# 1   Popular Perl Complaints and Myths

# 1.1 Description

This document tries to explain the myths about Perl and overturn the FUD certain bodies try to spread.

# 1.2 Abbreviations

- **M** = Misconception or Myth

- **R** = Response

## 1.2.1 Interpreted vs. Compiled

- **M:**

  Each dynamic perl page hit needs to load the Perl interpreter and compile the script, then run it each time a dynamic web page is hit. This dramatically decreases performance as well as makes Perl an unscalable model since so much overhead is required to search each page.

- **R:**

  This myth was true years ago before the advent of mod_perl. mod_perl loads the interpreter once into memory and never needs to load it again. Each perl program is only compiled once. The compiled version is then kept into memory and used each time the program is run. In this way there is no extra overhead when hitting a mod_perl page.

### 1.2.1.1 Interpreted vs. Compiled (More Gory Details)

- **R:**

  Compiled code always has the potential to be faster than interpreted code. Ultimately, all interpreted code needs to eventually be converted to native instructions at some point, and this is invariably has to be done by a compiled application.

  That said, an interpreted language CAN be faster than a comprable native application in certain situations, given certain, common programming practices. For example, the allocation and de-allocation of memory can be a relatively expensive process in a tightly scoped compiled language, wheras interpreted languages typically use garbage collectors which don't need to do expensive deallocation in a tight loop, instead waiting until additional memory is absolutely necessary, or for a less computationally intensive period. Of course, using a garbage collector in C would eliminate this edge in this situation, but where using garbage collectors in C is uncommon, Perl and most other interpreted languages have built-in garbage collectors.

  It is also important to point out that few people use the full potential of their modern CPU with a single application. Modern CPUs are not only more than fast enough to run interpreted code, many processors include instruction sets designed to increase the performance of interpreted code.

## 1.2.2  *Perl is overly memory intensive making it unscalable*

- **M:**

  Each child process needs the Perl interpreter and all code in memory. Even with mod_perl httpd processes tend to be overly large, slowing performance, and requiring much more hardware.

- **R:**

  In mod_perl the interpreter is loaded into the parent process and shared between the children. Also, when scripts are loaded into the parent and the parent forks a child httpd process, that child shares those scripts with the parent. So while the child may take 6MB of memory, 5MB of that might be shared meaning it only really uses 1MB per child. Even 5 MB of memory per child is not uncommon for most web applications on other languages.

  Also, most modern operating systems support the concept of shared libraries. Perl can be compiled as a shared library, enabling the bulk of the perl interpreter to be shared between processes. Some executable formats on some platforms (I believe ELF is one such format) are able to share entire executable TEXT segments between unrelated processes.

### 1.2.2.1  More Tuning Advice:

- Stas Bekman's Performance Guide

## 1.2.3  *Not enough support, or tools to develop with Perl. (Myth)*

- **R:**

  Of all web applications and languages, Perl arguable has the most support and tools. **CPAN** is a central repository of Perl modules which are freely downloadable and usually well supported. There are literally thousands of modules which make building web apps in Perl much easier. There are also countless mailing lists of extremely responsive Perl experts who usually respond to questions within an hour. There are also a number of Perl development environments to make building Perl Web applications easier. Just to name a few, there is `Apache::ASP`, `Mason`, `embPerl`, `ePerl`, etc...

## 1.2.4  *If Perl scales so well, how come no large sites use it? (myth)*

- **R:**

  Actually, many large sites DO use Perl for the bulk of their web applications. Here are some, just as an example: **e-Toys**, **CitySearch**, **Internet Movie Database**( http://imdb.com ), **Value Click** ( http://valueclick.com ), **Paramount Digital Entertainment**, **CMP** ( http://cmpnet.com ), **HotBot Mail/HotBot Homepages**, and **DejaNews** to name a few. Even **Microsoft** has taken interest in Perl via http://www.activestate.com/.

## 1.2.5  Perl even with mod_perl, is always slower then C.

- **R:**

The Perl engine is written in C. There is no point arguing that Perl is faster than C because anything written in Perl could obviously be re-written in C. The same holds true for arguing that C is faster than assembly.

There are two issues to consider here. First of all, many times a web application written in Perl **CAN be faster** than C thanks to the low level optimizations in the Perl compiler. In other words, its easier to write poorly written C then well written Perl. Secondly its important to weigh all factors when choosing a language to build a web application in. Time to market is often one of the highest priorities in creating a web application. Development in Perl can often be twice as fast as in C. This is mostly due to the differences in the language themselves as well as the wealth of free examples and modules which speed development significantly. Perl's speedy development time can be a huge competitive advantage.

## 1.2.6  Java does away with the need for Perl.

- **M:**

Perl had its place in the past, but now there's Java and Java will kill Perl.

- **R:**

Java and Perl are actually more complimentary languages then competitive. Its widely accepted that server side Java solutions such as `JServ`, `JSP` and `JRUN`, are far slower then mod_perl solutions (see next myth). Even so, Java is often used as the front end for server side Perl applications. Unlike Perl, with Java you can create advanced client side applications. Combined with the strength of server side Perl these client side Java applications can be made very powerful.

## 1.2.7  Perl can't create advanced client side applications

- **R:**

True. There are some client side Perl solutions like PerlScript in MSIE 5.0, but all client side Perl requires the user to have the Perl interpreter on their local machine. Most users do not have a Perl interpreter on their local machine. Most Perl programmers who need to create an advanced client side application use Java as their client side programming language and Perl as the server side solution.

## 1.2.8  ASP makes Perl obsolete as a web programming language.

- **M:**

With Perl you have to write individual programs for each set of pages. With ASP you can write simple code directly within HTML pages. ASP is the Perl killer.

- **R:**

  There are many solutions which allow you to embed Perl in web pages just like ASP. In fact, you can actually use Perl IN ASP pages with PerlScript. Other solutions include: `Mason`, `Apache::ASP`, `ePerl`, `embPerl` and `XPP`. Also, Microsoft and ActiveState have worked very hard to make Perl run equally well on NT as Unix. You can even create COM modules in Perl that can be used from within ASP pages. Some other advantages Perl has over ASP: mod_perl is usually much faster then ASP, Perl has much more example code and full programs which are freely downloadable, and Perl is cross platform, able to run on Solaris, Linux, SCO, Digital Unix, Unix V, AIX, OS2, VMS MacOS, Win95-98 and NT to name a few.

  Also, Benchmarks show that embedded Perl solutions outperform ASP/VB on IIS by several orders of magnitude. Perl is a much easier language for some to learn, especially those with a background in C or C++.

## 1.3 Credits

Thanks to the mod_perl list for all of the good information and criticism. I'd especially like to thank,

- Stas Bekman <stas@stason.org>

- Thornton Prime <thornton@cnation.com>

- Chip Turner <chip@ns.zfx.com>

- Clinton <clint@drtech.co.uk>

- Joshua Chamas <joshua@chamas.com>

- John Edstrom <edstrom@Poopsie.hmsc.orst.edu>

- Rasmus Lerdorf <rasmus@lerdorf.on.ca>

- Nedim Cholich <nedim@comstar.net>

- Mike Perry < http://www.icorp.net/icorp/feedback.htm >

- Finally, I'd like to thank Robert Santos <robert@cnation.com>, CyberNation's lead Business Development guy for inspiring this document.

## 1.4 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- Contact the mod_perl docs list

# 1.5  Authors

- Adam Pisoni <adam@cnation.com>

Only the major authors are listed above. For contributors see the Changes file.

# Table of Contents: