IAAS implementation for Moodle LMS with 5000 plus concurrent users in K-12 and High School XYZ in Surabaya with Google Cloud Platform Compute Engine Autoscaling

Lewi Supranata Kristianto [1], Resmana Lim[2,*]

[1,2]Program Studi Program Profesi Insinyur Universitas Kristen Petra, Surabaya, Jawa Timur

Email: resma@petra.ac.id (korespondensi)

The year 2020 has been seen as an expedited and faster process for digital transformation especially for education to adapt to the new learning environment caused by Covid-19. Schools facing problems on how to build platforms that are fast to deploy and cost-effective to serve all students at the same time or concurrent connections. The purpose of this study is to determine design and infrastructure requirements for enablement for the e-learning management system using the open-source learning management system Moodle in one of the biggest private schools in Surabaya. Moodle Learning management system will be used as a platform for whole students for learning and quiz. For this purpose a real case study, using PPDIOO Life-cycle with the result proved design architecture of Moodle learning management system, methods implementation starting with centralized load from 1,000 users for 1 school in the single server, reach scalability for the large load with auto scaling architecture, experiments with a tryout concurrent quiz for students, will determine the requirement for each functional server (database, cache, file) in the architecture design. Moving out from the single server architecture to architecture design with autoscale for web server and architecture that separate between resources database, cache, and file, the architecture is capable to serve connections and cost-effective with maximum Concurrent Users 5,459. The area of improvement is to scale for a minimum of 10,000 concurrent users to serve whole students accessing moodle at the same time for the peak time.

Keywords: Moodle Open Source LMS, Google Cloud Platform, Autoscaling, Surabaya - Indonesia

Introduction

Pandemic Covid-19 has forced us to perceive new ways for schools to adapt to the new normal. The Four Rationales for Introducing ICT in Education: [1]

1. Social: The perceived role that technology now plays in society and the need for familiarizing students with technology.
2. Vocational: Preparing students for jobs that require skills in technology.
3. Catalytic: Utility of technology to improve performance and effectiveness in teaching, management, and many other social activities.
4. Pedagogical: To utilize technology in enhancing learning, flexibility, and efficiency in curriculum delivery.

A learning management system definition from Wikipedia is a software application for the administration, documentation, tracking, reporting, automation, and delivery of educational courses, training programs, or learning and development programs. The learning management system concept emerged directly from e-Learning. [2]

e-Learning is learning utilizing electronic technologies to access educational curriculum outside of a traditional classroom. In most cases, it refers to a course, program or degree delivered completely online. A study by UNESCO says that still 200+ million students in 23 countries are affected by the pandemic now [3], and believe that we will not go back to normal nearly soon [4], e-learning has been emerging and increasing utilization for schools in the year 2020.

Moodle is open-source and free software, a learning management system providing a platform for e-learning and it helps the various educators considerably in conceptualizing the various courses, course structures, and curriculum thus facilitating interaction with online students. Moodle was devised by Martin Dougiamas and since its inception, its primary agenda has been to contribute suitably to the system

of e-learning and facilitate online education and attainment of online degrees.

Moodle stands for Modular Object-Oriented Dynamic Learning Environment and statistics reveal that Moodle Powering hundreds of thousands of learning environments globally, Moodle is trusted by institutions and organizations large and small, including Shell, London School of Economics, State University of New York, Microsoft and the Open University. Moodle's worldwide numbers of more than 213 million users[1] across both academic and enterprise level usage makes it the world's most widely used learning platform. With over 10 years of development guided by social constructionist pedagogy, Moodle delivers a powerful set of learner-centric tools and collaborative learning environments that empower both teaching and learning. [5]

The open-source nature of Moodle is a significant characteristic that sets it apart from other Learning Management Systems (LMS). Behind the philosophy of open source is the idea that by working together with others in a collaborative way, computer programs can be improved. Thus, the source code is not kept secret but is openly shared with the public. By doing so, open-source software developers from around the world can and do contribute to the continuous refinement of Moodle. The development of new plugins, themes, and modules is undertaken by a globally diffused network of commercial and non-commercial users. [6]

Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube. Google Cloud Platform is a provider of computing resources for deploying and operating applications on the web. Its specialty is providing a place for individuals and enterprises to build and run software, and it uses the web to connect to the users of that software.

Scalability is a key requirement as Moodle becomes a critical application for a school to extend education methods and teaching activities. The research will be focusing on the scalability and architecture required for the load, starting with 1,000 concurrent users in the single server, and continue with large loads that have been increasing more while students may require access to the quiz module concurrently to submit for quiz and grade evaluation faster and real-time for students and teachers, and load has been significantly having increased for the number of users for the PAS (Penilaian Akhir Semester) for all schools. The result for this study is design architecture and specification on Google Cloud Platform to solve requirements for the large load that is required for Moodle Learning Management Systems access with 5000 plus concurrent users.

Research Methodology

The general timeline for conduct testing and data collection was from September to October of 2020 with completion of data analysis in November 2020.

In performing this research, the methods used are the Prepare, Plan, Design, Implement, Operate, Optimize. The PPDIOO phases are as follows: [7]

1. Prepare:
Involves establishing the organizational requirements, literature study, and proposing a high-level conceptual architecture identifying technologies that can best support the architecture.

