PHPCON 2017

车轮Swoole微服务架构

@hantianfeng Rango-韩天峰 / 车轮互联

关于我

- 车轮互联总架构师
- PHP官方扩展开发组成员
- 微博：@hantianfeng
- Github: https://github.com/matyhtf

分享内容

一．Service通信框架

二．微服务架构

三．微服务公共组件

01

Swoole 实现 Service 服务框架

为什么要进行服务化治理（SOA）

基于数据库表、Redis实现服务

1. 存储层未隔离，数据不可迁移调整，耦合性极高
2. 扩展性、可维护性极差

PHP代码(函数/类)实现服务

1. 客户端必须include/require一个文件
2. 代码修改必须通知所有业务方升级
3. 可能存在不同版本的兼容问题
4. 无法跨语言，不支持C++、Java等其他语言程序

HTTP+JSON的Web API

1. 解决了耦合问题，可以提供良好的服务隔离
2. 优点：目前最通用的服务治理方案
3. 缺点：a) Http不支持并发，b) 长连接支持差，c) 不支持订阅与消息主动推送

基于Swoole实现Service方案

1. 基于Swoole提供的包头+包体自动协议处理
2. Server/Client两端无需写任何底层代码
3. 支持单连接并发，客户端只需要与服务器建立一条连接
4. 支持php-fpm中使用TCP长连接，需要依赖swoole扩展
5. 自带Task进程池功能，可直接将慢速请求异步执行
6. 支持跨语言调用，C++、Java等其他语言程序也可以方便使用
7. Server/Client两端可以实现异步

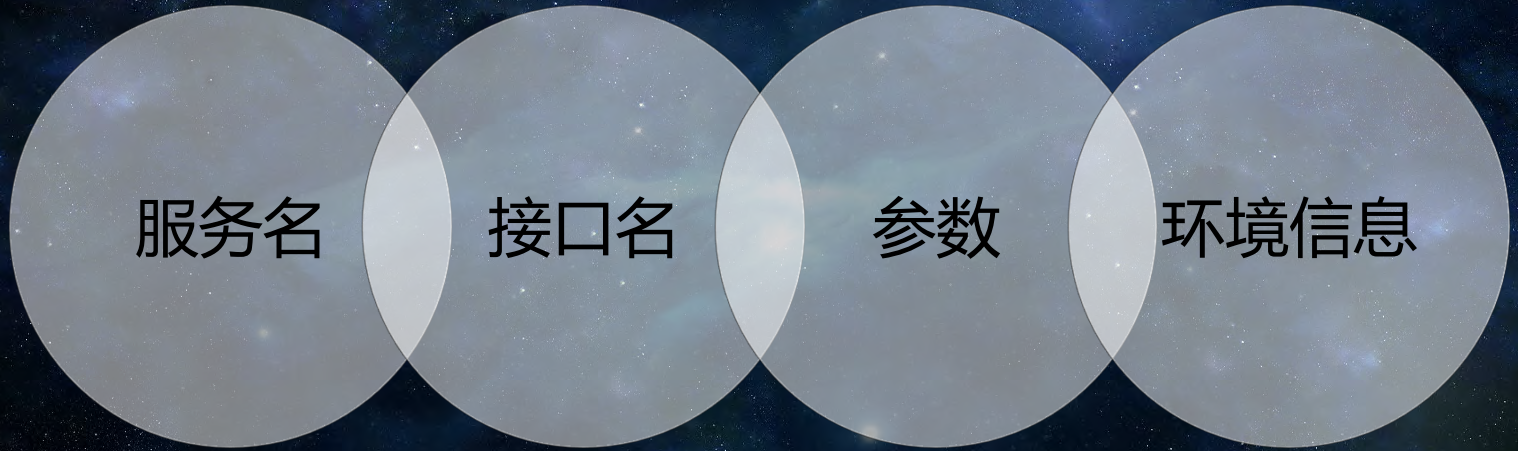
REST

1. Domain：dns + host + port
2. Header
3. Cookie
4. Method：GET/POST/DELETE/PUT
5. Parameter：name=rango&value=test#stop
6. POST：form-data格式、urlencode格式

微服务理念

1. **轻量**：客户端、服务端无需额外工作即可运行
2. **简单**：元素少，使用简单，无需培训，无需手册

车轮微服务



车轮微服务 ：调用方

```php
$service = new Service('payment');
```

```php
$service->call('User\Info::get', 100001);
```

```php
$service->putEnv(array(
    'appid' => 'payment',
    'appkey' => 'c36314d61',
));
```

