คู่มือติดตั้งปรับแต่งประสิทธิภาพ Moodle LMS

บริษัท คลัสเตอร์คิท จำกัด
www.clusterkit.co.th

สมาชิก สมาคมศึกษาและพัฒนาโอเพ่นซอร์ส
www.oseda.or.th

12 มกราคม 2564

สารบัญ

1. คุยกันก่อน....3
2. ซอฟต์แวร์ที่เกี่ยวข้อง....3
3. ติดตั้งเว็บเซิร์ฟเวอร์ PHP และ MariaDB....4
3.1 ติดตั้งเว็บเซิร์ฟเวอร์ และ PHP....4
3.2 ติดตั้ง MariaDB....5
4. การใช้ Caching กับ Moodle....6
4.1 Memcahced....6
4.2 Redis....10
5. ปรับแต่ง NGINX....12
5.1 ปรับแต่งตัวแปรเพื่อให้รับคอนเน็กชั่นพร้อม ๆ กันได้มากขึ้น....12
5.2 เปิดฟังก์ชั่น GZIP เพื่อบีบอัดข้อมูลที่ส่งมายัง Client....12
5.3 ปรับเวลา timeout ให้ NGINX....13
6. ปรับแต่ง PHP-FPM....13
6.1 ปรับจำนวนโปรเซส....13
6.2 ปรับค่า timeout ในการประมวลผล PHP....14
7. การทดสอบประสิทธิภาพเว็บเซิร์ฟเวอร์ด้วย Apache Benchmark....14
8. ปรับแต่ง MySQL/MariaDB....15
8.1 InnoDB Buffer Pool....15
8.2 ตัวแปรอื่น ๆ เพิ่มเติม....16
9. ตั้งเวลาการทำงานของสคริปต์ตามที่ Moodle กำหนด....17
10. ซอฟต์แวร์ตรวจตราระบบ (Monitoring Tools) เป็นสิ่งจำเป็น....17
11. ส่งท้าย....17
12. อ้างอิง....18

1. คุยกันก่อน

เอกสารนี้เป็นคู่มือแนะนำการปรับแต่งซอฟต์แวร์จัดการเรียนการสอน Moodle ที่นิยมใช้กันมากในสถานศึกษา เพื่อให้สามารถรองรับการใช้งานจากผู้ใช้งานจำนวนมากพร้อม ๆ กันได้ ในหลักการทั่วไปที่นิยมกันก็จะไล่ทำไปตามลำดับ ตั้งแต่การปรับแต่งประสิทธิภาพ (Tuning) การขยายขนาดฮาร์ดแวร์(Scale Up) ขั้นสุดท้ายคือการขยายจำนวนเครื่อง(Scale Out)

ในเอกสารนี้จะกล่าวถึงการปรับแต่งประสิทธิภาพเป็นหลัก เพราะเชื่อว่าเพียงแค่การปรับแต่งประสิทธิภาพที่เหมาะสม บนฮาร์ดแวร์ที่มีทรัพยากรเพียงพอ อาจจะขยายจำนวนเครื่องแบบง่าย ๆ เช่นการแยกเว็บเซิร์ฟเวอร์กับซอฟต์แวร์จัดการฐานข้อมูลเป็นคนละเครื่องกัน ก็รองรับการใช้งาน Moodle พร้อม ๆ กันหลักพันได้แล้ว ผู้เขียนหวังว่าคู่มือนี้จะเป็นประโยชน์กับผู้ที่ติดปัญหาเรื่องรับภาระงานของระบบอยู่ตามสมควร

2. ซอฟต์แวร์ที่เกี่ยวข้อง

ซอฟต์แวร์ที่กล่าวถึงในเอกสารนี้จะอ้างอิงกับระบบปฏิบัติการลีนุกซ์เป็นหลัก ในที่นี้เราใช้ CentOS-8 และซอฟต์แวร์เกี่ยวเนื่องกันดังต่อไปนี้

- NGINX
- PHP-7 (PHP-FPM)
- MariaDB/MySQL
- Memcached
- Redis

ในช่วงแรกจะกล่าวถึงการติดตั้งซอฟต์แวร์เพื่อเป็นเซิร์ฟเวอร์สำหรับ Moodle สักเล็กน้อย เพื่อให้เห็นสภาพว่าระบบในเอกสารคุยกันอยู่เป็นระบบแบบใด

เอกสารนี้เกิดจากความร่วมมือระหว่าง

สมาคมศึกษาและพัฒนาโอเพ่นซอร์ส และ บริษัท คลัสเตอร์คิท จำกัด

3. ติดตั้งเว็บเซิร์ฟเวอร์ PHP และ MariaDB

3.1 ติดตั้งเว็บเซิร์ฟเวอร์ และ PHP

1. คำสั่งต่อไปนี้เป็นการติดตั้งเว็บเซิร์ฟเวอร์ NGINX และ PHP

```
yum install -y nginx php-gd php-mbstring php-soap php-xml php-xmlrpc \
 php-pecl-zendopcache php-intl php-fpm php-json php-mysqlnd php-zip php-xml
```

2. สร้างคอนฟิกไฟล์สำหรับ Moodle ที่ /etc/nginx/conf.d/moodle.conf

```
server {
    listen 80;
    listen [::]:80;
    root /var/www/html/moodle;
    index  index.php index.html index.htm;
    server_name  _;

    location / {
    try_files $uri $uri/ =404;
    }

    location /dataroot/ {
    internal;
    alias /moodledata/;
    }

    location ~ ^(.+\.php)(.*)$ {
        fastcgi_split_path_info  ^(.+\.php)(.*)$;
        include /etc/nginx/default.d/php.conf;
        fastcgi_pass unix:/var/run/php-fpm/www.sock;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param   PATH_INFO       $fastcgi_path_info;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
```

