百万并发下PHP协程+非阻塞框架设计实践

@代维

PHPCON
www.phpconchina.com

About me

- 2014年加入PHP官方PECL开发组
- Yac Windows版本作者
- Memcache、Redis等扩展PHP7版本贡献者
- 现就职于有赞

1. Why & What
2. 协程 in Zan
3. Zan框架设计

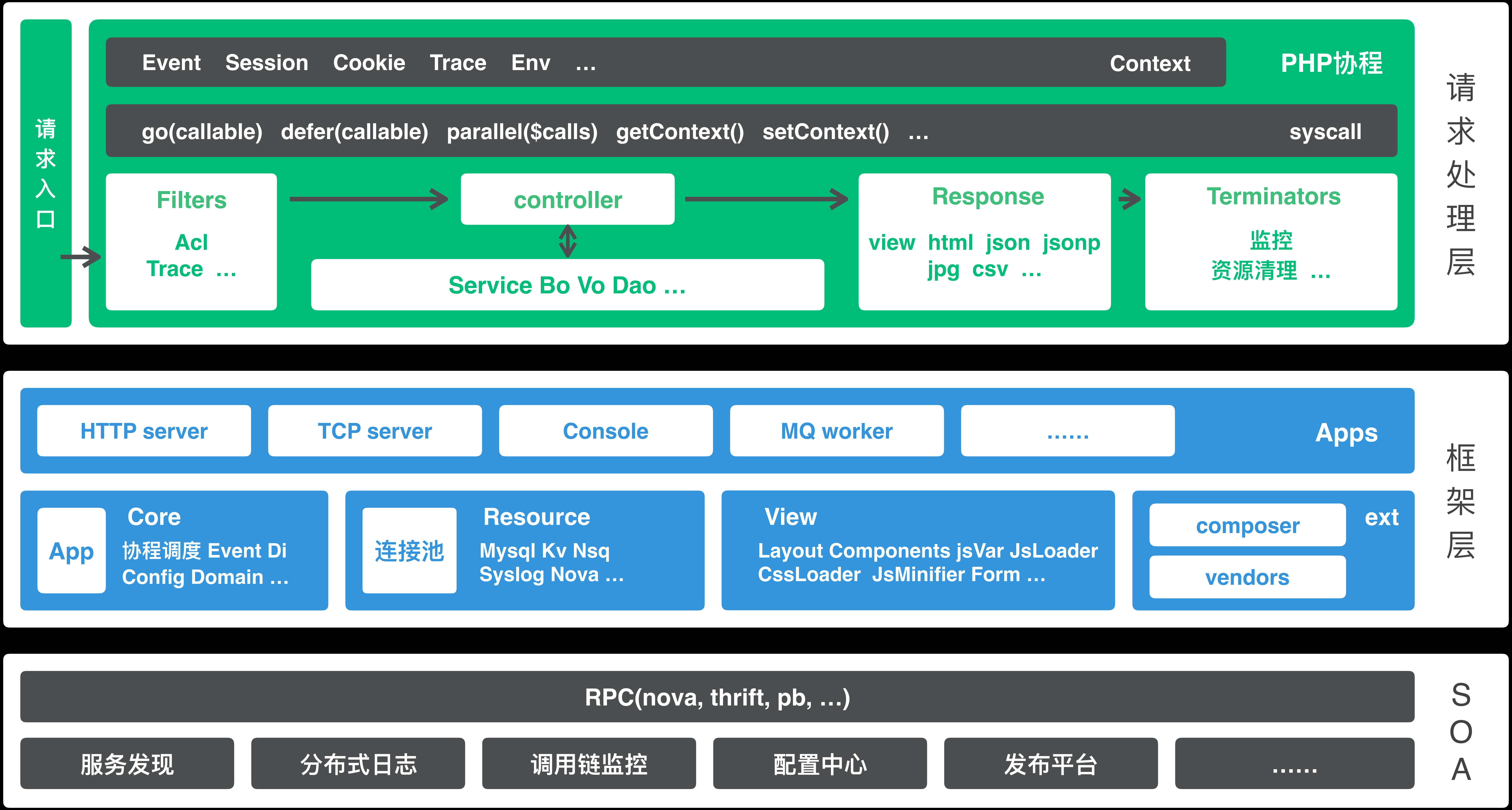
Part 1

Why & What

Why?!

- 连接数
- 并发
- 性能

What is Zan?

