参考：<https://www.ibm.com/developerworks/cn/java/j-lo-visualvm/>

# 使用 VisualVM 进行性能分析及调优

虞 立军, 王 建光, 和 倪 力  
2013 年 2 月 18 日发布

[Weibo](http://service.weibo.com/share/share.php?url=http%3A%2F%2Fwww.ibm.com%2Fdeveloperworks%2Fcn%2Fjava%2Fj-lo-visualvm%2Findex.html%25&title=%E4%BD%BF%E7%94%A8%20VisualVM%20%E8%BF%9B%E8%A1%8C%E6%80%A7%E8%83%BD%E5%88%86%E6%9E%90%E5%8F%8A%E8%B0%83%E4%BC%98&language=zh_cn)[Google+](https://plus.google.com/share?url=http%3A%2F%2Fwww.ibm.com%2Fdeveloperworks%2Fcn%2Fjava%2Fj-lo-visualvm%2Findex.html&t=%E4%BD%BF%E7%94%A8%20VisualVM%20%E8%BF%9B%E8%A1%8C%E6%80%A7%E8%83%BD%E5%88%86%E6%9E%90%E5%8F%8A%E8%B0%83%E4%BC%98)[用电子邮件发送本页面](mailto:?subject=%E4%BD%BF%E7%94%A8%20VisualVM%20%E8%BF%9B%E8%A1%8C%E6%80%A7%E8%83%BD%E5%88%86%E6%9E%90%E5%8F%8A%E8%B0%83%E4%BC%98&body=http%3A%2F%2Fwww.ibm.com%2Fdeveloperworks%2Fcn%2Fjava%2Fj-lo-visualvm%2Findex.html)

[Comments](https://www.ibm.com/developerworks/cn/java/j-lo-visualvm/#icomments)

[8](https://www.ibm.com/developerworks/cn/java/j-lo-visualvm/" \l "icomments)

## 概述

开发大型 Java 应用程序的过程中难免遇到内存泄露、性能瓶颈等问题，比如文件、网络、数据库的连接未释放，未优化的算法等。随着应用程序的持续运行，可能会造成整个系统运行效率下降，严重的则会造成系统崩溃。为了找出程序中隐藏的这些问题，在项目开发后期往往会使用性能分析工具来对应用程序的性能进行分析和优化。

VisualVM 是一款免费的性能分析工具。它通过 jvmstat、JMX、SA（Serviceability Agent）以及 Attach API 等多种方式从程序运行时获得实时数据，从而进行动态的性能分析。同时，它能自动选择更快更轻量级的技术尽量减少性能分析对应用程序造成的影响，提高性能分析的精度。

本文将对 VisualVM 的主要功能逐一介绍并探讨如何利用获得的数据进行性能分析及调优。

## 背景知识

### 性能分析的主要方式

* 监视：监视是一种用来查看应用程序运行时行为的一般方法。通常会有多个视图（View）分别实时地显示 CPU 使用情况、内存使用情况、线程状态以及其他一些有用的信息，以便用户能很快地发现问题的关键所在。
* 转储：性能分析工具从内存中获得当前状态数据并存储到文件用于静态的性能分析。Java 程序是通过在启动 Java 程序时添加适当的条件参数来触发转储操作的。它包括以下三种：
  + 系统转储：JVM 生成的本地系统的转储，又称作核心转储。一般的，系统转储数据量大，需要平台相关的工具去分析，如 Windows 上的 windbg 和 Linux 上的 gdb。
  + Java 转储：JVM 内部生成的格式化后的数据，包括线程信息，类的加载信息以及堆的统计数据。通常也用于检测死锁。
  + 堆转储：JVM 将所有对象的堆内容存储到文件。
* 快照：应用程序启动后，性能分析工具开始收集各种运行时数据，其中一些数据直接显示在监视视图中，而另外大部分数据被保存在内部，直到用户要求获取快照，基于这些保存的数据的统计信息才被显示出来。快照包含了应用程序在一段时间内的执行信息，通常有 CPU 快照和内存快照两种类型。
  + CPU 快照：主要包含了应用程序中函数的调用关系及运行时间，这些信息通常可以在 CPU 快照视图中进行查看。
  + 内存快照：主要包含了内存的分配和使用情况、载入的所有类、存在的对象信息及对象间的引用关系等。这些信息通常可以在内存快照视图中进行查看。
* 性能分析：性能分析是通过收集程序运行时的执行数据来帮助开发人员定位程序需要被优化的部分，从而提高程序的运行速度或是内存使用效率，主要有以下三个方面：
  + CPU 性能分析：CPU 性能分析的主要目的是统计函数的调用情况及执行时间，或者更简单的情况就是统计应用程序的 CPU 使用情况。通常有 CPU 监视和 CPU 快照两种方式来显示 CPU 性能分析结果。
  + 内存性能分析：内存性能分析的主要目的是通过统计内存使用情况检测可能存在的内存泄露问题及确定优化内存使用的方向。通常有内存监视和内存快照两种方式来显示内存性能分析结果。
  + 线程性能分析：线程性能分析主要用于在多线程应用程序中确定内存的问题所在。一般包括线程的状态变化情况，死锁情况和某个线程在线程生命期内状态的分布情况等

