谁说PHP不能异步和并行运行?

场景

在处理需要远程接口调用的大量数据时，我们面临一个关键问题：串行处理导致的效率低下。如果每个接口调用需要1秒，那么即使是10条数据，也需要10秒来完成，这还没有考虑到网络延迟和接口提供方可能出现的问题。在串行执行的情况下，一旦接口调用遇到问题，整个处理时间会成倍增加，这不仅降低了程序的响应速度，也增加了系统的不稳定性。

> 为了解决这个问题，我们可以采取以下几种优化策略：

* **异步执行**：通过异步调用远程接口，可以让程序在等待接口响应的同时继续执行其他任务，从而提高整体的处理速度。
* **并行处理**：利用多线程或多进程技术，同时发起多个远程接口调用，显著减少总的处理时间。

现有方案

远程接口案例

假设第三方或者远程接口调用伪代码如下：

```
<?php
public function sync(): \support\Response
{
    sleep(1);
    return json(['data' => date('Y-m-d H:i:s')]);
}
```

> 接口调用访问地址：`http://127.0.0.1:8888/index/sync`

业务系统案例

假设业务系统调用伪代码

```
<?php
declare(strict_types=1);

foreach (range(1, 10) as $key) {
    $list[] = file_get_contents("http://127.0.0.1:8888/index/sync");
}
print_r($list);
```

> 调用输出

```
[x] [系统调用耗时时间] 10.138074159622
Array
(
    [0] => {"data":"2024-05-16 22:38:00"}
    [1] => {"data":"2024-05-16 22:38:01"}
    [2] => {"data":"2024-05-16 22:38:02"}
    [3] => {"data":"2024-05-16 22:38:03"}
    [4] => {"data":"2024-05-16 22:38:04"}
    [5] => {"data":"2024-05-16 22:38:05"}
    [6] => {"data":"2024-05-16 22:38:06"}
    [7] => {"data":"2024-05-16 22:38:07"}
    [8] => {"data":"2024-05-16 22:38:08"}
    [9] => {"data":"2024-05-16 22:38:09"}
)
```

可以看出上面是按顺序调用接口，总共耗时`10.14`秒

异步并行调用

这个库提供了一个小而简单的`PHP PCNTL`扩展的包装器。它允许并行运行不同的进程，并具有易于使用的API。官方地址：`https://github.com/spatie/async`

安装

您可以通过composer安装该软件包

```
composer require spatie/async
```

> 注意：该扩展库异步并行执行需要所需的扩展`pcntl` 和 `posix`。没有安装在您当前的PHP运行时中， `Pool` 将自动回退到同步执行任务。

`Pool`类有一个静态方法 `isSupported`，你可以调用它来检查你的平台是否能够运行异步进程。

```
require '../vendor/autoload.php';

use Spatie\Async\Pool;
var_dump(Pool::isSupported());
```

支持异步进程则打印`true`，否则为`false`

使用

```
<?php
```

```
/**
 * @author Tinywan(ShaoBo Wan)
 * @date 2024/5/21 14:00
 */
declare(strict_types=1);

require '../vendor/autoload.php';

use Spatie\Async\Pool;

$timeOne = microtime(true);
$pool = Pool::create();

foreach (range(1, 10) as $item) {
    $pool[] = async(function () use ($item) {
        return file_get_contents("http://127.0.0.1:8888/index/sync");
    })->then(function (string $output) use (&$list) {
        // Handle success
        $list[] = $output;
    })->catch(function (Throwable $exception) {
        // Handle exception
        echo '[x] [异常] ' . $exception->getMessage() . PHP_EOL;
    });
}
await($pool);

$timeTwo = microtime(true);
echo '[x] [系统调用耗时时间] ' . ($timeTwo - $timeOne) . PHP_EOL,
print_r($list);
```

> 调用输出

```
[x] [系统调用耗时时间] 4.3443310260773
Array
(
    [0] => {"data":"2024-05-16 22:53:47"}
    [1] => {"data":"2024-05-16 22:53:47"}
    [2] => {"data":"2024-05-16 22:53:47"}
    [3] => {"data":"2024-05-16 22:53:47"}
    [4] => {"data":"2024-05-16 22:53:47"}
    [5] => {"data":"2024-05-16 22:53:47"}
    [6] => {"data":"2024-05-16 22:53:47"}
```

```
    [7] => {"data":"2024-05-16 22:53:47"}
    [8] => {"data":"2024-05-16 22:53:48"}
    [9] => {"data":"2024-05-16 22:53:49"}
)

```

可以看出上面是按并行调用接口，总共耗时`4.34`秒。节省了差不多一半多时间