向IT技术大牛们学习！

开源力量公开课第１期：

生产环境下的Java排错调优

2012年12月25日　18:30-21:30

http://www.osforce.cn

开源力量公开课，每周二晚线上线下同时开课

开源力量
Open Source Force

生产环境下的Java排错调优

@施懿民

知乎软件

开源力量
Open Source Force

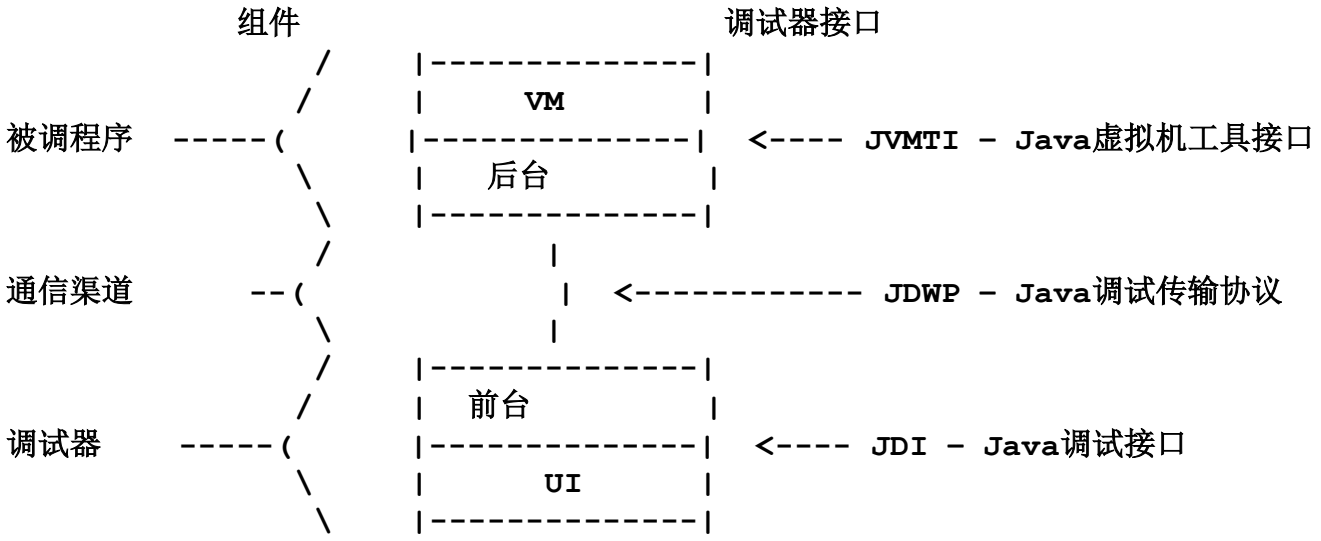
远程调试

知平软件

远程调试架构

开源力量 Open Source Force

- Java平台调试器架构



知乎软件

远程调试原理

开源力量 Open Source Force

- 通过客户机-服务器架构，可以在本地调试 Java 程序，也可以通过网络进行远程调试，JPDA 规范中的两个术语：连接器和传输。连接器是一个 JDI 抽象，用来在调试器应用程序和目标 VM 之间建立连接。传输定义应用程序如何进行访问，以及数据如何在前端和后端之间传输。连接器 “映射” 到可用的传输类型和连接模式。在 Sun 的 JPDA 参考实现中，为 Microsoft® Windows® 提供了两个传输机制：套接字传输和共享内存传输。可用的连接器：
- 连接套接字连接器
- 连接共享内存连接器
- 监听套接字连接器
- 监听共享内存连接器
- 启动命令行连接器

远程调试命令参数

- -Xdebug:　启用调试特性。
- -Xrunjdwp:<sub-options>：在目标 VM 中加载 JDWP 实现。它通过传输和 JDWP 协议与独立的调试器应用程序通信。下面介绍一些特定的子选项。
- 从 Java V5 开始，您可以使用 -agentlib:jdwp 选项，而不是 -Xdebug 和 -Xrunjdwp。但如果连接到 V5 以前的 VM，只能选择 -Xdebug 和 -Xrunjdwp。
- -Xrunjdwp 子选项。
- transport：这里通常使用套接字传输。但是在 Windows 平台上也可以使用共享内存传输。
- Server：如果值为 y，目标应用程序监听将要连接的调试器应用程序。否则，它将连接到特定地址上的调试器应用程序。
- address：这是连接的传输地址。如果服务器为 n，将尝试连接到该地址上的调试器应用程序。否则，将在这个端口监听连接。
- suspend：如果值为 y，目标 VM 将暂停，直到调试器应用程序进行连接。

