

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

PHIPCON

[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

<b>CPU消耗</b>	<b>耗时</b>	<b>单核</b>	<b>24核</b>
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

```
h1.php
5% = microtime(true);
$array = array (
0 => 'key_0_83581' => 'value_0_21754'
1 => 'key_1_95070' => 'value_1_38248'
2 => 'key_2_17909' => 'value_2_80813'
3 => 'key_3_77830' => 'value_3_12488'
4 => 'key_4_70487' => 'value_4_32428'
5 => 'key_5_32833' => 'value_5_80811'
6 => 'key_6_96156' => 'value_6_89611'
7 => 'key_7_53421' => 'value_7_96556'
8 => 'key_8_35021' => 'value_8_86495'
9 => 'key_9_89081' => 'value_9_34717'
10 => 'key_10_34611' => 'value_10_40090'
11 => 'key_11_92306' => 'value_11_84742'
12 => 'key_12_37544' => 'value_12_25780'
13 => 'key_13_81792' => 'value_13_65184'
14 => 'key_14_86726' => 'value_14_11012'
15 => 'key_15_75112' => 'value_15_79946'
16 => 'key_16_68180' => 'value_16_46927'
17 => 'key_17_54146' => 'value_17_14821'
18 => 'key_18_84782' => 'value_18_82375'
```

```
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



# 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_mysql;
$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_mysql $db, $r) {
        global $r;
        if ($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->DisLun->User->Token_check	00.00 - 23.55	47,720,592	47,720,592	0	100%	321ms	0ms	1.37ms	0ms
Service->DisLun->User->Info_isblocked	00.00 - 23.55	20,217,901	20,217,750	151	100%	500ms	0ms	1.27ms	500ms
Service->DisLun->Forum->User_getHost	00.00 - 23.55	19,125,811	19,125,771	40	100%	501ms	0ms	0.56ms	500.35ms
Service->DisLun->User->Info_gets	00.00 - 23.55	1,257,094	1,257,080	24	100%	3170ms	0ms	70.5ms	500.04ms
Service->WJZ->Oss->City_getCitySys	00.00 - 23.55	1,003,280	1,003,280	0	100%	1014ms	0ms	1.14ms	0ms



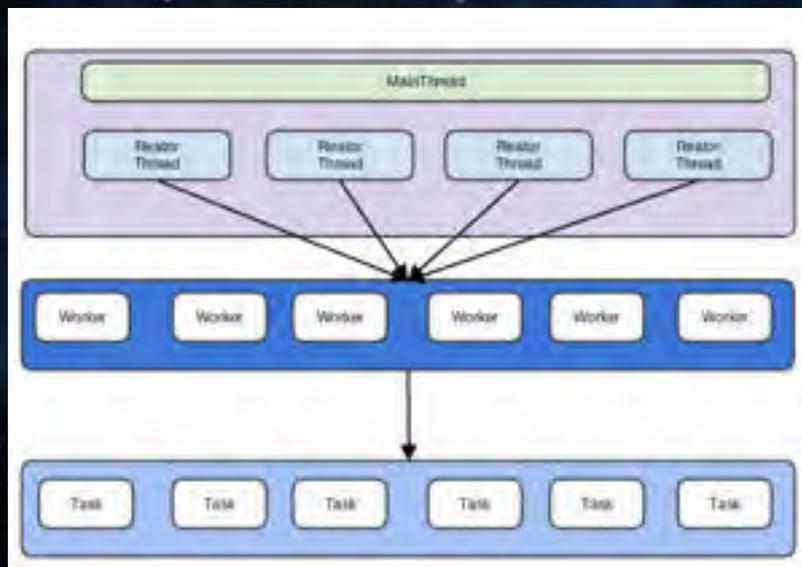
# 数据上报

```
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;

//600字左右为佳,发出数据报文,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'] = $params['use_ms']; //更新
}
if ($params['use_ms'] < $this->count[$key]['all']['min_time'])
{
    $this->count[$key]['all']['min_time'] = $params['use_ms']; //更新
}
```

# 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

27786 root	28	+	3786	386	3332	% 28.8	9.2	14346734	stats_servers	worker	32
27787 root	28	+	3786	386	3332	% 28.8	9.2	14347113	stats_servers	worker	33
27788 root	28	+	3786	386	3332	% 28.8	9.2	14346936	stats_servers	worker	34
27789 root	28	+	3786	386	3332	% 28.7	9.2	14347027	stats_servers	worker	35
27790 root	28	+	3786	386	3332	% 27.7	9.2	14346118	stats_servers	worker	36
27791 root	28	+	3786	386	3332	% 27.7	9.2	14345125	stats_servers	worker	37
27792 root	28	+	3786	386	3332	% 27.7	9.2	14346142	stats_servers	worker	38
27793 root	28	+	3786	386	3332	% 27.7	9.2	14346124	stats_servers	worker	39
27794 root	28	+	3786	386	3332	% 27.8	9.2	14346193	stats_servers	worker	40
27795 root	28	+	3786	386	3332	% 27.8	9.2	14346131	stats_servers	worker	41
27796 root	28	+	3786	386	3332	% 27.8	9.2	14346106	stats_servers	worker	42
27797 root	28	+	3786	386	3332	% 27.7	9.2	14346126	stats_servers	worker	43
27798 root	28	+	3786	386	3332	% 27.7	9.2	14346128	stats_servers	worker	44
27799 root	28	+	3786	386	3332	% 28.8	9.2	14347143	stats_servers	worker	45
27800 root	28	+	3786	386	3332	% 28.8	9.2	14346100	stats_servers	worker	46
27801 root	28	+	3786	386	3332	% 28.8	9.2	14346142	stats_servers	worker	47
27802 root	28	+	3786	386	3332	% 28.8	9.2	14346122	stats_servers	worker	48
27803 root	28	+	3786	386	3332	% 28.1	9.2	14346131	stats_servers	worker	49
27804 root	28	+	3786	386	3332	% 28.8	9.2	14346146	stats_servers	worker	50
27805 root	28	+	3786	386	3332	% 28.8	9.2	14346131	stats_servers	worker	51
27806 root	28	+	3786	386	3332	% 28.8	9.2	14346144	stats_servers	worker	52
27807 root	28	+	3786	386	3332	% 28.8	9.2	14346136	stats_servers	worker	53
27808 root	28	+	3786	386	3332	% 28.8	9.2	14346108	stats_servers	worker	54
27809 root	28	+	3786	386	3332	% 28.8	9.2	14346116	stats_servers	worker	55
27810 root	28	+	3786	386	3332	% 28.8	9.2	14346121	stats_servers	worker	56
33167 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402128	log_server	php	34
33168 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	35
33169 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402143	log_server	php	36
33170 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402184	log_server	php	37
33171 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402193	log_server	php	38
33172 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402142	log_server	php	39
33173 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	40
33174 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	41
33175 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	42
33176 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	43
33177 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	44
33178 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	45
33179 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	46
33180 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	47
33181 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	48
33182 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	49
33183 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	50
33184 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	51
33185 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	52
33186 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402148	log_server	php	53
33187 jenkins01	28	+	3486	326	3164	% 27.8	9.0	54402146	log_server	php	54

# 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", 8888);  
$ret = $cli->send("hello world in object");  
$ret = $cli->recv();  
$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_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_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