

高性能LAMP程序设计

超群.com

@fuchaoqun

<http://www.fuchaoqun.com>

PHP篇

Performance...

- ◆ 不要用 `array_key_exists`，用 `isset` 来判断键值是否在数组中
- ◆ 如有可能，采用 `static` 静态方法
- ◆ 避免使用 `__set`, `__get` 等魔术方法
- ◆ 使用 `echo` 代替 `print()`
- ◆ 使用 `include`、`require` 代替 `include_once`、`require_once`
- ◆ `@` 操作符是邪恶的
- ◆ 不要把 `count/strlen/sizeof` 放到 `for` 循环的条件语句中
- ◆

不好意思，今天不讲这些...

Why?

<http://www.garfieldtech.com/blog/magic-benchmarks>

循环200W次

原生获取：0.31μs/每次

__get：1μs/每次

原生设置：0.38μs/每次

__set：1.3μs/每次

```
<?php
class TestGetSet {
    public $foo = 1;
    public function __get($var) {
        return $this->foo;
    }
    public function __set($var, $val) {
        $this->foo = $val;
    }
}

$t = new TestGetSet();

$t->foo;
$t->bar;
$t->foo = 1;
$t->bar = 1;

?>
```

Results:

Get Native Property	0.619
Get Magic Property (__get())	2.066
Set Native Property	0.752
Set Magic Property (__set())	2.623

说实在的，我不care这些....

But...

- ◆代码洁癖，程序中最好不要有错误，哪怕是notice
- ◆干净的代码，非必要不引入
- ◆SQL语句不要放在for循环里面执行，最好能用group by之类解决，或者合并写入
- ◆出了问题再profile你的PHP代码
- ◆通过auto loading 实现 lazy loading
- ◆相比较运行速度，更需要注意memory limit，尤其是一些shell处理脚本

线上PHP监控

你的线上PHP代码运行正常吗？

偶发数据库连接失败、边界溢出、后台服务抖动、合作方数据异常.....

解决办法：

通过set_error_handler来捕获线上运行错误，统一收集日志、报警

通过register_shutdown_function来捕获fatal errors、记录运行时间

```
<?php
set_error_handler('handle_normal_error', E_ALL | E_STRICT);
function handle_normal_error($errno, $errstr, $errfile, $errline)
{
    switch ($errno) {
        case E_USER_ERROR:
            send_error_to_log_server($msg);
            break;
        case E_USER_WARNING:
            // do something
            break;
        default:
            // do something
            break;
    }
    return true;
}

register_shutdown_function('handle_fatal_error');
function handle_fatal_error()
{
    $error = error_get_last();
    if (isset($error['type']) && E_ERROR == $error['type']) {
        send_error_to_log_server($msg);
    }
}
```


Profiling

◆PHP工具:

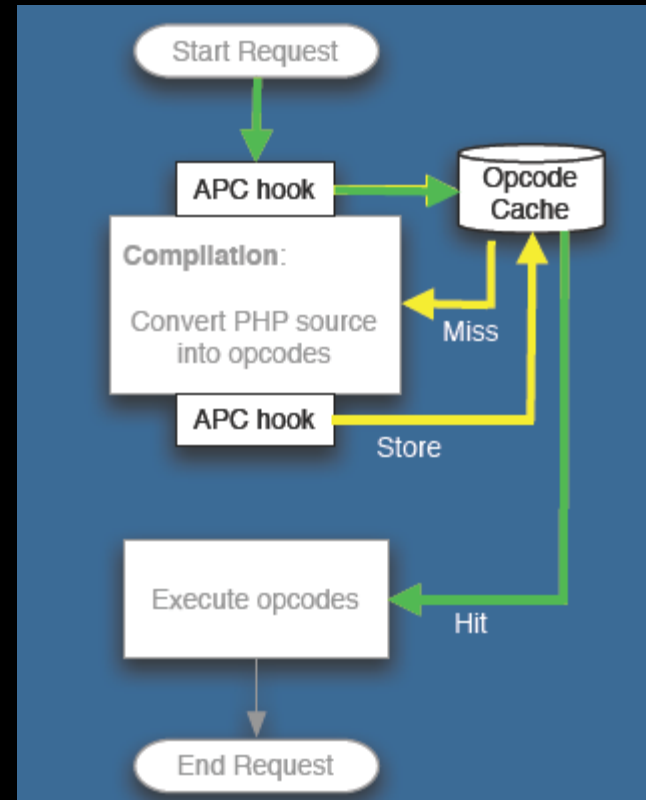
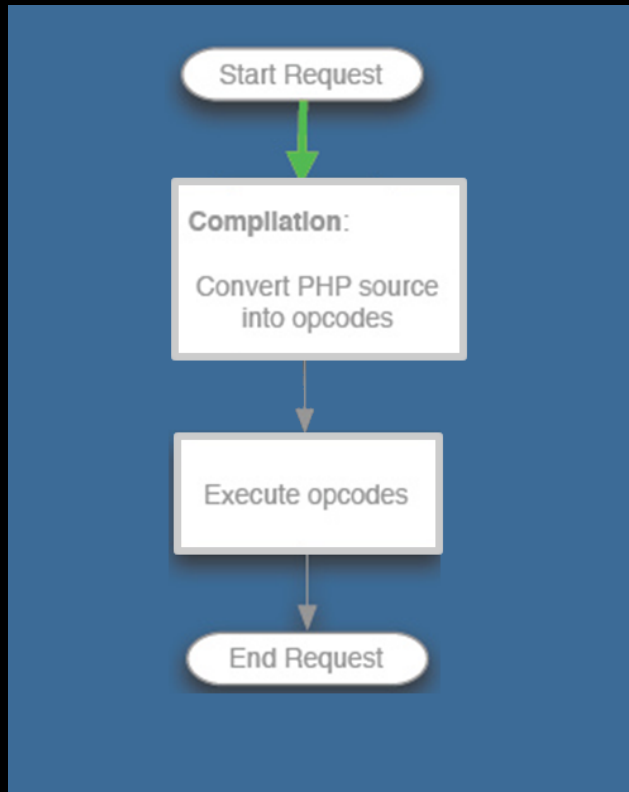
Xdebug、xhprof, 或者

```
1  <?php
2  $start = microtime(true);
3
4  // do some thing here
5
6  $cost = microtime(true) - $start;
```

