链滴

Java 对象的创建过程

作者：vcjmhg

原文链接：https://ld246.com/article/1619614850780

来源网站：链滴

概述

在之前的一篇文章《关于java继承的哪些事》简单讲了Java创建对象的过程，但具体细节当时并没有细讲。因而本篇文章以HotSpot虚拟机为例，来讲一下**Java虚拟机是如何创建一个对象的?**

简单来说Java对象的创建过程总共分为5步：

Java创建对象的过程

类加载检查

首先当虚拟机遇到一个new指令时，首先会去检查这个参数能否在**常量池**中定位到这个类的符号引用并且检查这个符号引用所代表的的类是否已经被加载、连接、解析和初始化过，如果没有需要先执行的加载操作（详细过程可参考"类的加载流程"）。

分配内存

当类加**载检查**通过后，接下来虚拟机需要为新生对象分配内存，为对象分配空间的任务等同于把一块定大小的内存从 Java 堆中划分出来。分配内存的方式主要有两种：指针碰撞和空闲列表。

具体选择哪种方式取决于Java堆是否规整。而Java堆是否规整取决于垃圾收集器所采用的垃圾回收算是否具有空间压缩整理的能力。

具体来说，**指针碰撞**分配内存空间的过程如下：

在Java堆规整的情况下，所有被使用过的内存放到一边，所有未被使用过的内存放置到另一边，中间置一个指针作为分界点的指示器，当需要分配内存空间时，只需要将**空闲指针**向空闲内存方向移动对内存大小的位置即可。

该算法能够使用的前提必须是**空间是规整的**，因为如果空间是碎片化的，很明显该算法就会失效。

另一种算法是**空闲列表**，其分配内存的过程如下：

虚拟机会维护一个列表，该列表中会记录那些内存块是可用的，在分配内存时，会在空闲列表中找到块足够大的内存块来给对象实例，最后**更新列表记录。**

当然该算法也有不足，由于需要时刻维护一个空闲列表，因而会增加空间和时间开销，但优点就是它以用来对离散的空间进行内存分配。

最后在分配内存时，可能也会存在并发安全的问题，为了解决该问题，虚拟机采用两种方式来解决：

- **CAS+失败重试：** CAS 是乐观锁的一种实现方式。所谓乐观锁就是，每次不加锁而是假设没有冲突去完成某项操作，如果因为冲突失败就重试，直到成功为止。**虚拟机采用 CAS 配上失败重试的方式证更新操作的原子性。**
- **TLAB**: **为每一个线程预先在 Eden 区分配一块儿内存**，JVM 在给线程中的对象分配内存时，首先在 LAB 分配，当对象大于 TLAB 中的剩余内存或 TLAB 的内存已用尽时，再采用上述的 CAS 进行内存配

初始化零值

内存分配完成后，虚拟机需要将分配到的内存空间都初始化为零值（不包括对象头），这一步操作保了对象的实例字段在 Java 代码中可以不赋初始值就直接使用，程序能访问到这些字段的数据类型所应的零值。

设置对象头

初始化零值完成之后，**虚拟机要对对象进行必要的设置** ，例如这个对象是哪个类的实例、如何才能找类的元数据信息、对象的哈希码、对象的 GC 分代年龄等信息。 **这些信息存放在对象头中。** 另外，据虚拟机当前运行状态的不同，如是否启用偏向锁等，对象头会有不同的设置方式。

执行init()方法

上边一些列工作完成之后，从虚拟机角度来看，实际上一个对象已经产生了。但从Java程序的角度来，对象创建才刚刚开始---构造函数，即Class文件中的<init>()方法还没有执行所有字段还都是零值并没有按照构造方法来对对象进行初始化，因而最后一步需要执行<init>()按照程序员的意愿来对代进行初始化。

好了，前边对象的创建过程，我们已经讲清楚了，那有了对象之后，如何定位对象到对象进行使用呢?

对象的访问定位

建立对象就是为了使用对象，我们的 Java 程序通过栈上的 **reference 数据**来操作堆上的具体对象。象的访问方式由虚拟机实现而定，目前主流的访问方式有① **使用句柄** 和② **直接指针** 两种：

句柄： 如果使用句柄的话，那么 Java 堆中将会划分出一块内存来作为句柄池，reference 中存储的是对象的句柄地址，而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；

直接指针：如果使用直接指针访问，那么 Java 堆对象的布局中就必须考虑如何放置访问类型数据的关信息，而 reference 中存储的直接就是对象的地址。具体寻址过程如下图所示：

这两种对象访问方式各有优势。使用句柄来访问的最大好处是 reference 中存储的是稳定的句柄地址在对象被移动时只会改变句柄中的实例数据指针，而 reference 本身不需要修改。使用直接指针访问式最大的好处就是速度快，它节省了一次指针定位的时间开销。

总结

本文主要讲了对象创建的五大过程以及每一步的具体作用，后续为了使用Java对象因而讲了一些对象问定位相关的知识。

参考

1. Java内存区域
2. 《深入理解JVM虚拟机》