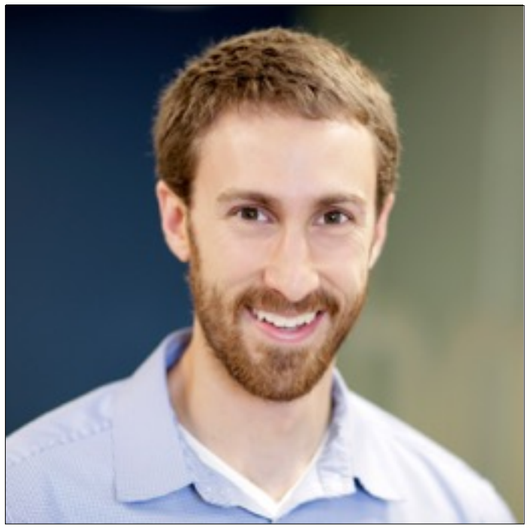
# High Performance PHP

## Optimizing PHP Code



Jonathan N. Klein

@jonathanklein | www.jonathanklein.net

# Topics to Be Covered

- Optimizing PHP code
- Choosing and configuring a Web server
- Database optimization
- Performance/load testing
- Frameworks and performance

# What We're NOT Covering

- ✗ Linux server configuration
- ✗ Front-end performance
- ✗ Hardware
- ✗ Queueing

# Demo Application

laravel

# Why Performance Matters

User Experience | Conversions | Scalability

# Performance Helps Scalability

Single Process

Each Request Takes 100ms

10 requests per second

Single Process

Each Request Takes 50ms

20 requests per second

# Performance Case Studies

Firefox
2.2s faster =
+10M downloads

Shopzilla
5s faster =
+7-12% conversions

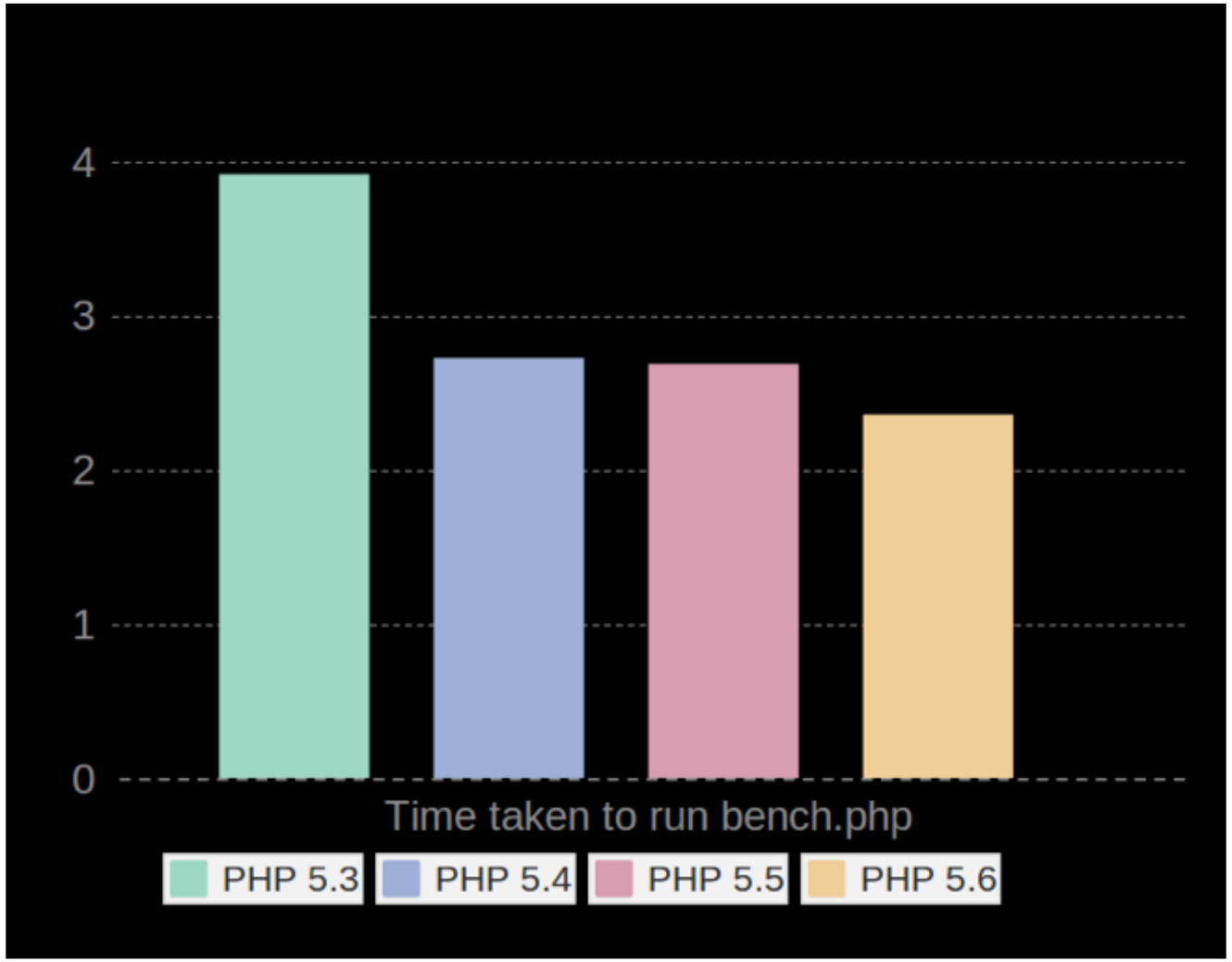
Bing
1s slower =
-2.8% revenue

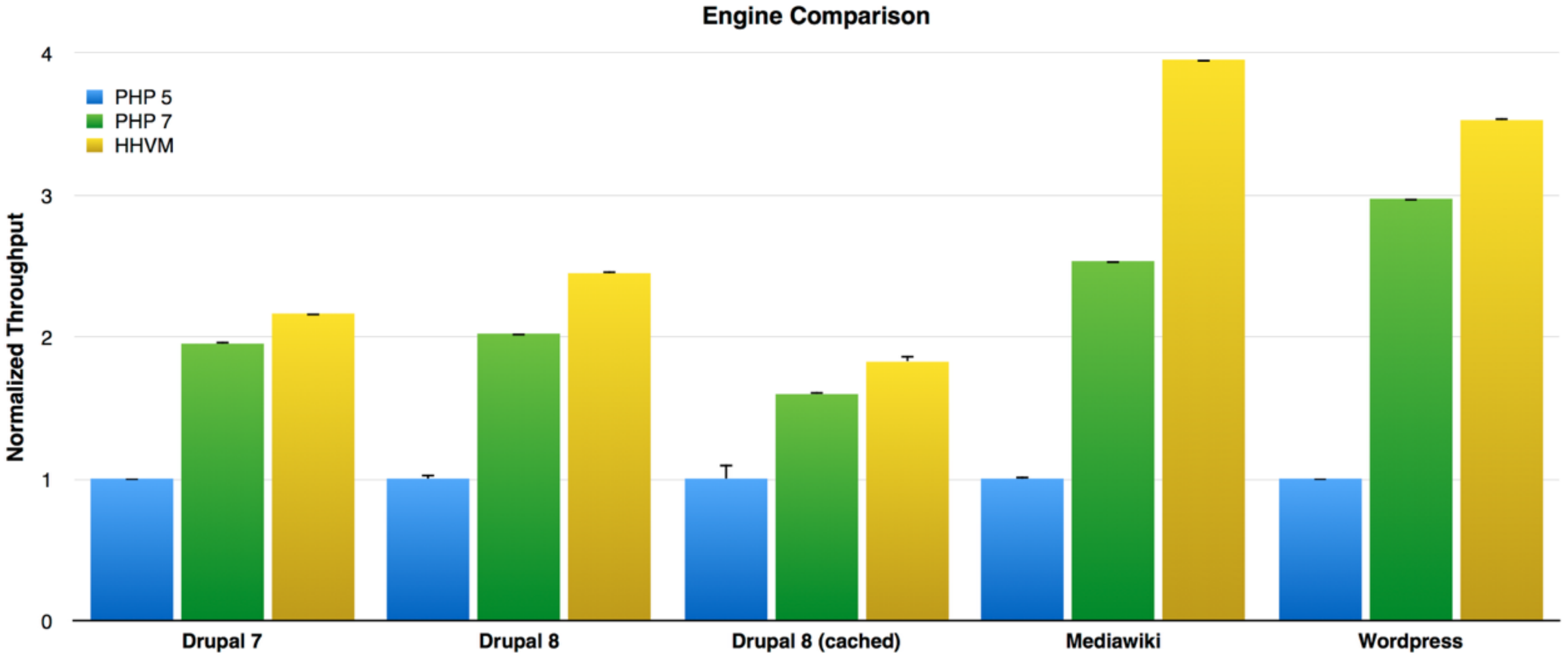
Yahoo!
0.4s slower =
-5-9% traffic

# PHP Versions

(Use the Latest One)

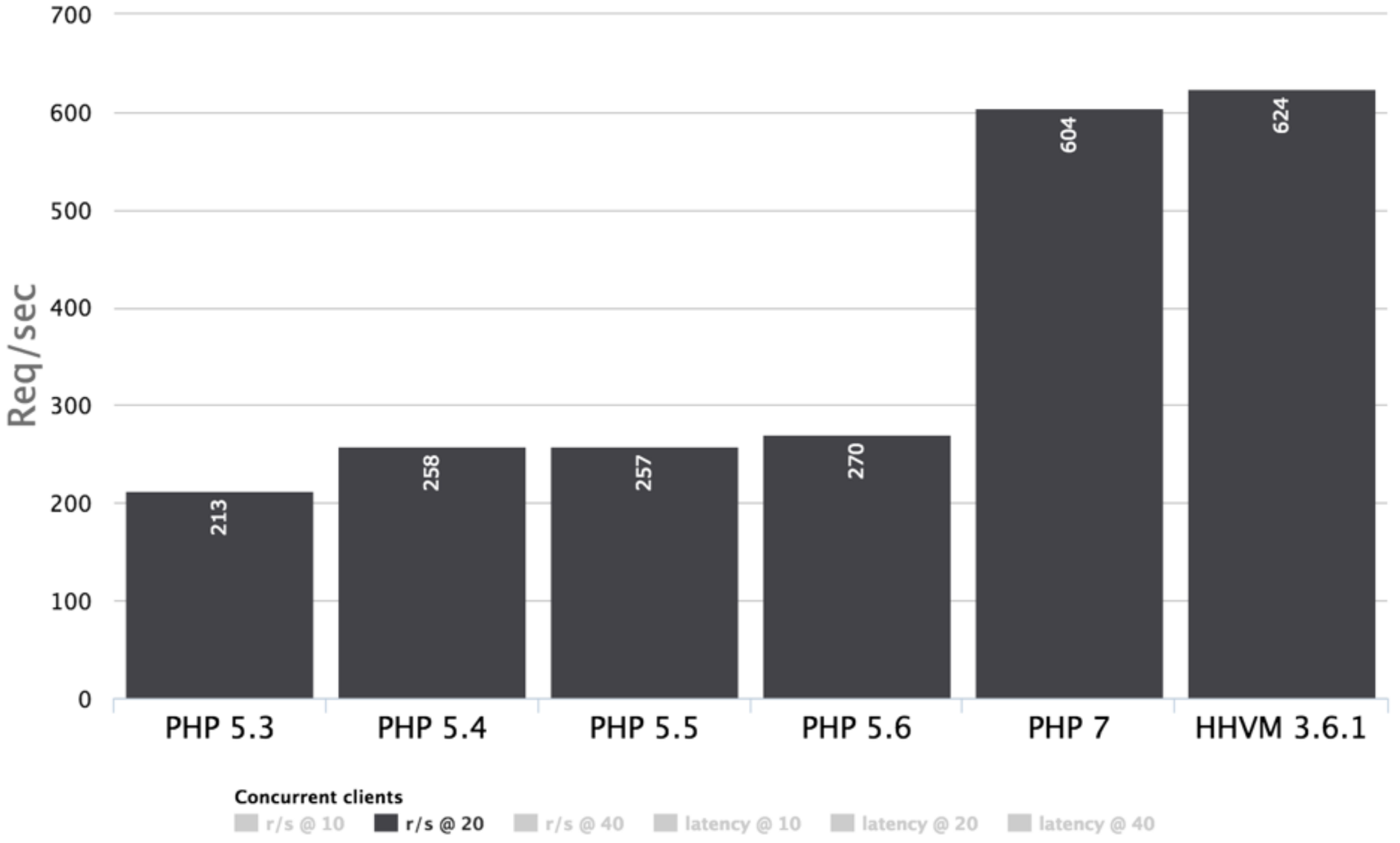
# PHP Performance by Version

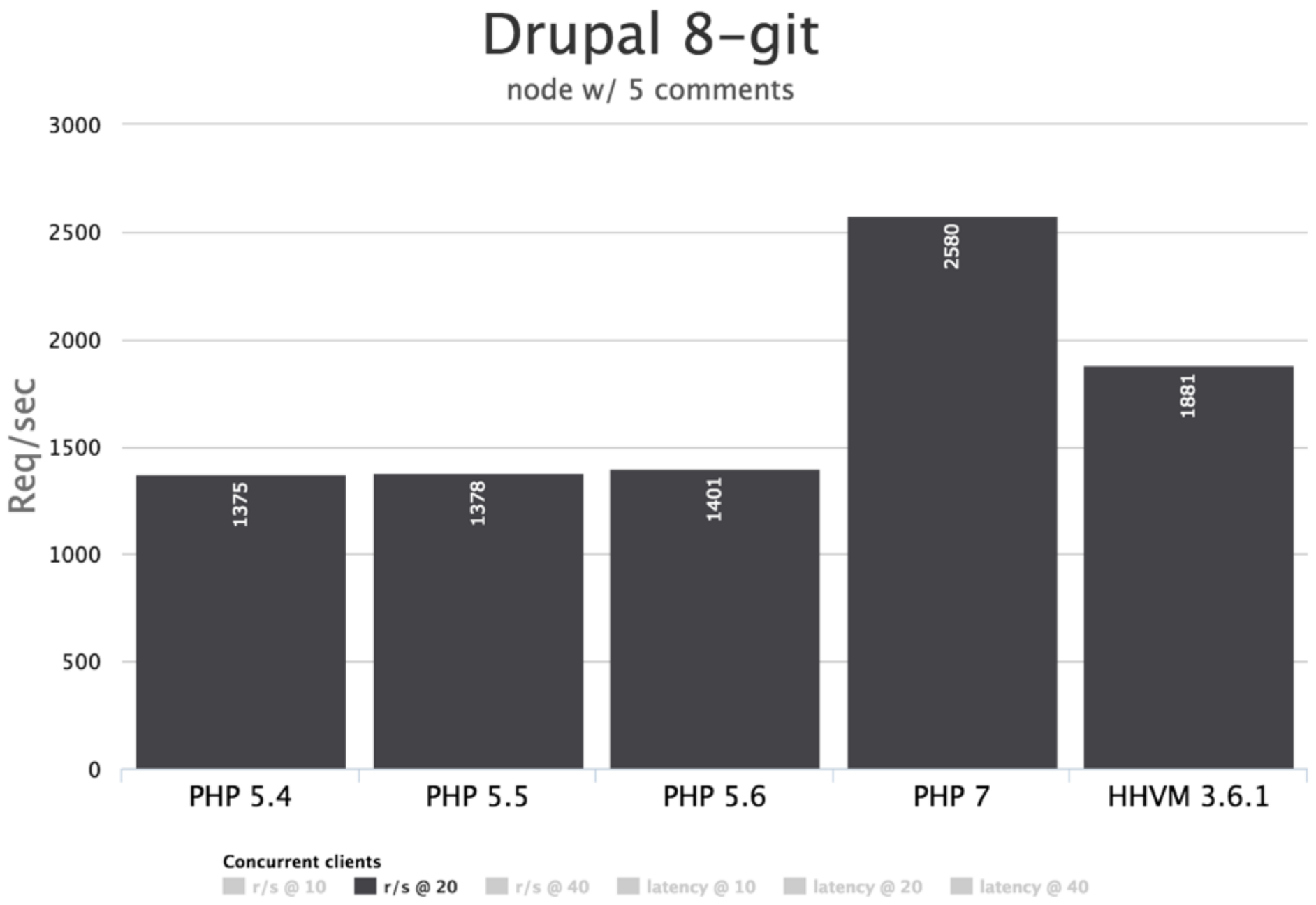



Engine Comparison
PHP 5
PHP 7
HHVM
Normalized Throughput
0
1
2
3
4
Drupal 7
Drupal 8
Drupal 8 (cached)
Mediawiki
Wordpress

# Wordpress-4.1.1

http://wordpress/?p=1




Drupal 8-git
node w/ 5 comments
Req/sec
3000
2500
2000
1500
1000
500
0
1375
1378
1401
2580
1881
PHP 5.4
PHP 5.5
PHP 5.6
PHP 7
HHVM 3.6.1
Concurrent clients
r/s @ 10
r/s @ 20
r/s @ 40
latency @ 10
latency @ 20
latency @ 40

You should be planning to move to either PHP7 or HHVM in 2016.

— Me, Right Now

# PHP Micro-Optimizations

## Largely a Fool's Errand

## Variable Type Checking

# isSet() vs. empty() vs. is_array()

What is the performance of isSet() and empty(). Call 2'000x

| | | | |
|---|---|---|---|
| + 109 % | isSet() with var that was set | Total time: 51 µs | *view code* |
| + 105 % | empty() with var that was set | Total time: 49 µs | *view code* |
| + 105 % | isSet() with var that was *not* set | Total time: 49 µs | *view code* |
| + 100 % | empty() with var that was *not* set | Total time: 47 µs | *view code* |
| + 105 % | isSet() with array-var that was set | Total time: 49 µs | *view code* |
| + 113 % | empty() with array-var that was set | Total time: 53 µs | *view code* |
| + 100 % | isSet() with array-var that was *not* set | Total time: 47 µs | *view code* |
| + 100 % | empty() with array-var that was *not* set | Total time: 47 µs | *view code* |
| + 390 % | is_array() of an array | Total time: 183 µs | *view code* |
| + 388 % | is_array() of a string | Total time: 182 µs | *view code* |
| + 1478 % | is_array() of a non set value | Total time: 694 µs | *view code* |
| + 1367 % | isSet() AND is_array() of a non set value | Total time: 642 µs | *view code* |

## Conclusion:

isSet() and empty() are identical. So alway check if val is set at all befor using type-checking. E.g. if (isSet($foo) AND is_array($foo))

## Quote Types

### double (") vs. single (') quotes

Is a there a difference in using double (") and single (') quotes for strings. Call 1'000x

| | | | |
|---|---|---|---|
| + 106 % | single (') quotes. Just an empty string: $tmp[] = ''; | Total time: 70 µs | *view code* |
| + 101 % | double (") quotes. Just an empty string: $tmp[] = ""; | Total time: 67 µs | *view code* |
| + 100 % | single (') quotes. 20 bytes Text : $tmp[] = 'aaaaaaaaaaaaaaaaaaaa'; | Total time: 66 µs | *view code* |
| + 101 % | double (") quotes. 20 bytes Text : $tmp[] = "aaaaaaaaaaaaaaaaaaaa"; | Total time: 67 µs | *view code* |
| + 100 % | single (') quotes. 20 bytes Text and 3x a $ : $tmp[] = 'aa $ aaaa $ aaaa $ a'; | Total time: 66 µs | *view code* |
| + 100 % | double (") quotes. 20 bytes Text and 3x a $ : $tmp[] = "aa $ aaaa $ aaaa $ a"; | Total time: 66 µs | *view code* |
| + 120 % | double (") quotes. 20 bytes Text and 3x a \$ : $tmp[] = "aa \$ aaaa \$ aaaa \$ a"; | Total time: 79 µs | *view code* |

### Conclusion:

In today's versions of PHP it looks like this argument has been satisfied on both sides of the line. Lets all join together in harmony in this one!

## Counting Loops

### For-loop test

Is it worth the effort to calculate the length of the loop in advance?
e.g. "for ($i=0; $i<$size; $i++)" instead of "for ($i=0; $i<sizeOf($x); $i++)"
A loop with 1000 keys with 1 byte values are given.

| | | | |
|---|---|---|---|
| + 105 % | With pre calc - count() | Total time: 62 µs | *view code* |
| + 50815 % | Without pre calc - count() | Total time: 29925 µs | *view code* |
| + 100 % | With pre calc - sizeof() | Total time: 59 µs | *view code* |
| + 50466 % | Without pre calc - sizeof() | Total time: 29719 µs | *view code* |

### Conclusion:

Unsurprising results... this is one of the easiest things to implement in any application and is the widest agreed upon benchmarking item within the online PHP community. The results basically speak for themselves.