◆整体性能工具

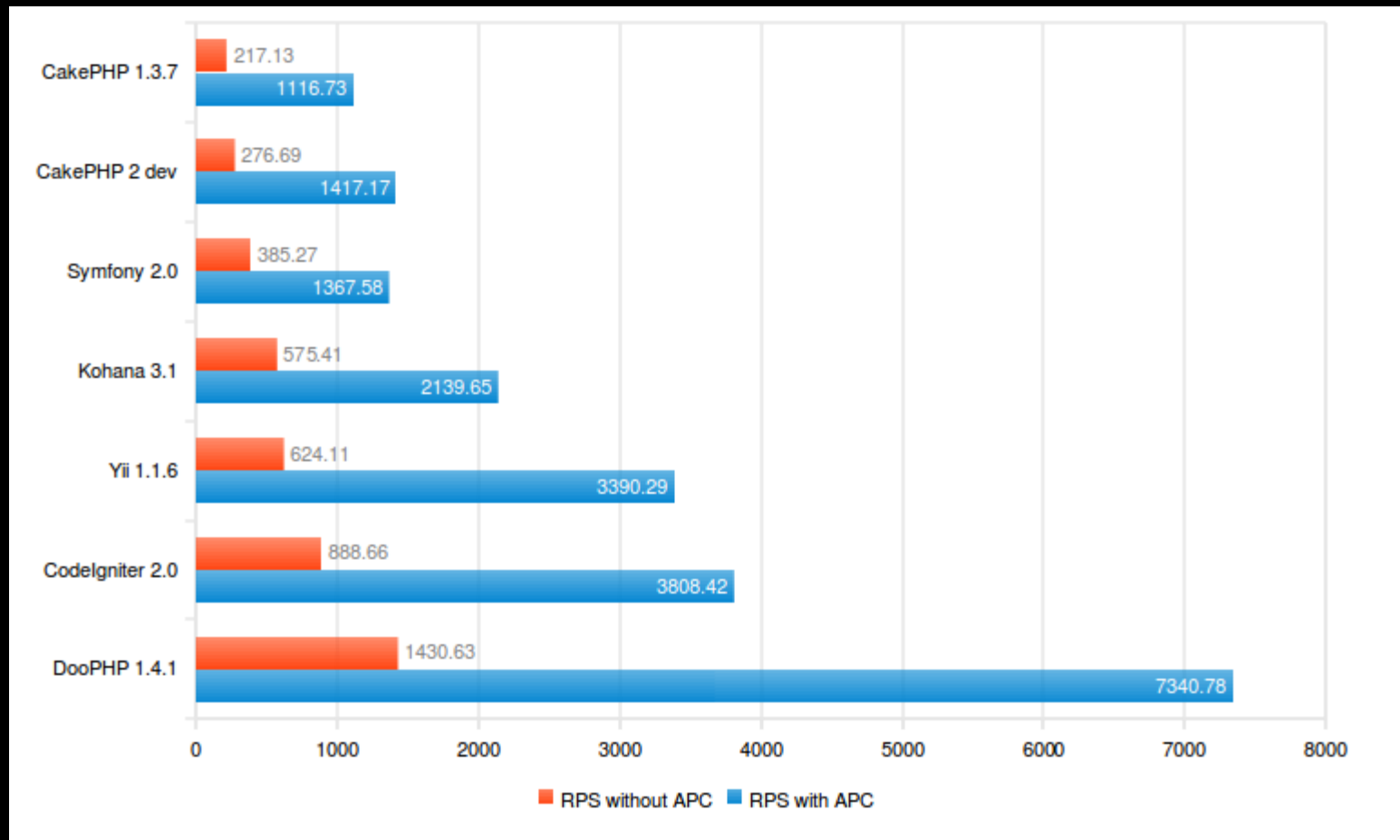
```
ab -n 100000 -c 200 -k http://127.0.0.1/test.php
```

