类加载过程

- 加载：根据类全限定名加载字节码，解析出数据结构放到方法区，在堆中创建能访问方法区信息的Class实例。
- 连接：验证：确保字节码符合虚拟机-准备：在方法区给类变量分配内存赋值-解析：将类的符号引用替换为直接引用。
- 初始化：执行类构造器：其中包括对static变量的复制和static代码块的语句。优先初始化父类

双亲委派

类加载器优先尝试让父类加载器加载，不行再自己加载，这样尽量保证核心类优先加载，而不被替代。

启动类加载器：C++编写，是JVM的一部分。拓展类加载器：Java编写。系统类加载器：Java编写。用户自定义加载器：继承系统类加载器。（可以通过重写loadClass方法打破双亲委派、加载的类可能被卸载）

CMS

关注停顿时间，基于标记清除，与应用线程并发，老年代。初始标记（STW）：**标记GCRoots直接**可达的；并发标记：根据对象**递归标记**可达；并发预清除：对变化的**重新标记**；重新标记（STW）：递归标记没有处理过的对象；并发清除。缺点：很吃吞吐量，无法收集浮动垃圾（需要提前进行回收），有内存碎片。

数据库三大范式、ACID

每一个字段都是不可分割的、每一个字段都和主键相关，而不是和主键的某一部分相关、确保每一个字段都是和主键直接相关，而不是间接相关。

A：原子性：事务不能拆分 C：一致性 I：隔离性 D：持久性：最后要永久存在数据库中

MySQL架构

- Server层：连接器、分析器、优化器、执行器
- 存储引擎层：MyISAM、InnoDB
    - InnoDB支持事务，有索引，支持表、行锁。MyISAM允许没有索引和主键的表存在，只能用表锁。

B+树

特征：数据存在叶子节点，叶子之间用指针相连，非叶子节点中存的关键字为了来索引到叶子。非叶子节点中的最大数就是所有子树叶子的最大值

MVCC

多版本并发控制，是并发控制的方法，特点是读不加锁，利用保存数据在某时刻的快照的思想。InnoDB的实现：三个关键：隐藏列、undolog、ReadView。保存一个事务开始前活跃事务列表，里面有事务提交就移除。比较查询数据行的事务id去比较，比最小小可见；比最大大不可见；在之间如果存在不可见，不存在可见。

优化

索引：最左匹配（在组合索引靠前的字段是排序的基础上后面的字段才有序）。联表查，被关联的表对应的字段要有索引。尽量使用覆盖索引来避免回表。

limit优化：把联表查询拆成，主表关联子查询，只对子查询做limit，子查询查出主键id。

Volatile

保证可见性，禁止指令重排。volatile修饰的变量的写操作在汇编代码中，会有一个lock前缀，保证会立刻把值写回到主存中，同时其他CPU缓存中的这个变量会被置为无效，在使用时需要重新从主存拿。

Syschronized

java关键字，可以修饰方法和代码块。方法：标记这个方法为同步方法，之前前需要先获得对象的监视器。代码块：字节码在前后使用monitorenter和monitorexit标记，也是需要获取对象监视器才能执行。1.6之后做了优化有了锁升级。偏向锁：对象的MarkWord和栈帧的锁记录中存放线程id。轻量级锁：CAS，MarkWord放锁记录的指针，锁记录放对象的MarkWord

线程池

ThreadPoolExecutor有七大参数。执行有两个方法：execute()不返回值，submit()可返回值，会返回一个Future对象。关闭：shutdown：等所有任务执行完关闭，新来任务直接拒绝。shutdownNow：给所有执行的线程中断信号，队列中的任务放到list中返回。isShutdown：返回是否已经开始关闭工作（是否执行了shutdown或者shotdownNow）

同步器

CountDownLatch：设置倒计时数，线程执行完可以让数-1，没完之前倒计时的await()方法会阻塞、CyclicBarrier：设置周期次数和完成后的操作，线程完成后可以执行await()、Semaphore：设置资源，让线程从里面获取资源，没有就会阻塞

AQS：抽象队列同步器。state表示资源，双线队列存储等待资源的线程，获取资源释放资源让子类实现。有独占方式和共享方式。

ReentrantLock：默认非公平。实现获取资源有公非公两个实现，区别在于公平实现中尝试获取资源时还要判断自己是否是阻塞队列中的第一个。两者通过判断自己是不是当前资源独占线程来实现重用锁。

