HPROF 代理

内容

- 1. HPROF 代理
 - 1. 内容
 - 2. 概述
 - 3. 启动
 - 4. <u>堆分配配置文件(堆=站点)</u>
 - 5. 堆转储(堆=转储)
 - 6. CPU 使用情况采样配置文件 (cpu=samples)
 - 7. CPU 使用时间配置文件 (cpu=times)
 - 8. <u>二进制转储格式 (format=b)</u>
 - 1. <u>套接字连接和通</u>信
 - 1. 数组的处理
 - 9. <u>源代码</u>

概述

本文档描述了 Java 开发工具包 (JDK) 中提供的 JVM TI 代理 HPROF。它旨在作为 JVM TI 的演示代码,并作为过去版本中提供的旧 HPROF JVMPI 代理的功能替代品。

JDK 的先前 1.4 和更早版本包含一个基于实验性 JVMPI 构建的 HPROF 代理。较新的 JVM TI 取代了 JVMDI 和 JVMPI。

注意:此 HPROF 实现与旧的基于 JVMPI 的 HPROF 之间的 差异 在本文档中以红色斜体标记。

启动

HPROF 是随 JDK 一起提供的简单分析器代理。它是一个动态链接库,与 JVM TI 交互,并将分析信息以 ascii 或二进制格式写入文件或套接字。此信息可以由探查器前端工具进一步处理。

它能够显示 CPU 使用情况、堆分配统计信息和监控争用配置文件。此外,它还可以报告 Java 虚拟机中所有监视器和线程的完整堆转储和状态。

HPROF 可以通过以下方式调用:

java - agentlib: hprof ToBeProfiledClass

根据所请求的分析类型,HPROF 指示虚拟机向其发送相关的 JVM TI 事件并将事件数据处理为分析信息。例如,以下命令获取堆分配配置文件:

java -agentlib:hprof=heap=站点 ToBeProfiledClass

以下是可以传递给 hprof 的完整选项列表:

2022/6/13 12:46

HPROF 代理 java -agentlib:hprof=help HPROF: 堆和 CPU 分析代理 (JVMTI 演示代码) hprof 用法: java -agentlib:hprof=[help]|[<option>=<value>, ...] 选项名称和值说明默认 _____ heap=dump|sites|all 堆分析所有 cpu=samples|times|old CPU 使用率 off monitor=y|n monitor contention n format=a|b text(txt) or binary output a file=<file> write data to file java.hprof[{.txt}] net =<host>:<port> 通过套接字发送数据 off depth=<size> 堆栈跟踪深度 4 间隔=<ms> 以毫秒为单位的采样间隔 10 截止=<值> 输出截止点 0.0001 lineno=y|n 跟踪中的行号?y thread=y|n thread in traces? n doe=y|n 退出时转储? y msa=y|n Solaris 微状态记帐 n force=y|n 强制输出到 <file> y verbose=y|n 打印有关转储的消息 y 过时选项 gc okay=y n 示例 - 每 20 毫秒获取样本 cpu 信息, 堆栈深度为 3: java -agentlib:hprof=cpu=samples,interval=20,depth=3 classname - 根据分配站点获取堆使用信息: java -agentlib:hprof=heap=sites 类名 注意 - 选项 format=b 不能与 monitor=y 一起使用。 - 选项 format=b 不能与 cpu=old|times 一起使用。 - 仍然可以使用 -Xrunhprof 接口, 例如 java -Xrunhprof:[help]|[<option>=<value>, ...]

java -agentlib:hprof=[help] 完全相同|[<option>=<value>, ...]

警告

- 这是 JVMTI 接口和 BCI 使用的演示代码, 它不是官方产品或 JDK 的正式部分。
- - Xrunhprof 接口将在未来版本中删除。
- 选项 format=b 被认为是实验性的,此格式可能会 在将来的版本中更改。

默认情况下,堆分析信息(站点和转储)被写入 java.hprof.txt (ascii)。monitor=y|n 选项已被证明 是有问题的,可以用更有用的东西代替。

在大多数情况下,输出将包含跟踪、线程、对象等的 ID。每种类型的 ID 通常以与其他 ID 不同的数 字开头, 例如跟踪可能以 300000 开头。

注意:不再支持 gc_okay 选项。

堆分配配置文件(堆=站点)

javac以下是通过在一组输入文件上 运行 Java 编译器()生成的堆分配配置文件。此处仅显示分析器输出的一部分。

堆配置文件中的一个关键信息是程序各个部分中发生的分配量。上面的SITES记录告诉我们,总空间的 44.13% 分配给了 java.util.zip.ZipEntry 对象。请注意,实时数据量只是给定站点发生的总分配量的一小部分;其余的已被垃圾收集。

