

# OPTIMIZING MU LINUX SERVER WORKFLOW

---

## Problem Statement

Efficient utilization of university resources when deploying Moodle on MU linux server.

## Introduction

The university Moodle instance Euclid (<https://euclid-mu.in/>) has always been deployed entirely on the university servers, or entirely on the cloud without any backup solutions. This naturally puts a lot of load on the university servers and bank accounts.

The goal of this project is to figure out the most efficient, sustainable and cost effective method of hosting Moodle for Mahindra University. We aim to produce the following:

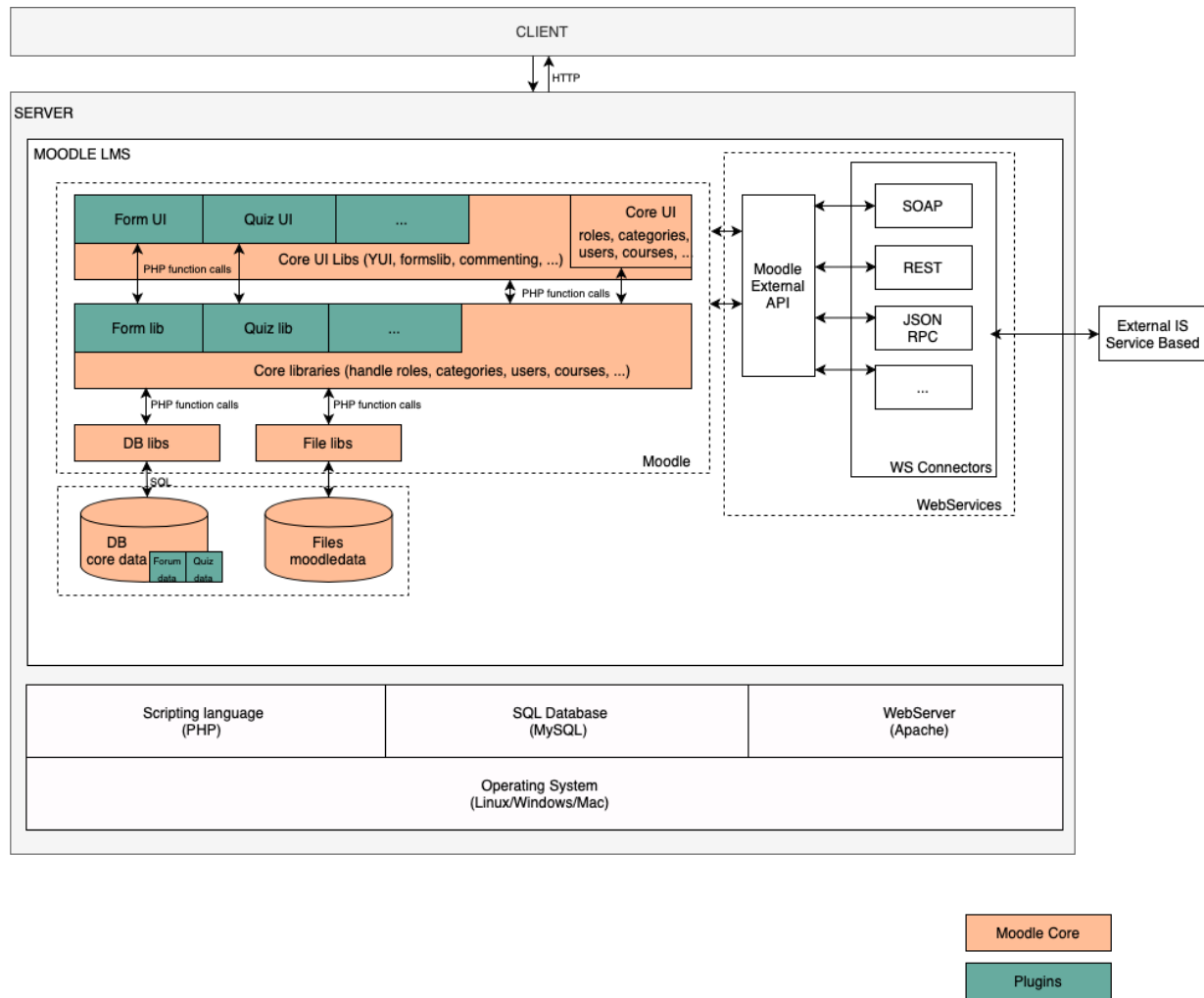
- Setup more than two instances of Moodle for the university, one on the university Linux server and another on the cloud.
- Setup a load balancer between the two Moodle instances to distribute traffic.
- Solve synchronization and latency induced problems in having two different instances of the same application.
- Create a disaster recovery solution to backup all data stored in Euclid.

