BIO、NIO 和 AIO 的区别

Java BIO ： 同步并阻塞，服务器实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理，如果这个连接不做任何事情会造成不必要的线程开销，当然可以通过线程池机制改善。

Java NIO ： 同步非阻塞，服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有 I/O 请求时才启动一个线程进行处理。

Java AIO： 异步非阻塞，服务器实现模式为一个有效请求一个线程，客户端的 I/O 请求都是由 OS 先完成了再通知服务器应用去启动线程进行处理。

NIO 比 BIO 的改善之处是把一些无效的连接挡在了启动线程之前，减少了这部分资源的浪费（因为我们都知道每创建一个线程，就要为这个线程分配一定的内存空间）

AIO 比 NIO 的进一步改善之处是将一些暂时可能无效的请求挡在了启动线程之前，比如在 NIO 的处理方式中，当一个请求来的话，开启线程进行处理，但这个请求所需要的资源还没有就绪，此时必须等待后端的应用资源，这时线程就被阻塞了。

适用场景分析：

BIO 方式适用于连接数目比较小且固定的架构，这种方式对服务器资源要求比较高，并发局限于应用中，JDK1.4 以前的唯一选择，但程序直观简单易理解，如之前在 Apache 中使用。

NIO 方式适用于连接数目多且连接比较短（轻操作）的架构，比如聊天服务器，并发局限于应用中，编程比较复杂，JDK1.4 开始支持，如在 Nginx，Netty 中使用。

AIO 方式使用于连接数目多且连接比较长（重操作）的架构，比如相册服务器，充分调用 OS 参与并发操作，编程比较复杂，JDK7 开始支持，在成长中，Netty 曾经使用过，后来放弃。

java 中常说的堆和栈，分别是什么数据结构；另外，为什么要分为堆和栈来存储数据

栈是一种具有后进先出性质的数据结构，也就是说后存放的先取，先存放的后取。

堆是一种经过排序的树形数据结构，每个结点都有一个值。通常我们所说的堆的数据结构，是指二叉堆。堆的特点是根结点的值最小（或最大），且根结点的两个子树也是一个堆。由于堆的这个特性，常用来实现优先队列，堆的存取是随意的。

为什么要划分堆和栈

1、从软件设计的角度看，栈代表了处理逻辑，而堆代表了数据。这样分开，使得处理逻辑更为清晰。

2、堆与栈的分离，使得堆中的内容可以被多个栈共享。一方面这种共享提供了一种有效的数据交互方式(如：共享内存)，另一方面，堆中的共享常量和缓存可以被所有栈访问，节省了空间。

3、栈因为运行时的需要，比如保存系统运行的上下文，需要进行地址段的划分。由于栈只能向上增长，因此就会限制住栈存储内容的能力。而堆不同，堆中的对象是可以根据需要动态增长的，因此栈和堆的拆分，使得动态增长成为可能，相应栈中只需记录堆中的一个地址即可。

4、体现了 Java 面向对象这一核心特点（也可以继续说一些自己的理解）

为什么要用线程池

那先要明白什么是线程池

线程池是指在初始化一个多线程应用程序过程中创建一个线程集合，然后在需要执行新的任务时重用这些线程而不是新建一个线程。

使用线程池的好处

1、线程池改进了一个应用程序的响应时间。由于线程池中的线程已经准备好且等待被分配任务，应用程序可以直接拿来使用而不用新建一个线程。
2、线程池节省了 CLR 为每个短生存周期任务创建一个完整的线程的开销并可以在任务完成后回收资源。
3、线程池根据当前在系统中运行的进程来优化线程时间片。
4、线程池允许我们开启多个任务而不用为每个线程设置属性。
5、线程池允许我们为正在执行的任务的程序参数传递一个包含状态信息的对象引用。
6、线程池可以用来解决处理一个特定请求最大线程数量限制问题。

msyql 优化经验

1、对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。
2、应尽量避免在 where 子句中使用!=或<>操作符，否则引擎将放弃使用索引而进行全表扫描。
3、尽量使用数字型字段，若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。
4、任何地方都不要使用 select * from t ，用具体的字段列表代替“*”，不要返回用不到的任何字段。
5、避免频繁创建和删除临时表，以减少系统表资源的消耗。诸如此类，等等等等……

悲观锁和乐观锁的区别，怎么实现

悲观锁：一段执行逻辑加上悲观锁,不同线程同时执行时,只能有一个线程执行,其他的线程在入口处等待,直到锁被释放。

乐观锁：一段执行逻辑加上乐观锁,不同线程同时执行时,可以同时进入执行,在最后更新数据的时候要检查这些数据是否被其他线程修改了(版本和执行初是否相同),没有修改则进行更新,否则放弃本次操作。

悲观锁的实现：

```
//0.开始事务
begin;/begin work;/start transaction;（三者选一就可以）
//1.查询出商品信息
select status from t_goods where id=1 for update;
//2.根据商品信息生成订单
insert into t_orders (id,goods_id) values (null,1);
//3.修改商品 status 为 2
update t_goods set status=2;
//4.提交事务
commit;/commit work;
```

乐观锁的实现

```
1.查询出商品信息
select (status,status,version) from t_goods where id=#{id}
2.根据商品信息生成订单
3.修改商品 status 为 2
update t_goods
set status=2,version=version+1
where id=#{id} and version=#{version};
```