

Forum:[Hack Forums](#)

Thread: [How to increase performance of PHP!](#)

VladHQ - December 17, 2013, 11:25 am

How to increase performance of PHP! Introduction

PHP is a language that compiles the scripts each time a user visits your website. Sometimes, it is a waste to recompile the same exact script. In this tutorial, I will introduce you to caching and other big performance tips.

Caching ([Click to View](#))

What is Caching?

Code Caching - store away in hiding or for future use. Google just defined it for us. Caching is good for any application, especially blogs.

Example

Say we have a script called websites.php which consisted of:

PHP Code

```
<?php
echo 'Google:<br/>';
echo file_get_contents('http://www.google.com/');
echo '<br/>';
```

```
echo 'Yahoo:<br/>';
echo file_get_contents('http://www.yahoo.com/');
echo '<br/>';
```

```
echo 'Facebook:<br/>';
echo file_get_contents('http://www.facebook.com/');
echo '<br/>';
```

?>It will take about 2-5 seconds depending on the location of the server to make requests to the three websites.

What if the user refreshes the page? Your server would make another and another and another request. That could be just a waste of bandwidth!

But what if we just made a request once and just put it in a file? And upon request we just read from a file? Would that save up resources? YES!

How Do I Cache?

It depends on what you are making. If you are making a search engine, [like me](#), you shouldn't cache because you can just cache your MySQL queries with [SQL_CACHE](#).

For the example's sake, I will use the website example.

PHP Code

```
<?php
// http://www.theukwebdesigncompany.com/articles/php-caching.php
```

```
$cachefile = './cache/' . $_SERVER['REMOTE_ADDR'] . '.html';
$minutes = '10';
$cachetime = $minutes * 60;
```

```

if(file_exists($cachefile) && (time() - $cachetime < filemtime($cachefile))) { // Check if the file exists
    include($cachefile); // if the file is cached, include it
    die();
}
ob_start(); // Make sure to include this

/* Your script goes here */
echo 'Google:<br/>';
echo file_get_contents('http://www.google.com/');
echo '<br/>';

echo 'Yahoo:<br/>';
echo file_get_contents('http://www.yahoo.com/');
echo '<br/>';

echo 'Facebook:<br/>';
echo file_get_contents('http://www.facebook.com/');
echo '<br/>';

file_put_contents($cachefile, ob_get_contents()); // Write to cache
ob_end_flush();
?>

```

Minification ([Click to View](#))

What is Minification?

Minification is the act of making an HTML/CSS/JS file smaller without having to remove the main components. By removing unused spaces you could save 20% of the space right there!

Example

PHP Code

```

<?php
<!DOCTYPE html>
<html>
<head>
<title>My Title</title>
<style>
body {
    background-color:yellow;
}
</style>
</head>
<body>
<h1>HI!</h1>
</body>
</html>

```

becomes this:

PHP Code

```

<?php
<!DOCTYPE html><html><head><title>My Title</title><style>body{background-color:yellow;}</style>
</head><body><h1>HI!</h1></body></html>

```

How Do I Minify Files?

If you would like to do it on run-time:

- Download this package -

<https://code.google.com/p/minify/downloads/detail?name=minify-2.1.7.zip>

- Put this on top of your PHP file:

PHP Code

```
<?php
function minify($content) {
    require './minify/min/lib/Minify/HTML.php';
    require './minify/min/lib/Minify/CSS.php';
    require './minify/min/lib/JSMin.php';
    $content = Minify_HTML::minify($content, array(

        'cssMinifier' => array('Minify_CSS', 'minify'),
        'jsMinifier' => array('JSMin', 'minify')
    ));
    return $content;
}

ob_start('minify');

/* .. */
```

Compiling JS ([Click to View](#))

Compiling JavaScript

This is a small section that in which I will explain how to make your JavaScript about 80% slower. It is called the Google Closure Compiler. It is an external link that can be found below:

<http://closure-compiler.appspot.com/home>

Tips ([Click to View](#))

Basic Tips Use commas to separate variables in echo statements.

Use echo instead of print.

Avoid loops!

Use single quotes (")

Don't use require_once() or include_once()

Don't copy variables (see example)

PHP Code

```
<?php
// Bad Way:
$string = '<sTrInG>';
$string = htmlentities($string);
$string = strtolower($string);

// Good Way
$string = htmlentities(strtolower('<sTrInG>'));
?> Use PDO or mysqli_* instead of mysql_*
```

I hope that you guys found this helpful. I will be adding some other stuff a little later, and if anyone knows any tips that should be on here, just reply to the thread. Thanks!

DJSanderrr - December 17, 2013, 11:30 am

Thanks for the tutorial!

but..

