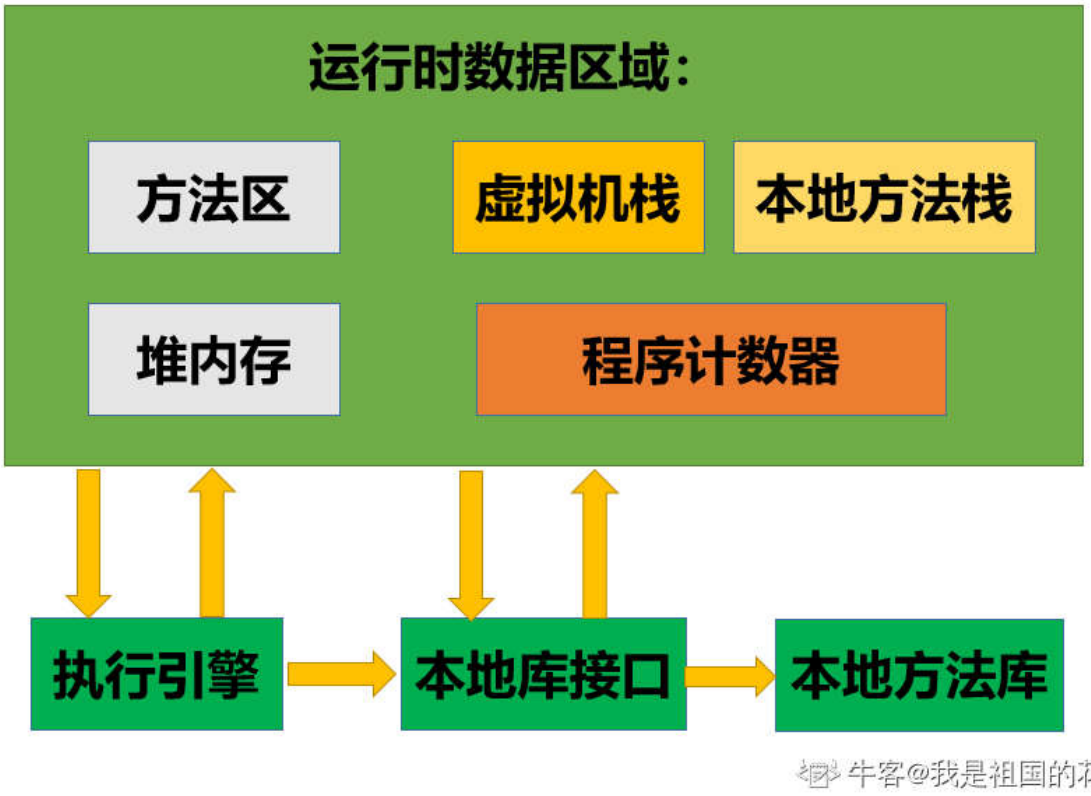
大家好，很高兴我们可以继续学习交流Java高频面试题。从本小节开始，我们将用三个小节来交流学习JVM内存机制相关的技术原理。 8

JVM内存机制是Java面试中的重中之重，面试中的绝对热点。作为一名优秀的Java开发工程师，日常工作中必不可少的要与JVM打交道，我们必须考虑到**如何对JVM进行内存分配，应该对当前服务采用哪种垃圾回收器的问题。当我们遇到OOM内存溢出的故障时，也必须去分析研究其原因，所以还需要使用到各个内存调优分析手段。**

作为Java面试中的重点，我希望大家可以扎实理解掌握JVM相关的技术原理。本小节所涉及的知识点包括JVM内存的分配与回收策略以及垃圾回收的基本概念等。好了，话不多说，让我们一起来学习吧~

（1）JVM中的内存是怎么划分的？（重点掌握）

答：JVM中的内存主要划分为5个区域，即**方法区，堆内存，程序计数器，虚拟机栈以及本地方法栈。**下边是Java虚拟机运行时数据区示意图：



方法区：方法区是一个**线程之间共享**的区域。**常量，静态变量以及JIT编译后的代码都在方法区。主要用于存储已被虚拟机加载的类信息**，也可以称为“永久代”，垃圾回收效果一般，通过-XX：MaxPermSize控制上限。

