

SACC 第八届中国系统架构师大会
2016 SYSTEM ARCHITECT CONFERENCE CHINA 2016

架构 创新之路

PHP高性能服务化框架

刘宇

2016-10-08

关于我

- 毕业于北京邮电大学计算机学院
- 曾混迹于Siemens、Motorola、Yahoo中国、淘宝、OpenTV等公司
- 现就职于易到用车担任架构师
- 参与过路由交换设备、手机App、机顶盒、互联网前后端的开发工作

提纲

- 服务化概况
- PSF介绍
- PSF应用情况

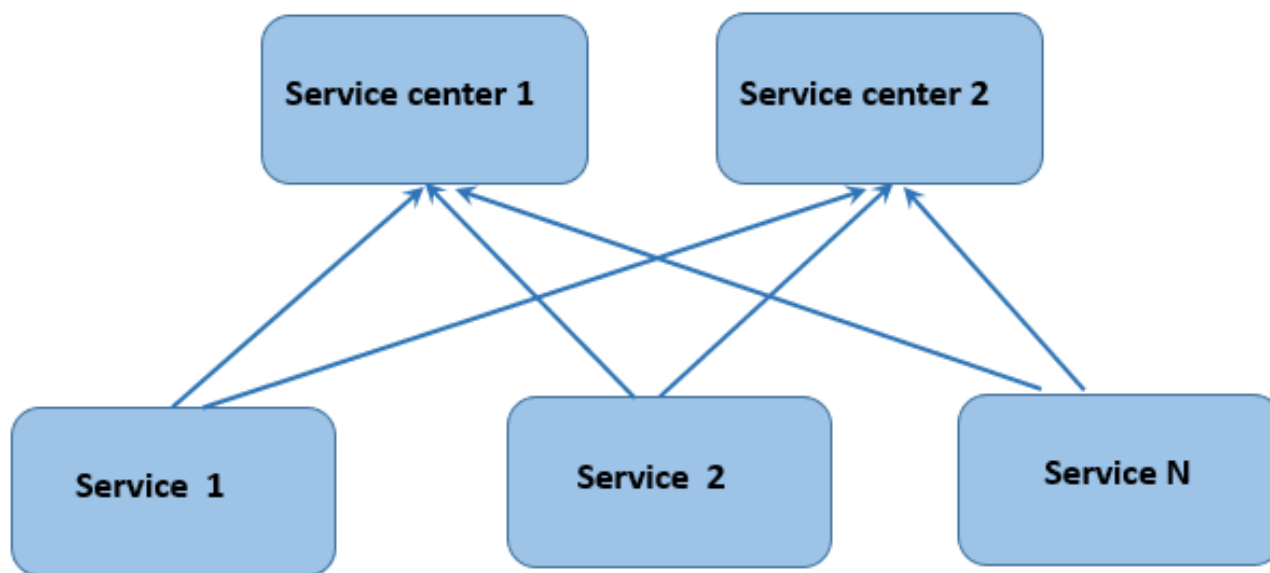
服务化框架概况

- 内部模块服务化是主流做法：RPC
- 阿里巴巴的HSF、dubbo (Java)
- Facebook的thrift (C++、Java、Python、Ruby、PHP)
- Google grpc (通用RPC框架)
- Twitter的finagle
- PHP swoole

