

PHPCon上海

2016

PHP7+Swoole开发超高性能后台程序

@hantianfeng Rango-韩天峰 / 车轮互联



**PHPCON**  
[www.phpconchina.com](http://www.phpconchina.com)

# 关于我

- 车轮互联总架构师
- Swoole开源项目创始人
- PHP官方PECL扩展开发组成员
- 微博：@hantianfeng
- Github: <https://github.com/matyhtf>



# 超高性能?

1. C10K , C100K , C1M

2. 1W+ QPS

# 大量并发连接（C10k）

1. 基于epoll实现异步IO处理
2. 选择nginx和swoole
3. 并发100万TCP连接不是难事

# 1W+ QPS

1. 单台服务器每日可处理超过10亿次动态请求
2. 类似淘宝秒杀系统、微信抢红包、刷朋友圈这样全民级应用。如果PHP程序能达到1W+ QPS , 只用几百台机器就能支撑。



# 如何实现1W+ QPS

1. IO操作要足够快 或者 异步

常见的IO操作：Redis、MySQL、CURL、磁盘读写

2. CPU消耗足够少

应用服务器、PHP框架、PHP应用程序

# 同步阻塞模型

IO类型	耗时	单进程	100 进程	100 进程 10次操作
Redis	1ms	1000qps	100000qps	10000qps
MySQL	5ms	200qps	20000qps	2000qps
CURL	100ms	10qps	1000qps	100qps
磁盘IO	10ms	100qps	10000qps	1000qps



# 同步阻塞模型

1. 多开进程就能增加处理能力
2. 增加进程会带来额外的进程切换开销

```
➔ ~ vmstat 1 1000
```

procs		-----memory-----				---swap--		-----io----		-system--		-----cpu-----					
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
1	0	0	5802212	427476	1052420	0	0	0	0	3	13	11	0	0	100	0	0
0	0	0	5802088	427484	1052420	0	0	0	64	215	505	0	0	99	1	0	
0	0	0	5802088	427484	1052420	0	0	0	0	197	462	0	0	100	0	0	
0	0	0	5802088	427484	1052420	0	0	0	0	200	472	0	0	100	0	0	
0	0	0	5802088	427484	1052420	0	0	0	0	218	522	0	0	100	0	0	
0	0	0	5802088	427484	1052420	0	0	0	0	205	473	0	0	100	0	0	
0	0	0	5802088	427484	1052420	0	0	0	0	203	485	0	0	100	0	0	
0	0	0	5802088	427484	1052420	0	0	0	0	217	511	0	0	100	0	0	

# CPU消耗

CPU消耗	耗时	单核	24核
kernel	-	-	-
php-fpm	2ms	500qps	12000qps
PHP框架	10ms	100qps	2400qps
PHP程序	20ms	50qps	1200qps
+	30ms	33qps	800qps

# C++性能更好怎么不用C++

1. 程序员学习时间+3年
2. 开发调试软件时间 x 3
3. QPS提升2倍

机器的时间可以用钱购买，人的时间无法用钱买



01

提升PHP程序性能



# 使用PHP7提升性能

CPU消耗	耗时	单核	24核
PHP框架	5ms	200qps	4800qps
PHP程序	10ms	100qps	2400qps
+	15ms	67qps	1600qps

· 程序员不需要做任何事情，升级PHP7就立即提升1倍性能

# 使用Yaf/Phalcon提升性能

CPU消耗	耗时	单核	24核
C扩展框架	1ms	1000qps	24000qps

# 使用Swoole应用服务器

1. 使用Swoole\Http\Server取代php-fpm将部分PHP对象常驻内存，减少传统LAMP架构每次请求创建销毁对象的开销
2. Swoole\Http\Server使用纯C编写，网络通信引擎和协议解析性能非常强悍。
3. Swoole最新版本可稳定运行在PHP7环境

# PHP7+Swoole压力测试

软件	QPS
Golang	166838.68
PHP7+Swoole	287104.12
Nginx-1.9.9	245058.70

```
ab -c 100 -n 1000000 -k http://127.0.0.1:8080/
```



