QQ

# Swoole2.0原生协程高性能开发实践

alvinzhu(朱新宇)

PHPCON
www.phpconchina.com

# 关于我

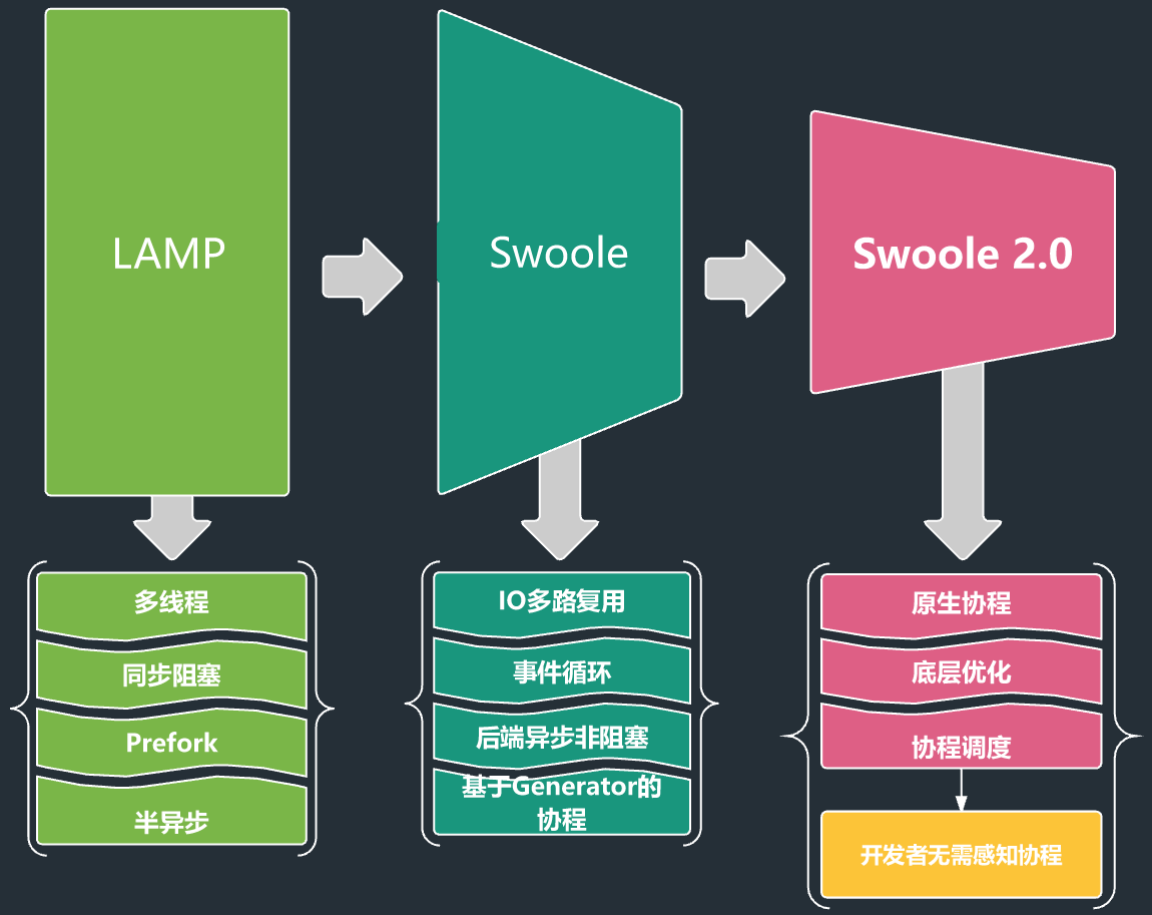
- 2015年 毕业于上海交通大学
- 2015年 进入腾讯即通综合部
  - 企业QQ
  - 营销QQ
  - QQ看点

- 开源社区：
  - Swoole
  - TSF

# 大纲

- Swoole2.0原生协程探索之路
- PHP高可用应用层框架建设
- QQ看点Web前后端高可用结构优化

# PHP的并发IO之路




```
foo($params, function($params){
    bar($params, function($params){
        baz($params, function($params){
            //......
        });
    }));
});
```

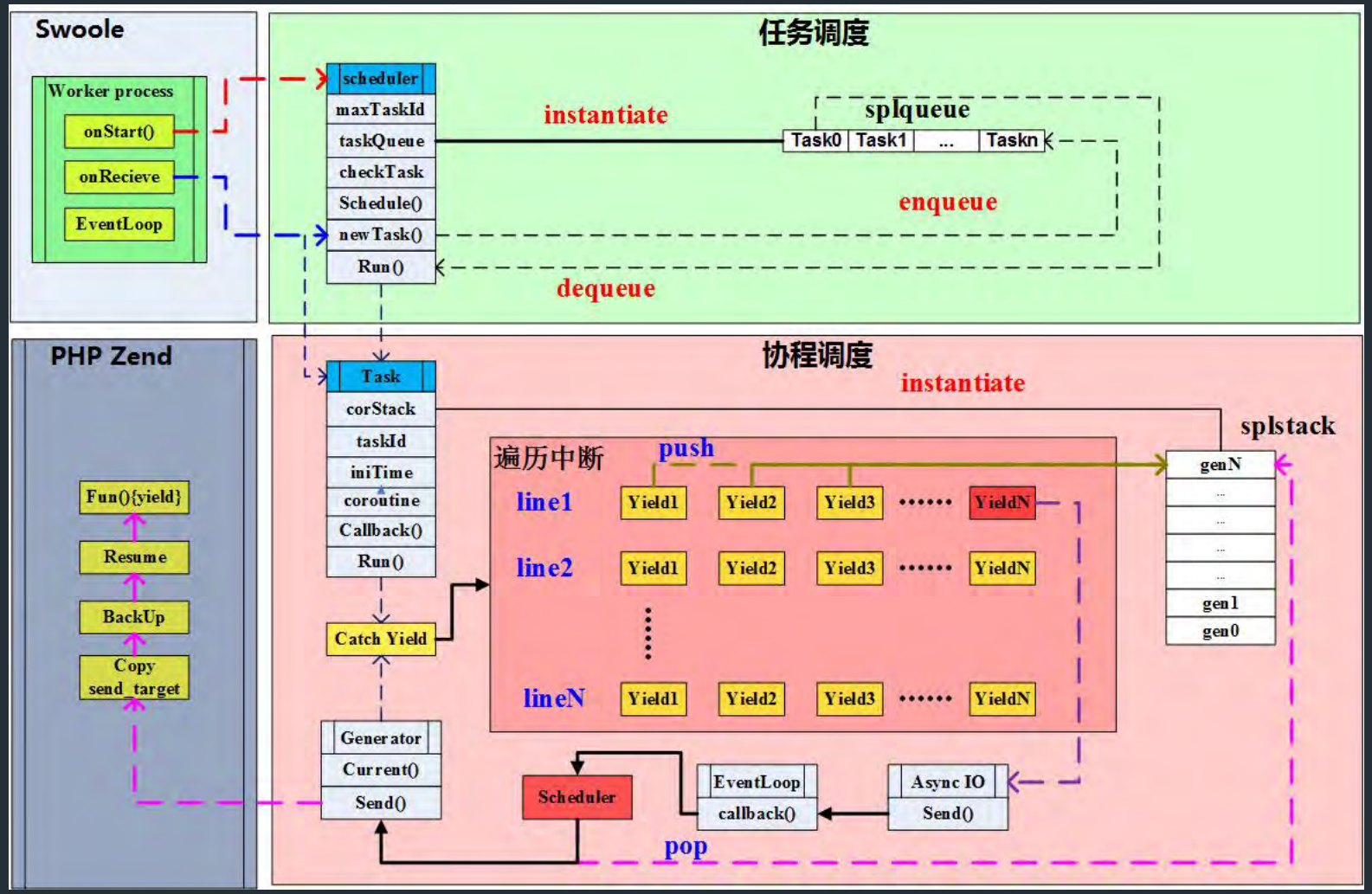
Generator

```
yield foo($params);
yield bar($params);
yield baz($params);
```

?

