基于PHP7+Swoole4实现无侵入式自动化监控和链路追踪

韩天峰

自我介绍

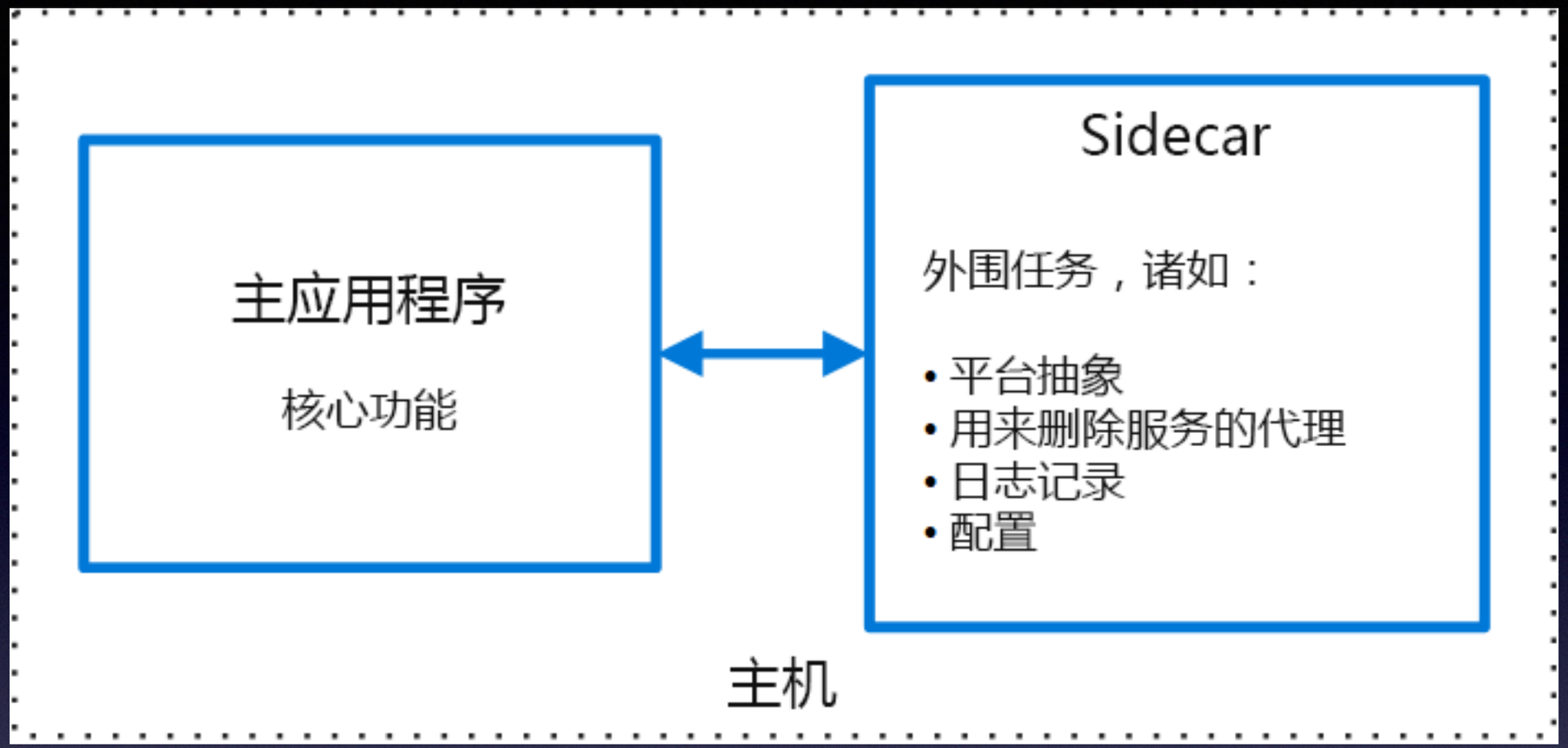
- Swoole 开源项目创始人
- 现任好未来网校首席架构师
- Weibo：@hantianfeng
- GitHub：https://github.com/matyhtf

