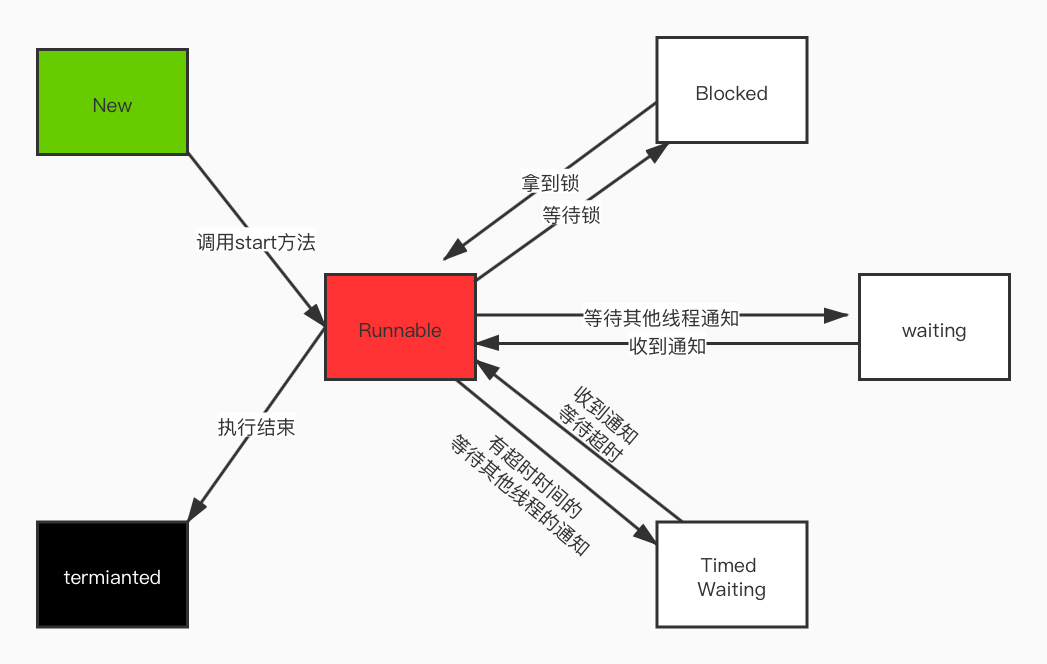
线程

线程状态



线程中断：stop过时方法，非线程安全。interrupt

线程通讯：suspend、resume不释放锁，队执行顺序也有要求，过时方法

wait、notify释放锁，对执行顺序没有要求

LockSupport. park/unpark，对执行顺序没有要求，但是不释放锁

线程池：线程池的执行顺序

1.判断是否达到核心线程数，如果没达到，则生成线程

2.如果达到核心线程数，则往队列里放任务（若为无界队列则一直放）

3.如果队列已满，则判断是否达到最大最大线程数，如果未达到，则继续生成直到最大线程数，否则拒绝执行（可定义拒绝策略）

线程封闭：ThreadLocal、局部变量。

指令重排：as-if-serial语义

happens-before原则

可见性及原子操作

AQS：

锁：

并发容器

ForJoin/FutureTask

Java系统性能调优

类加载机制：

1.加载：根据类的全限定名获取二进制字节流

2.验证：文件格式验证、元数据验证、字节码验证

3.准备：为类变量赋初始值

4.解析：

5.初始化：

垃圾回收机制：

引用计数法：对象被引用，计数+1，计数为0表示可回收，无法解决循环引用的问题。

可达性算法：静态变量引用的对象、常量引用的对象、栈中引用的对象，作为GC Roots，搜索与GC Roots相关联的对象，搜索的路径为引用链，当一个对象与引用链没有关联时，将被进行标记，至于会不会被回收，要看是否有必要执行finalize方法，当一个对象的finalize方法没有被重写，或者该方法已经被虚拟机执行过，则为没必要执行。若有必要执行，则将该对象放入F-QUEUE中，由一个低级别的finalizer线程执行，稍后gc会对F-QUEUE中的对象进行标记，如果在finalize方法中，对象与引用链上的其他对象关联，则会拯救自己，否则将进行回收。

复制算法：将内存分为两块，gc时，将存活的对象复制到另一边，并将这一边清除。实现简单，效率高，但是内存利用率低。

标记清除：将标记为回收的对象直接清除，会产生内存碎片，cms垃圾收集器采用这种算法，不过可以设置在回收后进行内存整理，若频繁整理耗时太大，可以设置多少次清除回收以后紧跟着来一次整理。