(Dec 17, 2013 12:25 PM)VladHQ Wrote: [\[CITE\]](#) PHP Code

<?php

// <http://www.theukwebdesigncompany.com/articles/php-caching.php>

```
$cachefile = './cache/' . $_SERVER['REMOTE_ADDR'] . '.html';
```

```
$minutes = '10';
```

```
$cachetime = $minutes * 60;
```

```
if(file_exists($cachefile) && (time() - $cachetime < filemtime($cachefile))) { // Check if the file exists
```

```
    include($cachefile); // if the file is cached, include it
```

```
    die();
```

```
}
```

```
ob_start(); // Make sure to include this
```

```
/* Your script goes here */
```

```
echo 'Google:<br/>';
```

```
echo file_get_contents('http://www.google.com/');
```

```
echo '<br/>';
```

```
echo 'Yahoo:<br/>';
```

```
echo file_get_contents('http://www.yahoo.com/');
```

```
echo '<br/>';
```

```
echo 'Facebook:<br/>';
```

```
echo file_get_contents('http://www.facebook.com/');
```

```
echo '<br/>';
```

```
ob_start();
```

```
file_put_contents($cachefile, ob_get_contents()); // Write to cache
```

```
ob_end_flush();
```

```
?>[/spoiler]why does it do ob_start(); two times? Is it required or wans't you meant to do it?
```

-DJS

t0rch - December 17, 2013, 11:31 am

This is a great tutorial! I don't entirely agree with Minification as it makes it hard for other people to read it. But for those who'd like to it's also great!

I've made this a book mark as a reference just in case I need it ?

BinKill-Ethical ?

badbrains - December 17, 2013, 11:33 am

Caching: Use APC

"Minification": Use GZip. GZip > minification

Compiling JS: This is not actually compilation of JS it seems, rather "minification" as you so eloquently put it. GZip would cover this if you install it correctly.

VladHQ - December 17, 2013, 12:18 pm

(Dec 17, 2013 12:30 PM)DJSanderrr Wrote: [\[CITE\]](#) Thanks for the tutorial!

but..

why does it do ob_start(); two times? Is it required or wans't you meant to do it?

-DJS

I did it on accident. I just fixed it. Thank you.

(Dec 17, 2013 12:33 PM)Wolf Haley Wrote: [\[CITE\]](#) Caching: Use APC

"Minification": Use GZip. GZip > minification

Compiling JS: This is not actually compilation of JS it seems, rather "minification" as you so eloquently put it. GZip would cover this if you install it correctly.

You can't install APC on shared hosting. Most people use shared hosting right now. You could use both to save space (that's what I do).

(Dec 17, 2013 12:31 PM)BinKill-Ethical Wrote: [\[CITE\]](#) This is a great tutorial! I don't entirely agree with Minification as it makes it hard for other people to read it. But for those who'd like to it's also great!

I've made this a book mark as a reference just in case I need it ?

BinKill-Ethical ?

Thanks for the awesome feedback. If you want people to view your code, just provide a link to the actual unminified full page.

I will be adding a Cloudflare setup guide, as it does make your page load faster.

badbrains - December 17, 2013, 12:20 pm

(Dec 17, 2013 01:18 PM)VladHQ Wrote: [\[CITE\]](#) I did it on accident. I just fixed it. Thank you.

You can't install APC on shared hosting. Most people use shared hosting right now. You could use both to save space (that's what I do).

If you're doing any serious project (that requires caching and compression) on shared hosting, you have a problem. Just my two cents.

VladHQ - December 17, 2013, 12:22 pm

(Dec 17, 2013 01:20 PM)Wolf Haley Wrote: [\[CITE\]](#) If you're doing any serious project (that requires caching and compression) on shared hosting, you have a problem. Just my two cents.

Well I am creating a search engine right now. I have developed it for a little less than a year, and I find it better to use shared hosting as it is maintained, and I use my laptop to host the crawler in Python. It's not that bad.

VladHQ - December 20, 2013, 7:22 pm

I would live to hear some more feedback on this tutorial!

Mafugsa Bob - December 21, 2013, 1:15 am

(Dec 20, 2013 08:22 PM)VladHQ Wrote: [\[CITE\]](#) I would live to hear some more feedback on this tutorial!

(Dec 17, 2013 12:33 PM)Wolf Haley Wrote: [\[CITE\]](#) Caching: Use APC

"Minification": Use GZip. GZip > minification

Compiling JS: This is not actually compilation of JS it seems, rather "minification" as you so eloquently put it. GZip would cover this if you install it correctly.

^that...

You shouldn't have to cache, minify or all of that on a project you're hosting on a shared host, as

Wolf said.

VladHQ - January 1, 2014, 2:48 am

(Dec 21, 2013 02:15 AM)Mafugsa Bob Wrote: [\[CITE\]](#) ^that...