OpCode



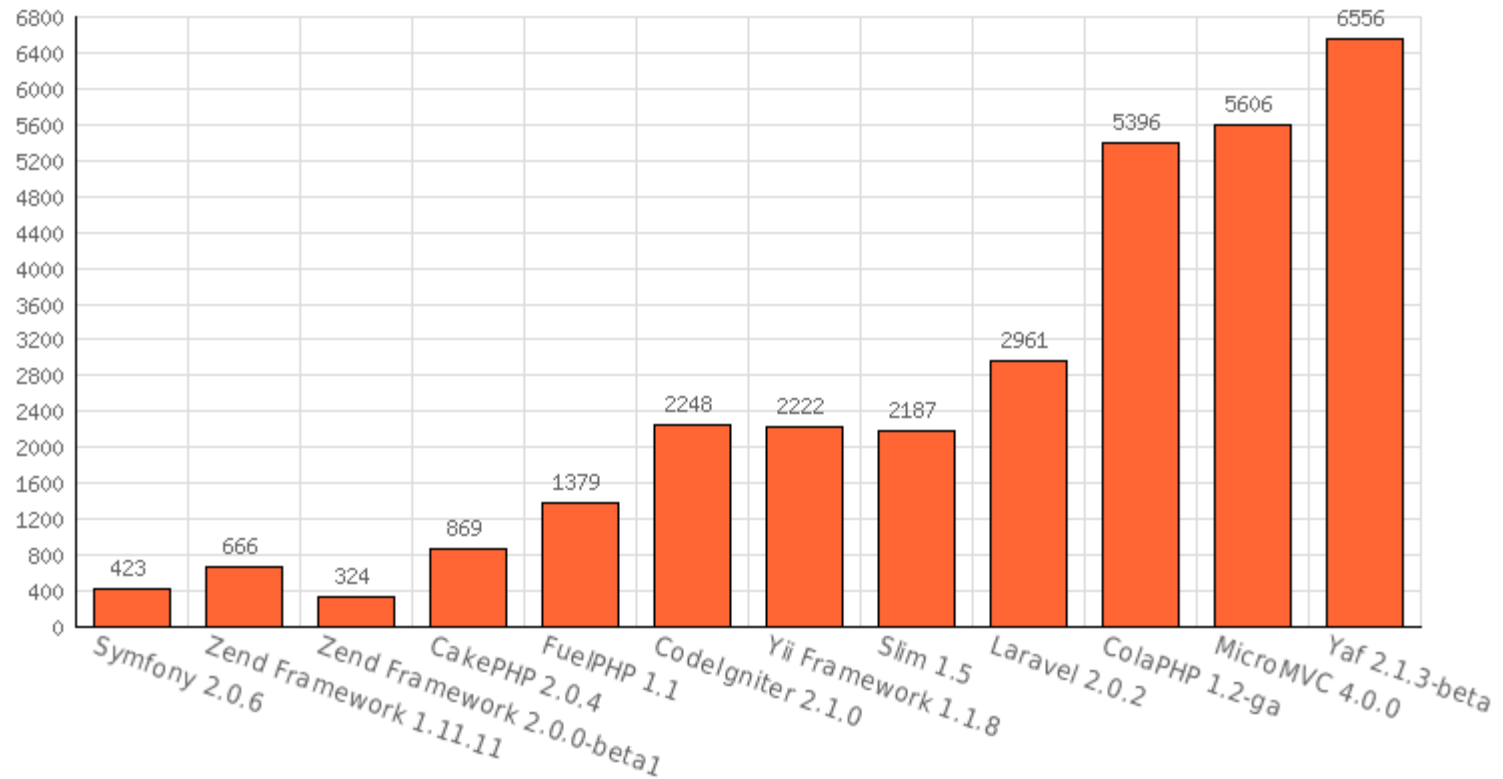
Try: APC、eAccelerator....