死锁

条件：互斥、不剥夺、请求和保持、循环等待。通过jps -l定位java出问题的进程，通过jstack 进程id显示详情

避免：1.按指定顺序加锁2.给获取锁加个超时时间3.通过在数据结构中构建依赖锁的链表，如果有环说明有死锁，可以选择自己释放所有锁，或者根据随机优先级让其他线程释放锁

Redis

处理过期数据：定期随机检查、惰性删除。内存淘汰：内存占用过大时删除数据：1.写时报错 2.LRU 3.随机删一个 4.有设置过期的LRU 5.随机删一个设置过期的 6.删除最快要过期的

跳跃表：支持随机插入和删除，查找效率高，有序。结构简单，相对平衡树不需要做过多平衡的动作。原理：链表中没次插入节点，随机分配层数，层数越高包含的节点越少，所在层中相邻节点跳过的低层的节点数就越多，通过高层链表的遍历可以加快整个完整链表的遍历速度。同时插入和删除影响很小。

RDB：指定时间间隔将内存数据快照写入磁盘。fork一个子进程先写入临时文件，写入成功后覆盖到上次持久化好的RDB文件中。写入性能好，恢复速度快，适合灾难恢复。可能会丢失小部分数据

AOF：以日志形式记录写（删）操作，追加文件中，可选同步策略。最大限度避免数据丢失，但是恢复慢，运行效率低。

布隆过滤器

基础

线程通信：文件、signal（除了特殊信号，一般可忽略捕获默认动作，一般默认动作是退出。软件层次上对中断机制的一种模拟，是一种异步通信方式）、管道（ps -ef|grep）、socket

中断

- stop()，Thread提供的，弃用，停止run()中的操作，释放所有锁
- interrupt()，在线程中做一个中断标记，被中断的线程需要自己加上处理中断标记的代码。Thread.isInterrupted()判断是否中断，Thread.interrupted()判断并清除中断

异常

- Throwable
  - Error：错误。无法被程序处理，OOM、ThreadDeath
  - Exception：异常。能被操作，比如抛出和catch。
    - 非运行时：不处理编译不能通过。IOException，ClassNotFoundException。
    - RuntimeException：编译器不会检查。NullPointerException、ArrayIndexOutOfBoundsException

HTTPS=HTTP+加密（对称、非对称）+认证（证书）+完整性保护

- 客户端请求发送可用的加密方式，服务端收到后回应，然后把自己的证书和公钥等信息发给客户端
- 客户端验证证书，生成一个对称的会话秘钥，用公钥加密后发给服务器
- 服务器收到后，用私钥解密得到密钥，然后和客户端相互检验，就握手完成了连接

getpost区别

参数位置，浏览器回退行为，浏览器是否存储参数，参数类型限制

session、cookie 区别

存储位置、用户是否可访问、cookie有个数限制

Socket

用来抽象进程之间的通信。客户端创建Socket实例，分配端口号，绑定远程地址和端口号，三次握手之后，返回Socket实例。服务端创建ServerSocket实例，执行端口号。调用accept()开始监听进入阻塞，连接到来创建套接字，来和客户端交互。

数据结构

HashMap

Cur：

- 1.7：get：两次hash定位到volatile修饰的hashEntry直接拿。put：hash找到segment，调用它的put：tryLock()获取锁，否则自旋
- 1.8：get：hash直接拿。put：hash，懒加载初始化（cas）Node数组，找节点插入用cas失败加锁

ThreadLocal：每个线程有以ThreadLocal为key的map。通过ThreadLocal实例去创建添加和获取键值对。

linux

netstat -ap | grep ssh

top

排序

稳定：冒泡n2、插入n2、归并nlogn

不稳定：选择n2、快速nlogn、堆nlogn（建堆n，维护堆logn）

项目

数据库主要操作：查博文带分类和标签，怎么映射。查对应博文最多的分类group by

Redis：存博文，存分数为view的博文排行榜。**缓存一致性**：博文查Redis没有或者过期了从MySQL中查，在增删改的时候把对应的更新Redis的任务放到队列中取处理。开始就放到Java内存的阻塞队列中。有个启动任务用take去消费，后来用试着放Redis的队列中，lpush放brpop取。view是直接更新Redis，定时任务同步到MySQL。**防缓存穿透**，用存null字符串。**将来某时刻的任务**用zset做时延队列，值为任务Json，用score存任务处理的时间，定时任务每1秒从zset中取0到当前时间范围的元素，zrange key 0 -1。用work类表示任务，用不同的hander子类组成责任链来处理work。