堆内存：堆内存是**垃圾回收**的主要场所，也是线程之间共享的区域，主要用来**存储创建的对象实例**，通过-Xmx 和-Xms 可以控制大小。

虚拟机栈（栈内存）：栈内存中主要保存**局部变量、基本数据类型变量**以及**堆内存中某个对象的引用变量**。每个方法在执行的同时都会创建一个栈帧（Stack Frame）用于存储局部变量表，操作数栈，动态链接，方法出口等信息。栈中的栈帧随着方法的进入和退出有条不紊的执行着出栈和入栈的操作。

程序计数器： 程序计数器是当前线程执行的**字节码的位置指示器**。字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令，是内存区域中唯一一个在虚拟机规范中**没有规定任何OutOfMemoryError情况**的区域。

本地方法栈： 主要是为JVM提供使用native 方法的服务。

8

解析：

JVM的内存划分主要由以上五个区域组成，我们需要重点掌握堆内存，栈内存以及方法区域的定义和作用，做到准确理解与阐述。

（2）可以说一下对象创建过程中的内存分配吗？

答：一般情况下我们通过new指令来创建对象，当虚拟机遇到一条new指令的时候，会去检查这个指令的参数是否能在常量池中定位到某个类的符号引用，并且检查这个符号引用代表的类是否已经被加载，解析和初始化。如果没有，那么会执行类加载过程。

通过执行类的加载，验证，准备，解析，初始化步骤，完成了类的加载，这个时候会为该对象进行内存分配，也就是把一块确定大小的内存从Java堆中划分出来，在**分配的内存**上完成对象的创建工作。

对象的内存分配有两种方式，即指针碰撞和空闲列表方式。

指针碰撞方式：

假设Java堆中的内存是绝对规整的，用过的内存在一边，未使用的内存在另一边，中间有一个指示指针，那么所有的内存分配就是把那个指针向空闲空间那边挪动一段与对象大小相等的距离。

空闲列表方式：

如果Java堆内存中不是规整的，已使用和未使用的内存相互交错，那么虚拟机就必须维护一个列表用来记录哪块内存是可用的，在分配的时候找到一块足够大的空间分配对象实例，并且需要更新列表上的记录。

需要注意的是，Java **堆内存是否规整**是由所使用的垃圾收集器**是否拥有压缩整理功能来决定**的，关于垃圾收集器我们在下一小节重点介绍。

看到这里，聪明的你肯定想到了内存分配是否也应该考虑线程安全的问题呢？

那么内存的分配如何保证线程安全呢？

- 对分配内存空间的动作进行同步处理，通过“**CAS + 失败重试**”的方式保证更新指针操作的原子性
- 把分配内存的动作按照线程划分在不同的空间之中，即给每一个线程都预先分配一小段的内存，称为**本地线程分配缓存（TLAB）**，只有TLAB用完并分配新的TLAB时，才需要进行同步锁定。 虚拟机是否使用TLAB，可以通过**-XX: +/-UserTLAB**参数来设定

（3）对象被访问的时候是怎么被找到的？

答：通过对栈内存和堆内存的介绍，我们知道了当创建一个对象的时候，在栈内存中会有一个引用变量，指向堆内存中的某个具体的对象实例。

Java虚拟机规范中并**没有规定这个引用变量应该以何种方式去定位和访问堆内存中的具体对象**。目前常见的对象访问方式有两种，即句柄访问方式和直接指针访问方式，分别介绍如下。

句柄访问方式：

在JVM的堆内存中划分出一块内存来作为句柄池，引用变量中存储的就是对象的句柄地址，句柄中包含了对象实例数据与类型数据各自的具体地址信息。在内存垃圾收集之后，对象会移动，但是引用reference中存储的是稳定的句柄地址，但是句柄地址方式不直接，访问速度较慢。