2. Plan:
Involves identifying initial server and software requirements based on goals, facilities, user needs for the new sites.

3. Design:
The server design specification is a comprehensive detailed design that meets current business and technical requirements and incorporates specifications to support availability, reliability, security, scalability, and performance. The design specification is the basis for the implementation activities.

4. Implement:
The server, software, autoscale is built or additional components are incorporated according to the design specifications.

5. Operate:
Operation is the final test of the appropriateness of the design. The operational phase involves maintaining server health through day-to-day operations, including maintaining high availability and reducing expenses. The connection problem detection, correction, and performance monitoring that occur in daily operations provide the initial data for the optimization phase.

6. Optimize:
Involves proactive management of the server. The goal of proactive management is to identify and resolve issues before they affect the organization. Reactive fault detection and correction (troubleshooting) is needed when proactive management cannot predict and mitigate failures.

Conceptual Model

The conceptual model describes the concept of logic description to help solve the problems that will be designed in this study. In the framework of the conceptual model described for Design Architecture Moodle on Google Cloud Platform recommendation to solve the problem.

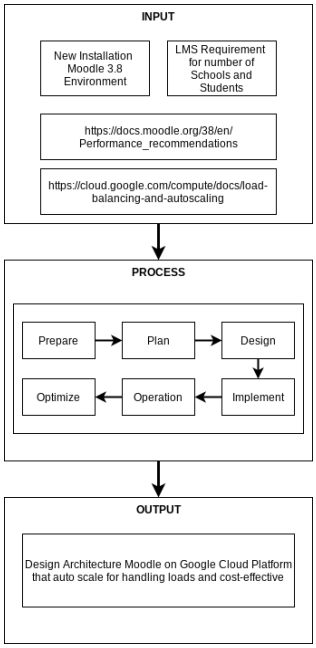


Figure 1. Conceptual Method

Systematic Research

This research used systematic research to determine the flow of the stages of research for problem resolution. Stages of research conducted following the stages available on the methods PPDIOO. Based on the problem definition these application methods PPDIOO result outcomes is design architecture moodle and specification on Google Cloud Platform.

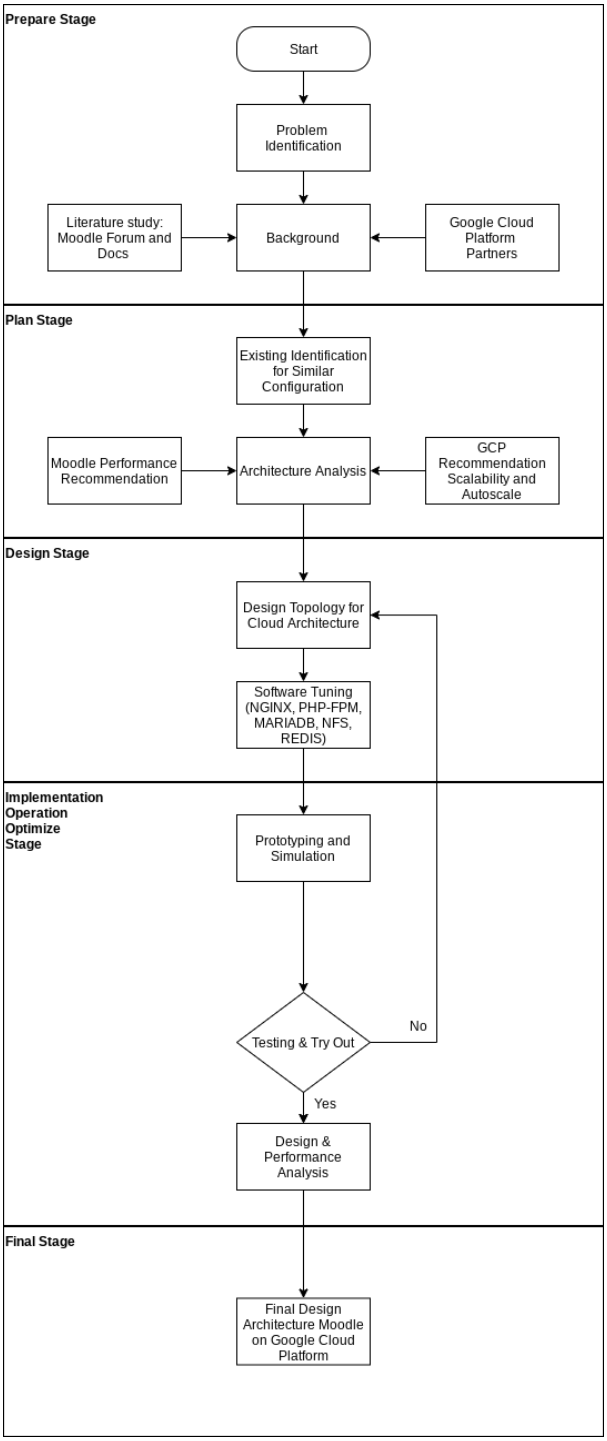


Figure 2. Systematic Research

This study component consists of:

1. **Operating System**
   An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs. Linux has long been the basis of commercial networking devices, but now it's a mainstay of enterprise infrastructure.