PHP Framework

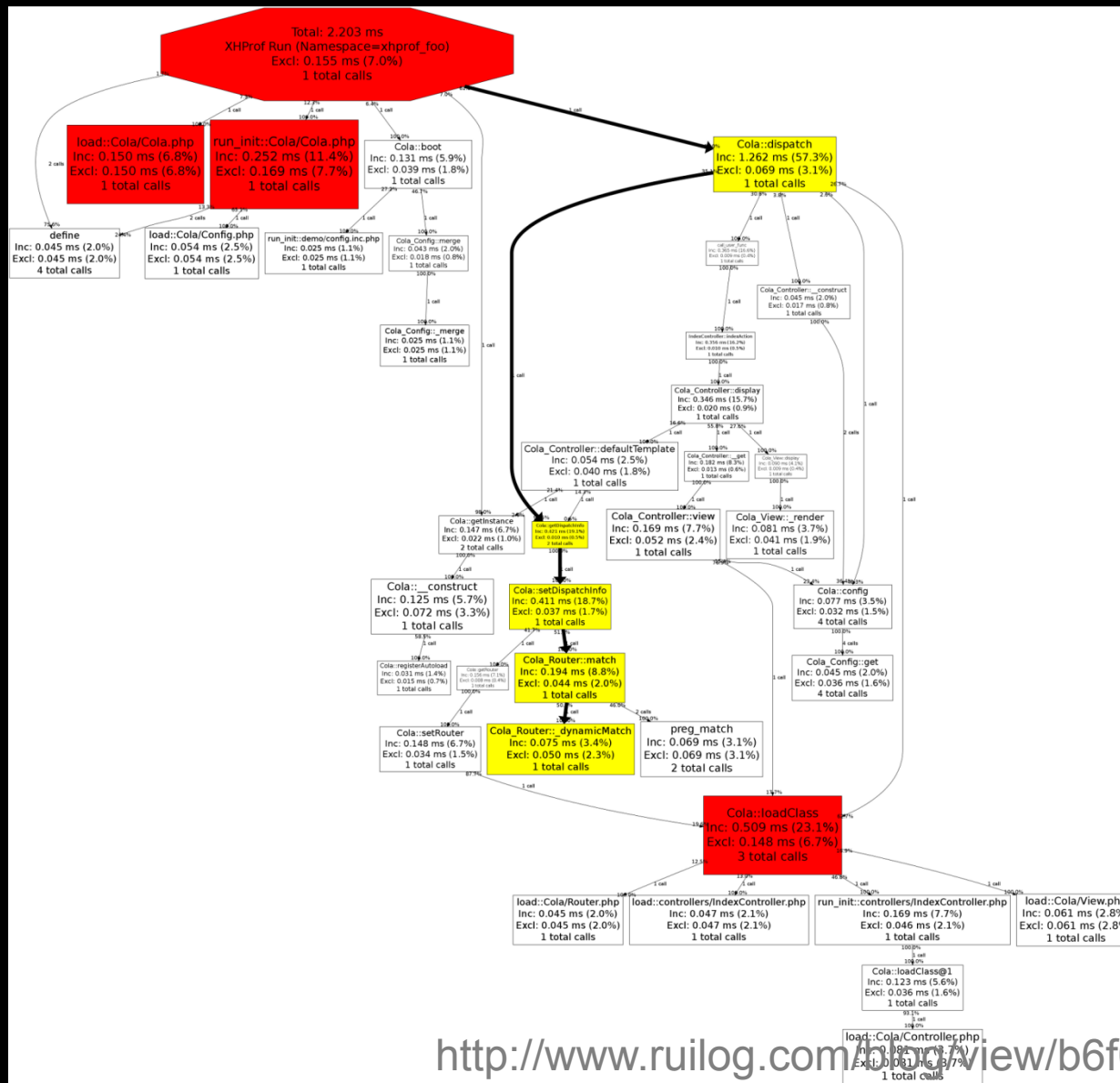


PHP Tunning Case: ColaPHP

ApacheBench (ab -c 200 -n 50000)



PHP Tunning Case: ColaPHP



D:\www\ColaPHP\demo\index.php

cachegrind.out.4308 @ 2012-03-05 07:01:49

Filter: (regex too)

Function

{main}

Called from script host

Calls	Count	Total Call Cost	
▶ Cola->dispatch @ 14	1	2362	
▶ require::D:\www\ColaPHP\Cola\Cola.php @ 7	1	403	
▶ Cola::boot @ 14	1	257	
▶ Cola::getInstance @ 9	1	138	
php::date_default_timezone_set @ 5	1	16	
php::error_reporting @ 2	1	5	
php::ini_set @ 3	1	4	

Cola->dispatch

Calls	Count	Total Call Cost	
▶ Cola->getDispatchInfo @ 318	1	951	
▶ Cola::loadClass @ 332	1	833	
▶ Cola_Controller->__construct @ 335	1	239	
php::call_user_func @ 343	1	54	
▶ Cola_Config->get @ 332	1	20	
php::extract @ 322	1	4	
php::is_callable @ 340	1	2	