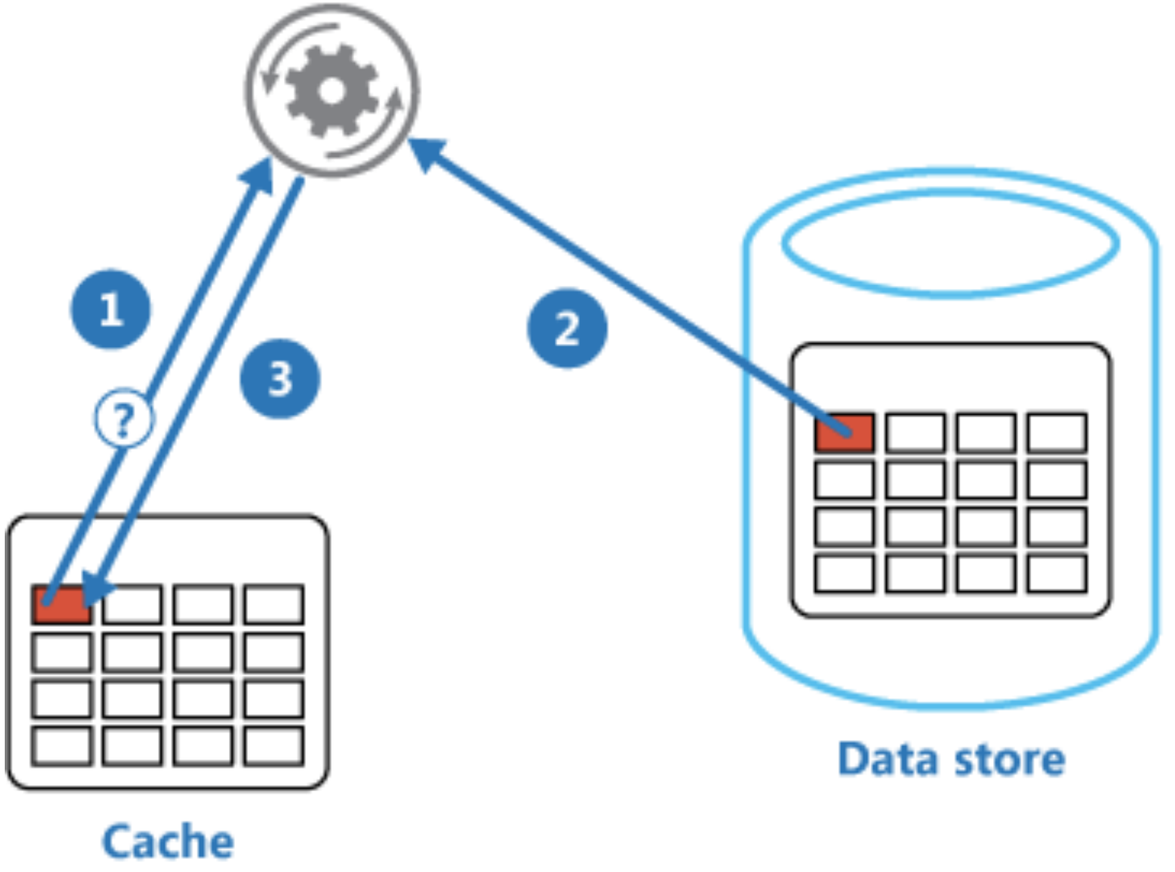
If Micro-Optimizations are worthless, how do we make PHP code fast?

# XHProf

## Profiling PHP Code

# Redis

## Drop in Replacement for Memcached




1: Determine whether the item is currently held in the cache.
2: If the item is not currently in the cache, read the item from the data store.
3: Store a copy of the item in the cache.

# Why Is Redis Better than Memcached?

- Syncs data to disk
- Includes sets, lists, hashes, and other useful data structures
- Built-in master/slave replication

# Summary

Performance Matters

Use the Latest Version of PHP

Avoid Focusing on Micro-Optimizations

Use XHProf

Cache with Redis

# Glossary

- Lornajane's PHP 5.6 Benchmarks
- HHVM Lockdown Results & Performance
- Firefox & Page Load Speed
- The Performance Business Pitch
- Friendster and Scalability
- PHP Micro-Benchmarks
- Cache Aside Pattern