涤生 jvm

<https://www.jianshu.com/u/150f36a73910>

Java 虚拟机面试题全面解析

<https://www.zybuluo.com/Yano/note/321063>

Java 8: 从永久代（PermGen）到元空间（Metaspace）

<https://blog.csdn.net/zhyhang/article/details/17246223/>

深入理解java虚拟机（全章节完整）

<https://blog.csdn.net/TJtulong/article/details/89598598>

《深入理解JVM》（周志明著）

<https://blog.csdn.net/qq_21306815/article/details/88906649>

《深入理解Java虚拟机》（周志明）阅读笔记

<https://blog.csdn.net/weixin_34351321/article/details/91889494>

========================================================================================================================================================================================================================================================================

Java8内存模型—永久代(PermGen)和元空间(Metaspace)

<https://www.cnblogs.com/paddix/p/5309550.html>

**PermGen（永久代）**

 “PermGen space”其实指的就是方法区。不过方法区和“PermGen space”又有着本质的区别。前者是 JVM 的规范，而后者则是 JVM 规范的一种实现，并且只有 HotSpot 才有 “PermGen space”，而对于其他类型的虚拟机，如 JRockit（Oracle）、J9（IBM） 并没有“PermGen space”。由于方法区主要存储类的相关信息，所以对于动态生成类的情况比较容易出现永久代的内存溢出。

**Metaspace（元空间）**

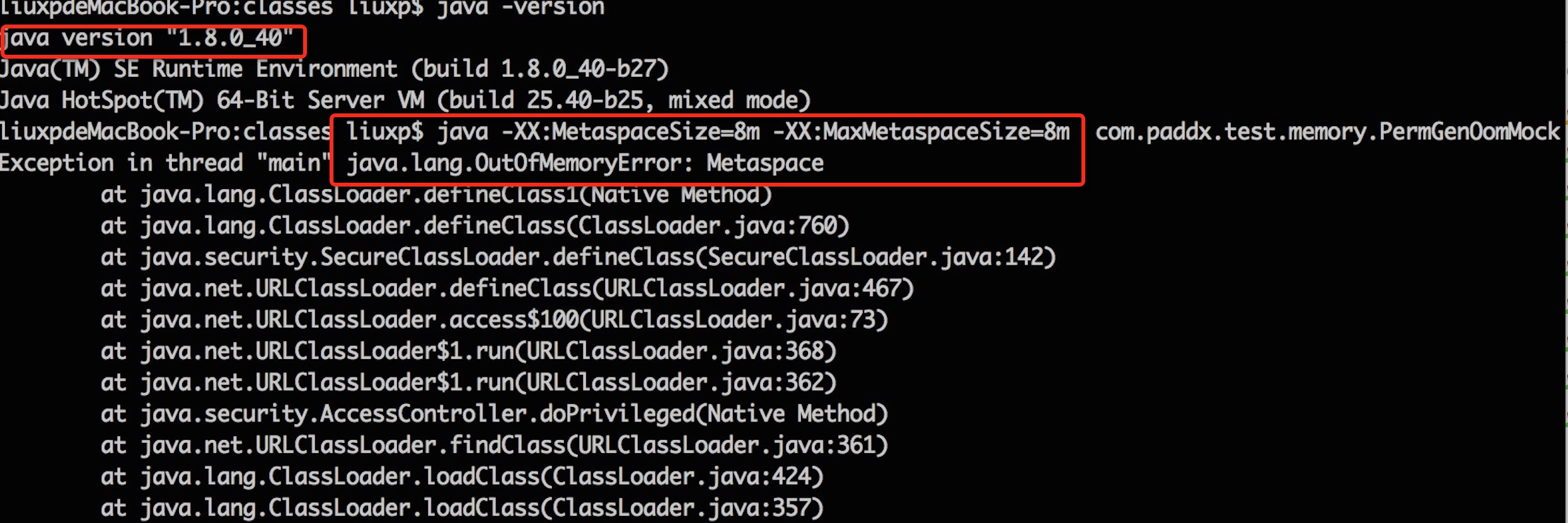
**元空间的本质和永久代类似，都是对JVM规范中方法区的实现。不过元空间与永久代之间最大的区别在于：元空间并不在虚拟机中，而是使用本地内存。因此，默认情况下，元空间的大小仅受本地内存限制，但可以通过以下参数来指定元空间的大小：**