## VisualVM 安装

VisualVM 是一个性能分析工具，自从 JDK 6 Update 7 以后已经作为 Oracle JDK 的一部分，位于 JDK 根目录的 bin 文件夹下。VisualVM 自身要在 JDK6 以上的版本上运行，但是它能够监控 JDK1.4 以上版本的应用程序。下面主要介绍如何安装 VisualVM 以及各种 VisualVM 上的插件。

### 安装 VisualVM

VisualVM 项目的官方网站目前提供英文版本和多语言支持版本下载。多语言版本主要支持英语、日语以及中文三种语言。如果下载安装多语言版本的 VisualVM，安装程序会依据操作系统的当前语言环境去安装相应 VisualVM 的语言版本。最新 VisualVM 版本主要支持的操作系统包括：Microsoft Windows (7, Vista, XP, Server)、Linux、Sun Solaris、Mac OS X、HP-UX 11i。本文以 Microsoft Windows XP 为安装环境并支持中文。

* 从 [VisualVM 项目的官方网站](http://visualvm.java.net/)上下载 VisualVM 安装程序。
* 将 VisualVM 安装程序解压缩到本地系统。
* 导航至 VisualVM 安装目录的 bin 目录，然后启动 jvisualvm.exe。

### 安装 VisualVM 上的插件

[VisualVM 插件中心](http://visualvm.java.net/pluginscenters.html)提供很多插件以供安装向 VisualVM 添加功能。可以通过 VisualVM 应用程序安装，或者从 [VisualVM 插件中心](http://visualvm.java.net/pluginscenters.html)手动下载插件，然后离线安装。另外，用户还可以通过下载插件分发文件 (.nbm 文件 ) 安装第三方插件为 VisualVM 添加功能。

从 VisualVM 插件中心安装插件安装步骤 :

* 从主菜单中选择“工具”>“插件”。
* 在“可用插件”标签中，选中该插件的“安装”复选框。单击“安装”。
* 逐步完成插件安装程序。

##### 图 1. VisualVM 插件管理器



根据 .nbm 文件安装第三方插件安装步骤 :

* 从主菜单中选择“工具”>“插件”。
* 在“已下载”标签中，点击"添加插件"按钮，选择已下载的插件分发文件 (.nbm) 并打开。
* 选中打开的插件分发文件，并单击"安装"按钮，逐步完成插件安装程序。

##### 图 2. 通过 .nbm 文件安装 VisualVM 插件



## 功能介绍

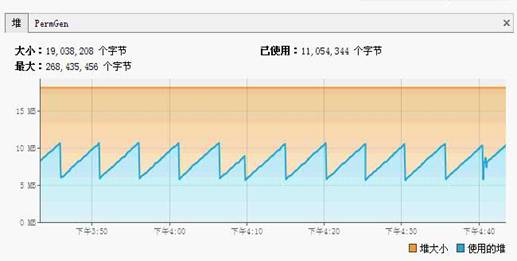
下面我们将介绍性能分析的几种常见方式以及如何使用 VisualVM 性能分析工具进行分析。

### 内存分析

VisualVM 通过检测 JVM 中加载的类和对象信息等帮助我们分析内存使用情况，我们可以通过 VisualVM 的监视标签和 Profiler 标签对应用程序进行内存分析。

在监视标签内，我们可以看到实时的应用程序内存堆以及永久保留区域的使用情况。

##### 图 3. 内存堆使用情况



##### 图 4. 永久保留区域使用情况



此外，我们也可以通过 Applications 窗口右击应用程序节点来启用“在出现 OOME 时生成堆 Dump”功能，当应用程序出现 OutOfMemory 例外时，VisualVM 将自动生成一个堆转储。

