面试大全之JVM篇

JVM

1. 内存模型以及分区，需要详细到每个区放什么。

> JVM 分为堆区和栈区，还有方法区，初始化的对象放在堆里面，引用放在栈里面，class类信息常量池（static变量）等放在方法区

1. 堆里面的分区：Eden，survival （from+ to），老年代，各自的特点。

> 堆里面分为新生代和老生代（java8取消了永久代，采用了Metaspace），新生代包含Eden+Survivor区，survivor区里面分为from和to区，内存回收时，如果用的是复制算法，从from复制到to，当经过一次或者多次GC之后，存活下来的对象会被移动到老年区，当JVM内存不够用的时候，会触发Full GC，清理JVM老年区
> 当新生区满了之后会触发YGC,先把存活的对象放到其中一个Survice区，然后进行垃圾清理。因为如果仅仅清理需要删除的对象，这样会导致内存碎片，因此一般会把Eden 进行完全的清理，然后整理内存。那么下次GC 的时候，就会使用下一个Survive，这样循环使用。如果有特别大的对象，新生代放不下，就会使用老年代的担保，直接放到老年代里面。因为JVM 认为，一般大对象的存活时间一般比较久远。

1. 对象创建方法，对象的内存分配，对象的访问定位。

> new 一个对象

1. GC的两种判定方法：

> 引用计数法：指的是如果某个地方引用了这个对象就+1，如果失效了就-1，当为0就会回收但是JVM没有用这种方式，因为无法判定相互循环引用（A引用B,B引用A）的情况
> 引用链法： 通过一种GC ROOT的对象（方法区中静态变量引用的对象等-static变量）来判断，如果有一条链能够到达GC ROOT就说明，不能到达GC ROOT就说明可以回收

1. SafePoint是什么

> 比如GC的时候必须要等到Java线程都进入到safepoint的时候VMThread才能开始执行GC，
>
> 1. 循环的末尾 (防止大循环的时候一直不进入safepoint，而其他线程在等待它进入safepoint)
> 2. 方法返回前
> 3. 调用方法的call之后
> 4. 抛出异常的位置

1. GC的三种收集方法：标记清除、标记整理、复制算法的原理与特点，分别用在什么地方，如果让你优化收集方法，有什么思路？

> 先标记，标记完毕之后再清除，效率不高，会产生碎片
> 复制算法：分为8：1的Eden区和survivor区，就是上面谈到的YGC
> 标记整理：标记完毕之后，让所有存活的对象向一端移动

1. GC收集器有哪些？CMS收集器与G1收集器的特点。

> 并行收集器：串行收集器使用一个单独的线程进行收集，GC时服务有停顿时间
> 串行收集器：次要回收中使用多线程来执行
> CMS收集器是基于**“标记—清除”**算法实现的，经过多次标记才会被清除
> G1从**整体来看是基于“标记—整理”**算法实现的收集器，从**局部（两个Region之间）上来看是基于“复制”**算法实现的

1. Minor GC与Full GC分别在什么时候发生？

> 新生代内存不够用时候发生MGC也叫YGC，JVM内存不够的时候发生FGC

1. 几种常用的内存调试工具：jmap、jstack、jconsole、jhat

> jstack可以看当前栈的情况，jmap查看内存，jhat 进行dump堆的信息

1. 类加载的五个过程：

> 加载、验证、准备、解析、初始化。然后是使用和卸载了
> 通过全限定名来加载生成class对象，然后进行验证这个class文件，包括元数据验证，字节码校验等。准备是对这个对象分配内存。
> 解析是将符号引用转化为直接引用（指针引用），初始化就是开始执行构造器的代码

1. 双亲委派模型：Bootstrap ClassLoader、Extension ClassLoader、ApplicationClassLoader。

> Bootstrap ClassLoader:启动类加载器，负责将$ Java_Home/lib下面的类库加载到内存中（比如rt.jar

公告

昵称：itar
园龄：8个月
粉丝：0
关注：0
+加关注

< 2018年5月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

> Extension ClassLoader：标准扩展（Extension）类加载器，它负责将$Java_Home /lib/ext或者由系统变量 java.ext.dir指定位置中的类库加载到内存中。
> ApplicationClassLoader：它负责将系统类路径（CLASSPATH）中指定的类库加载到内存中。开发者可以直接使用系统类加载器
> 双亲委派模型是某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。-----例如类java.lang.Object，它存在在rt.jar中，无论哪一个类加载器要加载这个类，最终都是委派给处于模型最顶端的Bootstrap ClassLoader进行加载，因此Object类在程序的各种类加载器环境中都是同一个类。相反，如果没有双亲委派模型而是由各个类加载器自行加载的话，如果用户编写了一个java.lang.Object的同名类并放在ClassPath中，那系统中将会出现多个不同的Object类，程序将混乱

1. 分派：静态分派（重载）与动态分派（重写）。
2. 你知道哪些JVM性能调优

> 设定堆最小内存大小-Xms
>
> 1. -Xmx：堆内存最大限制。
> 2. 设定新生代大小。
>   新生代不宜太小，否则会有大量对象涌入老年代
>
> -XX:NewSize：新生代大小
> -XX:NewRatio 新生代和老生代占比
> -XX:SurvivorRatio：伊甸园空间和幸存者空间的占比
>
> 1. 设定垃圾回收器
>
> 年轻代用 -XX:+UseParNewGC （串行） 年老代用-XX:+UseConcMarkSweepGC （CMS）
>
> 1. 设定锁的使用
>
> > 多线程下关闭偏向锁，比较浪费资源
>
> 1. g1 和 cms 区别,吞吐量优先和响应优先的垃圾收集器选择
>
> > 1CMS是一种以最短停顿时间为目标的收集器
> > 响应优先选择CMS,吞吐量高选择G1
>
> 1. 当出现了内存溢出，你怎么排错
>
> > 用jmap看内存情况，然后用 jstack主要用来查看某个Java进程内的线程堆栈信息
>
> 1. JVM内存模型的相关知识了解多少
>
> > JVM内存结构扯一下

itar
关注 - 0
粉丝 - 0
+加关注

posted @ 2017-08-24 17:45 itar 阅读(2554) 评论(0) 编辑 收藏

· 如何识别人的技术能力和水平？
· 写给自学者的入门指南
· 和程序员谈恋爱
· 学会学习
· 优秀技术人的管理陷阱
» 更多知识库文章...