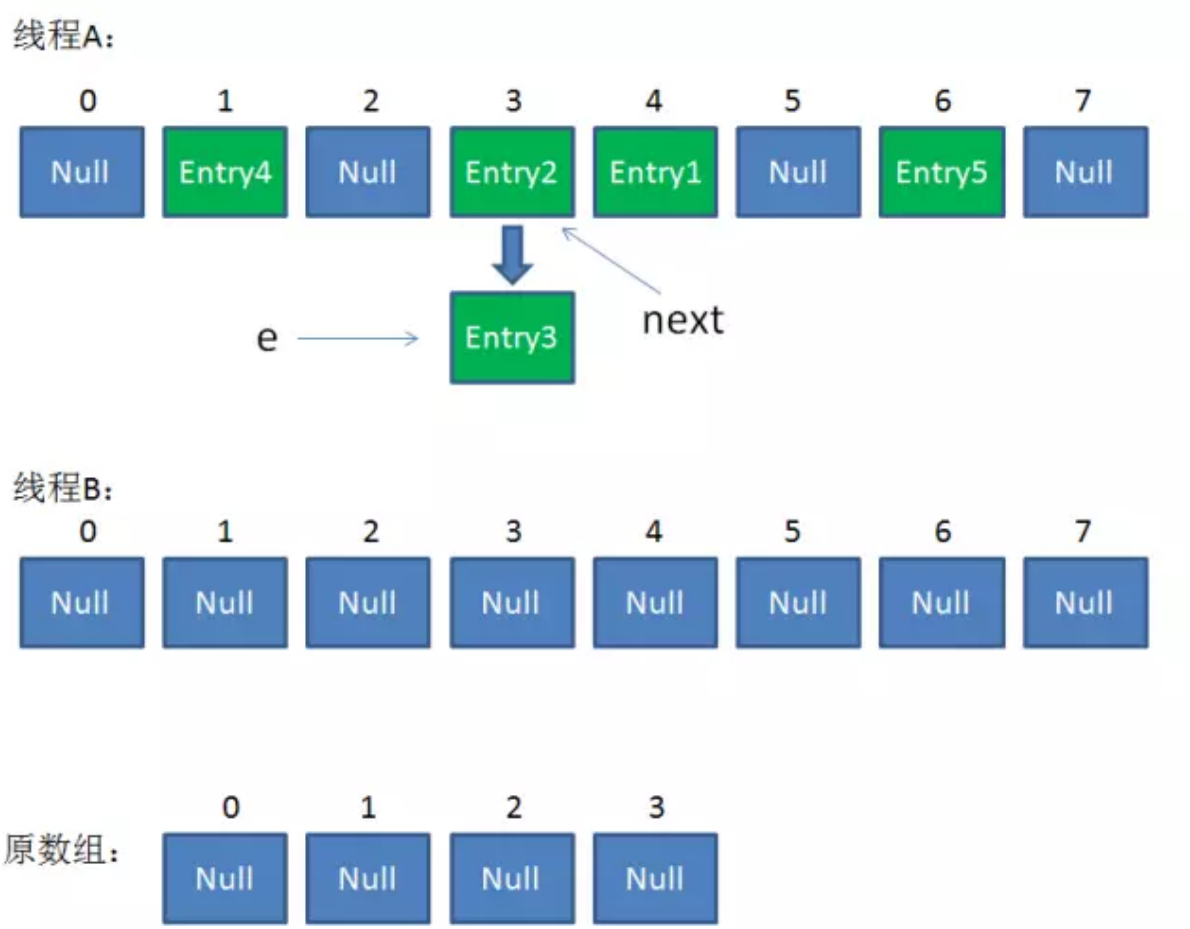
上午坐在图书馆前草坪的椅子上看《Netty权威指南》，差不多到了饭点，正准备起身去食堂吃饭，这时迎来一个突如其来的电话，上面显示：“浙江-杭州”。（因此前并未约面试，不知这就是阿里巴巴菜鸟网络的面试电话）