# Yield协程

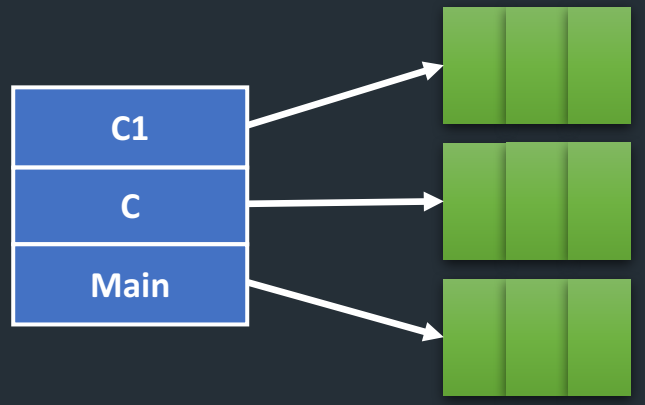
- 什么是协程
- 函数重入能力
  - PHP堆栈的分离，保存
  - Yield实现协程切出
  - 双向通信能力
- 如何协程实现非阻塞IO



# Yield问题

- 开发效率低，入门门槛高
  - 这里什么要yield？
  - 为什么代码没反应？
- 内存分配低效

```
A($params);
yield B($params);
C($params);
```




```
function B($params)
{
    yield $cli->TCPSend();
}
function C($params)
{
    return C1($params);
}
function C1($params)
{
    return “nosense”;
}
```

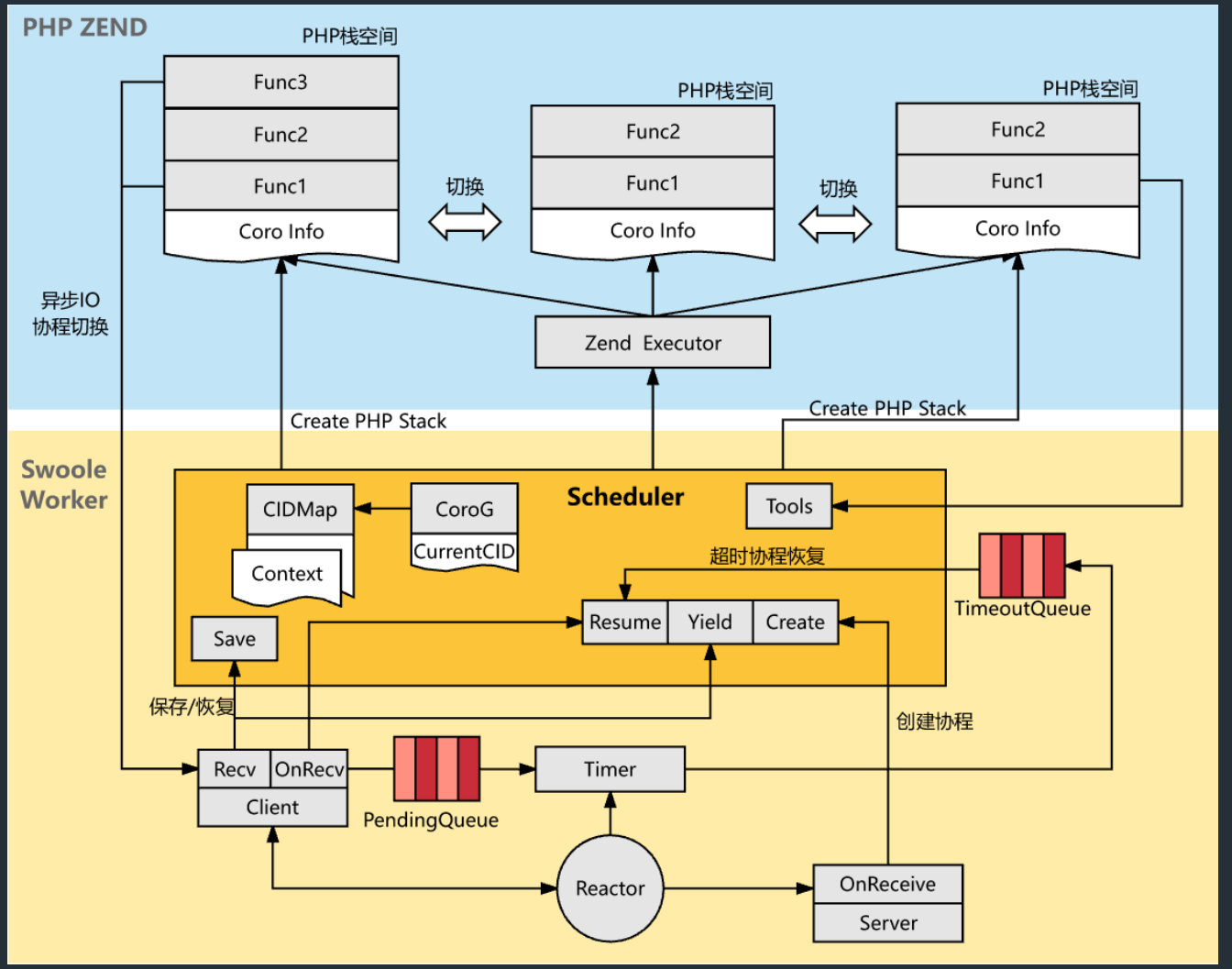
IF

```
function C1($params)
{
     yield $client->send();
}
```

对协程的认知被转变成了对Generator语法和原理的认知

# 原生协程

- 栈分离—打破迭代器限制
- swoole底层分配协程栈
- Scheduler负责协程管理
  - 协程创建
  - 上下文保存
  - 协程切出
  - 协程恢复
  - 超时管理



# 协程实践

## TCP/UDP

```
$cli = new Swoole\Coroutine\Client(SWOOLE_SOCK_UDP);
$ret = $cli->connect("127.0.0.1", 8888);
$ret = $cli->send("hello world at the first time");
$ret = $cli->recv(2);
$cli->close();
```

## Mysql

```
$cli = new Swoole\Coroutine\MySQL();
$cli->connect(['host' => '127.0.0.1', 'user' => ‘root',
    'password' => '1', 'database' => 'test']);
$ret = $cli->query('Select * from test limit 1');
$cli->close();
```

## HTTP

```
$cli= new Swoole\Coroutine\Http\Client('0.0.0.0', 9510);
$cli->set(['timeout' => 1]);
$cli->setHeaders([
    "Host" => "api.mp.qq.com",
    "User-Agent" => "Chrome/49.0.2587.3",
]);
$ret = $cliAA->get("/cn/token");
```

## Redis

```
$redis = new Swoole\Coroutine\Redis();
$res = $redis->connect('127.0.0.1', 6379);
$res = $redis->set('key_tmp', 'Hello World');
```

