**一、Java相关**

* 乐观悲观锁的设计，如何保证原子性，解决的问题；

悲观锁(Pessimistic Lock), 顾名思义，就是很悲观，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会block直到它拿到锁。传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。

乐观锁(Optimistic Lock), 顾名思义，就是很乐观，每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断一下在此期间别人有没有去更新这个数据，可以使用版本号等机制。乐观锁适用于多读的应用类型，这样可以提高吞吐量，像数据库如果提供类似于write\_condition机制的其实都是提供的乐观锁。

两种锁各有优缺点，不可认为一种好于另一种，像乐观锁适用于写比较少的情况下，即冲突真的很少发生的时候，这样可以省去了锁的开销，加大了系统的整个吞吐量。但如果经常产生冲突，上层应用会不断的进行retry，这样反倒是降低了性能，所以这种情况下用悲观锁就比较合适。

* char和double的字节，以及在内存的分布是怎样；
* 对象内存布局，然后讲下对象的死亡过程？
* 对象头，详细讲下；

HotSpot虚拟机的对象头（Object Header）分为两部分信息，第一 部分用于存储对象自身的运行时数据，如哈希码（25）、GC 分代年龄（4）、锁标志位（2）、1bit固定为0，这部分数据的长度在32位和64位的虚拟机中分别为32bit和64bit，官方称它为"Mark Word"，它是实现轻量级锁和偏向锁的关键。另外一部分用于存储指向方法区对象类型数据的指针，如果是数组对象的话，还会有一个额外的部分用于存储数组长度。

* sync原理详细，sync内抛异常会怎样，死锁吗？还是释放掉？怎么排查死锁？死锁会怎样？有没有什么更好的替代方案？

抛异常后会释放锁；

JConsole，JProfiler

* 详细讲一下集合，HashSet源码，HashMap源码，如果要线程安全需要怎么做？
* 多线程是解决什么问题的？线程池解决什么问题？

多线程有点：资源利用率更好（特别是CPU）；程序设计在某些情况下更简单；程序响应更快。

线程池：

**第一**：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。   
**第二**：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。   
**第三**：提高线程的可管理性。

* 线程池，如何设计的，里面的参数有多少种，里面的工作队列和线程队列是怎样的结构，如果给你，怎样设计线程池？
* AQS原理，ReentranLock源码，设计原理，整体过程。
* 继续聊多线程源码，sync原理，然后一个场景设计题；
* float f = 1.4f;double d = 1.4d; 与 float f = 1.5f;double d = 1.5d; 是否为true，内存是怎样的；

1.4f != 1.4d;

1.5f ==1.5d;

* split的源码，split("a|b|c");得出多少个数组；

遇到a或b或c就会拆分，如果abc连在一起会多2个空数组。

* 把所有认识熟用的JUC( java.util.concurrent(简称JUC)包)下的类写出来，讲下使用，然后讲下原生的线程操作;
* 开闭原则，解析工厂方法模式，建造者模式，区别。手撸出来。

开闭原则：软件实现应该对扩展开放，对修改关闭，其含义是说一个软件实体应该通过扩展来实现变化，而不是通过修改已有的代码来实现变化的。

* 讲下JVM的大页模式，JVM内存模型;
* 什么是敏捷开发，防御性编程，并行编程。Team Leader的思考;

**敏捷是态度而不是流程，是氛围而不是方法。**

“防御性编程”——它更多的，是一种努力，是一种编程习惯。

防御性编程是一种细致、谨慎的编程方法。为了开发可靠的软件，我们要设计系统中的每个组件，以使其尽可能地“保护”自己。我们通过明确地在代码中对设想进行检查，击碎了未记录下来的设想。这**是一种努力**，防止（或至少是观察）我们的代码以将会展现错误行为的方式被调用。

防御性编程**是一种编程习惯**，是指预见在什么地方可能会出现问题，然后创建一个环境来测试错误，**当预见的问题出现的时候通知你，并执行一个你指定的损害控制动作**，如停止程序执行，将用户重指向到一个备份的服务器，或者开启一个你可以用来诊断问题的调试信息。

|  |
| --- |
| 1> 使用好的编码风格和合理的设计  2> 不要仓促地编写代码  3> 不要相信任何人  4> 编码的目标是清晰，而不是简洁  5> 不要让任何人做他们不该做的修补工作  6> 编译时打开所有警告开关  7> 使用静态分析工具  8> 使用安全的数据结构  9> 检查所有的返回值  10>审慎地处理内存（和其他宝贵的资源）  11>在声明位置初始化所有变量  12>尽可能推迟一些声明变量  13>使用标准语言工具  14>使用好的诊断信息日志工具  15>审慎地进行强制转换  16>其他  a> 审慎地进行强制转换  b> 提供默认的行为  c> 检查数值的上下限 |

* 逃逸分析是什么，作用是什么，用途是什么;