Called From	Count	Total Call Cost	
▶ {main} @ 14	1	2362	

Invoca

PHP Tunning Case: ColaPHP

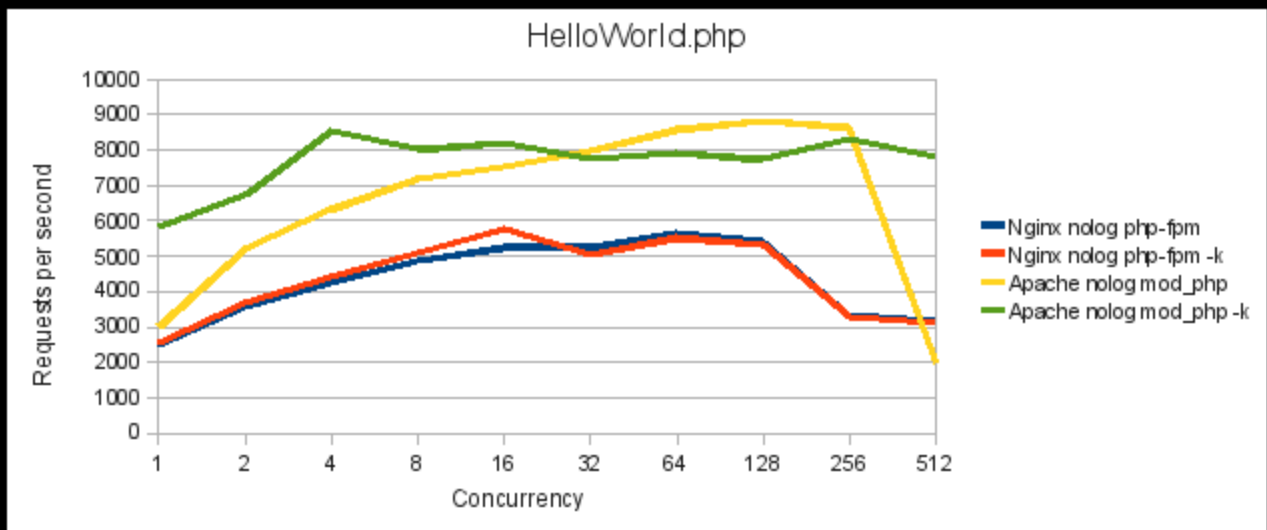
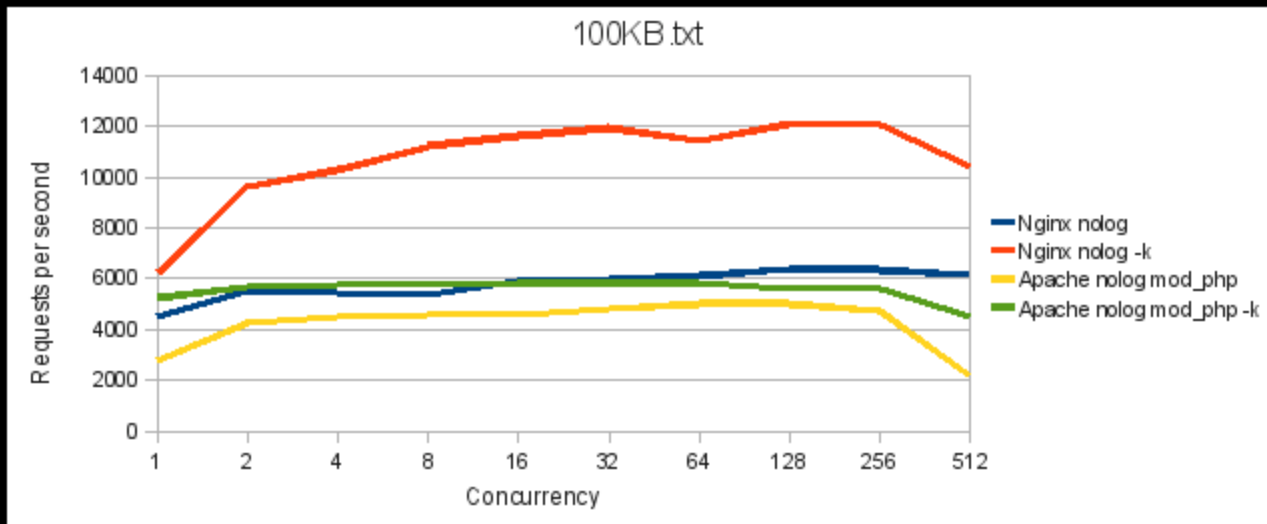
ColaPHP完成一个完整的调度（Frontcontroller、Router、Dispatcher、Controller、Responder），

消耗**0.5ms**

- ◆使用Xdebug跟踪代码运行效率，使用[webgrind](#)展示
- ◆大部分的消耗在文件引用上(include/require)，虽然用了opcode加速，>0.3ms
- ◆大量使用__set、__get等魔术方法来实现对象的按需生成
- ◆通过spl_autoload_register实现类的lazy loading，大大提高框架速度
- ◆“短”代码
- ◆框架流程可随时被终止，只需为用到的特性买单

Webserver篇

Apache VS Nginx



Apache Tips

- ◆ <http://httpd.apache.org/docs/2.4/misc/perf-tuning.html>
- ◆ 只加载用到的模块
- ◆ mpm选择(event/worker/prefork)
- ◆ AllowOverrides
- ◆ Google mod_pagespeed
- ◆ EnableSendfile
- ◆ keepalive

Nginx Tips

◆Epoll

◆worker_processes: CPU数目倍数, 动态应用的话1倍就好

◆ulimit -SHn 65535

◆CPU亲和性

```
worker_processes 4;
```

```
worker_cpu_affinity 0001 0010 0100 1000;
```

◆worker_connections 65535;

◆keepalive_timeout 60;

◆sendfile on;

◆tcp_nodelay on;