Linux is a tried-and-true, open-source operating system released in 1991 for computers, but its use has expanded to underpin systems for cars, phones, web servers, and more recently, networking gear. This study will use Ubuntu Linux 20.04 as an operating system.

2. **Web Server**
A web server is server software or hardware dedicated to running this software, that can satisfy client requests on the World Wide Web. A web server processes incoming network requests over HTTP / HTTPS and several other related protocols.

Nginx, stylized as NGINX or nginx or NginX, is a web server that can also be used as a reverse proxy, load balancer, mail proxy, and HTTP cache. The software was created by Igor Sysoev and publicly released in 2004. Nginx is free and open-source software, released under the terms of the 2-clause BSD license. This study will use Nginx as a web server.

3. **Scripting Language**
PHP is a general-purpose scripting language especially suited to web development.

PHP-FPM (FastCGI Process Manager) is a web tool used to speed up the performance of a website. It is much faster than traditional CGI based methods and has the ability to handle tremendous loads simultaneously. This study will use PHP-FPM 7.4.3 as a scripting language for Moodle.

4. **Database**
A database is an organized collection of data, generally stored and accessed electronically from a computer system.

MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system, intended to remain free and open-source software under the GNU General Public License. This study will use MariaDB 10.5 for databases.

5. **Network File System**
Network File System is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed. This study will use NFS 1.3.4 for network file sharing in an auto scaling architecture.

6. **Cache**
In computing, a cache is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere.

Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache, and message broker, with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes. This study will use Redis 5.0 as a cache server.

7. **Moodle**
Moodle is a free and open-source learning management system, Moodle 3.8 release highlights new features, brings forum enhancements, further learning analytics functionalities, and a first-stage H5P integration. This Study will use Moodle 3.8.x as a platform for Learning Management Systems.

Components Summary:

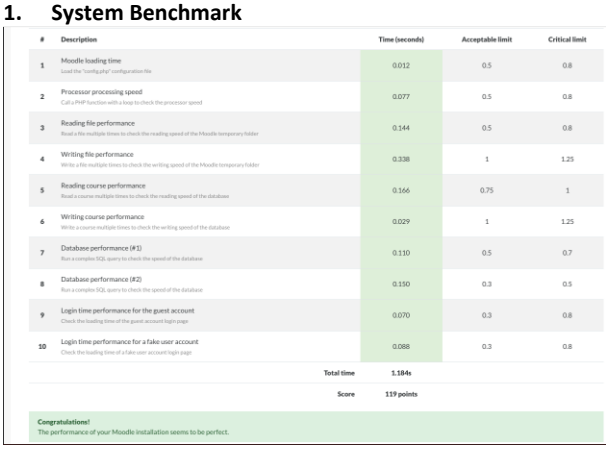
No	OS / Software	Version
1	Ubuntu Linux	20.04
2	Nginx	1.18.0
3	PHP-FPM	7.4.3
4	MariaDB	10.5
5	NFS	1.3.4
6	Redis Memory Store	5.0
7	Moodle	3.8.x

These components have their functions for Moodle Learning Management System that affect the performance and have important factors in the scalability.

In these experiments, real tryouts with sample concurrent quiz for students are showing real performance stress and load tests for all components.

There are 3 monitoring tools that the researcher use in this study:

1. **System Benchmark**



#	Description	Time (seconds)	Acceptable limit	Critical limit
1	Moodle loading time Load the "config.php" configuration file	0.012	0.5	0.8
2	Processor processing speed Call a PHP function with a loop to check the processor speed	0.077	0.5	0.8
3	Reading file performance Read a file multiple times to check the reading speed of the Moodle temporary folder	0.144	0.5	0.8
4	Writing file performance Write a file multiple times to check the writing speed of the Moodle temporary folder	0.338	1	1.25
5	Reading course performance Read a course multiple times to check the reading speed of the database	0.166	0.75	1
6	Writing course performance Write a course multiple times to check the writing speed of the database	0.029	1	1.25
7	Database performance (#1) Run a complex SQL query to check the speed of the database	0.110	0.5	0.7
8	Database performance (#2) Run a complex SQL query to check the speed of the database	0.150	0.3	0.5
9	Login time performance for the guest account Check the loading time of the guest account login page	0.070	0.3	0.8
10	Login time performance for a fake user account Check the loading time of a fake user account login page	0.088	0.3	0.8
	Total time	1.184s		
	Score	119 points		

Congratulations!
The performance of your Moodle installation seems to be perfect.

Figure 3. Moodle System Benchmark

Parameters that have been tested in this tool:
1. Moodle loading time
   Load the "config.php" configuration file

2. Processor processing speed
   Call a PHP function with a loop to check the processor speed
3. Reading file performance
   Read a file multiple times to check the reading speed of the Moodle temporary folder
4. Writing file performance
   Write a file multiple times to check the writing speed of the Moodle temporary folder
5. Reading course performance
   Read a course multiple times to check the reading speed of the database
6. Writing course performance
   Write a course multiple times to check the writing speed of the database
7. Database performance (#1)
   Run a complex SQL query to check the speed of the database
8. Database performance (#2)
   Run a complex SQL query to check the speed of the database
9. Login time performance for the guest account
   Check the loading time of the guest account login page
10. Login time performance for a fake user account
    Check the loading time of a fake user account login page

This tool will run continuously during peak time in the quiz concurrent test, to check which parameters show longer than Acceptable Limit and Critical Limit. Results should have a maximum time value below Critical Limit time.