ServiceMesh

SideCar

服务架构

- 服务发现
- 路由/负载均衡
- 熔断/降级
- 故障转移
- 流量控制
- 监控报警
- 日志
- 消息队列


主应用程序
核心功能
Sidecar
外围任务，诸如：
• 平台抽象
• 用来删除服务的代理
• 日志记录
• 配置
主机


业务逻辑
(Java/PHP/Go/Node/
Python)
http://127.0.0.1:9501/
服务
发现
监控
报警
路由
负载

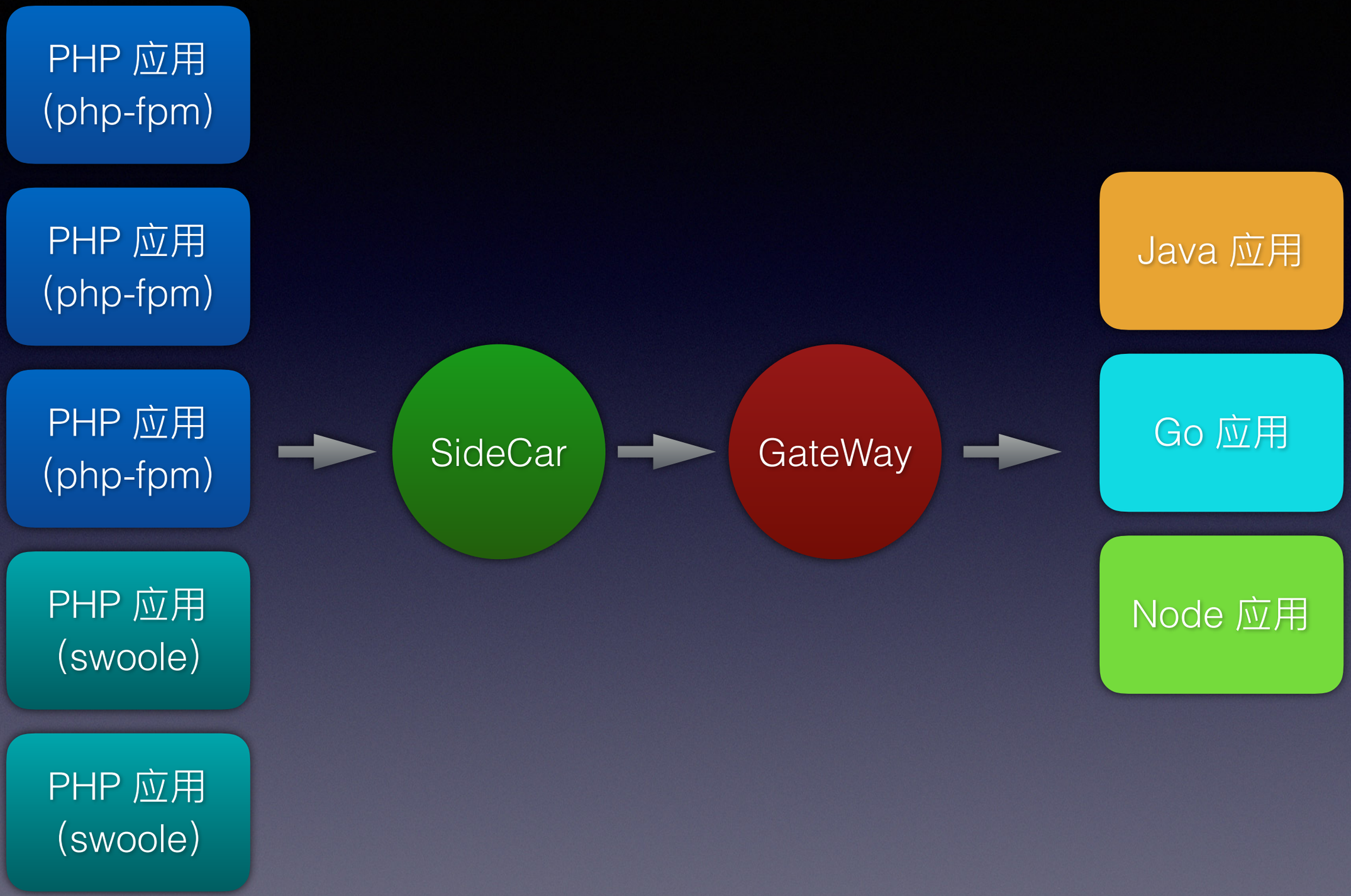
通信的成本

- 为什么不用 GateWay，而是使用 SideCar ？

- GateWay：10w/s，带宽：1G/s，响应：100us+
- SideCar ： 100w/s ， 带宽：40G/s，响应：1us+
- TimeWait：TCP短连接TimeWait导致端口不够用