◆tcp_nopush on;

Nginx 502



<http://blog.s135.com/post/361/>

Nginx 502

◆fast_cgi设置

```
fastcgi_connect_timeout 30;  
fastcgi_send_timeout 30;  
fastcgi_read_timeout 30;  
fastcgi_buffer_size 64k;  
fastcgi_buffers 4 64k;  
fastcgi_busy_buffers_size 128k;  
fastcgi_temp_file_write_size 128k;
```

◆Php-fpm设置

```
<value name="listen_address"> /dev/shm/php-fpm.sock </value>  
<value name="max_children">128</value>  
<value name="request_terminate_timeout">10s</value>  
<value name="request_slowlog_timeout">5s</value>  
<value name="slowlog">/path/to/slow.log</value>  
<value name="rlimit_files">65535</value>
```

数据库篇

MySQL

◆Linux server & MySQL server tuning

◆短、小

◆动静分离

◆分库分表

◆良好的索引 & Explain

◆主从同步，通过从库来扩展读

◆尽量Cache，减少SQL操作

◆批量操作 & 队列

◆讹传比性能更可怕，比如：left join都很慢、like用不到索引、char一定比varchar好....

Cache

◆ Browser Cache

Last modify、 Etag、 Expires

◆ Page Cache

Squid、 Varnish、 Nginx proxy_cache、 Nginx fast_cgi_cache

◆ Data Cache

Memcached、 Redis

<http://tech.idv2.com/2008/08/17/memcached-pdf/>

Nginx fast_cgi_cache

```
fastcgi_temp_path /data/nginx_fcgi_tmp;
```

```
fastcgi_cache_path /data/nginx_fcgi_cache levels=1:2
```

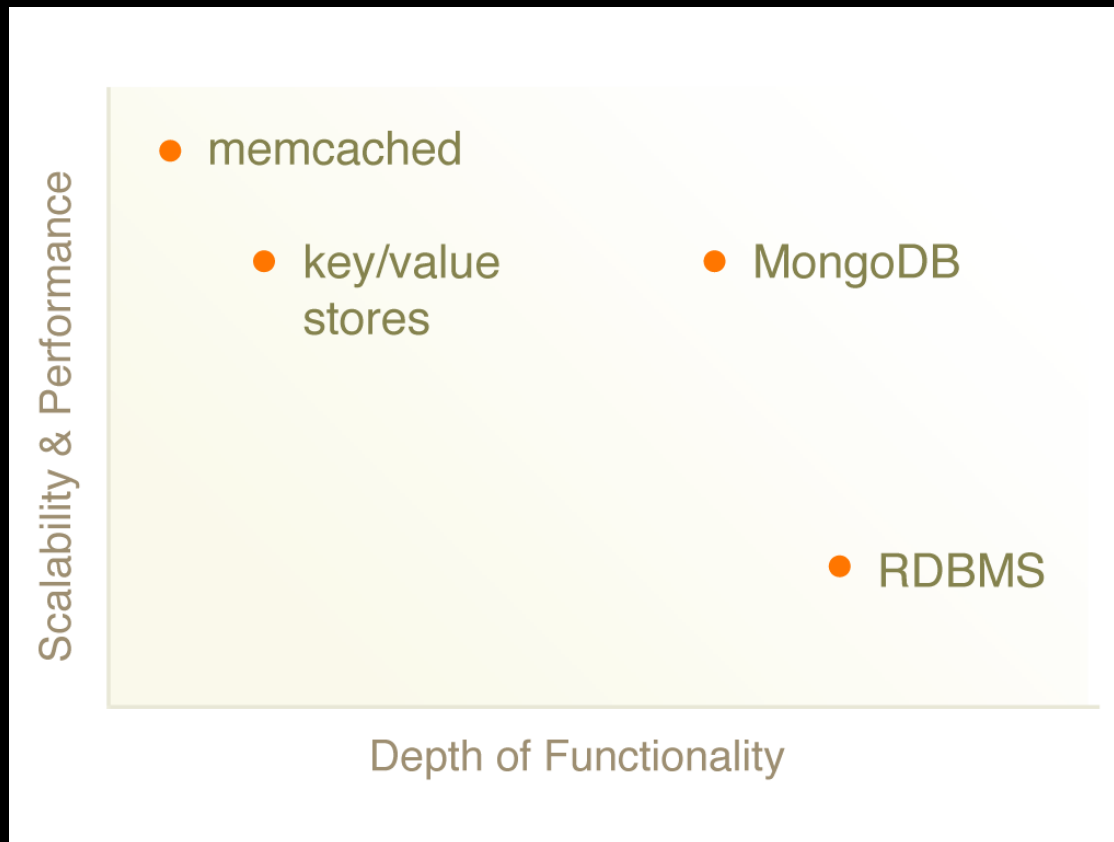
```
keys_zone=nginx_fcgi_cache:512m inactive=1d max_size=40g;
```

```
fastcgi_cache_valid 200 301 302 1d;
```

```
fastcgi_cache_use_stale error timeout invalid_header http_500;
```

```
fastcgi_cache_key $request_method ://$host$request_uri;
```