# 协程实践

## Multi-Call

```
$http_client= new Swoole\Coroutine\Http\Client('qq.com', 80);
$http_client->setDefer();
$http_client->get('/');
$mysql = new Swoole\Coroutine\MySQL();
$res = $mysql->connect(['host' => '192.168.244.128', 'user' => 'mha_manager',
                        'password' => 'mhapass', 'database' => 'tt']);
$mysql->setDefer();
$mysql->query('select sleep(1)', 2);
$res = $http_client->recv();
$mysql_res = $mysql->recv();
```

# 性能测试

压测环境：8核 16G虚拟机  c 200 d30s

Http长连接

| | Swoole2.0 | callback | PHP yield |
|---|---|---|---|
| **Echo** | 540000 | 530000 | 80000 |
| **1次后端** | 83900 | 82400 | 41000 |
| **3次后端** | 31000 | 31000 | 20000 |

Http短连接

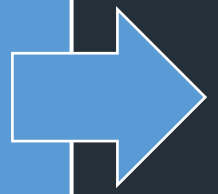
| | Swoole2.0 | callback | PHP yield |
|---|---|---|---|
| **Echo** | 56000 | 55000 | 43000 |
| **1次后端** | 43600 | 43300 | 30000 |
| **3次后端** | 27000 | 27000 | 17000 |