将分配站点与源代码相关联的一个好方法是记录导致堆分配的动态堆栈跟踪。以下是分析器输出的另一部分,它说明了上面显示的输出中的四个分配站点引用的堆栈跟踪。

```
TRACE 301926:
        java.util.zip.ZipEntry.<init>(ZipEntry.java:101)
        java.util.zip.ZipFile+3.nextElement(ZipFile.java:417)
        com.sun.tools.javac.jvm.ClassReader .openArchive(ClassReader.java:1374)
       com.sun.tools.javac.jvm.ClassReader.list(ClassReader.java:1631)
TRACE 301927:
       com.sun.tools.javac.util.List.<init>(List. java:42)
        com.sun.tools.javac.util.List.<init>(List.java:50)
       com.sun.tools.javac.util.ListBuffer.append(ListBuffer.java:94)
       com.sun, tools.javac.jvm.ClassReader.openArchive(ClassReader.java:1374)
TRACE 300996:
       com.sun.tools.javac.util.Name$Table.<init>(Name.java:379)
       com.sun.tools.javac.util.Name$Table.<init>(Name.java:481)
       com.sun.tools.javac.util.Name$Table.make(Name.java:332)
       com.sun. tools.javac.util.Name$Table.instance(Name.java:349)
TRACE 300995:
       com.sun.tools.javac.util.Name$Table.<init>(Name.java:378)
       com.sun.tools .javac.util.Name$Table.<init>(Name.java:481)
       com.sun.tools.javac.util.Name$Table.make(Name.java:332)
       com.sun.tools.javac.util .Name$Table.instance(Name.java:349)
```

堆栈跟踪中的每一帧都包含类名、方法名、源文件名和行号。用户可以设置 HPROF 代理收集的最大帧数。默认限制为 4。堆栈跟踪不仅揭示了哪些方法执行了堆分配,而且还揭示了哪些方法最终负责进行导致内存分配的调用。

堆转储(堆=转储)

可以通过以下方式获得堆中当前活动对象的完整转储:

使用的命令: javac -J-agentlib:hprof=heap=dump Hello.java

这是一个非常大的输出文件,但可以在任何编辑器中查看和搜索。

CPU 使用情况采样配置文件 (cpu=samples)

HPROF 可以通过采样线程来收集 CPU 使用信息。javac以下是从编译器 运行收集的部分输出。

```
使用的命令: javac -J-agentlib:hprof=cpu=samples Hello.java

CPU SAMPLES BEGIN (total = 462) Fri Feb 6 13:33:07 2004

rank self accum count trace method

1 49.57% 49.57% 229 300187 java.util .zip.ZipFile.getNextEntry
2 6.93% 56.49% 32 300190 java.util.zip.ZipEntry.initFields
3 4.76% 61.26% 22 300122 java.lang.ClassLoader.defineClass2
4 2.81% 64.07% 13 300188 java.util.zip.ZipFile .freeEntry
5 1.95% 66.02% 9 300129 java.util.Vector.addElement
6 1.73% 67.75% 8 300124 java.util.zip.ZipFile.getEntry
7 1.52% 69.26% 7 300125 java.lang.ClassLoader.findBootstrapClass
8 0.87% 70.13 % 4 300172 com.sun.tools.javac.main.JavaCompiler.<init>9 0.65% 70.78% 3 300030 java.util.zip.ZipFile.open
10 0.65% 71.43% 3 300175 com.sun.tools.javac.main.JavaCompiler.<init>...

CPU 采样结束
```

HPROF 代理定期对所有正在运行的线程的堆栈进行采样,以记录最频繁活动的堆栈跟踪。上面的 count 字段表示发现特定堆栈跟踪处于活动状态的次数。这些堆栈跟踪对应于应用程序中的 CPU 使用热点。

CPU 使用时间配置文件 (cpu=times)

HPROF 可以通过将代码注入每个方法进入和退出来收集 CPU 使用信息,跟踪准确的方法调用计数和每个方法花费的时间。这使用字节码注入 (BCI) 并且运行速度比 cpu=samples 慢得多。javac以下是从编译器 运行收集的部分输出。

```
使用的命令: javac -J-agentlib:hprof=cpu=times Hello.java

CPU TIME (ms) BEGIN (total = 2082665289) Fri Feb 6 13:43:42 2004
rank self accum count trace method

1 3.70% 3.70% 1 311243 com.sun.tools.javac.Main.compile
2 3.64% 7.34% 1 311242 com.sun.tools.javac.main.Main.compile
3 3.64% 10.97% 1 311241 com.sun.tools.javac.main.Main.编译
4 3.11% 14.08% 1 311173 com.sun.tools.javac.main.JavaCompiler.compile
5 2.54% 16.62% 8 306183 com.sun.tools.javac.jvm.ClassReader.listAll
6 2.53% 19.15% 36 306182 com. sun.tools.javac.jvm.ClassReader.list
7 2.03% 21.18% 1 307195 com.sun.tools.javac.comp.Enter.main
8 2.03% 23.21% 1 307194 com.sun.tools.javac.comp.Enter.complete
9 1.68% 24.90% 1 306392 com.sun.tools.javac.comp.Enter.classEnter
10 1.68% 26.58% 1 306388 com.sun.tools.javac.comp.Enter.classEnter
```

