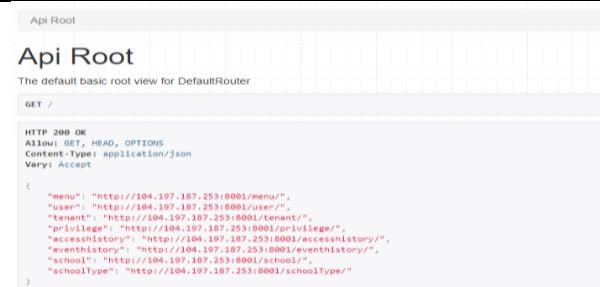


Fig. 5. API for LMS Core controller

The API for this LMS core will relate to the UI as a front end for multitenant LMS system administration. The architecture for the deployment strategy is to use the Kubernetes (K8S) platform which is used as a management application container & multitenancy [24]. The schema is shown in Fig. 6.

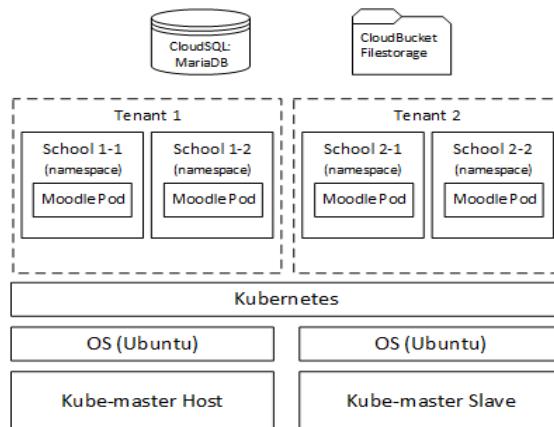


Fig. 6. Real scenario deployment of Core LMS Controller

The above architecture runs on the Google Cloud Platform (GCP) ecosystem in the Kubernetes cluster with two Ubuntu Virtual Machines (VMs), each with a VM specs with an Intel Cascade Lake CPU Platform with 2 vCPUs and 8GB RAM. Thus, the total cluster has 4 vCPU and 16 GB RAM as shown in Fig. 7. Meanwhile, the core LMS using Kubernetes to provide scalable multi-tenant LMS is described in Fig. 8. It consists of master and node components.

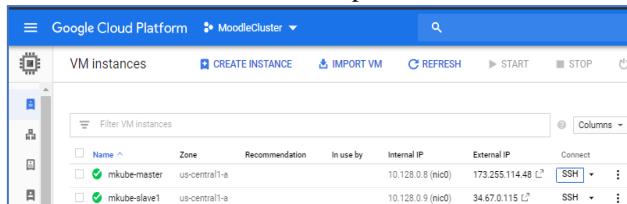


Fig. 7. VM deployment on GCP for provisioning Kubernetes Cluster

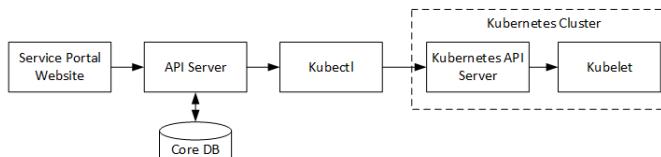


Fig. 8. Core LMS using Kubernetes to Provide Scalable Multi-tenant LMS

A. Master Component

The master component provides the control plane for the cluster. This component plays a role in the global retrieval process of the cluster (for example, the schedule mechanism), and plays a role in the process of detecting and responding to events that take place in the cluster (for example, scheduling a new pod if the number of replicas existing on the replication controller is not met).

The master component can be run on any machine in the cluster. Even so, to facilitate the existing process, the initial initiation script that is run usually starts the master component on the same machine and does not run containers for users on this machine. Kube-apiserver is the component in the master that exposes the Kubernetes API and acts as the

front-end of the Kubernetes control plane. This component is designed to be scaled horizontally. At this master node, there is Kubectl which is used to control the cluster.