# Zend引擎

```
1 k?php
2 $$ = microtime(true);
3 $array = array (
4   'key_0_83581' => 'value_0_21758',
5   'key_1_93870' => 'value_1_38248',
6   'key_2_17869' => 'value_2_66815',
7   'key_3_27630' => 'value_3_12406',
8   'key_4_70487' => 'value_4_32428',
9   'key_5_32035' => 'value_5_60011',
10  'key_6_96156' => 'value_6_69611',
11  'key_7_53421' => 'value_7_96556',
12  'key_8_35631' => 'value_8_88495',
13  'key_9_88685' => 'value_9_34737',
14  'key_10_44641' => 'value_10_40049',
15  'key_11_92306' => 'value_11_64742',
16  'key_12_57544' => 'value_12_25769',
17  'key_13_61792' => 'value_13_65184',
18  'key_14_90220' => 'value_14_11011',
19  'key_15_75112' => 'value_15_79946',
20  'key_16_68386' => 'value_16_86937',
21  'key_17_55140' => 'value_17_15894',
22  'key_18_34788' => 'value_18_65325',
```

```
htf@htf-All-Series:~/debug$ php a__.php
use 11.986017227173ms
htf@htf-All-Series:~/debug$ php a__.php
use 11.970043182373ms
htf@htf-All-Series:~/debug$ php a__.php
use 11.905908584595ms
htf@htf-All-Series:~/debug$ php a__.php
use 11.497020721436ms
```

```
htf@htf-All-Series:~/debug$ time php a__.php
```

```
real    0m0.090s
user    0m0.077s
sys     0m0.012s
```

# Zend引擎

php program.php

词法分析->生成Token

语法分析->生成OpCode

执行OpCode

创建HashTable, 添加数据

# Zend引擎

1. APC/OpCache只能优化PHP代码编译生成 OpCode的  
开销
2. OpCode执行构建内存中可用HashTable需要消耗CPU  
资源
3. LAMP每次请求结束会释放掉HashTable , 下一次请求  
再次构建HashTable内存



# Swoole应用服务器

1. 大数组、对象、常量等常驻内存，节省大量重复创建销毁的CPU消耗
2. PHP框架的环境路径计算、常量定义、初始化框架、类载入、解析注释语法等操作仅启动时执行一次
3. 业务代码仅剩最干净请求处理逻辑



# 超高性能





02

## PHP7+Swoole超高性能程序开发实践

# 短网址服务

- ◆ <http://chelun.com/url/D2M2qX>
- ◆ 3位检验码 + 自增ID ( 62进制 )
- ◆ swoole\_http\_server + redis 单机性能高达 5W+ QPS
- ◆ 作为车轮互联商业广告的流量入口和出口
- ◆ 统计设备号、UID、IP、UV、PV、地理位置等信息，为运营部门提供数据
- ◆ 统计逻辑基于Swoole Task功能实现，不影响核心逻辑



# 短网址服务

```
1 #!php
2 $serv = new swoole_http_server("127.0.0.1", 9502, SWOOLE_BASE);
3 $serv->set(array("worker_num" => 10));
4
5 $serv->on('workerStart', function() {
6     global $redis;
7     $redis = new redis;
8     $redis->connect('127.0.0.1', 6379);
9 });
10
11 $serv->on('request', function($request, $response) {
12     global $redis;
13
14     $request_url = $request->server['request_url'];
15     if (substr($request_url, 0, 5) != '/url/')
16     {
17         $response->status(400);
18         $response->end("<h1>bad request</h1>");
19     }
20     else
21     {
22         $url = substr($request_url, 5);
23         $key = substr($url, -5, 5);
24         $sid = substr($url, 0, strlen($url) - 5);
25         $result = $redis->hgetall($url.$sid);
26         if (empty($result))
27         {
28             $response->status(404);
29             $response->end("<h1>not found</h1>");
30         }
31         elseif ($result['key'] != $key)
32         {
33             $response->status(403);
34             $response->end("<h1>invalid key</h1>");
35         }
36         else
37         {
38             $response->status(202);
39             $response->header("location", $result['url']);
40             $response->end();
41         }
42     }
43 });
44
45 $serv->start();
```

```
Server Software:      swoole-http-server
Server Hostname:      127.0.0.1
Server Port:          9502

Document Path:        /url/abcefd
Document Length:       8 bytes

Concurrency Level:     100
Time taken for tests:   1.850 seconds
Complete requests:     100000
Failed requests:        0
Non-2xx responses:     100000
Keep-Alive requests:   100000
Total transferred:     18000000 bytes
HTML transferred:      0 bytes
Requests per second:   54066.33 [#/sec] (mean)
Time per request:      1.850 [ms] (mean)
Time per request:      0.018 [ms] (mean, across all c
Transfer rates:        9926.24 [Kbytes/sec] received
```