QQ

## Swoole

小而美

- 提供异步高性能IO
- 原生协程能力
- 通用原子能力封装



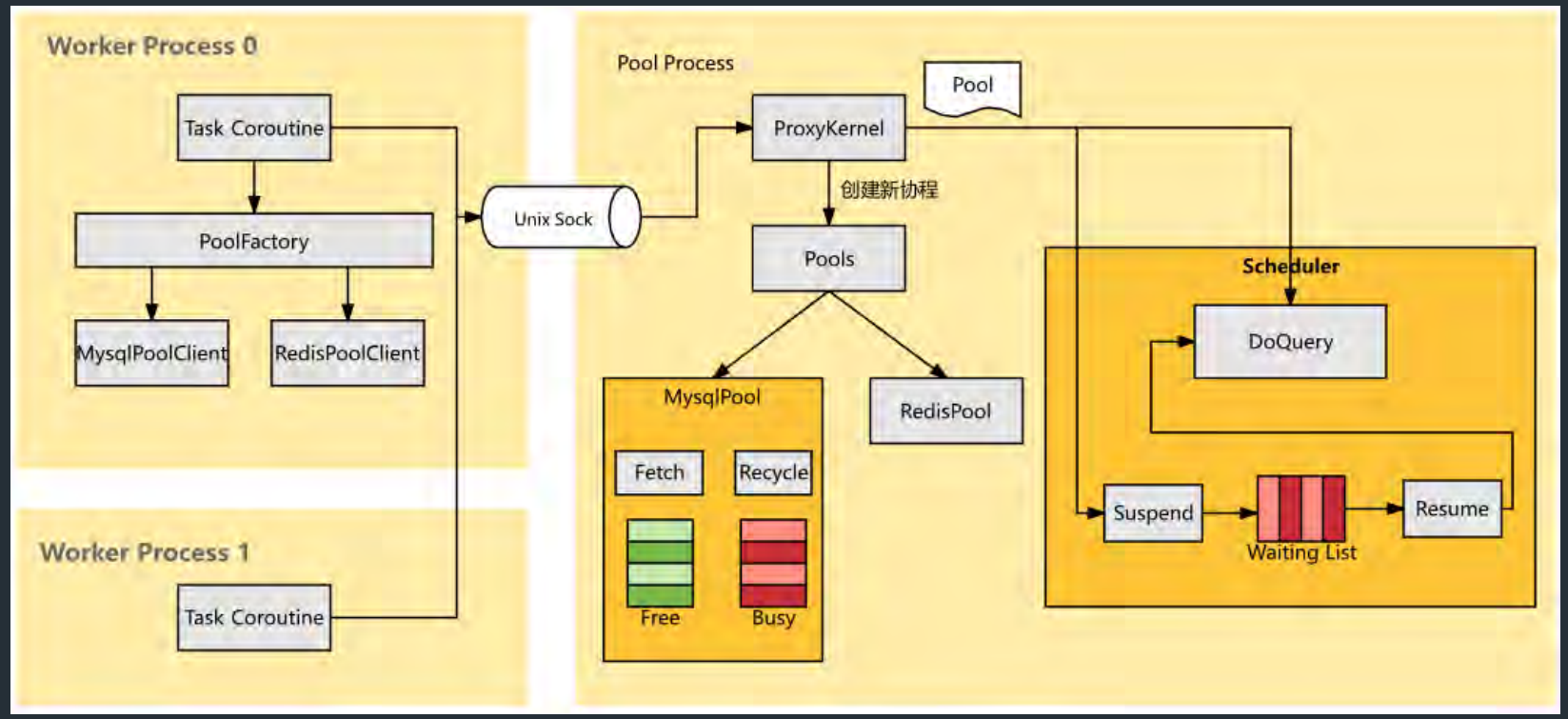
## TSF

系统级高可用

- 快速搭建、高效开发
- 连接池方案
- 频控组件
- 过载保护
- 监控上报
- 组件扩展

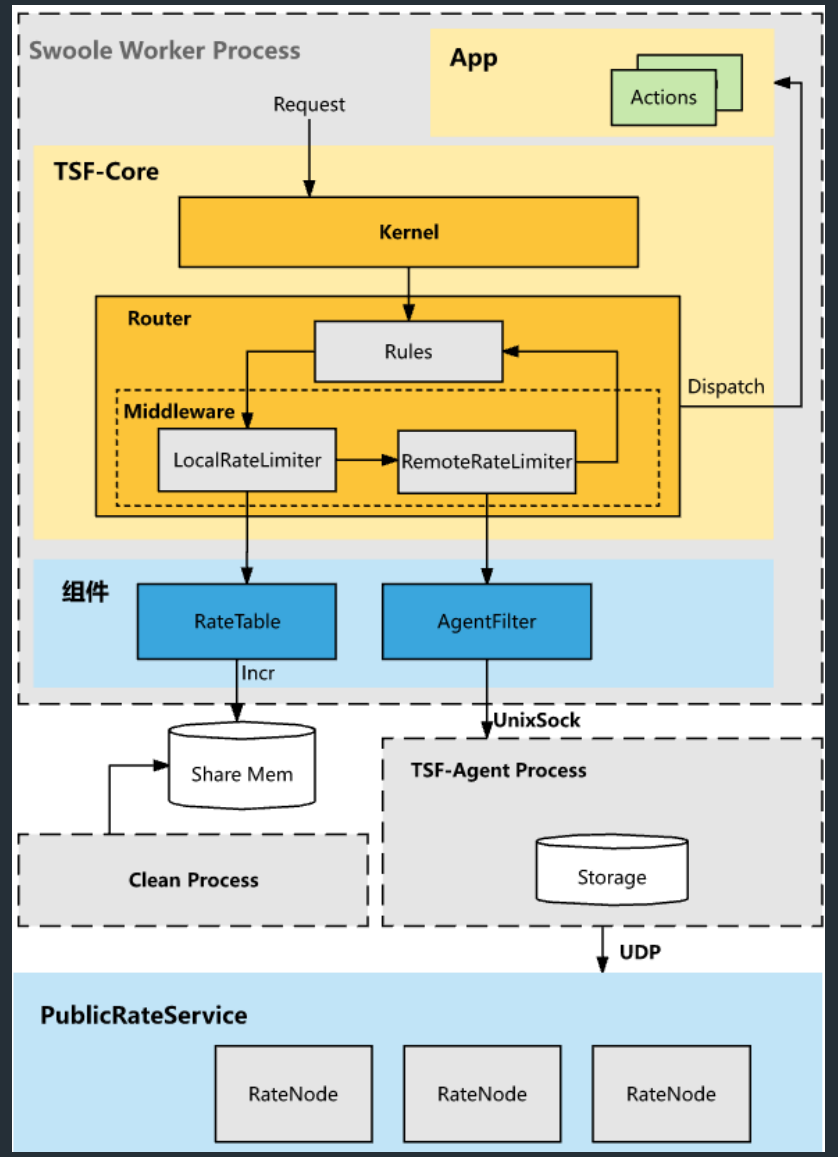
# 连接池

- 传统连接池为进程连接池，实现简单
  - 无法跨进程收敛连接
  - 服务定时重启带来短链攻击
- TSF3.0跨进程连接池，从worker层面收敛链接



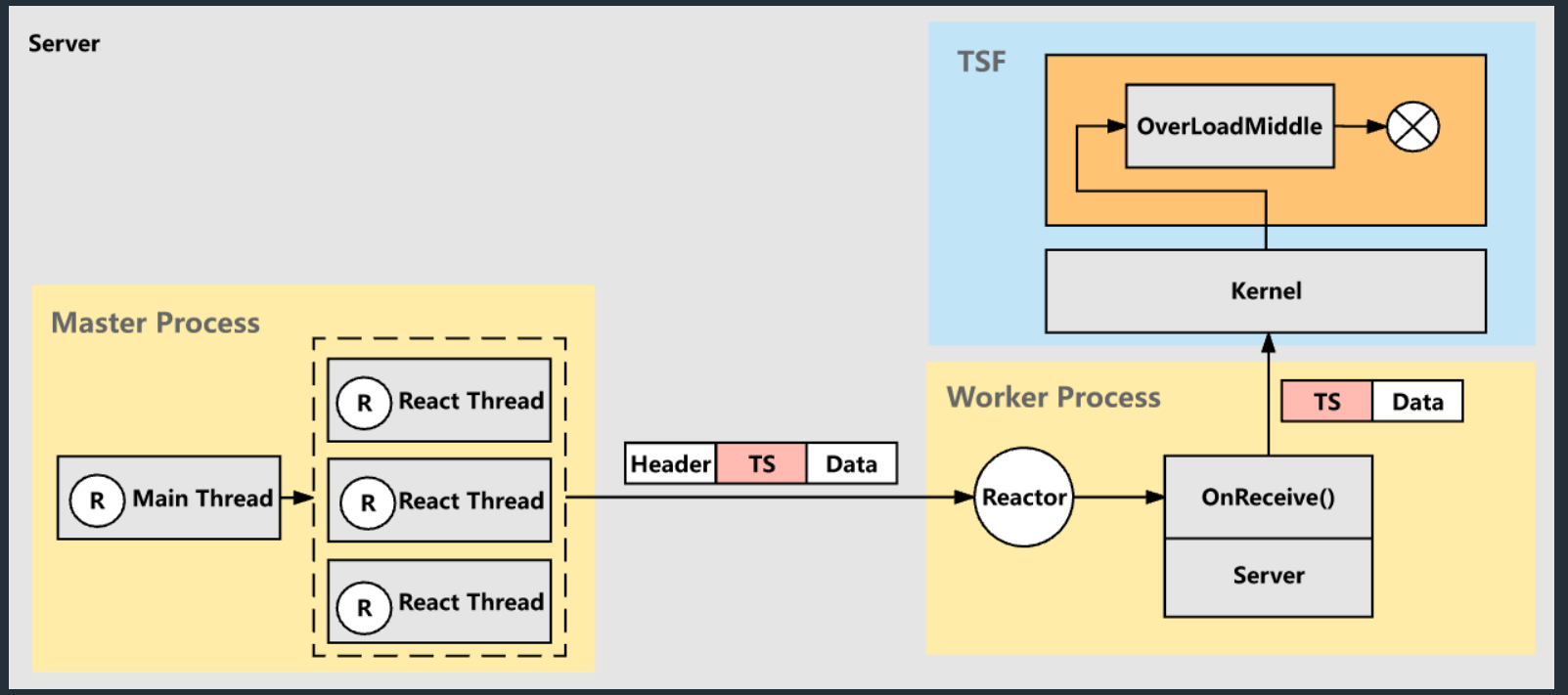
# 频率控制

- 频率控制保护服务在有效负载范围内
- 保护下游服务免受流量冲击
- TSF3.0提供频控保护组件
  - 单机频控：防止单机高负载导致成功率下降
  - 联机频控：从服务模块维度控制访问频次



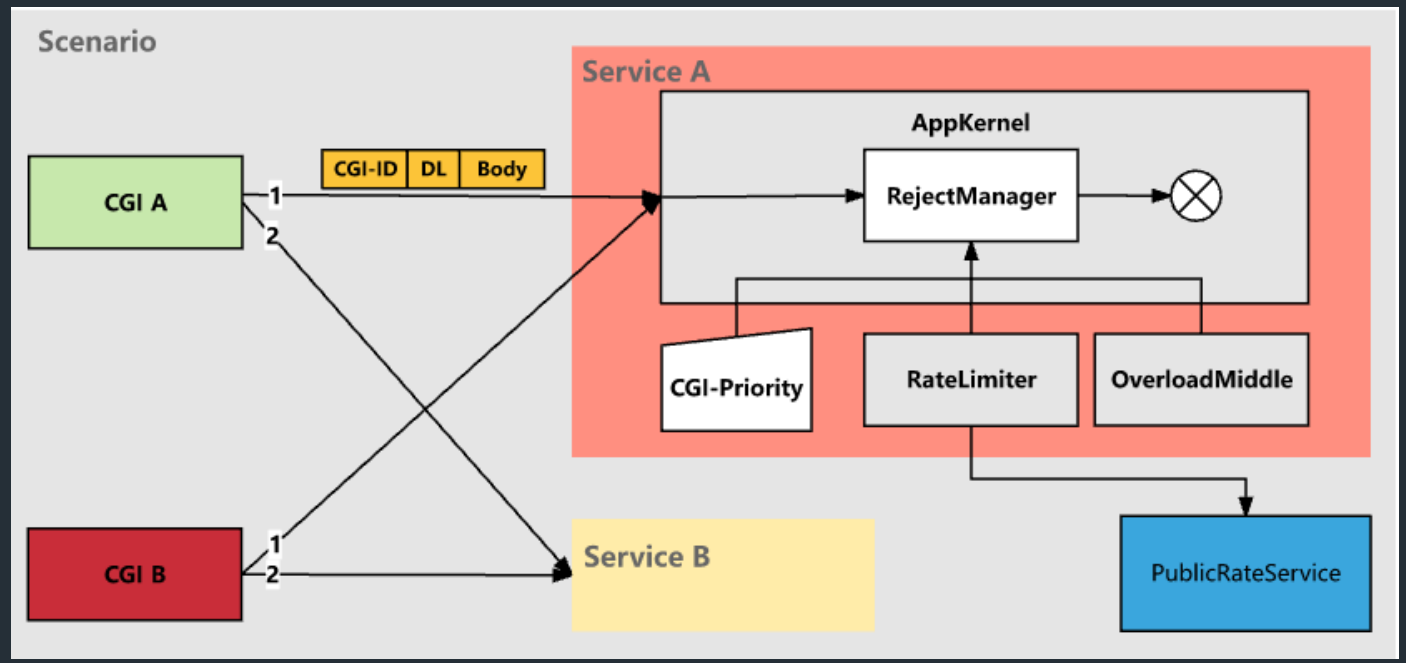
# 过载保护

- 超负载高流量会导致整体服务雪崩
- TSF提供单机和联机过载保护机制
- 单机过载保护：探测队列拥塞
  - Master进程标记请求收包时间
  - TSF根据收包时间戳判断队列拥塞情况