Benchmark scores show the time taken for the parameters tested, generally only testing for the CPU, File and Cache access, and Database. As these tools do not show real-world quiz load, also do not take stress load yet for the CPU and Memory, but this number should take into account while auto scaling configuration.

2. Caching Test

For caching performance we can use an internal test in Moodle, in this experiment we should succeed with a minimum of 10,000 unique requests for Cache store performance reporting on Instance groups.

Cache store performance reporting - 10000 unique requests per operation.

Test with 1, 10, 100, 500, 1000, 5000, 10000, 50000, 100000 requests

Store requests when used as an application cache.

Plugin	Result	Set	Get - Hit	Get - Miss	Delete
APC user cache (APCu)	Untestable	-	-	-	-
File cache	Tested	5.9043	1.1982	0.4740	0.5435
Memcached	Untestable	-	-	-	-
MongoDB	Invalid plugin	-	-	-	-
Redis	Tested	2.1157	2.0998	2.0115	1.3365
Session cache	Unsupported mode	-	-	-	-
Static request cache	Unsupported mode	-	-	-	-

Figure 4. Moodle Cache Store Performance reporting

File cache, testing configuration affected with parameters, that require to set in outside NFS directory:
$CFG->tempdir = '/var/www/moodledata-temp';
$CFG->cachedir = '/var/www/moodledata-cache';

Redis, testing configuration affected with Redis Server configuration that recommended to configure to get performance in large site Moodle deployment.

Benchmark scores show the time taken for the parameters tested, specifically for caching configuration. As these tools do not show real-world quiz load, also do not take stress load yet for the CPU and Memory, but this number should take into account while auto scaling configuration.

3. Stackdriver / System Monitoring

Stackdriver agent is a collectd-based daemon that gathers system and application metrics from virtual machine instances and sends them to monitoring.

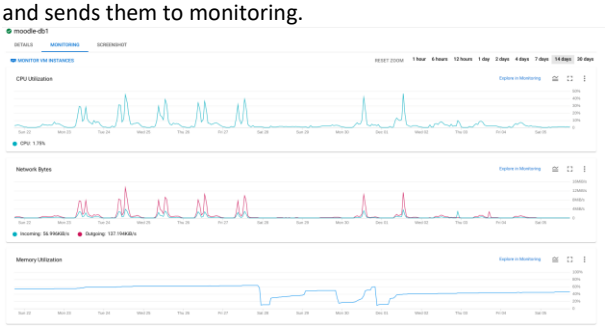


Figure 5. System Monitoring for Moodle Databases

Figure 6. System Monitoring for Moodle NFS

Figure 7. Network Bandwidth for Moodle Redis Memory Store

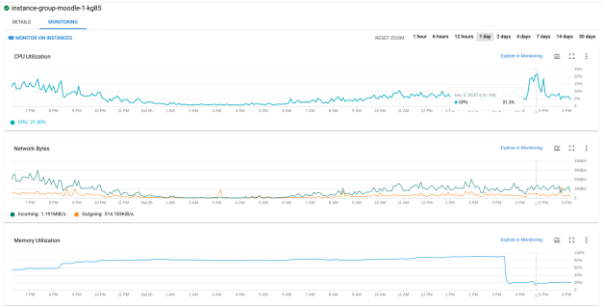


Figure 8. System Monitoring for Moodle Instance Group

Monitoring shows CPU, Network, Memory as parameters that we can use to determine for required specification for each server while the ongoing testing.

List tools used for monitoring:

- Moodle System Benchmark plugins: https://moodle.org/plugins/report_benchmark
- Caching test performance.
  Site Administration > Plugins > Caching > Test Performances.
- Stackdriver / System Monitoring: Google Cloud Platform for monitor utilization of CPU, Memory, Network, and Disk.

Single Server Architecture

Single server architecture has been used for initial testing environments that combine 4 components (Web Server, PHP, Database, Cache) in a single server with 8 Core CPU, 32 GB RAM, 500GB SSD Storage. The benefit for a single server always has simplicity but the main problem is lack of scalability, and while this should be able to handle the load with vertical scaling (a.k.a. "scaling up"), adding more power to each component that reaches peak their resources.

The configuration will not be optimal regarding cost for the cloud computing environment as load access in the experiment shows active only 09:00 - 15:00 and peak time in the morning.

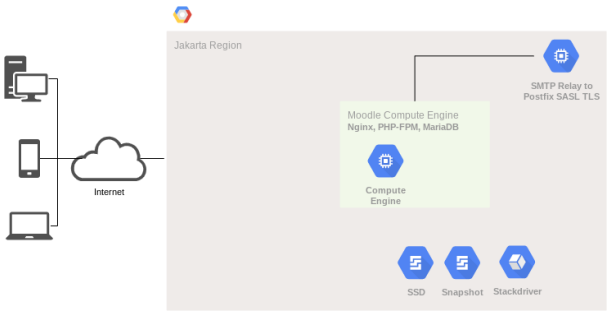


Figure 9. Moodle Single Server Architecture

Result with a single server with Nginx web server, shown maximum with 1,079 concurrent users, compared to Apache web server can reach 496 concurrent users only, this explains that Nginx requires much less computing resources compared to Apache in these experiments, in the process researcher use Nginx as a web server for the study.