##### 图 5. 开启“在出现 OOME 时生成堆”功能



在 Profiler 标签，点击“内存”按钮将启动一个内存分析会话，等 VisualVM 收集和统计完相关性能数据信息，将会显示在性能分析结果。通过内存性能分析结果，我们可以查看哪些对象占用了较多的内存，存活的时间比较长等，以便做进一步的优化。

此外，我们可以通过性能分析结果下方的类名过滤器对分析结果进行过滤。

##### 图 6. 内存分析结果

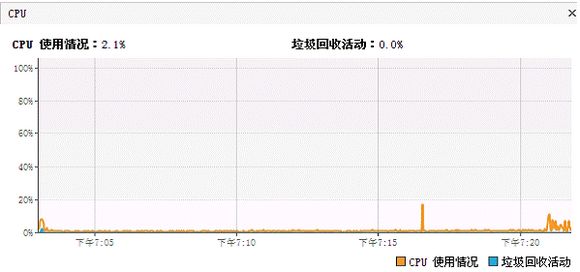


### CPU 分析

VisualVM 能够监控应用程序在一段时间的 CPU 的使用情况，显示 CPU 的使用率、方法的执行效率和频率等相关数据帮助我们发现应用程序的性能瓶颈。我们可以通过 VisualVM 的监视标签和 Profiler 标签对应用程序进行 CPU 性能分析。

在监视标签内，我们可以查看 CPU 的使用率以及垃圾回收活动对性能的影响。过高的 CPU 使用率可能是由于我们的项目中存在低效的代码，可以通过 Profiler 标签的 CPU 性能分析功能进行详细的分析。如果垃圾回收活动过于频繁，占用了较高的 CPU 资源，可能是由内存不足或者是新生代和旧生代分配不合理导致的等。

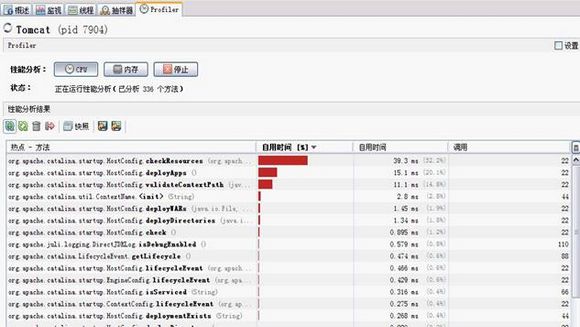
##### 图 7. CPU 使用情况



在 Profiler 标签，点击“CPU”按钮启动一个 CPU 性能分析会话 ,VisualVM 会检测应用程序所有的被调用的方法。当进入一个方法时，线程会发出一个“method entry”的事件，当退出方法时同样会发出一个“method exit”的事件，这些事件都包含了时间戳。然后 VisualVM 会把每个被调用方法的总的执行时间和调用的次数按照运行时长展示出来。

此外，我们也可以通过性能分析结果下方的方法名过滤器对分析结果进行过滤。

##### 图 8. CPU 性能分析结果



### 线程分析

Java 语言能够很好的实现多线程应用程序。当我们对一个多线程应用程序进行调试或者开发后期做性能调优的时候，往往需要了解当前程序中所有线程的运行状态，是否有死锁、热锁等情况的发生，从而分析系统可能存在的问题。

