PHP7+Swoole异步网络编程

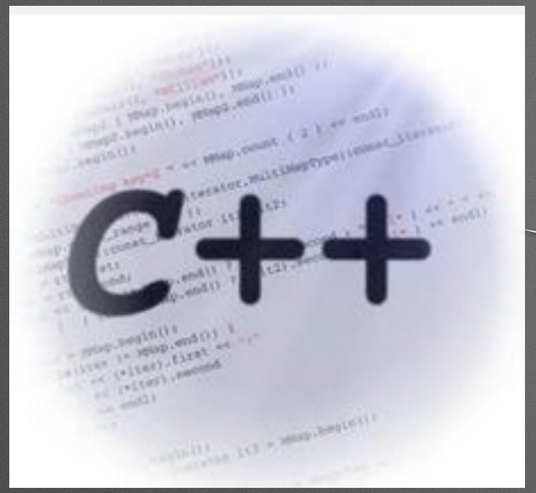
Rango-韩天峰 @车轮互联

关于我

- 车轮互联架构师，资深PHP程序员
- PHP官方扩展开发组成员
- 微博：@hantianfeng
- Github: https://github.com/matyhtf

程序员鄙视链

- C/C++ > Java > C# > Python > JavaScript > PHP
- 所有其他编程语言的程序员都鄙视**PHP**
- **PHP**就是做网站的，**PHP**就是个套页面的 ……



C++程序员眼中的
PHPer

- PHP除了可以实现Web系统，也可以编写通信服务器
- PHP语言开发更便捷，效率极高
- PHP7大幅提升了性能，实际上项目上和C/C++、Java、GO等静态语言，性能差距在2倍以内
- 使用PHP的Swoole扩展可以实现异步并行编程

PHP的高级扩展

- Stream：PHP内核提供的socket封装
- Sockets：对底层Socket API的封装
- Libevent：对libevent库的封装
- Event：基于Libevent更高级的封装，提供了面向对象接口、定时器、信号处理的支持
- Pcntl/Posix：多进程、信号、进程管理的支持
- Pthreads：多线程、线程管理、锁的支持

纯PHP实现Server框架

- React.php，PHP的node.js
- Phpdaemon，基于libevent实现的PHP异步Server端编程框架
- Workerman，纯PHP开发的开源高性能的PHP socket 服务器框架（作者是中国人）

PHP实现TCP服务器

```php
<?php
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr)
or die("create server failed");
while (1) {
    $conn = stream_socket_accept($serv);
    if (pcntl_fork() == 0) {
        $request = fread($conn);
        //do some thing
        //$response = “hello world”;
        fwrite($response);
        fclose($conn);
        exit(0);
    }
}
```

改良版，经典的Leader-Follower

```php
<?php
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr)
or die("create server failed");
for ($i = 0; $i < 32; $i++) {
    if (pcntl_fork() == 0) {
        while (1) {
            $conn = stream_socket_accept($serv);
            if ($conn == false) continue;
            $request = fread($conn);
            //do some thing
            //$response = “hello world”;
            fwrite($response);
            fclose($conn);
        }
        exit(0);
    }
}
```

基于libevent实现的异步服务器

```php
<?php
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr);
//for($i=0; $i < 32; $i ++)
$base = event_base_new();
$event = event_new();
function read_cb($socket, $flag, $base) {
    fread($socket);
    fwrite("hello world\\n");
}

function accept_cb($socket, $flag, $base) {
    $conn = stream_socket_accept($socket, 0);
    stream_set_blocking($conn, 0);
    $event = event_new();
    event_set($event, $conn, EV_READ | EV_PERSIST, 'read_cb');
    event_base_set($event, $base);
    event_add($event);
}

event_set($event, $socket, EV_READ | EV_PERSIST, 'accept_cb', $base);
event_base_set($event, $base);
event_add($event);
event_base_loop($base);
```