Date	Active Users	Concurrent	Spec (CPU, Memory)	Web Server
01-Aug-2020 - 07-Sep-2020	6,013	496	8 Core, 32 GB	Apache
01-Sep-2020 - 07-Oct-2020	7,547	1,079	8 Core, 32 GB	Nginx

Concurrent Report Data from moosh tools for report concurrency. [8]

1. 01-Aug-2020 - 07-Sep-2020

~user/moosh/moosh.php report-concurrency --from 20200801 --to 20200907
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 6013
Max Concurrent Users: 496
on Friday, 2020-09-04 01:20:00
Average concurrent users per day of the week
Monday: 41.69
Tuesday: 35.35
Wednesday: 35.49
Thursday: 31.77
Friday: 26.43
Saturday: 7.63
Sunday: 6.6
Global average concurrent users: 26.38
Average concurrent users considering working days & hours: 25.69

2. 01-Sep-2020 - 07-Oct-2020

~user/moosh/moosh.php report-concurrency --from 20200901 --to 202001007
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 7547
Max Concurrent Users: 1079
on Monday, 2020-09-28 03:30:00
Average concurrent users per day of the week
Monday : 80.26
Tuesday : 71.68
Wednesday : 77.87
Thursday : 67.15
Friday : 62.52
Saturday : 13.36
Sunday : 16.76
Global average concurrent users: 55.27
Average concurrent users considering working days & hours: 56.69

Autoscale Architecture

Autoscaling is a tool that allows web servers to efficiently handle increases in traffic by dynamically adding compute capacity but also reduce capacity and costs in periods of low traffic and resource demand.

Google Compute Engine uses managed instance groups (MIGs), or a collection of common VM instances created from the same API resource known as a template, to automatically add or remove instances based on traffic and demand to your application. MIGs are multiple, identical VMs that deliver reliable availability and performance for the same application. They're managed as a single entity, which is a perfect scenario for using autoscaling.

Autoscale server architecture has been chosen as a solution for the architecture that is able to horizontally scale while maintaining cost in cloud computing resources. In horizontal scaling (a.k.a. "scaling out"), you get the additional resources into your system by adding more machines to your network, sharing the processing and memory workload across web servers. Being able to add or remove VMs based on resource demand and traffic allows building scalability and cost-effective infrastructure for Moodle learning management systems.

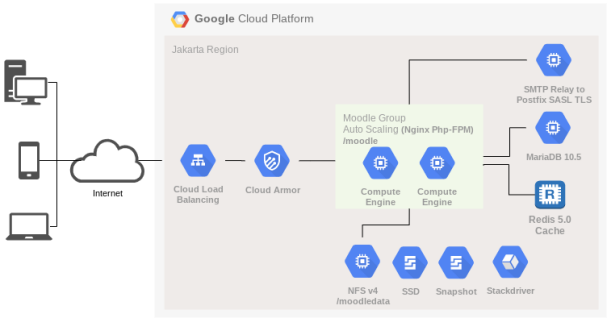


Figure 10. Moodle Autoscale Architecture

This study separates the web server and database on separate servers, although for smaller installations this is typically not necessary. It is possible to load-balance a Moodle installation, by using more than one web server. The separate web servers should query the same database and refer to the same filestore area, but otherwise the separation of the application layers is complete enough to make this kind of clustering feasible. [9]

Moodle Autoscale Architecture component consists of: [10]

1. **Cloud Load Balancing**
   Cloud load balancing is a type of load balancing that is performed in cloud computing. Cloud load balancing is the process of distributing workloads across multiple computing resources. Cloud load balancing reduces costs associated with document management systems and maximizes the availability of resources.

   A load balancer distributes user traffic across multiple instances of your applications. By spreading the load, load balancing reduces the risk that your applications experience performance issues.

2. **Cloud Armor**
   Google Cloud Armor security policies protect your application by regulating which requests are allowed and denied access to your load balancer. Each security policy is made up of a set of rules that filter traffic based on conditions like an incoming request's IP address, IP range, region code, or request headers.

   Google Cloud Armor security policies are available only for backend services behind an external HTTP(S) load balancer. Benefit using cloud armor:
   - Benefit from DDoS protection and WAF at Google scale
   - Detect and mitigate attacks against your Cloud - Load Balancing workloads
   - Mitigate OWASP Top 10 risks and help protect workloads on-premises or in the cloud

3. **Managed Instance Groups for Web Server and PHP-FPM**
   An instance group is a collection of virtual machine (VM) instances that you can manage as a single entity.

   Managed instance groups (MIGs) let you operate apps on multiple identical VMs. You can make your workloads scalable and highly available by taking advantage of automated MIG services, including: autoscaling, autohealing, regional (multiple zones) deployment, and automatic updating. Manage Instance Groups will hold PHP files from Moodle for path /var/www/moodle.

   Specification used:
   Instance type: n1-standard-2
   (2 Core CPU, 7.5 GB Memory)

4. **NFS Instance for directory moodledata with SSD Storage**
   A Google Compute Engine that is installed for NFS servers that shares across Managed Instance Groups, NFS export sharing directory is used for path /var/www/moodledata.

