

CONFIGURING MOD_FCGID ON WINDOWS APACHE 2.4 FOR HIGH-CONCURRENCY WORKLOADS

This guide provides an end-to-end, production-focused setup for `mod_fcgid` on Windows with Apache HTTP Server 2.4, optimized for high concurrency. It covers installation, sizing, tuning, validation, and troubleshooting. Examples emphasize PHP via `php-cgi.exe`, but the methodology applies to other FastCGI apps.

1) PREREQUISITES AND ASSUMPTIONS

- Windows Server 2016/2019/2022 (or Windows 10/11) with Apache 2.4.x installed under *C:/Apache24*.
- Administrative rights to install modules and create directories.
- For PHP: using `php-cgi.exe` (Non-Thread-Safe build) located at *C:/PHP*.
- Antivirus is configured to exclude Apache, PHP, and temp directories from on-access scanning (important for high I/O).

2) INSTALL AND ENABLE MOD_FCGID

1. Download a `mod_fcgid` binary compatible with your Apache 2.4 and VC runtime (e.g., from Apache Lounge). Ensure matching architecture (x64) and Visual C++ runtime.
2. Place **`mod_fcgid.so`** into *C:/Apache24/modules/*.
3. Create a temp and IPC directory for `mod_fcgid`:
C:/Apache24/temp/fcgid and *C:/Apache24/logs/fcgid* (ensure Apache service account has read/write).
4. In **`httpd.conf`**, enable the module and base settings:
`LoadModule fcgid_module modules/mod_fcgid.so`
`FcgidIPCDir "C:/Apache24/temp/fcgid"`
`FcgidProcessTableFile "C:/Apache24/logs/fcgid/fcgid_shm"`
`FcgidWin32PreventOrphans On`
`FcgidFixPathinfo 1` (often required for PHP `PATH_INFO`)
5. Restart Apache to confirm there are no load errors.

3) CORE APACHE (MPM) CONCURRENCY ON WINDOWS

Windows uses the **mpm_winnt** MPM (one process, many threads). Concurrency is primarily controlled via **ThreadsPerChild** and **MaxRequestWorkers**.

- **ThreadsPerChild**: Typically 150-500. Controls worker threads in the single child process.
- **MaxRequestWorkers**: In Windows, equals ThreadsPerChild. Set sufficiently above expected concurrent connections (consider static + dynamic requests).

Example in *httpd-mpm.conf* or *httpd.conf*:

```
ThreadsPerChild 400
MaxRequestWorkers 400
ListenBacklog 1024
```

Set **KeepAlive** judiciously. For high concurrency with many short requests, try:

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 2
```

4) MAPPING APPLICATIONS WITH MOD_FCGID

For PHP via php-cgi.exe (Non-Thread-Safe), in a global context or VirtualHost:

```
<IfModule fcgid_module>
# Map PHP scripts to FastCGI wrapper
AddHandler fcgid-script .php
FcgidWrapper "C:/PHP/php-cgi.exe" .php

# Pass environment to PHP
FcgidInitialEnv PHPRC "C:/PHP"
FcgidInitialEnv PATH "C:/Windows/System32;C:/Windows;C:/PHP"
FcgidInitialEnv TEMP "C:/Apache24/temp"
FcgidInitialEnv TMP "C:/Apache24/temp"
FcgidPassHeader Authorization
</IfModule>
```

Note: On Windows, **PHP_FCGI_CHILDREN** is not used by php-cgi; concurrency is controlled by mod_fcgid process counts.

5) CAPACITY PLANNING AND SIZING

Key parameters to size for high concurrency:

- **MaxRequestWorkers** (Apache threads): upper bound of concurrent connections (static + dynamic).
- **FcgidMaxProcesses**: global limit of all FastCGI processes (all classes).
- **FcgidMaxProcessesPerClass**: limit per wrapper (e.g., per PHP app).
- **FcgidIOTimeout**, **FcgidConnectTimeout**, **FcgidBusyTimeout**: resilience under load.
- **FcgidProcessLifeTime**, **FcgidMaxRequestsPerProcess** (via app env like `PHP_FCGI_MAX_REQUESTS`): recycling.

Simple sizing approach:

1. Estimate peak concurrent dynamic requests (e.g., PHP), call it **CD**.
2. Determine average requests per CGI process simultaneously served: with php-cgi on Windows this is *one* request per process.
3. Set **FcgidMaxProcessesPerClass** $\geq$ **CD** and **FcgidMaxProcesses** $\geq$ sum across apps (with headroom 10–30%).
4. Set **MaxRequestWorkers** $\geq$ **max(static + dynamic)**. If static offload (CDN), lower; otherwise include static.

Example: If 250 concurrent PHP requests are expected and 50 static, choose: MaxRequestWorkers 350, FcgidMaxProcessesPerClass 260, FcgidMaxProcesses 300.

6) RECOMMENDED MOD_FCGID BASELINE FOR HIGH CONCURRENCY

Add to global scope (httpd.conf) or inside the VirtualHost serving the app:

```
# Global FastCGI process limits
FcgidMaxProcesses 400
FcgidMaxProcessesPerClass 350
FcgidMinProcessesPerClass 4
FcgidProcessLifeTime 3600
FcgidMaxRequestInMem 131072 # 128 KB before temp file
FcgidMaxRequestLen 268435456 # 256 MB uploads

# Timeouts and scans
FcgidConnectTimeout 10
FcgidIOTimeout 120
```

```
FcgidBusyTimeout 300
FcgidBusyScanInterval 120
FcgidErrorScanInterval 5
FcgidZombieScanInterval 5

# Output buffering
FcgidOutputBufferSize 65536
FcgidFlushDelay 0

# IPC and Windows resiliency
FcgidWin32PreventOrphans On
FcgidIPCDir "C:/Apache24/temp/fcgid"
```

Notes:

- **FcgidMaxRequestLen** must be large enough for file uploads; align with app limits.
- **FcgidIOTimeout**: must exceed worst-case app time; too low causes 500s.
- **FcgidProcessLifeTime**: recycle to mitigate memory leaks; match with PHP_FCGI_MAX_REQUESTS if used.

7) PHP-SPECIFIC STABILITY SETTINGS

Set in Apache via environment (per vhost or global):

```
FcgidInitialEnv PHP_FCGI_MAX_REQUESTS 1000 # recycle php-cgi
after N requests
FcgidInitialEnv PHP_MAX_EXECUTION_TIME 120
```

Also in php.ini (C:/PHP/php.ini):

```
max_execution_time = 120
memory_limit = 256M (or per sizing)
upload_max_filesize = 128M
post_max_size = 128M
cgi.fix_pathinfo = 1
```

8) VIRTUALHOST EXAMPLE (PRODUCTION)

```
<VirtualHost *:80>
ServerName www.example.com
DocumentRoot "C:/Sites/example/htdocs"
```

```
# Static files (serve directly)
<Directory "C:/Sites/example/htdocs">
Options Indexes FollowSymLinks
AllowOverride All
Require all granted
</Directory>

# PHP via mod_fcgid
AddHandler fcgid-script .php
FcgidWrapper "C:/PHP/php-cgi.exe" .php
FcgidInitialEnv PHPRC "C:/PHP"
FcgidInitialEnv PHP_FCGI_MAX_REQUESTS 1000
FcgidInitialEnv TEMP "C:/Apache24/temp"
FcgidInitialEnv TMP "C:/Apache24/temp"
FcgidPassHeader Authorization

# Per-vhost tuning (optional override)
FcgidMaxProcessesPerClass 260
FcgidMinProcessesPerClass 6
FcgidIOTimeout 120

ErrorLog "logs/example-error.log"
CustomLog "logs/example-access.log" combined
</VirtualHost>
```

9) PERFORMANCE TUNING WORKFLOW

1. **Baseline:** Start with the recommended settings above. Confirm stability with a smoke test.
2. **Load test:** Use a realistic tool (k6, JMeter, wrk) simulating think time, connection reuse, and mixed payloads (static+dynamic).
3. **Measure** under increasing concurrency: p95/p99 latency, error rate, CPU, memory, context switches, and php-cgi.exe count.
4. **Adjust size:**
 - If 503/500 under load with php-cgi saturation: raise FcgidMaxProcessesPerClass and FcgidMaxProcesses (watch RAM).
 - If long queues but spare CPU: increase MaxRequestWorkers and corresponding Fcgid* limits.
 - If memory pressure: reduce processes, shorten ProcessLifeTime, and lower PHP memory_limit or app caches.
5. **Timeouts:** Tune FcgidIOTimeout and BusyTimeout to be slightly above worst-case app times to avoid premature kills.

6. **Recycling:** Set PHP_FCGI_MAX_REQUESTS and FcgidProcessLifeTime to mitigate leaks; stagger restarts under load.

10) WINDOWS-SPECIFIC CONSIDERATIONS

- **File paths:** Use forward slashes or escaped backslashes. Quote paths with spaces.
- **Antivirus:** Exclude C:/Apache24, C:/PHP, site directories, and C:/Apache24/temp from real-time scanning.
- **I/O temp dir:** Place FcgidIPCDir and TEMP on a fast local SSD. Avoid network shares for code or temp.
- **Service account:** Ensure the Apache service account has Modify permissions on temp/log/IPC paths.
- **Event Viewer:** Check Windows Event Viewer for crashes of php-cgi.exe during stress.

11) MONITORING AND OBSERVABILITY

- **mod_status:** Enable server-status to view busy/idle workers and request states.