B. Node Component

This component exists at each node, its function is to perform maintenance of the pod and provide a runtime environment for Kubernetes. Kubelet is one of the node components as the agent that is run on each node in the cluster and has task ensuring the container is run inside the pod. The other node component is Kube-proxy that helps abstraction Kubernetes service to do its work. This happens by maintaining network rules and forwarding the connection to a host. The dashboard of Core LMS as a service portal website is shown as Fig. 9. In this dashboard, admin can see the summary of school by tenant and school by type.



Fig. 9. Views of core LMS dashboard

In this dashboard also contains a list of tenants as shown in Fig. 10.

List of LMS tenants						
Code	Name	City	Contact	Mobile	Email	Active Date
YRPK-0011	Yayasan RPK Penabur 1	Jakarta 1	Rudi Gunawan	0813873818731	rudi.gunawan@yprk-penabur1.ac.id	2019-10-28
YRAS-001	Yayasan Residen	Bandung	Andi Hisyam	081295403004	andi.hisyam@yayasanresiden.org	2019-10-05
YALZ-001	Yayasan Al Amr	Jakarta Selatan	Iman Abdurrahman	085093203323	imanabdullah@yayasanal-amr.com	2019-10-03
YAUITY	Yayasan Audley	Tangerang Selatan	Lili Kartika	081220000000	Lili.kartika@audley.org	2019-10-18

Fig. 10. List of LMS tenants

Next, the user interface for adding school LMS is shown in Fig. 11.

Fig. 11. Form for Adding school LMS

The logic process of school LMS creation is described as Fig. 12.

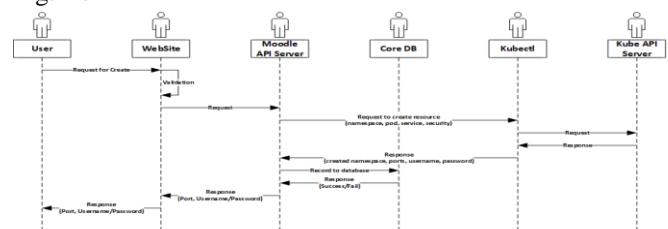


Fig. 12. Process of LMS pod deployment between the Core API server and Kubernetes

Every LMS creation process needs activation time and Fig.13 describes pop up notification for this process.



Fig. 13. Notification of School LMS activation

Here some result of activation time of school LMS of each tenant with interval time 60 second on every attempt:

Table- III: Activation time of Tenant LMS via Core Controller

LMS Creation Attempt	Activation Time (s)
LMS-1	171
LMS-2	168
LMS-3	175
LMS-4	180
LMS-5	173

For five attempts of LMS creation, the average of tenant LMS activation time is about 173.4 second.

Next, we evaluate the core of the LMS API for different concurrent users. Here are the measurement result of the stress test performance report using Siege Load Testing [25][26] :

Table- IV: Load testing result of Core LMS API

Concurrent User	Availability (%)	Data transferred (MB)	Response Time (s)
25	100	13,63	0,02
50	100	20,97	0,1
100	100	22,41	0,42
250	99,92	22,4	0,65

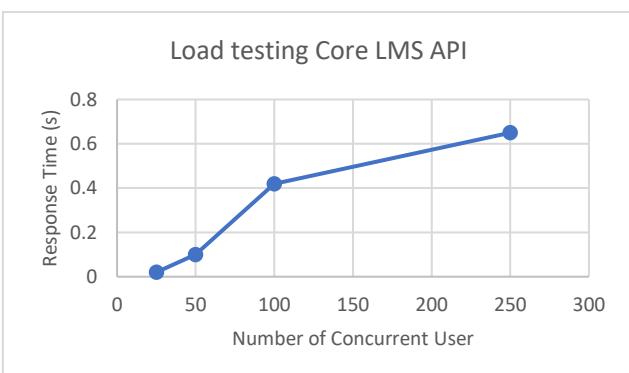


Fig. 14. Load testing Graphic of Core LMS API

From the test results above, we try to do testing for the Core LMS API, which serves as the core API for tenant management. Availability represents the percentage of connections that the server manages successfully. It is the result of socket faults divided by the total amount of connection efforts (including timeouts). Data transferred represent the sum of data transmitted to each simulated user, including the header information as well as content. Response time represent the average time taken to respond to the requests of each simulated user. From the above test, the