You shouldn't have to cache, minify or all of that on a project you're hosting on a shared host, as Wolf said.

But this is not just for shared hosting. If you combine this with GZIP and a cache plugin, it only benefits you.

Skidrowz Heir - January 2, 2014, 12:02 am

nice info man . already bookmark :)

?rthur - January 2, 2014, 5:14 am

(Dec 17, 2013 12:25 PM)VladHQ Wrote: [\[CITE\]](#) [*]Use PDO or mysqli_* instead of mysql_*

I know it's better to use PDO or mysqli these days but in the performance way it doesn't really matter.

<http://www.php.net/manual/en/mysqlinfo.api.choosing.php> Wrote:It is recommended to use either the mysqli or PDO_MySQL extensions. It is not recommended to use the old mysql extension for new development, as it has been deprecated as of PHP 5.5.0 and will be removed in the future. A detailed feature comparison matrix is provided below. The overall performance of all three extensions is considered to be about the same. Although the performance of the extension contributes only a fraction of the total run time of a PHP web request. Often, the impact is as low as 0.1%.

Further it's a nice tutorial

VladHQ - January 2, 2014, 2:56 pm

Thanks for the feedback, guys, I really appreciate it.

VladHQ - March 7, 2014, 8:03 am

Hate to see this thread go dead, I worked on this for a couple hours.

ASDA - March 8, 2014, 2:47 am

Cache assets and static files with Nginx and Varnish on frontend with Apache on backend. Optimise Apache configuration. Run an OPCode cacher (or maybe HipHop?). Run a data-store cacher.

Sir Paul McCartney - March 9, 2014, 10:39 am

Use single quotes (")

Use echo instead of print.

Could you expand on these two points? ' or " are same basically.. and same goes for echo and print? There's no difference I believe..

VladHQ - March 9, 2014, 11:14 am

(Mar 9, 2014 10:39 AM)Till Lindemann Wrote: [\[CITE\]](#) Use single quotes (")

Use echo instead of print.

Could you expand on these two points? ' or " are same basically.. and same goes for echo and print? There's no difference I believe..

Well since double-quotes go inside to check for PHP variables, it takes slightly more time.

So, say you have this:

PHP Code

```
<?php
```

```
$var = 'world';
```

```
echo "hello $var"; // slower
```

```
echo 'hello '.$var; // faster
```

Also, I read about the echo being faster the print but it's really nothing noticeable but still faster if you want that 0.0000001's second.

Sir Paul McCartney - March 9, 2014, 11:41 am

(Mar 9, 2014 11:14 AM)VladHQ Wrote: [\[CITE\]](#) Well since double-quotes go inside to check for PHP variables, it takes slightly more time.

So, say you have this:

PHP Code

```
<?php
```

```
$var = 'world';
```

```
echo "hello $var"; // slower
```

```
echo 'hello '.$var; // faster
```

Also, I read about the echo being faster the print but it's really nothing noticeable but still faster if you want that 0.0000001's second.

How much faster is the " and ' compared? and 0.0000001 doesn't bother me

K0tt3xZ - March 9, 2014, 12:17 pm

My favourite PHP performance tutorial so far, keep it up!

?

loonpwn - March 9, 2014, 11:18 pm

(Mar 9, 2014 11:41 AM)Till Lindemann Wrote: [\[CITE\]](#) How much faster is the " and ' compared? and 0.0000001 doesn't bother me

The main thing to remember is not to use " " when you don't have a string with a php variable inside it. The performance difference is based on how many strings you have and how big them strings are.

It won't ever be noticeable unless you have millions of strings.

Google released their own performance tips which builds on a few things said here:

<https://developers.google.com/speed/articles/optimizing-php>

One thing missed was don't use getters and setters for php when all they do is set/get the value. (making it public instead)

VladHQ - March 10, 2014, 5:38 am

(Mar 9, 2014 11:18 PM)loonpwn Wrote: [\[CITE\]](#) The main thing to remember is not to use " " when you don't have a string with a php variable inside it. The performance difference is based on how many strings you have and how big them strings are.

It won't ever be noticeable unless you have millions of strings.

Google released their own performance tips which builds on a few things said here:

<https://developers.google.com/speed/articles/optimizing-php>

One thing missed was don't use getters and setters for php when all they do is set/get the value. (making it public instead)

Ugh, I wish I coulda said that I haven't programmed in PHP in about a month because I am focused on music right now.

Private666 - March 11, 2014, 1:49 am

Very nice HQ and detailed tutorial thanks alot.

Eight - March 11, 2014, 8:58 am

> Avoid loops!