## Euclid (Moodle)

Euclid (<https://euclid-mu.in/>) is an instance of Moodle setup for Mahindra University. Moodle is a fully-fledged free and open source platform for online learning and course management. Moodle has been a popular choice for many institutions due to its abundance of features and simple design.

## Moodle architecture

---



## Web server

Moodle uses Apache2 web server by default for serving PHP pages. For this project, we have decided to replace Apache2 with Nginx in the Moodle stack.

## Apache2

Since 1996, Apache HTTP web server has been the most popular choice for web servers on Linux. It aims to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. Like other traditional web servers,

---

Apache2 spawns a new thread to serve every new request received, which results in higher memory consumption.

## **Nginx**

NGINX is a high-performance HTTP server and reverse proxy. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load.

Even though configuring Nginx for Moodle is more difficult than Apache2, we have made the choice for the following reasons:

- Nginx is based on event-driven architecture with non-blocking I/O. The web server can handle significantly more traffic than Apache2.
- During installation of Moodle on AWS EC2 instance, we constantly found the server with 1GB of RAM running out of memory with Apache2. However, this was resolved as soon as it was replaced with Nginx.
- Nginx also acts as a load balancer and reverse proxy.

## **Moodledata**

The `moodledata` directory is the location of files that are uploaded or created by the Moodle interface. Since it is a directory, it can have a different name and location than the default name of `moodledata`. The `moodledata` directory is initially created with an initial install of a Moodle site.

---

```
ubuntu@ip-172-31-5-95:/var/moodledata$ ls --color=none -l
total 32
drwxrwxrwx  3 www-data www-data 4096 May 30 11:38 cache
drwxrwxrwx 10 www-data www-data 4096 May 30 17:19 filedir
drwxrwxrwx  2 www-data www-data 4096 May 30 11:31 lang
drwxrwxrwx  7 www-data www-data 4096 Jun  1 06:14 localcache
drwxrwxrwx  2 www-data www-data 4096 May 30 11:36 muc
drwxrwxrwx  2 www-data www-data 4096 Jun  2 09:20 sessions
drwxrwxrwx  5 www-data www-data 4096 May 30 11:42 temp
drwxrwxrwx  2 www-data www-data 4096 May 30 11:35 trashdir
ubuntu@ip-172-31-5-95:/var/moodledata$
```

For the two instances of Moodle to be in sync, this moodledata directory needs to be synchronized between the two servers. There are several different approaches to this:

## Rsync

Rsync is a fast and extraordinarily versatile file copying tool. It can copy locally, to/from another host over any remote shell, or to/from a remote rsync daemon. It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied. It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination. Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

Rsync is an excellent tool for backups when set to run as cron jobs, but not for synchronizing remote directories. Let's assume we have two remote servers A and B with the following file contents

```
ubuntu@ip-172-31-5-95:/tmp$ ls A
file1
ubuntu@ip-172-31-5-95:/tmp$ ls B
file2
ubuntu@ip-172-31-5-95:/tmp$
```

Now when we sync the two directories with rsync, one operation to sync A to B and another operation to sync B to A, we get the following state

---

```
ubuntu@ip-172-31-5-95:/tmp$ rsync -a A/ B/
ubuntu@ip-172-31-5-95:/tmp$ rsync -a B/ A/
ubuntu@ip-172-31-5-95:/tmp$ ls A/
file1 file2
ubuntu@ip-172-31-5-95:/tmp$ ls B/
file1 file2
```

Now assume one of the files was an assignment submission from a student, which was later deleted from the Moodle UI. This will not be reflected on the other server and hence the filesystem storage keeps on growing.

```
ubuntu@ip-172-31-5-95:/tmp$ rm A/file1
ubuntu@ip-172-31-5-95:/tmp$ rsync -a A/ B/
ubuntu@ip-172-31-5-95:/tmp$ rsync -a B/ A/
ubuntu@ip-172-31-5-95:/tmp$ ls A/
file1 file2
ubuntu@ip-172-31-5-95:/tmp$ ls B/
file1 file2
ubuntu@ip-172-31-5-95:/tmp$
```

## SSHFS

In computing, SSHFS (SSH Filesystem) is a filesystem client to mount and interact with directories and files located on a remote server over a normal ssh connection. The client interacts with the remote file system via the SSH File Transfer Protocol (SFTP), a network protocol providing file access, file transfer, and file management functionality over any reliable data stream that was designed as an extension of the Secure Shell protocol (SSH) version 2.0.

