Optimizing Moodle 5.x on Ubuntu 24: A Complete Performance Guide

Stack: Ubuntu 24.04 LTS ·MariaDB ·Moodle 5.2+ ·PHP 8.3 ·Apache

---

Introduction

Moodle 5.x introduces a significant architectural change: the **`public/` directory** pattern. Unlike previous versions where the entire Moodle root was exposed to the web server, Moodle 5+ requires your web root to point to the `public/` subdirectory. This guide covers a complete production-ready tuning stack that respects this new structure while delivering sub-second page loads under heavy concurrent load.

Prerequisites

- Ubuntu 24.04 LTS (server edition)
- MariaDB 10.11+
- Moodle 5.2+ installed at `/var/www/moodle`
- Apache 2.4+
- Root or sudo access
- Assumed paths:
  - Moodle source: `/var/www/moodle`
  - Web root (DocumentRoot): `/var/www/moodle/public`
  - Moodle data: `/var/moodledata`
  - Moodle config: `/var/www/moodle/config.php`

The Moodle 5.x Public Directory

In Moodle 5+, **never** point Apache's `DocumentRoot` to `/var/www/moodle`. Always point it to `/var/www/moodle/public`. The `config.php` file lives one level above the web root, which prevents direct HTTP access to sensitive credentials.

Your Apache VirtualHost should look like this:

```
<VirtualHost *:443>
    ServerName elearning.yourdomain.com
    DocumentRoot /var/www/moodle/public

    <Directory /var/www/moodle/public>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch "\.php$">
        SetHandler "proxy:unix:/run/php/php8.3-fpm.sock|fcgi://localhost"
    </FilesMatch>

    SSLEngine on
    # ... SSL certificate directives ...
```

```
</VirtualHost>
```

Step 1: PHP 8.3 with PHP-FPM

Why: `mod_php` loads a PHP interpreter into every Apache process, consuming ~80–120 MB per connection. PHP-FPM runs as a separate process pool, reducing memory per user to ~50 MB and allowing Apache to serve static files efficiently with `mpm_event`.

Install PHP 8.3 and FPM

```
sudo apt update
sudo apt install -y php8.3-fpm php8.3-mysql php8.3-curl php8.3-gd \
    php8.3-mbstring php8.3-xml php8.3-zip php8.3-intl \
    php8.3-redis php8.3-opcache php8.3-apcu
```

Tune PHP-FPM

Edit `/etc/php/8.3/fpm/pool.d/www.conf`:

```
[www]
user = www-data
group = www-data
listen = /run/php/php8.3-fpm.sock
listen.owner = www-data
listen.group = www-data

; Static pool: pre-spawn workers to avoid spawn latency
pm = static
pm.max_children = 120
pm.max_requests = 1000
```

> **Rule of thumb:** `pm.max_children` ≈ `Available RAM / 50 MB`. For a 16 GB server, 120–150 children is safe.

Tune OPcache and APCu

Create `/etc/php/8.3/fpm/conf.d/99-moodle-tune.ini`:

```
memory_limit = 512M
max_execution_time = 600

opcache.enable = 1
opcache.memory_consumption = 256
opcache.max_accelerated_files = 10000
opcache.revalidate_freq = 60
opcache.save_comments = 1

apc.enabled = 1
apc.shm_size = 256M
```

Restart FPM:

```
sudo systemctl restart php8.3-fpm
```

Step 2: Switch Apache to mpm_event + proxy_fcgi

Why: `mpm_prefork` (required by mod_php) creates a full Apache process per connection. `mpm_event` uses threads, dramatically increasing concurrency.

```
# Disable mod_php and prefork
sudo a2dismod php8.3 mpm_prefork

# Enable event MPM and FPM proxy
sudo a2enmod mpm_event proxy proxy_fcgi setenvif
sudo a2enconf php8.3-fpm

# Enable performance modules
sudo a2enmod expires headers rewrite

# Restart
sudo systemctl restart apache2
```

Important: Remove any `php_admin_value`, `php_admin_flag`, `php_value`, or `php_flag` directives from your Apache vhosts. These are mod_php-only and will crash Apache under FPM. Move all PHP settings to `php.ini` or the FPM pool config.