คอนฟิกในส่วนของ location{} นั้นอ้างอิงจาก https://docs.moodle.org/39/en/Nginx

3. แก้ไขไฟล์ /etc/nginx/nginx.conf ให้คอมเม้นต์ระหว่างบรรทัด server{ } ทั้งหมด
4. แก้ไขไฟล์ /etc/php-fpm.d/www.conf เปลี่ยนค่าจาก user และ group จาก apache เป็น nginx

```
user = nginx
group = nginx
```

5. เริ่มการทำงานของ php-fpm และ NGINX

```
systemctl enable php-fpm --now
systemctl enable nginx --now
```

6. เปิดไฟร์วอลล์เพื่อให้บริการเว็บที่พอร์ต 80

```
firewall-cmd --zone=public –add-service=http
firewall-cmd --permanent --zone=public --add-service=http
```

3.2 ติดตั้ง MariaDB

MariaDB เป็นโครงการที่แยกมาจาก MySQL เพราะฉะนั้นคำสั่งการใช้งานพื้นฐานจะไม่ต่างกัน ใน CentOS ตั้งแต่เวอร์ชั่น 7 เลือกบรรจุ MariaDB มาแทน MySQL

1. ติดตั้ง MariaDB

```
yum install -y mariadb-server
```

2. เริ่มการทำงาน MariaDB

```
systemctl enable mariadb --now
```

3. รัน mysql_secure_install เพื่อความปลอดภัย

4. ถ้า DBMS อยู่คนละเครื่องกับเว็บ อย่าลืมเปิดไฟล์วอลล์ให้เครื่องเว็บติดต่อเข้ามาใช้งานได้

5. สร้างฐานข้อมูลและผู้ใช้สำหรับ moodle

```
create database moodle;
create user moodle@'10.0.2.15' identified by 'password';
grant all on moodle.* to moodle@'10.0.2.15';
```

ข้างต้นเป็นตัวอย่างสร้างฐานข้อมูล moodle และสร้างบัญชีผู้ใช้ชื่อ moodle ที่จะติดต่อมาจากเครื่องไอพี 10.0.2.15 มีรหัสผ่านเป็นคำว่า password (อย่าลืมตั้งให้เหมาะสม) และมีการให้สิทธิ์ในทุกตารางของฐานข้อมูล moodle กับผู้ใช้ `moodle@'10.0.2.15'`

ถึงขั้นนี้ก็ให้ติดตั้ง moodle ผ่านหน้าเว็บไปตามลำดับ

4. การใช้ Caching กับ Moodle

ตั้งแต่ Moodle 2.4 ได้มีการเพิ่มความสามารถในการทำ Caching ที่เรียกว่า the Moodle Universal Cache (MUC) เข้ามา Moodle แบ่งแคชเป็น 3 ส่วน คือ application cache, session cache และ request cache รองรับซอฟต์แวร์สำหรับทำแคชหลากหลาย เช่น Memcached, MongoDB, APC user cache (APCu) and Redis ในเอกสารจะกล่าวถึงทั้ง Memcached และ Redis

Redis จะคอนฟิกให้ Moodle ใช้ทั้ง application cache และ session cache ผ่านหน้าเว็บแอดมิน แต่กับ Memcached ที่หน้าแอดมินจะคอนฟิกให้ใช้ได้เฉพาะ application cache เท่านั้น ส่วน session cache ต้องไปคอนฟิกที่ไฟล์ config.php

4.1 Memcahced

1. ติดตั้ง Memcached (ในที่นี้ติดตั้ง Memcached ไว้ที่เดียวกับเว็บเซิร์ฟเวอร์)

```
yum install -y memcached
```

2. คอนฟิก Memcached ที่ไฟล์ /etc/sysconfig/memcached

```
PORT="11211"
USER="memcached"
MAXCONN="10000"
CACHESIZE="256"  # MB
OPTIONS="-l 127.0.0.1,::1"
```

ในส่วนนี้เราปรับตัวแปร MAXCONN เพื่อรับคอนเน็กชั่นที่มากขึ้น และปรับขนาด CACHESIZE เพิ่มขึ้น ไม่จำเป็นต้องมาก เพราะจาก https://docs.moodle.org/310/en/MUC_FAQ ระบุว่า 64MB ก็เพียงพอ

หากจะให้ Memcached ยอมรับการติดต่อจากเครื่องอื่น จะต้องปรับบรรทัด OPTIONS เผื่อให้ binding ไอพีที่ต้องการให้บริการ เช่น OPTIONS="-l 127.0.0.1,::1,10.0.2.15 -U 0" คือเพิ่มการเปิดให้บริการผ่านไอพีหมายเลข 10.0.2.15 ด้วย และยังกำหนดว่าให้บริการเฉพาะ TCP ไม่เปิด UDP ด้วยคำสั่ง “-U 0”

3. เริ่มการทำงาน Memcached

```
systemctl enable --now memcached
```

4. ทดสอบ Memcached

```
telnet 127.0.0.1 11211
stats
```

สั่ง quit เพื่อออกจาก telnet และสามารถดูสถานะการใช้งานหน่วยความจำของ Memcached ได้ผ่านคำสั่ง

```
systemctl status memcached
```

5. คอนฟิกให้ PHP คุยกับ Memcached ได้

เริ่มจากการติดตั้ง php-pear และแพ็กเกจที่เกี่ยวข้อง จากนั้นจะใช้ PECL (PHP Extension Community Library) ติดตั้ง Memcached client module อ้างอิง https://www.server-world.info/en/note?os=CentOS_8&p=memcached&f=4

```
dnf --enablerepo=PowerTools -y install php-pear php-devel zlib-devel \
      libmemcached-devel make
pecl install memcached
echo 'extension=memcached.so' >> /etc/php.d/20-memcached.ini
```