After mounting the SSHFS directory, it appears as an ordinary directory to moodle and all operations can be done just as they would on a local filesystem directory.

SSHFS seems to tick all the checkboxes for our requirements until we remember that it uses SSH under the hood and SSH is blocked by the MU firewall due to security concerns. Due to these restrictions, the following read and write benchmarks were performed over a Wireguard VPN connection and hence cannot be compared directly to its alternatives.

---

Writing 256MB to SSHFS mount from client side:

```
real@real-Standard-PC-Q35-ICH9-2009:~/testdir$ time dd if=/dev/random of=./testfile bs=16k count=16384
16384+0 records in
16384+0 records out
268435456 bytes (268 MB, 256 MiB) copied, 591.193 s, 454 kB/s

real    9m52.445s
user    0m0.060s
sys     0m9.708s
```

Reading 200MB from SSHFS mount from client side(interrupted due to very slow transfer speeds):

```
real@real-Standard-PC-Q35-ICH9-2009:~/testdir$ time dd if=./testfile of=/dev/null bs=16k
^C12830+0 records in
12829+0 records out
210190336 bytes (210 MB, 200 MiB) copied, 1603.28 s, 131 kB/s

real    26m43.600s
user    0m0.014s
sys     0m0.198s
```

## NFS

Network File System (NFS) 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. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system.

Although commonly used for file sharing within a network, NFS can also be used over the internet. The latest release of the NFS protocol, NFSv4 allows for encryption of file transfers, much like SSHFS. It appears as a regular filesystem directory to Moodle, introducing no troubles with read and write operations.

NFSv4 is also much faster than its competitors like Samba and AFP (Apple Filing Protocol). This is seen from the following benchmarks where it manages to saturate the internet connection speed, indicating that it has low processing overhead.

Writing 256MB to NFS mount from client side:

---

```
real@real-Standard-PC-Q35-ICH9-2009:/var/moodledata/FILEDIR$ time dd if=/dev/random of=./testfile bs=16k count=16384
16384+0 records in
16384+0 records out
268435456 bytes (268 MB, 256 MiB) copied, 11.9517 s, 22.5 MB/s

real    0m12.001s
user    0m0.012s
sys     0m7.465s
```

Reading 256MB from NFS mount from client side:

```
real@real-Standard-PC-Q35-ICH9-2009:/var/moodledata/FILEDIR$ time dd if=testfile2 of=/dev/null bs=16k
16384+0 records in
16384+0 records out
268435456 bytes (268 MB, 256 MiB) copied, 26.2989 s, 10.2 MB/s

real    0m26.357s
user    0m0.018s
sys     0m0.272s
```

After considering all alternatives, we found NFS to be the most suitable for this case.

## DB Core Data

The Moodle SQL database consists of around 200 tables. This database consists of every single customizable aspect of Moodle, everything ranging from usernames, emails to submission file names are stored in this database. Every table in the database has a prefix (mdl\_ by default), which can be customized during installation.

```
mysql> show tables;
+-----+
| Tables_in_moodle |
+-----+
| mdl_adminpresets |
| mdl_adminpresets_app |
| mdl_adminpresets_app_it |
| mdl_adminpresets_app_it_a |
| mdl_adminpresets_app_plug |
| mdl_adminpresets_it |
| mdl_adminpresets_it_a |
| mdl_adminpresets_plug |
```

Since we have more than one instance of Moodle, we need the database to be synchronized. This is done by having a centralized database on the AWS server. The

---

database cannot be shared using tools like NFS because ACID properties are not guaranteed.

## INSTALLING MOODLE ON AWS

### STEPS:-

1. Using Ubuntu 22.04 LTS AMI from AWS marketplace, create a new t2.small EC2 instance.
2. Store the private key file to some accessible location and change it's permission to make it read-only to the owner of the file using

```
sudo chmod 400 moodleaws.pem
```

3. Under AWS services go to Elastic IP addresses and allocate new Elastic IP addresses for the previously created EC2 instance.
4. Copy the public IP address of the EC2 instance and SSH into it using

```
ssh ubuntu@[IP address of the EC2 instance ] -v -i [path to  
private key file stored in Step 2]
```

-v displays verbose output to help in debugging

5. After successfully landing on the remote shell, update the repositories using

```
sudo apt update
```

6. To add php7 PPA (required for moodle), use

```
sudo add-apt-repository ppa:ondrej/php && sudo apt update
```

7. Install essential packages required for Moodle setup



---

```
sudo apt install graphviz aspell ghostscript clamav  
php7.4-pspell php7.4-curl php7.4-gd php7.4-intl php7.4-mysql  
php7.4-xml php7.4-xmlrpc php7.4-ldap php7.4-zip php7.4-soap  
php7.4-mbstring nginx git vim mysql-client mysql-server php7.4  
libapache2-mod-php php7.4-fpm php7.4-cli
```

#### 8. Download Moodle source code

```
cd /opt && sudo git clone git://git.moodle.org/moodle.git
```

#### 9. Switch to Moodle 4.0 branch

```
sudo git checkout MOODLE_400_STABLE
```

#### 10. Copy over the local repository to /var/www/html for nginx to serve the web pages

```
sudo cp -R /opt/moodle /var/www/html/
```

#### 11. Create and set appropriate permissions for moodledata directory

```
sudo mkdir /var/moodledata && sudo chown -R www-data  
/var/moodledata && sudo chmod -R 777 /var/moodledata && sudo  
chmod -R 777 /var/www/html/moodle
```

#### 12. Setup MYSQL root user password

```
sudo mysql -u root
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH  
mysql_native_password by 'rootroot';
```

Followed by

```
sudo mysql_secure_installation
```

In /etc/mysql/mysql.conf.d/mysqld.cnf, comment or remove these 2 lines

---

```
bind-address = 127.0.0.1
```

```
mysqlx-bind-address = 127.0.0.1
```

### 13. Create new MYSQL user and database

```
mysql> CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_unicode_ci;
```

```
mysql> CREATE USER 'moodleuser'@'%' IDENTIFIED BY  
'passwordformoodleuser';
```

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE  
TEMPORARY TABLES,DROP,INDEX,ALTER ON moodle.* TO  
'moodleuser'@'%';
```

```
mysql> flush privileges;
```

### 14. Setup nginx to serve moodle PHP pages

In /etc/nginx/nginx.conf, uncomment and change  
server\_names\_hash\_bucket\_size 128;

Then,

```
cd /etc/nginx/sites-available/
```

```
sudo cp default moodle
```

```
sudo vim moodle
```

Paste the following configuration:

---

```
server {  
  
    listen 80;  
  
    listen [::]:80;  
  
    Server_name  
  
    Ec2-13-126-241-30.ap-south-1.compute.amazonaws.com;  
  
    root /var/www/html/moodle;  
  
    index index.php index.html;  
  
    location / {  
  
        try_files $uri $uri/ =404;  
  
    }  
  
    # location ~ /\.php$ {  
  
        #         include snippets/fastcgi-php.conf;  
  
        #         # With php-fpm (or other unix sockets):  
  
        #         fastcgi_pass unix:/run/php/php7.4-fpm.sock;  
  
        #         # # With php-cgi (or other tcp sockets):  
  
        #         # fastcgi_pass 127.0.0.1:9000;  
  
    }# # deny access to .htaccess files, if Apache's document  
  
    root  
  
    # # concurs with nginx's one  
  
    location ~ [^/]\.php(/|$) {  
  
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
```

---

---

```
fastcgi_index            index.php;

fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;

include fastcgi_params;

fastcgi_param    PATH_INFO    $fastcgi_path_info;

fastcgi_param    SCRIPT_FILENAME
$document_root$fastcgi_script_name;

}

location ~ /\.ht {

deny all;