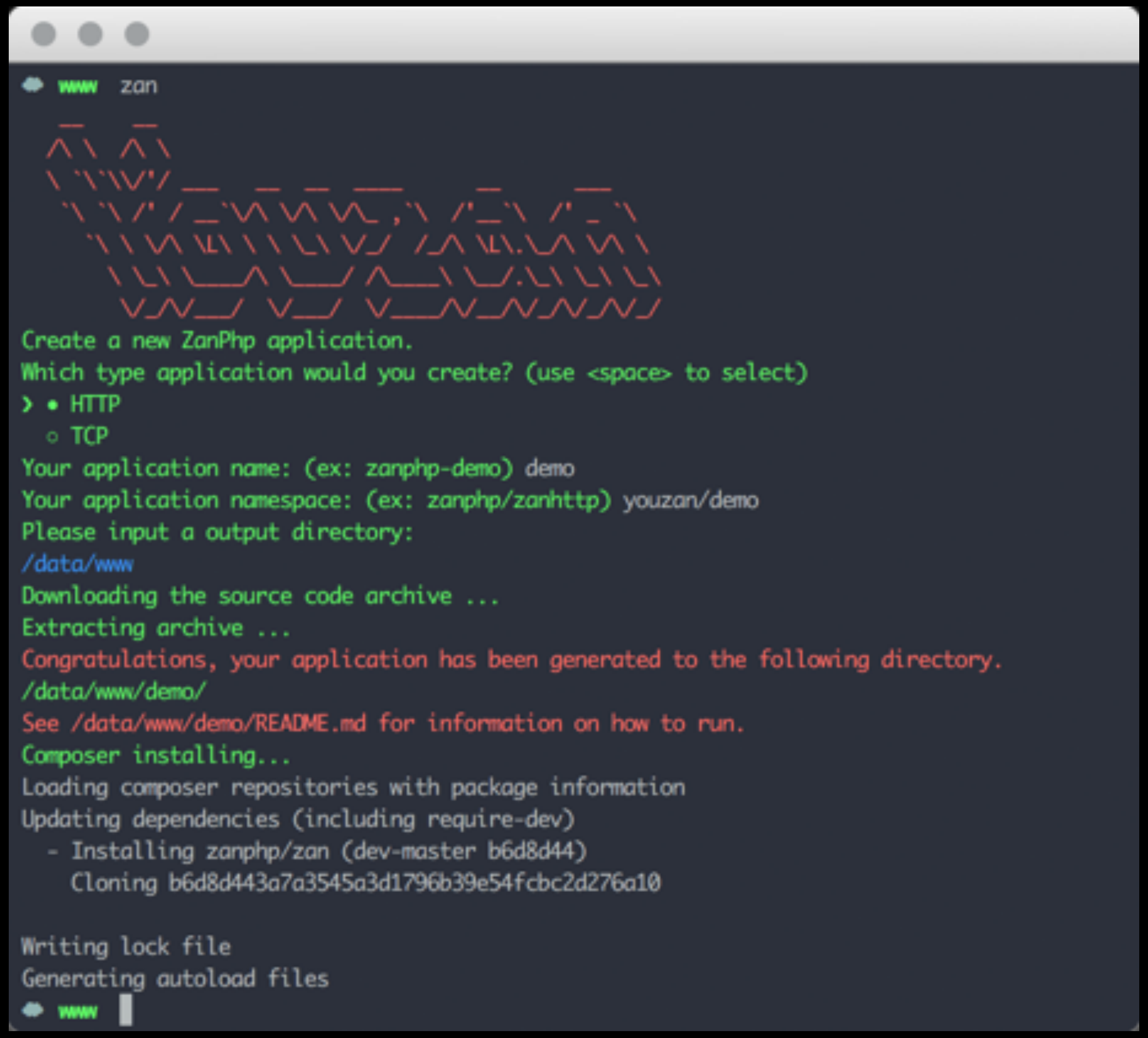


请求入口
Event Session Cookie Trace Env … Context
PHP协程
go(callable) defer(callable) parallel($calls) getContext() setContext() … syscall
Filters
Acl
Trace …
controller
Service Bo Vo Dao …
Response
view html json jsonp
jpg csv …
Terminators
监控
资源清理 …
请求处理层
HTTP server
TCP server
Console
MQ worker
……
Apps
App
Core
协程调度 Event Di
Config Domain …
连接池
Resource
Mysql Kv Nsq
Syslog Nova …
View
Layout Components jsVar JsLoader
CssLoader JsMinifier Form …
composer
vendors
ext
框架层
RPC(nova, thrift, pb, …)
服务发现
分布式日志
调用链监控
配置中心
发布平台
……
SOA