# MySQL-Proxy

1. 基于swoole\_mysql实现，支持php-fpm长连接
2. 后端使用连接池可以有效减少MySQL服务器的连接数。  
100台PHP机器 x 500进程 = 5W MySQL连接  
100台PHP机器 x 500进程 = 共用4000 MySQL连接
3. 支持MySQL后端服务路由，php-fpm到MySQL-Proxy只需要建立一个连接，即可向到多台MySQL服务器发送SQL
4. 近期会开源出来

# MySQL-Proxy

1. 早期版本基于 MYSQLI\_ASYNC、mysqli::reap\_async\_query、swoole\_mysqli\_get\_sock
2. 存在3个问题，1) connect连接是阻塞的，2) 结果较大可能会阻塞，3) 缓存区设置较小，有大量read, poll 系统调用导致sys很高
3. 新的API **swoole\_mysql** 自行解析MySQL二进制协议，阻塞时自动让出，实现了真正的**异步非阻塞**。并且改为 **64K** 缓存区，大大减少了系统调用次数。

# MySQL协议(query请求)

- ◆ 3字节长度+ 1字节packet\_id + 1字节cmd + n字节SQL语句
- ◆ <http://blog.csdn.net/wind520/article/details/43964821>

类型值	命令	功能
0x00	COM_SLEEP	(内部线程状态)
0x01	COM_QUIT	关闭连接
0x02	COM_INIT_DB	切换数据库
0x03	COM_QUERY	SQL查询请求
0x04	COM_FIELD_LIST	获取数据表字段信息



# MySQL协议(ResultSet)

响应报文类型	第1个字节取值范围
OK 响应报文	0x00
Error 响应报文	0xFF
Result Set 报文	0x01 - 0xFA
Field 报文	0x01 - 0xFA
Row Data 报文	0x01 - 0xFA
EOF 报文	0xFE



# MySQL异步API

```
$db = new mysqli;  
$server = array(  
    'host' => '192.168.56.102',  
    'user' => 'test',  
    'password' => 'test',  
    'database' => 'test',  
);  
  
$db->connect($server, function ($db, $r) {  
    if ($r === false) {  
        var_dump($db->connect_errno, $db->connect_error);  
        die;  
    }  
    $sql = "show tables";  
    $db->query($sql, function(mysqli $db, $r) {  
        global $s;  
        if ($r === false)  
        {  
            var_dump($db->error, $db->errno);  
        }  
        elseif ($r === true )  
        {  
            var_dump($db->affected_rows, $db->insert_id);  
        }  
        var_dump($r);  
        $db->close();  
    });  
});
```

# 高性能统计程序

接口名称	时间	调用次数	成功次数	失败次数	成功率	响应最大值	响应最小值	平均响应时间	失败平均时间
Service->CheLun->User->Token.check	00:00 - 23:55	47,720,592	47,720,592	0	100%	321ms	0ms	1.37ms	0ms
Service->CheLun->User->Info.isBlocked	00:00 - 23:55	20,217,901	20,217,750	151	100%	500ms	0ms	1.27ms	500ms
Service->CheLun->Forum->User.getHost	00:00 - 23:55	19,125,811	19,125,771	40	100%	501ms	0ms	0.56ms	500.35ms
Service->CheLun->User->Info.gets	00:00 - 23:55	1,257,104	1,257,080	24	100%	3170ms	0ms	70.5ms	500.04ms
Service->KJZ->Cms->City.getCityById	00:00 - 23:55	1,003,280	1,003,280	0	100%	1014ms	0ms	1.14ms	0ms