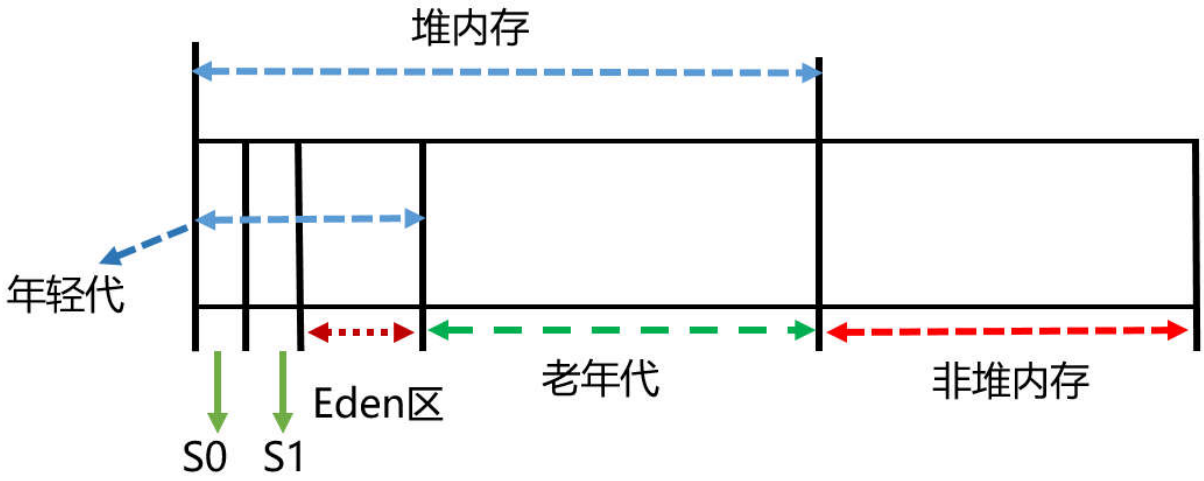
8

直接指针访问方式：

引用变量中存储的就是对象的直接地址，通过指针直接访问对象。直接指针的访问方式节省了一次指针定位的时间开销，速度较快。Sun HotSpot使用了直接指针方式进行对象的访问。

（4）内存分配与垃圾回收：（重点掌握）

答：JVM的内存可以分为**堆内存和非堆内存**。堆内存分为**年轻代和老年代**。年轻代又可以进一步划分为**一个Eden（伊甸）区和两个Survivor（幸存）区**组成。如下图所示：



JVM内存简单分配示意图

JVM堆内存的分配：

JVM初始分配的堆内存由**-Xms**指定，默认是物理内存的1/64。JVM最大分配的堆内存由**-Xmx**指定，默认是物理内存的1/4。

默认空余堆内存小于40%时，JVM就会增大堆直到-Xmx的最大限制。空余堆内存大于70%时，JVM会减少堆直到 -Xms的最小限制。因此我们一般设置-Xms和-Xmx相等以避免在每次GC 后调整堆的大小。

通过参数**-Xmn2G** 可以设置年轻代大小为2G。通过**-XX:SurvivorRatio**可以设置年轻代中Eden区与Survivor区的比值，设置为8，则表示年轻代中Eden区与一块Survivor的比例为8：1。注意年轻代中有两块Survivor区域。

JVM非堆内存的分配：

JVM使用**-XX:PermSize** 设置非堆内存初始值，默认是物理内存的1/64。由**-XX:MaxPermSize**设置最大非堆内存的大小，默认是物理内存的1/4。

接下来，我们具体阐述堆内存上的对象分配与回收过程吧。

堆内存上对象的分配与回收：

我们创建的对象会优先在Eden分配，如果是大对象（很长的字符串数组）则可以直接进入老年代。虚拟机提供一个**-XX:PretenureSizeThreshold参数**，令大于这个参数值的对象直接在老年代中分配，避免在Eden区和两个Survivor区发生大量的内存拷贝。

8

另外，**长期存活的对象将进入老年代**，每一次**MinorGC（年轻代GC）**，对象年龄就大一岁，默认15岁晋升到老年代，通过
-XX:MaxTenuringThreshold设置晋升年龄。

堆内存上的对象回收也叫做垃圾回收，那么垃圾回收什么时候开始呢?