1、被调试程序当作调试服务器。

-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765

2、被调程序当作调试客户端。

-Xdebug -Xrunjdwp:transport=dt_socket,address=127.0.0.1:8000

演示 – 设置远程调试

断点的实现原理

开源力量 Open Source Force

- 断点（Break Point）可以说是调试器的关键技术，需要软件和硬件的协作才能实现。一般断点的实现方式有下面几种：

1. 通过特定的指令通知中央处理器（CPU）来中断程序的执行。
2. 通过设置特定的寄存器来通知中央处理器中断程序的执行。
3. 通过强制处理器触发异常来中断程序执行并将控制权转交给调试器。

- 在 Intel兼容的处理器架构上，一般调试器是通过在进程中特定的位置插入INT 3指令来实现断点的。

- 调试器提供的单步执行，单步跳过执行以及跳出函数等功能，都是断点的变种。

1、在C程序中嵌入断点。

演示

特殊断点

开源力量
Open Source Force

- 除了简单的每次执行到断点位置中断程序执行这一种方式，调试器一般都提供了如下几种断点：

1. 条件断点 - 可以指定触发断点的条件，避免每次重复触发断点降低调试工作效率。
2. 监视断点 - 可以在访问数据的时候，中断程序的执行。
3. 函数断点 - 可以在执行函数前或者退出函数前中断程序的执行。
4. 异常断点 - 当程序发生指定异常的时候，中断程序的执行，第一时间发现问题所在。
5. 类型断点 - 当程序试图加载某个类型的时候，中断程序的执行。

开源力量
Open Source Force

1、禁用所有的断点。
2、设置条件断点。
3、监视断点。
4、异常断点。
5、函数断点。
6、在类型加载的时候中断

演示

知乎软件

1、在堆栈的任意位置重新执行语句。
2、在程序启动时进行调试。
3、使用变量窗口的逻辑视图。
4、单步过滤调试。
5、计算表达式。

演示 – Eclipse其他调试技巧

1、使用jdb调试java程序。

演示

在Emacs里运行JDB

开源力量 Open Source Force

- 由于直接使用JDB调试时，浏览源代码时很不方便，如果机器上安装了emacs，可以直接在emacs里启动JDB，获取跟eclipse相近的源代码级别的调试体验。

- 在emacs里运行JDB的方法：
- 在emacs里按下ALT+X键，在提示符后面输入JDB，敲击回车。
- 接着再输入JDB的启动参数。
- 按下CTRL + X，2键，将emacs分屏。
- 再按CTRL + X，B键，将其中一个屏幕显示源代码。
- 按CTRL + X，O键，再两个屏幕间切换。
- 在JDB的那个窗口里输入正常的调试命令。

开源力量
Open Source Force

JAVA内存调优

知平软件

概 述

开源力量 Open Source Force

- Java虽然有垃圾回收机制，但是程序编写不慎，还是会发生内存泄露导致OutOfMemoryError的发生。本节课讲解了GC的机制，以及JDK自带的HPROF工具以分析内存问题。

Java GC简介

开源力量 Open Source Force

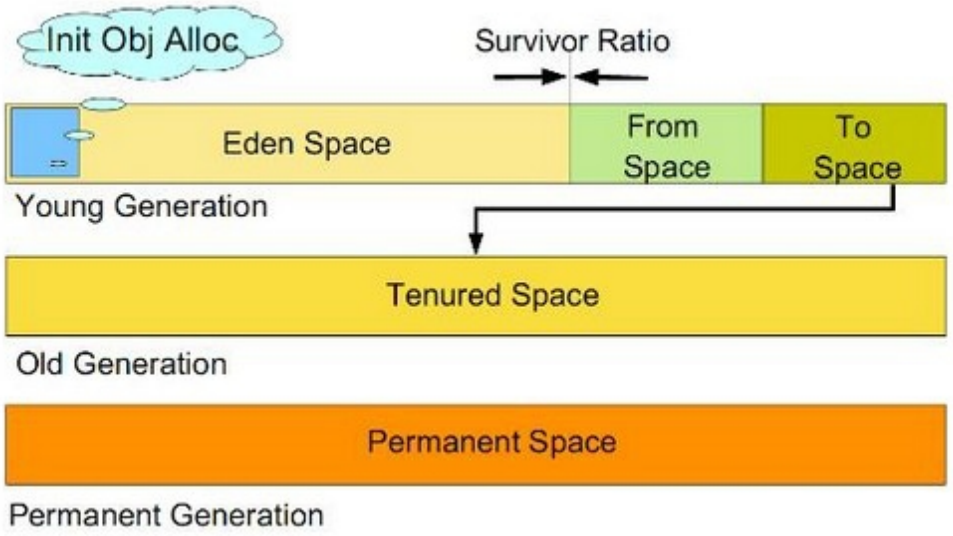
- 所有Java对象都是分配在Java堆上面的；
- Java上使用垃圾回收机制回收没有引用到的对象；
- Java虚拟机有专门的GC线程用来执行垃圾回收；
- 当GC线程从内存删除一个对象时，首先会调用对象的finalize函数，在这个函数里可以执行自定义的释放资源操作；
- Java程序自身无法强制启动GC，即使使用System.gc()和Runtime.gc()这样的函数，也只是递交一个GC请求给GC线程；
- 当无法在Java内存堆（Java heap）上创建对象时，Java虚拟机会抛出OutOfMemoryError。