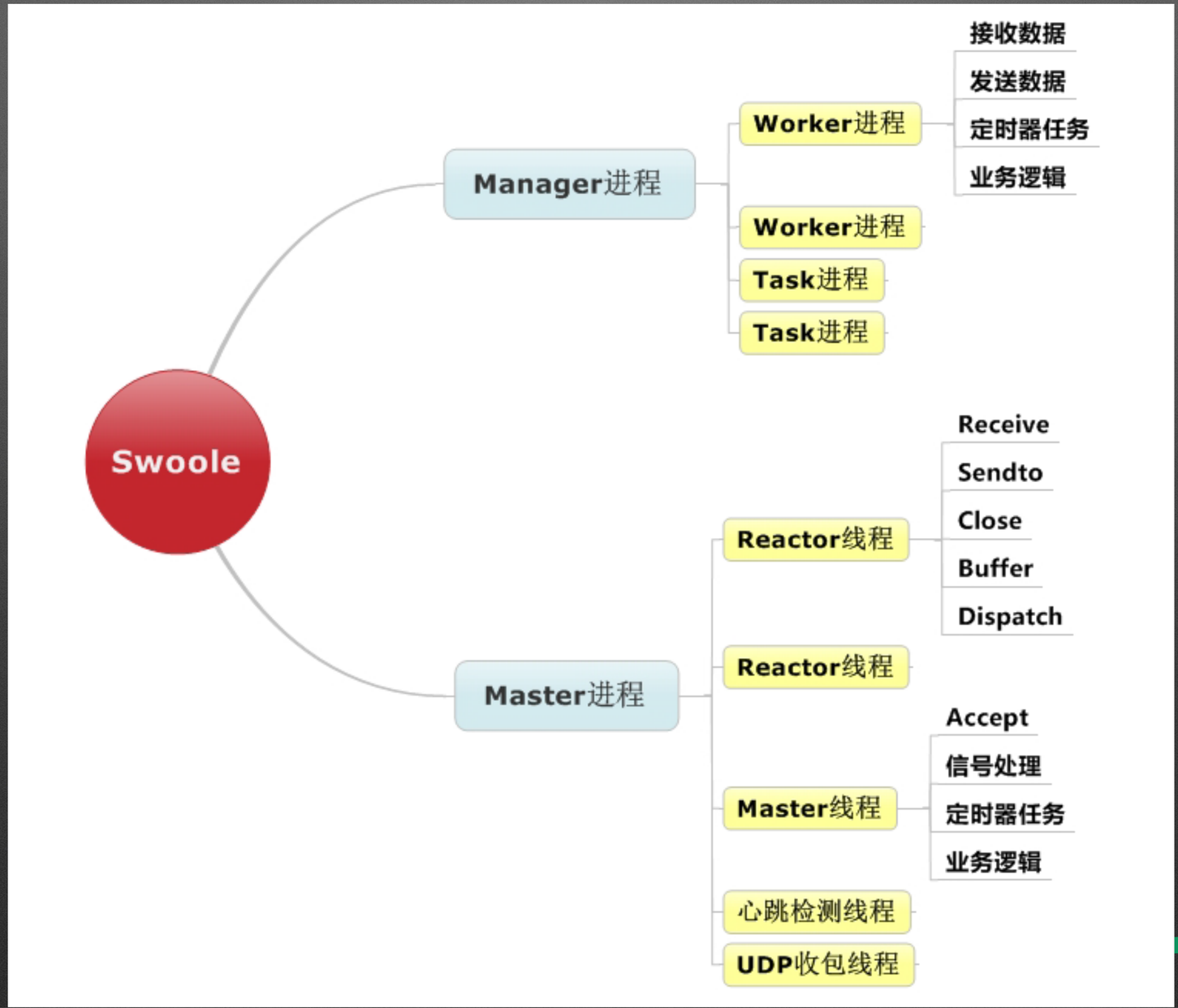
Swoole：C扩展实现异步IO引擎

- Server： TCP/UDP/UnixSocket 、IPv4/IPv6、SSL/TLS加密、Http/WebSocket/Redis/EOF/Length/Mqtt等服务器端协议支持、异步任务进程池、毫秒定时器
- Client：同步+异步的TCP/UDP/UnixSocket客户端，全异步的Http/WebSocket/Redis/MySQL客户端、异步的文件IO、DNS查询、毫秒定时器、异步连接池
- Process：更便捷的多进程组件
- Memory：提供了各类并发安全的数据结构

创建一个异步的TCP服务器

```php
$serv = new Swoole\Server("127.0.0.1", 9501);

//设置服务器参数
$serv->set(array(
    'worker_num' => 8,   //工作进程数量
    'daemonize' => true, //是否作为守护进程
));

//设置事件回调函数
$serv->on('connect', function ($serv, $fd) {
    echo "Client:Connect.\n";
});

$serv->on('receive', function ($serv, $fd, $reactor_id, $data) {
    $serv->send($fd, 'Swoole: ' . $data);
    $serv->close($fd);
});

$serv->on('close', function ($serv, $fd) {
    echo "Client: Close.\n";
});

//启动服务器
$serv->start();
```

Swoole-Server的进程模型



异步TCP客户端

```php
$client = new Swoole\Client(SWOOLE_SOCK_TCP, SWOOLE_SOCK_ASYNC);

//设置事件回调函数
$client->on("connect", function ($cli) {
    $cli->send("hello world\n");
});

$client->on("receive", function ($cli, $data) {
    echo "Received: " . $data . "\n";
});

$client->on("error", function ($cli) {
    echo "Connect failed\n";
});

$client->on("close", function ($cli) {
    echo "Connection close\n";
});

//发起网络连接
$client->connect('127.0.0.1', 9501, 0.5);
```

