

## 'Premature Optimization' - Keeping PHP and MySQL Query Benchmarking In Prospective

*Jacob Sanford - President, Elemental Codeworks Inc.*

### Introduction

Donald Knuth, the father of the analysis of algorithms once wrote "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil". Although often quoted, the message should be considered by project developers during the programming cycle. As someone who spends a large amount of time programming, I often begin obsessing about the performance and tuning aspect of a project before it is warranted by the development cycle. This often becomes a problematic sidebar and can derail the forward motion of the project instantly.

Is this a strange thing to happen to someone who is constantly preaching the use of cost/gain analysis to his clients? Absolutely – yet I feel that in this ‘information age’ it is easy to get bogged down in a flood of raw data and opinions if there is not a method present to effectively filter it.

### A Great Information Filter – Realistic Analysis

The PHP benchmark data given in Fig. 1 appears to indicate a stark contrast of string output function speed – and thus a significant difference in overall performance. One should, however, first consider the overall effect on a typical project. Based on the data given in Fig.1, for a website that receives 100,000 pageviews per day the total CPU time gained by using the absolute least efficient function compared to the one with the highest efficiency is:

$$\begin{aligned} Dif_{100kHits} &= 100(\text{worst} - \text{best}) \\ Dif_{100kHits} &= 100 \times (551 \times 10^{-6} s - 149 \times 10^{-6} s) \\ Dif_{100kHits} &= 100 \times (402 \times 10^{-6} s) \\ Dif_{100kHits} &= 402 \times 10^{-4} s \\ Dif_{100kHits} &= 0.0402s, \end{aligned}$$

which corresponds to less than 1/20th of a second per day. Hypothetically, if a developer were to spend approximately 4 hours to examine a project’s core engine to find 40 uses of the least efficient method, you would only save ~2 seconds/day of CPU time (Over the 100k pageviews).

This would correspond to a turnover of time investment in approximately 7200 days (or ~20 years) – well beyond the average lifespan of any web presence. Aside from this, it is unlikely that each visitor would be aware of the 40 nanosecond difference you saved them per pageview anyhow.

## Code Structure – A Priority over Function Optimization

As an alternative to function choice optimization, I propose that devoting time to ensuring an adherence to a standard set of PHP/MySQL ‘best practices’ provides a much greater long-term gain. Some of these best practices to keep in mind during development are:

- Brainstorming application features and scope before the start of production, followed up with a code blueprint.
- Sticking to a standardized style of coding layout and structure to ensure ease of review or update at a later date.
- Ensuring portability between browser and operating systems.
- Writing module-based code that allows easy expansion and portability.

## Page Load Times – The Elephant in Your Server Room?

Additionally, large benefits can be delivered through optimization of webpage load time through the size of media on the page. Whereas tuning of SQL and PHP function performance offers a performance boost in the order of microseconds per execution, analysis and optimization of on-page graphics and media (as well as redundant HTML code) provides the reduction of time-to-screen for surfers on the scale of seconds (and higher). By:

- Analyzing and reducing the size of images
- Enabling mod\_expires
- Reducing the whitespace in your HTML

you can successfully ‘performance tune’ with a positive and useful overall effect.

*Jacob Sanford is the founder and president of Elemental Codeworks Inc. (<http://www.elementalcodeworks.com>), a website development and promotion firm. Elemental Codeworks partners with new and established websites to develop and increase the popularity of their web presence through effective link building and buzz marketing. He can be reached via E-Mail at [jake@elementalcodeworks.com](mailto:jake@elementalcodeworks.com).*

String Output  
echo vs. print

Is there a difference between what option you use to output your content?. Called within Output Buffering 1'000x

+ 100 %	echo "	Total time: 96 µs	<a href="#">view code</a>
+ 104 %	print "	Total time: 100 µs	<a href="#">view code</a>
+ 155 %	echo 'aaaaaaaaaaaaaaaaaaaaaaaaaaaa'	Total time: 149 µs	<a href="#">view code</a>
+ 132 %	print 'aaaaaaaaaaaaaaaaaaaaaaaaaaaa'	Total time: 127 µs	<a href="#">view code</a>
+ 137 %	echo 'aaaaaaa','aaaaaaa','aaaaaaa','aaaaaaa'	Total time: 132 µs	<a href="#">view code</a>
+ 316 %	echo 'aaaaaaa','aaaaaaa','aaaaaaa','aaaaaaa'	Total time: 304 µs	<a href="#">view code</a>
+ 144 %	print 'aaaaaaa','aaaaaaa','aaaaaaa','aaaaaaa'	Total time: 138 µs	<a href="#">view code</a>
+ 531 %	<code>\$a = 'aaaaaaa'; echo 'aaaaaaa'.\$a.'aaaaaaa'.\$a</code>	Total time: 510 µs	<a href="#">view code</a>
+ 433 %	<code>\$a = 'aaaaaaa'; echo 'aaaaaaa'.\$a.'aaaaaaa'.\$a</code>	Total time: 416 µs	<a href="#">view code</a>
+ 551 %	<code>\$a = 'aaaaaaa'; print 'aaaaaaa'.\$a.'aaaaaaa'.\$a</code>	Total time: 529 µs	<a href="#">view code</a>
+ 533 %	<code>\$a = 'aaaaaaa'; echo \$a.\$a.\$a.\$a</code>	Total time: 512 µs	<a href="#">view code</a>
+ 426 %	<code>\$a = 'aaaaaaa'; echo \$a.\$a.\$a</code>	Total time: 409 µs	<a href="#">view code</a>
+ 573 %	<code>\$a = 'aaaaaaa'; print \$a.\$a.\$a</code>	Total time: 551 µs	<a href="#">view code</a>

Conclusion:

In reality the echo and print functions serve the exact purpose and therefore in the backend the exact same code applies. The one small thing to notice is that when using a comma to separate items whilst using the echo function, items run slightly faster.

**Fig. 1 – A series of benchmarks that compare string output function execution speed (source: [www.phpbench.com](http://www.phpbench.com)).**