}

}

cd ../sites-enabled/

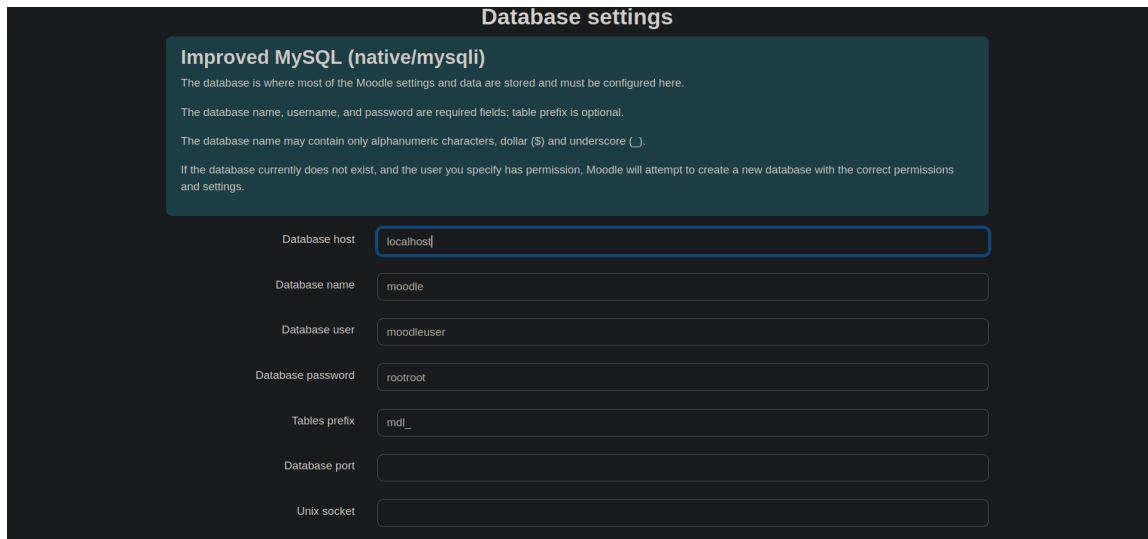
sudo ln -s ../sites-available/moodle .

sudo systemctl restart nginx

sudo systemctl restart php7.4-fpm
```

15. Go to [http://\[IP address of the EC2 instance\]/install.php](http://[IP address of the EC2 instance]/install.php)

Change data directory to `/var/moodldata/`



The screenshot shows the 'Database settings' form in Moodle. It features a dark blue header with the title 'Database settings' and a sub-header 'Improved MySQL (native/mysqli)'. Below this, there are three lines of instructional text: 'The database is where most of the Moodle settings and data are stored and must be configured here.', 'The database name, username, and password are required fields; table prefix is optional.', and 'The database name may contain only alphanumeric characters, dollar (\$) and underscore (\_).'. A final line of text states: 'If the database currently does not exist, and the user you specify has permission, Moodle will attempt to create a new database with the correct permissions and settings.' Below the text are several input fields: 'Database host' (containing 'localhost'), 'Database name' (containing 'moodle'), 'Database user' (containing 'moodleuser'), 'Database password' (containing 'rootroot'), 'Tables prefix' (containing 'mdl\_'), 'Database port' (empty), and 'Unix socket' (empty).

## INSTALLING MOODLE ON LOCAL NETWORK

Steps for installing Moodle on local network are same as AWS, however there are some minor changes in the following steps:

Step 13 is not required

In Step 14, server name is changed to be localhost

In Step 15 GUI installer, database host is changed from localhost to the domain name of the remote server (in this case AWS EC2 instance).

## SYNCHRONIZING DATA BETWEEN MOODLE INSTANCES (NFS)

```
ubuntu@ip-172-31-5-95:/var/moodldata$ ls --color=none -l
total 32
drwxrwxrwx  3 www-data www-data 4096 May 30 11:38 cache
drwxrwxrwx 10 www-data www-data 4096 May 30 17:19 filedir
drwxrwxrwx  2 www-data www-data 4096 May 30 11:31 lang
drwxrwxrwx  7 www-data www-data 4096 Jun  1 06:14 localcache
drwxrwxrwx  2 www-data www-data 4096 May 30 11:36 muc
drwxrwxrwx  2 www-data www-data 4096 Jun  2 14:35 sessions
drwxrwxrwx  5 www-data www-data 4096 May 30 11:42 temp
drwxrwxrwx  2 www-data www-data 4096 May 30 11:35 trashdir
ubuntu@ip-172-31-5-95:/var/moodldata$
```

---

Out of all the directories in moodledata, only 3 of them need to be synchronized:

- `filedir` - Stores all user uploaded images, documents, etc.
- `lang` - Stores language packs for Moodle.
- `sessions` - Stores sessions for all logged in users. This directory needs to be synced so that a user switching from AWS instance to local instance does not get logged out from their browser.

## Installation steps in AWS:

### 1. Download and install NFS server packages

```
Sudo apt-get update
```

```
sudo apt install nfs-kernel-server
```

### 2. Edit `/etc/exports`

Every file system being exported to remote users via NFS, as well as the access level for those file systems, are listed in the `/etc/exports` file.