-XX:MetaspaceSize，初始空间大小，达到该值就会触发垃圾收集进行类型卸载，同时GC会对该值进行调整：如果释放了大量的空间，就适当降低该值；如果释放了很少的空间，那么在不超过MaxMetaspaceSize时，适当提高该值。

-XX:MaxMetaspaceSize，最大空间，默认是没有限制的。

除了上面两个指定大小的选项以外，还有两个与 GC 相关的属性：

现在我们在 JDK 8下重新运行一下代码段 4，不过这次不再指定 PermSize 和 MaxPermSize。而是指定 MetaSpaceSize 和 MaxMetaSpaceSize的大小。输出结果如下：



从输出结果，我们可以看出，这次不再出现永久代溢出，而是出现了元空间的溢出。

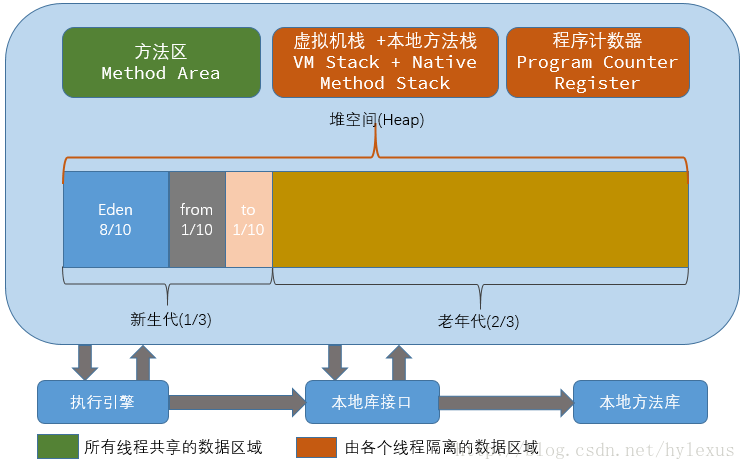
========================================================================================================================================================================================================================================================================

<https://www.cnblogs.com/dennyzhangdd/p/6770188.html>

<https://blog.csdn.net/bolg_hero/article/details/78189621>

**1.1 永久代（PermGen）在哪里？**

根据，hotspot jvm结构如下(虚拟机栈和本地方法栈合一起了)：



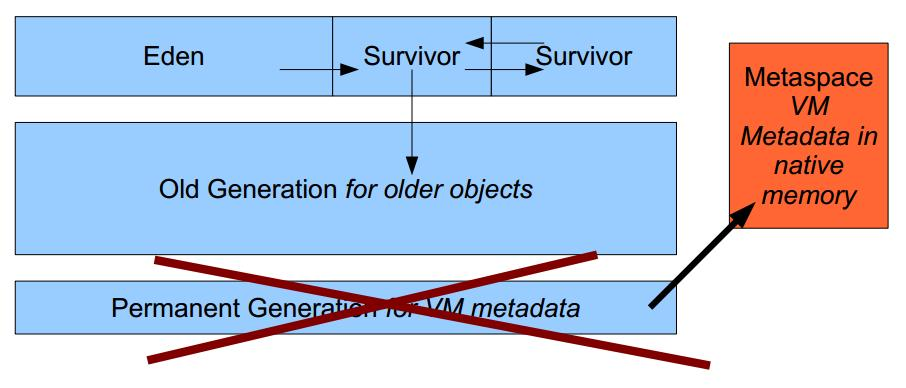
上图引自网络，但有个问题：方法区和heap堆都是线程共享的内存区域。

**关于方法区和永久代：**

在HotSpot JVM中，这次讨论的**永久代**，就是上图的方法区（JVM规范中称为方法区）。《Java虚拟机规范》只是规定了有方法区这么个概念和它的作用，并没有规定如何去实现它。在其他JVM上不存在永久代。