一分钟起步

1 `# composer global require youzan/zan-installer`

2 `# zan`

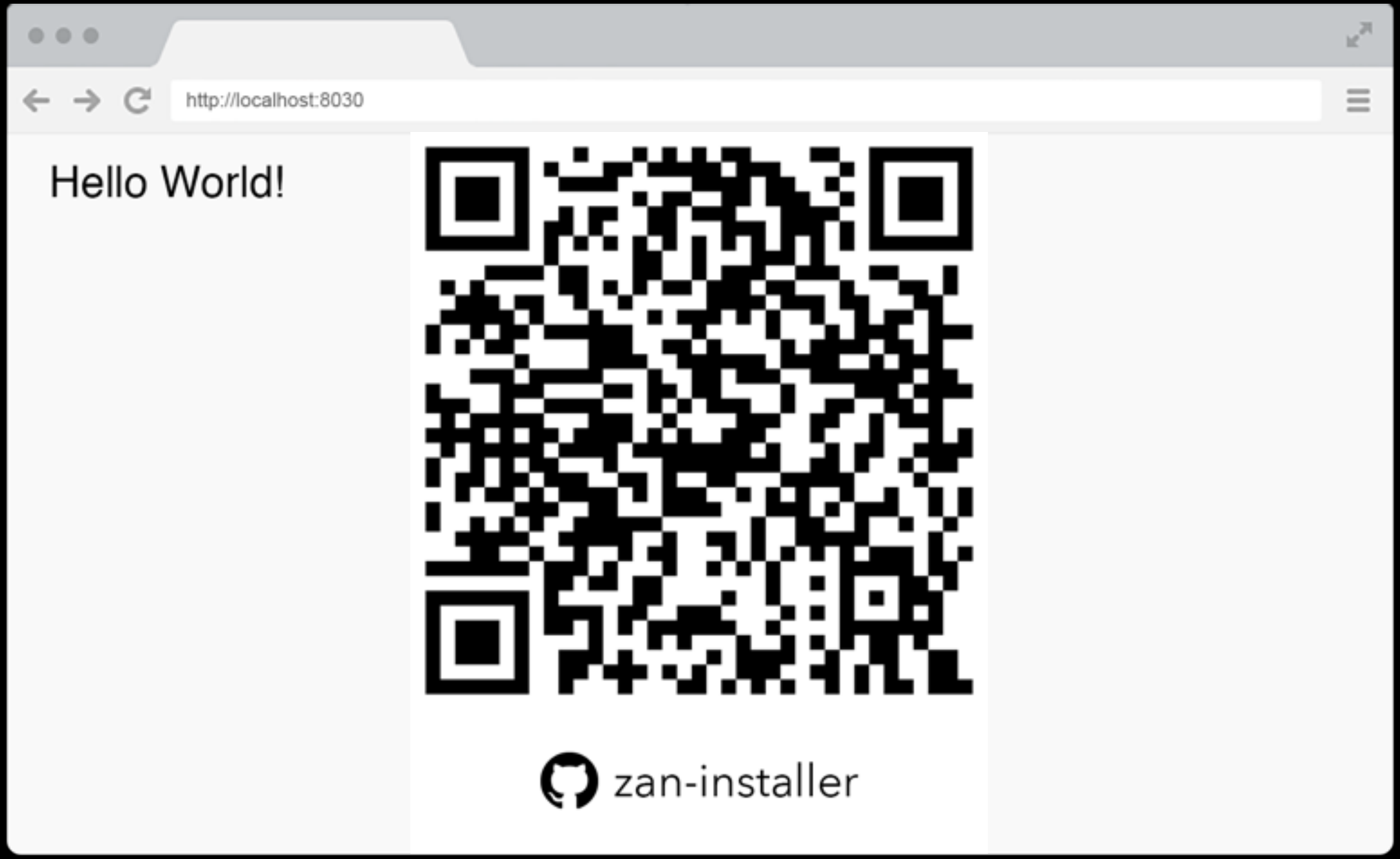


```
www  zan
  __    __
 /\ \  /\ \
 \ `\`\/'/ ___   __  __  ____     __     ___
  `\ `\ /' / __`\/\ \/\ \/\_ ,`\  /'__`\  /' _ `\
    `\ \ \/\ \L\ \ \ \_\ \/_/  /_/\ \L\.\_/\ \/\ \
      \ \_\ \____/\ \____/ /\____\ \__/.\_\ \_\ \_\
       \/_/\/___/  \/___/  \/____/\/__/\/_/\/_/\/_/
Create a new ZanPhp application.
Which type application would you create? (use <space> to select)
> • HTTP
  ○ TCP
Your application name: (ex: zanphp-demo) demo
Your application namespace: (ex: zanphp/zanhttp) youzan/demo
Please input a output directory:
/data/www
Downloading the source code archive ...
Extracting archive ...
Congratulations, your application has been generated to the following directory.
/data/www/demo/
See /data/www/demo/README.md for information on how to run.
Composer installing...
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing zanphp/zan (dev-master b6d8d44)
    Cloning b6d8d443a7a3545a3d1796b39e54fcbc2d276a10

Writing lock file
Generating autoload files
www  
```

3 `# bin/httpd`

4 立即打开浏览器访问吧!



http://localhost:8030

Hello World!

zan-installer

Part 2

协程 in Zan

Web IO模型

Web服务IO模型

PHP-FPM：	请求 — 进程
Java:	请求 — 线程
Golang:	请求 — 协程
Node.js:	请求 — Callback

Zan IO 模型:

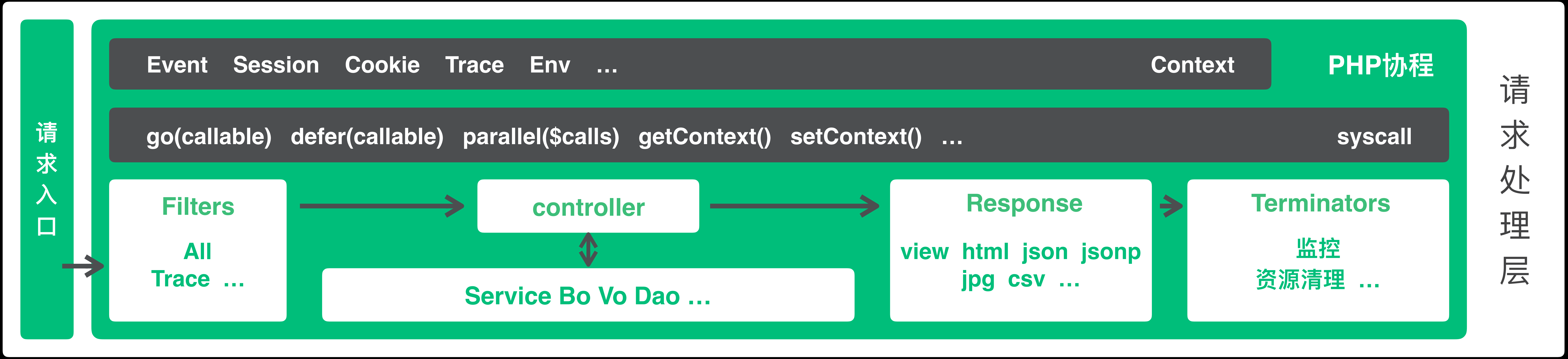
Just like Golang

基于Swoole非阻塞Callback模式

With (PHP + yield)

实现了独立堆栈

简单的并行实现



callback vs 协程

```php
<?php

mysql_async_query1($param1, function($res1) {
    mysql_async_query2($res1, function($res2) {
        mysql_async_query3($res2, function($res3) {
            mysql_async_query4($res3, function($res4) {
                mysql_async_query5($res4, function($res5) {
                    mysql_async_query6($res5, function($res6) {
                        mysql_async_query7($res6, function($res7) {
                            //.......
                        });
                    });
                });
            });
        });
    });
});
```

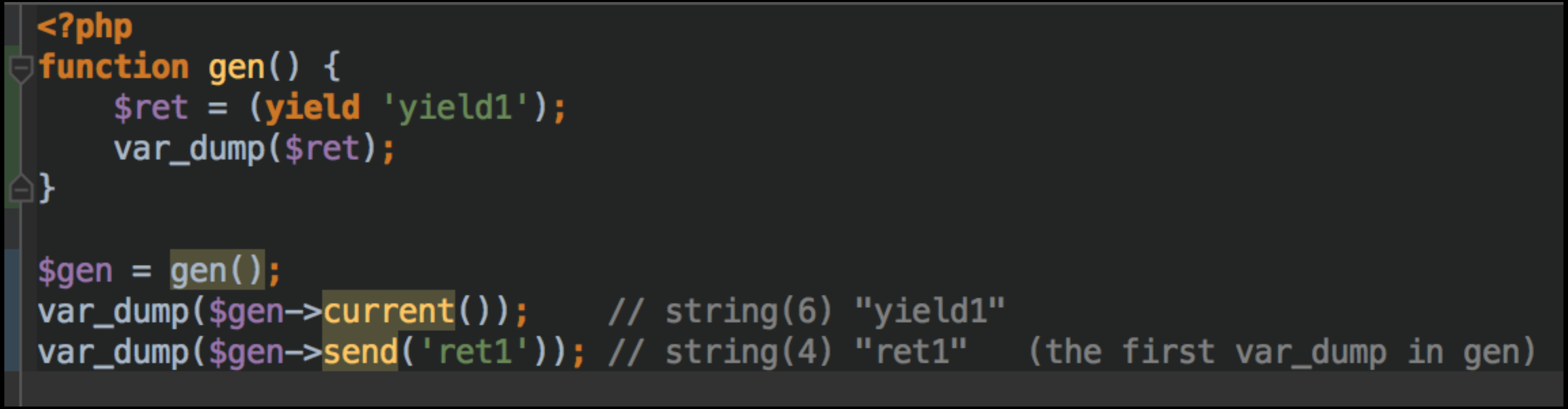
```php
<?php

$res1 = (yield mysql_async_query1($param1));
$res2 = (yield mysql_async_query2($res1));
$res3 = (yield mysql_async_query3($res2));
$res4 = (yield mysql_async_query4($res3));
$res5 = (yield mysql_async_query5($res4));
$res6 = (yield mysql_async_query6($res5));
$res7 = (yield mysql_async_query7($res6));
//....
```

