

Apache 2.0 and PHP: A Technology Overview Whitepaper

May 2003



Table of Contents

<i>Table of Contents</i>	2
<i>Introduction</i>	3
<i>What's New in Apache 2</i>	4
<i>Deciding to Upgrade to Apache 2</i>	4
<i>Multithreading vs. Multi-Process</i>	5
Apache 2 and Multithreading	5
How to Determine Whether a Server is Running as Multithreaded	5
<i>Decision factors</i>	6
<i>Zend's Recommendation</i>	8

Introduction

Recently, a new major version of the Apache Web server has been released – Version 2. This version already ships as the default Web server in a number of operating system distributions. Due to the fact that the choice of Web server is a key factor regarding the deployment of PHP, we have created this whitepaper about the subject of Apache 2 in general, and its relation with PHP in particular.

Apache 2 Overview

What's New

While detailing all of the changes and enhancements in Apache 2 is beyond the scope of this document, we highlight a few key points below:

- Ability to deploy the Web server in a multithreaded mode
- Better support for Windows
- IPv6 support
- Filtering

A complete list of new features are available at:

http://httpd.apache.org/docs-2.0/new_features_2_0.html

Deciding to Upgrade

While it is sometimes tempting to migrate towards the “latest and greatest” of any product, we advise caution when considering upgrading to Apache 2. The expression *“if it isn't broke, don't fix it”* holds true. The proven track record and rock solid PHP support still lean in favor of Apache 1 at this point. Also, the risks and costs involved in upgrading such a fundamental building block on the server can be significant, especially if you have a non-trivial setup. Furthermore, the status of Apache2 support in PHP (as of 4.3.x) is still tagged as experimental. (View the file `php-4.3.0/sapi/apache2filter/EXPERIMENTAL` in the distribution).

Therefore, our recommendation is not to upgrade, unless you have specific needs that are only addressed by Apache 2.

Note that Apache 1 is still supported by the Apache Software Foundation, and the deployments of Apache 1 far outnumber those of Apache 2.

If you do choose to upgrade to Apache 2, the next important choice you have to make is whether or not to deploy it in a multithreaded architecture.

Multithreading vs. Multi-Processing

Apache 2 and Multithreading

Web servers are designed to handle multiple concurrent users. Different Web servers approach the task of handling concurrent users in different ways. The most common ways are via multiple processes (multi-processing) or via multiple threads (multi-threading). Under UNIX, Apache 1 is exclusively a Multi-Process web server. Apache 2 is generally perceived by the public to be a multithreaded server. However, the truth is that Apache 2 can be configured to run in either of the two manners.

As a matter of fact, most operating system distributions that ship with Apache 2, deploy it in a single-threaded, multi-process model. In other words, unless you have built your own copy of Apache 2 and configured it to be multithreaded, chances are that it is *NOT* running in multithreaded mode.

How to Determine Whether a Server is Running as Multithreaded

To determine whether you are running Apache in multithreaded mode or not, you can run 'httpd -l'. This will display the list of modules that are built into your Apache binary. If `prefork.c` is included in the output, then you are running in multi-process mode, and *NOT* running the multithreaded version. If, however, you find `worker.c` – then you appear to be running the multithreaded version.

Decision factors

The fact that most operating system vendors chose not to deploy Apache 2 in its multithreaded form is not incidental. There are several good reasons not to use the multithreaded mode of Apache 2 at this time.

Apache Track Record

Apache 1.x has been on the market for many years, and has a proven stability track record. Apache 2.0 comes in several flavors – including a multithreaded flavor, a single threaded flavor, and a hybrid flavor. Out of those – the single-threaded flavor is the only one that builds upon the proven track record of Apache 1.x. The multithreaded and hybrid models are new to Apache 2, and have not yet gone through the same amount of stress testing.

PHP Stability

If Apache is deployed in multithreaded mode, PHP has to be built in a thread-safe mode. While PHP does in fact have a thread-safe mode, it is not yet considered entirely stable. The most problematic issue regarding stability in the context of a multithreaded server is that it only takes one single bug or crash to bring the entire server down. In other words – if you run Apache 2 in multithreaded mode, and you bump into a single thread-safety issue in PHP, or any other type of bug that results in a crash – your entire server may crash, and may stop responding to requests.

In contrast – if you run Apache 2 in single-threaded, multi-process mode – thread-safety issues will have no effect at all, and even crash bugs will rarely bring the entire server down – they will only affect the particular request that caused the crash.

It is important to note that thread-safety bugs tend to surface only under high loads. So even a typical QA process of a multithreaded PHP site is not a strong indicator for stability.

Speed Comparison

One reason that is often stated for adopting multithreaded Apache is to increase speed. This is in fact a misconception for PHP.

If you plan on using your server primarily for serving PHP content, then running your server in multithreaded mode may actually *slow you down* by as much as 15%! This overhead originates in the extra

work that PHP has to do in order to be thread-safe, including locking and passing around per-thread contexts to all functions.

Even if you're planning on serving mostly static files, using Apache 2's multithreaded mode is not guaranteed to improve performance. For example, if you are using Linux, the gain from using threads is typically quite negligible. The reason for that is that threads under Linux are implemented as processes with shared resources, so unlike other operating systems – threads are not more lightweight than processes. In other operating systems such as Solaris, the performance gain from using threads may be more visible.

Zend's Recommendation

It is Zend's recommendation to stick with Apache 1.x. This recommendation is based on:

- The risk in deploying PHP with Apache2 multithreaded, due to its experimental status.
- The lack of significant benefit in using Apache2 multi-process over Apache 1.x.
- The stability and reliability of Apache 1.x, based on its' track record.