**1.2 JDK8永久代的废弃**

JDK8 永久代变化如下图：



1.新生代：Eden+From Survivor+To Survivor

2.老年代：OldGen

3.永久代(方法区的实现)：PermGen----->替换为Metaspace(本地内存中)

**二、为什么废弃永久代（PermGen）**

**2.1 官方说明**

参照JEP122：http://openjdk.java.net/jeps/122，原文截取：

**Motivation**

This is part of the JRockit and Hotspot convergence effort. JRockit customers do not need to configure the permanent generation (since JRockit does not have a permanent generation) and are accustomed to not configuring the permanent generation.

 即：移除永久代是为融合HotSpot JVM与 JRockit VM而做出的努力，因为JRockit没有永久代，不需要配置永久代。

**2.2 现实使用中易出问题**

由于永久代内存经常不够用或发生内存泄露，爆出异常*java.lang.OutOfMemoryError: PermGen*

**三、深入理解元空间（Metaspace）**

**3.1元空间的内存大小**

元空间是方法区的在HotSpot jvm 中的实现，方法区主要用于存储类的信息、常量池、方法数据、方法代码等。方法区逻辑上属于堆的一部分，但是为了与堆进行区分，通常又叫“非堆”。

元空间的本质和永久代类似，都是对JVM规范中方法区的实现。不过**元空间与永久代之间最大的区别在于：元空间并不在虚拟机中，而是使用本地内存。**，理论上取决于32位/64位系统可虚拟的内存大小。可见也不是无限制的，需要配置参数。

**3.2常用配置参数**

1.MetaspaceSize：-XX:MetaspaceSize=N

初始化的Metaspace大小，控制元空间发生GC的阈值。GC后，动态增加或降低MetaspaceSize。在默认情况下，这个值大小根据不同的平台在12M到20M浮动。使用[Java](http://lib.csdn.net/base/javase) -XX:+PrintFlagsInitial命令查看本机的初始化参数

这个参数是初始化的Metaspace大小，该值越大触发Metaspace GC的时机就越晚。随着GC的到来，虚拟机会根据实际情况调控Metaspace的大小，可能增加上线也可能降低。在默认情况下，这个值大小根据不同的平台在12M到20M浮动。使用java -XX:+PrintFlagsInitial | findstr Metaspace命令查看本机的初始化参数，-XX:Metaspacesize为21810376B（大约20.8M）。

2.MaxMetaspaceSize：-XX:MaxMetaspaceSize=N

限制Metaspace增长的上限，防止因为某些情况导致Metaspace无限的使用本地内存，影响到其他程序。在本机上该参数的默认值为4294967295B（大约4096MB）。

3.MinMetaspaceFreeRatio：-XX:MinMetaspaceFreeRatio=N

当进行过Metaspace GC之后，会计算当前Metaspace的空闲空间比，如果空闲比小于这个参数（即实际非空闲占比过大，内存不够用），那么虚拟机将增长Metaspace的大小。默认值为40，也就是40%。设置该参数可以控制Metaspace的增长的速度，太小的值会导致Metaspace增长的缓慢，Metaspace的使用逐渐趋于饱和，可能会影响之后类的加载。而太大的值会导致Metaspace增长的过快，浪费内存。

4.MaxMetasaceFreeRatio：-XX:MaxMetaspaceFreeRatio=N

当进行过Metaspace GC之后， 会计算当前Metaspace的空闲空间比，如果空闲比大于这个参数，那么虚拟机会释放Metaspace的部分空间。默认值为70，也就是70%。

