一、请简单描述进程和线程，简述它们的区别和联系

进程是程序运行的活动实体，是资源分配的最小单位，操作系统会为一个进程分配PCB（进程控制块）、进程空间（虚拟地址空间）、页表等资源，不同的进程独占资源，互不影响，进程的创建、撤销需要付出较大的代价，进程间的切换也需要付出较大的代价。

线程是轻量级的进程，是CPU调度的最小单位，进程中的线程共享一个地址空间，线程只占有少量资源，包括线程ID、私有栈空间、上下文数据等，线程的创建和撤销、切换的代价较小。

多线程的优缺点

（1）线程是轻量级的进程，携带资源少，创建代价小

（2）线程切换代价小

（3）并发性较高，充分利用CPU资源

（4）对于计算密集型和IO密集型任务的效率高

缺点：安全性、编程难度

二、进程间通信

1、管道（pipe，半双工），流管道（s\_pipe，全双工），有名管道（FIFO，全双工）

2、信号量（sophomore/mutex）

3、消息队列

4、共享内存

5、套接字

二、线程同步的方式

本文档目的在于对自己写的内容有个大概的简要总结，以免忘记这些项目内容，同时对一些内容进行详细注释，以备在面试问起时及时回应

常见用的算法和数据结构

红黑树

5大性质：a、每个节点红色或黑色b、根节点黑色c、叶子节点为黑色，注：叶子节点指代null节点d、红色节点的子节点必须是黑色的e、从根节点到任意叶子节点的路径上具有相同数目的黑色节点

插入：插入的节点是红色的

通过不同的情况，通过旋转操作使红黑树保持其性质

能保持较高效率的原因是，红黑树是相对平衡的，即保证任意每个分支长度差距不大

机器学习算法

聚类算法：k-means，基于密度的聚类方法DBSCAN，层次聚类

分类算法：朴素贝叶斯，决策树，SVM（支持向量机），KNN（K近邻），神经网络

数据库索引结构

B-树、B+树、哈希索引、k-d树、四叉树、R-树

必须掌握的，B-树，B+树、k-树、哈希索引、四叉树

B树：任意非叶子节点最多有M个儿子；M>2，根节点外的非叶子节点的儿子数位[M/2,M]，每个节点存放的关键字数[M/2,M]，所有叶子节点位于同一层

B+树，所有关键字都在叶子节点中，更适合于文件系统，查找最终必须到达叶子节点

B+树的优点：由于B+树再叶子节点中不包含数据信息，因此在非叶子节点能存放更多的 key，数据存放更加紧密，具有更好的空间局部性，访问叶子节点上的关联数据具有更好的缓存命中率。

B+树的叶子节点是相连的，因此，对整棵树的遍历只需要进行依次线性遍历叶子节点即可，而且由于数据顺序排列且相连，所以便于取间查找和搜索。而B树需要对每一层进行递归遍历。

B+树的优势在于查找效率上

B+树相比B树的优势：

1. 单一节点存储更多的元素，使得查询的IO次数更少，即盘块能容纳的关键字数量多
2. B+树的范围查询效率要比B树的查询效率高，因为B+树可以对叶子节点进行遍历

　　2.所有查询都要查找到叶子节点，查询性能稳定；

　　3.所有叶子节点形成有序链表，便于范围查询。

B树，可以用于范围查询

哈希索引：基于哈希表实现，只能支持等值查询，查询速度快，对于重复很多的键值效率不高

项目经历：

正在进行的项目

LSMTree，Dostoevsky

LSMTree的应用，key-value型存储系统，牺牲了查询和存储空间，提高了写入速度，适用于写频率高的数据库

LSM树的核心思想，同cache类似，将数据库分为两部分，较小的在内存中，较大的在磁盘中，当出现更新、插入时，将key插入到内存中的部分，当内存部分达到一定大小时，则合并到磁盘中的部分。两种Merge方式的不同，第一种是，每次满了则merge，其它方式，到最后再merge

WAH位图压缩算法和图的位图存储

面试总结之开发岗

1. 开机过程，这是腾讯的面试问题

先从宏观的角度讲一下开机过程，即CPU加电到操作系统启动过程

1. 首先计算机通电后，从ROM中读取开机程序，即BIOS，这个叫做基本输入输出系统
2. 会把所有的寄存器设为默认值
3. 硬件自检：BIOS程序首先进行检查，判断计算机硬件是否能满足基本开机条件，这叫做硬件自检（POST，Power-on Self Test），如果硬件出现问题会有蜂鸣
4. 启动顺序
   1. 硬件自检完成后，BIOS会把控制权转交给下一阶段的启动程序，在BIOS中可以选择不同的启动设备
5. BIOS按照启动顺序，读取排在第一位的存储设备的第一个扇区，主引导记录，MBR，只有512个字节，MBR的最后两个字节是0x55和0xAA。MBR被读取到内存0x7C00处
6. 主引导记录分为三部分，
   1. 第一部分：调用操作系统的机器码，主引导程序
   2. 第二：分区表，
   3. 第三部分：主引导记录签名
7. 分区表：将硬盘分成若干个区，4个主分区中，有一个是激活的
8. 计算机读取激活分区的第一个扇区，叫做扇引导记录VBR，这个的作用是告诉计算机，操作系统在这个分区
9. 启动管理器，运行启动管理器，由用户选择启动哪一个操作系统

