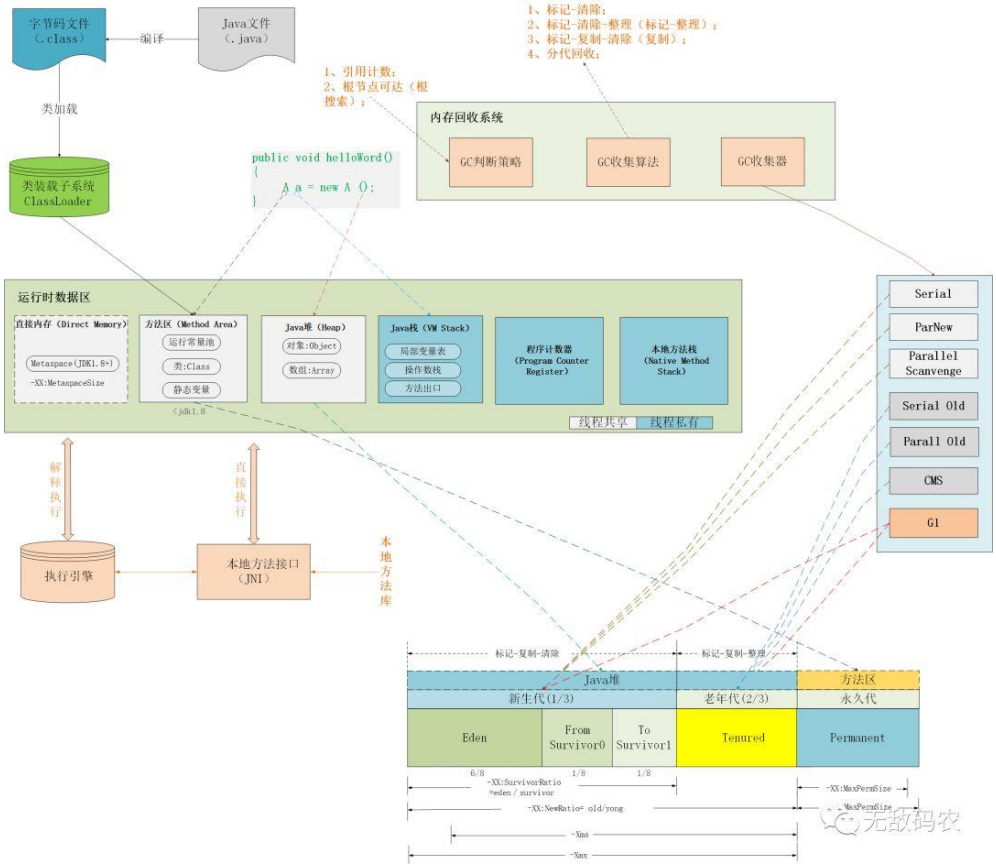
一张图看懂JVM（升级版）

知音小助手 Java基础 2018年9月16日

在上一版文章发出后，作者收到了很多朋友的反馈，有反馈图片不清晰的，也有反馈说内存部分画的不是太细、缺少必要的文字描述，在这里小码农要向这些朋友表示抱歉，同时也向这些朋友表示感谢，正是因为有了你们的鞭策，小码农才有了持续学习的动力。

所以，这两天小码阅读了更为详细的资料，并对之前的内容进行了更为细化的梳理，希望这次能让大家对JVM相关的知识点有更加深刻的理解，也欢迎大家多多批评指正。

JVM结构示意图



JVM总体概述

JVM总体上是由类装载子系统（ClassLoader）、运行时数据区、执行引擎、内存回收这四个部分组成。其中我们最为关注的运行时数据区，也就是JVM的内存部分则是由方法区（Method Area）、JAVA堆（Heap）、虚拟机栈（Stack）、程序计数器、本地方法栈这几部分组成；除此以外，在概念中还有一个直接内存的概念，事

知音小助手 编
关注 私信
Java之音网站的小助手，帮助大家整理有

最新文章
JAVA面试解析(有赞二面)
JAVA面试解析（有赞一面）
亿级PV物联网的基础架构
从0设计一个基于Redis的锁服务
这可能是最为详细的Docker入门吐血
关注 0 粉丝 15 点赞 36
极客快讯
Node 11.7.0 发布
2019年1月20日 10
拼多多出现大Bug：100无门槛券随
2019年1月20日 2.96K
红帽宣布从 Red Hat Enterprise Lin
删除 MongoDB
2019年1月19日 72
知名文件传输协议 SCP 被曝存在 35
洞
2019年1月18日 34
Angular.js 1.7.6 发布
2019年1月18日 68