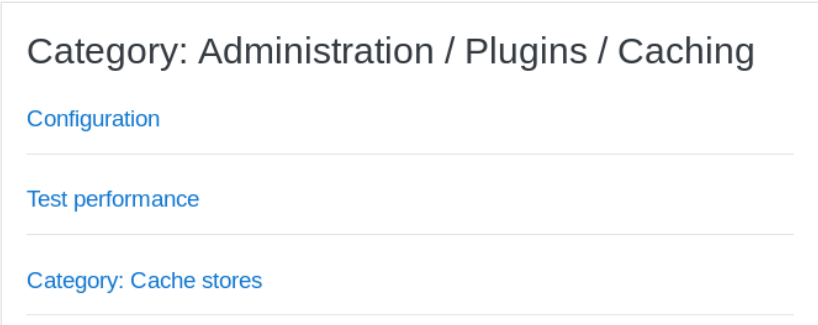
สั่ง systemctl restart php-fpm แล้วทดสอบโดยการสร้างไฟล์ phpinfo() ดูควรพบส่วนของ Memcached แบบรูปด้านล่าง ส่วนนี้บอกเราว่า php สามารถสื่อสารกับ Memcached ได้

memcached

memcached support	enabled
Version	3.1.5
libmemcached version	1.0.18
SASL support	yes
Session support	yes
igbinary support	no
json support	no
msgpack support	no

6. กำหนดให้ Moodle ใช้ Memcached

- ล็อกอินเป็น administrator ไปที่เมนู --> 'Site administration > Plugins > Caching'



ก่อนจะคอนฟิกให้ Moodle ใช้งาน Memcached เป็น Caching นั้น เราควรทดสอบดูก่อนว่า Moodle เราสามารถใช้งาน Memcached ที่เราติดตั้งไว้ได้หรือไม่ อย่างไร

จากรูปข้างต้น เมนู Configuration สำหรับคอนฟิกให้ Moodle ใช้งานแคช

เมนู Test performance สำหรับทดสอบแคช

เมนู Category: Cache stores สำหรับกำหนดค่าว่าเราใช้ซอฟต์แวร์ตัวไหนที่ไหนมาทำแคช (cache store) เราจะเริ่มคอนฟิกที่ส่วนนี้ก่อน

- คลิกไปที่ Category: Cache stores แล้วไปที่หัวข้อ Memcached ระบุไอพีของเครื่อง Memcached

Memcached

Test servers
cachestore_memcached | testservers

127.0.0.1

Default: Empty

One or more connection strings for memcached servers to test against. If a test server has been specified then memcached performance can be tested using the cache performance page in the administration block. As an example: 127.0.0.1:11211

กดปุ่ม SAVE CHANGES ที่ท้ายหน้าเพื่อบันทึก

จากนั้นไปที่หน้า Test Performance เพื่อทดสอบ

- ที่หน้า Test Performance ถ้า Moodle สามารถใช้งาน Memcached ได้จะแสดงผลดังรูปข้างล่าง คือมีเวลาในการเขียนและอ่านข้อมูลกับ Memcached แสดงขึ้นมา โดยจะเริ่มทดสอบที่ 100 requests พร้อมกัน สังเกตได้จากลิงก์ด้านบน เราสามารถเลือกทดสอบกับจำนวน requests ที่ต่างกันได้ เพียงแค่คลิกที่ลิงก์

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)	Invalid plugin	-	-	-	-
File cache	Tested	0.0239	0.0027	0.0004	0.0040
Memcached	Tested	0.0060	0.0094	0.0059	0.0046

ถ้า moodle ใช้งาน Memcached จึงไปคอนฟิกให้ Moodle ใช้งาน memcached เป็น caching ในหัวข้อถัดไป

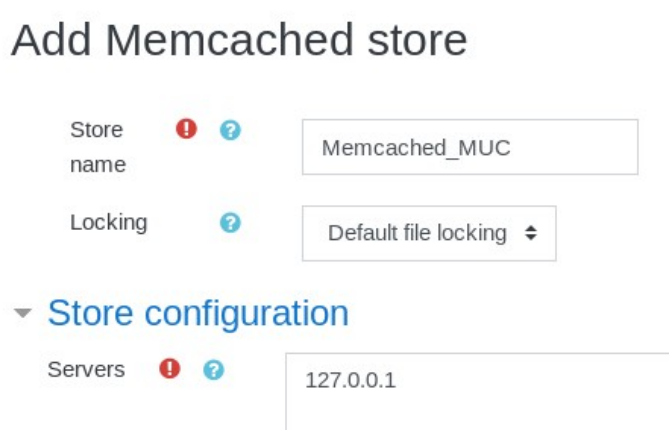
- คอนฟิกให้ Moodle ใช้ Memcached เป็น caching โดย**ไปที่เมนู Configuration**

Cache administration

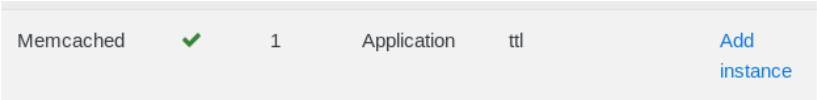
Installed cache stores

Plugin	Ready	Stores	Modes	Supports	Actions
APC user cache (APCu)		0	Application, Session	ttl, key awareness	
File cache	✔	1	Application, Session	data guarantee, ttl, key awareness	Add instance
Memcached	✔	0	Application	ttl	Add instance

