淘宝网
Taobao.com

Java Web应用诊断利器
——Serviceability Agent

阿里巴巴核心系统研发部专用计算组
梅路峣/云达

关于我

- 2011.7 开始在淘宝实习，从事JVM相关工作
- 2012.3 硕士毕业（北邮计算机专业）
- 对编程语言的实现很感兴趣，目前关注重点在JVM的实现
- 工作职责之一是解决阿里线上Java应用的各种故障

- yunda.mly@taobao.com
- http://weibo.com/u/1063244843

主要内容

- 什么是Serviceability Agent (SA)
- SA的工作原理
- 如何利用SA进行Java应用故障诊断
- 总结

主要内容

- 什么是Serviceability Agent (SA)
- SA的工作原理
- 如何利用SA进行Java应用故障诊断
- 总结

什么是Serviceability Agent (SA)

- **服务性（serviceability）**是指技术服务人员安装、配置及监控电脑产品，提供硬件或软件服务的能力，也包括在出现例外情形或故障时排除问题，使产品可以正常运作的相关能力，包括针对故障除错或隔离故障，进行根本原因分析等。（节选自维基百科）
- Oralce JVM (HotSpot)中的SA：The HotSpot™ Serviceability Agent (SA) is a set of APIs for the Java™ programming language which model the state of Java HotSpot Virtual Machine.

什么是Serviceability Agent (SA)

- 获取Java进程或Java core文件暴露出的原始二进制信息
- 根据这些二进制信息提取出JVM内部数据结构
- 根据JVM内部数据结构提取出Java对象
- 运行在一个独立的进程
- 主要由Java实现，小部分native代码
- SA的Java代码是HotSpot的C++代码的一个镜像
- SA组件是HotSpot标准组成的一部分

主要内容

- 什么是Serviceability Agent (SA)
- SA的工作原理
- 如何利用SA进行Java应用故障诊断
- 总结

SA的工作原理

- out-of-process
  - 和目标进程是两个独立的进程
  - 通过进程间通信实现调试
  - 不会影响目标进程
  - 依赖于操作系统提供的调试API
- 建立在一系列的debug原语上
  - 连接到目标进程/core
  - 查询目标进程/core的symbol
  - 读取目标进程/core的内存

SA的工作原理

- 在linux上和gdb类似
  - 使用ptrace系统调用
  - PTRACE_ATTACH来进行连接目标进程
  - PTRACE_PEEKDATA从目标进程读取数据
  - PTRACE_DETACH来断开连接

主要内容

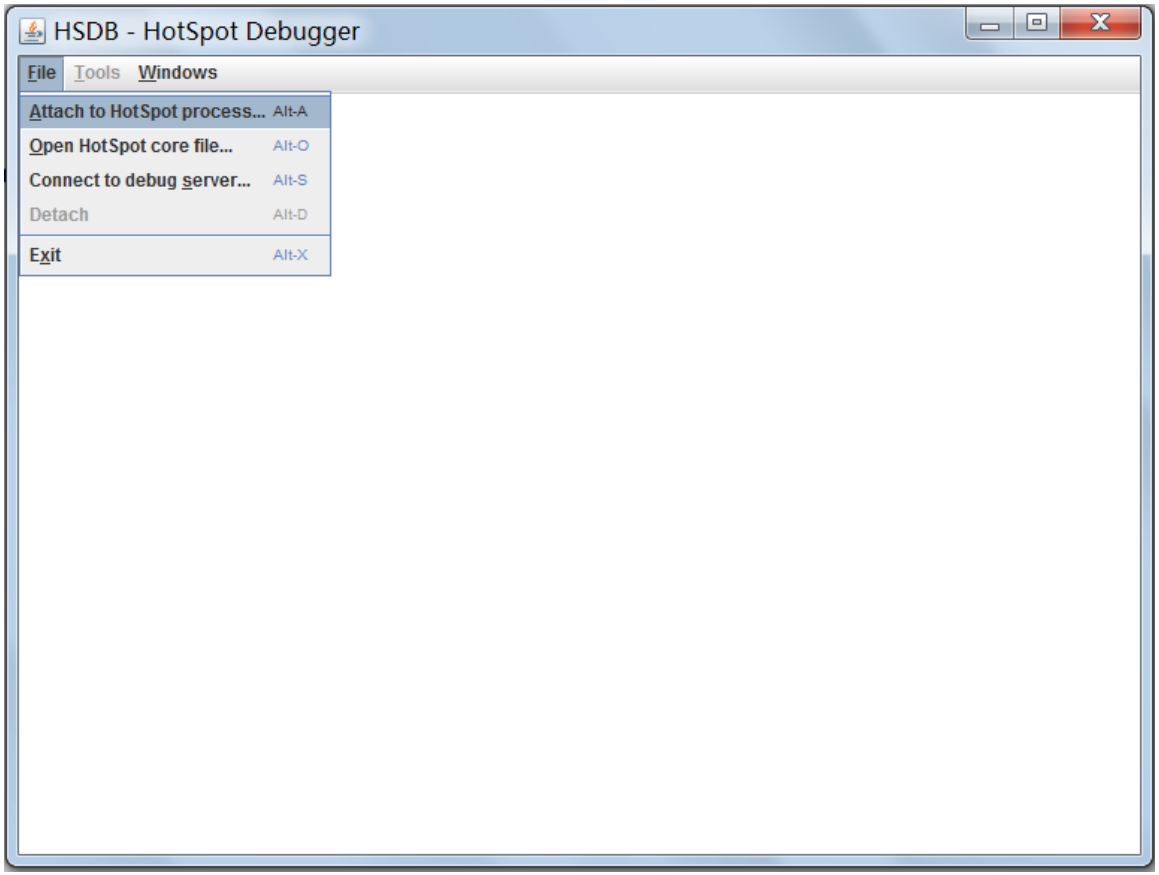
- 什么是Serviceability Agent (SA)
- SA的工作原理
- 如何利用SA进行Java应用故障诊断
- 总结