LMS results are quite responsive in handling the tenant school management process with a response time of 0.65 seconds for the number of concurrent users 250.

Next is a test for one school LMS (school level API) with a number of concurrent users that vary from 25, 50, 100 and 250. The results are as follows:

Table- V: Load Testing result of One School LMS

Concurrent User	Availability (%)	Data transferred (MB)	Response Time (s)
25	100	274,53	0,18
50	100	277,63	0,36
100	100	277,33	0,7
250	98,28	272,77	1,49

Load Testing of School LMS Pod

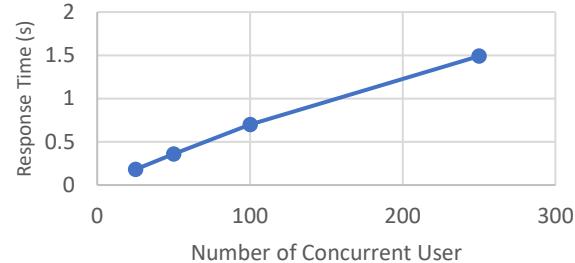


Fig. 15. Load Testing Graphic of One School LMS

From the test results above, it can be seen that the backend server gives a pretty good response time where the availability is 100% for concurrent users from 25, 50 and 100. But it decreases to 98.28% in concurrent user 250. Likewise, the response time is directly proportional to the number of concurrent users where it ranges from 1.49 seconds for 250 concurrent users.

Next is testing for the deployment process in Google Kubernetes Engine (GKE) with pod replication settings at 50% CPU level.

Table- VI: Load testing result of school LMS using GKE Pod Replication

Concurrent User	Availability (%)	Data transferred (MB)	Response Time (s)	VCPUs / RAM	Num Pods (Start/End)
25	98,03	47,39	0,79	3/11.25	1/8
50	100	23,47	2,45	3/11.25	10/10
100	96,48	32,46	3,26	3/11.25	10/10
250	97,24	26,85	5,05	3/11.25	10/10

Load Testing School LMS Pod with replication

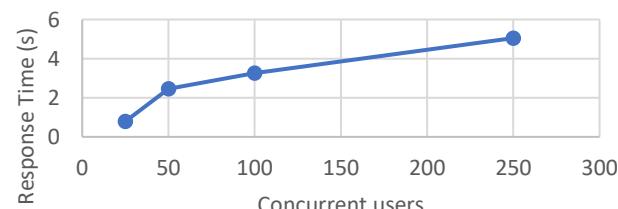


Fig. 16. Moodle LMS Deployment using pod replication via Google Kubernetes Engine (GKE)

The result is seen an increase in latency for this load testing process. Where the pod will automatically replicate as the number of concurrent users increases and there is a decrease in availability which is due to the cluster clustering executing the pod replication process. Response time for 250 concurrent users decreases to 5.05 seconds.

ATAM Output

Architectural evaluation produces ATAM output in the form of analytical results from the congress scenario. Architectural analysis is the result of evaluating concrete scenarios. Based on concrete scenarios, it is possible to analyze the architecture, risks, and tradeoffs of various quality attributes. Table- VII and VIII shows the analytical results of concrete scenarios. It is possible to analyze architecture not only based on its original quality attributes but also other quality attributes.