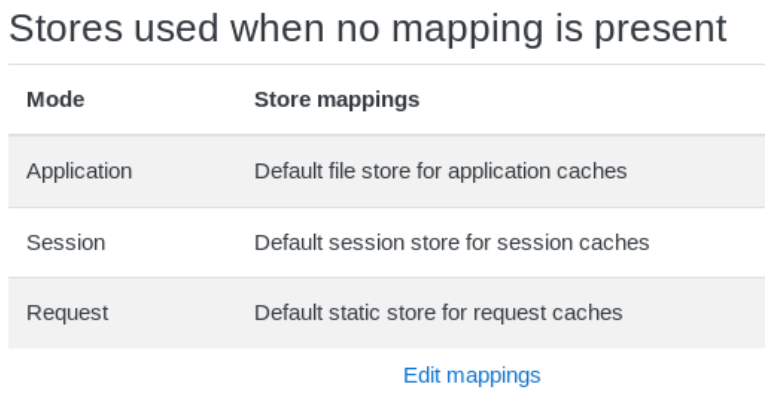
ดูที่ Installed cache stores ที่ Memcached เห็นเครื่องหมายถูกที่ช่อง Ready แสดงว่าคุยกับ memcached ได้ ที่คอลัมน์ Stores แสดงค่า 0 แสดงว่ายังไม่มี instance คอนฟิกไว้เลย ให้**กดที่ "Add instance"** เพื่อเพิ่ม instance



เมื่อกด **Add instance** เข้ามา ก็ให้**ตั้งชื่อ store name** ในที่นี้ตั้งว่า Memcached_MUC และ**กำหนดค่าไอพีแอดเดรสของเครื่องที่ให้บริการ Memcached ที่ช่อง Servers** ในที่นี้คือเครื่องเดียวกับเว็บเซิร์ฟเวอร์เลยอ้างไอพี loop back 127.0.0.1 ไม่จำเป็นต้องระบุพอร์ตถ้าใช้พอร์ตมาตรฐานของ Memcached (11211) **กดปุ่ม Save Changes**



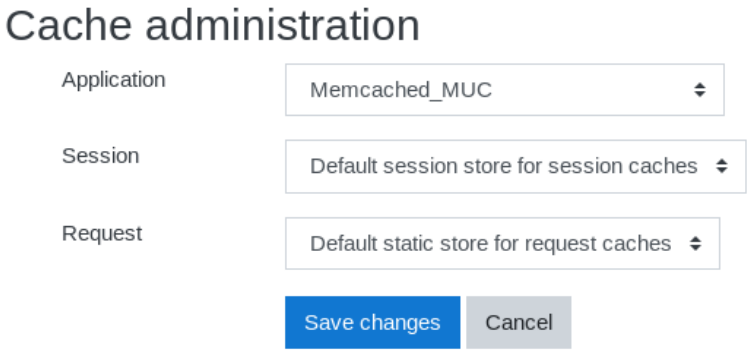
กลับมาที่หน้า Configuration ที่หัวข้อ Installed cache stores จะเห็นว่า Memcached คอลัมน์ Stores แสดงค่าเป็น 1 แล้วนั่นมีอยู่ 1 instance แล้วจากที่เราคอนฟิกไป



จากนั้นให้เลื่อนมาท้ายสุดของหน้าจอ ที่หัวข้อ Stores used when no mapping is present จะแสดงรายการตามรูปต่อไปนี้

สังเกตที่ Application จะเป็น file store ในส่วนนี้แหละที่เราจะเปลี่ยนเป็น memcached แทน

คลิกที่ลิงก์ Edit mappings



กำหนดค่าให้ Application cache ใช้ Memcached_MUC instance ที่เราสร้างไว้เป็น Caching ซึ่งจะเร็วกว่าแบบ file store ที่เขียนลงไฟล์ เพราะ Memcached เก็บข้อมูลบนหน่วยความจำ(แรม)

บนหน้าคอนฟิกนี้เราไม่สามารถกำหนดให้ใช้ Memcached กับ session ได้ แต่เรากำหนดโดยใช้วิธีไประบุใน config.php ของ Moodle แทนดังรายละเอียดในหัวข้อถัดไป

Configured store instances

Store name	Plugin	Ready	Store mappings	Modes	Supports	Locking	Actions
Memcached_MUC	Memcached	✔	0	Application	ttl	Default file locking	Edit store, Delete store, Purge

ที่กรอบ Configuration store instances ในหน้า Configuration จะแสดงชื่อ Store name ที่เราตั้งไว้ (**Memcached_MUC**) สามารถกด Edit store เพื่อเข้าไปดูค่าที่คอนฟิกไว้ได้ เพียงเท่านี้ Moodle ก็จะใช้ memcached เป็นแคชชิ่งสำหรับแอปพลิเคชั่นแล้ว

7. กำหนดให้ Moodle ใช้ Memcached เก็บ session

ให้เพิ่มคอนฟิกต่อไปนี้ที่ไฟล์ config.php ของ moodle เพื่อให้ Moodle ใช้ Memcached ทำ session cache (อ้างอิง https://docs.moodle.org/39/en/Session_handling)

```
$CFG->session_handler_class = '\core\session\memcached';
$CFG->session_memcached_save_path = '127.0.0.1:11211';
$CFG->session_memcached_prefix = 'memc.sess.key.';
$CFG->session_memcached_acquire_lock_timeout = 120;
$CFG->session_memcached_lock_expire = 7200; // Ignored if memcached extension <= 2.1.0
```

4.2 Redis

1. ติดตั้ง Redis

```
yum install -y redis
```

2. คอนฟิก Redis ที่ไฟล์ /etc/redis.conf

```
maxmemory 256mb
requirepass [PASSWORD]
```

