面试问题

1. StackOverflowError与OutOfMemoryError区别

栈溢出错误和内存不足错误

StackOverflowError 如果线程在计算所需要用到栈的大小大于配置允许最大的栈大小 那么Java抛出StackOverflowError错误

OutOfMemoryError 如果线程可以动态拓展本机方法栈 并且尝试本地方法栈拓展 并且没有大于配置允许的最大栈大小 但是内存不足以提供 则抛出 或者是 如果不能提供足够的内存来为新线程创建初始的堆 则抛出

解决方法 StackOverflowError 分配正确的内存空间 OutOfMemoryError 检查是不是对象过多 调整堆内存大小 通过jvm命令组合排查问题

常用jvm命令

Jps 查询当前用户下已经启动的Java进程 jps – lv

Jstack 查看java进程内当前时刻的线程快照 即方法栈使用情况 用于定位线程停顿 死锁等长时间等待的问题 jstack -l pid > stack.log打印到本地看

Jinfo 查看jvm配置参数 可以动态设置部分参数值

Jinfo -flags pid 查看jvm参数

Jinfo -flag 参数名 pid 可以查看指定参数值 比如看堆的最大值 -Xmx

Jinfo pid 查看jvm参数和系统参数

Jmap 查看给定进程 核心文件 远程调试服务器的共享对象内存映射和堆内存细节工具 可以查看堆的使用情况 堆内对象直方图 加载类 生成堆快照

Jmap -heap pid 打印堆配置 新生代 老生代信息

Jmap -clstats pid 打印类加载器信息 包括名称 个数 占用空间 父加载器 是否存活 类型信息

Jstat 通过回收相关信息来判断jvm性能问题 通过固定的时间间隔输出来观察 通过动态的变化观察gc是否过于密集

1. 一条更新的SQL语句是如何执行的

取出满足条件的所有行 然后将这些行操作 并写入内存当中

此时进行两段式提交

1. Update语句的结果写入内存同时将这个操作写入redo log 此时redo log处于prepare状态 并告知执行器随时可以提交事务
2. 执行器生成这个操作的binlog 并写入binlog日志 并状态改成commit状态 更新完成

Redo log是为了解决crash-safe问题产生的 是一种物理日志 是InnoDB引擎层的一种日志

Bin log是一种逻辑日志 是server层的一种日志 记录了所有SQL语句 用于备份和恢复数据库

1. 线程池参数详解

coolPoolSize 默认情况下是没有线程的 而是等到任务来才创建线程去执行任务 当线程数到达coolPoolSize后，就会把到达的任务放到缓存队列里。

但是核心线程在allowCoreThreadTimeout被设置为true时会超时退出，默认不退出。

maxPoolSize当线程大于或等于核心线程且任务队列已满时，线程池会创造新的线程，如果到达了maxPoolSize，那么就超出了线程池的处理能力，就会拒绝处理任务并抛出异常。

KeepAliveTime 当线程空闲时间达到KeepAliveTime该线程会退出 直到线程数量等于corePoolSize 如果allowCoreThreadTimeout为true，则线程数会为0默认为false

queueCapacity任务队列容量

workQueue一个阻塞队列，用来存储等待执行的任务，通常用LinkedBlockingQueue和SynchronousQueue。线程池排队策略与BlockingQueue有关。

线程池按以下行为执行任务

1. 当线程数小于核心线程数时，创建线程。
2. 当线程数大于等于核心线程数，且任务队列未满时，将任务放入任务队列。
3. 当线程数大于等于核心线程数，且任务队列已满
   1. 若线程数小于最大线程数，创建线程
   2. 若线程数等于最大线程数，抛出异常，拒绝任务
4. MySQL是如何实现四大隔离级别

