河北师范大学 软件学院
Software College of Hebei Normal University

Linux平台PHP服务端开发——

第七讲 消息队列

目录

1

消息队列基础

消息队列介绍

- 消息队列在计算机系统上是进程间通信的方式。
- 消息队列提供了异步处理的方式。消息队列的发布者和接受者不需要知道对方，只需要采用一个协议进行通信即可。
- 消息队列的实现方式有多种，Linux系统本身就提供了消息队列的系统调用，这里关注的是PHP的扩展Semaphore和Redis。
- 消息队列先进先出，数据被读取后，便不再保存。继续读取是下一条信息。

消息队列使用场景

- 用户论坛/微博发帖，同步操作的方式就是获取数据，进行合法检测，还要涉及到数据库写入操作，然后推送到客户端显示。存在的问题是：同步的方式整个过程完成后才返回结果，这个过程要和数据库通信，在高并发情况下直接导致服务系统崩溃。

- 解决方式是使用消息队列，用户提交数据后，后台程序仅仅是把这个请求放到消息队列中，有一个或多个监听消息队列的进程去处理。

- 消息队列提供了异步处理的方式。消息队列的发布者和接受者不需要知道对方，只需要采用一个协议进行通信即可。消息队列的实现方式有多种，Linux系统本身就提供了消息队列的系统调用，这里关注的是PHP的扩展Semaphore和Redis。

消息队列优势与缺点

- 消息队列最主要优点是异步和解耦，异步处理可以减少请求响应时间。解耦能够降低程序复杂度，提高可扩展性与可维护性。

- 缺点：编程相对麻烦，逻辑上不如同步的方式易于理解，并且数据会暂时不一致，要等到消息队列的监听进程处理完，数据库才会更新操作的数据。
- 使用消息队列的优势是明显的，缺点相对不足为虑。

- 消息队列适合对实时同步要求不太高的场景，对于需要即时返回结果的场景不适用。
- ！消息队列的使用不是必须的，盲目的在系统中引入消息队列并不会带来明显的效果，反而增加结构复杂度。对于小规模，并发量不高的网站来说是不必要的。

2

PHP消息队列扩展

Semaphore

- 基于System V IPC 函数族实现，包括信号量、共享内存，进程间通信。
- 此扩展在Windows上不可用。
- 如何安装：编译时加入--enable-sysvsem，--enable-sysvshm，--enable-sysvmsg。
- 依赖：不依赖其他扩展。

Semaphore消息队列函数

- 消息队列函数：

msg_get_queue	创建或重用队列
msg_receive	接收消息
msg_send	发送消息
msg_remove_queue	删除队列
msg_set_queue	设置队列数据
msg_queue_exists	检测队列是否存在

接收消息的函数

第二个参数可以为0，大于0，小于0的数，0可以接收任何消息，大于0则接收消息类型为$desiredmsgtype的类型，小于0则接收消息类型小于或等于$desiredmsgtype绝对值的类型。

```
bool msg_receive ( resource $queue , int $desiredmsgtype , int &$msgtype , int $maxsize , mixed &$message [, bool $unserialize = TRUE [, int $flags = 0 [, int &$errorcode ]]] )
```

Semaphore消息队列示例

- 发送方代码：
- $mq = msg_get_queue(1234);

```
$msg_err=0;

for ($i=0;$i<100;$i++) {
    msg_send($mq,1,mt_rand(100,1000),true,true,$msg_err);
}
```

- 此操作是发送消息的类型为1。

Semaphore消息队列示例

- 接收消息代码，接收类型为0（第二个参数），如果类型为5，则只有发送的消息类型为5的才能接收到：

```
$mq = msg_get_queue(1234);
$msg = ‘’; $msgtype=0; $errcode = 0; $count = 1;
while (true) {
    usleep(10000); //延迟10ms，实际并不需要，接收数据未到会阻塞。
    msg_receive($mq,0,$msgtype,100,$msg,true,0,$errcode);
    if (!$errcode) {
        echo "Received message($count): ",$msg,"\n";
    }
    else {
        echo "Errcode:",$errcode,"\n";
    }
    $count += 1;
}
```

多个接收者

- 多个进程监听同一个消息队列，发布者发布消息时，消息会被多个接收者读取。
- 一个接收者获取消息后，此消息便不存在，其他接收者获取的是后续的消息。
- 实际场景中根据需要来确定需要多少个进程监听，不同的业务使用不同的队列。