同步TCP客户端

```php
$client = new swoole_client(SWOOLE_SOCK_TCP);
if (!$client->connect('127.0.0.1', 9501, -1))
{
    exit("connect failed. Error: {$client->errCode}\\n");
}
$client->send("hello world\\n");
echo $client->recv();
$client->close();
```

毫秒定时器

```php
//每隔2秒输出字符串
Swoole\Timer::tick(2000, function ()
{
    echo "2 seconds.\n";
});

//2秒后输出字符串
Swoole\Timer::after(2000, function ()
{
    echo "2 seconds.\n";
});
```

异步Http服务器

```php
$serv = new Swoole\Http\Server("127.0.0.1", 9502);

$serv->on('Request', function ($request, $response) {
    var_dump($request->get);
    var_dump($request->post);
    var_dump($request->cookie);
    var_dump($request->files);
    var_dump($request->header);
    var_dump($request->server);

    $response->cookie("User", "Swoole");
    $response->header("X-Server", "Swoole");
    $response->end("Hello Swoole!");
});

$serv->start();
```

异步WebSocket服务器

```php
$serv = new Swoole\WebSocket\Server("127.0.0.1", 9502);

$serv->on('Open', function ($server, $req)
{
    echo "connection open: " . $req->fd;
});

$serv->on('Message', function ($server, $frame)
{
    echo "message: " . $frame->data;
    $server->push($frame->fd, json_encode(["hello", "world"]));
});

$serv->on('Close', function ($server, $fd)
{
    echo "connection close: " . $fd;
});

$serv->start();
```

异步MySQL客户端

```php
$db = new Swoole\MySQL;
$server = array(
    'host' => '127.0.0.1',
    'user' => 'test',
    'password' => 'test',
    'database' => 'test',
);

$db->connect($server, function ($db, $result)
{
    $db->query("show tables", function (Swoole\MySQL $db, $result)
    {
        var_dump($result);
        $db->close();
    });
});
```

异步Redis客户端

```
$redis = new Swoole\Redis;
$redis->connect('127.0.0.1', 6379, function ($redis, $result)
{
    $redis->set('test_key', 'value', function ($redis, $result)
    {
        $redis->get('test_key', function ($redis, $result)
        {
            var_dump($result);
        });
    });
});
```

异步Http/WebSocket客户端

```php
$cli = new Swoole\Http\Client('127.0.0.1', 80);
$cli->setHeaders(array('User-Agent' => 'swoole-http-client'));
$cli->setCookies(array('test' => 'value'));

$cli->post('/dump.php', array("test" => 'abc'), function ($cli)
{
    var_dump($cli->body);
    $cli->get('/index.php', function ($cli)
    {
        var_dump($cli->cookies);
        var_dump($cli->headers);
    });
});
```

异步客户端+连接池

```php
class RedisPool extends Swoole\\Async\\Pool
{
    protected function connect() {
        $redis = new \\swoole_redis();

        $redis->on('close', function ($redis) {
            $this->remove($redis);
        });

        return $redis->connect($this->config['host'], $this->config['port'], function ($redis, $result) {
            if ($result) {
                $this->join($redis);
            } else {
                $this->failure();
            }
        });
    }

    function __call($call, $params) {
        return $this->request(function (\\swoole_redis $redis) use ($call, $params) {
            call_user_func_array(array($redis, $call), $params);
            //必须要释放资源，否则无法被其他重复利用
            $this->release($redis);
        });
    }
}
```

异步客户端+连接池

```php
<?php
$config = array(
    'host' => '127.0.0.1',
    'port' => '6379',
);

$pool = new Swoole\\Async\\Redis($config, 10);

$pool->get("key", function ($redis, $result) use ($pool) {
    echo "get key: ";
    var_dump($result);
    $redis->incr("key_hello", function ($redis, $result) use ($pool) {
        echo "incr key_hello: ";
        var_dump($result);
    });
});
```

异步任务

```php
$serv = new Swoole\Server("127.0.0.1", 9502);

$serv->set(array('task_worker_num' => 4));

$serv->on('Receive', function ($serv, $fd, $from_id, $data)
{
    $task_id = $serv->task("Async");
    echo "Dispath AsyncTask: id=$task_id\n";
});

$serv->on('Task', function ($serv, $task_id, $from_id, $data)
{
    echo "New AsyncTask[id=$task_id]" . PHP_EOL;
    $serv->finish("$data -> OK");
});

$serv->on('Finish', function ($serv, $task_id, $data)
{
    echo "AsyncTask[$task_id] Finish: $data" . PHP_EOL;
});

$serv->start();
```

