<https://www.cnblogs.com/lhfcws/p/3732535.html>

一般来说，消息队列有两种场景，一种是发布者订阅者模式，一种是生产者消费者模式。利用redis这两种场景的消息队列都能够实现。  
**定义：**

* **生产者消费者模式**：生产者生产消息放到队列里，多个消费者同时监听队列，谁先抢到消息谁就会从队列中取走消息；即对于每个消息只能被最多一个消费者拥有。
* **发布者订阅者模式**：发布者生产消息放到队列里，多个监听队列的消费者都会收到同一份消息；即正常情况下每个消费者收到的消息应该都是一样的。

那么如此多的MQ产品，为什么要使用redis作消息队列呢？以下附上一份总结了别人的一些report或blog的表格，以及当初用来说服整个team的一句结论。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MQ** | **Env.** | **Weight** | **Disadvantage** |  |  |  |
| RabbitMQ | Erlang | Heavy | Bad scalability;Low speed; | |  |  |
| ZeroMQ | C | Light | difficult for development | |  |  |
| ActiveMQ | Java | - | Low performance |  |  |  |
| Redis | C | - | Low performance while enqueuing big data (>= 10k) | | | |

Redis is easy to use and configure since we have experience in Redis, and most importantly, its performance satisfies our requirement.

Then, how to use redis as a MQ?

首先，redis的队列实际在代码逻辑中不需要由我们自己实现，因此一个所谓的 RedisMQ 对象实际是一个 redis key以及对其操作的一些封装。

**PubSub Mode:**

redis 从 2.0.0 版本开始支持 pub/sub 指令。详情见 <http://redis.io/topics/pubsub>

实现思想很简单，Publisher调用redis的publish方法往特定的channel发送消息，Subscriber在初始化的时候要subscribe到该channel，一旦有消息就会立即接收。

比较简单的demo可参见：[http://shift-alt-ctrl.iteye.com/blog/1867454](http://shift-alt-ctrl.iteye.com/blog/1867454" \t "_blank) ，此链接博客中写得已较详细，本文便不再赘述。

**Producer/Consumer Mode:**

该方法是借助redis的list结构实现的。

Producer调用redis的lpush往特定key里塞入消息，Consumer调用brpop去不断监听该key。

producer：

1 // producer code

2 String key = "demo:mq:test";

3 String msg = "hello world";

4 redisDao.lpush(key, msg);

consumer:

[复制代码](javascript:void(0);)

1 // consumer code

2 String key = "demo:mq:test";

3 while (true) {

4 // block invoke

5 List<String> msgs = redisDao.brpop(BLOCK\_TIMEOUT, listKey);

6 if (msgs == null) continue;

7 String jobMsg = msgs.get(1);

8 processMsg(jobMsg);

9 }

[复制代码](javascript:void(0);)

当有多个consumers的时候，它会按照brpop调用的顺序分派消息，并非随机。

BLOCK\_TIMEOUT不建议设成infinity（有些redis驱动也直接不支持inifinity），我们目前设成30（单位是秒）情况良好。

P.S. 本文时间较久远，适合redis 2的版本，不保证redis自己会不会有其他新特性 ；同时消息队列产品有很多种，这里列的只是早年常用的，近两三年的kafka和阿里的rocketmq也很火，至于怎么选择，一部分是根据数据量，若数据量不大，容错要求不是极高，redis是个高效开发易维护的好选择；如果数据量很大或对消息准确性有一定要求，那应当考虑更成熟的消息队列产品比如kafka等。所以mq的选型并不是本文的重点，本文只是介绍一下基于redis 2.6的mq的简单封装实现。