   Specification used:
   Instance type: n1-custom-4-6144
   (6 Core CPU, 6 GB Memory)

5. **Stackdriver Agent installed in all instances**
   Using the Monitoring agent is optional but recommended. Monitoring can access some instance metrics without the Monitoring agent, including CPU utilization, some disk traffic metrics, network traffic, and uptime information. Monitoring uses the Monitoring agent to access additional system resources and application services in a virtual machine (VM) instances.

6. **Snapshot use as disk backup method**
   Use snapshot schedules as a best practice to back up Compute Engine workloads.

7. **Database Instance use with MariaDB**
   A Google Compute Engine that is installed for MariaDB databases.

   Specification used:
   Instance type: n1-custom-12-12288
   (10 Core CPU, 12 GB Memory)

8. **Redis 5.0 Memory Store**
   Memorystore automates complex tasks for open source Redis and Memcached like enabling high availability, failover, patching, and monitoring, reduce latency with scalable, secure, and highly available in-memory service for Redis.

   Specification used:
   Redis Memory Store 5.0 with 2GB memory

9. **SMTP Relay for outgoing email notification**
   Sending email from an instance is blocked from Google. By default, Compute Engine allows outbound connections on all ports except port 25, which is blocked because of the risk of abuse. All other ports are open, including ports 587 and 465.

Current usage will require email notification for 13,000 active students, that will minimum of 150,000 emails sent every month from Moodle learning management systems, this setup requires an external own postfix SMTP server, that relay from one of instance that installed postfix with SMTP relay. [11]

Configuration for Managed Instance Groups

Each instance have been configure with PHP-FPM pool config with below:

pm = dynamic
pm.max_children = 1000
pm.start_servers = 200
pm.min_spare_servers = 200
pm.max_spare_servers = 1000
pm.max_requests = 2000

Monitoring process and memory utilize in the instance showing information that require to determine the size for the VM in the Managed Instance Groups

```
# ps_mem.py
 Private  +   Shared  =  RAM used       Program

100.0 KiB +   1.5 KiB = 101.5 KiB       stackdriver-collectdmon
200.0 KiB +  12.5 KiB = 212.5 KiB       atd
212.0 KiB +   2.5 KiB = 214.5 KiB       grub-efi-amd64-
316.0 KiB +  31.5 KiB = 347.5 KiB       cron
268.0 KiB +  94.0 KiB = 362.0 KiB       agetty (2)
432.0 KiB +  69.5 KiB = 501.5 KiB       rpcbind
508.0 KiB +  41.5 KiB = 549.5 KiB       grub-multi-inst
480.0 KiB + 207.0 KiB = 687.0 KiB       chronyd (2)
  1.2 MiB +  33.5 KiB =   1.2 MiB       whiptail
524.0 KiB + 722.5 KiB =   1.2 MiB       systemd-udevd
932.0 KiB + 433.0 KiB =   1.3 MiB       sftp-server (2)
  1.1 MiB + 205.5 KiB =   1.3 MiB       dbus-daemon
  1.2 MiB + 582.5 KiB =   1.8 MiB       systemd-logind
  1.1 MiB + 874.5 KiB =   2.0 MiB       polkitd
  1.9 MiB + 127.5 KiB =   2.0 MiB       systemd-networkd
  2.2 MiB +  50.5 KiB =   2.3 MiB       rsyslogd
  1.5 MiB + 927.0 KiB =   2.4 MiB       sudo (2)
  1.7 MiB + 856.5 KiB =   2.5 MiB       accounts-daemon
  3.0 MiB +  31.5 KiB =   3.1 MiB       dpkg
  3.5 MiB + 841.5 KiB =   4.3 MiB       freshclam
  4.5 MiB + 482.5 KiB =   5.0 MiB       systemd-resolved
  4.7 MiB +   1.2 MiB =   5.9 MiB       bash (4)
  3.2 MiB +   4.4 MiB =   7.6 MiB       sshd (5)
  7.5 MiB + 226.5 KiB =   7.7 MiB       networkd-dispat
  7.6 MiB + 370.5 KiB =   8.0 MiB       unattended-upgr
  6.4 MiB +   5.5 MiB =  11.9 MiB       systemd (3)
 12.9 MiB + 338.5 KiB =  13.3 MiB       multipathd
 13.6 MiB +  54.5 KiB =  13.7 MiB       google_metadata
 13.8 MiB + 486.5 KiB =  14.2 MiB       google_clock_sk
 13.9 MiB + 538.5 KiB =  14.4 MiB       google_network_
 14.4 MiB +  45.5 KiB =  14.5 MiB       frontend
 14.1 MiB + 715.5 KiB =  14.8 MiB       google_accounts
 14.6 MiB + 378.5 KiB =  15.0 MiB       apt-get
 12.6 MiB +   4.9 MiB =  17.6 MiB       systemd-journald
 19.6 MiB +   2.5 MiB =  19.6 MiB       snapd
 17.0 MiB +  35.7 MiB =  52.7 MiB       nginx (3)
 74.5 MiB + 998.5 KiB =  75.4 MiB       stackdriver-collectd
  5.8 GiB + 104.2 MiB =   5.9 GiB       php-fpm7.4 (1001)
---------------------------------
                        6.3 GiB
=================================
```

Figure 11. Memory and Process used on Nginx and PHP-FPM Managed Instance Groups

While in the peak load, checking from tools ps_mem.py, showing that process and memory calculation [12], showing that each instance can running php-fpm7.4 with total 1,001 process, each process having memory 5,894 MB with total memory 5,9 GB, from this information, this input determines each instance require to have minimum 7.5 GB.

