



A Comparison of: PHP and Perl and SQL and Flat File

Editors:

Corey Chapman, YaBB Project Leader

Contributors:

Corey Chapman, YaBB Project Leader
Torsten Mrotz, YaBB 3 Lead Developer

Abstract:

This white paper discusses the advantages and disadvantages of PHP, Perl, mod_perl, flat file databases, and SQL databases.

Last Updated:

August 31, 2005

Executive Summary

We often receive questions as to if and when we are going to rewrite YaBB in PHP or add SQL support. When we ask “Why?” the answer is usually “I heard PHP is much better, faster, more stable, and more secure than Perl. And if you use an SQL database like MySQL, it’s that much better.”

Here I’ll try to explain why these rumors are truly rumors.... Keep in mind, we too embrace new technology. You can see that we do in fact use PHP and MySQL for YaBB’s websites (<http://www.yabbforum.com>, <http://www.boardmod.org>, and <http://www.yabbisback.com>). It is a great language for quick scripts, building dynamic websites, and integrating things together. We have no doubt that it has its place in the future of the Internet for a long time to come. But we must debunk the myths of PHP versus Perl and SQL versus flat file.

This white paper is based on my personal experiences, Internet research, and information given to me by forum users. Please do not quote it as pure fact, but use it to spawn discussion and help you think outside the box. Of course, I’m using this literature to help persuade you that YaBB is not all that bad....

Do not confuse the arguments in here as an attempt to slander the PHP language!

This paper will be updated throughout the future to include more information and a more factual argument.

Table of Contents

1. History of the Languages	3
2. Language Stability	3
3. PHP and Perl: What is Faster?	3
4. Ease-of-use	4
5. Availability, Portability, and Configuration	5
6. Power of the Language	5
7. Summary of PHP vs. Perl	6
8. YaBB’s Perl Bias	6
9. SQL versus Flat File	6
10. Wrap it Up!	7

1. History of the Languages

PHP (PHP: Hypertext Preprocessor) is a free server-side scripting language that embeds into HTML. This language was started by Rasmus Lerdorf in 1995 as a set of Perl scripts known as PHP/FI (Personal Home Page / Forms Interpreter). It eventually evolved into a new and popular language by mid 1998. PHP can be compiled into web servers such as Apache and was designed from the ground up as a language for the Internet. It is very easy to use when trying to provide dynamic web content because of its dedication to HTTP. The language provides a lot of built-in functions which support most things a developer may need. Some call it bloated, but others think the integration means the functions are more harmonized and faster. Today's PHP is very powerful and supports Object-Oriented Programming (OOP). More information can be found at <http://www.php.net> and http://en.wikipedia.org/wiki/Mod_php.

Perl (Practical Extraction and Report Language) is a similar free language, which was started in 1987 by Larry Wall. It was developed long before PHP as a general purpose scripting language to fill the void between a conglomeration of UNIX tools such as AWK, 'sed', shell programming languages, and C programs. It is a text-manipulation language that allows powerful operations to be performed that would otherwise be complex in C, such as regular expressions. Whereas PHP has its built-in functions, Perl makes more use of modules (freely available at <http://www.cpan.org>). It too can be embedded in HTML with the use of SSI (Server-Side Include), but it is typically used in standalone applications. Perl uses the Common Gateway Interface (CGI) to communicate with the web server. It uses a standalone processor that runs at script execution time. You can learn more about Perl from the O'Reilly Network at <http://www.perl.com> or at <http://www.perl.org>.

2. Language Stability

As stated above, Perl was started in 1987. PHP only first appeared in 1995.

Perl is more stable than PHP hands down. Perl 4.000 was released in 1991. The development team audited itself to determine why other languages were still being chosen over Perl. This resulted in the revolutionary Perl 5.000 in 1994. The most current versions have been in development since March 2000.

PHP 3.0 appeared in June 1998 and was the first version of PHP similar to today's PHP. The most popular strain, version 4.0, appeared in May 2000. Since spring of 2000, there were 24 more stable releases of PHP 4 and 4 releases of PHP 5. Less than 10 stable releases of Perl have come out during that time. If you view change logs, you'll see that PHP changes are mostly bug fixes, while Perl changes are more often feature enhancements.

I'm not going to copout with a stability discussion though. Read on....

3. PHP and Perl: What is Faster?

All too often, people confuse PHP and the Apache mod_php module or Perl and the Apache mod_perl (<http://modperlbook.org>) module. In this language combat one must compare the ordinary PHP preprocessor to the ordinary Perl interpreter (not mod_php vs. Perl or PHP vs. mod_perl). Keep in mind that BOTH languages are interpreted, which means they are not pre-compiled and executed as a binary. They are parsed by an interpreter on-the-fly. For a

comparison of compiled and interpreted programs, please see <http://kb.iu.edu/data/agsz.html> and http://perl.apache.org/docs/general/perl_myth/perl_myth.pdf. Truthfully, PHP (not mod_php) is slower, but hear me out before you moan at my ignorance.