.Struct和Union的区别

Struct 可以存储多个成员，不同成员都占用一定的空间

Union不同成员共用一个存储空间，只存放了一个被选中的成员

线程和进程

不同线程的全局变量是共享的，可以互相访问，即读取，也可以写，

不同线程使用不同的栈空间

栈：先进后出

队列：先进先出

最大堆：堆顶元素比子节点大

最小堆：堆顶元素比子节点小

堆：

树

二叉查找树：左子树小于根节点，右子树大于根节点

AVL树：左子树和右子树高度差最大为1，

通过旋转来保持平衡

在插入节点时，AVL和红黑树都是最多两次旋转，但是在删除节点时，红黑树只需要O(1)的时间，而AVL树是O(logn)时间

AVL树的搜索效率比红黑树高

旋转次数比红黑树多，同时在实际应用的测试中，红黑树性能比AVL树好

如果搜索次数远大于插入和删除次数，则AVL树比红黑树好

线段树是一棵完美二叉树，树上的每个节点都维护一个区间。根维护的是整个区间，每个节点维护的是父亲的区间二等分后的其中一个子区间。当有n个元素时，对区间的操作可以在O（logn）的时间内完成

红黑树：最大高度和最小高度为2倍关系

C++中STL中的map和set都是红黑树实现的

Linux进程调度模块是用红黑树管理进程控制块的

I/O的多路复用epoll采用红黑树管理

Java中的TreeMap实现

哈夫曼编码和哈夫曼树：贪心策略

统计各个出现次数，并对出现次数进行排序

将最小的两个合并成一个节点，新的节点值为两个子节点值的和，直到只有一个节点

编码，往左0，右1

动态哈夫曼树编码

第i+1个字符的编码是由前i个字符建立的哈夫曼树为依据进行的，动态哈夫曼树没有固定的编码

因为哈夫曼编码压缩需要压缩扫描两次，第一次是构建哈夫曼树，第二次是压缩过程，而动态哈夫曼树的压缩只需要一次扫描，在扫描的同时建立哈夫曼树

Hashmap和红黑树

在JDK1.8版本后，如果链表长度大于8，后面的数据将存放在红黑树中

单链表：

如果元素小于8个，查询成本高，新增成本低

元素大于8个，查询成本低，新增成本高

散列与哈希（Hash）

1. 直接寻址：用关键字当作散列地址
2. 平方取中，关键字平方后取中间的几位
3. 随机数法
4. 折叠法
5. 保留余数法

处理散列冲突

1. 链接法：具有相同散列地址的记录存放在一条线性表中
2. 多哈希，多种哈希函数
3. 开放定址法：Hi = (H(key)+di) MOD m，如果di=1，则向后移动一位，如果di=1,-1,4,-4…则为二次再探测散列，di为随机数列，则称为伪随机探测

100亿的数据中找最大的k个数

方法1：建立一个大小为k的最小堆，扫描所有数，如果比堆的最小值小，则替换

方法2：将100亿个数分成1000个大分区，每个大分区1000万个数据；每个大分区再分成100个小分区，总共有1000\*100=10万个小分区，计算每个小分区上最大的k个数

方法3：Hash法，通过Hash法去除重复数

ARP协议：将IP地址解析为MAC地址

主机A的IP地址为192.168.1.1，MAC地址为0A-11-22-33-44-01；

主机B的IP地址为192.168.1.2，MAC地址为0A-11-22-33-44-02；

主机A要与主机B通信时：

第1步：主机A在本地ARP中，检查主机B的匹配MAC地址

第2步：没有找到主机B的ARP，则它将询问192.168.1.2的地址，将ARP请求帧广播到本地所有主机。源主机的IP地址和MAC地址都再ARP请求中。本地网络中的每台主机都将接受到ARP请求，并且检查是否与自己的IP地址匹配。如果不匹配，则它丢弃该请求包。