车轮微服务 ：串行调用

```php
$res1 = $service->call('User\Info::get', 100001);
//如果返回NULL，表示网络调用失败了，请检查$res->code
$result1 = $res->getResult();

$res2 = $service->call('User\Info::get', 100002);
$result2 = $res2->getResult();

$res3 = $service->call('User\Info::get', 100003);
$result3 = $res3->getResult();
```

车轮微服务 ：并行调用

```php
$res1 = $service->call('User\Info::get', 100001);
$res2 = $service->call('User\Info::get', 100002);
$res3 = $service->call('User\Info::get', 100003);

//0.5表示500毫秒超时，$n表示成功返回的请求个数
//如果少于发起的请求数，证明有个别请求超时了
$n = $service->wait(0.5);

$result1 = $res1->getResult();
$result2 = $res2->getResult();
$result3 = $res3->getResult();
```

车轮微服务 ：提供方

```php
namespace Payment\User;

class Info
{
    static function get($uid)
    {
        $info = array(/* data */);
        return $info;
    }
}
```

```
[root@s0084-gz service]# php server.php start -d
[root@s0084-gz service]# ps aux|grep Pay
root     46806  0.0  0.0 610180  9648 ?        Ssl  13:58   0:00 PayServer: master -host=127.0.0.1 -port=8808
root     46807  0.0  0.0 381540  9152 ?        S    13:58   0:00 PayServer: manager
root     46812  0.0  0.0 385460  9296 ?        S    13:58   0:00 PayServer: worker
root     46813  0.0  0.0 385460  9296 ?        S    13:58   0:00 PayServer: worker
root     46814  0.0  0.0 385460  9296 ?        S    13:58   0:00 PayServer: worker
root     46815  0.0  0.0 385460  9296 ?        S    13:58   0:00 PayServer: worker
root     46820  0.0  0.0 103248   856 pts/30   S+   13:58   0:00 grep Pay
[root@s0084-gz service]# php server.php
================================================================================
Usage: php server.php start|stop|reload
================================================================================
           -d, --daemon      启用守护进程模式
    -h, --host [<value>]     指定监听地址
    -p, --port [<value>]     指定监听端口
               --help        显示帮助界面
            -b, --base       使用BASE模式启动
  -w, --worker [<value>]     设置Worker进程的数量
  -r, --thread [<value>]     设置Reactor线程的数量
  -t, --tasker [<value>]     设置Task进程的数量
```

```php
<?php
$serv = new swoole_server("127.0.0.1", 9501, SWOOLE_BASE);

$serv->set(array(
                'open_length_check'     => true,
                'dispatch_mode'         => 1,
//              'worker_num'            => 4,
                'package_length_type'   => 'N',
                'package_length_offset' => 0,       //第N个字节是包长度的值
                'package_body_offset'   => 4,       //第几个字节开始计算长度
                'package_max_length'    => 2000000,  //协议最大长度
));
```

```php
<?php
$client = new swoole_client(SWOOLE_SOCK_TCP);

$client->set(array(
    'open_length_check'     => true,
    'package_length_type'   => 'N',
    'package_length_offset' => 0,       //第N个字节是包长度的值
    'package_body_offset'   => 4,       //第几个字节开始计算长度
    'package_max_length'    => 2000000,  //协议最大长度
));
```

```
struct
{
    uint32_t length;
    uint32_t type;
    uint32_t uid;
    uint32_t serid;
    char body[0];
}
```

- length: 包体的长度
- type: 包体的打包格式，=1使用PHP串化格式，=2使用JSON格式，其他格式暂未支持
- uid: 用户自定义的ID，保留字段
- serid: Request/Response 串号

为什么不用Thrift、ProtoBuf

1. 优点：解包/打包性能好，IDL，自动生成多语言调用代码，对静态语言友好
2. 缺点：服务提供者需要编写维护IDL文件，门槛较高，不方便抓包调试

运行环境

1. Server支持 Win/Mac（仅开发）、Linux（生产环境）
2. 服务提供方可以使用 Java、C++、GO 实现接口
3. PHP Client支持 stream、sockets、swoole 3种驱动
4. 支持语言客户端 Java、C++、Node.js、GO、Python