并行编程的难题

- **多线程编程：** 1) 读写全局需要加锁，一旦忘记加锁会出现数据同步问题，大量并发时出现严重错误。2) 锁的粒度过大，导致同时只有一个线程在跑（Python的GIL）。3) 复杂的锁逻辑，可能出现重复加锁，导致线程死锁。
- **多进程编程：** 1) 进程是隔离的，A进程无法读取B进程的变量。2) 使用管道进行进程间通信，有一定开销，不方便。3) 使用共享内存，会出现和多线程一样的难题。
- **Lock-Free编程**，非常复杂，容易出错，如同走钢丝。只有极少数熟悉底层CPU硬件原理的人能掌握。

原子计数器

```php
<?php
use Swoole\\Atomic;

$atomic = new Atomic(0);
if (pcntl_fork()) {
    for($i=0; $i<10000; $i++) {
        $atomic->add(1); //+1
    }
} else {
    for($i=0; $i<10000; $i++) {
        $atomic->sub(1); //-1
    }
    $atomic->cmpset(5, 0); //等于5时，设置为0，
    $atomic->get(); //获取当前计数
}
```

并发Table

```php
<?php
use Swoole\\Table;

$table = new Table(1024*1024);
$table->column('id', swoole_table::TYPE_INT, 4);       //1,2,4,8
$table->column('name', swoole_table::TYPE_STRING, 64);
$table->create();

if (pcntl_fork()) {
    for($i=0; $i<10000; $i++) {
        $table->set('test_key_'.$i, array('id' => $i, 'name' => 'v'.$i));
    }
} else {
    for($i=0; $i<10000; $i++) {
        $table->get('test_key_'.$i);
    }
}
```

通道Channel

```php
<?php
$chan = new Swoole\\Channel(1024 * 256);
$n = 100000;
if (pcntl_fork() > 0) {
    for ($i = 0; $i < $n; $i++) {
        $data = str_repeat('A', rand(100, 200));
        if ($chan->push($data) === false) {
            usleep(1000);
            $i--;
            continue;
        }
    }
} else {
    for ($i = 0; $i < $n; $i++) {
        $data = $chan->pop();
        if ($data === false) {
            usleep(1000);
            $i--;
            continue;
        }
    }
}
```

管道通信

```php
<?php
$serv = new swoole_server("0.0.0.0", 9501);

$serv->set(array(
    'worker_num' => 2,
));

$serv->on('pipeMessage', function($serv, $src_worker_id, $data) {
    echo "#{$serv->worker_id} message from #$src_worker_id: $data\\n";
});

$serv->on('receive', function ($serv, $fd, $reactor_id, $data) {
    $dst_worker_id = $serv->setting['worker_num'] - ($serv->worker_id + 1);
    $serv->sendMessage("hello", $dst_worker_id);
});

$serv->start();
```

还有问题...

- 异步回调反人类
- Yield/Generator太难理解

Swoole2.0原生协程

- 服务器的onConnect、onReceive，onRequest，onMessage等事件处理函数中自动创建携程
- 业务逻辑全部使用同步写法
- 底层自动调度，切换协程

Swoole2.0原生协程

```php
<?php
$client = new Swoole\\Coroutine\\Client(SWOOLE_SOCK_TCP);
if (!$client->connect('127.0.0.1', 9501, 0.5))
{
    exit("connect failed. Error: {$client->errCode}\\n");
}
$client->send("hello world\\n");
echo $client->recv();
$client->close();
```

Swoole2.0原生协程

```php
<?php
$cli = new Swoole\\Coroutine\\Http\\Client('127.0.0.1', 80);
$cli->setHeaders([
    'Host' => "localhost",
    "User-Agent" => 'Chrome/49.0.2587.3',
    'Accept' => 'text/html,application/xhtml+xml,application/xml',
    'Accept-Encoding' => 'gzip',
]);
$cli->set([ 'timeout' => 1]);
$cli->get('/index.php');
echo $cli->body;
$cli->close();
```

Swoole2.0原生协程

```php
<?php
$redis = new Swoole\\Coroutine\\Redis();
$redis->connect('127.0.0.1', 6379);
$val = $redis->get('key');
```

Swoole2.0原生协程

```php
<?php
$swoole_mysql = new Swoole\\Coroutine\\MySQL();
$swoole_mysql->connect([
    'host' => '127.0.0.1',
    'user' => 'user',
    'password' => 'pass',
    'database' => 'test',
]);
$res = $swoole_mysql->query('select sleep(1)');
```

Thanks!