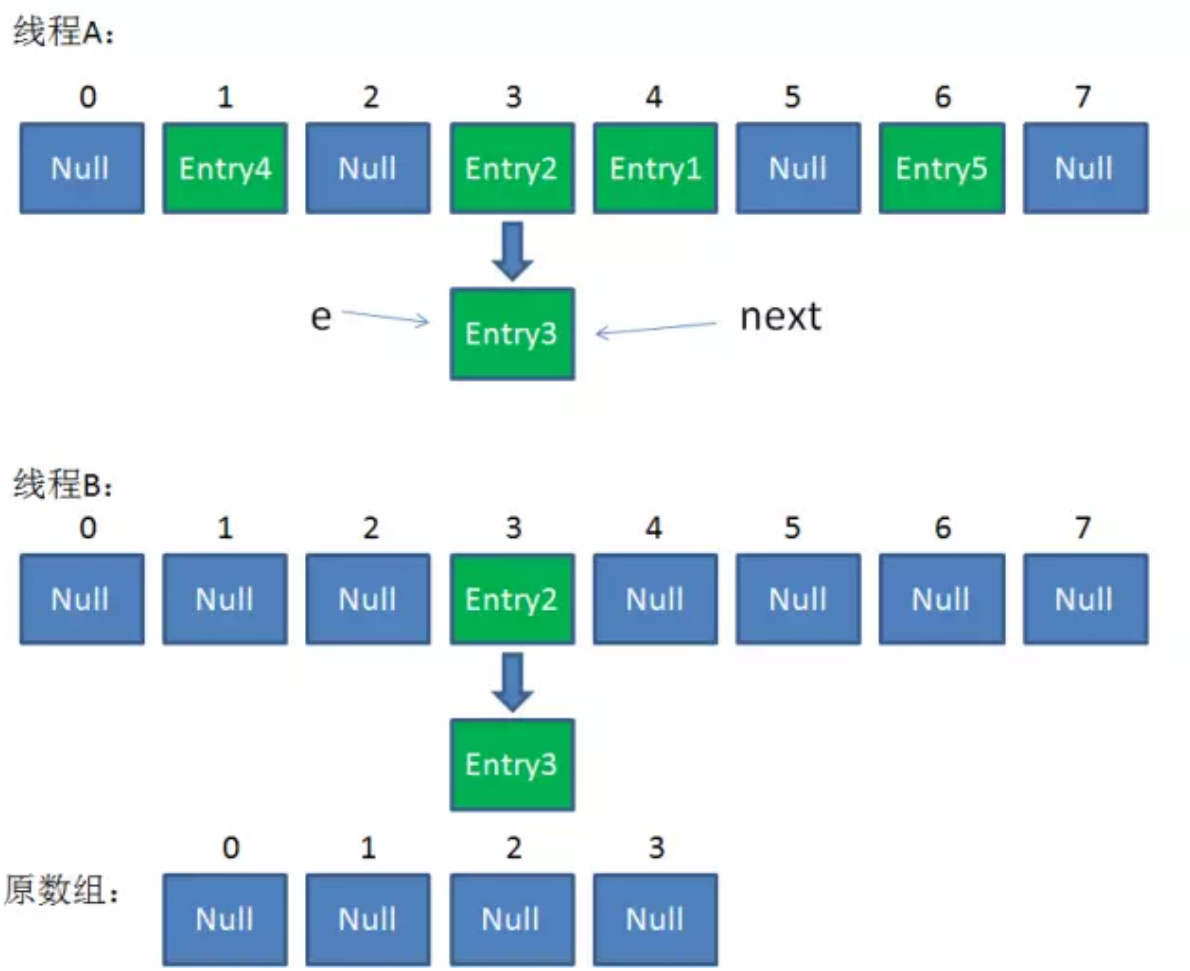
- 先介绍一下你自己吧
  - 略……
- 你做的项目中，你觉得最有挑战的一件事是什么事啊？
  - 我觉得最有挑战的是一次商品搜索功能的实现，当时本来是准备用`Solr`做的，但是当时的项目用的是`SpringBoot`，之前学Solr的时候是用`Solr`是和`Spring`整合，我用`SpringBoot`集成`Solr`发现版本差异很大，API也不用了，然后`Solr`初始配置繁琐，当时就想换一个搜索框架。然后我听说`Elasticsearch`挺好用的，于是就从0开始研究它的API，最后实现了商品搜索必需的一些功能，如关键字检索、结果高亮、聚合等。虽然我对ES的底层了解不多，但这一次是我独立对一个陌生框架的学习并实现了需要的基本功能。
- 就是你觉得用ES替换Solr比较有技术含量是吧？
  - 不是的。只是说有这么一个难题吧，现在Solr没办法用了，要逃离舒适区接触一个陌生的ES，从无到有的将商品检索功能实现，不说有多么高性能，起码实现了基本的功能。
- ES的源码，你有看过吗？
  - emmm……，没有。

