Java虚拟机基础知识

3 回复 30 查看

(https://www.shiyanlou.com/user/8490) 实验楼管理员 (https://www.shiyanlou.com/vip) 1小时前

技术分享 (https://www.shiyanlou.com/questions/?tag=技术分享)

写在前面

之前老大让做一些外包面试，我的问题很简单：

1、介绍一下工作中解决过比较有意思的问题。

2、HashMap使用中需要注意的点。

第一个问题主要是想了解一下对方项目经验的含金量，第二个问题则是测试下是否知道一些细节，比如HashMap是线程不安全的、用HashMap来做缓存的话可能导致内存泄露等，自我感觉问题设计的还可以:D~但是看了其他同事的题目就泪崩了：

1、设计模式XXX

2、垃圾回收XXX

擦，怎么感觉这个问题我也不会。。。

虚拟机给人的感觉像是操作系统、编译器：非常高大上。但是Java程序就跑在上面，遇到问题还得去排查，性能不行还得去优化，基础的知识还是需要的！

分享到微博

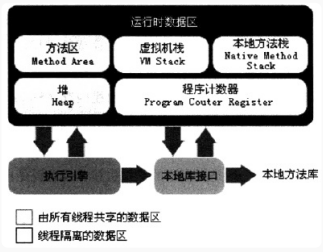
全部回答

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)

(https://www.shiyanlou.com/user/8490)

内存管理

Java虚拟机在执行的过程中会把它所管理的内存划分为若干个不同的数据区域，大致如下：



各部分的功能如下：

在内存管理部分比较大的一块内容是GC（垃圾回收），所谓垃圾回收就是将垃圾占用的内存回收掉。那么第一个问题：**什么是垃圾?**

- 引用计数算法：被引用次数为0的对象。

- 根搜索算法：从GC Roots沿着引用找不到的对象。

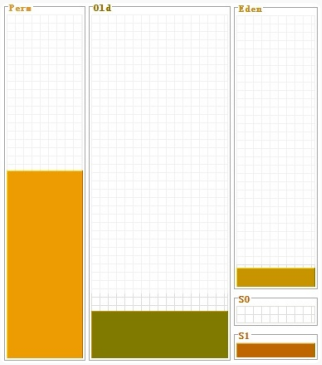
这里都提到了**引用**，在JDK 1.2之后Java就已经对引用的概念进行了扩充，那么第二个问题：**有哪些类型的引用?**

- 强引用：Object o = new Object()这种都是强引用。
- 弱引用：还有用但非必须的，在OOM之前被回收。
- 软引用：更弱的引用，在下次GC的时候被回收。
- 虚引用：最弱的，唯一的作用是在对象被回收的时候可以收到通知。

这里只有强引用才能对对象的生命周期造成影响。在虚拟机发展的过程中进化出不少垃圾回收算法，比如：

- 标记-清除算法
- 复制算法
- 标记-整理算法
- 分代收集算法

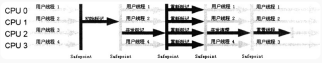
在实际中用到的回收器都是这几种算法的组合，比如从VisualVM中看到的内存是这样的（需要明白各部分都是怎样互相配合的）：



整体上来看是分代收集算法，而S0、S1这两部分可以看做是标记-整理算法。那么第三个问题：**常见的CMS垃圾回收器的执行流程是怎样的?**

- 初始标记：GC Roots直接关联的对象。
- 并发标记：Root Tracing。
- 重新标记：修复由于程序运行导致标记产生变动。
- 并发清除

具体如下图所示：



可以看到只有在初始标记和重新标记的时候才需要Stop The World，其他都是和用户线程一起执行，不要以为这就完美了，并行执行的过程会消耗掉一些CPU资源。

1小时前

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)

(https://www.shiyanlou.com/user/8490)

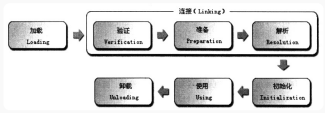
代码执行

把Java源码丢给JVM肯定是不能执行的，需要先用javac编译成class文件才行，那么第一个问题：**class文件的结构是怎样的?**

- 常量池
- 访问标志

- 类索引、父类索引和接口索引
- 字段表
- 方法表
- 属性表

虚拟机规范并没有规定在什么时候要加载类，但是规定了在遇到**new、反射、父类、Main**的时候需要初始化完成。整个类的生命周期如下：



在虚拟机中通过ClassLoader来进行类的加载，这地方需要明白：

- 两个类是否相同，除了类名外还需要判断ClassLoader是否相同。
- 双亲委派模式并不是一个强制约束。

在类加载完成之后就可以开始执行了，和线程运转相关的东西都放在栈帧中，其结构如下：

执行中具体调用哪个方法是个头疼的问题，需要处理：

- 静态分派：相同名称、不同参数类型的方法。
- 动态分派：继承中复写的方法。

字节码中的指令都是基于栈的操作，比如要完成1+1这样的计算，对应的指令如下：

```
iconst_1 // 将常量1压入栈
iconst_1
iadd // 把栈顶的两个值相加并出栈，然后把结果放回栈
istore_0 // 将栈顶的值放到局部变量表第0个Solt
```

