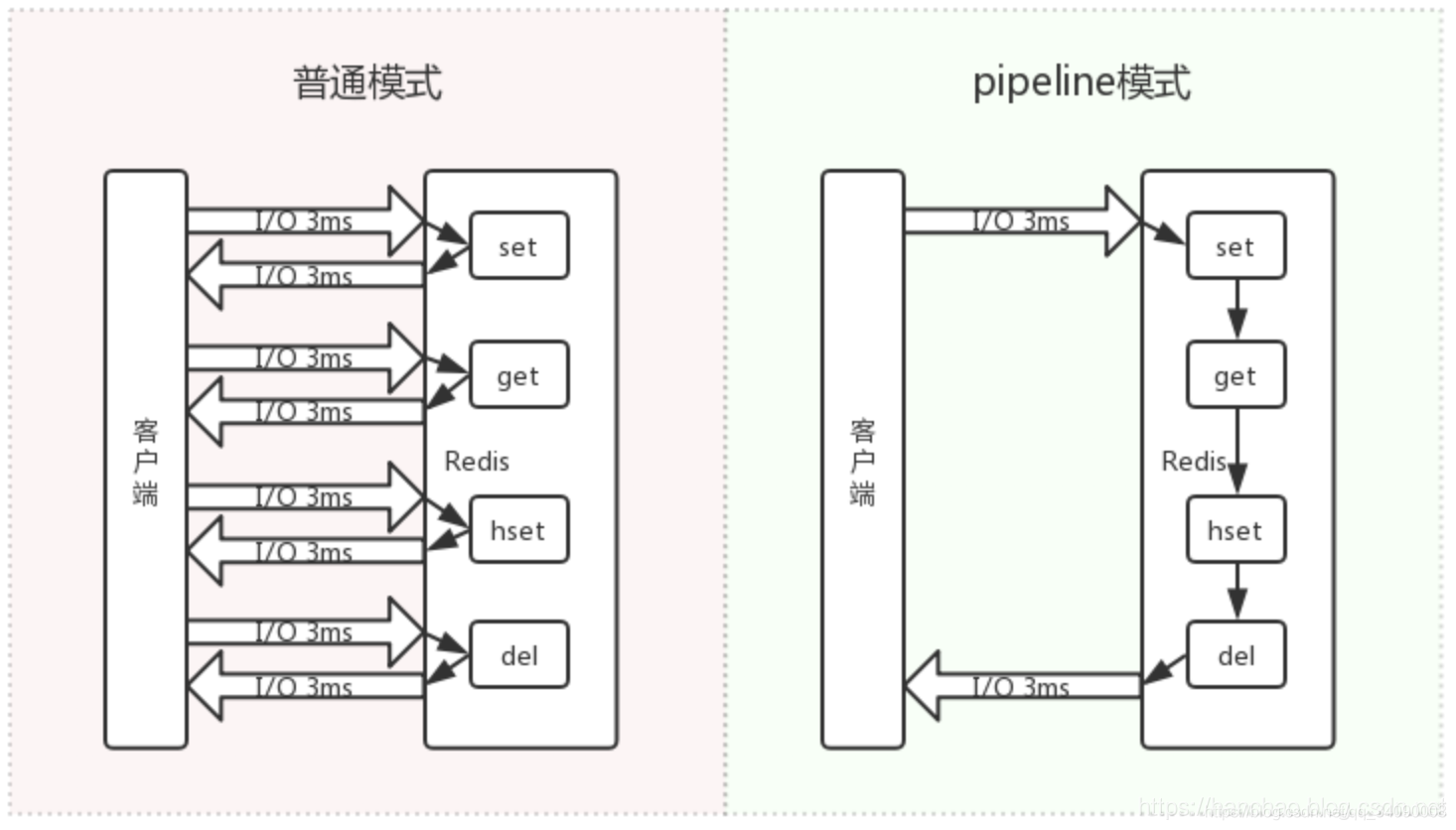
**Redis之Pipeline**

**概述**

好哥哥们还记得 Redis 执行一条需要经过发送命令、命令排队、命令执行、结果响应四个步骤。由于 Redis 本身是基于 Request/Response协议（停等机制）的，虽然 Redis 已经提供了像 mget 、mset 这种批量的命令，但是好哥哥们想一下，如果某些操作根本就不支持或没有批量的操作，是不是就要一条一条的执行命令。那这样岂不是和我大 Redis 高性能背道而驰了（因为每执行一条命令都要消耗请求与响应的时间）。好哥哥们会问了，那有什么办法能解决这个问题呢，答案就是Pipeline。  
Pipeline 字面意思是管道也可以说是流水线。Pipeline也并不是什么新的技术或机制，像在Jenkins 、Netty 都有运用到。 在 Redis 中通过Pipeline机制能改善上面这类问题，**它能将一组 Redis 命令进行组装，通过一次传输给 Redis 并返回结果集**。如下图  


**怎么用**

用的话很简单，redis-cli 的 --pipe参数实际上就是使用 Pipeline 机制，例如下面操作将 set hello world 和 incr count 两条命令组装。如下代码。当然，正常我们都是通过 Redis 客户端像 Jedis 来操作，这个会放在后面讲。

