facebook

Scaling PHP with HHVM

And making development in PHP less annoying

Sara Golemon
http://www.hhvm.com
Nov 08, 2013

Agenda

1. From Problem to Solution
2. The History of HipHop
3. Bringing up a new Server
4. Why (not) HHVM
5. Questions

From Problem to Solution

Web serving at Facebook

- Millions of lines of PHP, growing exponentially
- Front-end memory bound (Apache prefork)
- Pointless copying around of data (APC, MySQL, etc...)
- Blocking actions (File, Database, Network, ...)
- Hundreds of millions of users (now over a Billion)
- Most layers of LAMP really starting to hurt

Low hanging fruit

Do the easy stuff right, first

- UPGRADE PHP!
  - Stop using 5.2/5.3
- Database tuning
  - Heavy use of caching
- Efficient, modular code
- Stronger/More hardware

PHP 5.4 is awesome (sorta)

- Big performance wins
  - zend_op
  - znode
  - zend_op_array
  - zend_class_entry
- APC... yeah, about that...

PHP 5.5 is awesomerish

- Even more perf!
- Zend Opcache Plus
- 5.4 and 5.5 also bring lots of new toys and features

PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

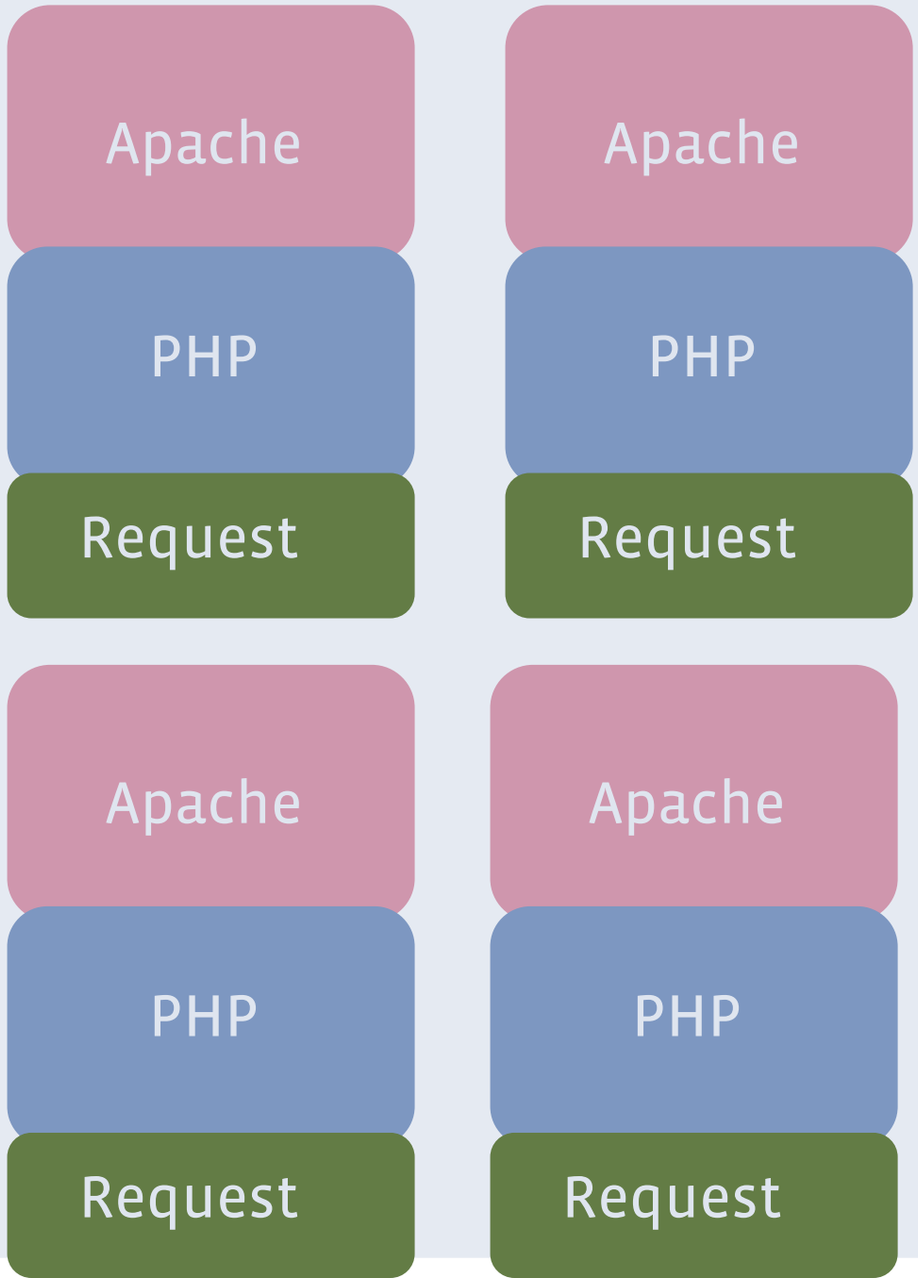
PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

