一：Redis里面的5种数据结构

1.String

service.set<string>(“12”，“34”);

2.Hashtable

service.SetEntryInHash("tyhash", "1","123");

3.set

4.ZSet

5.List

二：Redis的其他特性

缓存穿透：大量查询缓存中不存在的数据，导致所有的请求都去请求存储数据库，导致存储数据库压力 很大。

解决方式：

布隆过滤器：

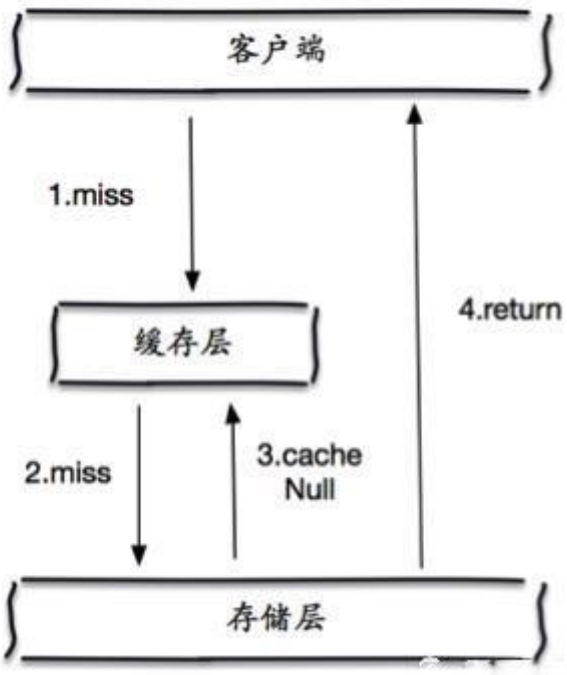
BF是由一个**长度为m比特的位数组（bit array）**与**k个哈希函数（hash function）**组成的数据结构。位数组均初始化为0，所有哈希函数都可以分别把输入数据尽量均匀地散列。

当要**插入**一个元素时，将其数据分别输入k个哈希函数，产生k个哈希值。以哈希值作为位数组中的下标，将所有k个对应的比特置为1。

当要**查询**（即判断是否存在）一个元素时，同样将其数据输入哈希函数，然后检查对应的k个比特。如果有任意一个比特为0，表明该元素一定不在集合中。如果所有比特均为1，表明该集合有（较大的）可能性在集合中。为什么不是一定在集合中呢？因为一个比特被置为1有可能会受到其他元素的影响，这就是所谓“假阳性”（false positive）。相对地，“假阴性”（false negative）在BF中是绝不会出现的

缓存空对象

当存储层不命中后，即使返回的空对象也将其缓存起来，同时会设置一个过期时间，之后再访问这个数据将会从缓存中获取，保护了后端数据源；



但是这种方法会存在两个问题：

如果空值能够被缓存起来，这就意味着缓存需要更多的空间存储更多的键，因为这当中可能会有很多的空值的键；即使对空值设置了过期时间，还是会存在缓存层和存储层的数据会有一段时间窗口的不一致，这对于需要保持一致性的业务会有影响

缓存雪崩：缓存直接出错了，全部都去存储数据库查找 。

（1）redis高可用

这个思想的含义是，既然redis有可能挂掉，那我多增设几台redis，这样一台挂掉之后其他的还可以继续工作，其实就是搭建的集群。

（2）限流降级

这个解决方案的思想是，在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。

（3）数据预热

数据加热的含义就是在正式部署之前，我先把可能的数据先预先访问一遍，这样部分可能大量访问的数据就会加载到缓存中。在即将发生大并发访问前手动触发加载缓存不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀

BIO和NIO的区别是什么呢?

在BIO模式下，调用read，如果发现没数据已经到达，就会Block住。

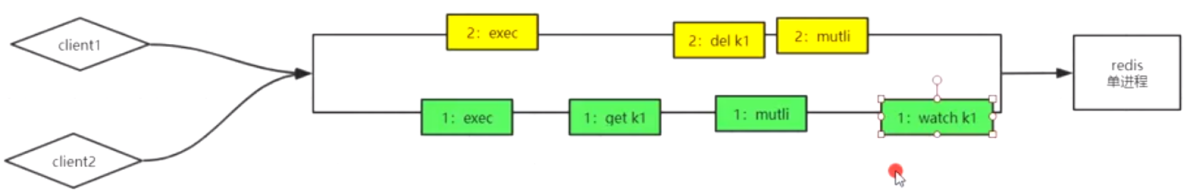
BIO底层原理

```java
package bio_nio.study;

import java.io .IOException;
import java.io .InputStream;
import java.io .OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 服务端
 */
public class FirstServer {

    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8000);
            System.out.println("服务器启动成功，监听端口8000");
            // 不断监听客户端的请求
            while (true) {
                Socket socket = serverSocket.accept();  //阻塞
                InputStream inputStream = socket.getInputStream();
                byte[] buffer = new byte[1024];
                int length = 0;
                // 读取客户端的数据
                while ((length = inputStream.read(buffer)) > 0) {
                    System.out.println(new String(buffer, 0, length));
                }
                // 向客户端写数据
                OutputStream outputStream = socket.getOutputStream();
                outputStream.write("hello java".getBytes());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

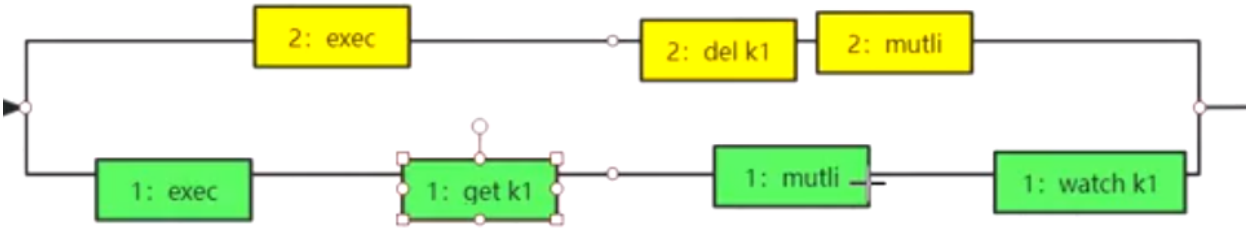
在NIO模式下，调用read，如果发现没数据已经到达，就会立刻返回-1, 并且errno被设为EAGAIN

管道：可以同时向redis发送多个命令，节省往返时间

事务：同时有多个事务的时候，那个事务的exec先到达则先执行那个客户端的事务



Watch查询对应的数据，如果被更改了话对应的事务就不会执行



零拷贝

1.传统I/O

硬盘—>内核缓冲区—>用户缓冲区—>内核socket缓冲区—>协议引擎

2.sendfile

硬盘—>内核缓冲区—>内核socket缓冲区—>协议引擎

3.sendfile（ DMA 收集拷贝）

硬盘—>内核缓冲区—>协议引擎

。

二：Redis的栈和队列

1.栈

2.队列

生产者，消费者

发布订阅