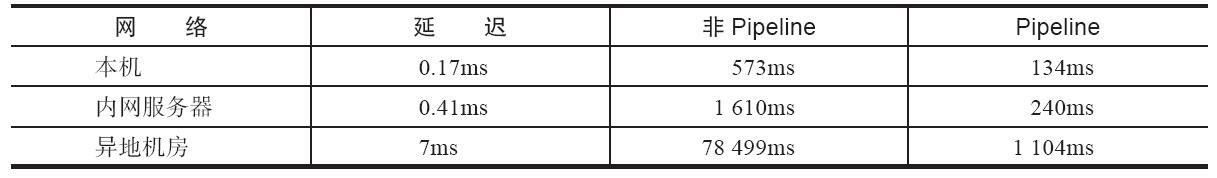
1. ## 格式
2. echo -en '\*3\r\n$3\r\nSET\r\n$5\r\nhello\r\n$5\r\nworld\r\n\*2\r\n$4\r\nincr\r\
3. n$7\r\ncounter\r\n' | redis-cli --pipe
4. 复制代码

**原理**

需要实现Pipeline 功能，需要客户端和服务器端的支持。  
Redis 服务器端支持处理一个客户端通过同一个 TCP 连接发来的多个命令。可以理解为，这里将多个命令切分，和处理单个命令一样，处理完成后会将处理结果缓存起来，所有命令执行完毕后统一打包返回。这里就会涉及到一个问题就是如果Pipeline一次的命令条数过多时，会使响应结果撑爆Socket接收缓冲区，所以好哥哥们要控制一下每次的命令条数。  
客户端方面像Jedis实现的逻辑是，通过 API 生成一个Pipeline对象，每次往Pipeline中添加命令时，Jedis会将命令写入（只是写入还没有发送给 Redis）到Outputstream（Jdedis 封装了自己的输入/输出流）。当真正调用获取结果时会调flush()方法然后阻塞获取返回结果，然后将结果包装返回给调用方。

**性能测试**

在不同网络下，10000 条 set 非 Pipeline 和 Pipeline 的执行时间对比。 需要注意的是，针对于环境和网络的不同，测试出来的结果肯定是不一样的。如果网络延迟也高，那么 Pipeline 的性能肯定是越好的。



**批量命令、Pipeline 对比**

1. 原生批量命令是原子的，Pipeline 是非原子的。
2. 原生批量命令是一个命令对应多个 key，Pipeline 支持多个命令。
3. 原生批量命令是 Redis 服务端支持实现的，而 Pipeline 需要服务端和客户端的共同实现

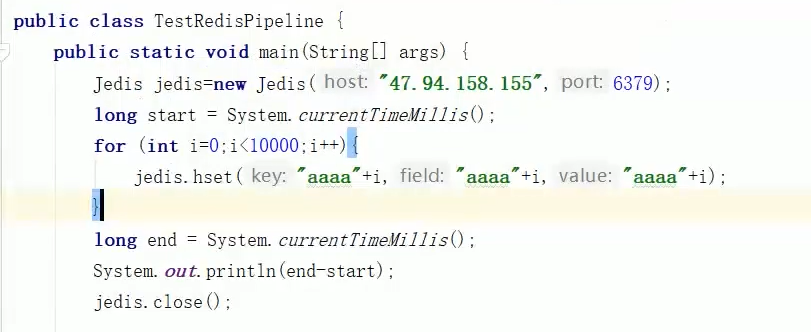
**适用场景**

Peline是 Redis 的一个提高吞吐量的机制，适用于**多 key 读写场景**，比如同时读取多个key 的value，或者更新多个key的value，并且允许一定比例的**写入失败**、**实时性**也没那么高，那么这种场景就可以使用了。比如 10000 条一下进入 redis，可能失败了 2 条无所谓，后期有补偿机制就行了，像短信群发这种场景，这时候用 pipeline 最好了。

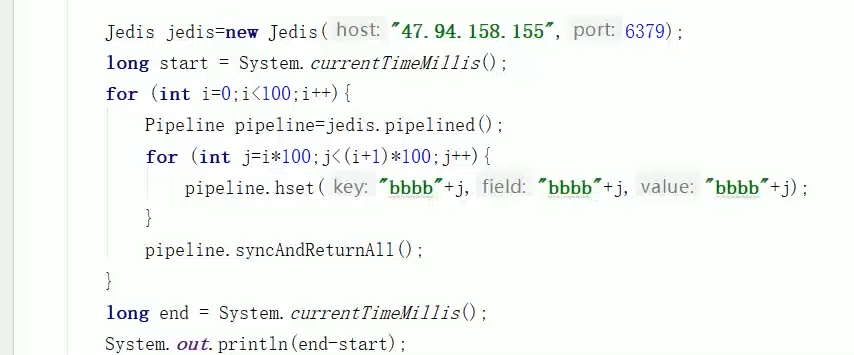
**注意点**

1. Pipeline是非原子的，在上面原理解析那里已经说了就是 Redis 实际上还是一条一条的执行的，而执行命令是需要排队执行的，所以就会出现原子性问题。
2. Pipeline中包含的命令不要包含过多。
3. Pipeline每次只能作用在一个 Redis 节点上。
4. Pipeline 不支持事务，因为命令是一条一条执行的。

**示例：**



以上代码需要执行7分钟



使用管道后执行时间缩短至7秒；