> 在自己的简历中写一些不知底层原理没有读过源码的花里胡哨的框架技术等于给自己埋坑

- Java的源码有没有看过？
  - Java的源码，多线程的看过一些，集合的也看过一些。
- 那你给我讲一下`ConcurrentHashMap`和`HashMap`的区别。
  - `ConcurrentHashMap`的核心思想就是降低锁粒度以提高并发性能。`HashMap`不是线程安全的，`HashTable`虽然是线程安全的，但其相对于`HashMap`来说，只是简单地将每个访问集合的方法都加了一个`synchronized`关键字，也就是说任何访问集合的线程都需要先获取集合对象对应的锁，这样的话同一时刻只能有一个线程操作Map，并发性能弱，如果并发线程较多还会引起多次上下文切换。而`ConcurrentHashMap`采用锁分段的技术，默认在集合内部维护了16把锁，每几个`Bucket`作为一个组，每个组对应一把锁，这样最多就能支持16个线程同时访问Map了。
- 每个Bucket对应一把锁是吧？
  - 我记得好像是一把锁对应几个Bucket
- `HashMap`在并发场景下，它不安全的点是哪里？
  - 就是可能会造成Bucket中的链表形成环形链表，导致后续的Map.get操作可能会造成无限循环，导致CPU的100%占用。
- 仔细说一下形成环形链表的过程。（解析参考）
  - `HashMap`在元素个数到达阀值（容量x复杂引子）后会进行扩容，首先新建一个扩容后的空Map，然后遍历Entry将其Rehash到新的Map上。Rehash（对应方法`transfer`）中有一行代码是`Entry next = e.next`，其中`e`是当前遍历到的Entry。假设现在有两个线程A、B都在Rehash，B遍历到Entry3刚得到next（比如Entry2）时就消耗完时间片被挂起了，此时对于B来说`e=Entry3,next=Entry2`。这时A抢到CPU执行权，畅通无阻的执行完了Rehash操作：



- 然后B恢复执行，将Entry3重新哈希到了自己新Map的第3个桶上，并且`e=next`指向了Entry2，`next=e.next`指向了Entry3，于是将Entry2重新哈希到自己新Map的第3个同上，发现桶里已有Entry3，于是将Entry3的next指针指向Entry2，这在A中的新Map上表现为Entry2和Entry3形成了环形链表：



其实我当时答得并不准确……

- 桶里底层放的数据结构是怎样的？
  - 低版本的放的是链表，后来为了避免哈希冲突带来的时间复杂度线性增长改为了红黑树。
- ConcurrentHashMap和HashMap，他们除了一个线程安全一个线程不安全，还有其他的区别吗？
  - emmm……一下想不起来
- 看到你项目中写使用消息中间件实现短信发送功能的请求方和发送短信方之间的解耦，用的是什么中间件？
  - activemq
- 如果你发短信，发送失败了怎么办？
  - 这个没有考虑过
- 那叫你优化的话，你怎么优化？
  - 我觉得发送短信方无论是否发送成功都要给请求方一个ack，也就是要有一个反馈，让请求方知道。
- 所以，具体该怎么做？请求方将号码给到mq，短信发送方怎么做？发送成功了怎么做、失败了又怎么做？ack怎么做？一套流程，你说个1、2、3、4、5步出来。
  - 完全懵了……跟着黑马项目视频做的项目，工程能力是真的差！根本没有思考业务如何实现、为何这样实现、这样实现后会不会有什么问题、有问题又该如何解决。

> 不要觉得项目用了很多框架就很厉害，如果面试官问你是否看过源码、如何排除故障、如何解决故障你却答不上来，那这个项目就是在给自己埋坑。简历上的项目最好写那种有一些自己思考在里面的，而不是培训机构那些填鸭式的项目。