使用SA进行问题诊断

- 利用已有工具
  - jstack(-F, -m), jmap某些选项的功能是由SA来实现的
  - 图形工具HSDB, VisualVM的SA Plugin
  - 命令行工具CLHSDB

图形工具HSDB

- HotSpot Debugger
  - 启动方式：java -classpath .:$JAVA_HOME/lib/sa-jdi.jar sun.jvm.hotspot.HSDB
  - Windows JDK 6u32后版本有支持



命令行工具CLHSDB

- Command line HSDB
  - 启动方式：java -classpath .:$JAVA_HOME/lib/sa-jdi.jar sun.jvm.hotspot.CLHSDB
  - 有一系列的操作命令，通过help进行查看
  - 可以直接attach到core文件，方便解决JVM crash
  - 有javascript支持

命令行工具CLHSDB

- CLHSDB的几个常用命令：
  - help：查看所有命令的基本使用方式
  - attach：连接到目标进程或core
  - universe：查看Java heap的情况
  - inspect：查看某个地址对应的数据结构的内容
  - scanoops：扫描某个地址段的Java对象
  - jseval：执行javascript脚本

命令行工具CLHSDB

- CLHSDB对javascript的支持
  - 输出整个Perm gen的内容：
    jseval "sa.objHeap.iteratePerm(new sapkg.oops.HeapPrinter(java.lang.System.out))"
  - 输出整个堆里所有String的内容：
    jseval "io = java.io; sa.objHeap.iterateObjectsOfKlass(new sapkg.oops.HeapPrinter(new io.PrintStream(new io.FileOutputStream('perm.log'))),sa.systemDictionary.find('java/lang/String',null,null))"
  - 一个更复杂的案例

命令行工具CLHSDB

- 使用CLHSDB来实际解决一个JVM crash

使用SA进行问题诊断

- 使用SA API自己编写程序

asm
c1
ci
classfile
code
compiler
debugger
gc_implementation
gc_interface
interpreter
jdi
memory
oops
opto
prims
runtime
tools
types
ui
utilities

adlc
asm
c1
ci
classfile
code
compiler
gc_implementation
gc_interface
interpreter
libadt
memory
oops
opto
precompiled
prims
runtime
services
shark
trace
utilities

使用SA API编写程序

- 使用$JAVA_HOME/lib/sa-jdi.jar
- 继承*sun.jvm.hotspot.tools.Tool*并实现一个 *run()* 方法，在该方法内使用SA的API访问JVM即可
- 需要对JVM的内部结构有一定的了解

使用SA API编写程序

- 查看NIO direct memory的使用情况（完整代码）：

```
long reservedMemory =
    getStaticLongFieldValue("java.nio.Bits", "reservedMemory");

long directMemory =
    getStaticLongFieldValue("sun.misc.VM", "directMemory");
```

使用SA API编写程序

- TBJMap：
  - 对jmap的增强
  - 可以方便的输出Java堆中每一个分区的对象实例个数和大小的Histogram图

```
1: 1831111/70874072 1798386/68395648 char[]

2: 1871467/59886944 1840369/58891808 java.lang.String

3: 1224464/39182848 1224384/39180288 java.util.HashMap$Entry

4: 188390/26556456 188317/26547800 java.util.HashMap$Entry[]

5: 395298/25299072 395298/25299072 forest.dataobject.impl.DefaultPropertyValueDO

6: 1046319/25111656 1046181/25108344 java.lang.Long

7: 332206/23910336 329739/23821816 java.lang.Object[]
```

主要内容

- 什么是Serviceability Agent (SA)
- SA的工作原理
- 如何利用SA进行Java应用故障诊断
- 总结

总结

- SA适合解决相对比较底层的问题
- 建议从工具上手，熟练使用后尝试使用API编写程序
- SA会把目标进程hang住，线上环境谨慎使用

参考资料

- 官方介绍SA的论文：
  https://www.usenix.org/legacy/events/jvm01/full_papers/russell/russell_html/index.html
- OpenJDK关于SA的介绍：
  http://openjdk.java.net/groups/hotspot/docs/Serviceability.html#tsa
- Iteye高级语言虚拟机论坛（HLL VM）上关于SA的讨论：
  http://hllvm.group.iteye.com/group/topic/34278
- 借HSDB来探索HotSpot VM的运行时数据：
  http://rednaxelafx.iteye.com/blog/1847971
- TaobaoJVM主页：
  http://jvm.taobao.org/index.php

淘宝网
Taobao.com

Thank you

Q&A section