PHP 服务治理难题

- ZooKeeper
- Kafka
- hbase
- 连接池
- 协议转换


PHP 应用
(php-fpm)
PHP 应用
(php-fpm)
PHP 应用
(php-fpm)
PHP 应用
(swoole)
PHP 应用
(swoole)
SideCar
GateWay
Java 应用
Go 应用
Node 应用

PHP7 (Type Hint)	可选类型，在大型项目使用 PHP 也可以保持很高的可维护性
PHP8 （JIT，2020）	密集计算使用 PHP 可以有接近 C/C++ 的性能。
Swoole4	长生命周期编程，Web 之外的网络服务器程序开发，异步 IO 和协程
PHP-FPM	短生命周期，热更新，无内存泄漏，编程调试非常简单，依然是最具效率的工具。

```php
class A
{
    public int $a;
    public float $b;
    public TestClass $c;
    public array $d;

    function test(int $a, float $b, TestClass $c, array $d)
    {
        $this->a = $a;
        $this->b = $b;
        $this->c = $c;
        $this->d = $d;
    }
}
```

```php
$object = new A;
//正确
$object->test(1234, 56.78, new ClassA, ["a", "b"]);
//错误
$object->test(0.1, "hello", new StdClass, null);
```

```php
Swoole\Runtime::enableCoroutine();
$n = 1024;
while($n) {
    go(function () {
        file_get_contents("https://www.baidu.com/");
    });
}
```

```php
$n = 1024;
while($n) {
    go(function () {
        $redis = new redis;
        $redis->connect("127.0.0.1");

        for($i = 0; $i < 100; $i++) {
            $data = $redis->get("key");
        }

        $db = new PDO("$dbms:host=$host;dbname=$database", $user, $passwd);

        for($i = 0; $i < 100; $i++) {
            $data = $db->query("select * from test_table")->fetchAll();
        }
    });
}
```

应用监控

接口调用

- Redis/Memcache
- MySQL/MongoDB/ElasticSearch
- Http/RPC
- TCP Socket

性能指标	实例	作用
请求量	60000/分钟	计算 QPS，得到应用的负载状况
成功率	99.2%	存在失败，得到服务稳定性状况
返回码	200/502/403	分析错误原因，快速定位问题
响应时间	100ms	定位慢请求，进行优化
流量	2M/s	入口、出口流量状况

PHP 探针技术

- 替换 php function 指针，插入埋点代码
- 函数列表：EG(function_table)
- 类列表：EG(class_table)

```
1. zend_function* fbc = zend_hash_str_find_ptr(EG(function_table), ZEND_STRL("curl_init"));
```

```
union _zend_function {
    zend_uchar type;    /* MUST be the first element of this struct! */
    uint32_t   quick_arg_flags;

    struct {
        zend_uchar type;  /* never used */
        zend_uchar arg_flags[3]; /* bitset of arg_info.pass_by_reference */
        uint32_t fn_flags;
        zend_string *function_name;
        zend_class_entry *scope;
        zend_function *prototype;
        uint32_t num_args;
        uint32_t required_num_args;
        zend_arg_info *arg_info;
    } common;

    zend_op_array op_array;
    zend_internal_function internal_function;
};
```

```c
typedef struct _zend_internal_function {
    /* Common elements */
    zend_uchar type;
    zend_uchar arg_flags[3]; /* bitset of arg_info.pass_by_reference */
    uint32_t fn_flags;
    zend_string* function_name;
    zend_class_entry *scope;
    zend_function *prototype;
    uint32_t num_args;
    uint32_t required_num_args;
    zend_internal_arg_info *arg_info;
    /* END of common elements */

    zif_handler handler;
    struct _zend_module_entry *module;
    void *reserved[ZEND_MAX_RESERVED_RESOURCES];
} zend_internal_function;
```

```c
/* zend_internal_function_handler */
typedef void (ZEND_FASTCALL *zif_handler)(INTERNAL_FUNCTION_PARAMETERS);
```

```c
#define INTERNAL_FUNCTION_PARAMETERS zend_execute_data *execute_data, zval *return_value
```

```c
struct _zend_execute_data {
    const zend_op       *opline;           /* executed opline                */
    zend_execute_data   *call;             /* current call                   */
    zval                *return_value;
    zend_function       *func;             /* executed function              */
    zval                 This;             /* this + call_info + num_args    */
    zend_execute_data   *prev_execute_data;
    zend_array          *symbol_table;
#if ZEND_EX_USE_RUN_TIME_CACHE
    void               **run_time_cache;   /* cache op_array->run_time_cache */
#endif
};
```

```c
  zval *param_ptr = ZEND_CALL_ARG(EG(current_execute_data), 1);
  int arg_count = ZEND_CALL_NUM_ARGS(EG(current_execute_data));
```