```
sudo vim /etc/exports
```

Add the following at the end of the file:

```
/var/moodledata [Public IP address of the server on local  
network] (rw, sync, no_subtree_check, insecure)
```

`rw` indicates that the NFS mount has read and write permissions

`insecure` indicates that the client can use ports > 1024 for NFS

This allows the public IP of the server on the local network to mount `/var/moodledata` with appropriate permissions. In this case we are giving it Single Host accessibility, so we specify one particular host with a fully qualified IP address or domain name.

### 3. Restart NFS server daemon

---

```
sudo systemctl restart nfs-kernel-server
```

## Installation steps in local instance:

### 1. Download and install NFS client packages

```
Sudo apt-get update
```

```
sudo apt install nfs-common
```

### 2. Move the current moodledata directory

```
cd /var/moodledata
```

```
mv filedir filedir2
```

```
mkdir filedir
```

```
sudo chown www-data:www-data filedir
```

www-data is the default user and group for web servers on Ubuntu.

```
sudo chmod -R 777 filedir
```

### 3. Mount the NFS share on the local instance

```
sudo mount -t nfs4 -vvv [IP address of AWS  
instance]:/var/moodledata/filedir /var/moodledata/filedir
```

```
sudo mount -t nfs4 -vvv [IP address of AWS  
instance]:/var/moodledata/sessions /var/moodledata/sessions
```

```
sudo mount -t nfs4 -vvv [IP address of AWS  
instance]:/var/moodledata/lang /var/moodledata/lang
```

### 4. To make the mount permanent, i.e. mount every time the server boots up, add these entry to /etc/fstab file

---

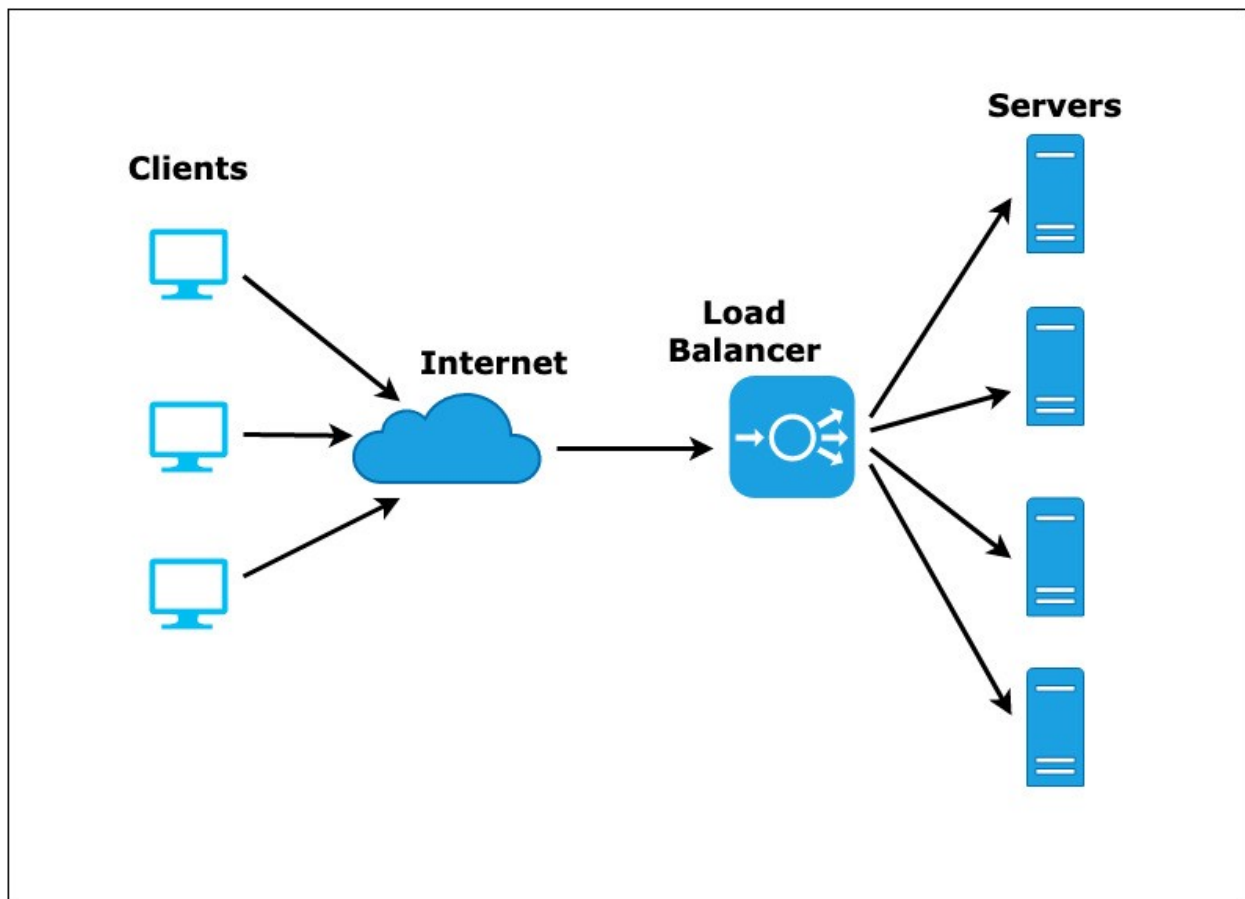
```
[IP address of AWS instance]:/var/moodledata/filedir
/var/moodledata/filedir nfs4 rw 0 0