** ถ้า bind ไอพี ต้องตั้งรหัสผ่าน ไม่งั้นจะ bind แค่ loopback บน Moodle ขึ้น Exception - Connection refused

3. เริ่มการทำงาน Redis

```
systemctl enable --now redis
```

4. ทดสอบ Redis ที่ command-line

```
redis-cli
AUTH [PASSWORD]
keys *
```

5. ติดตั้ง PHP extension สำหรับ Redis

```
dnf --enablerepo=PowerTools -y install php-pear php-devel zlib-devel make
yum install -y libzstd-devel php-json
pecl install redis
echo 'extension=redis.so' >> /etc/php.d/20-redis.ini
```

รีโหลด php-fpm เพื่อให้รู้จักคอนฟิกกูเรชั่นที่เพิ่มเข้าไปใหม่

```
systemctl reload php-fpm
```

ทดสอบโดยการสร้างไฟล์ phpinfo() เปิดดูควรพบส่วนของ redis

6. คอนฟิกให้ Moodle ใช้งาน Redis เป็น cache ให้ล้อตามข้อ 6. ของหัวข้อ Memcached ก่อนหน้า
7. เพื่อประสิทธิภาพของ Redis ให้กำหนดค่า Linux kernel overcommit memory setting to 1 ทำได้โดยสั่ง

```
sudo sysctl vm.overcommit_memory=1
```

การสั่งตามข้างต้น เกิดผลเฉพาะครั้งนั้น หากมีการรีบูตเครื่องค่าจะกลับไปเป็นค่า default กำหนดให้ถาวรได้ด้วยการเขียนบรรทัด vm.overcommit_memory = 1 ใส่ไฟล์ /etc/sysctl.d/redis.conf

5. ปรับแต่ง NGINX

5.1 ปรับแต่งตัวแปรเพื่อให้รับคอนเน็กชั่นพร้อม ๆ กันได้มากขึ้น

ตัวแปร	**คำอธิบาย**
worker_processes auto;	โดยหลักคือ 1 worker process ต่อ cpu cores ถ้าเครื่องมี 20 core ระบบก็จะสร้าง 20 process
worker_connections 4096;	จำนวน TCP sessions สูงสุดต่อ worker
worker_rlimit_nofile 4096;	Changes the limit on the maximum number of open files (RLIMIT_NOFILE) for worker processes. Used to increase the limit without restarting the main process.

จะใช้ตัวแปร worker_rlimit_nofile ตามข้างต้น ต้องปรับลิมิตในการเปิดไฟล์ของระบบปฏิบัติการด้วยการเพิ่มไฟล์คอนฟิก /etc/security/limits.d/nginx.conf แล้วใส่บรรทัดต่อไปนี้ลงไป

```
nginx soft nofile 4096
```

ดูการค่าว่าเปลี่ยนแปลงหรือไม่ สั่ง

```
sudo -u nginx ulimit -Sn
```

5.2 เปิดฟังก์ชั่น GZIP เพื่อบีบอัดข้อมูลที่ส่งมายัง Client

สร้างไฟล์ /etc/nginx/conf.d/gzip.conf แล้วใส่คำสั่งต่อไปนี้

```
gzip on;
gzip_proxied any;
gzip_types text/plain text/xml text/css application/x-javascript;
gzip_vary on;
gzip_disable “MSIE [1-6]\.(?!.*SV1)”;
```

จากนั้นสั่งให้ nginx อ่านคอนฟิกใหม่

```
systemctl reload nginx
```

5.3 ปรับเวลา timeout ให้ NGINX

แก้ไขไฟล์ /etc/nginx/conf.d/moodle.conf (https://docs.moodle.org/39/en/Nginx) เพิ่มตัวแปร fastcgi_read_timeout เข้าไป จากค่า default 60 วินาที แล้วแต่เราว่าจะปรับเท่าไหร่ แต่ขอให้ปรับสอดคล้องกันกับ PHP ด้วย ในตัวอย่างปรับเป็น 3 นาที แก้ไขไฟล์คอนฟิกอย่าลืมรีโหลดคอนฟิกด้วย

```
fastcgi_read_timeout 180s;
```

6. ปรับแต่ง PHP-FPM

6.1 ปรับจำนวนโปรเซส

โดยปรกติ PHP-FPM จะทำงานในโหมด dynamic อยู่แล้ว แต่กำหนดจำนวนโปรเซสที่เริ่มทำงานไว้ไม่มาก ถ้าระบบเราจะต้องรองรับผู้ใช้งานจำนวนมากพร้อม ๆ กัน การกำหนดให้มีโปรเซสที่เริ่มทำงานในจำนวนเหมาะสมย่อมดีกว่า หากเป็นระบบที่คนเข้าใช้พร้อม ๆ กันมากอย่างต่อเนื่องจริง ๆ ควรปรับเป็นแบบ static ระบบจะสร้างโปรเซสที่ max_children เลยตั้งแต่เริ่มการทำงาน แต่ต้องไม่ลืมว่าทุกโปรเซสใช้ทรัพยากรของระบบ (CPU + RAM) โปรดหาจุดที่เหมาะสมกับระบบของท่าน

ต่อไปนี้เป็นตัวอย่างการปรับแต่ไฟล์ /etc/php-fpm.d/www.conf กรอบทางซ้ายเป็นค่าเริ่มต้นที่ระบบกำหนดมาเมื่อติดตั้ง กรอบขวาเป็นตัวอย่างการปรับแต่ง

```
# Default
pm = dynamic
pm.max_children = 50
pm.start_servers = 5
pm.min_spare_servers = 5
pm.max_spare_servers = 35
```

```
# Tuning
pm = dynamic
pm.max_children = 2000
pm.start_servers = 300
pm.min_spare_servers = 100
pm.max_spare_servers = 500
```