Experiments show that peak load happened on Monday date 16-Nov-2020, Managed Instance Groups showing numbers of instances peak to 18 for 5,549 concurrent users.

Date	Active Users	Concurrent	Instance #	Web Server
22-Oct-2020	7,400	955	8	Nginx
28-Oct-2020	6,822	1,537	12	Nginx
02-Nov-2020	8,109	911	4	Nginx
16-Nov-2020	11,866	5,549	18	Nginx

Concurrent Report Data from moosh tools for report concurrency. [8]

Concurrent Report from Autoscale Architecture

1. 22-Oct-2020

~user/moosh/moosh.php report-concurrency --from 20201022 --to 20201022
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 7400
Max Concurrent Users: 955
on Thursday, 2020-10-22 06:00:00
Average concurrent users per day of the week
Thursday : 136.78
Global average concurrent users: 136.78
Average concurrent users considering working days & hours: 136.78

2. 28-Oct-2020

~user/moosh/moosh.php report-concurrency --from 20201028 --to 20201028
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 6822
Max Concurrent Users: 1537
on Wednesday, 2020-10-28 01:50:00
Average concurrent users per day of the week
Wednesday : 137.31
Global average concurrent users: 137.31
Average concurrent users considering working days & hours: 137.31

3. 02-Nov-2020

~user/moosh/moosh.php report-concurrency --from 20201102 --to 20201102
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 8109
Max Concurrent Users: 911
on Monday, 2020-11-02 01:30:00
Average concurrent users per day of the week
Monday : 149.86
Global average concurrent users: 149.86
Average concurrent users considering working days & hours: 149.86

4. 16-Nov-2020

~user/moosh/moosh.php report-concurrency --from 20201116 --to 20201116
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 11866
Max Concurrent Users: 5549
on Monday, 2020-11-16 01:30:00
Average concurrent users per day of the week
Monday : 443.69
Global average concurrent users: 443.69
Average concurrent users considering working days & hours: 443.69

Weekly Concurrent Report

1. Month: September
- **21-Sep-2020 until 28-Sep-2020**

~user/moosh/moosh.php report-concurrency --from 20200921 --to 20200928
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 6199

Max Concurrent Users: 1079
on Monday, 2020-09-28 03:30:00
Average concurrent users per day of the week
Monday : 128.47
Wednesday : 18.61
Thursday : 69.62
Friday : 69.01
Saturday : 15.79
Sunday : 22.41
Global average concurrent users: 56.63
Average concurrent users considering working days & hours: 53.99

2. Month: October
- **28-Sep-2020 until 05-Oct-2020**
~user/moosh/moosh.php report-concurrency --from 20200928 --to 20201005
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 6423
Max Concurrent Users: 1079
on Monday, 2020-09-28 03:30:00
Average concurrent users per day of the week
Monday : 104.48
Tuesday : 115.44
Wednesday : 98.24
Thursday : 89.65
Friday : 81.86
Saturday : 10.53
Sunday : 20.48
Global average concurrent users: 78.14
Average concurrent users considering working days & hours: 78.14
- **05-Oct-2020 until 12-Oct-2020**
~user/moosh/moosh.php report-concurrency --from 20201005 --to 20201012
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 7386
Max Concurrent Users: 725
on Tuesday, 2020-10-06 01:20:00
Average concurrent users per day of the week
Monday : 71.14
Tuesday : 70.18
Wednesday : 50.29
Thursday : 48.83
Friday : 58.43
Saturday : 12.74
Sunday : 14.1
Global average concurrent users: 49.6
Average concurrent users considering working days & hours: 49.61

- **21-Sep-2020 until 28-Sep-2020**
~user/moosh/moosh.php report-concurrency --from 20201012 --to 20201019
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 9716
Max Concurrent Users: 609
on Monday, 2020-10-19 01:25:00
Average concurrent users per day of the week
Monday : 82.28
Tuesday : 68.95
Wednesday : 45.48
Thursday : 32.75
Friday : 31.06
Saturday : 13.48
Sunday : 25.4
Global average concurrent users: 47.71
Average concurrent users considering working days & hours: 47.71

- **19-Oct-2020 until 26-Oct-2020**
~user/moosh/moosh.php report-concurrency --from 20201019 --to 20201026
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 11809
Max Concurrent Users: 955
on Thursday, 2020-10-22 06:00:00
Average concurrent users per day of the week
Monday : 105.03
Tuesday : 110.21
Wednesday : 120.18
Thursday : 136.89
Friday : 111.85
Saturday : 26.3
Sunday : 35.45
Global average concurrent users: 93.75
Average concurrent users considering working days & hours: 93.87