解释执行的好处是下载后启动速度快，但是确定也非常明显：运行速度慢。JIT正是用来解决这个问题的，能够将**多次调用的方法、多次执行的循环体**编译成本地代码。

优化是个很好玩的题目，记得在参加一次变成比赛的时候用gcc -O3编译之后的代码把printf()都没输出了。。在JIT中比较常见的优化手段有：

1小时前

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)

程序执行一定会涉及到内存操作，在Java中定义了八种操作来完成：

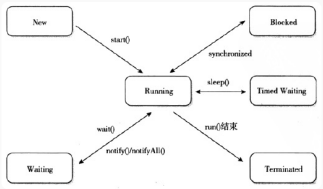
这里有必要讲一下**volatile**的作用，在使用到的时候能明白下面两条即可：

- 保证变量对所有线程是可见的。
- 禁止指令重排 (http://ifeve.com/jvm-memory-reordering/)优化。

如果Java中所有的操作都需要程序员来控制的话，会有大量的重复代码，而且写起来很累，那么我们可以通过**先行发生原则**来判断并行的两个操作是否存在冲突：

- 程序次序规则：单线程内按照程序书写顺序。
- 管程锁定规则：unlock必须在lock之前。
- volatile变量规则：写操作先行发生于读操作。
- 线程启动规则：Thread.start()先于线程的其他任意方法。
- 线程终止规则：线程中所有的操作都先于对此线程的终止检测。
- 线程中断规则：interrupt()先于中断检测。
- 对象终结规则：对象的初始化完成先于它的finalize()方法。
- 传递规则：如果A先于B、B先于C，那么A先于C。

Thread的底层实现还是比较麻烦的，但是最起码应该知道Thread的状态是如何进行转换：



最后，常见的同步方式是**synchronized**或者**aqs**的各种实现，这里就不讲了，因为每个都足够写一大篇。

附： JVM常用的参数和工具 (http://naotu.baidu.com/viewshare.html?shareId=avypaf5uum0w)

文章地址：http://wsztrush.github.io/%E7%BC%96%E7%A8%8B%E6%8A%80%E6%9C%AF/2015/05/08/JVM-Basis.html

作者：wsztrush

1小时前

登录后才能回答问题哟~

我要提问

标签

Linux (https://www.shiyanlou.com/questions/?tag=Linux)　Python (https://www.shiyanlou.com/questions/?tag=Python)

C/C++ (https://www.shiyanlou.com/questions/?tag=C/C++)　实验环境 (https://www.shiyanlou.com/questions/?tag=实验环境)

技术分享 (https://www.shiyanlou.com/questions/?tag=技术分享)　功能建议 (https://www.shiyanlou.com/questions/?tag=功能建议)

课程需求 (https://www.shiyanlou.com/questions/?tag=课程需求)　Java (https://www.shiyanlou.com/questions/?tag=Java)

其他 (https://www.shiyanlou.com/questions/?tag=其他)　SQL (https://www.shiyanlou.com/questions/?tag=SQL)

NodeJS (https://www.shiyanlou.com/questions/?tag=NodeJS)　Hadoop (https://www.shiyanlou.com/questions/?tag=Hadoop)

Web (https://www.shiyanlou.com/questions/?tag=Web)　常见问题 (https://www.shiyanlou.com/questions/?tag=常见问题)

Shell (https://www.shiyanlou.com/questions/?tag=Shell)　PHP (https://www.shiyanlou.com/questions/?tag=PHP)

Git (https://www.shiyanlou.com/questions/?tag=Git)　HTML (https://www.shiyanlou.com/questions/?tag=HTML)

HTML5 (https://www.shiyanlou.com/questions/?tag=HTML5)　信息安全 (https://www.shiyanlou.com/questions/?tag=信息安全)

网络 (https://www.shiyanlou.com/questions/?tag=网络)　GO (https://www.shiyanlou.com/questions/?tag=GO)

NoSQL (https://www.shiyanlou.com/questions/?tag=NoSQL)　Android (https://www.shiyanlou.com/questions/?tag=Android)

训练营 (https://www.shiyanlou.com/questions/?tag=训练营)　Ruby (https://www.shiyanlou.com/questions/?tag=Ruby)

Perl (https://www.shiyanlou.com/questions/?tag=Perl)

相关问题

[译]Linux性能分析的前60000毫秒 (https://www.shiyanlou.com/questions/3037)

MySQL之终端（Terminal）管理数据库、数据表、数据的基本操作 (https://www.shiyanlou.com/questions/3019)

C++静态库与动态库 (https://www.shiyanlou.com/questions/3017)

谈Runtime机制和使用的整体化梳理 (https://www.shiyanlou.com/questions/3010)

JavaScript：彻底理解同步、异步和事件循环(Event Loop) (https://www.shiyanlou.com/questions/3009)