Apache MPM perchild

Description:	Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids
Status:	MPM
Module Identifier:	mpm_perchild_module
Source File:	perchild.c

Summary

This MPM does not currently work on most platforms. Work is ongoing to make it functional.

This Multi-Processing Module (MPM) implements a hybrid multi-process, multi-threaded web server. A fixed number of processes create threads to handle requests. Fluctuations in load are handled by increasing or decreasing the number of threads in each process.

Topics

Directives

(➟ This directive is defined elsewhere. See: `mpm_common`)

See also

- Setting which addresses and ports Apache uses[1]

How it works

A single control process launches the number of child processes indicated by the `NumServers` directive at server startup. Each child process creates threads as specified in the `StartThreads` directive. The individual threads then listen for connections and serve them when they arrive.

Apache 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 new threads to be created. For each child process, Apache assesses the number of idle threads and creates or destroys threads 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 is determined by multiplying the number of server processes that will be created (`NumServers`) by the maximum number of threads created in each process (`MaxThreadsPerChild`).

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 Apache as a less-privileged user. The `User` and `Group` directives are used to set the privileges of the Apache 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 [2] is used, these directives also set the privileges which will be inherited by CGI scripts.

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

Working with different user-IDs

The `perchild` MPM adds the extra ability to specify that particular processes should serve requests under different user-IDs. These user-IDs can then be associated with specific virtual hosts. You have to use one `ChildPerUserID` directive for every user/group combination you want to be run. Then you can tie particular virtual hosts to that user and group IDs.

The following example runs 7 child processes. Two of them are run under `user1`/`group1`. The next four are run under `user2`/`group2` and the remaining process uses the `User` and `Group` of the main server:

Global config

```
NumServers 7
ChildPerUserID user1 group1 2
ChildPerUserID user2 group2 4
```

Using unbalanced numbers of processes as above is useful, if the particular virtual hosts produce different load. The assignment to the virtual hosts is easily done as in the example below. In conclusion with the example above the following assumes, that `server2` has to serve about twice of the hits of `server1`.

Example

```
NameVirtualHost *

<VirtualHost *>
  ServerName fallbackhost
  # no assignment; use fallback
</VirtualHost>

<VirtualHost *>
  ServerName server1
  AssignUserID user1 group1
</VirtualHost>

<VirtualHost *>
  ServerName server2
  AssignUserID user2 group2
</VirtualHost>
```

AssignUserID Directive

Description:	Tie a virtual host to a user and group ID
Syntax:	`AssignUserID` *`user-id group-id`*
Context:	virtual host

Status:	MPM
Module:	perchild

Tie a virtual host to a specific user/group combination. Requests addressed to the virtual host where this directive appears will be served by a process running with the specified user and group ID.

The user and group ID has to be assigned to a number of children in the global server config using the `ChildPerUserID` directive. See the section above for a configuration example.

ChildPerUserID Directive

Description:	Specify user ID and group ID for a number of child processes
Syntax:	`ChildPerUserID` *`user-id group-id num-children`*
Context:	server config
Status:	MPM
Module:	perchild

Specify a user ID and group ID for a number of child processes. The third argument, *num-children*, is the number of child processes to start with the specified user and group. It does *not* represent a specific child number. In order to use this directive, the server must be run initially as `root`. If you start the server as a non-root user, it will fail to change to the lesser privileged user.

If the total number of child processes, found by totaling all of the third arguments to all `ChildPerUserID` directives in the config file, is less than `NumServers`, then all remaining children will inherit the `User` and `Group` settings from the main server. See the section above for a configuration example.

Security

Don't set *user-id* (or *group-id*) to `root` unless you know exactly what you are doing, and what the dangers are.

MaxThreadsPerChild Directive

Description:	Maximum number of threads per child process
Syntax:	`MaxThreadsPerChild` *`number`*
Default:	`MaxThreadsPerChild 64`
Context:	server config
Status:	MPM
Module:	perchild

This directive sets the maximum number of threads that will be created in each child process. To increase this value beyond its default, it is necessary to change the value of the `ThreadLimit` directive and stop and re-start the server.

NumServers Directive

Description:	Total number of children alive at the same time

Syntax:	`NumServers` *number*
Default:	`NumServers 2`
Context:	server config
Status:	MPM
Module:	perchild

The `NumServers` directive determines the number of children alive at the same time. This number should be large enough to handle the requests for the entire site. To increase this value beyond the value of `8`, it is necessary to change the value of the `ServerLimit` directive and stop and re-start the server. See the section above for a configuration example.

URI References

[1] http://httpd.apache.org/docs-2.1/bind.html
[2] http://httpd.apache.org/docs-2.1/suexec.html