这里的计数表示输入此方法的次数的真实计数,百分比表示在这些方法中花费的线程 CPU 时间的度量。

二进制转储格式 (format=b)

二进制输出中的基本字段是 u1(1 字节)、u2(2 字节)、u4(4 字节)和 u8(8 字节)。此实现中的 ID 是 u4,但是 ID 的大小实际上由标头中的"标识符大小"字段确定。

警告:这种格式仍然被认为是高度实验性的,但是,所有的尝试都是为了匹配过去 HPROF 实现的格式。

二进制输出以以下信息开头:

	表示格式名称和版本的初始 NULL 终止的一系列字节,在此实现中和历史上,字符串"JAVA PROFILE 1.0.1"(18 个 u1 字节)后跟一个 NULL 字节。如果使用标记"HEAP DUMP SEGMENT",则此字符串将为"JAVA PROFILE 1.0.2"。
11 1	标识符的大小。标识符用于表示 UTF8 字符串、对象、堆栈跟踪等。它们可以具有与主机指 针或 sizeof(void*) 相同的大小,但不是必须的。
u4	自格林威治标准时间 0:00,70 年 1 月 1 日以来毫秒数的高字
u4	自格林威治标准时间 0:00,70 年 1 月 1 日以来毫秒数的低字

后跟一个 se如下所示的记录数量:

u1	TAG:表示记录的类型						
u4	TIME: 自标头中的时间戳以来的微秒数						
u4	LENGTH:跟随这个 u4 字段并且属于这个记录的字节数						
[u1]*	BODY:与上述 u4 字段中指定的字节数一样多						

支持以下标签:

		I					
UTF8 格式的字	0x01	ID	此字符串的 ID				
1寸中 	守串		字符串的 UTF8 字符(不以 NULL 结尾)				
负载等级	0x02	u4	类序列号(总是 > 0)				
		ID	类对象 ID				
		u4	堆栈跟踪序列号				
		ID	类名字符串ID				
卸载类	0x03	u4	班级序号				
		<u> </u>					
堆栈框架	0x04	ID 堆栈帧 ID					
ID 方法名称字符串 ID 方法签名字符串 ID			方法名称字符串 ID				
			方法签名字符串 ID				
ID		ID	源文件名 字符串 ID				

		u4		班级序号			
		u4		> 0		电话号码	
				О		没有可用的线路信息	
				_1		位置不明	
				-2			
				-3		本机方法(<u>未实现</u>)	
	0.05						
堆栈跟踪 	0x05	u4				堆栈跟踪序列号 	
		u4				线程序列号	
		u4				帧数	
		[ID]*			一系列堆栈帧 ID	
分配站点 	0x06	u2	位掩	码标			
			0x1		增	量与完整	
			0x2	2	按	分配与行排序	
			0x4	1	是	否强制GC(未实现)	
			+151		\	N/L\	
			は 截止比(浮点数)				
			4 总活动字节 				
		╟──					
				的总:			
				的总:			
		u4 	li r -	的网		<u>里·</u> ·符:0 表示不是数组,非零表示此类型的数组(参见	
				基本			
			u4	班级	序号		
			u4	堆栈	跟踪	序列号	
					数		
			u4	活动	实例	数	
			u4	分配	的字	节数	
			u4	例数			
 堆摘要 	0x07	u4	4 总活动字节				