# 数据上报

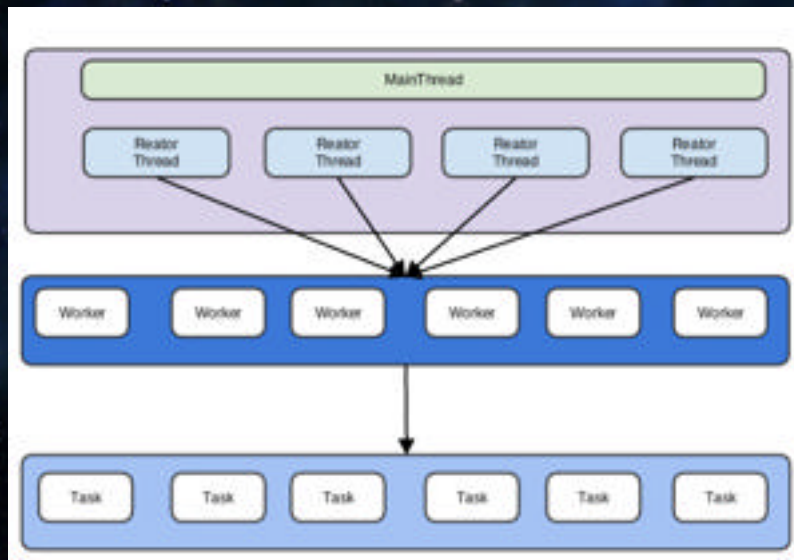
```
typedef struct
{
    uint32_t interface_id;
    uint32_t module_id;
    uint8_t success;
    uint32_t ret_code;
    uint32_t server_ip;
    uint32_t use_ms;
    uint32_t addtime;
} MostatPkg;
```

```
self::$_send_udp_pkg .= $pkg;

//60个统计时发送数据包, 避免超过最大传输单元, 1500 MTU
if (strlen(self::$_send_udp_pkg) >= self::STATS_PKG_LEN * self::STATS_PKG_NUM)
{
    self::sendPackage();
}
```



# 数据汇总与计算



```
$this->count[$key]['all']['total_count'] += 1;
$this->count[$key]['all']['total_time'] += $params['use_ms'];
if ($params['success'] == 0)
{
    $this->count[$key]['all']['fail_count'] += 1;
    $this->count[$key]['all']['total_fail_time'] += $params['use_ms'];
}

if ($params['use_ms'] > $this->count[$key]['all']['max_time'])
{
    $this->count[$key]['all']['max_time'] = $params['use_ms'];//更新最大值
}

if ($params['use_ms'] < $this->count[$key]['all']['min_time'])
{
    $this->count[$key]['all']['min_time'] = $params['use_ms'];//更新最小值
}
```



# 超高性能统计运算程序

1. 单机日均计算100亿条统计数据
2. 使用PHP Array 全内存 存储、计算数据。超大规模读写Redis会成为瓶颈
3. 使用Worker/Task进程实现数据的Map-Reduce
4. 推荐使用PHP的SPL数据结构，性能很好
5. 中间数据可以使用MySQL内存表，汇总计算后删除数据
6. PHP的GC非靠谱，及时unset掉不用的数据，连续运行无内存泄漏