- CPU Bound
- Memory Bound
- I/O Bound

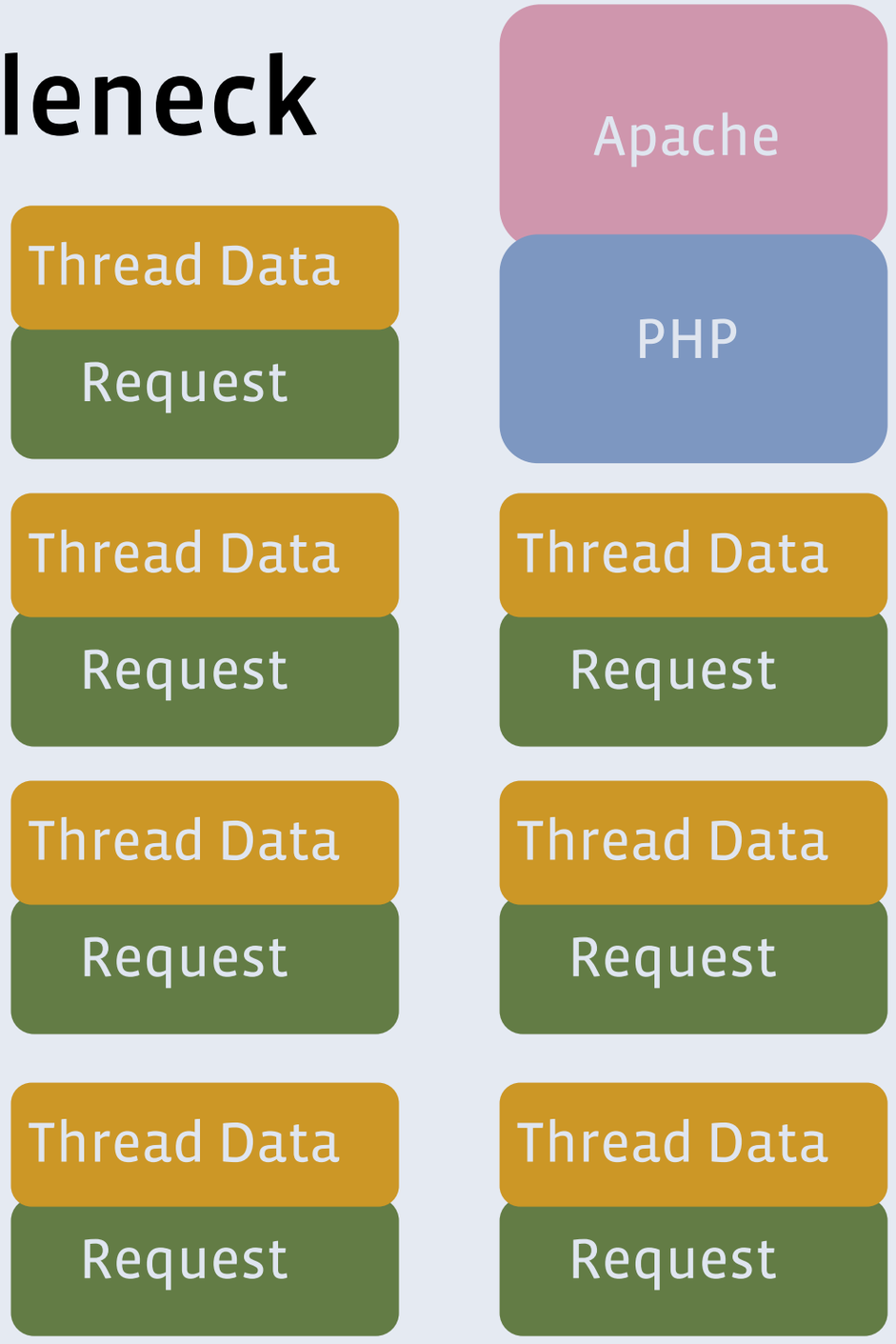
PHP's share of the bottleneck

- **Memory bound with APE Prefork**
- Type juggling/anonymity
- Interpreter overhead



PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead



PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

```
<?php
  $three  = 1 + 2;
  $twelve = 4 * $three;

  1. Make op1 an int
  2. Make op2 an int
  3. Add op1 + op2
  4. Store result to $three
  5. Make op3 an int
  6. Fetch op4 from $three
  7. Make op4 an int
  8. Multiply op3 * op4
  9. Store result to $twelve
```

PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

```
<?php
  int $three  = 1 + 2;
  int $twelve = 4 * $three;




  3. Add op1 + op2
  4. Store result to $three



  8. Multiply op3 * op4
  9. Store result to $twelve
```

PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

```
<?php

  int $twelve = 12;
```

```
9. Store 12 in $twelve
```

PHP's share of the bottleneck

- Memory bound with APE Prefork
- Type juggling/anonymity
- Interpreter overhead

Escaping the bottleneck

- Threaded web server
- Type inference
- Optimization and code locality
- Native JIT compilation
- Shared r/o code segments

Removing Apache

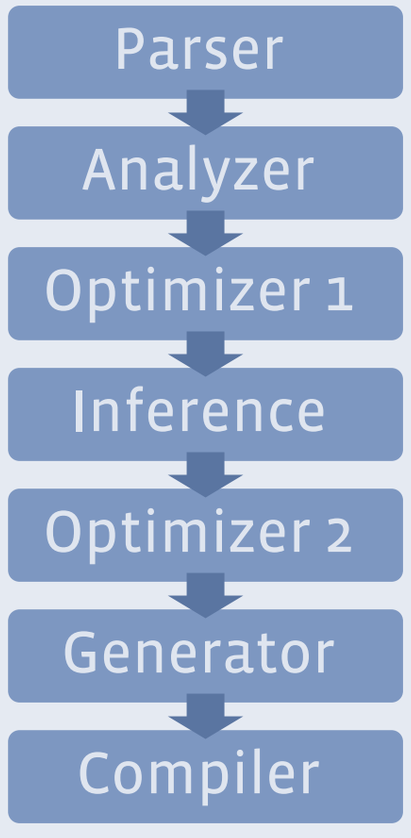
- Currently runs atop libevent
- Multi-threaded (and safe)
- Conf integrated with HPHP
- Plans to expand SAPI layer to Proxygen, FastCGI, and Apache2

```
PidFile = /var/www.pid

Server {
  Port = 80
  SourceRoot = /var/www/
  DefaultDocument = index.php
}

Eval {
  Jit = true
  JitASize = 1073741824
  JitAStubsSize = 268435456
  JitGlobalDataSize = 268435456
}

Log {
  Level = Warning
  NoSilenver = true
  AlwaysLogUnhandledExceptions = true
  RuntimeErrorReportingLevel = 8191
  UseLogFile = true
  UseSyslog = false
  File = /var/error.log
  Access {
    * {
      File = /var/access.log
      Format = %h %l %u % t \"%r\" %>s %b
    }
  }
}

AdminServer {
  Port = 8080
  password = superSecret
}
```

The History of HipHop

HPHPc – The Compiler

- Source analysis of PHP script
- Type inference & optimization
- Output C++ and pass to make
- Builds a single massive binary



- Slow and painful for dev

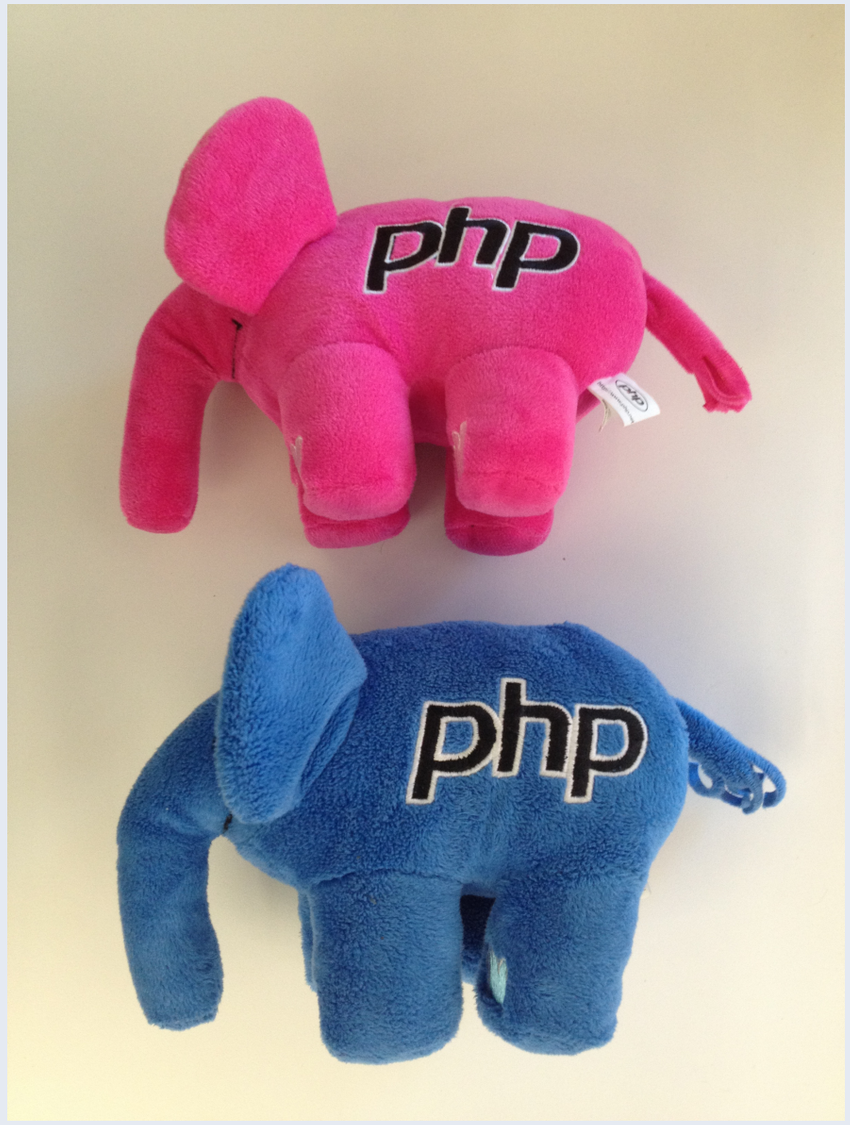
HPHPc + APE (Apache PHP Environment)