		4	ਨਾ⊩ਨਾ	ᄺᆚᆇ	147			
		u4 实时实						
		u8	分配的总字节数					
		u8	分配的	总实例				
开始线程	始线程 0x0A u4 线程序列号			IJ 号				
		ID ½	线程对象	え ID				
		u4 ±	隹桟跟踪	宗列号				
		ID ½	线程名称	京字符串 ID				
ID 线程组名称 ID			S称 ID					
		ID ½	线程父组	且名 ID				
结束线程	0x0B	u4	线和	程序列号				
			·					
堆转储 <i>或</i>	0x0C 或	包含任意数量示顺序):	的子标	签,每个子标	示签都以一个 u1 字段开头(这里没有暗			
堆转储段	0x1C	根未知	0xFF	ID	对象 ID			
		根 JNI 全球	0x01	ID 对	象 ID			
				ID JN	全局参考 ID			
		根 JNI 本地	0x02	ID 对象	D			
				u4 线程序				
				u4 堆栈足				
		根 JAVA 框	0x03	ID 対象	D			
		架		u4 线程序				
		根本机堆栈	0x04	ID	对象 ID			
				u4	线程序列号			
		 根粘性类	0x05	ID	对象 ID			
					7.130			
		根螺纹块	0x06	ID	对象 ID			

		u4
		<u> </u>
使用的根监 视器	0x07	ID 对象 ID
根线程对象	0x08	ID 线程对象 ID u4 线程序列号 u4 堆栈跟踪序列号
班级转储	0x20	D 类对象 D U4 堆栈跟踪序列号 D 超类对象 D D 数据数对象 D D 签名者对象 D D 预订的 D 预订的 D 预订的 D 预订的 U2 常量池索引 U1 条目类型: (见基本类型) 价 条目的类型) 价 条目的类型) 例 条目的类型) 价 条目值 (u1、u2、u4 或 u8,取决于值 字段类型: (见基本类型) 价 条目值 (u1、u2、u4 或 u8,取决于字值 段类型) 价 条目值 (u1、u2、u4 或 u8,取决于字值 段类型) (见基本类型) (见其本类型) (见其本文型) (见
实例转储	0x21	

				ID .	対象 ID
				u4	堆栈跟踪序列号
				ID	类对象 ID
				u4	后面的字节数
				[价值]*	实例字段值(此类,后跟超类等)
		对象数组转	0x22	T T	
		储 		ID	数组对象 ID
				u4	世栈跟踪序列号 ————————————————————————————————————
				u4	
				ID	数组类对象 ID
				[ID]*	元素
		F-10 25 40 ++			
		原始数组转 储	0x23	ID	数组对象 ID
				u4	堆栈跟踪序列号
				u4	元素数量
				u1	元素类型(见 <u>基本类型</u>)
				[u1]*	元素(压缩数组)
 堆转储端	Ovac	/女 LL 亥 5川+4	: 左	LIEAD F	
	UXZC	DUMP。	: 구マ 旧 F又	, NEAF L	JOINIL SEGINICIALS 的中軟会] ― HEAP
CPU 样本	0x0D	u4 样	本总数		
		u4 跟	踪的数:	量:	
		u	4	样本数	
		u	4	堆栈跟跟	宗序列号
控制设置	0x0E	u4 位	掩码标	志:	
			x1	分配	跟踪开/关
			x2	сри	采样开/关
			15		
		u2 堆	栈跟踪》 ————	深度 —————	

基本型

2	目的				
4	布尔值				
5	字符				
6	漂浮				
7	双倍的				
8	字节				
9	短的				
10	整数				
11	长				

数组的处理

转储中每个数组的类型类型都会有一个"LOAD CLASS"标签。在 LOAD CLASS 记录中,类名字符串 ID 将引用具有数组类型的人类可读名称的字符串,该数组类型的格式设置为 Java 源代码中的类型名称。因此,类型 char[] 的 LOAD CLASS 记录将为"char[]",short[][][] 为"short[][]",MyType[] 为"MyType[]"。

套接字连接和通信

警告:这个命令格式仍然被认为是高度实验性的,然而,所有的尝试都是为了匹配过去 HPROF 实现的格式。

命令可以通过套接字连接发送到 HPROF, 接受的命令标签是:

FORCE GC(<u>未实施</u>)	0x01
转储堆	0x02
分配站点	0x03
堆摘要	0x04
退出虚拟机	0x05
转储跟踪	0x06
CPU 样本	0x07
控制	0x08
EOF(用于终止套接字连接)	0xFF

命令采用以下形式:

u1	命令标签							
u4	序列号							
u4	后面的字节数							
[u1]*	分配站点	u2 u4	率(0.0 和 1.0 之间的浮点数)					
	CPU 样本	u2 u4						
	控制	u2	子选项: 0x1 0x2 0x3 0x4 0x5 0x6	打开分配跟踪 打开 CPU 采样: ID 线程对象 ID (所有线程为 0) 关闭 CPU 采样: ID 线程对象 ID (所有线程为 0) 清除 CPU 采样 设置最大堆栈深度: u2 新的最大堆栈深度				

源代码

HPROF 的源代码可在 demo/jvmti/hprof 目录中的 JDK 下载中找到。

*在本网站上使用的术语"Java 虚拟机"或"JVM"是指 Java 平台的虚拟机。

最后修改: 2005