# 超高性能统计运算程序

```
27788 root 20 0 373m 50m 1112 5 29.0 0.2 14388:54 stats_server: worker #2
27789 root 20 0 373m 49m 1112 5 29.0 0.2 14442:11 stats_server: worker #11
27792 root 20 0 373m 49m 1112 5 19.4 0.2 14386:38 stats_server: worker #4
27783 root 20 0 374m 49m 1112 5 18.7 0.2 14407:27 stats_server: worker #5
27698 root 20 0 373m 49m 1112 5 17.7 0.2 14348:16 stats_server: worker #0
27699 root 20 0 373m 49m 1112 5 17.7 0.2 14495:25 stats_server: worker #1
27700 root 20 0 373m 49m 1112 5 17.7 0.2 14448:42 stats_server: worker #10
27718 root 20 0 374m 49m 1112 5 17.7 0.2 14447:24 stats_server: worker #28
27701 root 20 0 373m 49m 1112 5 17.4 0.2 14403:01 stats_server: worker #3
27714 root 20 0 375m 50m 1112 8 17.4 0.2 14283:57 stats_server: worker #16
27716 root 20 0 373m 48m 1112 5 17.4 0.2 14379:06 stats_server: worker #18
27713 root 20 0 373m 49m 1112 5 17.1 0.2 14258:10 stats_server: worker #15
27705 root 20 0 374m 50m 1112 5 16.7 0.2 14341:29 stats_server: worker #7
27704 root 20 0 370m 45m 1112 5 16.4 0.1 14302:43 stats_server: worker #6
27707 root 20 0 375m 48m 1112 5 16.4 0.2 14429:02 stats_server: worker #9
27710 root 20 0 375m 50m 1112 5 16.4 0.2 14300:42 stats_server: worker #12
27721 root 20 0 373m 49m 1112 5 16.4 0.2 14452:25 stats_server: worker #23
27717 root 20 0 374m 50m 1112 8 16.1 0.2 14447:31 stats_server: worker #19
27706 root 20 0 372m 48m 1112 5 15.8 0.2 14387:46 stats_server: worker #8
27711 root 20 0 374m 50m 1112 5 15.8 0.2 14406:31 stats_server: worker #13
27712 root 20 0 374m 49m 1112 8 15.8 0.2 14392:44 stats_server: worker #14
27719 root 20 0 373m 50m 1112 5 15.8 0.2 14442:38 stats_server: worker #21
27720 root 20 0 373m 49m 1112 8 15.8 0.2 14261:55 stats_server: worker #22
27715 root 20 0 373m 49m 1112 8 14.8 0.2 14395:16 stats_server: worker #17
27695 root 20 0 420m 684 148 5 7.2 0.0 7355:13 stats_server: master
35207 yanchunh 20 0 340m 12m 5588 5 2.6 0.0 534:22.29 php server.php
38888 server_u 20 0 326m 5548 3132 5 2.3 0.0 103:31.19 logserver tasker 31
48859 server_u 20 0 326m 5836 3164 5 2.3 0.0 567:42.67 logserver tasker 32
48867 server_u 20 0 326m 5848 3168 5 2.3 0.0 547:04.63 logserver tasker 48
48389 server_u 20 0 326m 5580 3164 5 2.3 0.0 248:39.17 logserver tasker 25
3814 hantianf 20 0 874m 10m 8572 5 2.0 0.0 3525:32 /usr/local/php/bin/php ap
38889 server_u 20 0 326m 5548 3136 5 2.0 0.0 102:31.96 logserver tasker 27
38810 server_u 20 0 326m 5516 3104 5 2.0 0.0 103:09.28 logserver tasker 34
```

# Swoole 2.0 即将到来!

1. Swoole底层内置协程，不依赖PHP的Yield、Generator
2. 同步的代码，异步的IO
3. 告别callback hell，告别Go语言

```
$cli = new swoole_client_coro(SWOOLE_SOCK_TCP);  
$ret = $cli->connect("127.0.0.1", 8889);  
$ret = $cli->send("hello world in object");  
$ret = $cli->recv();  
$cli->close();
```

```
$cli = new swoole_mysql_coro("127.0.0.1", 9999);  
$cli->connect(['host'=>'127.0.0.1', 'user'=>'test',  
$ret = $cli->query("select sleep(1)", 2000);  
$cli->close();
```

```
$cli = new swoole_http_client_coro("127.0.0.1", 9999);  
$cli->setHeader([  
    "Host" => "api.mp.qq.com",  
]);  
$ret = $cli->get("/token?appid=test");  
$cli->close();
```

```
$cli = new swoole_redis_coro();  
$cli->connect("127.0.0.1", 9998);  
$ret = $cli->get("key");  
$cli->close();
```



# Swoole 2.0 为了更好的PHP!

1. 由TSF团队研发完成，提交到Swoole开源项目
2. 2.0预计在7月份发布，新版TSF 2.0预计8月发布

## 开发者列表

- 司超 chalesi chalesi@tencent.com
- 朱新宇 alvinzhu alvinzhu@tencent.com
- 王广超 winterswang winterswang@tencent.com
- 袁易之 markyuan markyuan@tencent.com
- 杨锡坤 rokettyyang rokettyyang@tencent.com





# THANK YOU

- 期待2017年再见 -

## Q & A