- Slight inconsistencies

- New language features

XHP

Generators

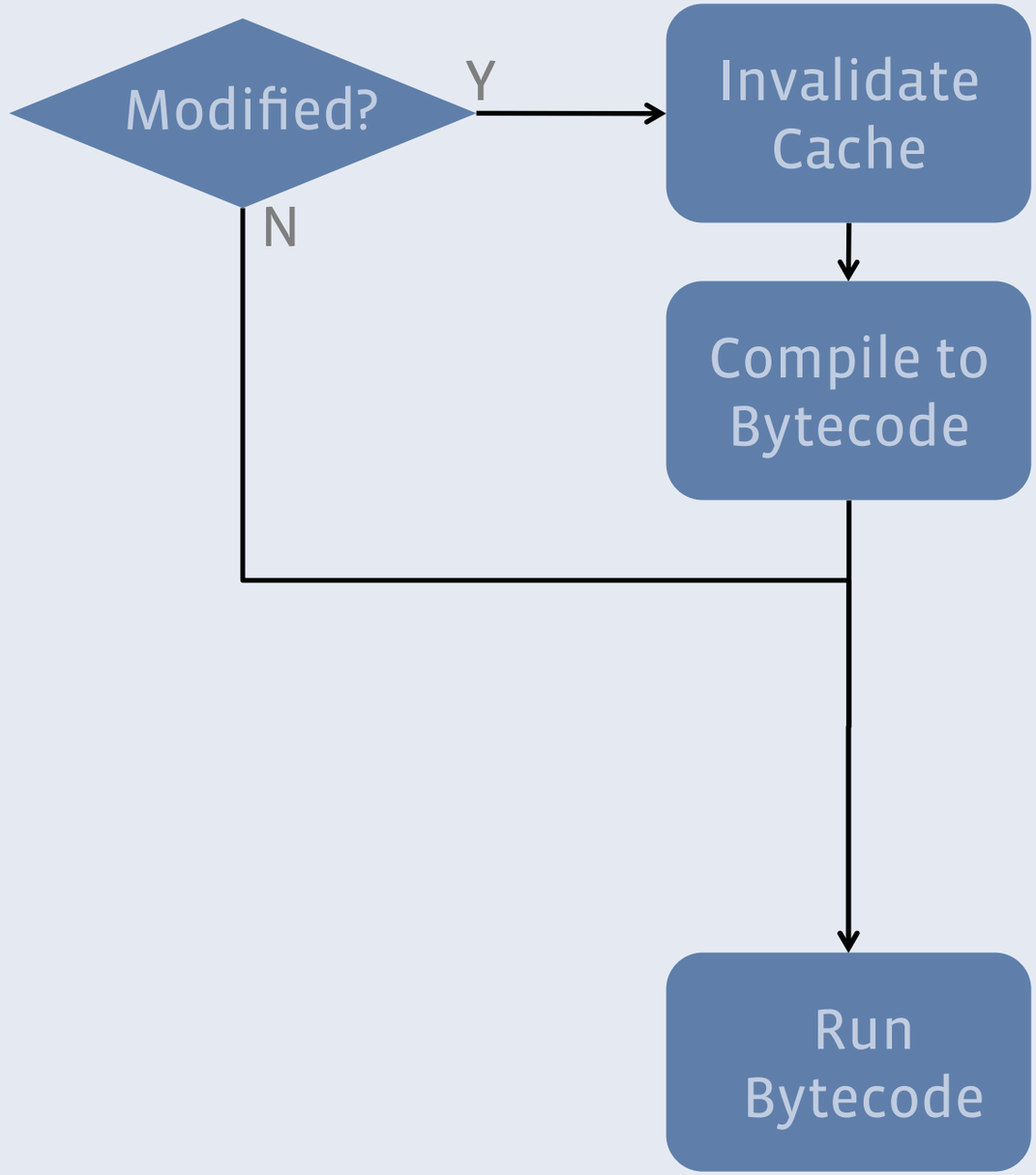


HPHPi – Interpreter Mode

- PHP2 style execution
- Avoid lengthy HPHPc compile
- Slower than PHP
- “Good enough” for dev mode
- Still minor inconsistencies

