




效率与优化

PHPChina PCTP 课程

主讲人

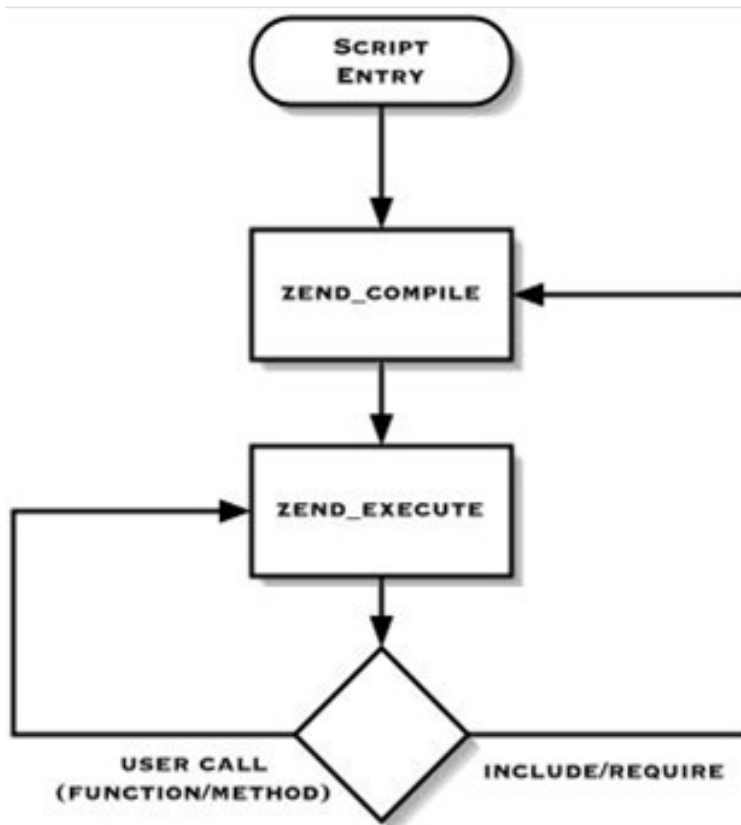
SOHU 互动产品技术主管



效率与优化

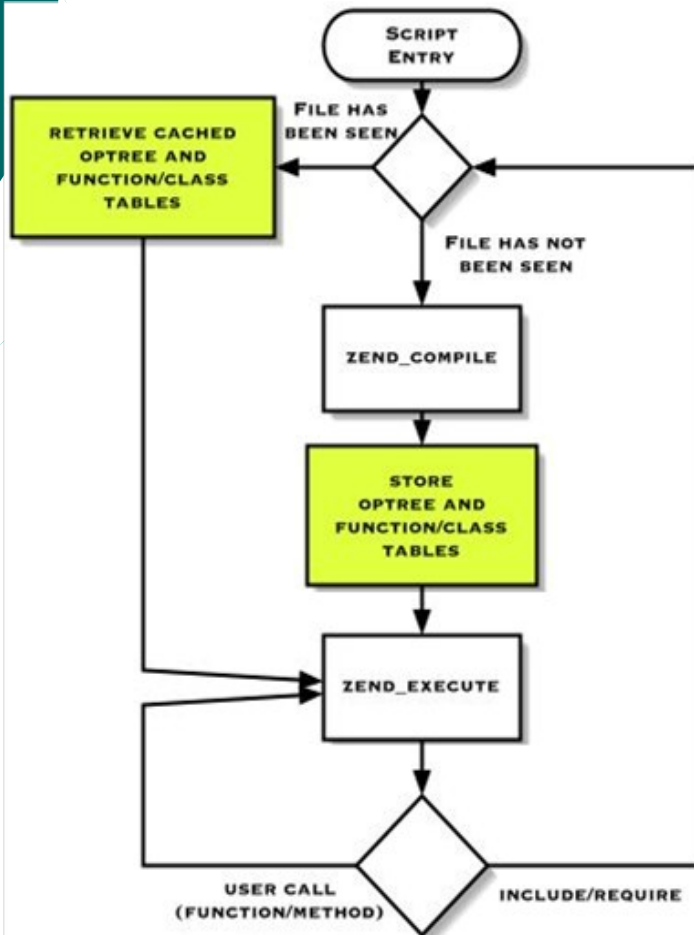
- 效率与优化的概述
- **PHP** 安装的优化
- **apache** 的优化
- **PHP** 开发的细节优化
- 系统的优化
- 数据库的优化
- **CACHE** 的优化

