https://www.jianshu.com/p/9e4ccd705709

**Visual GC 插件使用**

[](https://www.jianshu.com/u/f452fd5ac95f)

[帅气滴糟老头](https://www.jianshu.com/u/f452fd5ac95f) 关注

2019.02.28 18:02\* 字数 1236 阅读 209评论 0喜欢 0

**下载安装**

**首先要下载VisualVM**

下载地址：[https://visualvm.github.io/download.html](https://visualvm.github.io/download.html" \t "_blank)

如果是mac系统就下载下面的，下载完成后将压缩包解压到本地目录，无需安装，找到bin目录下的visualvm.exe双击即可启动。

**然后安装Visual GC 插件**

点击菜单Tools，选择Pulgins，在第二个选项中找到Visual GC，选中，点击Install，然后一路下一步即可安装成功，如下图

**插件界面介绍**

安装完插件后重启软件，让后打开idea，随后便可以在左边了Local中看到启动的idea应用，双击即可进入监控页面，如下图：

右边就是Visual GC 插件的主要界面了，我们可以看到软件运行时的内存变化情况

下面对上图中的各个窗口区域做简单介绍，整个界面分为三个区域，分别为：Spaces、Graphs和Histogram。

**Spaces窗口：**

上图是呈现了程序运行时我们比较关注的几个区域的内存使用情况

* **Metaspace：**方法区，如果JDK1.8之前的版本，就是Perm，JDK7和之前的版本都是以永久代(PermGen)来实现方法区的，JDK8之后改用元空间来实现(MetaSpace)。
* **Old：**老年代
* **Eden:** 新生代Eden区
* **S0和S1**：新生代的两个 Survivor 区

**Graphs窗口：**

该窗口区域包含8个图标，以时间为横坐标动态展示各个指标的运行状态

下面从上往下对上图中的各个图标表及其状态进行说明

* **Compile Time：编译情况**  
  24266 compoles - 39.416s 表示编译总数为24266，编译总耗时为39.416s。  
  一个脉冲表示一次JIT编译，脉冲越宽表示编译时间越长。
* **Class Loader Time：类加载情况**  
  49052 loaded，39 unloaded - 29.937s表示已加载的数量为49052，卸载的数量为39，耗时为29.537s。
* **GC Time：总的（包含新生代和老年代）gc情况记录**  
  123 collections，859.203ms Last Cause：Allocation Failure表示一共经历了123次gc(包含Minor GC和Full GC)，总共耗时859.203ms。
* **Eden Space：新生代Eden区内存使用情况**  
  (200.00M，34.125M): 31.52M，109 collections，612.827ms表示Eden区的最大容量为200M，当前容量为34.125M，当前已使用31.52M，从开始监控到现在在该内存区域一共发生了109次gc(Minor GC)，gc总耗时为612.827ms。
* **Survivor 0和Survivor 1：新生代的两个Survivor区内存使用情况**  
  (25.000M，4.250M):1.757M表示该Survivor区的最大容量为25M（默认为Eden区的1/8），当前已用1.757M。
* **Old Gen：老年代内存使用情况**  
  (500.000M,255.195M):206.660M,14 collections，246.375ms表示老年区的最大容量为500M，当前容量为255.195M，当前已用206.660M，从开始监控到现在在该内存区域一共发生了14次gc(Full GC)，gc总耗时为246.375ms，换算下可以看出单次Full GC要比Minor GC耗时长很多。
* **Metaspace：方法区内存使用情况**  
  (1.053G,278.250M):262.345M表示方法区最大容量为1.053G,当前容量为278.250M,当前使用量为262.345MM。

**Histogram窗口：**

Histogram窗口是对当前正在被使用的Survivor区内存使用情况的详细描述，如下

* **Tenuring Threshold：**我们知道Survivor区中的对象有一套晋升机制，就是其中的每个对象都有一个年龄标记，每当对象在一次Minor GC中存活下来，其年龄就会+1，当对象的年龄大于一个阈值时，就会进入老年代，这个阈值就是Tenuring Threshold，要注意这个值不是固定不变的，一般情况下Tenuring Threshold会与Max Tenuring Threshold大小保持一致，可如果某个时刻Servivor区中相同年龄的所有对象的内存总等于Survivor空间的一半，那Tenuring Threshold就会等于该年龄，同时大于或等于该年龄的所有对象将进入老年代。
* **Max Tenuring Threshold：**表示新生代中对象的最大年龄值，这个值在JDK1.8中默认为6，在JDK1.7及之前的版本中默认为15，可以通过参数-XX:MaxTenuringThreshold来指定。
* **Desired Survivor Size：**Survivor空间大小验证阈值(默认是survivor空间的一半)，用于给Tenuring Threshold判断对象是否提前进入老年代。
* **Current Survivor Size：**当前Survivor空间大小，单位为kb。
* **Histogram柱状图：**表示Survivor中不同年龄段对象分布。