```
LoadModule status_module modules/mod_status.so  
<Location /server-status> SetHandler server-status Require  
local </Location>
```
- **Logs:** Inspect Apache error.log for mod_fcgid timeouts (IOTimeout, BusyTimeout) and spawn failures.
- **Process count:** Monitor php-cgi.exe instances to confirm FcgidMaxProcessesPerClass behavior.
- **App metrics:** Expose application response times and error rates; correlate with web server metrics.

12) SECURITY AND HARDENING

- **Limit uploads:** Set FcgidMaxRequestLen, post_max_size, and upload_max_filesize consistently.
- **Restrict execution:** Use file system ACLs to limit where CGI binaries can execute.
- **Timeouts:** Prefer conservative timeouts plus app-level circuit breakers.

- **Headers:** Add security headers via `mod_headers` (X-Frame-Options, X-Content-Type-Options, etc.).
- **Isolation:** Consider separate VirtualHosts and wrappers for different apps to isolate resource pools.

13) TROUBLESHOOTING COMMON ISSUES

- **500 with Premature End of Script Headers:** Increase `FcgidIOTimeout`; verify `php.ini` syntax; check PHP error logs.
- **503 Service Unavailable:** `FcgidMaxProcessesPerClass` too low; raise limits or reduce concurrency.
- **Spawn failed / No such file:** Wrong path quoting or permissions; verify `FcgidWrapper` path and service account rights.
- **Stalls on uploads:** Increase `FcgidMaxRequestLen`; ensure temp directories have space and correct permissions.
- **Memory growth:** Lower PHP `memory_limit`; decrease `PHP_FCGI_MAX_REQUESTS` or `FcgidProcessLifeTime` for faster recycling.

14) QUICK REFERENCE: KEY DIRECTIVES AND SUGGESTED RANGES

Directive	Purpose	Suggested Starting Point
<code>FcgidMaxProcesses</code>	Global FastCGI process cap	$1.2 \times$ peak dynamic concurrency
<code>FcgidMaxProcessesPerClass</code>	Per app/wrapper cap	$\approx$ peak app concurrency
<code>FcgidMinProcessesPerClass</code>	Warm pool size	2–8
<code>FcgidIOTimeout</code>	Backend read/write timeout	60–180s (app-dependent)
<code>FcgidBusyTimeout</code>	Kill long-busy processes	180–600s
<code>FcgidProcessLifeTime</code>	Recycle age	1800–7200s
<code>FcgidOutputBufferSize</code>	App output buffer	64 KB
<code>FcgidMaxRequestLen</code>	Max upload size	256 MB (align with app)
<code>ThreadsPerChild</code>	Apache worker threads	200–600
<code>MaxRequestWorkers</code>	Max concurrent connections	Match <code>ThreadsPerChild</code>

15) VALIDATION CHECKLIST

- Apache starts without errors; mod_fcgid loaded.
- server-status shows expected worker counts.
- Load test achieves target concurrency with acceptable p95 latency.
- php-cgi.exe process count scales up to FcgidMaxProcessesPerClass and recycles over time.
- No persistent 500/503 errors under steady load.

APPENDIX: EXAMPLE FULL MINIMAL CONFIG SNIPPET

```
# httpd.conf (relevant parts)
LoadModule fcgid_module modules/mod_fcgid.so
LoadModule status_module modules/mod_status.so
ThreadsPerChild 400
MaxRequestWorkers 400
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 2

FcgidIPCDir "C:/Apache24/temp/fcgid"
FcgidProcessTableFile "C:/Apache24/logs/fcgid/fcgid_shm"
FcgidWin32PreventOrphans On
FcgidFixPathinfo 1
FcgidMaxProcesses 400
FcgidMaxProcessesPerClass 350
FcgidMinProcessesPerClass 6
FcgidIOTimeout 120
FcgidBusyTimeout 300
FcgidConnectTimeout 10
FcgidProcessLifeTime 3600
FcgidOutputBufferSize 65536
FcgidMaxRequestLen 268435456

<Location /server-status> SetHandler server-status Require local
</Location>

# VirtualHost for PHP app
<VirtualHost *:80>
ServerName www.example.com
DocumentRoot "C:/Sites/example/htdocs"
```

```
<Directory "C:/Sites/example/htdocs"> Options Indexes
FollowSymLinks AllowOverride All Require all granted </Directory>
AddHandler fcgid-script .php
FcgidWrapper "C:/PHP/php-cgi.exe" .php
FcgidInitialEnv PHPRC "C:/PHP"
FcgidInitialEnv PHP_FCGI_MAX_REQUESTS 1000
FcgidInitialEnv TEMP "C:/Apache24/temp"
FcgidInitialEnv TMP "C:/Apache24/temp"
FcgidPassHeader Authorization
ErrorLog "logs/example-error.log"
CustomLog "logs/example-access.log" combined
</VirtualHost>
```

NOTES ON ASSUMPTIONS

- Assumed PHP via php-cgi.exe as the most common mod_fcgid workload on Windows; adjust wrappers for other runtimes (Python, Ruby, etc.).
- Assumed Apache installed at C:/Apache24 and PHP at C:/PHP; update paths accordingly.
- Assumed antivirus exclusions and SSD-backed temp/log directories for performance.

David Keith Jr "mrdj" ✓