在 VisualVM 的监视标签内，我们可以查看当前应用程序中所有活动线程和守护线程的数量等实时信息。

##### 图 9. 活跃线程情况



VisualVM 的线程标签提供了三种视图，默认会以时间线的方式展现。另外两种视图分别是表视图和详细信息视图。

时间线视图上方的工具栏提供了缩小，放大和自适应三个按钮，以及一个下拉框，我们可以选择将所有线程、活动线程或者完成的线程显示在视图中。

##### 图 10. 线程时间线视图



##### 图 11. 线程表视图



我们在详细信息视图中不但可以查看所有线程、活动线程和结束的线程的详细数据，而且也可以查看某个线程的详细情况。

##### 图 12. 线程详细视图



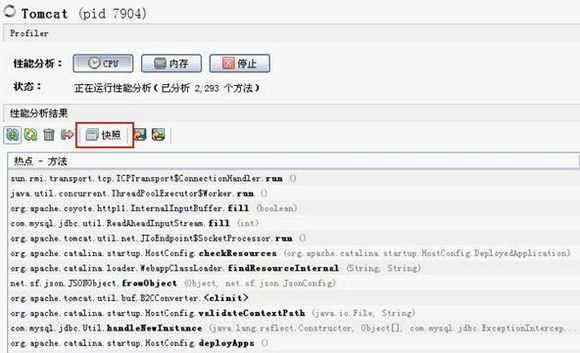
## 快照功能

我们可以使用 VisualVM 的快照功能生成任意个性能分析快照并保存到本地来辅助我们进行性能分析。快照为捕获应用程序性能分析数据提供了一个很便捷的方式因为快照一旦生成可以在任何时候离线打开和查看，也可以相互传阅。

VisualVM 提供了两种类型的快照：

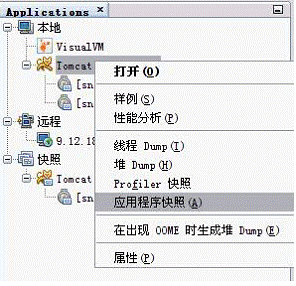
* Profiler 快照：当有一个性能分析会话（内存或者 CPU）正在进行时，我们可以通过性能分析结果工具栏的“快照”按钮生成 Profiler 快照捕获当时的性能分析数据。

##### 图 13. Profiler 快照



* 应用程序快照：我们可以右键点击左侧 Applications 窗口中应用程序节点，选择“应用程序快照”为生成一个应用程序快照。应用程序快照会收集某一时刻的堆转储，线程转储和 Profiler 快照，同时也会捕获 JVM 的一些基本信息。

##### 图 14. 应用程序快照

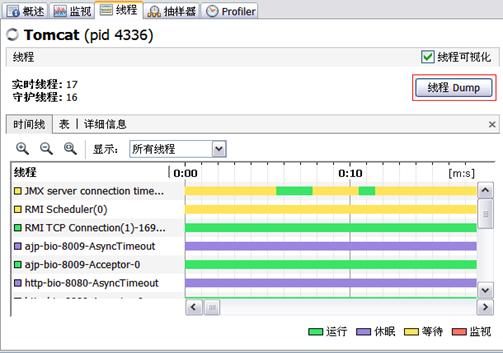


## 转储功能

### 线程转储的生成与分析

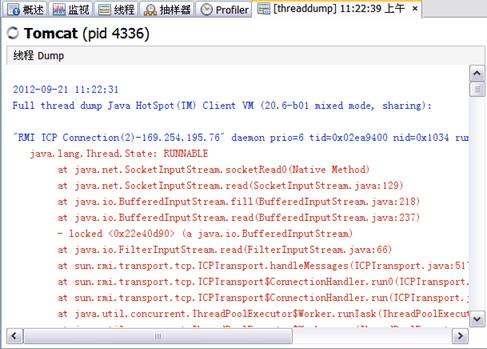
VisualVM 能够对正在运行的本地应用程序生成线程转储，把活动线程的堆栈踪迹打印出来，帮助我们有效了解线程运行的情况，诊断死锁、应用程序瘫痪等问题。