- curl_init, curl_exec
- mysqli::query
- PDO::query
- redis::get/set/…
- memcache::get
- MongoDB
- ElasticSearch

start

函数调用

时间

return

- 返回值，成功失败
- 实例的 IP:PORT
- 错误码
- 请求响应字节数

PHP 探针

- 无侵入，无需修改业务代码
- 自动获取接口调用信息
- 自动化监控、统计、上报
- 链路跟踪
- 接口依赖关系

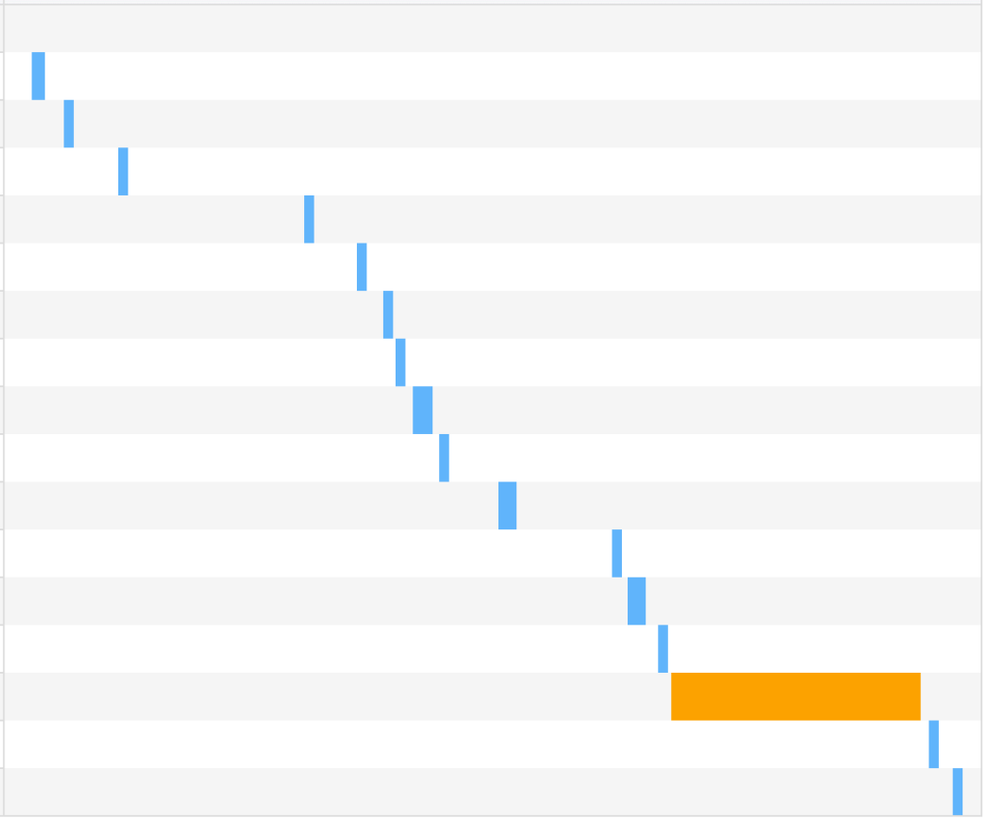
接口汇总

接口名称 | 过滤 | 所有接口 | 1: CGI | 所有实例 | 2019-03-21

接口名称	所属实例	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
CGI /api/getLoginInfo	-	1,453	1,453	0	100%	2080ms	0ms	15.53ms	0ms
CGI /	-	152	152	0	100%	2076ms	18ms	185.78ms	0ms
CGI /wiki/page/p-server.html	-	95	95	0	100%	7615ms	18ms	300.2ms	0ms
CGI /wiki/diff/	-	66	66	0	100%	8190ms	11ms	559.98ms	0ms
CGI /wiki/page/1.html	-	63	63	0	100%	6213ms	17ms	302.13ms	0ms
CGI /wiki/version/	-	47	47	0	100%	5730ms	10ms	596.38ms	0ms
CGI /wiki/page/56.html	-	42	42	0	100%	908ms	19ms	256.19ms	0ms
CGI /wiki/page/397.html	-	41	41	0	100%	8423ms	22ms	520.73ms	0ms
CGI /wiki/page/15.html	-	41	41	0	100%	2233ms	18ms	291.27ms	0ms
CGI /wiki/page/326.html	-	33	33	0	100%	1540ms	23ms	311ms	0ms
CGI /wiki/search/	-	31	31	0	100%	1530ms	3ms	309.42ms	0ms