- 对象没有被其他对象引用，是指从GC Root开始遍历，无法遍历到的对象，GC Root包括：
- Class - 系统里加载的类，这些类不会被卸载，类里的静态变量可能会引用其它Java对象。
- Thread - 正在运行的线程。
- 堆栈上的局部变量 - 堆栈上的函数还要运行，因此他们引用到的对象都是有用的。
- JNI参数和局部变量。
- 锁（Monitor） - 用于线程同步。

Java GC堆结构

•Java堆分成3个部分，或称作届（generation），分别是Young（New） generation, Tenured（Old） generation，Perm Area of heap。而New generation又细分成Eden space, Survivor 1和Survivor 2。新的对象总是先在Eden space上创建，经过minor GC后，剩余的会挪到Survivor 1，接着就是Survivor 2。当执行过Major GC后，对象会被挪到Old generation中。

•Perm area主要是用来保存Java类型以及函数的元数据信息，有的内在化（internalization）的字符串也会放在里面。



Java GC器

•虚拟机提供了好几个GC收集器，Java 5在Serial GC之外，还添加了几个GC收集器：

• Serial GC：只使用一个GC线程执行垃圾回收工作，在执行GC时，其他线程都会暂停工作，收集young generation里的垃圾对象。

• Parallel（Throughput）GC：使用多个GC线程并行执行垃圾回收工作，在执行GC时，其他线程都会暂停工作，收集young generation里的垃圾对象。

• Concurrent Mark Sweep（CMS） GC：用于收集old generation里的垃圾对象，跟其他线程并行工作，当需要标注某个线程里的垃圾对象时，会暂停线程一小会，其他时候可以与线程并行执行。

•在GC时，通常young generation里的GC，即minor GC很快，当old generation空间不够时，Java虚拟机首先会尝试CMS GC并行收集，如果这样空间还不能快速回收时，那Java虚拟机会暂停所有线程执行GC，这个时候称为Full GC。一般来说，Full GC的执行效率要比minor GC慢很多，程序优化的目标也是尽量减少Full GC的执行次数。

Java 内存堆

开源力量 Open Source Force

•Java虚拟机在启动时，会从操作系统申请一大块内存，后续Java程序运行时，所有对象都在这个内存块里分配，这个内存块叫Java Heap。

•Java内存堆在Java程序启动后无法更改，只能通过修改Java程序的-Xms、 - Xmx、-Xmn等启动参数改变设置。当Java堆内存用光，而且GC也无法收集更多的内存时，抛出OutOfMemoryError。

•可以使用Jconsole、Runtime.maxMemory()、Runtime.totalMemory()以及Runtime.freeMemory()等函数查询Java程序的内存堆设置。

JDK内存工具使用介绍 - JMap

开源力量 Open Source Force

•Jmap可以用来打印一个运行中的Java程序或者Java内存文件的内存使用率统计。Jmap还可以与jsadebugd后台出现同时使用，查询统计远程机器上的Java程序的内存使用率情况。

•使用-heap选项来收集下列Java内存使用率情况：

•1、垃圾回收算法相关的信息。

•2、内存堆（Heap）的设置。

•3、内存使用率情况。

•在ubuntu上需要执行命令：

•echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope

•使用-histo选项获取按类型统计的内存使用率情况。

•使用-permstat选项来统计永久性内存的使用率情况，永久性内存是Java虚拟机用来保存类型、函数等对象和内在化（internalization）字符串的区域。一般在调试JSP等经常生成和加载大量类型的Java程序很有用。

1、jmap -heap的输出解释
2、jmap -histo的输出解释
3、jmap -permstat的输出解释

示例

JDK内存工具使用介绍 - JHat

开源力量 Open Source Force

jHat可以分析一个二进制的hprof日志文件，提供一个web界面用来查询和分析java的内存使用情况，其提供了一种类似SQL的OQL查询语句使得这个查询变得很简单。

可以用它来找出一些垃圾对象的意外引用 - 从而导致垃圾无法回收，也就引发了内存泄露。

OQL的语法是：

- Select 选择用的Javascript表达式
- from [instanceof] 类名 对象名
- where Javascript语法的布尔表达式。

知乎软件

1、使用jmap和jhat等工具分析Java程序内存泄露问题。

示例

开源力量
Open Source Force

THANK YOU

知平软件