可串行化：MySQL执行S2PL并发控制协议，一阶段申请，一阶段释放。读写都需要加锁

可重复读：读不加锁，只有写才加锁，读写互不阻塞，并发度对于可串行化级别要高，但会有写偏斜异常【如果是串行执行则不会发生】

读已提交：读使用了MVCC机制【读取最新快照】，写数据加排他锁

读未提交：读不会加锁，写会加排他锁，并且到事务结束之后释放

锁：

Shared Locks S锁：如果事务T对数据A加上S锁，则事务T只能读A，其他事务也只能给A加S锁，不能加X锁。保证了可以被读，不能被改

Exclusive Locks X锁：如果事务T对数据A加上X锁，那么可读可写，其他事务不能操作

Record Locks 行锁：加在索引行上的锁 id = 10

Gap Locks 间隙锁：锁住两个索引之间的区域 id< 10

Next-key Locks 间隙锁：闭合区间 id<= 10

组合起来就有行级共享锁，表级共享锁，行级排他锁，表级排他锁

1. Linux中的各种锁

自旋锁：在临界区加锁的时候，不断自循环直到可以获取到执行权。可以避免上线切换，开销小，在内核中使用得多。

互斥锁：对共享数据加锁，访问完成后进行解锁操作，在锁的时候，其他使用者没有释放权限。加锁后，任何试图加锁的线程会被阻塞，直到当前进程解锁，线程还必须被唤醒。

读写锁：共享互斥锁，就是读模式共享，写模式互斥。有三种状态，读加锁，写加锁和不加锁状态。读写锁默认是读优先，就是当读写锁在读模式时，任何线程可读，写操作的线程被阻塞。

【用得少，只有路由表的查询更新等设备状态更新】RCU锁：支持多读多写同时加锁。Read Copy Update Lock 在写者访问时，先拷贝一个临界区副本，然后对副本进行修改。RCU机制将在在适当时机使用一个回调函数吧指向原来临界区的指针重新指向新的被修改的临界区，锁机制中的垃圾收集器负责回调函数的调用。

可重入锁和不可重入锁：即是递归锁和非递归锁，同一个线程可以可以多次获取同一个递归锁，不会产生死锁。而一个线程多次获取同一个非递归锁，则会产生死锁。

条件变量：用来等待线程而不是上锁与互斥锁使用，生产者-消费者模型。

1. Java内部类的四大作用
2. 内部类可以很好的实现隐藏 一般非内部类是不允许有private何protected权限的，但内部类可以
3. 内部类拥有外围类的所有元素的访问权限
4. 可以实现多重继承
5. 可以避免修改借口而实现同一个类中的两种同名方法 即用内部类来实现接口，避免和外围类的方法冲突
6. Linux中排查CPU高占用率
7. 用top命令查看进程的CPU，内存等资源的使用情况 如果有消耗大的进程 直接输入k和PID就可以终止进程。如果CPU很空闲，但负载比较高，说明任务队列越长，处于等待执行的任务越多。可能是僵死进程导致，只能恢复其依赖资源或者重启系统。
8. Ps –mp pid显示线程列表 并且将需要查看的线程的ID转化为16进制
9. Jstack pid 查看该线程的资源使用情况
10. Vmstat可以查看系统整体的CPU，内存的使用情况
11. Mybatis的缓存机制

Mybatis的提供查询缓存，用于减轻数据压力，提高数据库性能

Mybatis提供一级缓存和二级缓存：

一级缓存：在操作数据库的时候，构建SQLSession对象，在对象中使用HashMap用于存储缓存数据。不同的SQLSession的HashMap互不影响

在同一个SQLSession中两次执行相同的sql语句，并不用访问数据库。从而提高查询效率

二级缓存：多个SQLSession去操作同一个Mapper的sql语句。不同的SQLSession在同一Namespace下写入，就会被二级缓存读取，如果没有update 其他SQLSession再读取的时候就直接从二级缓存读取

1. 在浏览器输入URL后发生了什么
2. URL解析 判断URL是否合法，检查缓存，如果有缓存读取缓存，否则向服务器请求
3. DNS查询 浏览器缓存-》操作系统缓存-》路由器缓存-》客户端DNS服务器缓存-》根域名服务器缓存
4. TCP连接 三次握手
5. 服务器处理请求 将收到的数据解析验证封装，判断是否重定向，如果是返回301，不是的话查看URL从写规则，如果是个文件就直接返回文件，如果不是服务器会按照规则将请求重写到REST风格的URL上，根据动态语言的脚本调用动态文件处理请求
6. 浏览器接受响应 根据状态码处理 对资源做和缓存
7. 渲染页面

1. Cookies 和 session的区别

A screenshot of a cell phone

Description automatically generated

1. 产生死锁的**四个必要条件**：  
   （1） 互斥条件：一个资源每次只能被一个进程使用。  
   （2） 占有且等待：一个进程因请求资源而阻塞时，对已获得的资源保持不放。  
   （3）不可强行占有:进程已获得的资源，在末使用完之前，不能强行剥夺。  
   （4） 循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。