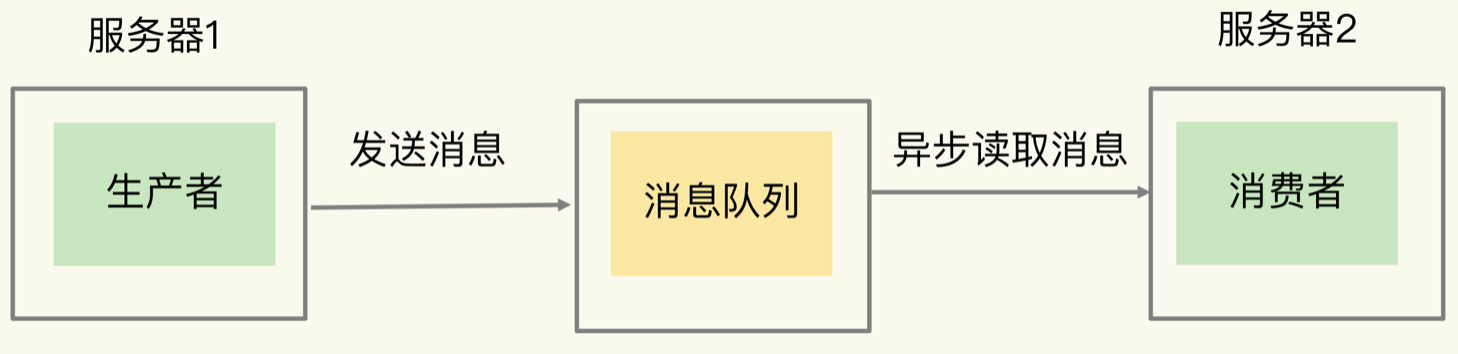
消息队列

一、消息存取需求



在使用消息队列时，消费者可以异步读取生产者消息，然后再进行处理。即使生产者发送消息的速度远远超过了消费者处理消息的速度，生产者已经发送的消息也可以缓存在消息队列中，避免阻塞生产者，这是消息队列作为分布式组件通信的一大优势。

1.1 消息保序

虽然消费者是异步处理消息，但是，消费者仍然需要按照生产者发送消息的顺序来处理消息，避免后发送的消息被先处理了。

1.2 重复消息处理

消费者从消息队列读取消息时，有时会因为网络堵塞而出现消息重传的情况。此时，消费者可能会收到多条重复的消息。

1.3 消息可靠性保证

消费者在处理消息的时候，还可能出现因为故障或宕机导致消息没有处理完成的情况。此时，消息队列需要能提供消息可靠性的保证，也就是说，当消费者重启后，可以重新读取消息再次进行处理，否则，就会出现消息漏处理的问题了。

二、基于List的消息队列解决方案

List本身就是按先进先出的顺序对数据进行存取的，所以，如果使用List作为消息队列保存消息的话，就已经能满足消息保序的需求了。

具体来说，生产者可以使用LPUSH命令把要发送的消息依次写入List，而消费者则可以使用RPOP命令，从List的另一端按照消息的写入顺序，依次读取消息并进行处理。

Redis提供了BRPOP命令。BRPOP命令也称为阻塞式读取，客户端在没有读到队列数据时，自动阻塞，直到有新的数据写入队列，再开始读取新数据。

消费者程序本身能对重复消息进行判断。

- [ ] 消息队列要能给每一个消息提供全局唯一的ID号；

- [ ] 消费者程序要把已经处理过的消息的ID号记录下来。

当收到一条消息后，消费者程序就可以对比收到的消息ID和记录的已处理过的消息ID，来判断当前收到的消息有没有经过处理。如果已经处理过，那么，消费者程序就不再进行处理了。这种处理特性也称为幂等性，幂等性就是指，对于同一条消息，消费者收到一次的处理结果和收到多次的处理结果是一致的。

为了留存消息，List类型提供了BRPOPLPUSH命令，这个命令的作用是让消费者程序从一个List中读取消息，同时，Redis会把这个消息再插入到另一个List（可以叫作备份List）留存。这样一来，如果消费者程序读了消息但没能正常处理，等它重启后，就可以从备份List中重新读取消息并进行处理了。

问题：生产者消息发送很快，而消费者处理消息的速度比较慢，这就导致List中的消息越积越多，给Redis的内存带来很大压力。

这个时候启动多个消费者程序组成一个消费组，一起分担处理List中的消息。但是，List类型并不支持消费组的实现。

三、基于Streams的消息队列解决方案

Redis从5.0版本开始提供的Streams数据类型，Streams是Redis专门为消息队列设计的数据类型。和List相比，Streams同样能够满足消息队列的三大需求。而且，它还支持消费组形式的消息读取，提供丰富的消息队列操作命令。

- [ ] XADD：插入消息，保证有序，可以自动生成全局唯一ID；

- [ ] XREAD：用于读取消息，可以按ID读取数据；

- [ ] XREADGROUP：按消费组形式读取消息；

使用消费组的目的是让组内的多个消费者共同分担读取消息，所以，我们通常会让每个消费者读取部分消息，从而实现消息读取负载在多个消费者间是均衡分布的。

- [ ] XPENDING和XACK：XPENDING命令可以用来查询每个消费组内所有消费者已读取但尚未确认的消息，而XACK命令用于向消息队列确认消息处理已完成。

为了保证消费者在发生故障或宕机再次重启后，仍然可以读取未处理完的消息，Streams会自动使用内部队列（也称为PENDING List）留存消费组里每个消费者读取的消息，直到消费者使用XACK命令通知Streams“消息已经处理完成”。如果消费者没有成功处理消息，它就不会给Streams发送XACK命令，消息仍然会留存。此时，消费者可以在重启后，用XPENDING命令查看已读取、但尚未确认处理完成的消息。

四、小结