First of all, when you run mod_php or mod_perl, no PHP or Perl processor/interpreter is loaded in memory at script execution time because it is already loaded in memory when the server starts. This avoids the startup time of the external interpreter and makes it possible to create smaller children (threads) of the interpreter to handle one request. Additionally, threads can handle several requests in parallel by multithreading. This “mod” version would give you a benefit of about 0.01 – 0.02 seconds per process in Perl and 0.7 sec using PHP.

The second benefit is permanent database connections. Normally your mod_php or mod_perl script won't connect to a database, this saves you 0.13 sec using Perl with the DBI module and 0.2 sec with PHP. As you can see from this, Perl is innately quite efficient as the module version is not able to save as much time as the PHP module version does.

Average memory usage for PHP scripts is *3.5 times higher than Perl* (0.7% vs. 0.2%), while average CPU usage for PHP scripts (1.3%) is more than *13 times higher than Perl* (less than 0.1%). It's well-known that PHP needs more resources than Perl. Why does it require mod_php to achieve good speeds? Perl can fairly easily match these speeds, and it can even surpass them when coupled with mod_perl.

4. Ease-of-use

Did you ever notice that PHP-developers are the first who say “PHP rules! It's easy to do everything!”? Sure, PHP is a good programmers' choice for websites, no doubt. That's what it was developed for, and that's why it gets so much Internet hype. But for a programmer who wants to deliver the working product ASAP. Yes, PHP is easy, but is it easy to customize?

What is easier for you? To make a customization in a template like:

```
Author Name: [[USER_NAME]]
```

or include PHP-embedding like:

```
Author Name: <?php echo $user_variables['user_name']; ?>
```

Do you want your own programmer who will be making your desired customizations to find 2 strings of code between kilobytes of HTML, or do you want to call for programmer or product support every time when you want to change your site layout only because you think that your designer can corrupt the script code? I think not.

Of course, I do have to mention that although PHP's method looks uglier, it doesn't require any modules or coding on your part. Perl would require a templating module or your own code to parse the strings. YaBB uses keywords like <yabb something> in its template files, and a few simple lines of Perl code parses it to display \$yabbsomething variables. The benefit is that you are removing scripting code from the template for much cleaner HTML (or XHTML). If you do this with PHP by creating a templating system, you'll achieve the same results but lose a lot of PHP's advantages of HTML integration.

5. Availability, Portability, and Configuration

Sure PHP is as wide-spread as Perl nowadays, but let's be realistic: different web hosts have different PHP versions with their own settings. Some servers use PHP in "safe mode," which many applications will not run under. Providing a PHP version of YaBB would require modifications for about 10 different versions of PHP. Unfortunately, different PHP versions are not 100% compatible, so neither you, nor other developers can be absolutely sure it would work after moving to another web host. This is one downside to PHP having most functions built-in: backwards compatibility.

The next problem can be PHP configuration. Since PHP is not as universal as Perl, most features can be switched on or off *not* in your script, but during PHP (or mod_php) compile time, in the php.ini file, or in the Apache configuration files. Usually you would be unable to do this on virtual servers, as most web hosts are today. For some reasons you may not want to perform these operations on dedicated servers either. PHP does allow you to make "on the fly" changes to php.ini in your script, but not every setting can be changed even if your web host allows you to use this feature. What you may theoretically lose: file uploads, error tracking, include directories support, and many others.

One may ask about Perl modules that are needed by the scripts, but may be unavailable on the web server. Well, I am aware that not all Perl configurations are the same either. If you have a newer version of Perl, you may have newer included modules, or you may have an older version with missing modules. The answer is easy: frequently-used Perl modules (like CGI, LWP, etc.) often come with Perl distributions, so you may find them on near 100% of servers running Perl. Web hosting companies running MySQL on their servers usually have appropriate DBI modules loaded too. You can also do one of two things: include working versions of the needed modules with your script in case the server doesn't have them OR don't rely on "standard" modules. YaBB 1 doesn't require any modules. YaBB 2 requires a few for some advanced features, but working copies of the Perl modules are included in the package in case your server doesn't have them.

Moreover, web hosting companies typically prefer installing additional Perl modules rather than recompiling the server software or changing configuration files, such as is required by PHP changes or additions.

6. Power of the Language

I cannot deny that PHP is VERY powerful. After all, much of it was modeled after Perl. It is very easy to see that it is not much more than a more structured C-based language written to do much of what Perl does on the Internet. However, Perl works much closer to the operating system than PHP. And its regular expression support is no doubt the best in the world. That's why languages like PHP are using the PCRE (Perl Compatible Regular Expressions) library – they want to also benefit from these powerful tools.

PHP is not for general purpose use. It was made for the Internet. Perl was modified from a system "shell" language to allow it to be used over the Internet. Perhaps this is where PHP has its advantages. But Perl is designed with crunching text in mind and has facilities for handling strings and the like that put most other languages to shame – including PHP. It is very efficient at handling flat files such as those that YaBB currently uses. If we were to write YaBB in PHP but still use flat files, we probably would struggle to achieve the same performance results we have today.