This. Screw iteration, use recursion instead. It's what the cool kids are all doing nowadays :

PHP Code

```
<?php
loop(3, function($i) {
    echo $i . PHP_EOL;
});

function loop($up_to, \closure $cb)
{
    if($up_to == 0)
        return;
    $cb($up_to);
    loop(--$up_to, $cb);
}
```

No seriously, "avoid loops" is a broad statement. I'd say use them, but use them intelligently. For example one thing I see people do sometimes is :

PHP Code

```
<?php
for($i = 0; $i < sizeof($array); $i++)
```

The problem here is that the length of the array gets computed in every iteration, and sometimes you don't really need your code to behave that way. Instead of doing this, one should use a foreach loop or at the very least pre-calculate the length of the array then store it in a temporary variable to use throughout the loop :

PHP Code

```
<?php
for($i = 0, $l = sizeof($array); $i < $l; $i++)
```

Another thing you should try to avoid as much as possible is to execute an SQL query inside of a loop. [I wrote something about it back in the day](#), and surprisingly it isn't as cringe inducing as I thought it would be (if you look past the mysql_ calls).

This trick isn't only applicable to databases, you can also do it for files, sockets, etc.

An example with file_put_contents :

PHP Code

```
<?php
```

```

for($i = 1; $i < 10; $i++)
{
    file_put_contents('file', $data, FILE_APPEND);
    // bad : it opens and closes the file each iteration, which introduces a certain overhead
}

$fh = fopen('file', 'a');
for($i = 1; $i < 10; $i++)
{
    fwrite($data); // better, opens the file one time only.
}
fclose($fh);

```

And finally : don't optimize prematurely. Focus on writing code that is functional, you can then optimize it once it's done being written, benchmarked AND you find it to be too slow for your needs. Micro optimization will only get you a few microseconds in this regard, so it's better to profile it in order to find the bottleneck, and then optimize that. For example, you might narrow it down to an SQL query that takes too long, and you can solve it by improving it if it's badly written, or by creating an index on the appropriate column.

I don't really know what's the idiomatic way to do that in PHP, but I heard xdebug offers that functionality.

Edit : And sorry for hijacking your thread ?

VladHQ - March 13, 2014, 10:20 am

(Mar 11, 2014 08:58 AM)Eight Wrote: [\[CITE\]](#) > Avoid loops!

This. Screw iteration, use recursion instead. It's what the cool kids are all doing nowadays :

PHP Code

```

<?php
loop(3, function($i) {
    echo $i . PHP_EOL;
});

function loop($up_to, \closure $cb)
{
    if($up_to == 0)
        return;
    $cb($up_to);
    loop(--$up_to, $cb);
}

```

No seriously, "avoid loops" is a broad statement. I'd say use them, but use them intelligently. For example one thing I see people do sometimes is :

PHP Code

```

<?php
for($i = 0; $i < sizeof($array); $i++)

```

The problem here is that the length of the array gets computed in every iteration, and sometimes you don't really need your code to behave that way. Instead of doing this, one should use a foreach loop or at the very least pre-calculate the length of the array then store it in a temporary variable to use throughout the loop :

PHP Code

```
<?php
for($i = 0, $l = sizeof($array); $i < $l; $i++)
```

Another thing you should try to avoid as much as possible is to execute an SQL query inside of a loop. [I wrote something about it back in the day](#), and surprisingly it isn't as cringe inducing as I thought it would be (if you look past the `mysql_` calls).

This trick isn't only applicable to databases, you can also do it for files, sockets, etc.

An example with `file_put_contents` :

PHP Code

```
<?php
for($i = 1; $i < 10; $i++)
{
    file_put_contents('file', $data, FILE_APPEND);
    // bad : it opens and closes the file each iteration, which introduces a certain overhead
}

$fh = fopen('file', 'a');
for($i = 1; $i < 10; $i++)
{
    fwrite($data); // better, opens the file one time only.
}
fclose($fh);
```

And finally : don't optimize prematurely. Focus on writing code that is functional, you can then optimize it once it's done being written, benchmarked AND you find it to be too slow for your needs. Micro optimization will only get you a few microseconds in this regard, so it's better to profile it in order to find the bottleneck, and then optimize that. For example, you might narrow it down to an SQL query that takes too long, and you can solve it by improving it if it's badly written, or by creating an index on the appropriate column.

I don't really know what's the idiomatic way to do that in PHP, but I heard `xdebug` offers that functionality.

Edit : And sorry for hijacking your thread ?

Oh, that's fine, thank you for contributing, I will link your post in my thread!

Mavspammer - March 14, 2014, 9:19 am

nice and good tutorial if i may say.. thanks

VladHQ - March 14, 2014, 10:51 am

(Mar 14, 2014 09:19 AM)Mavspammer Wrote: [\[CITE\]](#) nice and good tutorial if i may say.. thanks

You are welcome, man! If you need any help with simple tasks in PHP, feel free to PM me once you get the 25 posts!