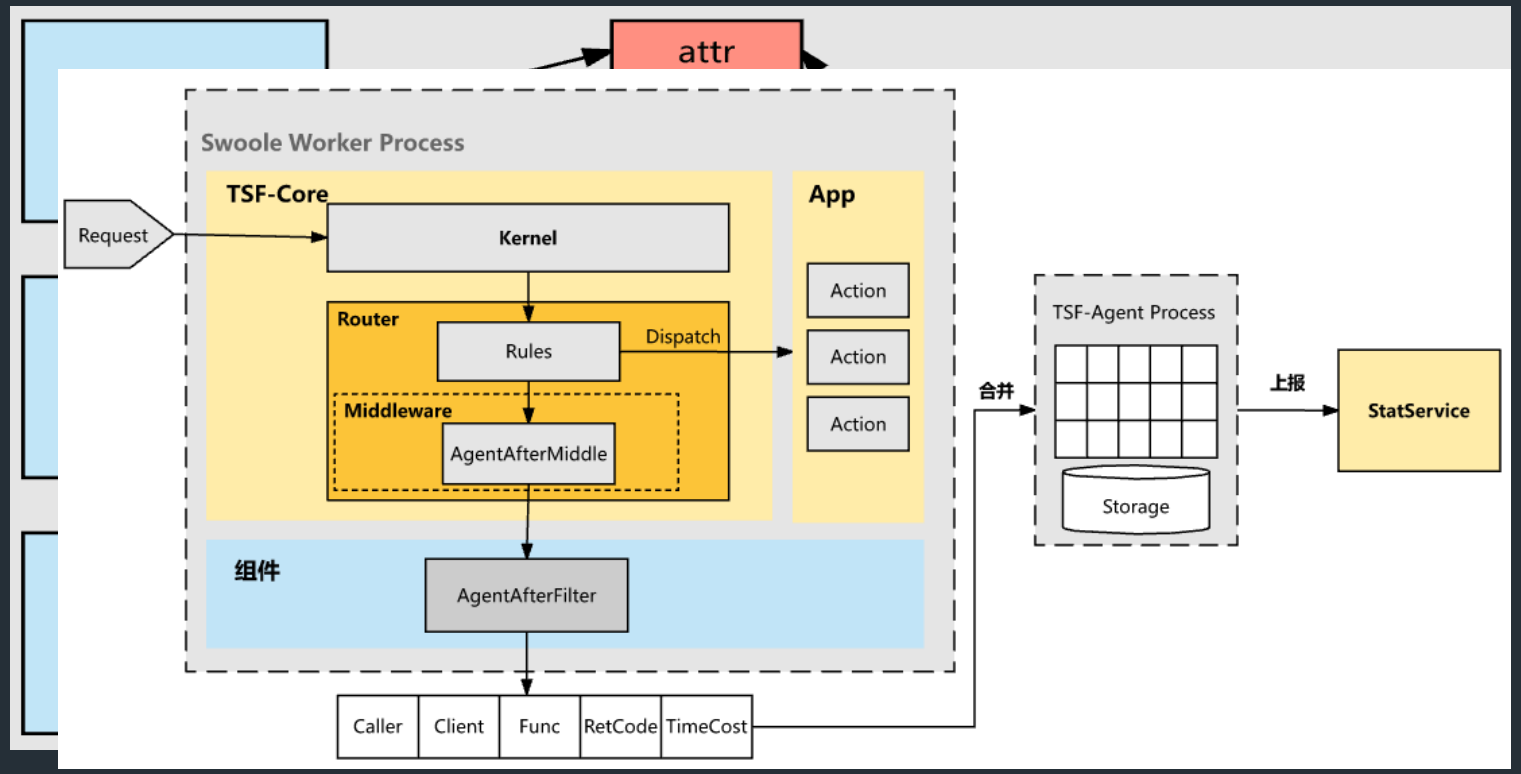
# 过载保护

- 单机服务过载后粗暴丢弃
  - 后端依赖过载会导致整个调用链成功率下降
  - 单个CGI过载会导致其他CGI成功率降低
- 联机过载保护：从全局保证损失最小化
  - CGI优先级划分，保证高优先级能力
  - CGI协议指定后端deadline，放弃超时请求



