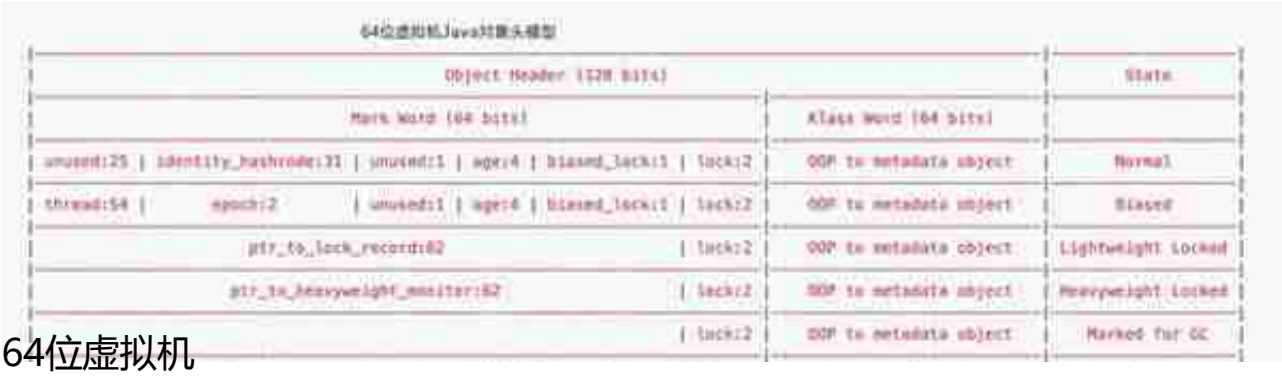
概述

Java 作为一个面向对象语言，给我们带来了多态，继承，封装等特性，使得我们可以利用这些特性很轻松的就能构建出易于扩展，易于维护的代码。作为一个Javaer，天天搞“对象”，那你写的对象究竟占用了多少内存呢？我们来看看你的“对象”是如何“败家”的。

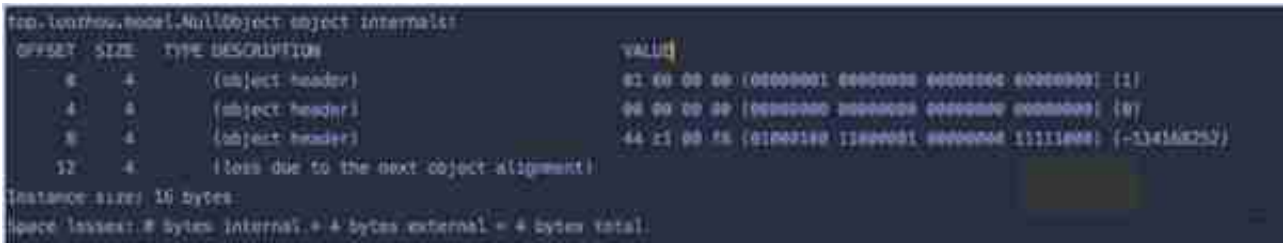
本文环境：jdk1.8_64

Java 对象头内存模型

想要了解Java对象究竟占用多少内存必定先要了解一个Java 对象的内存模型是怎么样的？由于我们的虚拟机是分为32位和64位，那肯定它们的模型也是有区别的，下面我列出列32位虚拟机和64位虚拟机下的Java对象头内存模型。



64位虚拟机

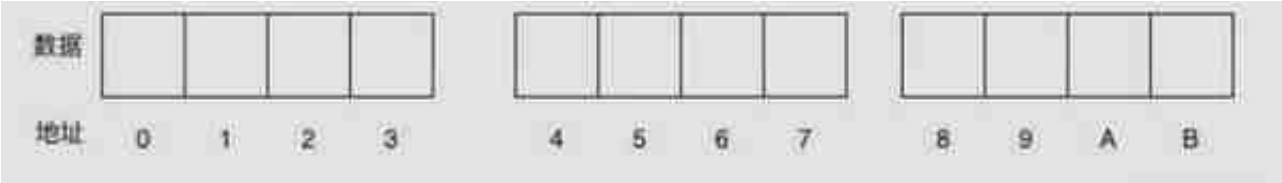


这里我们发现结果显示：Instance size：16 bytes,结果就是16字节，与我们之前预测的12字节不一样，为什么会这样呢？我们看到上图中有3行 object header，每个占用4字节，所以头部就是12字节，这里和我们的计算是一致的，最后一行是虚拟机填充的4字节，那为什么虚拟机要填充4个字节呢？