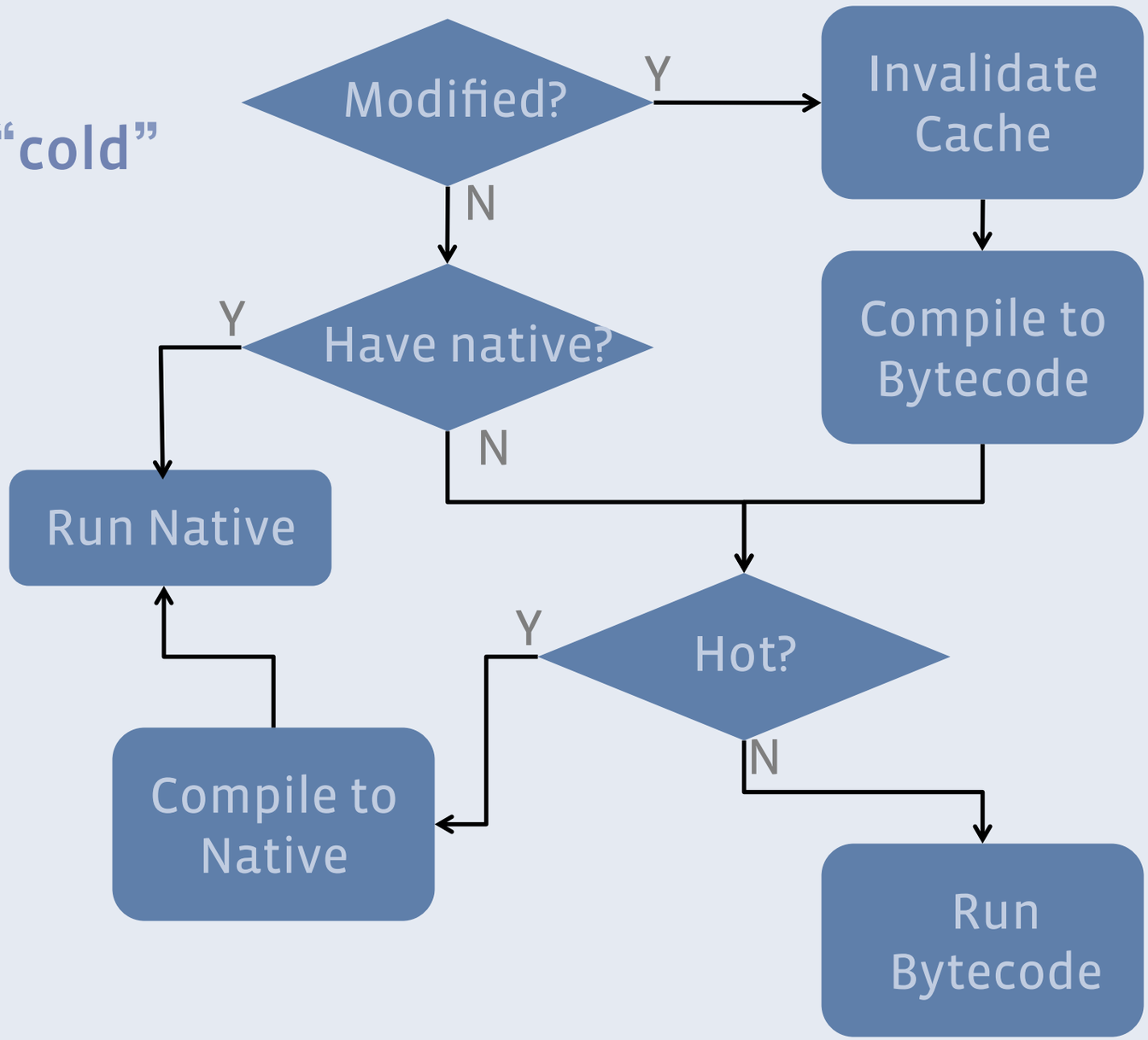
HHVM – Bytecode interpreter

- PHP5 style bytecode execution
- APC-like caching of bytecodes



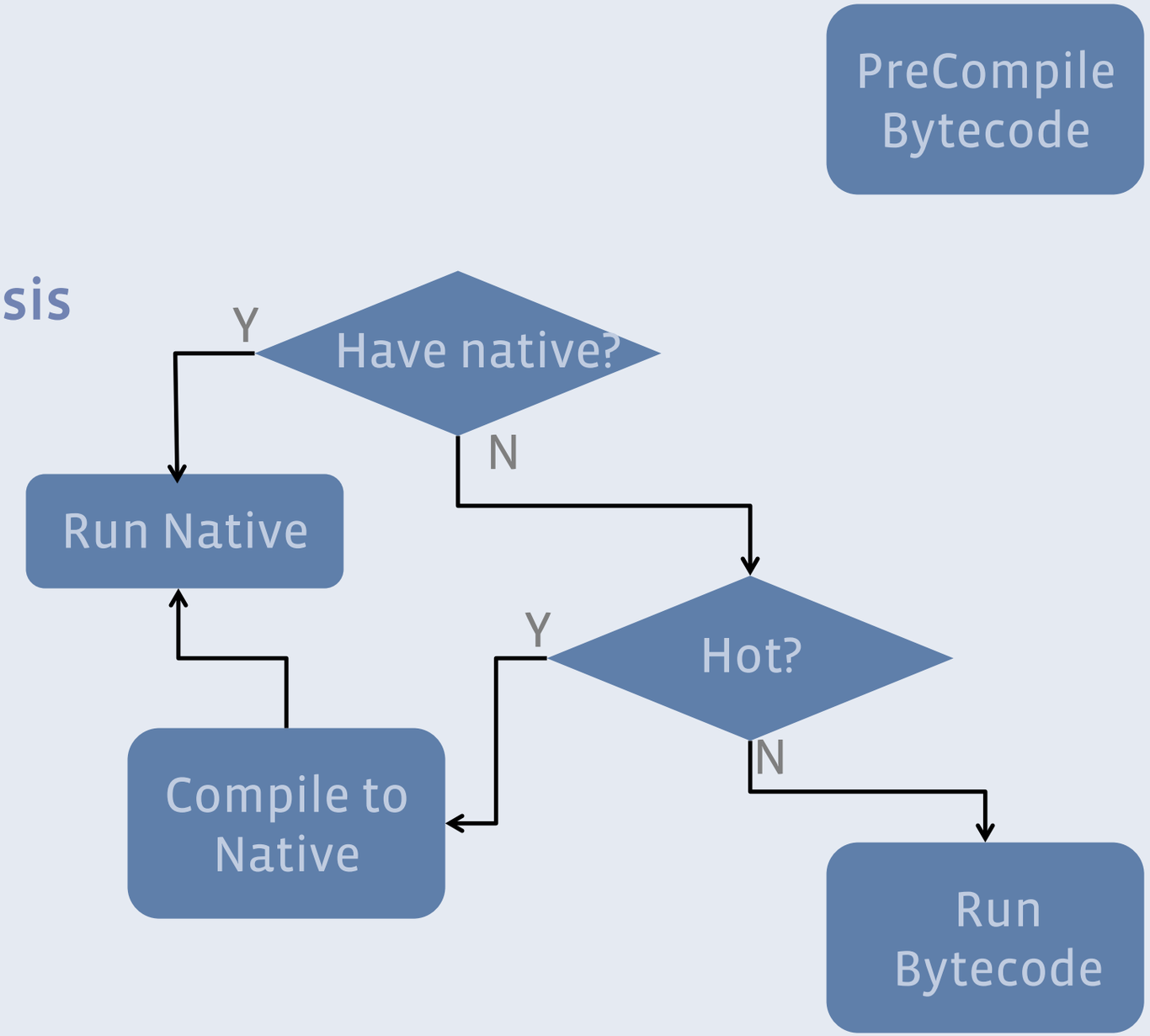
HHVM – Native code JIT

- Bytecodes run a few times “cold”
- Variable type inference
- Hotpath detection
- Transform to native code



HHVM – Repo Authoritative Mode

- "Production Mode"
- Improved offline pre-analysis
- Assumes no changes



HHVM – At Facebook Today

- 100% of Production Traffic
- 100% gain over HPHPc
  - Fewer Servers
  - Faster TTI for users
- >500% performance gain over PHP*
- HPHPc removed from HipHop
- GitHub branch “hphpc”

Bringing up a new Server

Installing HHVM from binaries

- Ubuntu 12.04, 12.10, 13.04, 13.10
- Add to /etc/apt/sources.list
  `deb http://dl.hhvm.com/ubuntu precise main`
- apt-get update
- apt-get install hhvm
- Mint 15
- Debian 7
- Fedora 19
- CentOS 6.4