[IP address of AWS instance]:/var/moodledata/sessions
/var/moodledata/sessions nfs4 rw 0 0

[IP address of AWS instance]:/var/moodledata/lang
/var/moodledata/lang nfs4 rw 0 0
```

## LOAD BALANCING

Traditional load balancer:





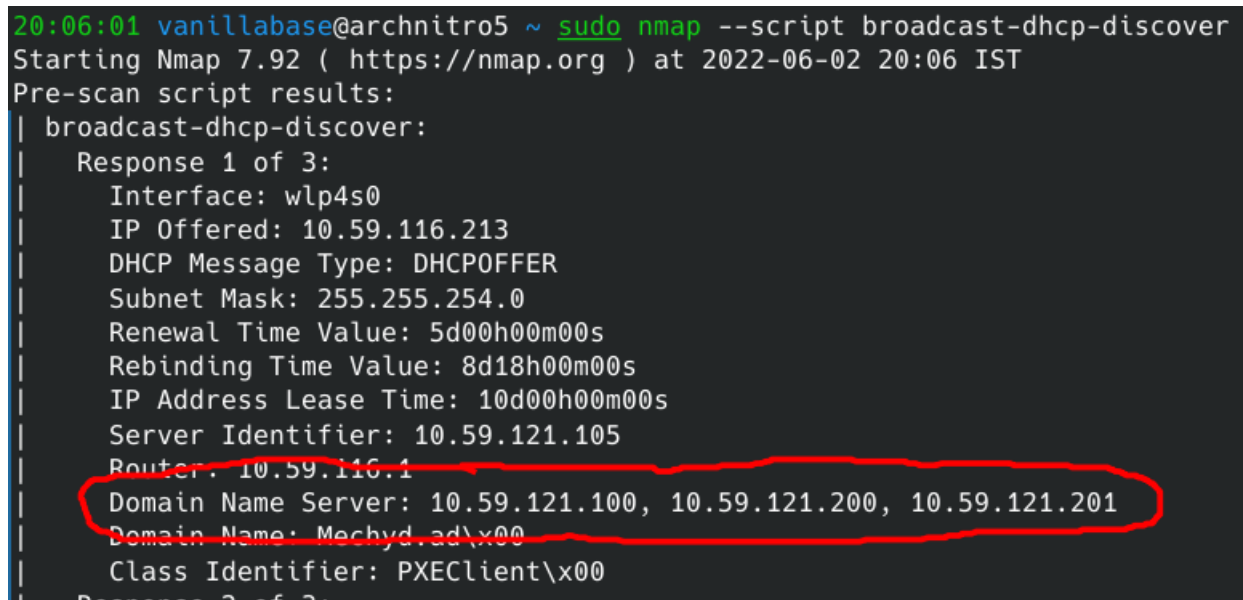
---

A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks.

## DNS server, a natural Load Balancer

Now that we have solved all synchronization issues, a load balancer will distribute traffic between the two Moodle instances using some algorithm, e.g. round robin, resource based, etc. But this load balancer also has to be on one of these servers, whose IP address is fetched from A records of a DNS server.

The following is a DHCP response from MU network:

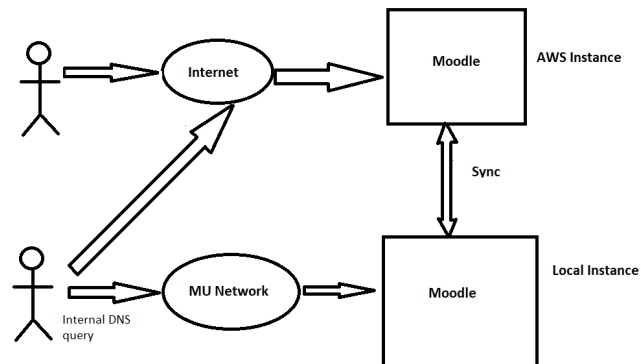