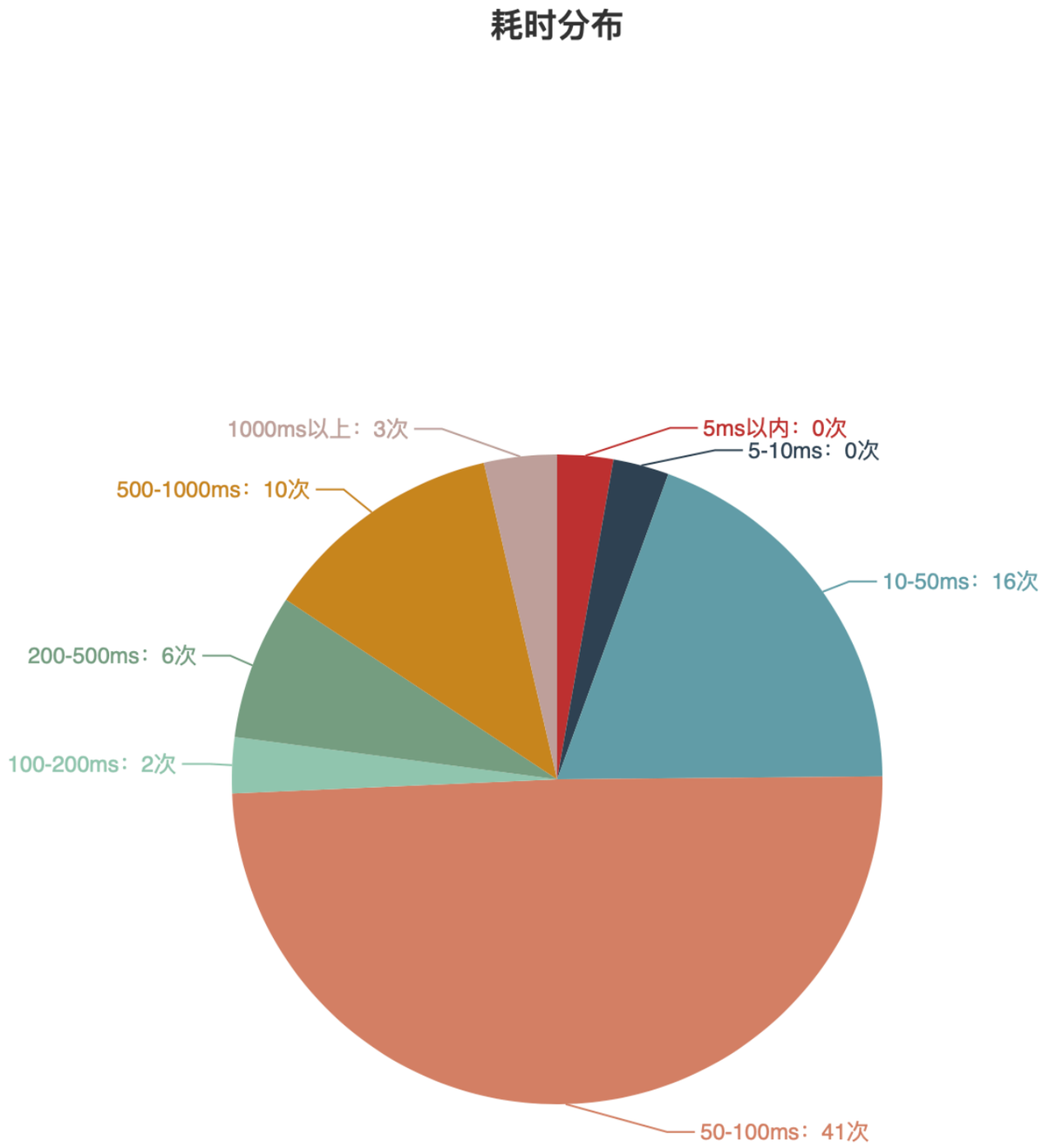
接口名称	所属实例	调用次数	成功次数	失败次数	成功率	响应最大值	响应最小值	平均响应时间	失败平均时间
MySQL Wiki.php:App\\Content::getTree3::query:select	127.0.0.1:3306:wiki4swoole	10,930	10,930	0	100%	1143ms	0ms	2.46ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getPageInfo->query:select	127.0.0.1:3306:wiki4swoole	5,636	5,636	0	100%	1810ms	0ms	4.36ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getComments->query:select	127.0.0.1:3306:wiki4swoole	2,953	2,953	0	100%	1042ms	0ms	11.11ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getProjectInfo->query:select	127.0.0.1:3306:wiki4swoole	1,874	1,874	0	100%	978ms	0ms	10.15ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getPageInfo->query:update	127.0.0.1:3306:wiki4swoole	1,871	1,871	0	100%	1114ms	0ms	9.64ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getProjectLinks->query:select	127.0.0.1:3306:wiki4swoole	1,860	1,860	0	100%	932ms	0ms	8.23ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getReadCount->query:select	127.0.0.1:3306:wiki4swoole	1,696	1,696	0	100%	1180ms	0ms	10.41ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getPageInfo->mysqli_connect	127.0.0.1:3306:wiki4swoole	1,682	1,682	0	100%	2033ms	0ms	17.08ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getProjectInfo->mysqli_connect	127.0.0.1:3306:wiki4swoole	187	187	0	100%	1133ms	0ms	68.51ms	0ms
MySQL Wiki.php:App\\Controller\\Wiki->getDiff->query:select	127.0.0.1:3306:wiki4swoole	184	184	0	100%	1160ms	0ms	57.09ms	0ms