Installing HHVM from source

- May whatever God you believe in have mercy on your soul.
- Refer to the github wiki for dependencies
- gcc >= 4.7 required, curl >= 7.28.0 recommended
- libevent must be patched for built-in webserver
- Some libraries not found on most distributions

Starting the server

- All files parsed as PHP
- Bytecode cache stored in user's home directory
- Files served from CWD
- PID file written to CWD
- JIT enabled
- Non-RepoAuthoritative

```
$ hhvm --mode server
```

Typical Dev Config

- Produce logs
- Ignore PHP in static files
- Serve from a webroot
- --daemon (background)

```
$ hhvm --mode daemon \
  --user web \
  --config /var/config.hdf
```

```
PidFile = /var/www.pid

Server {
  Port = 80
  SourceRoot = /var/www/
  DefaultDocument = index.php
}

Log {
  Level = Warning
  NoSilenver = true
  AlwaysLogUnhandledExceptions = true
  RuntimeErrorReportingLevel = 8191
  UseLogFile = true
  UseSyslog = false
  File = /var/error.log
  Access {
    * {
      File = /var/access.log
      Format = %h %l %u % t \"%r\" %>s %b
    }
  }
}

#include "/usr/share/hhvm/hdf/static.mime-types.hdf"
StaticFile {
  FilesMatch {
    * {
      pattern = .*\.(dll|exe)
      headers { * = Content-Disposition: attachment }
    }
  }
  Extensions : StaticMimeTypes
}
```

Typical Prod Config

- Quieter logs
- Enable the Admin Server
- Use pre-analyzed Repo

```
#include “/var/config.hdf”

Log {
  Level = Error
}

Repo {
  Authoritative = true
  Central {
    Path = /var/myapp.hhbc.sq3
  }
}
```

```
// Repo will be in /tmp/hphp_XXXXX/myapp.hhbc.sq3
$ hhvm --hphp \
       --target hhbc --input-list /tmp/files.lst \
       --output-file myapp.hhbc.sq3 –k1
$ mv /tmp/hphp/XXXXX/myapp.hhbc.sq3 /var/
$ hhvm --mode daemon --user web \
       --config /var/config-prod.hdf
```

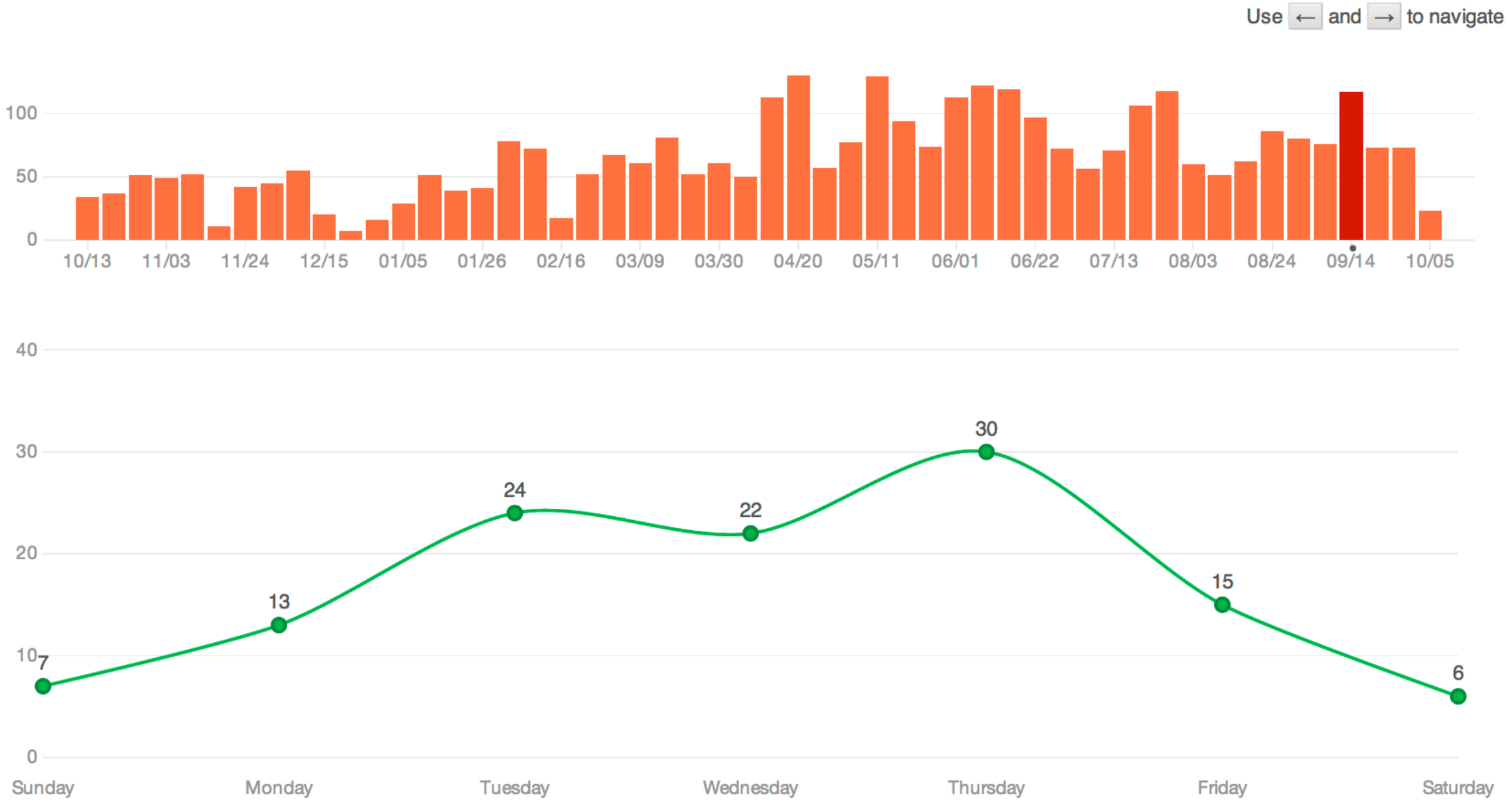
Why (not) HHVM

Why NOT HHVM

- HHVM is fast, but it doesn't run my code
  http://hhvm.com/blog/875/wow-hhvm-is-fast-too-bad-it-doesnt-run-my-code
- Perf is not my concern/problem
- I fear change / Facebook will give the NSA a back-door onto my servers