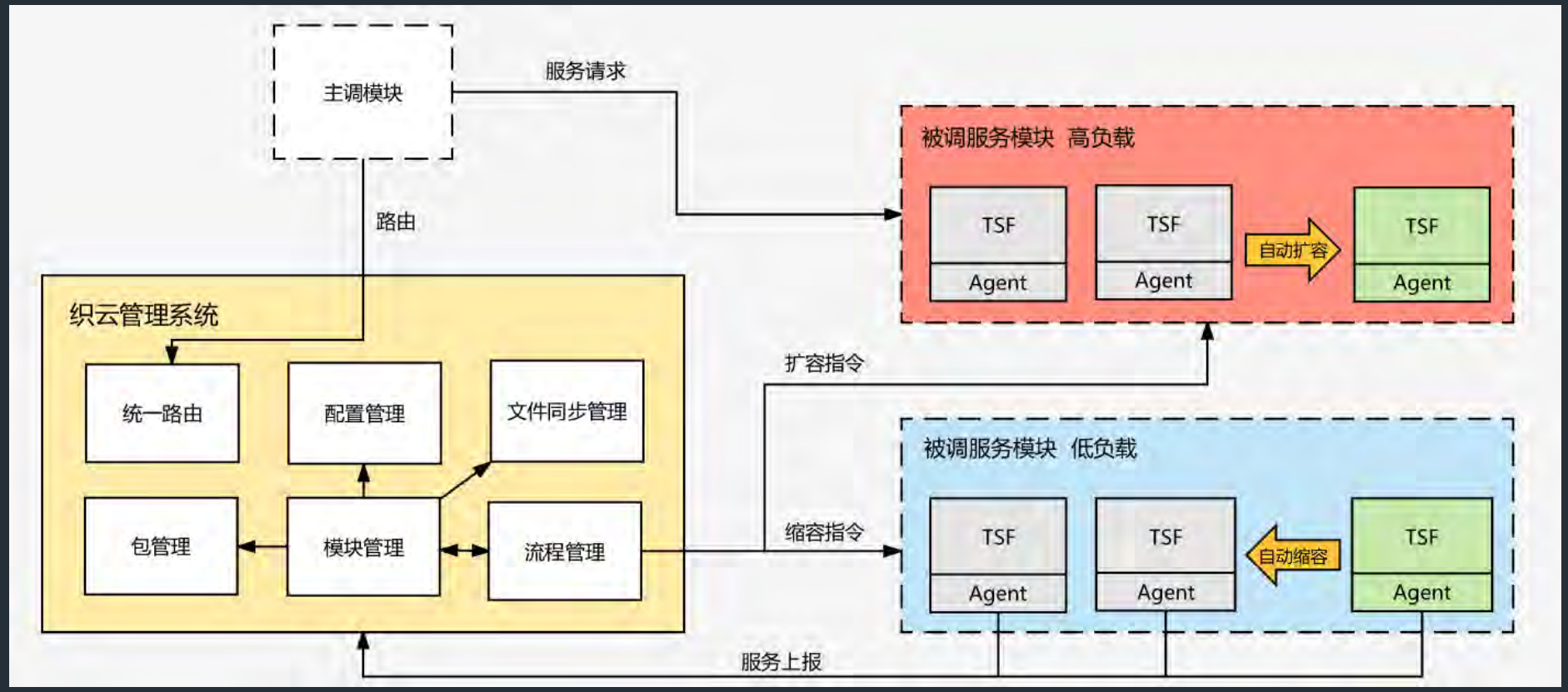
# 服务监控

- 服务实时可用性由调用链上服务健康度决定
- TSF提供服务统一上报能力
- Kernel收集服务统信息，写入共享内存
- TSF-Agent读取worker的信息，做统计合并后上报



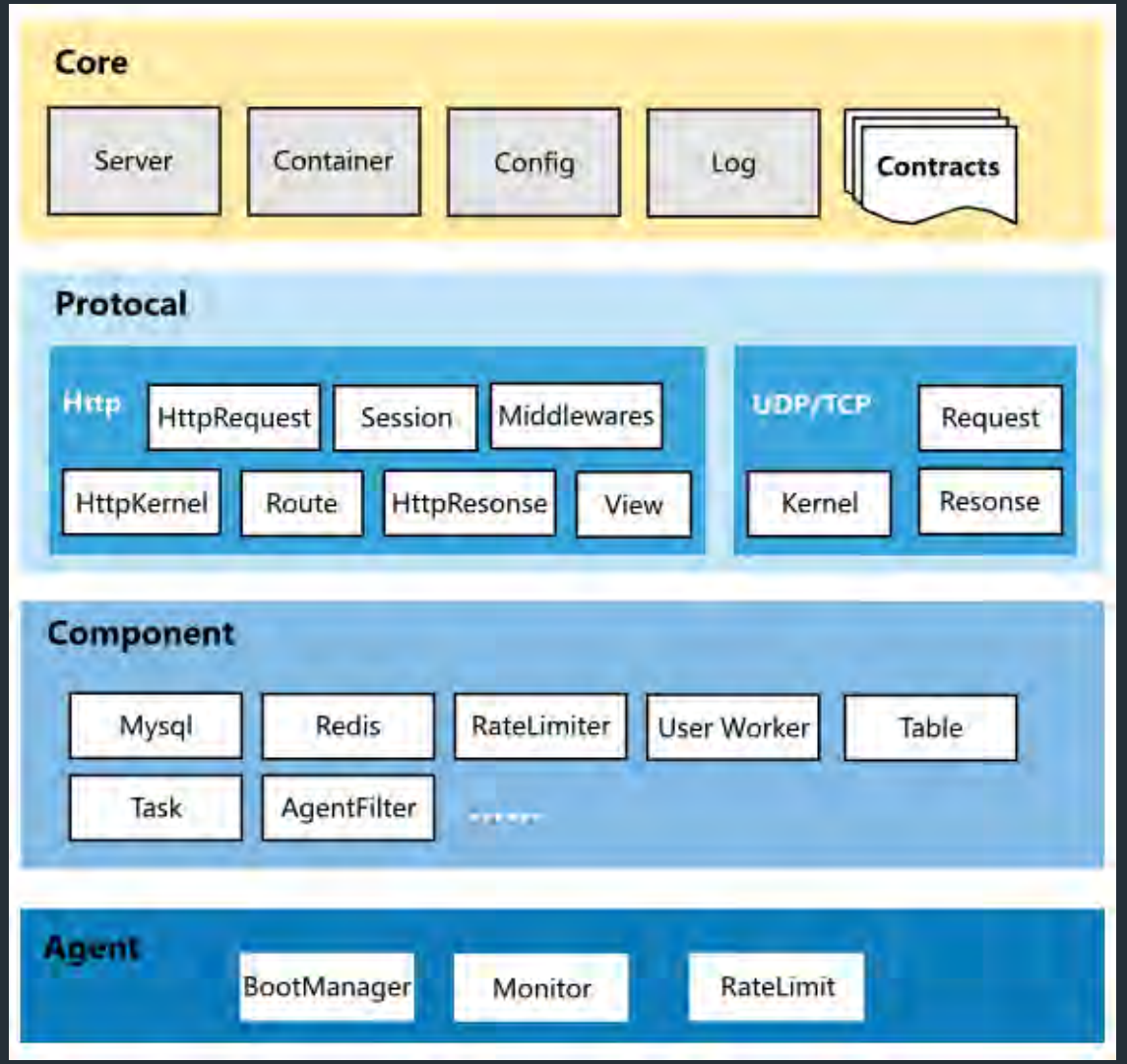
# 集群动态管理

- TSF服务抽象为模块提供集群服务
- 模块内配置、系统一致化
- 织云系统提供管理服务
  - 模块监控
  - 自动部署
  - 弹性伸缩
  - 服务发现



# TSF整体架构

- 核心层：抽象swoole基础网络能力
- 协议层：适配不同协议的网络接口
- 组件层：提供框架可扩展能力，封装具有独立能力的组件
- Agent服务：底层支持框架启停/监控/数据同步的能力



# TSF实践

## 连接池:

```
'mysqlpool' => [
    'conns' => [
        'target_mysql' => [
            'serverInfo' => [
              'host' => '','user' => '','password' => '',
              'database' => ''
            ],
            'maxSpareConns' => 10,
            'maxConns' => 20
        ]],
    'maxProxyConns' => 10,
    'maxSpareProxyConns' => 5
],


$conn = TSF\Pool\Factory::fetch(‘mysql‘,'target_mysql');
$res = $conn->query('select version()');
$conn->recycle();
```

## 单机频控:

```
Route::facade()->any('*', '*', [
    'include' => [
        'before' => ['TSF\Http\Middleware\RateLimiter'],
    ]
]);
```

## User Worker:

```
"TSF\\Component\\UserWorker\\UserWorkerManager" => [
    [
        "name" => "hippoworker",
        "num" => 1,
        "workerClass" => "\\App\\UserWorker\\HippoConsumer",
    ]
]
```