效率与优化 —— 运行的方式



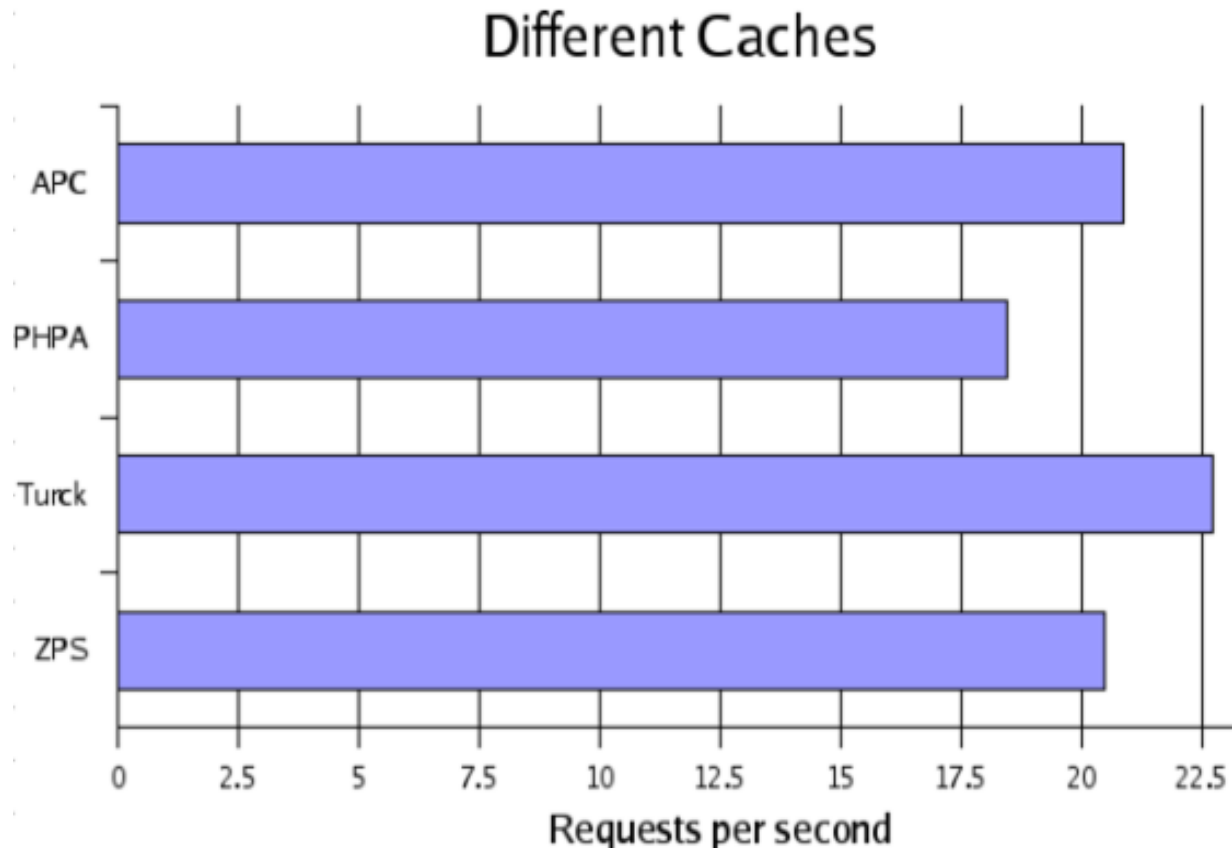
- This cycle happens for every include file, not just for the "main" script.
- Compilation can easily consume more time than execution.

编译和缓存的优化



- Each PHP script is compiled only once for each revision.
- Reduced File IO, opcodes are being read from memory instead of being parsed from disk.
- Opcodes can optimised for faster execution.