什么是内存对齐

想要知道为什么虚拟机要填充4个字节，我们需要了解什么是内存对齐？

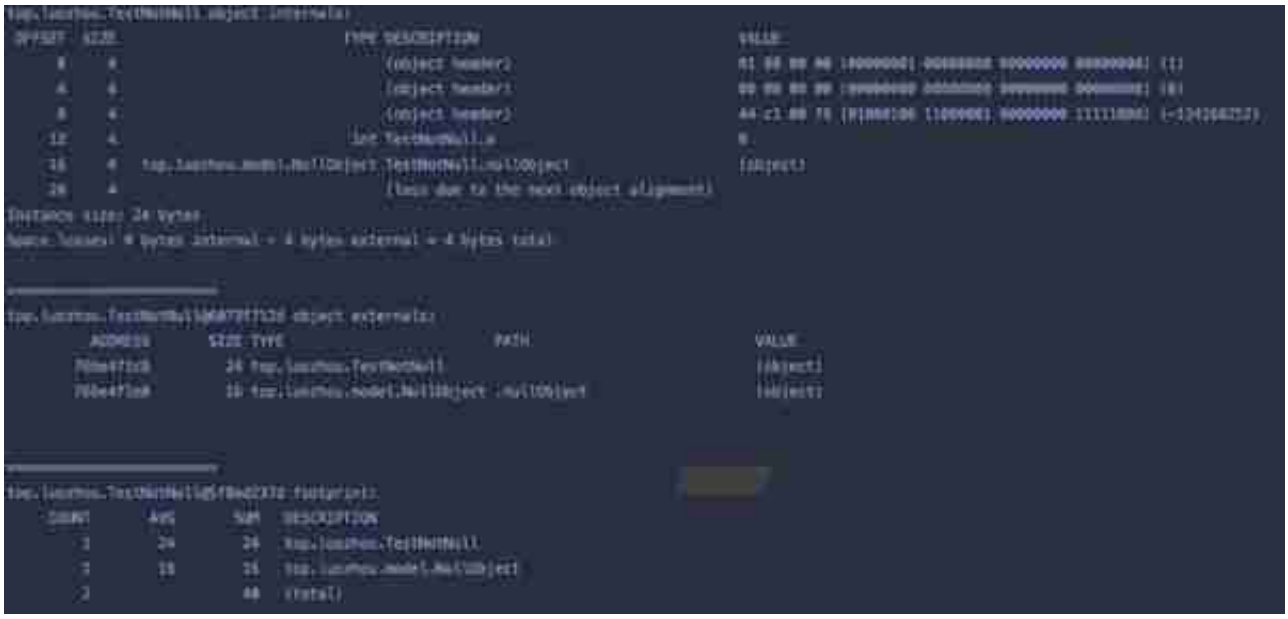
我们程序员看内存是这样的：



假设一个32位平台的 CPU，那它就会以4字节为粒度去读取内存块。那为什么需要内存对齐呢？主要有两个原因：

- 平台（移植性）原因：不是所有的硬件平台都能够访问任意地址上的任意数据。例如：特定的硬件平台只允许在特定地址获取特定类型的数据，否则会导致异常情况。
- 性能原因：若访问未对齐的内存，将会导致 CPU 进行两次内存访问，并且要花费额外的时钟周期来处理对齐及运算。而本身就对齐的内存仅需要一次访问就可以完成读取动作。

我用图例来说明 CPU 访问非内存对齐的过程：



我们可以看到TestNotNull的类占用空间是24字节，其中头部占用12字节，变量a是int类型，占用4字节,变量nullObject是引用，占用了4字节，最后填充了4个字节，总共是24个字节，与我们之前的预
测一致。
但是，因为我们实例化了NullObject,这个对象一会存在于内存中，所以我们还需要加上这个对象的内存占用16字节，那总共就是24bytes+16bytes=40bytes。我们图中最后的统计打印结果也是40字节，所以我们的分析正确。

这也是如何分析一个对象真正的占用多少内存的思路，根据这个思路加上openJDK的jol工具就可以基本的掌握自己写的“对象”究竟败家了你多少内存。

总结

本文我主要讲述了如何分析一个Java对象究竟占用多少内存空间，主要总结点如下：

1. Java对象头部内存模型在32位虚拟机和64位虚拟机是不一样的，64位虚拟机又分为开启指针压缩和不开启指针压缩两种对象头模型，所以总共有3种对象头模型。
2. 内存对齐主要是因为平台的原因和性能的原因，本文主要解析的是性能方面的原因。

3. 空对象的内存占用计算注意要计算内存对齐，非空对象的内存计算注意加上引用内存占用和原实例对象的空间占用。