# 看点图文页优化

性

靠性

- 网络请求
- 终端预加
- 加载策略

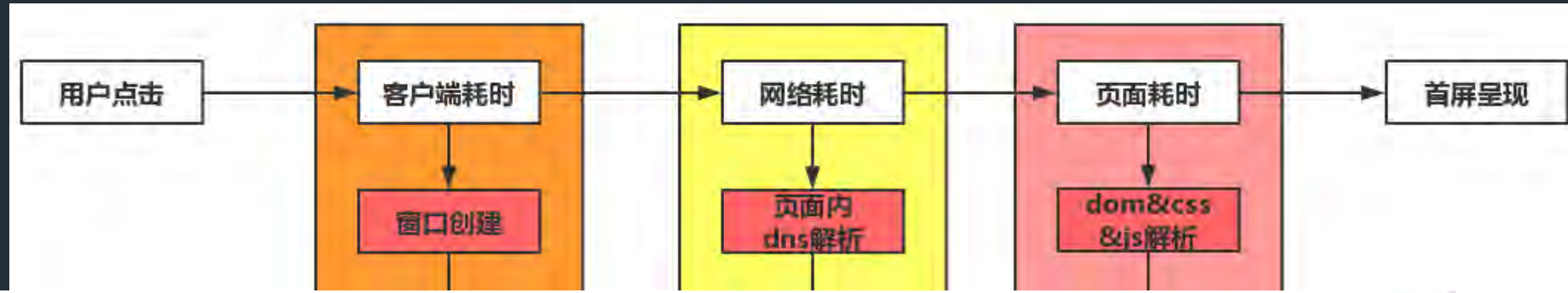
高可用

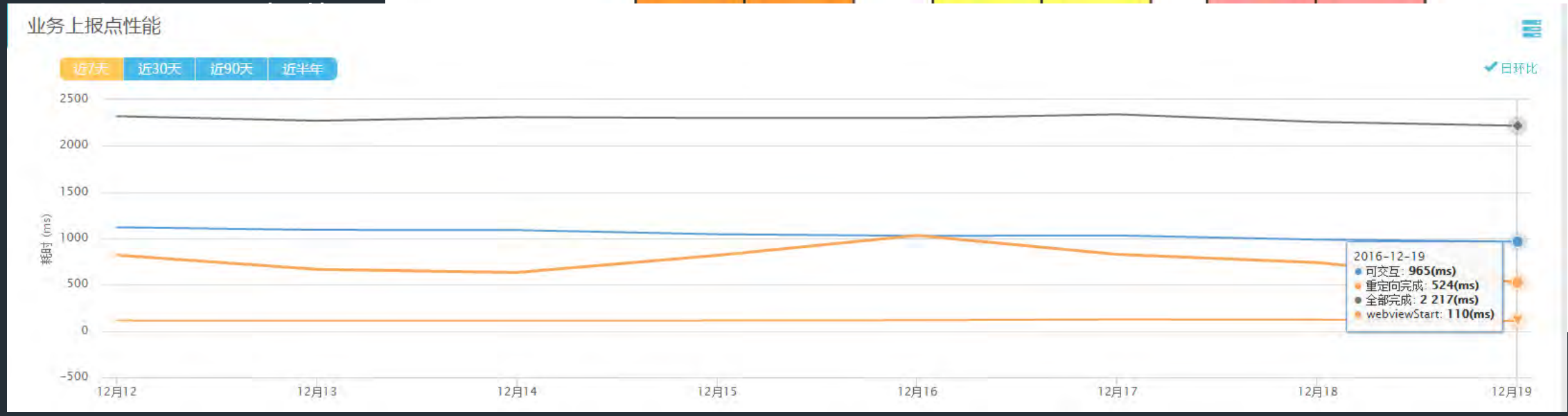
高可用

空


# 看点图文页优化

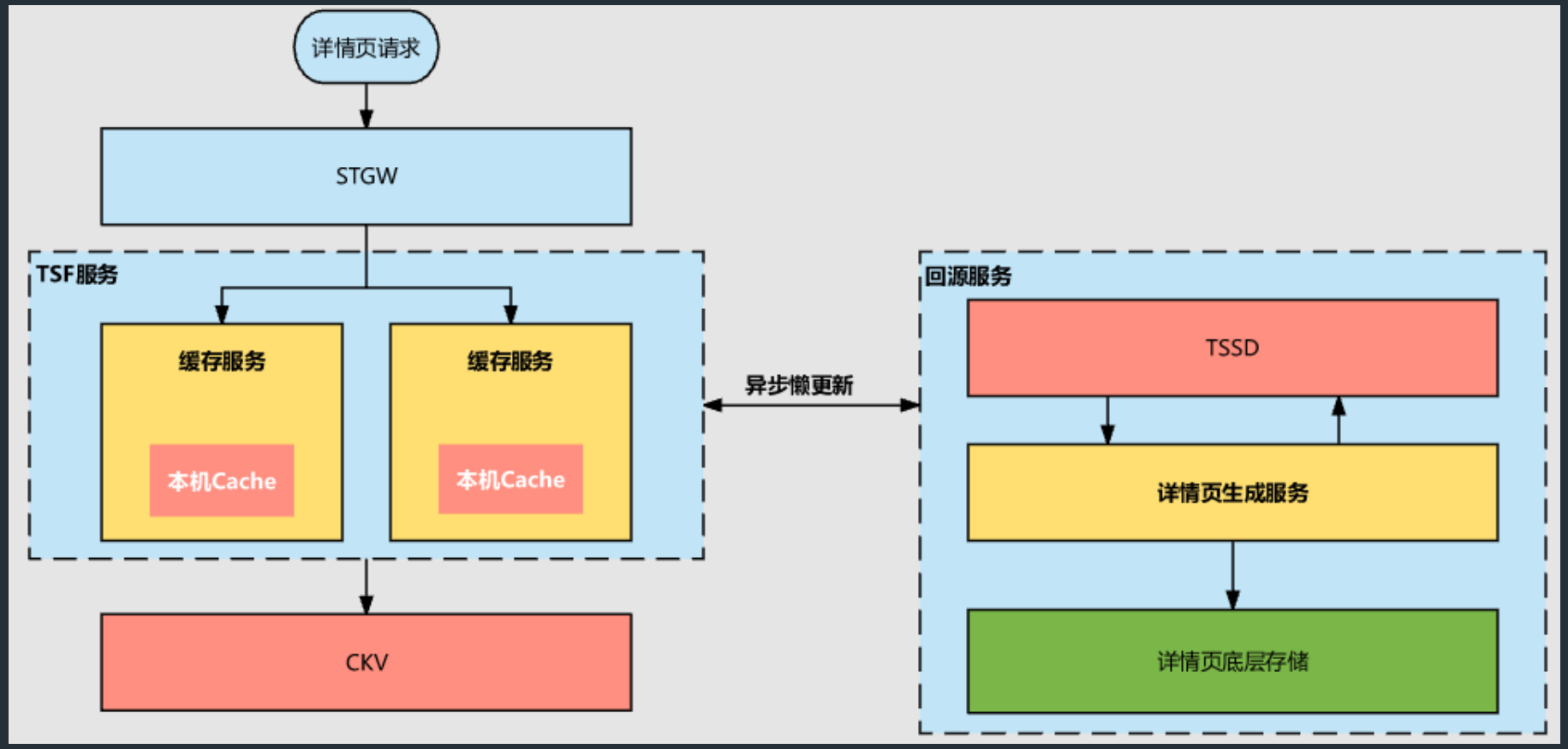
- 明确首屏直出目标：1s内（wifi/4g）
- 网络请求耗时全链路分析
- 网络请求路径优化
  - webview预启动
  - DNS预解析
  - 首屏sea.js内联
  - 首屏CSS内联