จะอธิบายกรอบทางขวามือ เมื่อเริ่มการทำงาน php-fpm จะสร้างโปรเซสมารอให้บริการ 302 โปรเซส (โปรเซสแม่ 2)

ถ้ามีคอนเน็กชั่นใช้งานระบบเข้ามาเรื่อย ๆ จนกระทั่งมีโปรเซสที่ว่างเหลือไม่ถึง 100 โปรเซส (min_spare_servers) ระบบจะสร้างโปรเซสเพิ่มขึ้นเพื่อให้มีโปรเซสที่ว่างพร้อมรอให้บริการอย่างน้อย 100 โปรเซส แต่จะสร้างสูงสุดโดยมีจำนวนโปรเซสรวมไม่เกิน 2000 โปรเซส(max_children)

เมื่อจำนวนคอนเน็กชั่นลดน้อยลง มีโปรเซสว่างมาก ระบบจะทำลาย(kill)โปรเซสลงให้เหลือไม่เกิน 500 โปรเซส (max_spare_servers)

ในโหมด pm=static ระบบจะสร้างโปรเซสมารอไว้เลยตามจำนวนที่กำหนดในตัวแปร pm.max_children แบบนี้เหมาะสำหรับระบบที่โหลดหนักมาก ๆ มีคนใช้งานพร้อม ๆ กันจำนวนมากตลอดเวลา จนไม่สามารถรอสร้างโปรเซสใหม่ได้ทัน แต่อย่าลืมว่าทุกโปรเซสมีต้นทุน คือ ทรัพยากรเครื่อง(ซีพียู+แรม) เปิดไว้เยอะไม่มีคนใช้ก็เปลือง เปิดไว้ไม่พอผู้ใช้เข้ามาพร้อมกันมากก็สร้างโปรเซสใหม่ไปให้บริการไม่ทัน ผู้ใช้จะพบหน้า Error ก็จะดูไม่ดี หมั่น monitor ระบบแล้วกำหนดค่าให้เหมาะสมจะดีที่สุด

6.2 ปรับค่า timeout ในการประมวลผล PHP

ในช่วงที่ระบบทำงานหนักการประมวลอาจจะเสร็จช้า การที่เราขยาย timeout ก็จะช่วยขยายเวลารอให้ระบบประมวลผลมากขึ้น แต่ในทางตรงกันข้ามผู้ใช้งานก็จะรู้สึกว่ารอนานเช่นกัน ทั้งยังคงทำให้เกิด connection ค้างอยู่ในระบบมากขึ้นด้วย ควรพิจารณาในการปรับ ค่า Default คือ 60 วินาที ถ้าจะปรับก็แก้ที่ไฟล์ /etc/php.ini เช่น

```
max_execution_time = 120
```

แก้ไขแล้วอย่าลืมที่จะรีสตาร์ทหรือรีโหลด php-fpm เพื่อให้ระบบปรับตามค่าคอนฟิกใหม่

```
systemctl restart php-fpm
```

7. การทดสอบประสิทธิภาพเว็บเซิร์ฟเวอร์ด้วย Apache Benchmark

การจะทราบว่าค่าที่เราปรับแต่งนั้นส่งผลต่อระบบอย่างไร ดีขึ้นไหม ถ้าไม่ดูจากของจริงก็ต้องทดลองในส่วนนี้ แนะนำโปรแกรมง่าย ๆ คือ Apache Benchmark คำสั่งที่ใช้คือ ab ซึ่งก็ย่อมาตรงตัว การวัดประสิทธิภาพนี้ทำเปรียบเทียบกันก่อนและหลังปรับแต่งประสิทธิภาพแล้วก็จะทำให้เห็นความเปลี่ยนแปลง มาดูตัวอย่างคำสั่งกัน

ab -n 500 -c 10 http://localhost/

คำสั่งข้างต้นเป็นการบอกว่าจะให้มีการทดสอบเรียกหน้าเว็บ 500 ครั้ง(request) โดยเปิดเว็บพร้อมกันครั้งละ 10 Connection จากจุดนี้เราก็ลองประมาณการว่าระบบของเราจะมีคนเข้าใช้พร้อมกัน (หมายถึงเปิดมาในวินาทีนั้นพร้อมกัน) เท่าไหร่

นอกจากนั้นการทดสอบนี้จะทำให้เราเข้าใจถึงค่าที่เราคอนฟิกให้กับ PHP-FPM, NGINX และทรัพยากรของเครื่องเซิร์ฟเวอร์ได้เป็นอย่างดี ขอให้ลองปรับขนาด แล้วดูที่ Log file ทั้ง PHP-FPM (/var/log/php-fpm/error.log) และ NGINX (/var/log/nginx/error.log) พร้อมทั้งดูการใช้ทรัพยากรผ่านซอฟต์แวร์มอร์นิเตอริ่ง หรือ ถ้าง่าย ๆ ก็คำสั่ง htop เปิดไว้ระหว่างทดสอบจะได้เห็นว่าใช้ซีพียูและแรมไปแค่ไหน

ในการปรับแต่งระบบจำเป็นอย่างยิ่งที่จะต้องปรับค่าเหล่านั้นให้สอดคล้องกับทรัพยากรที่ระบบมี

8. ปรับแต่ง MySQL/MariaDB

ค่า Default ที่ได้มาพร้อมกับการติดตั้ง MySQL/MariaDB นั้นรับภาระงานได้ไม่มาก เช่น ค่า max_connections=151 และ ค่า innodb_buffer_pool_size = 128M สองค่านี้มีผลอย่างมากต่อการให้บริการในภาวะที่มีผู้ใช้งานฐานข้อมูลพร้อม ๆ กันจำนวนมาก

```
[mysqld]
max_connections=1000              # default 151
innodb_buffer_pool_size = 1024M   # default 128M
innodb_flush_method = O_DIRECT    # In MySQL 8.0, use numerically O_DIRECT = 4.

character-set-server=utf8
collation-server=utf8_thai_520_w2
```