Step 3: MariaDB Tuning

Why: Moodle is read-heavy. A properly sized InnoDB buffer pool caches the entire working set in RAM, eliminating disk I/O.

Edit `/etc/mysql/mariadb.conf.d/50-server.cnf` inside the `[mysqld]` section:

```
[mysqld]
; 50-70% of total RAM for the buffer pool
innodb_buffer_pool_size = 4G
innodb_buffer_pool_instances = 4

; Logging
innodb_log_file_size = 512M
innodb_log_buffer_size = 64M
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = O_DIRECT

; Query cache (useful for Moodle's repeated reads)
query_cache_type = 1
query_cache_size = 128M

; Temp tables
```

```
tmp_table_size = 256M
max_heap_table_size = 256M

; Connection handling
table_open_cache = 8000
thread_cache_size = 100

; Security: bind only to localhost
bind-address = 127.0.0.1
```

> **Sizing note:** If your Moodle database is under 2 GB (check with `mysqltuner`), 4 GB is ample. If your DB grows beyond 8 GB, scale the buffer pool to 8–16 GB.

Restart MariaDB:

```
sudo systemctl restart mariadb
```

Step 4: Redis for Sessions and Application Cache

Why: File-based sessions create disk I/O bottlenecks during concurrent logins. Redis offloads sessions and Moodle's MUC (Multi-Use Cache) to memory.

Install and configure Redis

```
sudo apt install -y redis-server
sudo systemctl enable redis-server
```

Edit `/etc/redis/redis.conf`:

```
bind 127.0.0.1
protected-mode yes
maxmemory 1gb
maxmemory-policy allkeys-lru
```

Restart:

```
sudo systemctl restart redis-server
```

Configure Moodle `config.php`

Add **before** the `require_once(__DIR__ . '/lib/setup.php');` line:

```
$CFG->session_handler_class = '\core\session\redis';
$CFG->session_redis_host = '127.0.0.1';
$CFG->session_redis_port = 6379;
$CFG->session_redis_database = 0;
$CFG->session_redis_acquire_lock_timeout = 120;
$CFG->session_redis_lock_expire = 7200;
```

Step 5: tmpfs RAM Disk for Temporary Files

Why: Moodle constantly writes cached CSS, compiled templates, and backup temp files. A RAM disk (`tmpfs`) eliminates physical disk writes for ephemeral data.

```
sudo mkdir -p /mnt/moodle-tmp
sudo mount -t tmpfs -o size=4G,mode=1777 tmpfs /mnt/moodle-tmp
```

Make it permanent in `/etc/fstab`:

```
tmpfs /mnt/moodle-tmp tmpfs size=4G,mode=1777 0 0
```

Point Moodle to it in `config.php`:

```
$CFG->tempdir = '/mnt/moodle-tmp';
```

Step 6: Moodle GUI — Map MUC Caches to Redis

Log in as admin and navigate to:

Site Administration → Server → Caching

1. **Add a cache store instance:**
   - Store: **Redis**
   - Name: `redis_local`
   - Server: `127.0.0.1:6379`
2. **Map cache types:**
   - **Application** → `redis_local`
   - **Session** → `redis_local`
   - **Request** → `default_application` (or `redis_local`)
3. Save and purge caches:

```
sudo -u www-data php /var/www/moodle/admin/cli/purge_caches.php
```

Step 7: Moodle Application Tuning

Reduce quiz autosave frequency

Quiz autosave fires an AJAX request every 60 seconds by default. With 500 concurrent quiz takers, that is 500 background requests per minute.

```
sudo -u www-data php /var/www/moodle/admin/cli/cfg.php --name=quizautosaveperiod --set=120
```

This halves the background load with minimal user impact.