http://hhvm.com/repo/graphs/commit-activity

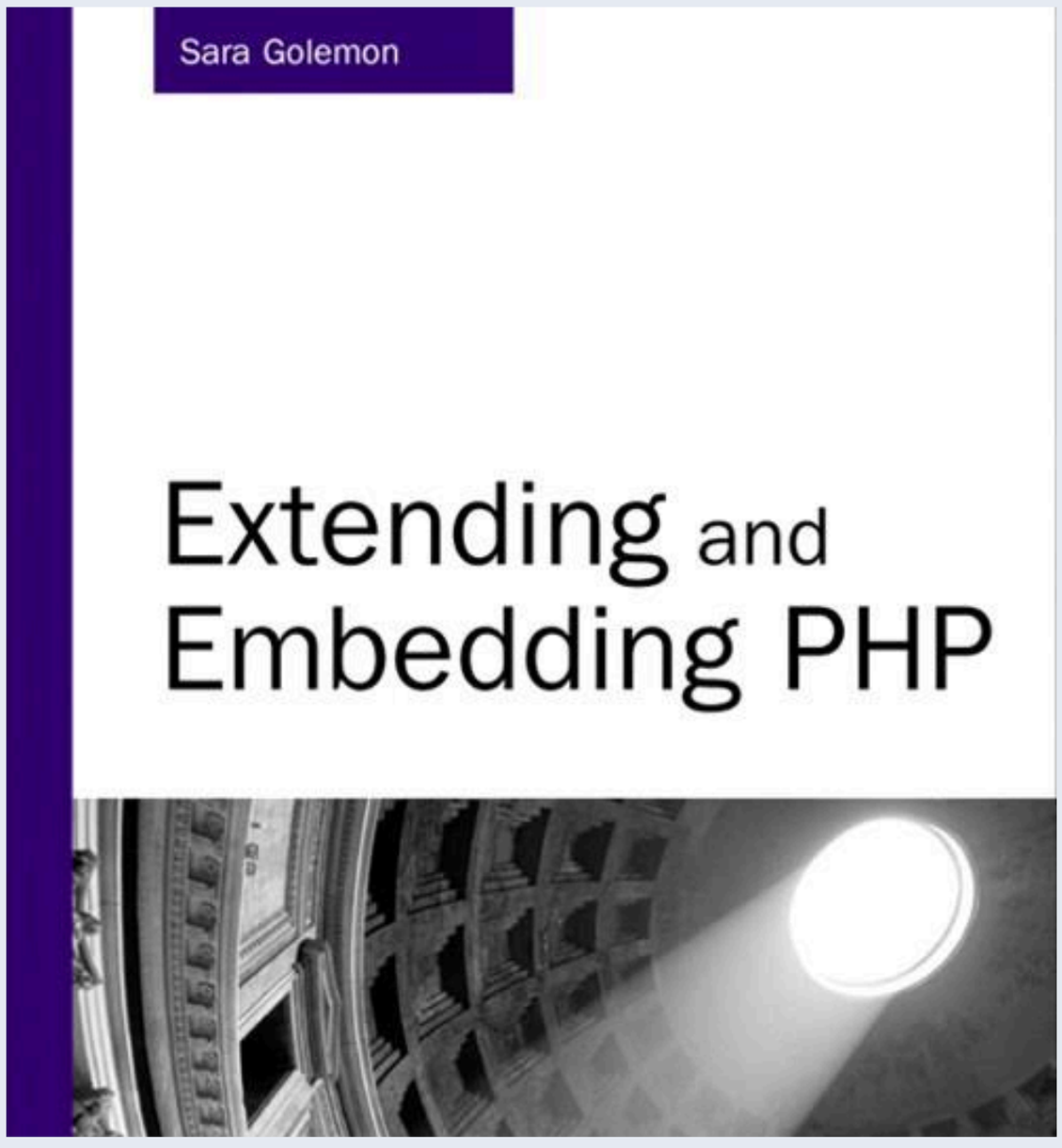
Contributors | Commits | Code Frequency | Punchcard

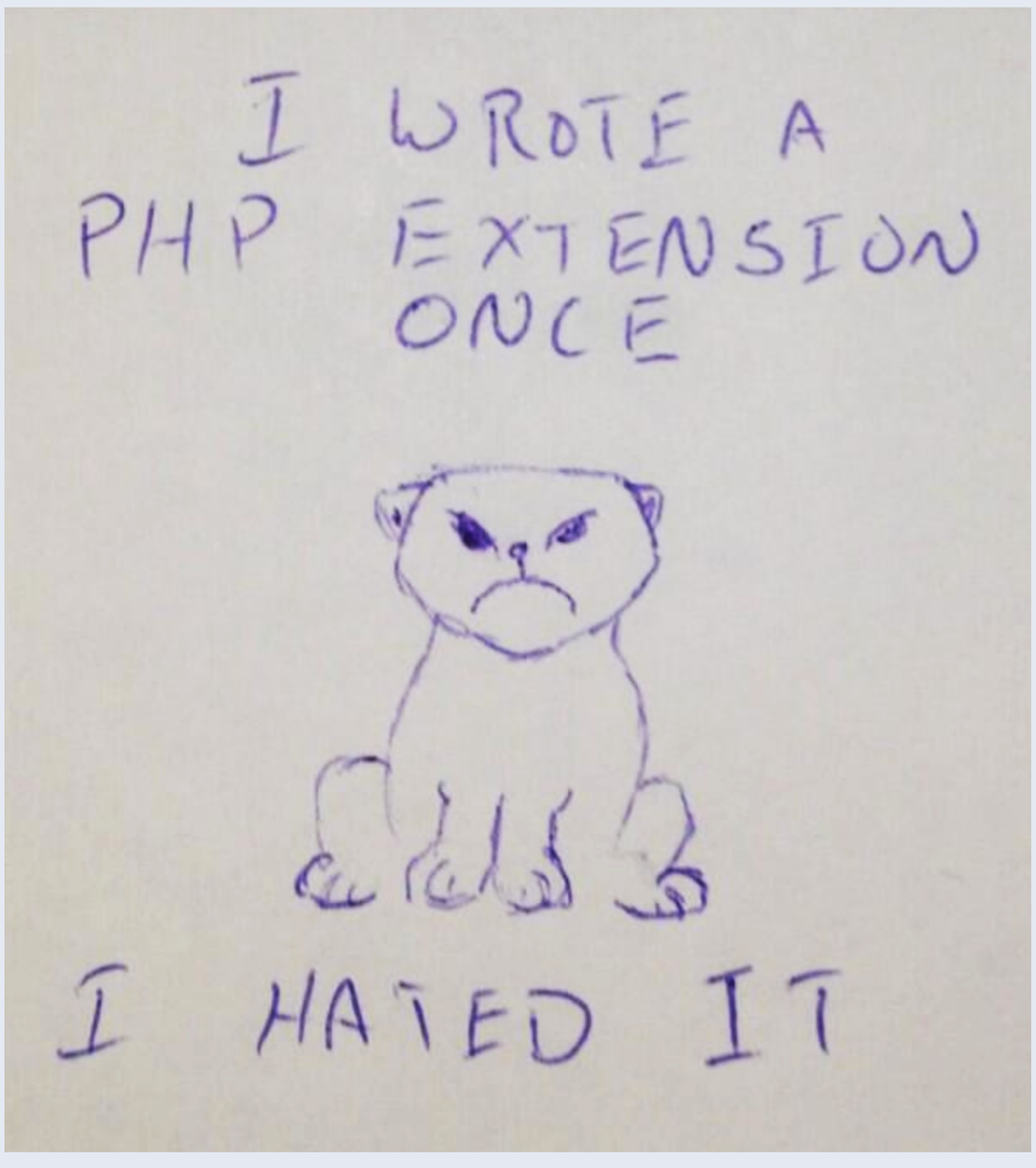


Why HHVM

- 2x – 10x performance boost
- MUCH easier to hack
- Language Features not in PHP
  - Generators
  - Generics
  - Strict Typing
  - Collections
  - Async Routines
  - [redacted]
  - [redacted]