- 好的，没事，说不上来也没关系。（面试官人很好，多次鼓励和引导我，感谢：））
- 什么是双亲委派机制？

    - 是JDK类加载的一个模型。JDK有三个类加载器：用来加载核心类库的引导类加载器、用来加载扩展类库的扩展类加载器、用来加载自定义类的应用程序类加载器。双亲委派机制就是当JVM收到一个类加载请求时，不会直接让当前类的类加载器加载该类，而是将请求委派给其上层类加载器，直到请求到达最顶端的引导类加载器，然后逐层向下递归加载该类，某一层找到了该类则直接加载并返回，否则如果到了当前类的类加载器还找不到，就会抛出没有该类的异常。
- 双亲委派有什么好处?
    - 避免自定义的类覆盖核心类库的行为。比如我们自己也可以定义一个`java.lang.Object`，如果没有双亲委派机制，那么`java.lang.Object`的类加载请求就可能被应用程序类加载器受理，它则会加载我们自定义的`java.lang.Object`从而屏蔽了核心类库的Object，这样的话会损害JVM
- 有几种情况会导致Full GC
    - 第一就是分配担保时担保失败。在进行Minor GC时，新生代如果采用复制算法会将Eden和From Survivor中的存活对象复制到To Survivor，To Survivor空间不够时会让老年代进行分配担保，一种极端的情况就是新生代的对象全部存活，意味着会有很多对象进入老年代，如果这时老年代没有足够的可用空间，将导致Full GC
- 嗯，你说的这是老年代空间不足会导致Full GC，还有其他情况吗?
    - emmm……一时想不起来了
- `synchronized`

  修饰一个类，那么是不是这个类所有的方法都会加锁?
    - `synchronized`的语法规则是只能修饰方法和代码块
- `synchronized`

  修饰静态方法和实例方法有什么区别?
    - 一个锁定的是当前类的Class对象，一个锁的是当前this对象
- 那你给我讲一下

  `synchronized`

  锁的膨胀过程是怎样的?
    - 您指的是偏向锁、轻量级锁和重量级锁吗？就是如果临界区只有一个线程访问，这时会将锁对象的Mark Word中的偏向线程ID指向该线程，此后该线程进入临界区将省去锁获取-释放，毕竟锁获取-释放是有开销的。但是如果访问临界区的线程变多了，这时撤销偏向和重置偏向会有一定的开销，因而膨胀成轻量级锁，思路是会在线程的栈中存一份锁对象的Mark Word副本，称之为Displaced Mark Word，并该内存的指针存入锁对象的Mark Word。每个线程在获取锁的时候都需要CAS替换该指针，使其指向自己栈中的Displaced Mark Word。CAS替换失败则说明并发程度较高，膨胀成重量级锁，具有排他性。
- CAS有什么缺点嘛?
    - CPU开销较大，因为自旋会一直占用CPU，如果CAS一直更新不成功那么就会一直占用CPU。
- 还有吗?
    - emm……想不起来。（ABA问题啊，当时紧张没想到……）
- 什么是聚集索引，什么是非聚集索引?
    - 是指BTree和B+Tree吗?
- 不是，你说的只是索引的实现方式
    - emm不太清楚，应该是索引关键之和对应的记录不放在一起吧
- 我现在有一张表，里面有ABC三个字段，有一个ABC三列联合索引，根据AB两个字段去查能否利用索引?

    - 不能，因为复合索引只对第一个声明的字段有效
    - 哦，不对，可以利用索引。因为索引是先按A有序然后在按B有序最后再按C有序的。根据AB两个字段查，会在整体按A有序的情况下快速定位A条件的对应区间，然后在这个区间中A平等B有序又可以快速定位B条件对应的区间。
- DB的事务怎么实现的，你有了解吗？
    - 不太了解……
- MySQL的主从同步怎么做的
    - 读写分离、数据源动态切换。（错了，应该是在master执行的每个sql语句都会被记录在日志中发往slave执行）
- 系统间解耦，不用MQ，还有其他方式吗
    - 远程服务调用吧，如dubbo或者是Http
- 那你这样就没有解耦了，远程调用时同步的
    - 不太能想出来
- 那我提醒你一下，中间件是一个独立的系统，你要确保有一个独立的数据源在那就行了
    - redis好像可以吧
- 但是redis是缓存啊，不稳定，你为什么不直接说mysql、db就可以了
- 面试大概到这里就可以了，你有什么想问我的？
    - 我还有下次面试的机会吗？
- 这需要等评估在做决定，有第二面的话会在一个星期内联系你，过了一个星期没消息的话就没过。
    - 您能对我此次面试的表现提些建议吗？
- 系统中用到的东西，底层的实现原理，该如何优化，为什么要优化，这些都要搞清楚，不然一问你项目就答不上来的话不太好，实现逻辑要搞清楚。
    - 好，谢谢老师