

Enhancements.

Processes and Threads.

Apache 1.3 under Linux used multiple child processes for each connection. Apache 2 uses either child processes, multithreading or a combo.

Each process is a separate instance of the program or another subprogram. This means if a problem occurs with a process it only effects itself. The parent process and other children still run.

In a multithreading enviroment, a section of the program is spun off by it's self to run by itself. This is not a full process but part of the same process just running concurrently with the other threads. It takes less overhead to use a thread than a second instance of the program or a subprogram. But unlike child processes, if an error occurs in a thread the whole process can be affected.

The problem with using child processes for each connection under Apache is that it takes a fair bit more horsepower than using multithreading. As a result it usually requires upgrading hardware as a website grows. Also using child processes under Unix doesn't translate well to Windows so Apache under Windows actually ran slower.

Multithreading allows Apache to get make better use of the hardware it's given. Though not as stable as using child processes, the code is written to be as stable as possible.

prefork – Default. Most Closely imitates behaviour of v1.3. Currently the default for Unix and sites that require stability, though we hope that threading will become the default later on.

threaded (worker) – Suitable for site that require the benefits brought by threading, particularly reduced memory footprint and improved interthread communications.

perchild – is used to assign a different process and userid/groupid to each virtual host. Each different process then uses threading. So this combines both multiple process with multithreading.

Finding out what was used to build your version of Apache

```
$ /usr/sbin/httpd -V
```

```
Server version: Apache/2.0.40
```

```
Server built: Feb 25 2003 05:01:56
```

```
Server's Module Magic Number: 20020628:0
```

```
Architecture: 32-bit
```

```
Server compiled with....
```

```
-D APACHE_MPM_DIR="server/mpm/prefork"
```

```
-D APR_HAS_SENDFILE
```

```
-D APR_HAS_MMAP
```

```
-D APR_HAVE_IPV6
```

```
-D APR_USE_SYSVSEM_SERIALIZE
```

```
-D APR_USE_PTHREAD_SERIALIZE
```

```
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
```

```
-D APR_HAS_OTHER_CHILD
```

```
-D AP_HAVE_RELIABLE_PIPED_LOGS
```

```
-D HTTPD_ROOT="/etc/httpd"
```

```
-D SUEXEC_BIN="/usr/sbin/suexec"  
-D DEFAULT_PIDLOG="logs/httpd.pid"  
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"  
-D DEFAULT_LOCKFILE="logs/accept.lock"  
-D DEFAULT_ERRORLOG="logs/error_log"  
-D AP_TYPES_CONFIG_FILE="conf/mime.types"  
-D SERVER_CONFIG_FILE="conf/httpd.conf"
```

Migrating to Apache 2.

Some of the modules have changed for Apache2. They have a new mod_ssl for instance. The api for writing modules also changed so every module has to be rewritten for Apache 2.

Also some of the configuration options have changed BindAddress has been replaced by Listen while the ServerType, AgentLog and ReferLog have been removed. You can't just use your configuration from Apache 1.3. But all is not lost, CoValent, who sells a commercial version of apache, has a script that will help you move your configuration to version 2.

<http://apache.covalent.net/tools/index.php?PHPSESSID=29f827f6d376dbd297e1b0a093c1ecb1>

```
#!/usr/bin/perl httpd.conf
```

This converts your old httpd.conf file into one more suitable for Apache 2. Though this was not completely successful in converting my httpd.conf into a useable Apache 2 file. As I mentioned some of the plugin modules for Apache have changed. I had to go in by hand to change these values.

Here's the procedure I used.

I copied my current httpd.conf file to my home directory and then ran the confconv.pl file on it. The script provided me with a new version of httpd.conf and a backup ([httpd.conf.bak](#)). I then ran a test on the httpd file to see if there were problems with the new configuration. “/usr/sbin/httpd -t -f ~/httpd.conf” told me some of my modules were incorrect so I had to go in and disable some and fix others. I don't think the fact that the conversion script is missing support for changed modules is a big problem. The script is probably fairly old and names and numbers of Apache 2 modules has changed quite a bit since Covalent released their version last year.

You can find out what modules you have installed by listing the contents of your Apache Modules Directory.

```
$ ls /etc/httpd/modules
```

So after you have fixed your module names and such, I also had to fix my calls to php4 and ssl as they were conditional and not tested by “httpd -t”. After this I was ready to start running my Apache 2 webserver.