我们不赞成用异步回调的方式去做功能开发，传统的PHP同步方式实现功能和逻辑是最简单的，也是最佳的方案。像node.js这样到处callback，只是牺牲可维护性和开发效率。

PHP协程

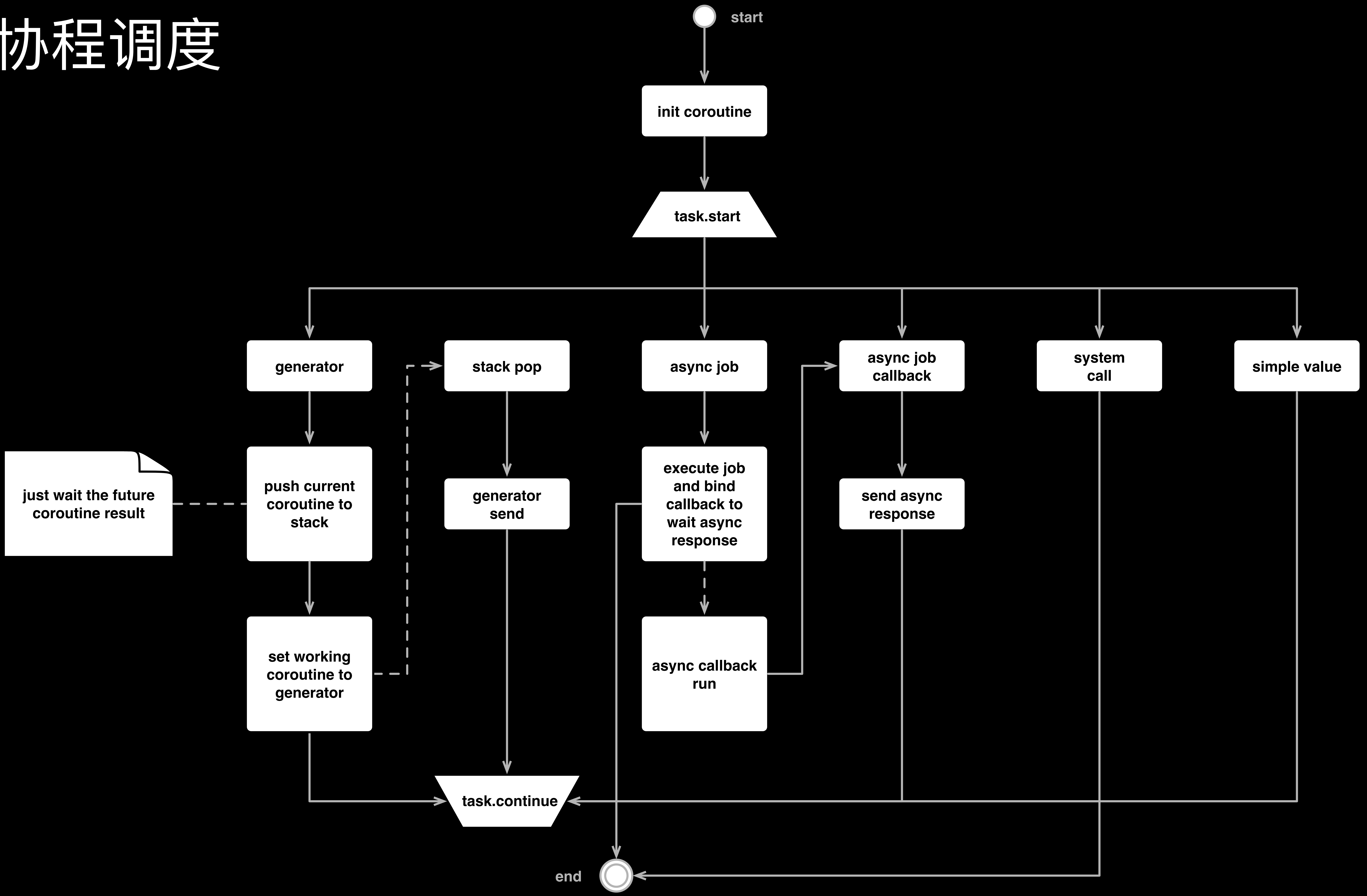
- yield关键字
- Generator
- 函数中断
- 双向通信



```php
<?php
function gen() {
    $ret = (yield 'yield1');
    var_dump($ret);
}

$gen = gen();
var_dump($gen->current());    // string(6) "yield1"
var_dump($gen->send('ret1')); // string(4) "ret1"   (the first var_dump in gen)
```

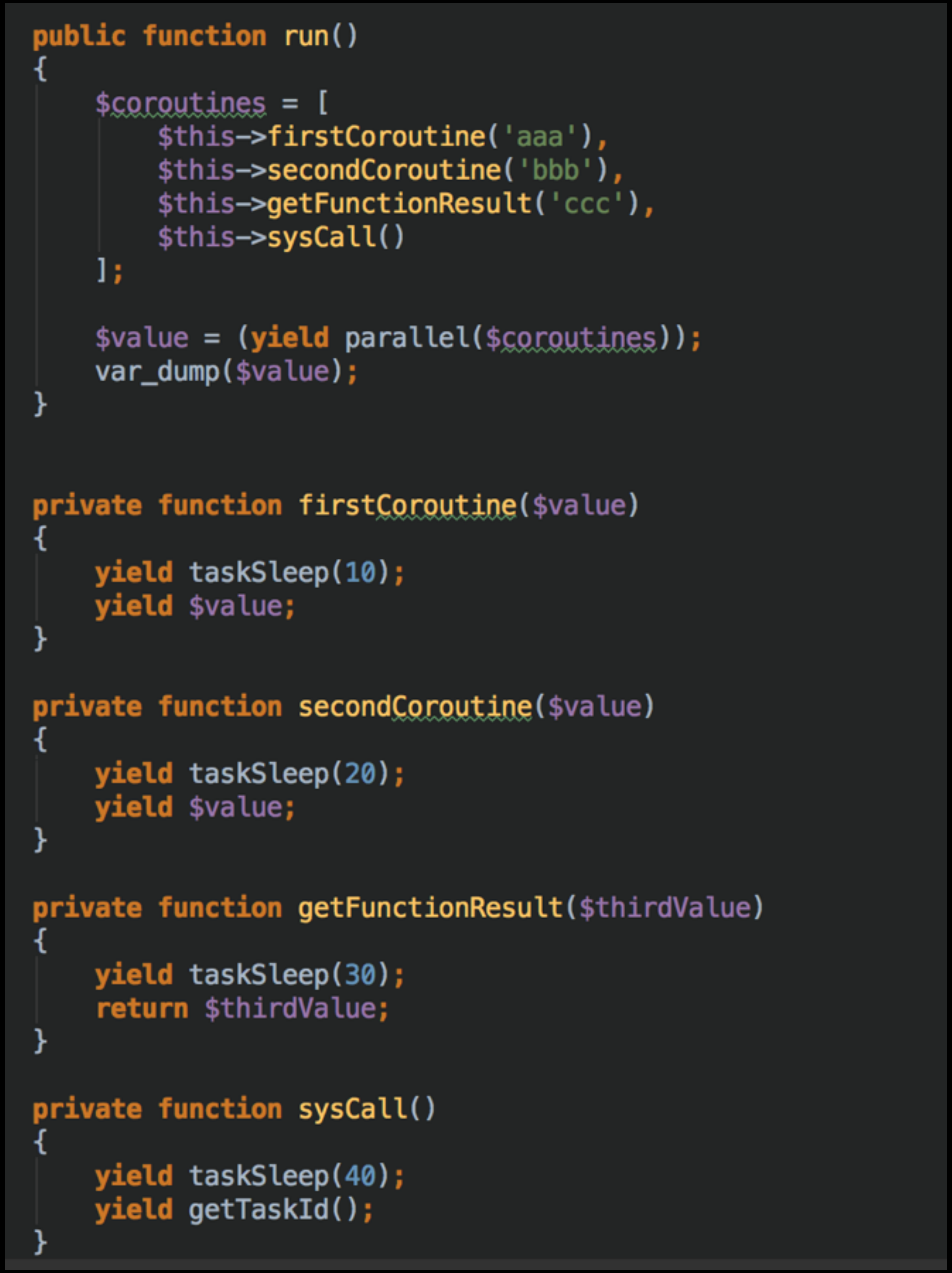
协程调度


start
init coroutine
task.start
generator
stack pop
async job
async job callback
system call
simple value
just wait the future coroutine result
push current coroutine to stack
generator send
execute job and bind callback to wait async response
send async response
set working coroutine to generator
async callback run
task.continue
end

Syscall

- go(Generator $coroutine)
- defer(callable $cb)
- deferRelease(Resource $res)
- parallel($coroutines)
- taskSleep($ms)
- …

并行



```php
public function run()
{
    $coroutines = [
        $this->firstCoroutine('aaa'),
        $this->secondCoroutine('bbb'),
        $this->getFunctionResult('ccc'),
        $this->sysCall()
    ];

    $value = (yield parallel($coroutines));
    var_dump($value);
}


private function firstCoroutine($value)
{
    yield taskSleep(10);
    yield $value;
}

private function secondCoroutine($value)
{
    yield taskSleep(20);
    yield $value;
}

private function getFunctionResult($thirdValue)
{
    yield taskSleep(30);
    return $thirdValue;
}

private function sysCall()
{
    yield taskSleep(40);
    yield getTaskId();
}
```

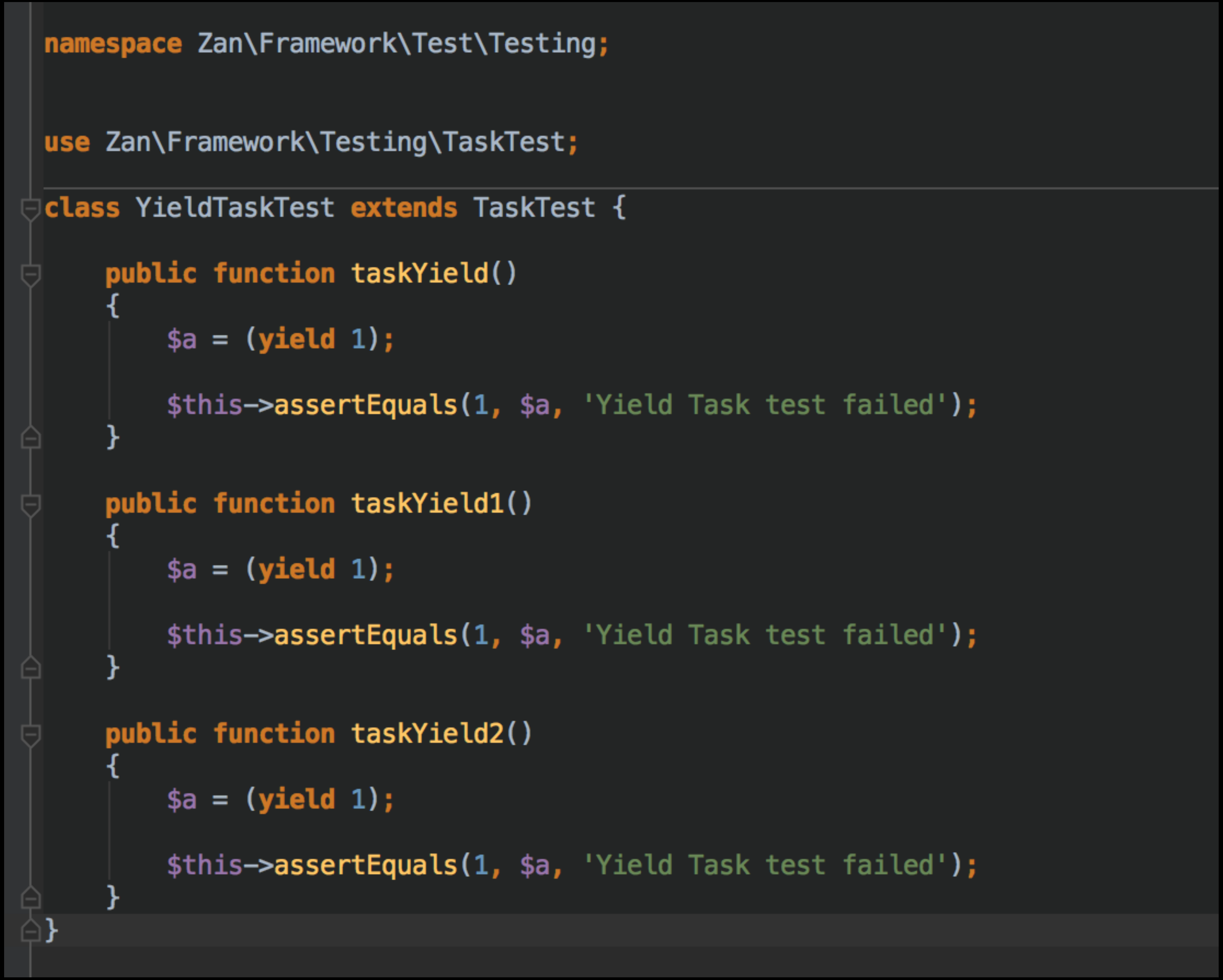
- 基于SysCall
- 轻量级的并行实现
- 业务开发无需关注内部实现

异常处理

- 全流程异常捕获
- 完美支持callback后的异常处理
- RPC异常透传

```
public function index()
{
    try {
        $res = (yield foo());
    } catch (Exception $e) {
        //handle exception
    }
}
```

单元测试支持



```php
namespace Zan\Framework\Test\Testing;

use Zan\Framework\Testing\TaskTest;

class YieldTaskTest extends TaskTest {

    public function taskYield()
    {
        $a = (yield 1);

        $this->assertEquals(1, $a, 'Yield Task test failed');
    }

    public function taskYield1()
    {
        $a = (yield 1);

        $this->assertEquals(1, $a, 'Yield Task test failed');
    }

    public function taskYield2()
    {
        $a = (yield 1);

        $this->assertEquals(1, $a, 'Yield Task test failed');
    }
}
```

- 继承自UnitTest
- 支持异步非阻塞
- task前缀
- case并行调用

Part 3

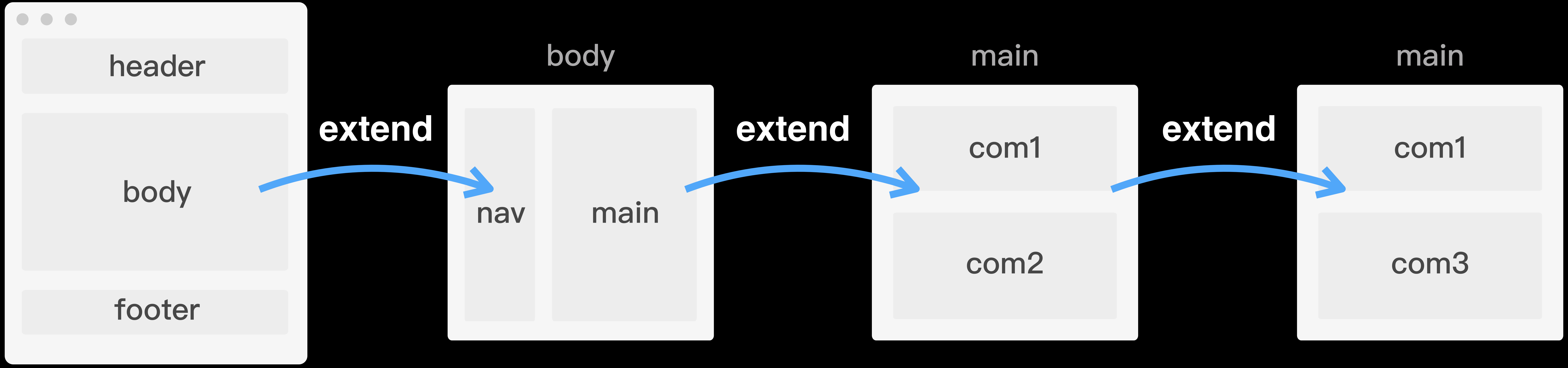
Zan框架设计

SqlMap

```
return [
    'row_by_id'=>[
        'require' => ['user_id'],
        'limit'   => [],
        'sql'     => 'SELECT * FROM users WHERE user_id=#{user_id}',
    ],
    'row_by_name_and_tag'=>[
         'require' => [],
         'limit'   => [],
         'sql'     => 'SELECT * FROM users
                        WHERE uname=#{uname}
                        AND tag_name= #{tag_name}
                        ORDER BY id desc',
    ],
];
```

- SQL定位
- Sharding
- Cache
- 建模驱动
- …

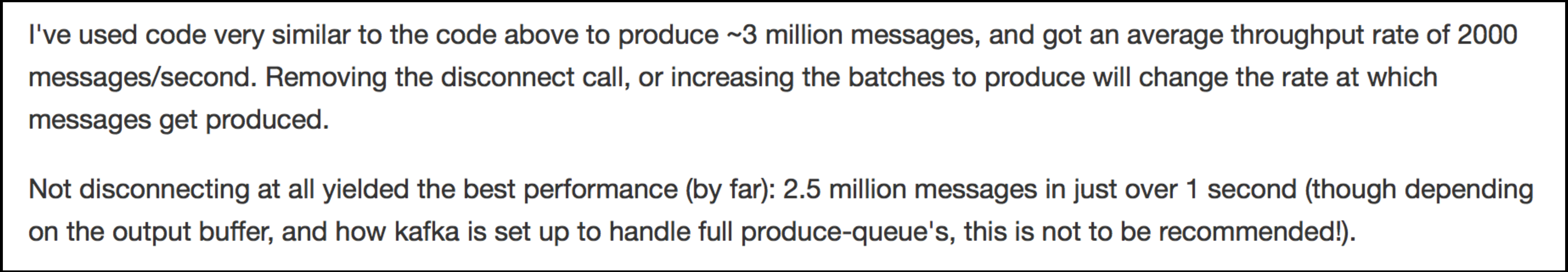
View



- 无限继承
- 组件化
- BigPipe & BigRender

连接池

2000 vs 2500000



I've used code very similar to the code above to produce ~3 million messages, and got an average throughput rate of 2000 messages/second. Removing the disconnect call, or increasing the batches to produce will change the rate at which messages get produced.

Not disconnecting at all yielded the best performance (by far): 2.5 million messages in just over 1 second (though depending on the output buffer, and how kafka is set up to handle full produce-queue's, this is not to be recommended!).

文字来源: https://github.com/EVODelavega/phpkafka

非阻塞Libs

- Mysql
- Redis
- KV
- TCP
- HTTP
- …

SOA

Zan只是SOA路上的第一步...

压测数据

参数: c 300 n 1000000

机器: 32核 64G(受压机 * 1+压测机 * 1)

HttpServer	
场景	TPS
4次串行 -> TCP	14000
4次并行 -> TCP	20000
直接返回	60000

TcpServer	
场景	TPS
20ms延迟返回 2500并发	75000
直接返回	100000

谢谢

期待各路大神加入

chiyou@youzan.com

Zan: http://github.com/youzan/zan