JVM内存概述

各内存部分的功能及具体组成部分，总结如下：

需要说明的是，堆内存是GC重点回收区域，其中分代回收机制将堆内存划分为年轻代、老年代两个区域，默认情况下年轻代占整个堆内存空间的1/3,而老年代则占2/3,可以通过“-XX:NewRatio”设置年轻代与老年代的比值，默认为2，表示比值年轻代与老年代的比值为“1：2”，在JVM调优时可根据应用实际情况进行调整。

而年轻代又分为Eden、Survivor0、Survivor1，这三个区域占整个新生代空间的比值为8:1:1，即Eden区占8/10，其他两个区域分别占1/10,可通过“-XX:SurvivorRatio”参数进行设置，默认值为8。

正确理解并发问题

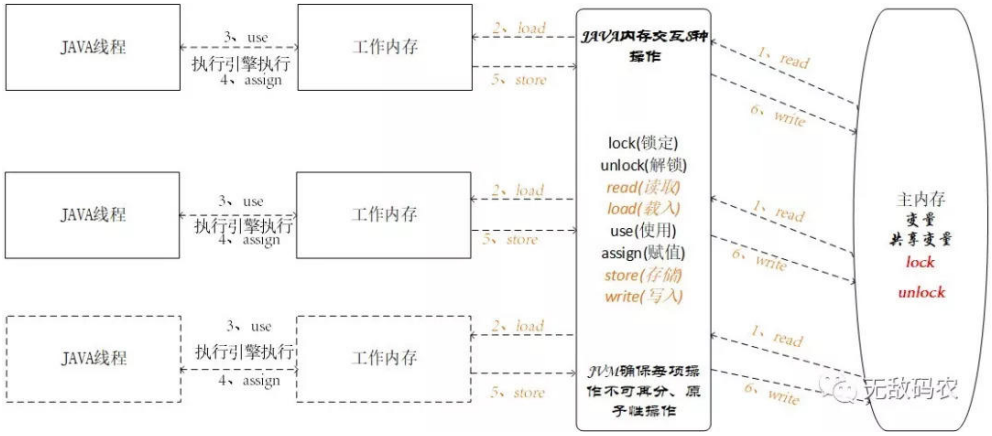
在了解了JVM结构，特别是内存结构后，我们再说说并发问题产生的原因。在上面的内容中我们分析了Java堆、Java栈，知道Java堆存储的是对象，而Java栈内存是方法执行时所需要的局部变量，其中就包括堆中对象的引用，如果多个线程同时修改堆中同一引用对象的数据，就可能会产生并发问题，导致多个线程中某些线程得到的数据值与实际值不符，造成脏数据。

那么这种问题为什么会发生呢?

实际上，线程操作堆中共享对象数据时并不是直接操作对象所在的那块内存，这里称之为主内存；而是将对象拷贝到线程私有的工作内存中进行更新，完成后再将最新的数据值同步回主内存，而多个线程在同一时刻将一个对象的值改得七七八八，然后再同时同步给对象所在的内存区域，那么以谁更新的为准就成了问题了。

所以，为了防止这种情况出现，Java提供了同步机制，确保各个线程按照一定的机制同一时刻只能运行一个线程更新主内存的值。

具体逻辑示意图如下：



注意，这里所讲的主内存、工作内存与Java内存区域中的Java堆、栈内存、方法区等并不是同一个层次的内存划分。如果勉强类比，从变量、主内存、工作内存的定义来看，主内存主要对应于Java堆中对象实例数据部分，而工作内存则对应于虚拟机栈中使用的部分内存区域；从更低层次类比，主内存就直接对应于物理硬件的内存，而为了获取更好的运行速度，虚拟机（甚至是硬件系统本身的优化措施）可能会让内存优先存储于寄存器和高速缓存中，因为程序运行时主要访问读写的是工作内存。

而主内存与工作内存之间具体的交互协议，即一个变量如何从主内存拷贝到工作内存、如何从工作内存同步回主内存之间的实现细节，Java内存模型中定义了8种操作来完成。

而且还规定在执行上述8种基本操作时必须满足如下规则：

不允许read和load、store和write操作之一单独出现，即不允许一个变量从主内存读取了但工作内存不接受，或者从工作内存发起了回写了但主内存不接受的情况出现。

不允许一个线程丢弃它的最近的assign操作，即变量在工作内存中改变了之后必须把该变化同步回主内存。

不允许一个线程无原因地（没有发生任何assign操作）把数据从线程的工作内存同步回主内存中。

一个新的变量只能在主内存中“诞生”，不允许在工作内存中直接使用一个未被初始化（load或assign）的变量，换句话说，就是对一个变量实施use、store操作之前，必须先执行过了assign和load操作。