02

Swoole
微服务架构


服务发现
集群管理
负载均衡
调用统计
监控报警
故障转移
微服务

配置中心

1. 每台服务器安装一个NodeAgent程序
2. 配置文件可以拉也可以推
3. 可以根据集群key得到机器列表

服务发现 & 负载均衡

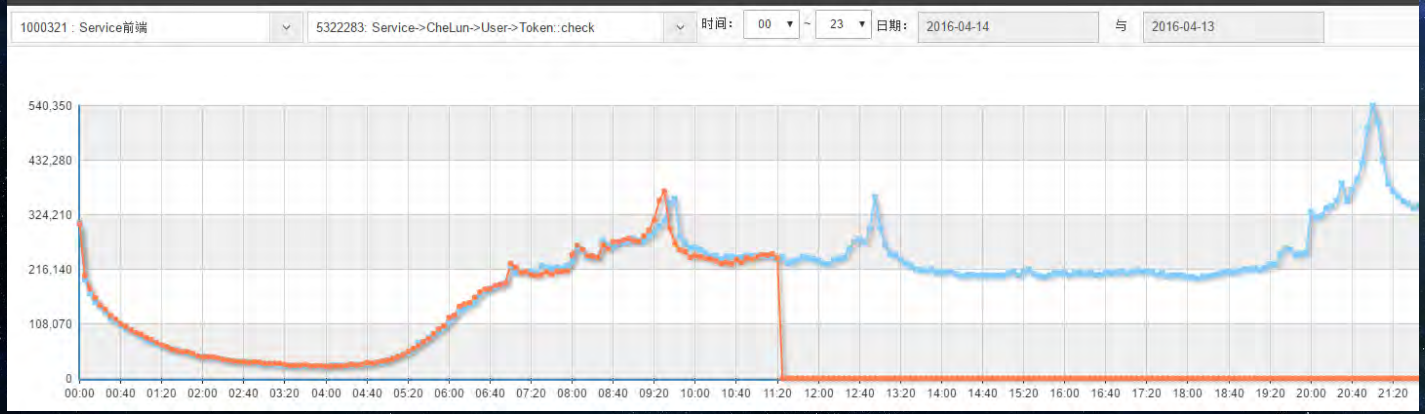
1. 服务器程序 *onStart* 时调用 *curl http://config_center/api/online* 注册到集群，并设置为在线
2. 终止运行脚本 *php server.php stop* 前调用 curl http://config_center/api/offline 从集群中摘除
3. 配置中心收到节点变更时会主动推送新的机器列表到调用端
4. 基于客户端实现的权重+随机，选择不同的集群节点

KeepAlived守护进程

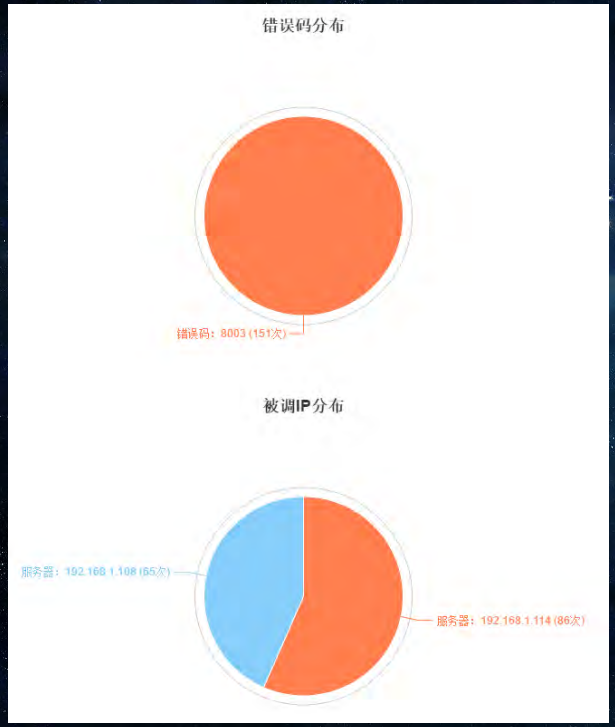
1. 每秒调用一次Service接口侦测集群每个节点是否可用
2. 发现节点无法访问，自动将此节点从Service集群中摘除
3. 发现节点重新可用时，自动将此节点加入Service集群
4. 配置中心发现有变更时，自动推送新配置到调用端