分析对象的动态作用域：当一个对象在方法中被定义后，他可能被外部方法所引用，作为参数传递到其他方法中去，称之为方法逃逸。如果被线程访问到，就叫线程逃逸。

1.栈上分配替代堆上分配，用完就释放内存；2.同步消除，提高性能；3.标量替换，为了栈上分配，以及进一步优化做准备。但是逃逸分析目前十分不成熟。

* 怎么认为一个类是线程安全？线程安全的定义是什么？Java有多少个关键字进行同步？为什么这样设计？（聊了一大堆，一堆为什么）；

所有只读的类都是线程安全的，线程安全问题都是由全局变量及静态变量引起的。

单线程访问都是安全的。

* 两个线程设计题。记得一个是：t1,t2,t3，让t1，t2执行完才执行t3，原生实现。
* 写个后缀表达式，为什么要设计后缀表达式，有什么好处？然后写下中缀。
* 我看你做过性能优化，比如你怎么分析项目里面的OOM的，内存泄露呢？详细说思路;
* 说下多线程，我们什么时候需要分析线程数，怎么分析，分析什么因素;
* 抽象方法和类方法的区别，static的抽象方法可以吗？

不可以static抽象方法。

* 说下Java的克隆体系;

浅克隆、深克隆。序列化与反序列化。

* 涉及OOM、JVM优化、源码问题、数据库优化、多线程等问题;
* CPU高？什么情况CPU高？解决什么问题？
* 你有遇到过临界区问题吗？有遇到过吗？你在项目遇到这个问题是怎样解决的？
* volatile关键字作用;

可见性、顺序性。

* Java的多态怎么实现;
* 解释一下自旋;（多线程竞争）

互斥同步对性能最大的影响是阻塞的实现，挂起线程和恢复线程的操作都需要 转入内核态中完成，这些操作给系统的并发性能带来了很大的压力。由于多线程并发的时候就是等待锁的释放，这个时间是很短的，如果为了这段时间来挂起和恢复线程并不值得。所以我们让线程执行一个忙循环来等待锁的释放，这个忙循环就叫做自旋。这项技术就是紫选锁。

* 解释一下信号量;
* Semaphore是用来保护一个或者多个共享资源的访问，Semaphore内部维护了一个计数器，其值为可以访问的共享资源的个数。一个线程要访问共享资源，先获得信号量，如果信号量的计数器值大于1，意味着有共享资源可以访问，则使其计数器值减去1，再访问共享资源。
* 如果计数器值为0,线程进入休眠。当某个线程使用完共享资源后，释放信号量，并将信号量内部的计数器加1，之前进入休眠的线程将被唤醒并再次试图获得信号量。使用了信号量还是要用锁的。
* 什么情况下会触发类加载；
* Java内存抖动严重，优化的思路；

**二、数据库相关**

* SQL优化思路，联合索引与底层树结构的映像关系，索引结构（B+、B-），为什么用这样的结构；
* 讲下MySQL的集群？集群遇到过什么问题？sql的优化？
* 你目前为止遇到的最大数据量是多少？知道100万时候怎么设计吗？1000万呢？过几十亿呢？
* MySQL有多少个参数可调，除了最大连接数。全部列出来，一个个分析。
* 聊下优化过的索引，怎么优化;
* 红黑树和平衡树的区别，为什么数据库不用红黑树;

|  |
| --- |
| * 1、红黑树放弃了追求完全平衡，追求大致平衡，在与平衡二叉树的时间复杂度相差不大的情况下，保证每次插入最多只需要三次旋转就能达到平衡，实现起来也更为简单。  2、平衡二叉树追求绝对平衡，条件比较苛刻，实现起来比较麻烦，每次插入新节点之后需要旋转的次数不能预知。  AVL树是最早出现的自平衡二叉(查找)树 * 红黑树和AVL树类似，都是在进行插入和删除操作时通过特定操作保持二叉查找树的平衡，从而获得较高的查找性能。 红黑树和AVL树的区别在于它使用颜色来标识结点的高度，它所追求的是局部平衡而不是AVL树中的非常严格的平衡。 红黑树是牺牲了严格的高度平衡的优越条件为代价红黑树能够以O(log2 n)的时间复杂度进行搜索、插入、删除操作。 此外，由于它的设计，任何不平衡都会在三次旋转之内解决。 当然，还有一些更好的，但实现起来更复杂的[数据结构](http://lib.csdn.net/base/datastructure)能够做到一步旋转之内达到平衡，但红黑树能够给我们一个比较“便宜”的解决方案。 红黑树的[算法](http://lib.csdn.net/base/datastructure)时间复杂度和AVL相同，但统计性能比AVL树更高. |

|  |
| --- |
| 为什么使用B+树？言简意赅，就是因为：  1.文件很大，不可能全部存储在内存中，故要存储到磁盘上  2.索引的结构组织要尽量减少查找过程中磁盘I/O的存取次数（为什么使用B-/+Tree，还跟磁盘存取原理有关。）  3.**局部性原理与磁盘预读，预读的长度一般为页（page）的整倍数，（在许多操作系统中，页得大小通常为4k）**  **4.数据库系统巧妙利用了磁盘预读原理，将一个节点的大小设为等于一个页，这样每个节点只需要一次I/O就可以完全载入，(由于节点中有两个数组，所以地址连续)。而红黑树这种结构，h明显要深的多。由于逻辑上很近的节点（父子）物理上可能很远，无法利用局部性** |