垃圾回收主要是完成**清理对象，整理内存**的工作。上面说到GC经常发生的区域是堆区，堆区还可以细分为新生代、老年代。新生代还分为一个Eden区和两个Survivor区。垃圾回收分为年轻代区域发生的Minor GC和老年代区域发生的Full GC，分别介绍如下。

Minor GC（年轻代GC）：
对象优先在Eden中分配，当Eden中没有足够空间时，虚拟机将发生一次Minor GC，因为Java大多数对象都是朝生夕灭，所以Minor GC非常频繁，而且速度也很快。

Full GC（老年代GC）：
Full GC是指发生在老年代的GC，当老年代没有足够的空间时即发生Full GC，发生Full GC一般都会有一次Minor GC。

接下来，我们来看关于内存分配与回收的两个重要概念吧。

动态对象年龄判定：

如果Survivor空间中相同年龄所有对象的大小总和大于Survivor空间的一半，那么**年龄大于等于该对象年龄的对象即可晋升**到老年代，不必要等到-XX:MaxTenuringThreshold。

空间分配担保：

发生Minor GC时，虚拟机会检测之前每次晋升到老年代的平均大小是否大于老年代的剩余空间大小。如果大于，则进行一次**Full GC（老年代GC）**，如果小于，则查看**HandlePromotionFailure设置是否允许担保失败**，如果允许，那只会进行一次**Minor GC**，如果不允许，则改为进行一次**Full GC**。

解析：

关于JVM堆内存上对象的分配与回收是面试中考察的重点，希望大家可以对相关知识点熟练掌握，并且清晰阐述各个关键概念。

（5）JVM如何判定一个对象是否应该被回收？（重点掌握）

答： 判断一个对象是否应该被回收，主要是看其是否还有引用。判断对象是否存在引用关系的方法包括**引用计数法以及root根搜索方法**。

引用计数法：

是一种比较古老的回收算法。原理是此对象有一个引用，即增加一个计数，删除一个引用则减少一个计数。垃圾回收时，只需要收集计数为0的对象。此算法最致命的是**无法处理循环引用**的问题。

root根搜索方法：

8

root搜索方法的基本思路就是通过一系列可以做为root的对象作为起始点，从这些节点开始向下搜索。当一个对象到root节点没有任何引用链接时，则证明此对象是可以被回收的。以下对象会被认为是root对象：

- **栈内存中引用的对象**
- **方法区中静态引用和常量引用指向的对象**
- 被启动类（bootstrap加载器）加载的类和创建的对象
- Native方法中JNI引用的对象。

解析：

关于对象是否可以被回收的问题也是JVM考察中常见的题目。主要掌握引用计数法的基本原理与优缺点。然后对于root根搜索方法也应该掌握理解。说了这么多，那么我们来看看**什么是对象的引用吧。**

如果reference类型的数据中存储的数值代表的是另外一块内存的起始地址，就称为**这块内存代表一个引用**。JDK1.2以后将引用分为**强引用，软引用，弱引用和虚引用四种。**

- **强引用：**普通存在， P p = new P()，只要强引用存在，垃圾收集器永远不会回收掉被引用的对象。
- **软引用：**通过SoftReference类来实现软引用，在内存不足的时候会将这些软引用回收掉。
- **弱引用：**通过WeakReference类来实现弱引用，每次垃圾回收的时候肯定会回收掉弱引用。
- **虚引用：**也称为幽灵引用或者幻影引用，通过PhantomReference类实现。设置虚引用只是为了对象被回收时候收到一个系统通知。

总结：

本小节中，我们主要阐述了JVM内存机制的相关基础概念，包括内存的分配与回收基本策略，并且给出了垃圾回收的相关技术原理。本小节是基础，同时也是面试必考知识点，相关高频考点已经在文中标出。下一小节，我们将在本节基础知识之上来进一步阐述垃圾回收算法以及垃圾收集器等相关知识点。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论 评论

浮报酒酒陆 1#

希望栏主多添加点图，有的知识点图示比语言更直观简练

发表于 2019-12-06 15:27:54 赞(0) 回复(1)

我是祖国的花朵 N 作者 ： 好的，后续我会注意 。

2019-12-16 21:00:46 赞(0) 回复(0)

请输入你的观点

回复

柳杰201905011049420 2#

感觉消化的不太好，楼主这块还有更新吗

发表于 2020-01-02 16:26:59 赞(1) 回复(1)

8

我是祖国的花朵 作者 ： JVM的深入学习，建议入手一本周志明老师的《深入理解Java虚拟机-JVM高级特性与最佳实践》第三版，这本书不管是备战面试还是实际工作中都值得我们研读多遍。

2020-01-02 20:33:50 赞(1) 回复(0)

请输入你的观点

回复

刘畅201904211606816 3#

打卡

发表于 2020-01-02 23:23:37 赞(0) 回复(0)

牧水s 4#

打卡....在内存分配图里没有画survivor区吧...在堆内存大小设置的时候，应该是虚拟机设置不是服务器设置吧...