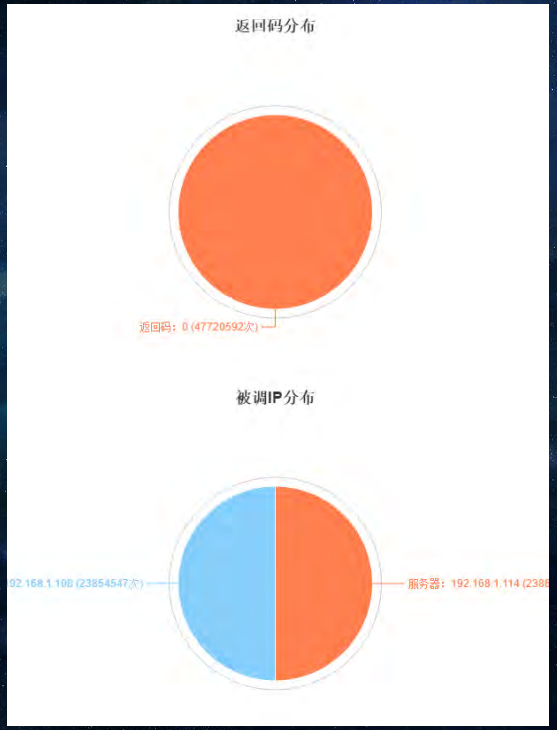
统计与监控报警

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->Crm->City::getCityById	00:00 ~ 23:55	1,003,280	1,003,280	0	100%	1014ms	0ms	1.14ms	0ms



统计与监控报警


错误码分布
错误码：8003 (151次)
被调IP分布
服务器：192.168.1.108 (65次)
服务器：192.168.1.114 (86次)


返回码分布
返回码：0 (47720592次)
被调IP分布
服务器：192.168.1.114

统计与监控报警

Service->CheLun->User->Token::check	18:45 ~ 18:50	195,447	195,447	0	100%	202ms	0ms	1ms	0ms
Service->CheLun->User->Token::check	18:50 ~ 18:55	195,846	195,846	0	100%	129ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	18:55 ~ 19:00	192,205	192,205	0	100%	243ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:00 ~ 19:05	193,641	193,641	0	100%	205ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:05 ~ 19:10	191,228	191,228	0	100%	205ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:10 ~ 19:15	195,286	195,286	0	100%	152ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	19:15 ~ 19:20	193,705	193,705	0	100%	214ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	19:20 ~ 19:25	193,876	193,876	0	100%	420ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:25 ~ 19:30	195,251	195,251	0	100%	206ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:30 ~ 19:35	196,855	196,855	0	100%	204ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	19:35 ~ 19:40	198,460	198,460	0	100%	208ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	19:40 ~ 19:45	207,626	207,626	0	100%	209ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:45 ~ 19:50	202,181	202,181	0	100%	237ms	1ms	2ms	0ms
Service->CheLun->User->Token::check	19:50 ~ 19:55	200,920	200,920	0	100%	212ms	0ms	2ms	0ms
Service->CheLun->User->Token::check	19:55 ~ 20:00	200,106	200,106	0	100%	242ms	0ms	2ms	0ms

统计与监控报警

模块名称

Service前端

报警策略

开启 关闭

报警间隔时间(分钟)

30

成功率阀值(0-100)

99.00

调用量波动阀值(0-100)