* mysql有哪些锁，意向锁有什么用;
* 数据库高并发下的优化思路;
* 数据库什么情况下索引会失效;

**三、数据结构和操作系统相关**

* 数据结构学过吧，聊一下？学过什么结构？讲下树和队列？B树呢？
* 操作系统学过吧，聊一下？讲一下系统内存是怎样的？分段分页虚拟内存？
* 页面置换算法呢？多少种？有最优的置换算法吗？
* 你学过什么课程？然后聊下操作系统，内核、用户之类。
* 反转链表手撸;
* 快排，给一串数组，把具体每次patition写下，最终结果也写45, 32, 41, 35, 38, 20, 50;
* 一个整数status, 判断第K个比特位是否为比特1;
* 把递归实现的快排改成非递归，你知道非递归有什么好处吗;
* 举例使用分治思想的算法;
* （1）二分搜索
* （2）大整数乘法
* （3）Strassen矩阵乘法
* （4）棋盘覆盖
* （5）合并排序
* （6）快速排序
* （7）线性时间选择  
  （8）最接近点对问题
* （9）循环赛日程表
* **（10）汉诺塔**

**四、网络相关**

* 讲下请求头细节？
* Http和Https？Http1.0,1.1,2.0，讲下长连接和短连接？Https是怎样的？如果我篡改了公钥呢？怎么防止？
* Get和Post，讲下区别，要我模拟出抓包来。
* 详细讲下Cookie和Session，Token，OAuth2.0协议;
* 拥塞算法知道吗？哪些，分别怎样？
* 学过计算机网络是吧？socket熟悉吗？对它的读写缓冲区有理解吗？怎么的？那滑动窗口是怎样的？为什么这样设计？
* 再聊下Http的Http basic authentication;
* Https的过程;

**五、框架相关**

* 聊下Spring源码，知道多少，都聊一下；
* 聊下Spring注解，@Autowire，@Resource，以及他们的解析过程；
* 聊一下架构，接入层架构，服务层架构。聊下技术栈，Spring Boot，Spring Cloud、Docker；
* Spring ioc的具体优势，和直接New一个对象有什么区别;
* Servlet生命周期，是否单例，为什么是单例;
* Spring Mvc初始化过程；

**五、分布式相关**

* 多少种RPC框架？

GRPC,BRPC,DUBBO,THRIFT,RMI,HESSIAN

* 一致性哈希是干嘛的？
* 搭建高并发高可用系统需要怎样设计？考虑哪些东西，有多少说多少。
* 你对缓存有什么理解？缓存是解决什么问题？后端缓存有哪些，分别解决什么问题？
* 聊一下分布式锁；
* 你是怎么设计系统缓存的，为什么，什么场景；
* 也来说下，削峰的多种实现，Redis？MQ？
* 为什么用mq就能削峰？解决什么问题？

**六、设计题**

* 有几台机器存储着几亿淘宝搜索日志，你只有一台2g的电脑，怎么选出搜索热度最高的十个搜索关键词;

1.分批处理，然后把结果写入磁盘；2.将结果合并写入磁盘；3.搜索磁盘10次；

* 如何设计算法压缩一段URL;
* 既然是短链接，那么地址池是有限的。所以在业务上会要求接受短链接失效，保证可回收利用。但是总有一些场景要求长期有效，所以可预留一些特殊的。  
  以上是最基本的系统功能要求。  
  至于技术上怎么做，太简单了。DB索引，随机都可以，搞个分库分表，读多写少。随机的时候碰撞了就重试几次，没什么问题。在对总短链接数做个监控，地址用的太多会造成随机碰撞越来越多，也可能是回收机制出问题了。  
  另外防攻击做好。别个把你链接池地址耗光。

2.自增ID转换成62进制。

* 有一个页面能同时展示两个广告，现在有五个广告，设计算法使五个广告展示概率为1:2:3:4:5；
* 有25匹马，五个赛道，用最少比赛次数将25匹马排序；
* 1，假设最快的5匹马都分在同一组
* 2，假设最快的5匹马分别为各组的第一名

**七、其他相关**

* Tomcat缓存，聊下缓存的整体理解，知道多少种缓存；
* 解释下Mucene原理，倒排索引，怎样进行中文分词，基于什么进行分词；
* TopN的大数据量题；
* 你对接入层要思考什么东西？遇到过哪些问题？搭建系统要考量哪些因素？
* 然后项目问题，优化问题；
* 熟悉maven是吧？我们来聊下Maven的源码原理，Maven冲突的时候，怎么选择依赖包，我们怎么查，我们遇到两个不一样的版本，我们应该如何去选择，为什么？
* 项目如何分组，性能优化小组应该做哪些;
* 我们来说下接入层的搭建，认知分析;
* 问下项目的系统构建，思考，为什么这样构建？
* 如何判断一段代码的好坏;