Disable debug mode in production

```
sudo -u www-data php /var/www/moodle/admin/cli/cfg.php --name=debug --set=0
sudo -u www-data php /var/www/moodle/admin/cli/cfg.php --name=debugdisplay --set=0
```

Ensure cron runs via CLI

Web-triggered cron is slow and creates unnecessary HTTP requests. Use the system cron:

```
sudo crontab -u www-data -e
```

Add:

```
* * * * * /usr/bin/php /var/www/moodle/admin/cli/cron.php >/dev/null 2>&1
```

Step 8: Firewall & Security Hardening

A fast server is useless if it is compromised. Lock down the perimeter:

```
# Default policies
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Allow only necessary services
sudo ufw allow 22/tcp      # SSH (change to your actual port)
sudo ufw allow 80/tcp      # HTTP
sudo ufw allow 443/tcp     # HTTPS

# If MariaDB is accessed from another server only:
sudo ufw allow from 192.168.1.10 to any port 3306

# Enable
sudo ufw enable
```

Never expose these to the internet:

- Port 3306 (MariaDB) — unless IP-restricted
- Port 6379 (Redis) — must stay on localhost
- Ports 139/445 (SMB) — remove immediately if present

Apache security headers

Add to your SSL VirtualHost:

```
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-Content-Type-Options "nosniff"
Header always set Referrer-Policy "strict-origin-when-cross-origin"
Header always set Permissions-Policy "geolocation=(), microphone=(), camera=()"
```

Enable the module:

```
sudo a2enmod headers
```

Hide server version banners

Edit `/etc/apache2/conf-available/security.conf`:

```
ServerTokens Prod
ServerSignature Off
```

```
sudo a2enconf security
sudo systemctl restart apache2
```

Protect config.php

```
sudo chmod 640 /var/www/moodle/config.php
sudo chown root:www-data /var/www/moodle/config.php
```

Verification Checklist

After completing all steps, verify with these commands:

```
# PHP-FPM is running with 120 workers
sudo systemctl status php8.3-fpm --no-pager

# Apache is using mpm_event, not prefork
sudo apache2ctl -M | grep -E "mpm|proxy|fcgi|php"

# MariaDB buffer pool is sized correctly
sudo mysql -e "SHOW VARIABLES LIKE 'innodb_buffer_pool_size';"

# Redis is responding
redis-cli ping

# tmpfs is mounted
df -h | grep moodle-tmp

# Moodle caches purge successfully
sudo -u www-data php /var/www/moodle/admin/cli/purge_caches.php
```

Expected results:

- `php8.3-fpm` → `active (running)`
- `mpm_event_module (shared)` present, **no** `php_module`
- FPM handler line present in your vhost
- Redis returns `PONG`
- No `php_admin_flag/value` lines in vhosts

Performance Expectations

Metric	Before (mod_php)	After (PHP-FPM + Redis)
Safe concurrent quiz takers	~150–200	500+
Memory per connection	~100 MB	~50 MB
Session storage	Disk I/O	RAM (Redis)
DB queries	Disk reads	RAM cache (InnoDB)
Temp file writes	Physical disk	RAM disk (tmpfs)

With this stack, a single 16 GB server can comfortably handle **500 simultaneous quiz attempts**. Scale horizontally behind a load balancer when you exceed that threshold.

Rollback Plan

If anything breaks, revert to mod_php quickly:

```
sudo a2dismod mpm_event proxy_fcgi proxy setenvif
sudo a2disconf php8.3-fpm
sudo a2enmod mpm_prefork php8.3
sudo systemctl stop php8.3-fpm
sudo systemctl restart apache2
```

Conclusion

Moodle 5.x's `public/` directory is a security win, but it does not change the performance fundamentals. The bottleneck in most self-hosted Moodle instances is not the database size — it is the web server architecture. Moving from `mod_php` to `PHP-FPM + mpm_event`, caching the database in MariaDB's buffer pool, and offloading sessions to Redis transforms a sluggish LMS into one that handles hundreds of concurrent users without breaking a sweat.

Test your changes in a staging environment first. Monitor with `htop`*,* `iotop`*, and MariaDB's* `SHOW PROCESSLIST` *during peak load.*