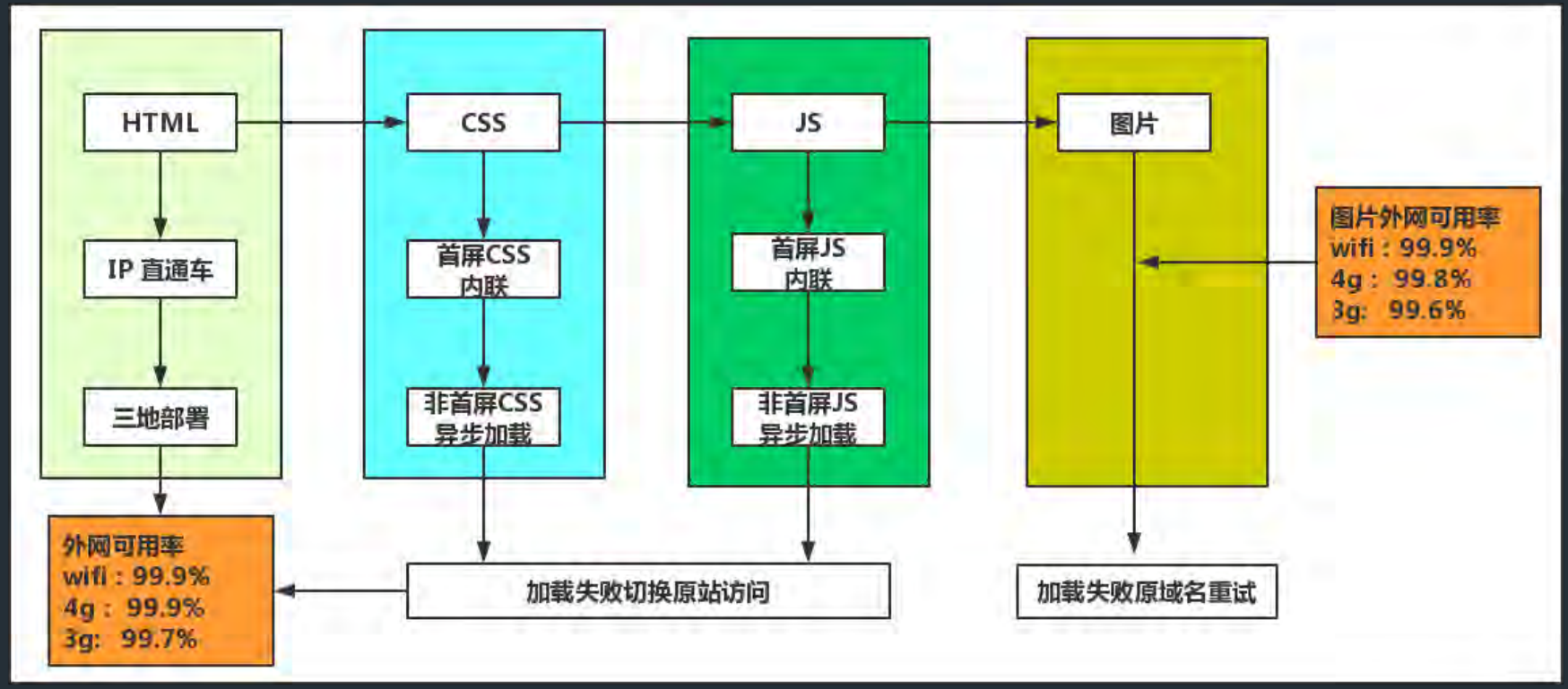
# 看点图文页优化

- 多级缓存实现详情页高性能直出
  - 本机缓存
  - 联机缓存
  - TSSD存储
- 缓存异步懒更新



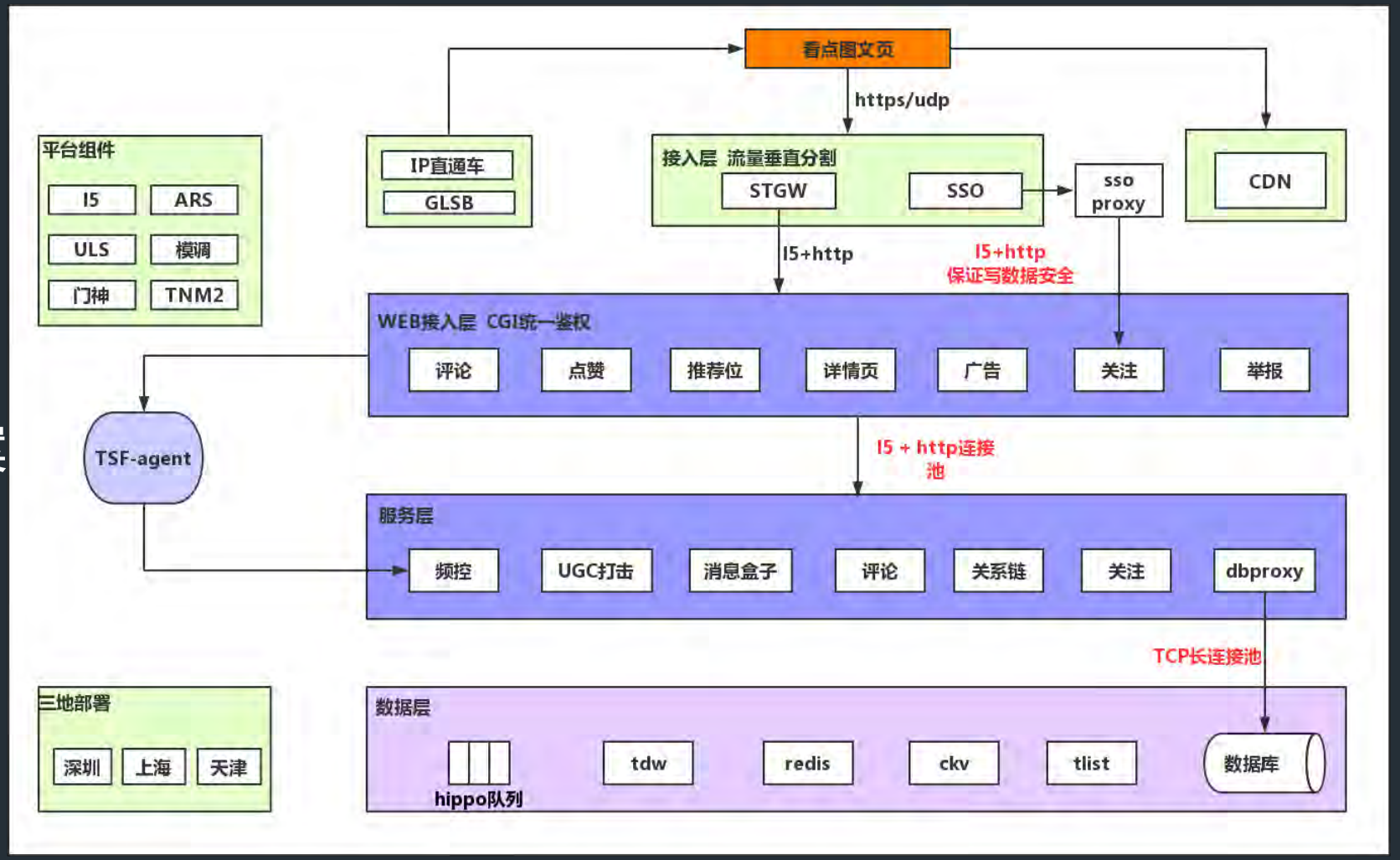
# 看点图文页优化

- web前台常见故障
  - 运营商劫持
  - cdn节点故障
  - cgi后端访问超时
- Web前台容灾方案
  - Html容灾
  - Js容灾
  - Css容灾
  - 图片容灾



# QQ看点图文页优化

- Web后台常见故障
  - 单机故障
  - 机房故障
  - 服务雪崩
  - 外部攻击
- Web后台台容灾方案
  - 故障容错
  - 多地部署
  - 过载保护
  - 频率控制




# Thanks