NoSQL: MongoDB



mySQL

SELECT

```
Dim1, Dim2,
SUM(Measure1) AS MSum,
COUNT(*) AS RecordCount,
AVG(Measure2) AS MAvg,
MIN(Measure1) AS MMin
MAX(CASE
  WHEN Measure2 < 100
  THEN Measure2
END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
AND (Filter2 = 'C')
AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
```

- ① Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
- ② Measures must be manually aggregated.
- ③ Aggregates depending on record counts must wait until finalization.
- ④ Measures can use procedural logic.
- ⑤ Filters have an ORM/ActiveRecord-looking style.
- ⑥ Aggregate filtering must be applied to the result set, not in the map/reduce.
- ⑦ Ascending: 1; Descending: -1

MongoDB

```
db.runCommand({
  mapreduce: "DenormAggCollection",
  query: {
    filter1: { '$in': [ 'A', 'B' ] },
    filter2: 'C',
    filter3: { '$gt': 123 }
  },
  map: function() { emit(
    { d1: this.Dim1, d2: this.Dim2 },
    { msum: this.measure1, recs: 1, mmin: this.measure1,
      mmax: this.measure2 < 100 ? this.measure2 : 0 }
  );},
  reduce: function(key, vals) {
    var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
    for(var i = 0; i < vals.length; i++) {
      ret.msum += vals[i].msum;
      ret.recs += vals[i].recs;
      if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
      if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
        ret.mmax = vals[i].mmax;
    }
    return ret;
  },
  finalize: function(key, val) {
    val.mavg = val.msum / val.recs;
    return val;
  },
  out: 'result1',
  verbose: true
});
db.result1.
  find({ mmin: { '$gt': 0 } }).
  sort({ recs: -1 }).
  skip(8).
  limit(4);
```

Redis

◆智能选股 <http://finance.qq.com/data/#znxg,hyg>

http://smartstock.gtimg.cn/get.php?func=filter&hs_hsl=0.05&hs_zf=0.03,0.05&hs_lb=1

◆1300+ RPS

◆MySQL数据持久化, Redis缓存（对比Memcached 30%+提升）

◆优势：丰富的数据结构

智能选股-优选模式

活跃股 资金股 研报股 敢死队股 指标股

活跃股 股性相对活跃意味着有资金积极参与推动，据统计：在同样的市场中，股性活跃的品种往往可以为投资者带来更多更丰厚的收益。

1. 当日换手率大于 5%

2. 当日涨幅超过 3%

开始选股

设置默认选股条件

选股结果：共选出股票 21 只 数据源日期：2012-02-28 11:14:50

股票名称	选股成立条件	股票名称	选股成立条件
------	--------	------	--------

常用组件篇

消息队列

◆用处：

异步处理耗时操作，比如发邮件、发微博等

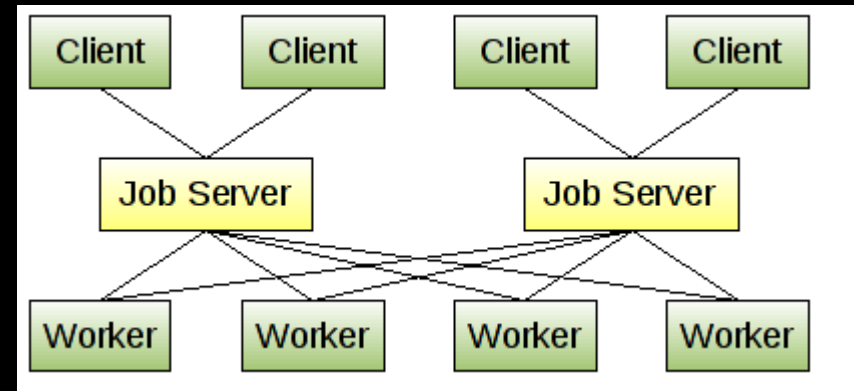
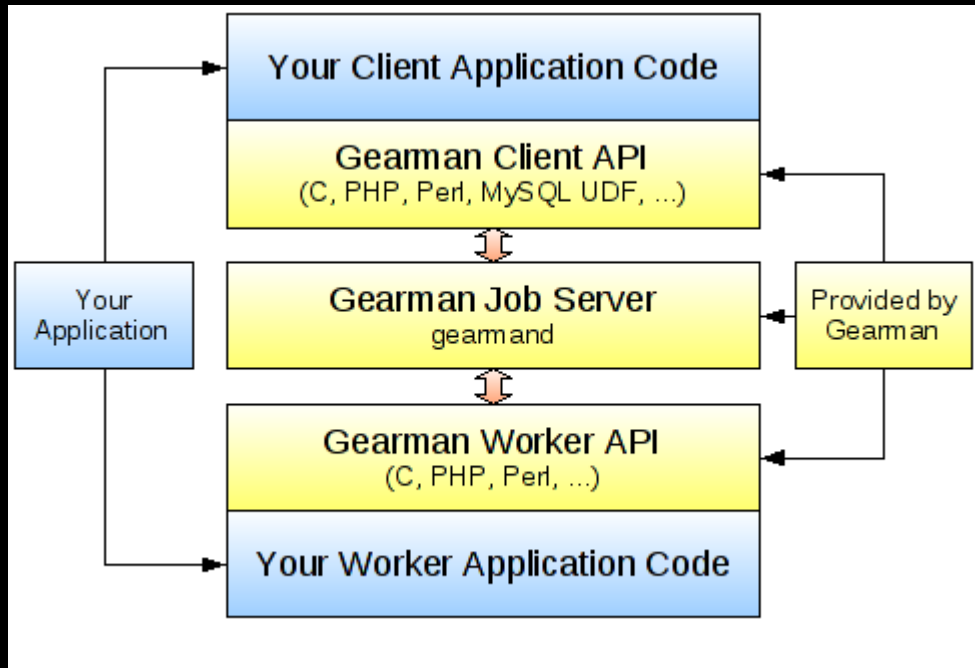
◆开源项目：