7. Summary of PHP vs. Perl

The purpose of this argument was not to counter PHP-lovers with “Perl rules!” It was to sway your harsh opinion of Perl and to help you realize that Perl is just as powerful and fast as PHP. It truly depends on what application you are writing as to which is best. A program is only as good as the programmer and the coffee he or she drinks.

In the end, it's also more of a user preference. Some people prefer to program in PHP and others more enjoy Perl – some people like both. With Perl, you often have several ways to do the same thing, and developers have a lot of freedom. It may be true that it is much easier for a Perl programmer to be lazy and create a poorly-coded application that fails miserably in benchmarks – but Perl also has options called Strict and Warnings that, if used, can reduce this factor to the level that it is no different than PHP.

If used properly, either of these languages can beat the other. Mod_perl, mod_php, object-oriented programming, database systems, and other tools can greatly improve any Internet application.

8. YaBB's Perl Bias

Certainly, YaBB has a bias over Perl. It's what all web applications were made out of 7 years ago and earlier when PHP didn't exist. PHP was a toddler when YaBB began, so it was too experimental for Zef Hemel to try to use when creating YaBB. Since YaBB started as a Perl program, we have vowed to keep it that way. There is absolutely nothing wrong with the language. Every other system out there is jumping ship to PHP. Very few systems remain in Perl. This gives us another competitive advantage: servers without PHP support and webmasters that don't like PHP. More often than not, you'll find servers WITH Perl support and without PHP support. Although, this statement probably won't hold true through the end of 2006.

Perl will be in development for a very long time to come. YaBB has always had a tradition of supporting older systems longer than most projects do. Old doesn't always mean worse, it often means mature! We held onto support for Netscape browsers and early versions of Internet Explorer for a long time, but we also supported browsers like Safari, Opera, and Firefox very early on. We kept options available for those without JavaScript support in their browser, and we had styles setup to work with browsers that didn't support CSS. But eventually we did abandon people who refused to upgrade their Internet browser or allow JavaScript. The way we see it, servers will have Perl on them for the unforeseeable future, so there is no argument that we have supported an outdated technology too long.

9. SQL versus Flat File

Relational databases have gotten all sorts of hype since the inception of MySQL (<http://www.mysql.com/>). It has been popularized by PHP because most dynamic websites that use PHP also use MySQL for data storage. That is certainly true of forum systems (i.e. IPB, SMF, vB, phpBB, etc.).

With a relational database like MySQL, the database has to be connected to, and then a table must be found. Finally you must search records for the data you need. There is a lot of overhead involved in those transactions. However, one of its main advantages is that the engine to search for data is already running in the background, taxing your script less.

If a flat file system is written efficiently, data can be loaded simply by opening the file and reading in its contents. It doesn't have to go through a special engine, and you can write your own "database" code to optimize how information is read and written. Of course, it can be written poorly causing performance degradation as can be seen with YaBB 1's file usage. The new version of YaBB, version 2, corrects this issue by using summary files and hashes and accessing files less often.

For heavy actions such as a search, MySQL and other database systems have very heavy overhead. A lot of resources are required to locate the database, the table in the database, and the line(s) in the table. Then the database system must read through multiple lines of data.

Another problem is indexing. Sad, but true – some applications use MySQL or other database systems, but they do not use the built-in indexing abilities. Sure, you'll get other benefits like data integrity or caching, but as said – indexing is the main factor influencing productivity in relational databases. Without it, the advantages over flat file systems diminish greatly.

Unmotivated usage: That's an opposite for using flat file – some developers prefer to store everything, like templates, large files and images, and script parameters in the database. Unfortunately, these steps only decrease performance of your system. Code authors only do this to provide cleaner code and reduce the amount of files that need to be uploaded for the application. Call it laziness! Flat file is certainly much faster when only a small amount of data is being stored. There is no need to open a database, then a table, then a record. The file with the data you need can be slurped right in instead!

Again, as with the *PHP versus Perl* argument, it depends on the application one is writing. It also depends on how the application is written as to which database, SQL or flat file, is best.

10. Wrap it Up!

I hope this paper has turned your gears a bit and put you in a position to push the box rather than sit in it. Perl and PHP are BOTH great languages. They each have advantages and disadvantages. Both can exist in harmony, just like YaBB and PHP forums like Simple Machines do. Everyone needs healthy competition, and there is no "one way" to do anything. The same goes for MySQL. It is a great database system. I use it often. However, there is a time and place for flat file databases too. Sure, YaBB could probably benefit from abandoning flat files, but it can also be written so efficiently that you don't notice the difference between it and PHP/MySQL forums.

If you think large sites only use PHP nowadays, think again. E-Toys, CitySearch, Internet Movie Database, Value Click, Paramount Digital Entertainment, CMP, HotBot, and DejaNews rely on Perl for their websites....