300.00

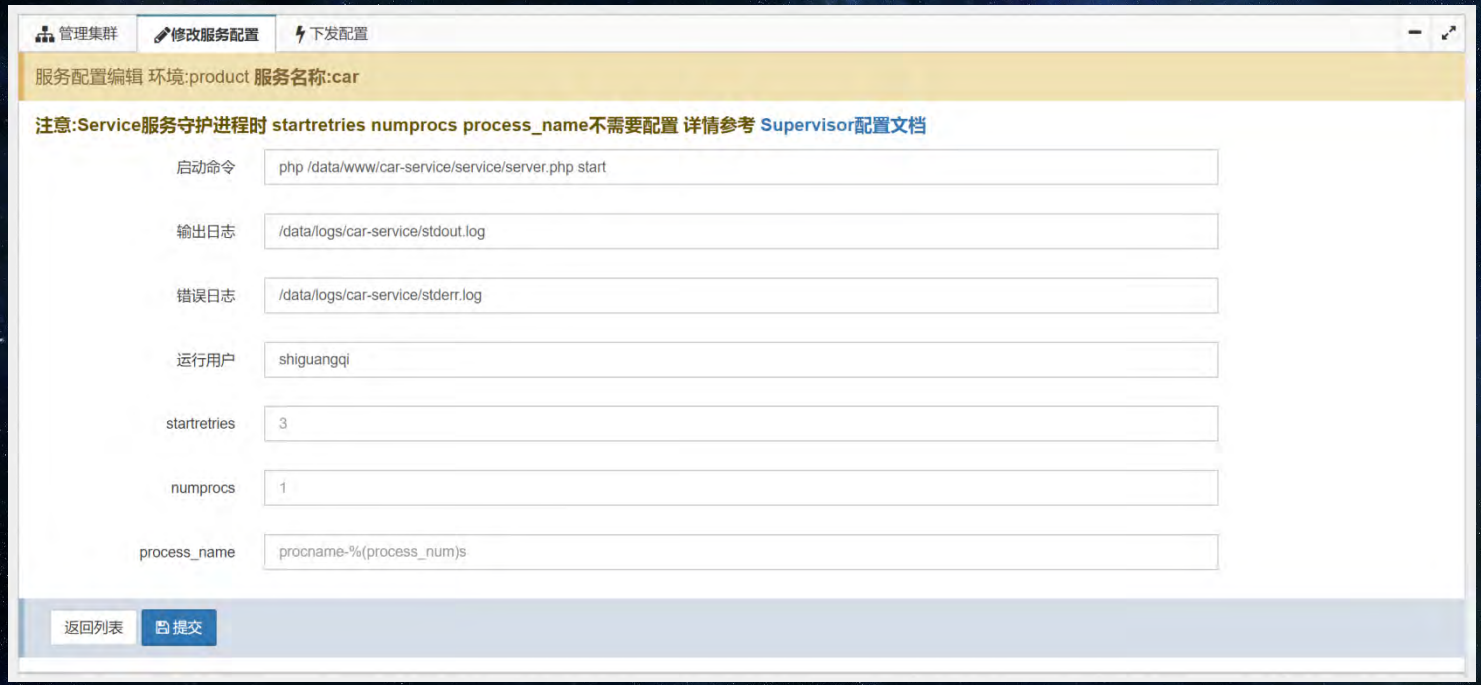
负责人

韩天峰 [hantianfeng]

备份负责人

王欢 [wanghuan] 徐志强 [xuzhiqiang] 项东东 [xiangdongdong] 石光启 [shiguangqi]

服务注册


管理集群
修改服务配置
下发配置
服务配置编辑 环境:product 服务名称:car
注意:Service服务守护进程时 startretries numprocs process_name不需要配置 详情参考 Supervisor配置文档
启动命令
php /data/www/car-service/service/server.php start
输出日志
/data/logs/car-service/stdout.log
错误日志
/data/logs/car-service/stderr.log
运行用户
shiguangqi
startretries
3
numprocs
1
process_name
procname-%(process_num)s
返回列表
提交

服务注册


管理集群
修改服务配置
下发配置
服务器IP
服务器PORT
权重（0-100）
+添加到集群
返回列表
IP PORT 权重 配置状态 运行状态 操作
1 192.168.1.238 8805 100 online 0:car RUNNING pid 33310, uptime 6 days, 1:37:29 下线 重启 删除
2 192.168.1.239 8805 100 online 0:car RUNNING pid 13375, uptime 16 days, 22:55:47 下线 重启 删除

进程调度

1. 基于 Supervisor 实现，机器启动时自动启动 Service 进程
2. 远程调度和扩容，在管理端增加新节点，通知 SuperVisor 启动 Service 进程
3. 断电重启，自动恢复全部服务

基于发布系统自动部署

1. 发布代码后执行*php server.php reload*
2. Service程序会重启工作进程加载最新的代码

Swoole Compiler

1. 将PHP源码编译为二进制程序，保护源代码不被泄漏和修改
2. 通过发布系统发布二进制版本到线上集群
3. 无需opcache/apc即可运行

03

基于 Swoole 开发公共组件

StatsServer

1. 基于Swoole开发，日均处理100亿以上请求
2. UDP协议
3. Master -> Worker(x24) -> Task(x24)
4. Map-Reduce
5. cpp-swoole + protobuf 密集计算性能优化

NodeAgent

1. 部署到线上每台机器
2. 加密传输大文件（1G）
3. mcrypt扩展AES 128位加密
4. 收集机器节点信息
5. 发送reload信号到Server程序
6. 配置中心基于NodeAgent程序实现配置文件主动推送

MySQL-Proxy 2.0

1. 基于Swoole+PHP-X实现，完全实现MySQL协议
2. 客户端直接使用MySQLi/PDO长连接
3. 超高性能，比Kingshard（GO语言实现）性能高20%
4. 后端使用连接池可以有效减少MySQL服务器的连接数
5. 支持自动读写分离，支持事务处理
6. 管理端可统计SQL执行耗时、慢SQL、超大结果集请求
7. 管理端可实时配置数据库、用户、IP授权

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(ResuleSet)

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

THANK YOU

- 期待2018年再见-

Q & A