Apache HTTP Server Version 2.4

Apache MPM worker

Description:	Multi-Processing Module implementing a hybrid multi-threaded multi-process web server
Status:	MPM
Module Identifier:	mpm_worker_module
Source File:	worker.c

Summary

This Multi-Processing Module (MPM) implements a hybrid multi-process multi-threaded server. By using threads to serve requests, it is able to serve a large number of requests with fewer system resources than a process-based server. However, it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

The most important directives used to control this MPM are `ThreadsPerChild`, which controls the number of threads deployed by each child process and `MaxRequestWorkers`, which controls the maximum total number of threads that may be launched.

Topics

- How it Works

Directives

- CoreDumpDirectory
- EnableExceptionHook
- Group
- Listen
- ListenBacklog
- MaxConnectionsPerChild
- MaxMemFree
- MaxRequestWorkers
- MaxSpareThreads
- MinSpareThreads
- PidFile
- ReceiveBufferSize
- ScoreBoardFile
- SendBufferSize
- ServerLimit
- StartServers
- ThreadLimit
- ThreadsPerChild
- ThreadStackSize
- User

Bugfix checklist

- httpd changelog
- Known issues
- Report a bug

See also

- Setting which addresses and ports Apache HTTP Server uses
- Comments

How it Works

A single control process (the parent) is responsible for launching child processes. Each child process creates a fixed number of server threads as specified in the `ThreadsPerChild` directive, as well as a listener thread which listens for connections and passes them to a server thread for processing when they arrive.

Apache HTTP Server always tries to maintain a pool of *spare* or idle server threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new threads or processes to be created before their requests can be served. The number of processes that will initially launch is set by the `StartServers` directive. During operation, the server assesses the total number of idle threads in all processes, and forks or kills processes to keep this number within the boundaries specified by `MinSpareThreads` and `MaxSpareThreads`. Since this process is very self-regulating, it is rarely necessary to modify these directives from their default values. The maximum number of clients that may be served simultaneously (i.e., the maximum total number of threads in all processes) is determined by the `MaxRequestWorkers` directive. The maximum number of active child processes is determined by the `MaxRequestWorkers` directive divided by the `ThreadsPerChild` directive.

Two directives set hard limits on the number of active child processes and the number of server threads in a child process, and can only be changed by fully stopping the server and then starting it again. `ServerLimit` is a hard limit on the number of active child processes, and must be greater than or equal to the `MaxRequestWorkers` directive divided by the `ThreadsPerChild` directive. `ThreadLimit` is a hard limit of the number of server threads, and must be greater than or equal to the `ThreadsPerChild` directive.

In addition to the set of active child processes, there may be additional child processes which are terminating, but where at least one server thread is still handling an existing client connection. Up to `MaxRequestWorkers` terminating processes may be present, though the actual number can be expected to be much smaller. This behavior can be avoided by disabling the termination of individual child processes, which is achieved using the following:

- set the value of `MaxConnectionsPerChild` to zero
- set the value of `MaxSpareThreads` to the same value as `MaxRequestWorkers`

A typical configuration of the process-thread controls in the `worker` MPM could look as follows:

```
ServerLimit          16
StartServers          2
MaxRequestWorkers   150
MinSpareThreads      25
MaxSpareThreads      75
ThreadsPerChild      25
```

While the parent process is usually started as `root` under Unix in order to bind to port 80, the child processes and threads are launched by the server as a less-privileged user. The `User` and `Group` directives are used to set the privileges of the Apache HTTP Server child processes. The child processes must be able to read all the content that will be served, but should have as few privileges beyond that as possible. In addition, unless `suexec` is used, these directives also set the privileges which will be inherited by CGI scripts.

`MaxConnectionsPerChild` controls how frequently the server recycles processes by killing old ones and launching new ones.

This MPM uses the `mpm-accept` mutex to serialize access to incoming connections when subject to the thundering herd problem (generally, when there are multiple listening sockets). The implementation aspects of this mutex can be configured with the `Mutex` directive. The performance hints (↗ ../misc/perf-tuning.html) documentation has additional information about this mutex.

Comments

Notice:
This is not a Q&A section. Comments placed here should be pointed towards suggestions on improving the documentation or server, and may be removed again by our moderators if they are either implemented or considered invalid/off-topic. Questions on how to manage the Apache HTTP Server should be directed at either our IRC channel, #httpd, on Freenode, or sent to our mailing lists.