发表于 2020-01-18 17:28:44 赞(0) 回复(3)

我是祖国的花朵 作者 ： 你好，S0和S1就是两个Survivor区域哈，至于说后边，应该是我们一般设置...会好点，我改改

2020-01-18 17:31:45 赞(0) 回复(0)

牧水s ： Eden区应该在左边，s0，s1指的是from和to区。是这样吧。☹

2020-01-21 14:42:13 赞(0) 回复(0)

我是祖国的花朵 作者 回复 牧水s ： 是滴，其实不用纠结是左边右边哈

2020-02-20 17:58:14 赞(0) 回复(0)

请输入你的观点

回复

禾田誉子 5#

有一个问题想请教一下，上文说Full GC是发生在老年代的GC，那Major GC呢？我的理解是Full GC是全体范围的GC,Major GC是只发生在老年代的GC.求解答

发表于 2020-02-20 10:58:05 赞(0) 回复(2)

我是祖国的花朵 作者 ： 你好，是这样的。一般情况下，我们可以认为垃圾回收主要是针对堆内存所进行的内存回收。堆内存又分为年轻代和 老年代。对象优先分配在年轻代的Eden区，当Eden区

域内存不足时，会触发一次minor gc（年轻代区域的内存清理工作）；minor gc之后，会有一些对象达到了年龄阈值要求，需要进入老年代，根据是否开启担保机制以及老年代所剩内存等 因素来决定是否进行一次Full gc，用来清理老年代区域的内存。一般情况下，我们可以认为Full gc和major gc表达的 是同一个意思。也就是说，在每一次Full gc(major gc)之前都已经进行了一次minor gc了（这个和不同垃圾收集器实现 有关）。 还有就是你说的文章单薄的意见哈，是这样的，这个专栏主要是给大家提供面试高频题目并且尽量给大家梳理知识点， 大家看了之后还需要对不熟悉的地方多加学习才对，不可能凭借一个专栏就把这些知识点都吃透的，加油~

2020-02-20 17:57:18 赞(0) 回复(0)

禾田誉子 回复 我是祖国的花朵 作者： 嗯嗯，感谢解答

2020-02-20 18:26:11 赞(0) 回复(0)

请输入你的观点

回复

ZZZ13 6#

堆内存上的分配与回收，第一个参数应该是-XX:PretenureSizeThreshold

发表于 2020-02-26 16:47:24 赞(0) 回复(1)

我是祖国的花朵 作者： 赞，已经改正。多谢提醒。

2020-02-26 20:24:43 赞(0) 回复(0)

请输入你的观点

回复

Joe_Hu 7#

这一页真是太难理解了，特别是对象创建过程中的内存分配这个问题。建议大家配合b站和博客上的一些视频进行学习，搜索创建对象内存分析就有很多很清楚的表达。

发表于 2020-04-29 16:57:21 赞(0) 回复(0)

现在的菜鸡，未来的顶尖技术总监 8#

楼主你好，
发生Minor GC时，虚拟机会检测之前每次晋升到老年代的平均大小是否大于老年代的剩余空间大小。这句话好难理解啊，之前晋升的若小于老年代的剩余空间大小那还怎么晋升呢?

发表于 2020-05-23 09:21:37 赞(0) 回复(1)

刘子君®： 比较的是之前晋升的对象大小的平均值，就像一个大小为100的空间已经晋升了10个对象，占了95大小的空间，那平均值就是9.5，这时候剩余空间就只剩5，5小于9.5，所以要进行一次full GC

2020-06-14 21:07:46 赞(0) 回复(0)

请输入你的观点

回复

8