Table- VII: ATAM Output for Tenant Creation Scenario

Scenario Summary	After a new tenant is created, the system automatically creates a tenant database, tenant LMS (client-side logic) and tenant endpoint API (server-side logic). New tenants are made and used in less than 5 minutes.
Business Goal(s)	Tenant placement is easy and does not reduce user experience
Quality Attributes	Scalability (SC1), Performance (P1)
Architectural Analysis	<ul style="list-style-type: none"> - Tenant management is managed through the admin panel. - Tenant removal time depends on tenant data
Risk	Anyone who has valid access to the system admin can use and delete tenants.
Tradeoff	Tenants can be made easily and the placement of the LMS tenant system can be easily done. This improves performance and scalability rather than a single tenant architecture. If this process is not completed with two-step verification, anyone who has access can create and delete tenants and reduce security.

Table- VIII: ATAM Output for System Scalability

Scenario Summary	GET requests are sent to the learning material API (course) in tenant A. Learning material data (i.e. pdf, docx, pptx files) comes from other storage outside the web server. The system will not store learning data in the same storage as the server.
Business Goal(s)	Make a scalable architecture system
Quality Attributes	Scalability (SC2)
Architectural Analysis	The learning material is configured separately from the Google Cloud Storage service (Cloud FileStore). Likewise, the database is configured on the Google Cloud SQL service.
Risk	<ul style="list-style-type: none"> - Can increase rental cost of cloud storage. - Proper configuration is very important so sensitive data cannot be accessed by the public.
Tradeoff	By separating storage media, this can increase scalability and performance because the server does not clone media data when scaling horizontally (scale out). However, this adds to the cost of renting cloud storage and this must be configured properly to avoid data leakage.

VI. CONCLUSION AND FUTURE WORK

From the LMS core deployment architecture for school LMS provisioning, it can be concluded that application

containerization in the Kubernetes cluster can effectively simplify and optimize the REST core LMS architecture in the cloud computing ecosystem. With this LMS providers can effectively provide multitenant LMS solutions that lead to SaaS. For the scalability aspect, the provider only needs to add the Kubernetes node so that it can add many LMS tenants horizontally.

From the above results, it can be seen that the Core LMS API is responsive enough for the tenant management process with a response of 0.65 seconds for 250 concurrent users. The LMS deployment process for each school is around 173.4 seconds. The maximum response time for a Pod LMS school for 250 concurrent users is 1.49 seconds. The response time for LMS pod testing with replication mode is seen to decrease to around 5.05 seconds with data transfer also decreasing compared to testing on a single LMS pod. This large response time is due to the monolithic Moodle LMS architecture so that the pod replication process becomes heavier in the GKE cluster. Therefore, for future work, it is suggested that the replication process runs well, so it is recommended that the Moodle architecture can be converted to Microservices. Overall ATAM output in tradeoff and risk can be used to enhance the architecture in the next iteration.