不同 CACHE 的效果



APC, PHP Accelerator, Turck MM Cache / eAccelerator,
Zend Performance Suite,

编译和优化

- 优化的最大话， 编译的时候最好和硬件相配合

编译的时候使用 `-O3` 这个参数

- 按照不同的 CPU 加参数 `-march -mcpu`
 - 用 CPU 特殊的特性 `-msse -mmmx -mfpmath=sse`
- **`export CFLAGS="-O3 -msse -mmmx -march=pentium3 \`**
 - **`-mcpu=pentium3 -funroll-loops -mfpmath=sse"`**

Apache/PHP 的联合编译

- 把 PHP 用静态方式编译到 APACHE 里面去
能够提高 30 % 的效率

如何编译

- **# PHP**
- **./configure --with-apache=/path/to/apache_source**
- **# Apache**
- **./configure --activate-module=src/modules/php4/libphp4.a**
- 在 4.3.11+ 或者更高的帮本中
- **--prefer-non-pic is default.**

APACHE 的优化

- **How to compile PHP statically into Apache?**
 - **# PHP**
 - **./configure --with-apache=/path/to/apache_source**
 - **# Apache**
 - **./configure --activate-module=src/modules/php4/libphp4.a**
 - Or use PHP 4.3.11+ where
 - **--prefer-non-pic** is default.

APACHE httpd.conf 配置优化

- 不要放太多的文件数在 `DirectoryIndex` 这个列表参数
- 如果不需要 **LOG** 文件，就把他禁止掉
- 如果确实需要，就只要一个文件
- `HostnameLookups`. 不要开启
- `ServerSignature` 关闭

Apache : KeepAlive 这个参数

- **KeepAlive** 这个设计的目地是为了能快点 查看页面

- 如果打开这个参数

KeepAliveTimeout 要设置尽可能的小
一般是设置成为 10

为啥不设置成为零？

推荐是关闭这个参数 设置为 **OFF**

其他的 WEB 服务器

动态的来说， APACHE 还是最快的

- lighttpd
- Boa
- Tux
- Thttpd

静态他们比 apache 要快 3 倍 — 4 倍

为什么 apache 是最流行的服务器？

内容的压缩

- 针对 **APACHE**

现在所有的浏览器都支持自动解压缩，所以可以把内容在服务器端压缩了再输出

- **Apache 1** (`mod_gzip`)

- **Apache 2** (`mod_deflate`)

- **PHP**

- From PHP configuration

- `zlib.output_compression=1`

- From inside the script

- `ob_start("ob_gzhandler")`

- 会消耗 3%-5% of CPU.



Tuning PHP Configuration

- `register_globals = Off **`
- `magic_quotes_gpc = Off`
- `expose_php = Off`
- `register_argc_argv = Off`
- `always_populate_raw_post_data = Off **`
- `session.use_trans_sid = Off **`
- `session.auto_start = Off **`
- `session.gc_divisor = 1000 or 10000`
- `output_buffering = 4096`

Ab14 -c 1000 -n 3000

- **Server Software:** Apache
- **Server Hostname:** localhost
- **Server Port:** 80
- **Document Path:** /php.php
- **Document Length:** 46844 bytes
- **Concurrency Level:** 10
- **Time taken for tests:** 0.265 seconds
- **Complete requests:** 100
- **Failed requests:** 0
- **Broken pipe errors:** 0
- **Total transferred:** 5077082 bytes
- **HTML transferred:** 5061168 bytes
- **Requests per second:** 377.36 [#/sec] (mean)
- **Time per request:** 26.50 [ms] (mean)
- **Time per request:** 2.65 [ms] (mean, across all concurrent requests)
- **Transfer rate:** 19158.80 [Kbytes/sec] received
- **Connection Times (ms)** min mean[+/-sd] median max
- **Connect:** 0 8 5.2 8 20
- **Processing:** 22 16 5.2 16 25
- **Waiting:** 3 14 5.5 14 24
- **Total:** 22 24 3.2 24 44

程序的优化

不好

```
<?php include "file.php"; ?>
```

绝对或者相对地址

```
<?php
```

```
include "/path/to/file.php";
```

```
// or
```

```
include "../file.php";
```

```
?>
```



正确的硬件驱动

- 提高整个速度
- 调整硬件的设置参数

内存硬盘

- 把文件和尽量的放在内存中间
- 再 **linux** 中间使用 `tmpfs`.

```
# Speed Up /tmp Directory
```

```
mount --bind -ttmpfs /tmp /tmp
```

```
# Accelerate Scripts Directory
```

```
mount --bind -ttmpfs /home/webroot /home/webroot
```