Extending PHP





Writing a PHP Extension...

```
zend_bool array_column_param_helper(zval **param,
                                    const char *name TSRMLS_DC) {
  switch (Z_TYPE_PP(param)) {
    case IS_DOUBLE: convert_to_long_ex(param);
    case IS_LONG: return 1;
    case IS_OBJECT: convert_to_string_ex(param);
    case IS_STRING: return 1;
    default: php_error_docref(NULL TSRMLS_CC, E_WARNING, "The %s key should be either a string or an integer", name);
             return 0;
  }
}
PHP_FUNCTION(array_column) {
  zval **zcolumn = NULL, **zkey = NULL, **data;
  HashTable *arr_hash;
  HashPosition pointer;

  if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "hZ!|Z!", &arr_hash, &zcolumn, &zkey) == FAILURE) { return; }
  if ((zcolumn && !array_column_param_helper(zcolumn, "column" TSRMLS_CC)) ||
      (zkey && !array_column_param_helper(zkey, "index" TSRMLS_CC))) {
    RETURN_FALSE;
  }
  array_init(return_value);
  for (zend_hash_internal_pointer_reset_ex(arr_hash, &pointer);
       zend_hash_get_current_data_ex(arr_hash, (void**)&data, &pointer) == SUCCESS;
       zend_hash_move_forward_ex(arr_hash, &pointer)) {
    zval **zcolval, **zkeyval = NULL;
    HashTable *ht;
    if (Z_TYPE_PP(data) != IS_ARRAY) { continue; }
    ht = Z_ARRVAL_PP(data);
    if (!zcolumn) { zcolval = data; }
    else if ((Z_TYPE_PP(zcolumn) == IS_STRING) &&
             (zend_hash_find(ht, Z_STRVAL_PP(zcolumn), Z_STRLEN_PP(zcolumn) + 1, (void**)&zcolval) == FAILURE)) { continue; }
    else if ((Z_TYPE_PP(zcolumn) == IS_LONG) &&
             (zend_hash_index_find(ht, Z_LVAL_PP(zcolumn), (void**)&zcolval) == FAILURE)) { continue; }
    if (zkey && (Z_TYPE_PP(zkey) == IS_STRING)) { zend_hash_find(ht, Z_STRVAL_PP(zkey), Z_STRLEN_PP(zkey) + 1, (void**)&zkeyval); }
    else if (zkey && (Z_TYPE_PP(zkey) == IS_LONG)) { zend_hash_index_find(ht, Z_LVAL_PP(zkey), (void**)&zkeyval); }
    Z_ADDREF_PP(zcolval);
    if (zkeyval && Z_TYPE_PP(zkeyval) == IS_STRING) { add_assoc_zval(return_value, Z_STRVAL_PP(zkeyval), *zcolval); }
    else if (zkeyval && Z_TYPE_PP(zkeyval) == IS_LONG) { add_index_zval(return_value, Z_LVAL_PP(zkeyval), *zcolval); }
    else if (zkeyval && Z_TYPE_PP(zkeyval) == IS_OBJECT) {
      SEPARATE_ZVAL(zkeyval);
      convert_to_string(*zkeyval);
      add_assoc_zval(return_value, Z_STRVAL_PP(zkeyval), *zcolval);
    } else {
      add_next_index_zval(return_value, *zcolval);
    }
  }
}
```

The same function in HHVM

```
<?php

function array_column(array $arr, $col, $key = null) {
  $ret = [];
  foreach($arr as $key => $val) {
    if (!is_array($val)) continue;
    $cval = ($col === null) ? $val : $val[$col];
    if ($key === null || !isset($val[$key])) {
      $ret[] = $cval;
    } else {
      $ret[$val[$key]] = $cval;
    }
  }
  return $ret;
}
```

Crossing the PHP->C++ barrier

```
Array HHVM_FUNCTION(array_column, CArrRef arr, CVarRef col, CVarRef key) {
  Array ret;
  for (auto &pair : arr) {
    Variant key = pair.first, val = pair.second;
    if (val.isArray()) continue;
    Array aval = val.toArray();
    Variant cval = col.isNull() ? aval : aval[col];
    if (key.isNull() || !aval.exists(key)) {
      ret.append(cval);
    } else {
      ret[aval[key]] = cval;
    }
  }
  return ret;
}
```

Blurring the PHP/C++ line

```
class MyClass {
  public static function escapeString(string $str): string {
    return str_replace(“’”, “\\’”, $str);
  }

  <<__Native>>
  public funtion query(string $sql): array;
}
```

```
Array HHVM_METHOD(MyClass, query, const String& sql) {
  // Call C/C++ library function and return result…
}
```

XHP

- XML as a first-class construct
- Parser knows context
- Data escaped, say goodbye XSS
- Compostable elements
- Extensible

```
<?php

include “/usr/share/hhvm/xhp/init.php”

$good = <p>Welcome, {$_GET[‘username’]}!</p>;
$evil = “<script>alert();</script>”;

echo <html>
       <head>
         <title>Hello World!</title>
       </head>
       <body>
         {$good}
         {$evil}

       </body>
     </html>;
```

```
<html><head><title>Hello World!</title>
</head><body><p>Welcome, SaraMG!</p>
&lt;script&gt;alert()&lt;/script&gt;
</body></html>
```

XHP

- XML as a first-class construct
- Parser knows context
- Data escaped, say goodbye XSS
- Compostable elements
- Extensible

```
<?php

include “/usr/share/hhvm/xhp/init.php”

$good = <p>Welcome, {$_GET[‘username’]}!</p>;
$evil = “<script>alert();</script>”;

echo <html>
       <head>
         <title>Hello World!</title>
       </head>
       <body>
         {$good}
         {$evil}
         {HTML($evil)}
       </body>
     </html>;
```

```
<html><head><title>Hello World!</title>
</head><body><p>Welcome, SaraMG!</p>
&lt;script&gt;alert()&lt;/script&gt;
<script>alert()</script></body></html>
```