第3步：主机B接受到ARP请求后，将主机A的IP地址和MAC地址映射写到本地ARP缓存中

第4步：主机B将包含其MAC地址的ARP回复消息直接发送会主机A。

synchronized跟Lock的区别

当一个代码块被synchronized修饰时，当一个线程获取了对应的锁，并执行了该代码块时，其它线程只能一直等待，直到获取锁的线程释放锁。

所以，如果这个拥有锁的线程由于要等待IO或者其它原因被阻塞了，但是又没有释放锁，那么其它线程只能等待，影响程序执行效率

Lock知道线程没有成功获取到锁，Lock是一个类，通过这个类可以实现同不访问，synchronized不需要用户主动释放锁，而Lock必须由用户主动释放锁

mysql范式

第一范式：原子性，保证每一列不可再分

第二范式：主键，保证表必须有一个主键，除了主键外的其它列都依赖于主键

第三范式：一个关系中不包含已在其它关系中已包含的非关键字信息

操作系统进程调度算法

先来先服务、最短作业优先、相应比高优先（利于短作业）

时间片轮转调度（对系统的效率影响大）、优先级调度、多级队列、

浮点数的表示

阶码和尾数

R^ExM

浮点数规格化，最高位必须是一个有效值

左规和右规

0.1xxx表示正数

1.1xxx表示负数

浮点数加减运算：对阶、尾数求和、规格化、舍入、溢出判断

C语言关键字：

1、auto，2、short，3、int，4、long，5、floa，6、double，7、char，8、struct，9、union，10、enum，11、const，12、unsigned，13、signed，14、extern，15、register，16、static，17、volatile，18、void，19、else，20、switch，21、case，22、case，23、for，24、do，25、while，26、goto，27、continue，28、break，29、default，30、sizeof，31、return，32、if

C++malloc能分配的最大内存和地址空间有关，和实际内存大小无关，

秒杀架构的设计

核心思想，尽量在系统上游拦截流量，降低下游压力，不让请求落到数据库中，充分利用缓存和队列

在四个层面拦截：客户端、服务控制器网关、服务层、数据库层

客户端方案

1. 页面静态化2、禁止重复提交，设置重复提交时间限制3、对于用户的多次点击，只进行一次请求来欺骗用户

服务端控制器层（站点层）

限制uid和IP访问频率，对uid请求次数进行计数

对于大量的读请求，利用页面缓存分担读请求压力，对于同一个uid在某个时间段内只返回同一个页面

如果在这一层压力太大，则增加服务器数量，随机丢弃请求

服务层

采用消息队列缓存请求，利用缓存对应读请求，避免某些黑客拥有很多uid。

利用缓存分担部分数据库压力，利用写缓存对应请求。

数据库层

这是最脆弱的一层，一般设计时在上游把请求拦截掉，数据库层只承担能力范围内的访问请求。

服务设计，分段放票

秒杀后未支付如何处理，设定计时器，如果超时丢弃

分片，解决数据量太大的问题，数据路由，哈希路由

分组，通过复制数据解决可用性问题

分片和分组结合。

数据库的CAP特性，可用性、一致性和、分区容错性

如何保证数据的可用性

冗余来保证数据的可用性，复制服务，冗余服务

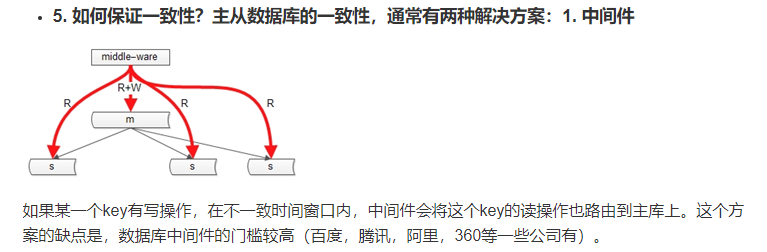
如何保证数据库的读高可用，冗余读库

但是会带来不一致的问题

写高可用，冗余写库，双写时可能产生数据冲突，解决方案是，两个数据库采用不同初始值，相同步长来增加id

实际应用中服务+数据库+缓存，会引发一致性问题

保证主从一致性，中间件方法



对于多个IP多个uid的请求，通过账号行为的数据挖掘来提前清理掉它们。

悲观锁和乐观锁

悲观锁：总是假设最坏情况，每次拿数据都认为别人会修改，所以每次拿数据都会上锁，这样别人想拿数据就会阻塞直到它拿到锁。Java里的synchronized关键字的实现就是悲观锁

乐观锁：假设每次拿数据别人都不会修改，但是在更新的时候会判断一下在此期间有没有人去更新这个数据，乐观锁适用于多读的应用类型，原子变量类使用的就是乐观锁

乐观锁的实现细节CAS，compare and swap

当多个线程同时更新同一个变量时，只有一个线程能更新变量的值，而其它线程都失败，失败的线程会被挂起

CAS操作中，包含三个数，要读写的内存位置V，进行比较的预期值A，拟写入更新的值B，如果内存位置V的值与预期原值A相匹配，则处理器会自动将该位置的值更新为新值B，否则不做任何操作。CAS会返回是否成功

资源竞争少时，选择CAS，因为CAS基于硬件实现，且不需要用户态与内核态的转换

资源竞争严重，选择synchronized，因为CAS自旋概率比较大，会浪费比较多的CPU资源

有哪些要问的？

如果是终面，你最后可以直接问，「您觉得，这份工作所需的能力，我还有哪些不具备？

如果我通过了，大概还有几次面试？您大概什么适合给我回复？可以加个微信吗？

为了胜任这个岗位我还需要学习哪些技术知识？

对象HR或用人单位领导

这个职位为什么要招人？为什么大家都想留在你们公司？请问您工作多久了？您觉得在这工作开心吗？