尽量利用现成的函数

```
$data = '' ;  
$fp = fopen("some_file", "r") ;  
while ($fp && !feof($fp)) {  
    $data .= fread($fp, 1024) ;  
}  
fclose($fp) ;  
  
// vs the much simpler & faster  
$data = file_get_contents("some_file") ;
```

了解一些原理能使工作更快

```
$a = "abc";
```

```
// 比较慢一点的
```

```
function a($str) { return $str . "d"; }
```

```
$a = a($a);
```

```
// 比较高效的
```

```
function b(&$str) { $str .= "d"; }
```

```
b($a);
```



说到这里？ 系统的级别优化还有很多

- Mount nodirtime noatime
- Ulimit -c 65536
- 可以使用 ICC 代替 GCC

.....

这些是不是我们优化的主要方向？



数据库的优化

- **MYSQL** 的编译优化
- 数据库 **SQL** 优化
- 数据库设计优化
- 数据库索引优化
- 数据库架构优化

MYSQL 的编译优化

- 尽量使用 MYSQL 4.0 以上的，但不推荐 5.0 my.cnf 的配置
- skip-locking
- skip-innodb
- set-variable = back_log=400 ?
- set-variable = key_buffer=128M
- set-variable = max_allowed_packet=1M
- set-variable = table_cache=1024 ?
- set-variable = sort_buffer=2M ?
- set-variable = record_buffer=2M ?
- set-variable = thread_cache=16
- set-variable = long_query_time=10
- set-variable = wait_timeout=15
- query_cache_size = 8M ?
- query_cache_type = 1
- query_cache_limit = 2M
- set-variable = thread_concurrency=8
- set-variable = myisam_sort_buffer_size=2M
- set-variable = max_connections=400
- set-variable = max_connect_errors=100000

先检查你的 SQL

SLOW

```
EXPLAIN select * from users where login LIKE '%ilia%';
```

table	type	possible_keys	key	key_len	ref	rows	Extra
mm_users	ALL	NULL	NULL	NULL	NULL	27506	where used

quick

```
EXPLAIN select * from users where login LIKE 'ilia%';
```

table	type	possible_keys	key	key_len	ref	rows	Extra
mm_users	range	login	login	50	NULL	2	where used

数据库设计和索引优化

- 尽量的使用单表， 少用复合查询
- 对最多的 **where** ， 进行特殊的优化
符合索引的使用， 索引采用左对齐的方式
如 (id, sid) 这样的两个

当

where id='1' and sid ='2'

where id = '1'

where sid = '2'

当访问量非常大的时候

- 一般来说， 一个再强的机器和再强的数据库

每秒种他处理的请求数

也不会超过 **1000**

分成不同机器来进行分担

MYSQL 数据库的主从表

ORACLE 的集群

但数据库优化是有限的

- 当你的访问量超过 2000 万的时候， 需要考虑自己写 **CACHE** 了
- 这种 **CACHE** 不是简单意义上的文件缓存
- 他需要专门的内存进行 **CACHE**
- 需要针对不同的情况进行特殊的设计

CACHE 设计的重点

数据结构： 散列表和数组

- 单纯的散列表例子， 有很好的成型的产品

MEM CACHED BARKELY DB

如： 我们的新闻

数组

常见的如用户表

算法

- 二分法查找
- 如查 **IP** 地址对应的地域
- 二叉树的动态排序
- 快排

这些速度都远远优于 数据库的 **WHERE**

Cache 的设计问题？

- CACHE 的稳定！
- CACHE 的同步
- CACHE 的设置和监控

都是需要考虑， 如果太麻烦， 会触动另
外一个效率
开发的效率



开发的效率

- 良好的文档的管理
- 请写的代码大部分人能看懂， 不要高效到什么人都看不懂
- OO 的编程方式
- 良好的架构设计
- 良好的可维护性



总结

开发是一种艺术，希望每个工程师都能理解开发的特点，不要做的特别过，效率，开发的可维护，都是适可，做到能够自己能理解，能控制，就是好程序，好产品