```
20:06:01 vanillabase@archnitro5 ~ sudo nmap --script broadcast-dhcp-discover
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-02 20:06 IST
Pre-scan script results:
| broadcast-dhcp-discover:
|   Response 1 of 3:
|     Interface: wlp4s0
|     IP Offered: 10.59.116.213
|     DHCP Message Type: DHCP OFFER
|     Subnet Mask: 255.255.254.0
|     Renewal Time Value: 5d00h00m00s
|     Rebinding Time Value: 8d18h00m00s
|     IP Address Lease Time: 10d00h00m00s
|     Server Identifier: 10.59.121.105
|     Router: 10.59.116.1
|     Domain Name Server: 10.59.121.100, 10.59.121.200, 10.59.121.201
|     Domain Name: Mechyd.sd\x00
|     Class Identifier: PXEClient\x00
|   Response 2 of 3:
```

We can see that we get 3 internal DNS servers. The A records corresponding to <https://euclid-mu.in> can be changed to point to the Moodle instance running on the local network. The following are advantages of this approach:

- Users on the university network do not need internet access to visit Euclid.
- No port forwarding has to be configured as in case of traditional load balancing.

- 
- No incoming traffic from the internet.
  - This can also be used to seclude users present outside the university network.



## BACKUP SOLUTION (RSYNC)

As mentioned earlier, Rsync is not suitable for data synchronization, but when combined with CRON jobs, it makes a sophisticated and powerful backup utility. We will use Rsync to keep a backup of the moodledata directory and this backup will be done at the end of every day.

### Steps to configure Rsync for moodledata backup:

1. Create and edit a script to be run as a CRON job. Name the script as `moodle_backup.sh`:

---

```
/usr/bin/rsync --exclude='muc' --exclude='localcache'
--exclude='cache' --exclude='trashdir' -aPvze "/usr/bin/ssh -i
[path to identity/private key file]" ubuntu@[IP address of the
server to backup]:/var/moodledata /var/
```

2. Open the crontab file of the root user:

```
sudo crontab -u root -e
```

3. At the end of the crontab file, add the following lines:

```
0 0 * * * [path to moodle_backup.sh] >> /var/cronlogs.txt
```

After this, every day at 0000 hrs, the server will fetch the moodledata directory from the remote server and all the logs of every operation performed on every single day will be stored in `/var/cronlogs.txt` file.

## Conclusion

Now that everything is in place, we benchmark the two Moodle instances and see how they perform.

For testing purposes, this will redirect all requests intended for euclid-mu.in to the local instance.

Edit `/etc/hosts` file and add the line at the end:

```
[Internal IP address of the local Moodle instance] euclid-mu.in
```

Fetching the home page of Euclid from AWS instance:

```
real@real-Standard-PC-Q35-ICH9-2009:~$ time curl -L --silent --output /dev/null http://ec2-13-126-241-30.ap-south-1.compute.amazonaws.com/
real    0m0.201s
user    0m0.009s
sys     0m0.006s
```

Fetching the home page of Euclid from local instance:

---

```
real@real-Standard-PC-Q35-ICH9-2009:~$ time curl -L --silent --output /dev/null http://euclid-mu.in/
real    0m0.953s
user    0m0.005s
sys     0m0.008s
real@real-Standard-PC-Q35-ICH9-2009:~$
```

## References

[https://miro.medium.com/max/1400/1\\*tEaZGz-p1-E2ytNjl5RPJg.jpeg](https://miro.medium.com/max/1400/1*tEaZGz-p1-E2ytNjl5RPJg.jpeg)

<https://moodle.org/mod/forum/discuss.php?d=403016>

[https://docs.moodle.org/dev/Moodle\\_architecture](https://docs.moodle.org/dev/Moodle_architecture)

<https://techexpert.tips/moodle/moodle-installation-nginx/>