一个变量在同一时刻只允许一条线程对其进行lock操作，但lock操作可以被同一条线程重复执行多次，多次执行lock后，只有执行相同次数的unlock操作，变量才会被解锁。

如果对一个变量执行lock操作，那将会清空工作内存中此变量的值，在执行引擎使用这个变量前，需要重新执行load或assign操作初始化变量的值。

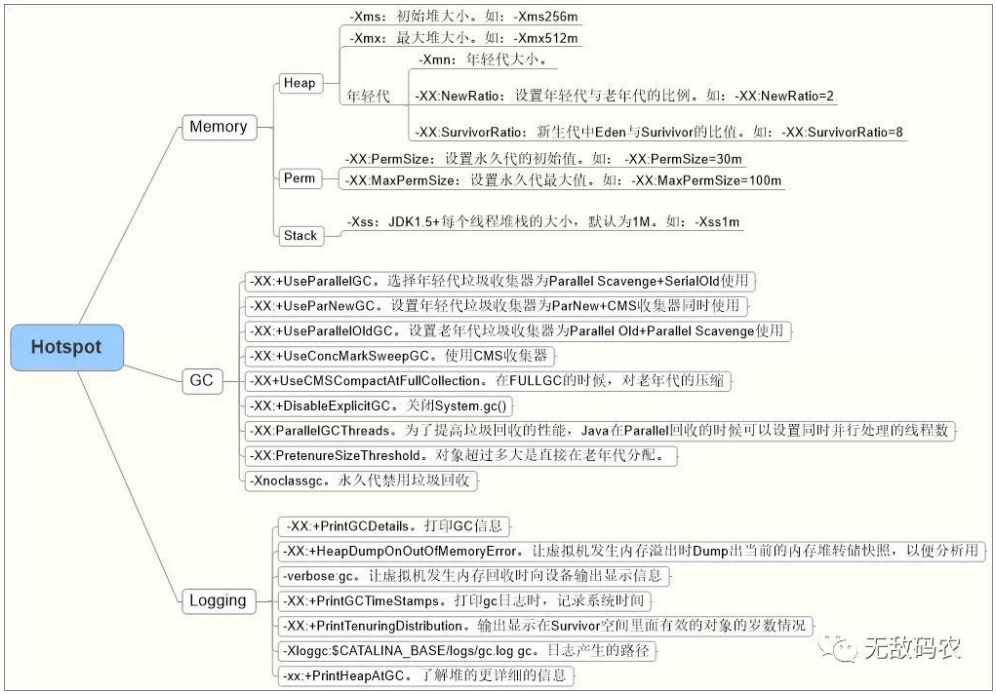
如果一个变量事先没有被lock操作锁定,那就不允许对它执行unlock操作,也不允许去unlock一个被其他线程锁定住的变量。

对一个变量执行unlock操作之前，必须先把此变量同步回主内存中（执行store、write操作）。

以上8种内存访问操作以及上述规则限定，再加上volatile的一些特殊规定以及final不可变特性，就已经完成确定了JAVA程序中那些内存访问操作在并发下是安全的!

JVM参数总结

为了方便大家对于JVM有关参数有一个参照，如下：



后记

读到这里，小编希望能够对大家温习基础知识起到一定的帮助，特别是从事Java开发工作时间并不长的朋友希望本文能对你们有所促进，因为根据作者的经验，有时候很多从事Java开发工作好几年的同学，都会对这些知识点产生模糊的认识，一方面是目前各类Java开源工具比较完备，另一方面是很多人从事的是业务研发工作，时间久了难免会对基础知识有所遗忘。在上面的部分中还有一块垃圾回收的知识点没有总结到，基于篇幅的原因后面再单独给大家总结！谢谢你们的关注~

————————END————————

如果觉得小哥在认真写文章，可以关注下公众号，支持下哦

jvm(17)

分享：

精简阅读 生成封面

赞 3

上一篇：使用PLSQL Developer客户端连接Oracle 11g...

下一篇：Docker 生态概览

Java 8简明教程

Java集合系列[3]HashMap源码分析

Java POI 导出EXCEL经典实现

这次我们来看看equals()与hashCode()，你还记得他们吗?

java虚拟机学习之旅(1)——jps命令使用

十分钟学会Java8：lambda表达式和Stream API

发表评论

评分 ☆☆☆☆☆

您的评论一针见血（必填，该内容可在后台设置）

昵称 邮箱

网址

发表评论

繁