หัวใจสำคัญในการปรับแต่งให้ MariaDB นี้รับโหลด มีหัวใจหลักอยู่ที่ 2 เรื่อง (ในนี้กล่าวถึง storage engine มาตรฐานคือ InnoDB) คือ InnoDB Buffer Pool Size และ Max Connection สำหรับ innodb_flush_method = O_DIRECT นั้น กำหนดได้เฉพาะบนระบบปฏิบัติการลีนุกซ์และยูนิกซ์เท่านั้น

8.1 InnoDB Buffer Pool

พื้นที่ในหน่วยความจำที่จะใช้เก็บข้อมูลและอินเด็กซ์ เพื่อลดภาระให้กับดิสก์ แทนที่จะต้องไปอ่านหรือเขียนที่ดิสก์ ก็มาอ่านหรือเขียนที่หน่วยความจำซึ่งมีความเร็วสูงกว่าดิสก์แทน ถ้าตรงนี้มีมากพอก็จะทำให้คล้าย ๆ เป็น in-memory database **จะกำหนดขนาดตัวแปรนี้ต้องมีแรม (RAM) เพียงพอ**ไม่เช่นนั้นจากเร็วจะกลายเป็นช้าแทน

มีตัวแปรที่เกี่ยวข้องกันที่ควรรู้ดังต่อไปนี้

```
innodb_buffer_pool_size=          # default 128M
innodb_buffer_pool_instances =    # default 8 (or 1 if innodb_buffer_pool_size < 1GB)
innodb_buffer_pool_chunk_size =   # default 128M
```

1. innodb_buffer_pool_size คือขนาดของหน่วยความจำที่แบ่งมาทำ Buffer ถ้าตัวแปรนี้น้อยกว่า 1 GB จะส่งผล innodb_buffer_pool_instances มีค่าเป็น 1 แต่ถ้ามีค่าตั้งแต่ง 1GB ขึ้นไป instances จะมีค่าเป็น 8 ซึ่งแตกต่างกันมาก innodb_buffer_pool_instances จะเป็นเธรดที่ค่อยทำงานจัดการ buffer pool ซึ่งแน่นอนค่า Default กำหนด innodb_buffer_pool_size ไว้ 128MB ฉะนั้นควรอย่างยิ่งที่จะปรับค่า innodb_buffer_pool_size เป็น 1GB เป็นอย่างน้อย เพื่อให้ Instance ทำงานที่ 8 Instances
2. innodb_buffer_pool_chunk_size เป็นขนาดของหน่วยความจำ (Buffer pool) ที่ถูกแบ่งออกเป็นส่วน ๆ ตามค่า default คือ 128MB ถ้าเรากำหนด innodb_buffer_pool_size เป็น 1GB, Buffer Pool Instance ก็ทำงานที่ 8 Instances เท่ากับว่า แต่ละ instace จัดการ 1 Chunk (128MB * 8 = 1024MB)

ถ้ากำหนด innodb_buffer_pool_size เป็น 2GB แต่ละ instace จัดการ 2 Chunk (128MB * 2 * 8 = 2048MB) จุดนี้นำมาซึ่งกฎที่ว่า ขนาดของ innodb_buffer_pool_size จะต้องสอดคล้องกับ innodb_buffer_pool_instances * innodb_buffer_pool_chunk_size ถ้าเรา ปรับ innodb_buffer_pool_size ขนาดไม่ลงตัว เช่น กำหนด innodb_buffer_pool_size=3000MB ระบบจะปรับขนาดให้อัตโนมัติเป็น 3072 เพื่อให้แบ่งกันลงตัว

3. ถ้าเครื่องนี้ทำงานเฉพาะฐานข้อมูลเลย ก็อาจจะกำหนด innodb_buffer_pool_instances ให้เพิ่มขึ้นได้ตามจำนวน thread ที่เครื่องรองรับ เช่น เครื่อง 8 cores เมื่อเปิด hyperthreading จะได้ 16 threads ก็อาจจะกำหนด innodb_buffer_pool_instances = 12 เหลือ 4 threads ไว้ทำงานของระบบปฏิบัติการ
4. innodb_buffer_pool_size มีมากเพียงพอยิ่งดี แต่ถ้าไม่รู้จะเริ่มกำหนดที่เท่าไหร่ ก็อาจจะกำหนดให้พอดีกับขนาดของฐานข้อมูล Moodle แล้วค่อย Monitor ดูว่าเหมาะหรือไม่ สามารถดูขนาดของฐานข้อมูลได้ด้วยคำสั่งต่อไปนี้

```
SELECT table_schema "moodle",
        ROUND(SUM(data_length + index_length) / 1024 / 1024, 1) "DB Size in MB"
FROM information_schema.tables
GROUP BY table_schema;
```

** แต่ทั้งนี้ก็ต้องไม่กำหนดเกินกว่าแรมที่เครื่องมี ควรดูให้ดีก่อนว่าระบบปฏิบัติการใช้งานแรมเท่าไหร่ ต้องเผื่อเรื่อง buffer ต่าง ๆ ของระบบปฏิบัติการด้วย ถ้าแรมเยอะเรากำหนดมากหน่อยได้ ถ้าแรมน้อยก็คงต้องกำหนดน้อยหน่อย กำหนดแล้วลอง monitor ดูให้ดี ให้ปริ่มแรม แต่ไม่แตะ swap ได้เป็นแจ๋ว