REFERENCES

1. "Internet World Stats," 2019. [Online]. Available: <https://www.internetworldstats.com/asia.htm#id>. [Accessed: 01-Dec-2019].
2. M. Nasir, "RistikDikti," Media Pustakawan, vol. 8, pp. 1–56, 2018.
3. R. Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, 2000.
4. R. T. Fielding et al., "Reflections on the REST architectural style and 'principled design of the modern web architecture' (impact paper award)," 2017, pp. 4–14.
5. "Moodle Statistics," 2019. [Online]. Available: <https://moodle.net/stats/>. [Accessed: 09-Apr-2019].
6. B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, "Evaluating a Representational State Transfer (REST) architecture: What is the impact of REST in my architecture?," in Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014, 2014, pp. 105–114.
7. T. Erl, SOA Design Patterns. Prentice Hall, 2009.
8. M. Barbacci, P. Clements, A. Lattanzi, L. Northrop, and W. Wood, "Using the Architecture Tradeoff Analysis Method SM (ATAM SM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study," 2003.
9. N. M. Rao, C. Sasidhar, and V. S. Kumar, "Cloud Computing Through Mobile-Learning," Int. J. Adv. Comput. Sci. Appl., vol. 1, no. 6, 2010.
10. Debasish De, Mobile Cloud Computing: Architectures, Algorithms and Applications. CRC Press, 2016.
11. S. Kitanov and D. Davey, "Mobile Learning in Mobile Cloud Computing Environment," Int. Trans. Syst. Sci. Appl., vol. 8, no. December, pp. 27–39, 2012.
12. C. Holdgraf, A. Culich, A. Rokem, F. Deniz, M. Alegro, and D. Ushizima, "Portable learning environments for hands-on computational instruction using container-and cloud-based technology to teach data science," ACM Int. Conf. Proceeding Ser., vol. Part F1287, 2017.
13. A. Al-Ajlan and H. Zedan, "E-Learning (MOODLE) Based on Service Oriented Architecture," in Proceeding of the EADTU's 20th Anniversary Conference, 2007.
14. Z. Al-Khanjari, Y. Al-Roshti, and N. Kraiem, "Developing virtual lab to support the computer science education in moodle," in International Journal of Software Engineering and Its Applications, 2014.
15. Z. Al-Khanjari, Z. Al-Kindi, E. Al-Kindi, and N. Kraiem, "Developing educational mobile application architecture using SOA," Int. J. Multimed. Ubiquitous Eng., vol. 10, no. 9, pp. 247–254, 2015.
16. P. Adamczyk, P. H. Smith, R. E. Johnson, and M. Hafiz, REST and Web Service: In Theory and in Practice. Springer Science+Business Media, LLC, 2011.

17. M. Garriga, C. Mateos, A. Flores, A. Cechich, and A. Zunino, "RESTful service composition at a glance: A survey," *J. Netw. Comput. Appl.*, vol. 60, pp. 32–53, 2016.
18. T. Erl, B. Carlyle, C. Pautasso, and R. Balasubramanian, *SOA with REST. Principles, Pattern & Constraints for Building Enterprise Solutions with REST*. New Jersey: Prentice Hall, 2012.
19. R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The Architecture Tradeoff Analysis Method," 1998.
20. L. Bass, M. Klein, and G. Moreno, "2001 Applicability of General Scenarios to ATAM," Pittsburgh, PA, 2001.
21. H. Reza and W. Helps, "Security Trade-off Analysis of Service-oriented Software Architecture," *World J. Comput. Appl. Technol.*, vol. 1, no. 4, pp. 110–120, 2013.
22. B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, "Evaluating REST architectures - Approach, tooling and guidelines," in *Journal of Systems and Software*, 2016, vol. 112, pp. 156–180.
23. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Commun. ACM*, vol. 59, no. 5, pp. 50–57, 2016.
24. A. Furda, C. Fidge, A. Barros, and O. Zimmermann, *Reengineering Data-Centric Information Systems for the Cloud – A Method and Architectural Patterns Promoting Multitenancy*, 1st ed. Elsevier Inc., 2017.
25. R. Abbas, Z. Sultan, and S. N. Bhatti, "Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege," *Int. Conf. Commun. Technol. ComTech* 2017, pp. 39–44, 2017.
26. Jeffrey Fulmer, "Siege – An Open Source Stress Tester," 2017. [Online]. Available: <https://www.joedog.org/>.

AUTHORS PROFILE



Faisal, Graduate Scholar, Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480. He can be reached at email: faisal.ssi@binus.ac.id



Gede Putra Kusuma received PhD degree in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2013. He is currently working as a Lecturer and Research Coordinator in Computer Science Department, Bina Nusantara University, Indonesia. Before joining Bina Nusantara University, he was working as a Research Scientist in I2R – A*STAR, Singapore. His research interests include pattern recognition, machine learning, face recognition, appearance-based object recognition, mobile learning, and gamification of learning.