- **26-Oct-2020 until 02-Nov-2020**
~user/moosh/moosh.php report-concurrency --from 20201026 --to 20201102
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 11457
Max Concurrent Users: 1537
on Wednesday, 2020-10-28 01:50:00
Average concurrent users per day of the week
Monday : 128.6
Tuesday : 127.5
Wednesday : 137.38
Thursday : 34.12
Friday : 114.05
Saturday : 28.23
Sunday : 37.81
Global average concurrent users: 92.2
Average concurrent users considering working days & hours: 92.04

3. Month: November
- **02-Nov-2020 until 07-Nov-2020**
~user/moosh/moosh.php report-concurrency --from 20201102 --to 20201107
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 11623
Max Concurrent Users: 911
on Monday, 2020-11-02 01:30:00
Average concurrent users per day of the week
Monday : 149.86
Tuesday : 121.34

Wednesday : 131.23
Thursday : 130.63
Friday : 141.97
Saturday : 37.49
Global average concurrent users: 119.03
Average concurrent users considering working days & hours: 118.75

- **07-Nov-2020 until 16-Nov-2020**
~user/moosh/moosh.php report-concurrency --from 20201107 --to 20201116
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 12949
Max Concurrent Users: 5549
on Monday, 2020-11-16 01:30:00
Average concurrent users per day of the week
Monday : 307.33
Tuesday : 222
Wednesday : 181.95
Thursday : 210.34
Friday : 428.09
Saturday : 41.11
Sunday : 57.87
Global average concurrent users: 175.84
Average concurrent users considering working days & hours: 185.5

- **16-Nov-2020 until 23-Nov-2020**
~user/moosh/moosh.php report-concurrency --from 20201116 --to 20201123
Name: https://lmsmoodle.schooldomain.sch.id
Active Users: 12568
Max Concurrent Users: 5549
on Monday, 2020-11-16 01:30:00
Average concurrent users per day of the week
Monday : 600.7
Tuesday : 374.01
Wednesday : 253.77
Thursday : 214
Friday : 279.81
Saturday : 69.59
Sunday : 89.84
Global average concurrent users: 273.12
Average concurrent users considering working days & hours: 310.3

System monitoring performance on peak usage on Monday date 16-Nov-2020:

VM	CPU	CPU Peak	Memory	Memory Peak	SSD	Network
Databases	12 core	80%	12 GB	96.48 %	200 GB	35 MB/s
NFS	4 core	50%	6 GB	31%	750 GB	78 MB/s
Managed Instance Groups	@2 core	74%	7.5 GB	80%	20 GB	464 MB/s
Redis1	2 core	90%	4 GB	20%	10 GB	325 MB/s

Design architecture for Moodle Autoscale Architecture has been optimal for current usage with 5,549 concurrent users, there is optimization that has been concluded for minimum specification with above 5,000 plus concurrent users for the autoscale architecture.

VM	CPU	Memory	SSD	Instance #
DB	12 core	24 GB	250 GB	1
NFS	4 core	6 GB	750 GB	1
Managed Instance Group	@2 core	8 GB	10 GB	min: 2 max: 20
Redis Memory Store	Managed by Google	2 GB	Managed by Google	Managed by Google

Conclusion

Autoscale architecture on Google Cloud Platform enables for faster deployment, cost-effective and scalable for Moodle Learning Management Systems. Web server resources have been showing the most resources demand from the deployment, in this study autoscaling has been successfully to solve the problem with auto scaling and load balance technology.

The main goal for this research was to determine and testing infrastructure scalability in Google Cloud Platform while maintaining cost-effectiveness in cloud computing environments as an Infrastructure as a Services (IaaS) with autoscale and load balance architecture, this will benefit end-users to maintain cost while off-peak access and schools holiday.

Advice for the readers is to improve and test for high availability and performance in the future. Area for improvement but not limited to MariaDB Database Galera Clustering, multi-region High Availability, and Content Delivery Network. In the databases area, scalability improvement can be reached with MariaDB Galera clustering, and to improve for high availability for infrastructure in multi region, and using Content Delivery Network to improve latency for user accessing in multi different region areas.

Acknowledgment

Thank you to Mr. Dwi Budiman for the opportunity for the study experiment, and Pendidikan Profesi Insinyur (PPI) from Universitas Kristen Petra for the opportunity and assistance for the research and advice for this research project. Hopefully, this research can be useful for the input for readers and other schools that require developing scalability Moodle learning management systems.

References

1 Source: Cross, M. & Adam, F. (2007). ICT Policies and Strategies in Higher Education in South Africa: National and Institutional Pathways', Higher Education Policy 20(1), 73–95.
2 https://en.wikipedia.org/wiki/Learning_management_system
3 https://en.unesco.org/covid19/educationresponse
4 https://www.technologyreview.com/2020/03/17/905264/coronavirus-pandemic-social-distancing-18-months/
5 http://etec.ctlt.ubc.ca/510wiki/Moodle_and_Constructivism
6 https://docs.moodle.org/en/About_Moodle
7 Cisco. (2005). CREATING BUSINESS VALUE AND OPERATIONAL EXCELLENCE. 4.
8 https://moosh-online.com/commands/
9 https://docs.moodle.org/20/en/Performance_recommendations#Scalability
10 https://cloud.google.com/compute/docs/load-balancing-and-autoscaling
11 https://www.linode.com/docs/guides/postfix-smtp-debian7
12 http://www.pixelbeat.org/scripts/ps_mem.py