标记整理算法：将存活的对象向一边移动，在边界外的对象清除。

新生代采用复制算法，新生代对象基本都是朝生夕死，用复制算法效率较高

新生代分为survivor0、survivor1、eden区，默认比例1:1:8，可以通过

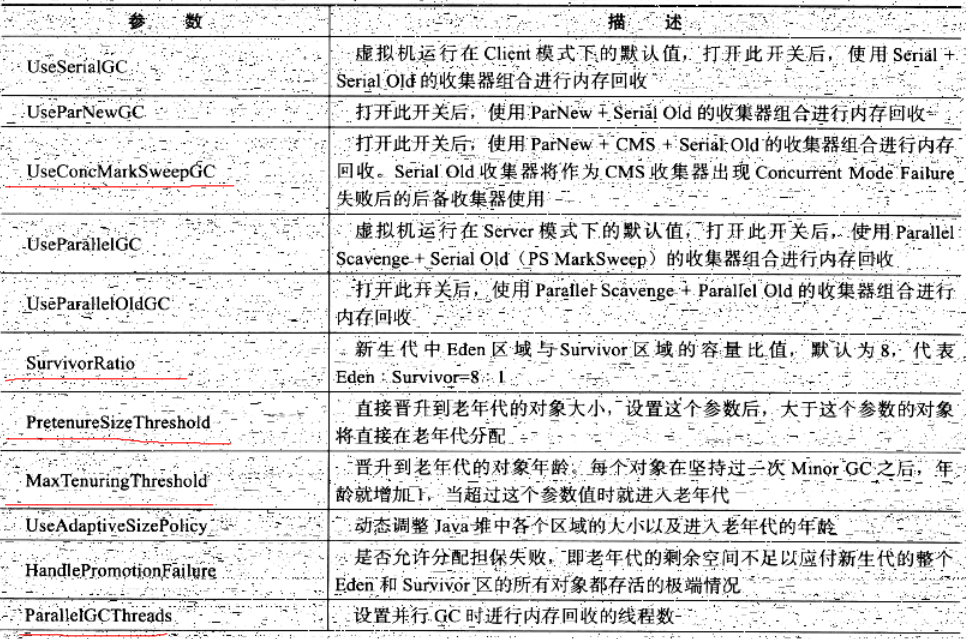
-XX:surviorRatio=?设置s区和eden区的比例。回收时，将eden和s0或s1的存活的对象复制到另外一个s区中。

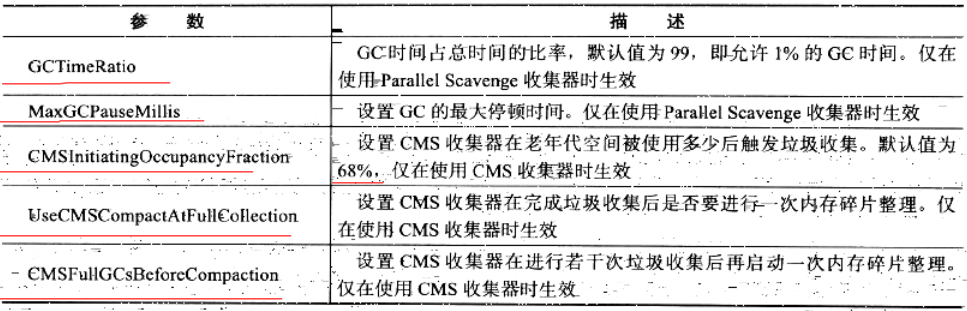
serial、serialOld收集器中，对象优先在eden区分配，若eden区没有足够空间时，将进行一次monior gc，若此时仍然没有足够空间，则判断老年代是否有足够的连续空间可以容纳新生代所有存活的对象，若可以，那么这次gc是安全的，大对象将会提前进入老年代，若不可以容纳，则要查看虚拟机是否允许分配担保失败（HandlePromotionFailure），若不允许，将进行Full GC，若允许，则判断老年代连续存储空间是否大于历次晋升到老年代的对象的平均大小，若大于，将进行monior gc，尽管这次gc是有风险的，若小于，则进行full gc。

大对象直接进入老年代，可以设置XX:PretenureSizeThreshold=1024\*1024 此参数的单位只能是字节，如题为1M。

长期存活的对象进入老年代，-XX:MaxTenuringThreshold=15 设置经过多少次moniorGC还未被回收则进入老年代 默认为15次。但是当survivor中相同年龄的对象超过survivor空间的一半时，则大于等于该年龄的对象提前进入老年代，就算其年龄未超过设置的值。

垃圾收集器：





jdk内置命令工具

jps

jinfo -flags

jstat -options 查看类加载 gc信息等

jmap -dump 内存映像 mat分析

jstack 线程快照

jvm 监控工具

jconsole：

jvisiualvm：

mat: 分析内存映像，导入hrof文件

gcviewer：

生产问题定位

内存问题：-XX:+HeapDumpOnOutOfMemoryError，当内存溢出后会生成映像文件，利用mat分析，可以很容易的定位到内存溢出的原因。网易云课堂遇到的问题是，查询订单时，未检查参数的大小，导致查询了11w条订单，list中保存11w订单，导致内存溢出。

cpu问题：生产上cpu使用率很高，top命令查看cpu信息，按H可查看线程使用cpu的信息，jstack输出线程快照，根据top中线程id到线程快照中查看该线程的信息

jvm调优思路