5.MaxMetaspaceExpansion

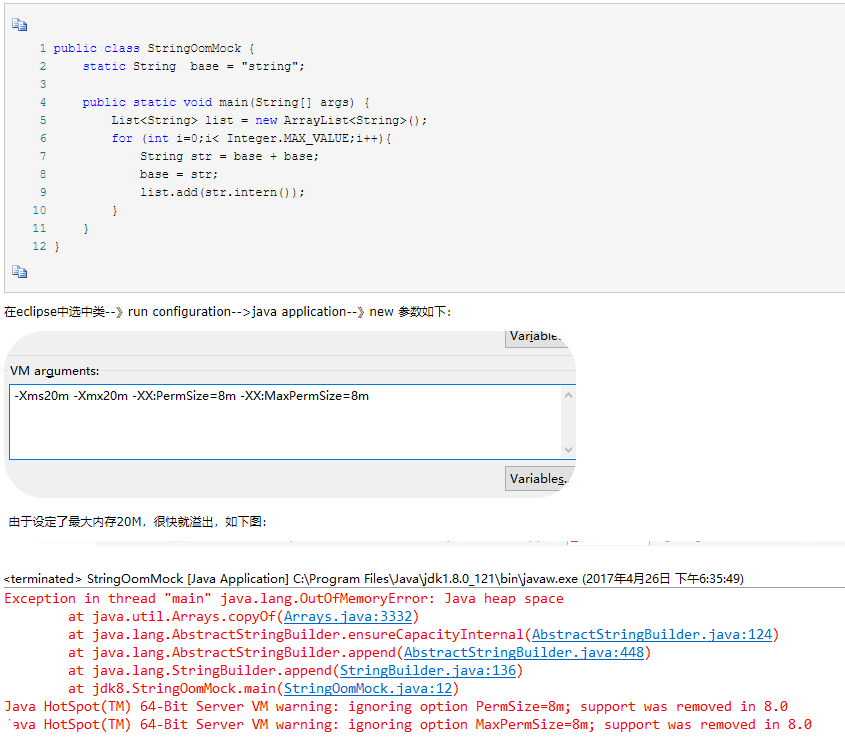
Metaspace增长时的最大幅度。在本机上该参数的默认值为5452592B（大约为5MB）。

6.MinMetaspaceExpansion

Metaspace增长时的最小幅度。在本机上该参数的默认值为340784B（大约330KB为）。

**3.3测试并追踪元空间大小**

**3.3.1.测试字符串常量**



可见在jdk8中：

1.字符串常量由永久代转移到堆中。

2.持久代已不存在，PermSize MaxPermSize参数已移除。（看图中最后两行）

**3.3.2.测试元空间溢出**

根据定义，我们以加载类来测试元空间溢出，代码如下：

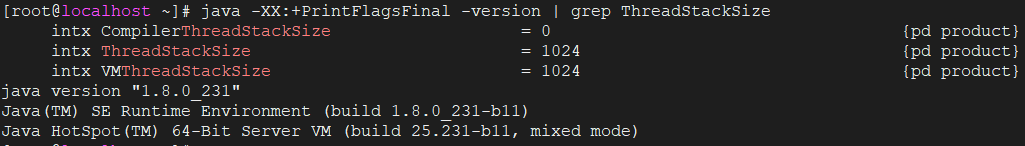




========================================================================================================================================================================================================================================================================

查看默认栈空间大小：

java -XX:+PrintFlagsFinal -version | grep ThreadStackSize



192.168.132.128/24

========================================================================================================================================================================================================================================================================

java新生代和老年代

<https://blog.csdn.net/qq_41383905/article/details/88937834>



新生代 ( Young ) 与老年代 ( Old ) 的比例的值为 1:2 ( 该值可以通过参数 –XX:NewRatio 来指定 )，即：新生代 ( Young ) = 1/3 的堆空间大小。老年代 ( Old ) = 2/3 的堆空间大小。其中，新生代 ( Young ) 被细分为 Eden 和 两个 Survivor 区域，这两个 Survivor 区域分别被命名为 from 和 to，以示区分。

默认的，Edem : from : to = 8 : 1 : 1 ( 可以通过参数 –XX:SurvivorRatio 来设定 )，即： Eden = 8/10 的新生代空间大小，from = to = 1/10 的新生代空间大小。

================================================================================================================================================================================