为啥选择自造轮子

- 简单可控，自成体系：采用标准数据格式：URL + JSON
- 高性能要求和追求
- 只有RPC还不够

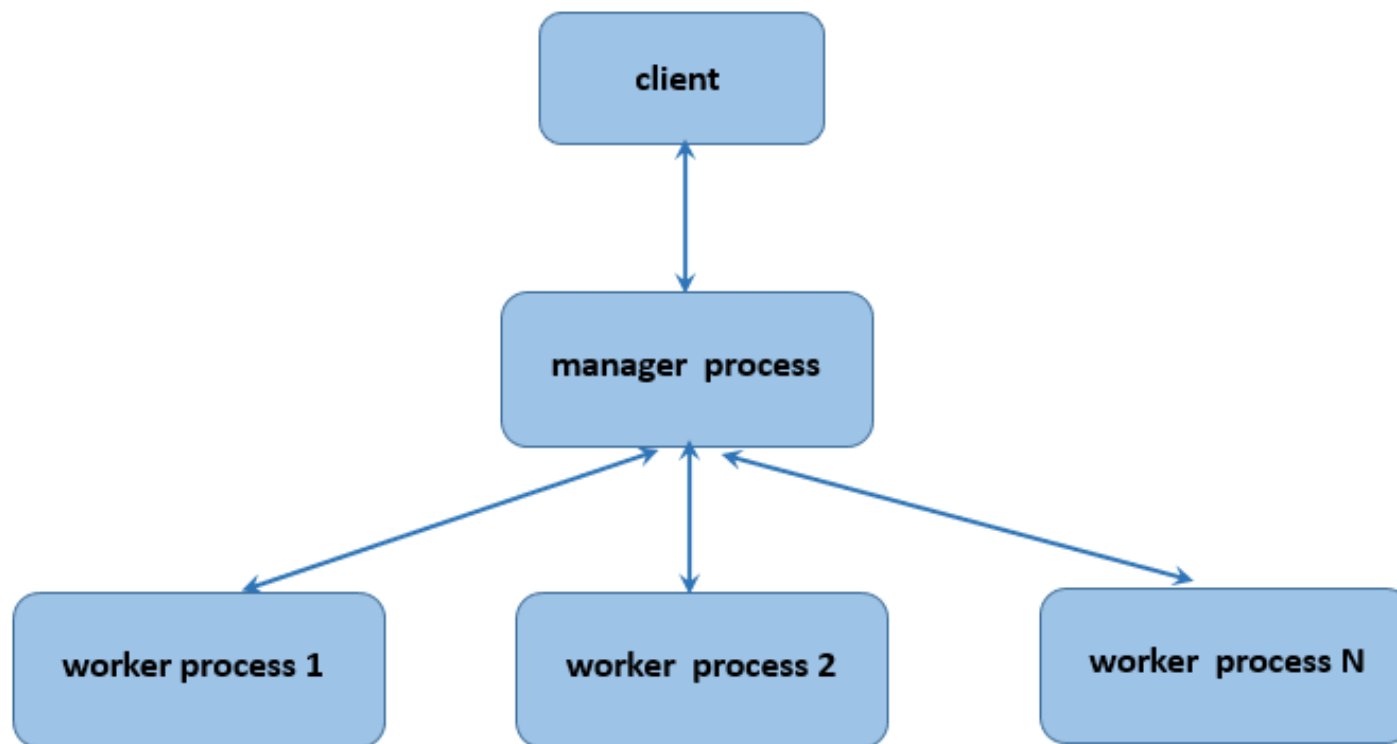
PSF系统架构



架构解读

- service center服务注册和服务发现，负载均衡
- 多台service center机器相互独立
- client直接和service机器连接和通信
- 基于连接数的负载均衡

PSF工作机制



工作机制解读

- client和服务通过socket通信
- manager进程和worker进程通过pipe通信
- manager进程为C进程
- worker进程为PHP进程。用C实现基于pipe通信的处理框架，调用PHP处理函数

PSF特点

- powered by C : 我们用C实现了PHP RPC应用服务器, C接管了网络连接和网络IO, 性能非常强劲, 框架层面不存在任何性能瓶颈
- 改变PHP运行方式: PHP以后台daemon方式高效运行
- 简洁高效的协议: 采用IP直连的长连接和私有二进制协议

PSF和Web Server对比优势

- daemon vs. request
- 不需要负载均衡层
- 不需要域名解析
- 协议非常简洁，协议头只有10字节。没有臃肿的HTTP header，如Accept、Host、Content-Type、Content-Length等等

PSF程序及PHP扩展

- psf-server : psf_server_managerd 和 psf_service_centerd
- psf-tool : psf_stat、 psf_server_stat、 psf_client_call等
- psf_server_worker扩展
- psf_client_sdk扩展
- psf-tsar-module : tsar插件

PSF依赖的基础库

- libfastcommon
- libserverframe

PSF 原生Service编写

```
$start = $argc - 4;
```

```
$read_fd = $argv[$start];
```

```
$write_fd = $argv[$start + 1];
```

```
$handle = psf_worker_init($read_fd, $write_fd, 'deal_message');
```

```
psf_worker_loop($handle);
```

```
function deal_message($service_uri, $data) {
```

```
...
```

```
}
```

Service中使用全局变量

- 尽可能兼顾web程序开发方式
- 支持URL方式，注入\$_REQUEST
- 自动识别body中的URL encoded数据
- 支持HTTP Header，注入\$_SERVER

PHP RPC框架

- PHP-RPC项目
- `psf_php_client_sdk_call('driver',
'auth/checkSession?user_id=12345&sid=888888',
, 30);`
- class file : `Driver/Service/Auth/Auth.php`

```
class Auth {
```

```
    public static function checkSession($params, $data) {
```

PSF发展历程

- 2014年9月发布第一个版本
- 使用PSF的模块（10+）：

设备认证

风控系统

消息中心

设备中心

新订单和新派单

订单和派单系统重构

- 使用PHP 7.0
- 后端资源全部采用长连接

DB : DBConnection & DB & PartitionDB

Cache : PartitionRedis

RabbitMQ : DMQ

- ConfigCache

DBConnection简介

- 连接复用：连接基于DB机器，而不是database
- DBConnectionCache

PartitionRedis简介

- 采用 $M * N$ 方式，配置简单
- 数据分片，按key求模
- failover机制：所有机器构成一个环，后为前备
- 数据预热加载机制：onConnected回调函数

性能表现

- 设备中心：单机峰值QPS达到2w，RT不到1ms
- 新老订单系统对比：

RT : 30ms vs 3ms

机器数 : 20台 vs. 5台

期待你的加入

- 发邮件: liuyu@yongche.com

THANKS

SequeMedia
盛拓传媒

IT168.com
专注 服务 16 年

ChinaUnix

ITPUB
www.itpub.net