ดูค่าที่กำหนดด้วยคำสั่ง

```
show global variables like 'innodb_%';
```

8.2 ตัวแปรอื่น ๆ เพิ่มเติม

- max_connections กำหนดให้พอกับโหลดที่จะเข้ามา และโหลดที่เครื่องเซิร์ฟเวอร์เรารับได้ benchmark ทดสอบดูก่อนได้ MySQL มี mysqlslap มาให้ใช้ทดสอบ
- innodb_io_capacity ค่า default กำหนดมา 200 หากดิสก์เรามีค่า number of I/O operations per second (IOPS) สูง เราสามารถปรับตัวแปรนี้เพิ่มขึ้นได้ เช่นเราใช้ SSD อาจปรับเป็น 2000 ได้
- innodb_log_buffer_size = 32M # default 16M
- innodb_log_file_size = 256M # default 48M

9. ตั้งเวลาการทำงานของสคริปต์ตามที่ Moodle กำหนด

ทำตามคู่มือ https://docs.moodle.org/39/en/Cron กำหนดให้ตั้ง cron ทุกนาที ในที่นี้เราใช้ NGINX เป็นเว็บเซิร์ฟเวอร์ มี Owner ที่รันชื่อ nginx เลยกำหนด crontab ในสิทธิ์ของ nginx

```
crontab -u nginx -e
```

cron ใช้ interface แบบ vi กดปุ่ม I เพื่อพิมพ์ เขียนบรรทัดต่อไปนี้ลงไป โดยแก้ไข path ไปยังตำแหน่งที่ติดตั้ง Moodle ที่ถูกต้อง

```
* * * * * /usr/bin/php  /path/to/moodle/admin/cli/cron.php >/dev/null
```

สามารถดูสถานะของ cron ได้โดยสั่ง

```
crontab -u nginx -l
```

10. ซอฟต์แวร์ตรวจตราระบบ (Monitoring Tools) เป็นสิ่งจำเป็น

สำหรับระบบนี้แนะนำเป็น percona monitoring ซึ่งสามารถดูได้ตั้แต่ทรัพยากรเครื่องทั่วไปแล้ว ยังสามารถแสดงรายละเอียดของซอฟต์แวร์ฐานข้อมูลตระกูล MySQL ได้อย่างละเอียดอีกด้วย ทำให้เราเห็นภาพรวมการทำงานได้เป็นอย่างดี วิธีการติดตั้งมีตามลิงค์ต่อไปนี้ https://www.percona.com/software/pmm/quickstart

11. ส่งท้าย

เอกสารนี้อาจไม่ละเอียดนัก เดิมตั้งใจไว้ว่าจะกล่าวถึงเฉพาะการปรับแต่งประสิทธิภาพ แต่มาคิดว่าเพิ่มส่วนติดตั้งด้วยจะเป็นประโยชน์มากขึ้น ผู้เขียนเชื่อว่าการปรับแต่งประสิทธิภาพรวมกับการแยกเว็บกับฐานข้อมูลออกจากกัน ใช้เครื่องเซิร์ฟเวอร์ที่มีทรัพยากรเหมาะสม การรองรับผู้ใช้งาน moodle พร้อมกันพันต้น ๆ นั้นทำได้แน่นอน จากประสบการณ์ผู้ใช้ไม่ได้เข้ามาพร้อม ๆ กัน ทยอยเข้ามาเรียนในวิชาเดียวกัน ผู้ใช้อาจจะมาก แต่รายวิชาน้อยกว่ามาก ตรงนี้พอเราทำ Caching ทำ Buffer ในส่วนต่าง ๆ ไว้อย่างพอเพียง จะช่วยให้ระบบสามารถให้บริการได้ นั้นหมายถึงมีหน่วยความจำหรือ RAM อย่างเพียงพอ และใช้ดิสก์ที่เร็วมีค่า IOPS สูงอย่าง SSD ก็จะช่วยได้มาก

ถ้าจะรองรับปริมาณผู้ใช้ที่มากกว่านี้เราก็ควรรู้ว่าภาระงานตกกับส่วนไหน จะได้ขยายส่วนนั้น ภาษาของชาวคณะผู้เขียนเรียกกว่าคอขวดเกิดที่ไหนก็ไปแก้ที่นั่น การจะรู้ว่าคอขวดเกิดขึ้นที่ไหน ต้องมีเครื่องมือช่วย นั่นคือ Monitoring Software ที่กล่าวไป แรมหรือซีพียูหมดเรารู้ แต่ซอฟต์แวร์ไหนที่ใช้หมด จะได้ขยายส่วนนั้นกระจายข้ามเครื่องทำ Load Balanced ได้ ซึ่งทำได้ทั้งเว็บ ฐานข้อมูลและระบบไฟล์ ก็จะเป็นเรื่องที่ได้ศึกษาได้กล่าวถึงในโอกาสถัด ๆ ไป

12. อ้างอิง

- Moodle Session Handling https://docs.moodle.org/39/en/Session_handling
- Memcached : Use it on PHP https://www.server-world.info/en/note?os=CentOS_8&p=memcached&f=4
- Install and Configure Redis on CentOS 7 https://www.linode.com/docs/databases/redis/install-and-configure-redis-on-centos-7/
- Background saving fails with a fork() error under Linux even if I have a lot of free RAM! https://redis.io/topics/faq#background-saving-fails-with-a-fork-error-under-linux-even-if-i-have-a-lot-of-free-ram
- Tuning NGINX for Better Performance https://rollout.io/blog/tuning-nginx/
- NGINX Core functionality http://nginx.org/en/docs/ngx_core_module.html