memcached： <http://memcached.org/memcached/>

beanstalkd： <http://kr.github.com/beanstalkd/>

RabbitMQ： <http://www.rabbitmq.com/>

分布式任务处理：Gearman



全文检索

◆开源项目

sphinx、xapian、lucence

◆中文分词：

scws： <http://www.ftphp.com/scws/>

◆集成方案：

coreseek： <http://www.coreseek.com/>

架构篇

几个原则

- ◆抗住，然后再优化
- ◆过渡设计比不设计更龌龊
- ◆越简单越好
- ◆如非必要，不要引入
- ◆层次清晰
- ◆可随时替换
- ◆可水平扩展
- ◆良好的监控预警

动态应用通用结构



自选股

- ◆2300w+注册用户，400w左右活跃用户
- ◆每天1.8亿次动态请求
- ◆4个IDC，20台前端服务器(nginx+php-fpm)
- ◆MySQL持久化数据(主从，跨IDC专线同步)
- ◆Memcached缓存
- ◆通过队列同步数据

接口平台

- ◆前端服务器Nginx+PHP-FPM
- ◆PHP框架基于ColaPHP定制修改
- ◆规范开发，快速开发，快速部署
- ◆监控和日志上报
- ◆缺点：隔离性存在一些风险

接口平台

```
<?php
/**
 * 演示Demo
 *
 * @author chaoqunfu
 * @version 1.0
 */
class DemoModel extends BaseModel
{
    protected $_db = 'DemoDb';
    protected $_ttl = 86400;

    /**
     * 获取用户信息
     * @param int $userId
     * @return array
     */
    public function getUserInfo($userId)
    {
        if (!$this->_checkUserId($userId)) {
            return array('code' => -1, 'msg' => '用户ID有误');
        }

        $sql = "select * from user where id = {$userId}";
        $res = $this->sql($sql);
        return $this->_format($res);
    }
}
```

```
<?php
/**
 * 演示Controller
 *
 * @author chaoqunfu
 * @version 1.0
 */
class DemoController extends BaseController {
    /**
     * 获取用户信息
     *
     * @return array
     */
    public function klineAction()
    {
        $id = $this->get('id');
        return $this->model->getUserInfo($id);
        //return $this->model->cached('getUserInfo', array($id));
    }
}
```

前端篇

PHP is rarely the bottleneck, 80-90% front-end.



--Rasmus Lerdorf

无连接

- ◆Browser Cache (Expires)

- ◆CSS Sprite

- ◆Lazy loading

 - 图片、头像等

- ◆合并请求

- ◆避免重定向

没流量

- ◆ JS、CSS压缩
- ◆ Gzip
- ◆ 图片压缩
- ◆ Browser Cache (Last modify、Etag)
- ◆ Lazy loading
- ◆ 使用ajax减少流量
- ◆ `` is evil

其他

- ◆CDN

- ◆静态文件使用无cookie域名

- ◆CSS放顶部，JS放底部（通常情况下）

工具

◆Page Speed

◆Yslow

◆使用Javascript记录页面渲染时间

Thanks & QA

百度技术沙龙

畅想

交流

争鸣

聚会

关注我们：t.baidu-tech.com

资料下载和详细介绍：infoq.com/cn/zones/baidu-salon

“畅想•交流•争鸣•聚会”是百度技术沙龙的宗旨。百度技术沙龙是由百度与InfoQ中文站定期组织的线下技术交流活动。目的是让中高端技术人员有一个相对自由的思想交流和交友沟通的平台。主要分讲师分享和OpenSpace两个关键环节，每期只关注一个焦点话题。

讲师分享和现场Q&A让大家了解百度和其他知名网站技术支持的先进实践经验，OpenSpace环节是百度技术沙龙主题的升华和展开，提供一个自由交流的平台。针对当期主题，参与者人人都可以发起话题，展开讨论。

InfoQ 策划·组织·实施

关注我们：weibo.com/infoqchina