XHP

```
<?php

class :ui:menu extends :xhp:html-element {
  attribute string title;
  children (ui:menuitem)*;

  public function stringify() {
    $ret = ‘<div onclick=“menuExpand(this)”>’ .
           ‘<span class=“title”>’ .
             htmlspecialchars($this->title) .
           ‘</span>’;
    foreach($this->getChildren as $child) {
      $ret .= :x:base::renderChild($child);
    }
    return $ret . ‘</div>’;
  }
}

Class :ui:menuitem extends :xhp:html-element {
  attribute string $title, string $url;

  public function stringify() {
    return ‘<a href=“’.htmlspecialchars($this->url).‘”>’.
           htmlspecialchars($this->title) . ‘</a>’;
  }
}
```

```
<?php
include “/usr/share/hhvm/xhp/init.php”
Include “/var/www/xhp/ui.php”

$body = <body>
          <ui:menu title=“HPHP’s Corner”>
            <ui:menuitem title=“Option1”
                         url=“http://example.com”/>
          </ui:menu>
        </body>;

$page = <html>
           <head><title>MyApp</title></head>
        </html>;
$page->appendChild($body);

echo $page;
```

```
<html><head><title>MyApp</title>
</head><body><div onclick=“menuExpand(this)”>
<span class=“title”>HPHP&apos;s Corner</span>
<a href=“http://example.com”>Option1</a></div>
</body></html>
```

Hack

- Stronger typing
- Better native code generation
- Java style generics
- Intelligent Collections


```
<?hh

Class Hello {
  public string $name;

  function foo(string $bar, ?int $baz) : bool {
    // $bar is known to be a string
    // $baz is either an int or NULL
    // Caller knowns that foo() will return a bool
    if ($baz) {
      return $bar == “true”;
    } else {
      return strlen($bar) > baz;
    }
  }

  function max(Vector<int> $numbers) : int {
    $max = 0;
    $first = true;
    foreach($numbers as $number) {
      if ($first || $max < $number) {
        $max = $number;
      }
      $first = false;
    }
    return $max;
  }
}

class Proxy<T> {
  public function __construct(protected T $val) {}
  public getVal(): T { return $this->val; }
}
```

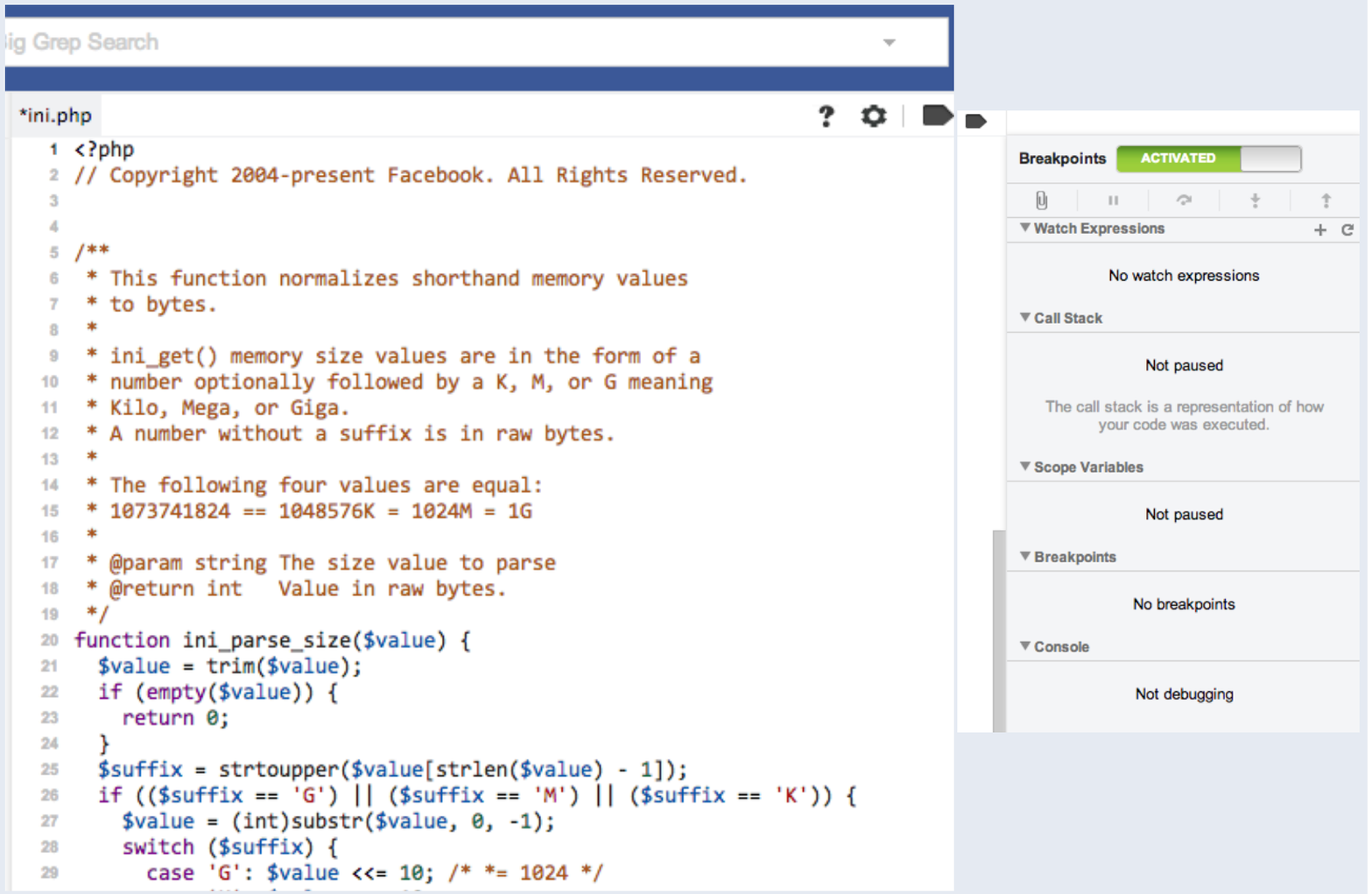
HPHPd – HPHP Debugger

- Interactive shell
- GDB-like debugging
- Standalone or w/ server
- Breakpoints
- Watches
- Macros

```
Welcome to HipHop Debugger!
Type "help" or "?" for a complete list of commands.

hphpd> =str_rot13("Hello World!");
Uryyb Jbeyq!
hphpd> $fp = fopen("php://stdout", "w");

hphpd> fwrite($fp, "Example\n");
Example

hphpd> $a = 123;

hphpd> $b = 456;

hphpd> =($a+$b);
579
hphpd>
```

FBIDE (Web)

- Integrated HPHPd
- Autocompletion
- Highlighting
- Inline git annotate
- Linting
- Phabricator integration
- Shared editing



Async Routines

- Run multiple paths in parallel
- Wait on network/db resources
- Non-preemptive multi-tasking

```
<?php

function get_profile(int $userid) {
  $h = mysql_query_async("SELECT …");
  await $h;
  return mysql_fetch_assoc($h);
}

function read_pipe(Socket $sock, int $len) {
  $ret = '';
  while (strlen($ret) < $len) {
    await $sock;
    $need = $len – strlen($ret);
    $ret .= fread($sock, $need);
  }
  return $ret;
}


list($profile, $data) =
  await(get_profile($userid),
        read_pipe($sock, 1024)
       );
```

Questions and Resources

Resources

- http://hhvm.com/repo
- http://hhvm.com/blog
- http://hhvm.com/fb
- http://hhvm.com/twitter

- src/doc in git repository for Options and Technical... Stuff