

Thursday, February 4, 2010

### HipHop PHP - Nifty Trick?

In a response to a question from ReadWriteWeb, among other things, I wrote:

My main worry here is that people think this is some kind of magic bullet that will solve their site performance problems. Generating C++ code from PHP code is a nifty trick and people seem to have gotten quite excited about it. I'd love to see those same people get excited about basic profiling and identifying the most costly areas of an application. Speeding up one of the faster parts of your system isn't going to give you anywhere near as much of a benefit as speeding up, or eliminating, one of the slower parts of your overall system.

The "nifty trick" part of that seems to have become the story, and them injecting a "just" in front of it makes it sound more derogatory. Anyone who knows me knows that I am a big fan of nifty tricks that solve the problem. When I first heard about the Facebook effort I was assuming they were writing a JIT based on LLVM V8 or something along those lines. Writing a good JIT is hard. Doing static code analysis and generating compilable C++ from it is indeed a nifty trick. It's not "just" a nifty trick, it is a cool trick that takes advantage of a number of characteristics of PHP. The main one being that you can't overload PHP functions. `strlen()` is always `strlen`, for example. In Python, this would be harder because you can overload everything.

I also noted that most sites on the Web have a lot of lower hanging fruit that would provide a much bigger performance improvement, if fixed, than doubling the speed of the PHP execution phase. The ReadWriteWeb site, for example, needs 160 separate HTTP requests and 41 distinct DNS lookups to load the front page. And once you get beyond the frontend inefficiencies you usually find Database issues, inefficient system call issues and general architecture problems that again aren't solved by speeding up PHP execution.

If you have done your homework and find that your web servers are cpu-bound, you are already using an opcode cache like APC and your Callgrind callgraph shows you that the PHP executor is a significant bottleneck, then HipHop PHP is definitely something you should be looking at.

Posted by Rasmus at 10:50

Eye opener for the fanboys :)

Anonymous on Feb 4 2010, 11:32

"The main one being that you can't overload PHP functions. `strlen()` is always `strlen`, for example"

Do you think it will be difficult to make HipHop PHP5.3 ready? Are namespaces etc. problematic?

Anonymous on Feb 4 2010, 11:55

We were pretty careful with the new features in 5.3 and kept the amount of runtime magic to an absolute minimum which should mean there are no significant hurdles for HipHop to support 5.3.

Anonymous on Feb 4 2010, 12:04

Very well said. HipHop indeed is well thought & done, and it is no doubt that it serves facebook's purpose.

Anonymous on Feb 4 2010, 12:27

## Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

I totally agree on your point that HipHop shouldn't be the first solution in place when it comes to optimize web applications with a lack of performance.

Anonymous on Feb 4 2010, 13:25

Most of my questions surround the viability of code conversion. It's long been history that this kind of technology is bug-prone at best.

It just sounds like it's going to generate lots of problems that exist only in the conversion later, and that will never be found when the developer has to go back and debug in pure PHP.

Nevermind that Hip Hop only supports "several popular extensions"... which leaves anyone who uses one of the "unpopular" extensions out in the cold.

--

It sounds like a magic bullet to people, but it's not. You've hit it pretty well on the head in that there are generally much bigger problems than the actual code execution taking place.

Anonymous on Feb 4 2010, 17:10

Correct me if I'm wrong, but as long as PHP 6 is not released we have to rely on mbstring in order to have an UTF-8 site. I guess HipHop won't support mbstring overloading of functions (strlen with mb\_strlen, for example) for the same reasons you're exposing in your post. Am I right?

Anonymous on Feb 9 2010, 07:05

Mauro, since the mbstring overloading is done internally in PHP when PHP is built, it would work ok assuming the mbstring extension has been converted to HipHop. The overloading I mentioned was userspace function overloading.

Anonymous on Feb 9 2010, 08:01

I think that HipHop is pretty cool, but not being much of a C/C++ programmer, I was hoping for a code transformer that output Zend API C code, that could then be compiled in the normal phpize way. Something along the lines of:

```
function x ($a, $b)
{
    return $a * $b;
}
```

might output an extension skeleton that contained a function like

```
PHP_FUNCTION(x)
{
    /*I have no idea on how to implement this in the Zend API, but you get the idea*/
    long *a, b;
    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "ll", &a, &b) == FAILURE) {
        return;
    }
    return_value = a * b;
}
```

or something to that extent.

I think it would be useful, even if it had to require type hinting and return type hinting. But even that could be overcome with the zval struct couldn't it? Have you heard of anyone trying to do this type of code transformation? I feel like it would make these really complex frameworks so much simpler to implement IMHO.

(Couldn't figure out how to prevent \*\* from being parsed as bold, sorry)

Anonymous on Feb 11 2010, 19:21

We already have ext/skel in the source tree and the more powerful CodeGen\_Pecl. See [http://pear.php.net/package/CodeGen\\_PECL](http://pear.php.net/package/CodeGen_PECL) which lets you define much of an extension in an xml file. Just about all parts of an extension can be described, and while it doesn't mean you don't have to write some C or C++ code to do what your extension is supposed to do, it does free you from needing to know the details of the PHP APIs. Documentation is here: [http://php-baustelle.de/CodeGen\\_PECL/manual.html](http://php-baustelle.de/CodeGen_PECL/manual.html)

Anonymous on Feb 11 2010, 20:49

Yes, I have seen CodeGen\_PECL, I guess I was just asking about something that didn't involve writing C, but could still be compiled.

Anonymous on Feb 12 2010, 05:39

I think it would be useful, even if it had to require type hinting and return type hinting. But even that could be overcome with the zval struct couldn't it?

Anonymous on Mar 10 2010, 12:27

I agree with Rasmus - it is a nifty trick, but I have burned by nifty tricks before.

For instance, setting long Expires headers (at the advice of ySlow, your old employer) was not worth it when I've done it. Nobody I worked with understood why files weren't refreshing without changing the URL. Since then I have seen many people not enthralled

## Blog Export: Rasmus' Toys Page, <http://toys.lerdorf.com/>

with the same procedure.

This said, I haven't yet been in a situation where CPU optimization was necessary (yet). If the situation ever arose, though, I would like first consider APC or another opcode caching system before considering the (relatively drastic) compiling to C++ (only difference is JIT -> binary, though this can be a big sometimes).

My worry with HipHop is that small shops or developers will pick it up because they think it will solve any scalability problems. But what it may do in reality is just be more confusing, complex to release / maintain (someone might even have to know C++, =O), and be more prone to error. This would be especially angering if their weakest link is more likely to be an unshielded RDBMS or the algorithms themselves.

"Premature optimization is the root of all evil" - Knuth  
Anonymous on May 24 2010, 09:53