3

Redis

Redis基本介绍

- Redis 是一个开源（BSD许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。使用ANSI C语言编写。

- 它支持多种类型的数据结构，如 字符串（strings），散列（hashes），列表（lists）等。

- 官方不支持Windows版本的Redis,但微软开发和维护着支持win-64 的Redis版本。
- 应用场景：
  - 将Redis作为一个高效的网络的缓存数据功能使用。
  - 使用发布/订阅功能用作消息队列。

PHP的Redis扩展与Redis服务程序

- PHP提供了Redis扩展用于连接Redis服务进行操作。从http://pecl.php.net/package/redis可以下载最新版的Redis扩展。
- 安装过程使用之前讲到的编译安装：

```
./configure  --with-php-config=PHP_INSTALL_DIR/bin/php-config
sudo  make  install
```

- 也可以使用pecl install redis
- 安装完成后要配置php.ini启用redis扩展，并重启web服务。
- 安装redis服务：sudo apt install redis-server
- 安装redis-server会自动安装redis-tools，安装后有redis-cli用于连接redis-server。
- redis-server默认运行在6379端口。

Redis客户端工具

- 运行redis-cli直接就会连接redis-server，不需要参数。

```
wy@helloworld:~$ redis-cli
127.0.0.1:6379>
```

- 命令示例：
  - 添加： append brave master //如果重复此操作，则会在key索引的值后面添加数据
  - 获取： get brave
  - 重置： set brave programmer //如果没有此索引，则会创建
  - 删除： del brave
  - 列表： lpush [LIST KEY] [VALUE] //把VALUE值压入LIST KEY，重复操作元素会不断加入LIST KEY
  - 列表： lpop [LIST KEY] 弹出最近压入的元素

PHP获取Redis扩展调用方法

- PHP获取Redis扩展支持的调用方法：

```
<?php
$rd = new Redis();
$rd_calls = get_class_methods($rd);

foreach ($rd_calls as $call) {
    echo $call . "\n";
}
```

- 方法名称都是和Redis对应的命令，参数顺序一般都符合Redis命令的逻辑，个别命令在使用PHP扩展的时候有所不同。

PHP连接Redis添加数据

- PHP连接Redis并添加一条数据：

```
<?php
$rd = new Redis();

$rd->connect('127.0.0.1',6379);

$rd->append('test','php');
```

发布/订阅

- Redis支持发布订阅模式。
- 客户端运行subcribe [CHANNEL1] [CHANNEL2]…订阅给指定频道的信息。

- 一旦客户端进入订阅状态，客户端就只可接受订阅相关的命令

- 使用publish [CHANNEL] [DATA] 向指定频道发布数据。
- 示例：
  客户端1运行：subscribe chan1 chan2
  客户端2运行：publish chan1 ‘hello php’ ； publish chan2 ‘hello redis’

PHP实现PUBLISH

- 使用$redis->publish方法，并使用redis-cli订阅频道获取数据

```
<?php
$rd = new Redis();

$rd->connect('127.0.0.1',6379);

for ($i=0;$i<10;$i++) {
    $rd->publish('redis_test',mt_rand(1000,10000));
}
```

PHP实现SUBSCRIBE

- 使用$redis->subscribe方法，并编写客户端发布消息，订阅代码示例：

```
function sub_handle($rd, $chan, $msg)
{
    switch ($chan) {
        case 'chan1':
            echo "chan1: $msg\n";
            break;
        case 'chan2':
            echo "chan2: $msg\n";
            break;
        case 'chan3':
            echo "chan3: $msg\n";
            break;
        default:;
    }
}

$rd = new Redis();
$rd->connect('127.0.0.1',6379);
$rd->subscribe(['chan1','chan2','chan3'], 'sub_handle');
```

SUBSCRIBE超时

- 使用$redis->subscribe方法，如果没有数据到达，默认会60s超时。如果需要长期运行则需要进行超时设置。

- 代码示例：

```
$rd = new Redis();
$rd->pconnect('127.0.0.1',6379);

$rd->setOption(Redis::OPT_READ_TIMEOUT, -1); //超时设置

$rd->subscribe(['chan1','chan2','chan3'], 'sub_handle');
```