##### 图 15. 线程标签及线程转储功能



当 VisualVM 统计完应用程序内线程的相关数据，会把这些信息显示新的线程转储标签。

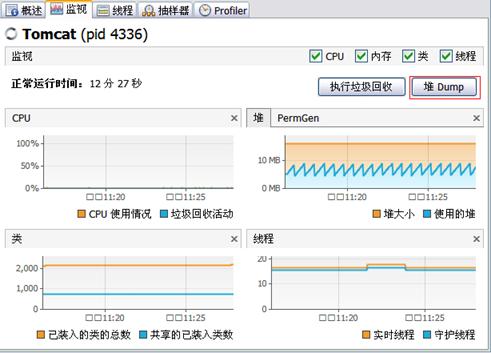
##### 图 16. 线程转储结果



### 堆转储的生成与分析

VisualVM 能够生成堆转储，统计某一特定时刻 JVM 中的对象信息，帮助我们分析对象的引用关系、是否有内存泄漏情况的发生等。

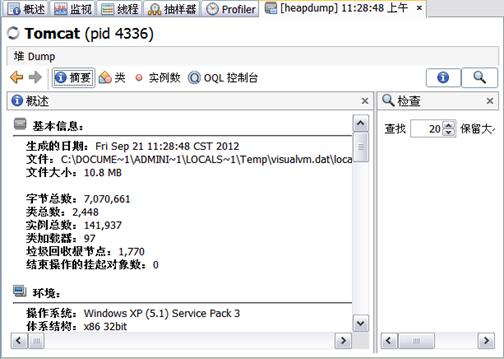
##### 图 17. 监视标签及堆转储功能



当 VisualVM 统计完堆内对象数据后，会把堆转储信息显示在新的堆转储标签内，我们可以看到摘要、类、实例数等信息以及通过 OQL 控制台执行查询语句功能。

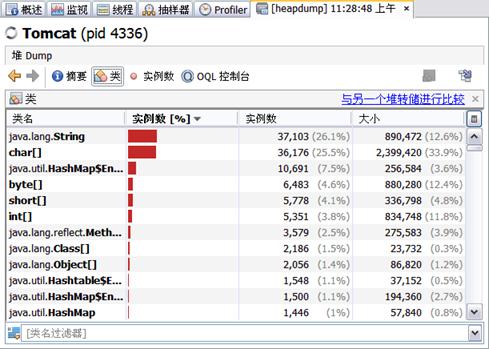
堆转储的摘要包括转储的文件大小、路径等基本信息，运行的系统环境信息，也可以显示所有的线程信息。

##### 图 18. 堆转储的摘要视图



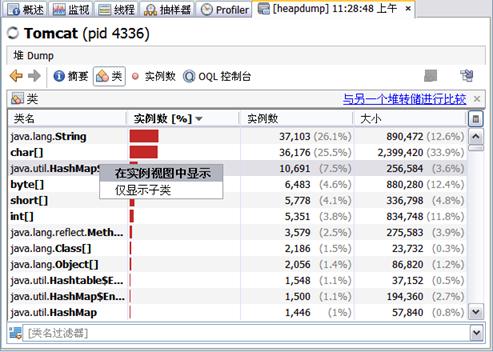
从类视图可以获得各个类的实例数和占用堆大小数，分析出内存空间的使用情况，找出内存的瓶颈，避免内存的过度使用。

##### 图 19. 堆转储的类视图

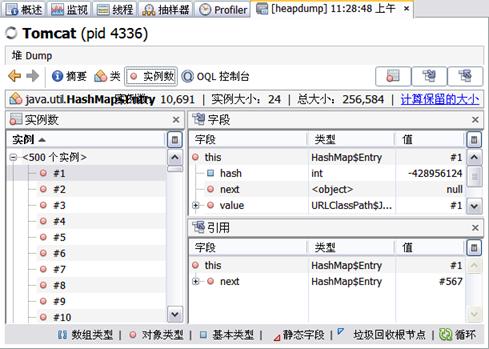


通过实例数视图可以获得每个实例内部各成员变量的值以及该实例被引用的位置。首先需要在类视图选择需要查看实例的类。

##### 图 20. 选择查询实例数的类



##### 图 21. 实例数视图

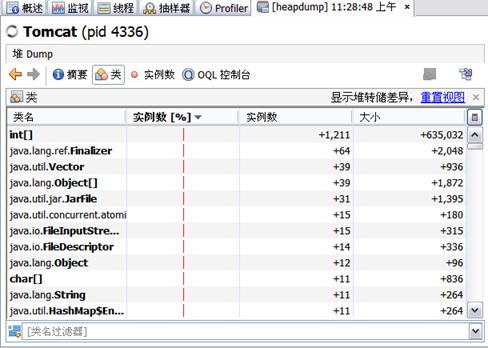


此外，还能对两个堆转储文件进行比较。通过比较我们能够分析出两个时间点哪些对象被大量创建或销毁。

##### 图 22. 堆转储的比较



##### 图 23. 堆转储的比较结果



线程转储和堆转储均可以另存成文件，以便进行离线分析。

##### 图 24. 转储文件的导出



## 总结

本文首先简要列举了一些性能分析相关的背景知识。然后介绍了 VisualVM 的下载和安装。最后从内存性能、CPU 性能、快照功能以及转储功能四个方面展开，进一步说明了如何使用 VisualVM 进行性能分析。通过本文的介绍，相信读者对性能分析会有一定的了解，并可以利用 VisualVM 进行性能分析。