耗时: 31.848 ms， 包含span 17个， 子调用0个

Service	Name	Duration	
default	GET http://wiki.swoole.com/wiki/page/41.html	31.848ms	
default	MySQL mysqli_connect true	0.409ms	
default	MySQL query select * from wiki_tree where id ='41' limit 1	0.283ms	
default	MySQL query select * from wiki_content where id ='41' limit 1	0.18ms	
default	MySQL query update `wiki_content` set `read_count`='155380' …	0.18ms	
default	MySQL query select * from wiki_tree where id ='41' limit 1	0.159ms	
default	MySQL query select * from wiki_project where id ='1' limit 1	0.152ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.143ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.64ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.134ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.593ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.136ms	
default	MySQL query select id,text,link,pid,order_by_time,publish from …	0.586ms	
default	MySQL query select * from `wiki_project` where `id` IN (15) ord…	0.149ms	
default	MySQL query select * from `duoshuo_posts` where thread_key…	8.119ms	
default	MySQL query select id, nickname, avatar from `user_login` whe…	0.153ms	
default	MySQL query select * from wiki_content where id ='41' limit 1	0.154ms	

耗时分布

- 5ms以内：0次
- 5-10ms：0次
- 10-50ms：16次
- 50-100ms：41次
- 100-200ms：2次
- 200-500ms：6次
- 500-1000ms：10次
- 1000ms以上：3次



Q & A