# PHP Performance Profiling

Jonathan Oxer

# PHP Performance Profiling

Do you know what your code is doing right now?

Really?

# PHP Performance Profiling

- Factors Affecting Performance
- Profiling tools
- Installing And Testing APD
- Gathering Data With APD
- Interpreting APD Data
- Profiling A Live Site
- Editor/Debugger Integration

# Factors Affecting Performance

- Web server configuration
- Database performance
- Data structure
- Application design
- Application implementation

# Profiling Tools

- Zend Studio

  www.zend.com

- DBG

  dd.cron.ru/dbg

- XDebug

  www.xdebug.org

- Advanced PHP Debugger

  pear.php.net/apd

# Getting APD

- Source
- PEAR
- Debian Package

# Checking Your Installation

- phpinfo()
- echo “<?php phpinfo(); ?>” > /var/www/info.php

# Your First Test

- Squirrelmail
- apd_set_pprof_trace();

# Gathering Some Data

- Data stored as tracefiles in apd.dumpdir.
- Tracefiles not intended to be human-readable.
- “pprofp <tracefile>” to process the tracefile.

# Interpreting The pprof Tracefile

- pprofp -u /var/log/php-apd/pprof.2523

# pprofp Sort Options

- -a: sort by alphabetic names of subroutines.
- -l: sort by number of calls to subroutines
- -m: sort by memory used in a function call.
- -r: sort by real time spent in subroutines.
- -R: sort by real time spent in subroutines (inclusive of child calls).
- -s: sort by system time spent in subroutines.
- -S: sort by system time spent in subroutines (inclusive of child calls).
- -u: sort by user time spent in subroutines.
- -U: sort by user time spent in subroutines (inclusive of child calls).
- -v: sort by average amount of time spent in subroutines.
- -z: sort by user+system time spent in subroutines. (default)

# pprofp Display Options

- -c: display real time elapsed alongside call tree.
- -i: suppress reporting for PHP built-in functions
- -O <cnt>: specify max number of subroutines to display. (default 15)
- -t: display compressed call tree.
- -T: display uncompressed call tree.

# Function Call Tree

- pprofp -t /var/log/php4-apd/pprof.235:

```
main
require_once
  php_self
require_once (2x)
  session_is_registered
  require_once
require_once
  require_once (3x)
    require_once (2x)
  require_once
    require_once
      require_once
        require_once
         is_array
          use_plugin
            file_exists
            include_once
            function_exists
            ... etc
```

# My Most Embarrassing PHP Moment

```
# pprofp -T /var/log/php-apd/pprof.2523 | wc -l
```

# My Most Embarrassing PHP Moment

```
# pprofp -T /var/log/php-apd/pprof.2523 | wc -l
1162
#
```

# My Most Embarrassing PHP Moment

```
# pprofp -T /var/log/php-apd/pprof.2523 | wc -l
1162
#
```

Oops.

# APD Function Reference: apd_set_pprof_trace()

The most useful APD function as far as profiling is concerned, this dumps a tracefile named pprof.<pid> in your apd.dumpdir. The tracefile is a machine-parsable output file that can be processed with the pprofp <tracefile> command.

# APD Function Reference: apd_set_session_trace()

similar to apd_set_pprof_trace(), it dumps a human-readable session trace named apd_dump_<pid> in your apd.dumpdir. This is the old way of doing things, noted here because it still works (for now). It's been deprecated, so it's better to use a pproftrace instead. N is an integer that sets the items to be traced; use a value of 99 to turn on all implemented options.

# APD Function Reference: array apd_callstack()

Returns the current call stack at that stage of execution as an array.

# APD Function Reference: apd_cluck([error],[delimiter])

Behaves like Perl's Carp::croak module.

Throws an error, a callstack and then exits.

The default line delimiter is '<BR />\n'. This function is deprecated for users of PHP4.3+; use the internal debug_backtrace() and debug_print_back-trace() instead.

# APD Function Reference: array apd_dump_regular_resources()

Returns all current regular resources as an array.

# APD Function Reference: array apd_dump_persistent_resources()

Returns all current persistent resources as an array.

# APD Function Reference: override_function(name, args, code)

Syntax is similar to create_function().

Overrides built-in functions by replacing them in the symbol table.

# APD Function Reference: rename_function(orig_name, new_name)

Renames orig_name to new_name in the global function table. Useful for temporarily overriding built-in functions.

# Profiling A Live Site

```
<?php
$DEBUGIPS = array('203.222.90.204','192.168.0.23');
if(array_search($_SERVER[REMOTE_IP],$DEBUGIPS))
        apd_set_pprof_trace();
?>
```

# Editor Integration

- Using APD is a pain: set trace, load page, find file, process with pprofp, read output, blah blah blah
- If it's a pain, you won't do it. At least not often.
- Andy Jeffriess, gPHPEdit author, looking at integrating APD calls directly into gPHPEdit. Others may follow.

# Questions

?