JDK Debugging Tools

Sang Shin
Michèle Garoche
www.javapassion.com
"Learn with Passion!"



jps

- List the JVMs that are currently running including embedded VMs
 - Associate a 'process number' with the running application

```
    JPS 27798 Jps 25301 Main
    JPS -S 28029 sun.tools.jps.Jps 25301 org.netbeans.Main
    JPS -I 25301 Main -Djdk.home=/opt/java/javase/jdk1.6 -Dnetbeans.dirs=/opt/java/tools/netbeans-6.0.1/nb6.0:/opt/java/tools/netbeans-6.0.1/ide8:/opt/java/tools/netbeans-6.0.1/java1:/opt/java/tools/netbeans-6.0.1/java1:/opt/java/tools/netbeans-
```

6.0.1/xml1:/opt/java/tools/netbeans-6.0.1/...

jinfo

- List configuration information from a running VM or a core file
 - Information includes VM properties and command line flags
- jinfo <pid from jps>

```
Java System Properties:
java.vendor = Sun Microsystems Inc.
netbeans.user = /home/shulk/.netbeans/6.0
sun.java.launcher = SUN STANDARD
sun.management.compiler = HotSpot Client Compiler
```

VM Flags:

-Djdk.home=/opt/java/javase/jdk1.6 -Dnetbeans.dirs=/opt/java/tools/netbeans-6.0.1/nb6.0:/opt/java/tools/netbeans-6.0.1/ide8:/opt/java/tools...

jstat

- List the statistics for a given VM
 - Class loading, GC on all spaces, hotspot compilation
 - > See Troubleshooting Guide
- Provide a sample interval and the number of samples to take



jstat -gcutil 25301 1000 10

```
S0
                        YGC
                               YGCT
                                      FGC
                                            FGCT
      0.00 81.32 42.10
                       99.56
                                     8.880
                                             36
                                                 17.363
                              1344
                                                         26.243
      0.00 81.32 42.10 99.56
                                             36
                                                 17.363
                                                         26.243
                              1344
                                     8.880
      0.00 81.32 42.10 99.56
                              1344
                                     8.880
                                             36
                                                 17.363
                                                         26.243
```

jstack

- Prints the stack traces of all the threads attached to a virtual machine
 - > Application thread, internal VM thread,
- Also performs deadlock detection with -I option
- Use -F to force stack if VM is hung
 - Solaris and Linux only

jstack - Sample Output

Locked ownable synchronizers:

```
"Java Source Worker Thread" prio=10 tid=0x08267800 nid=0x63a5 waiting on condition [0x4532c000..0x4532d040]

java.lang.Thread.State: TIMED_WAITING (parking)

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <0x54b8d090> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)

at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:198)

...

at java.util.concurrent.ThreadPoolExecuto$Worker.run(ThreadPoolExecutor.java:907)

at java.lang.Thread.run(Thread.java:619)
```

- <0x54b8cdd0> (a java.util.concurrent.locks.ReentrantLock\$NonfairSync)

jmap

- Prints memory related statistics
 - > Details the overall memory configuration
 - Details section on each of the space with capacity, free and used

jmap Sample Output (Solaris/Linux)

```
Mark Sweep Compact GC
Heap Configuration:
 MinHeapFreeRatio = 40
 MaxHeapFreeRatio = 70
 MaxHeapSize = 169869312 (162.0MB)
 NewSize = 1048576 (1.0MB)
 MaxNewSize = 4294901760 (4095.9375MB)
 OldSize = 4194304 (4.0MB)
 NewRatio = 12
 SurvivorRatio = 8
 PermSize = 33554432 (32.0MB)
                = 209715200 (200.0MB)
 Max PermSize  
Eden Space:
 capacity = 6881280 (6.5625MB)
 used = 4437312 (4.23175048828125MB)
       = 2443968 (2.33074951171875MB)
 64.48381696428571% used
```

jmap (Solaris/Linux)

```
cmlee
[cmlee@linux:~]{1004} jmap
Usage:
    jmap [option] <pid>
        (to connect to running process)
    jmap [option] Kexecutable Kcore>
        (to connect to a core file)
    .jmap [option] [server_id@]<remote server IP or hostname>
        (to connect to remote debug server)
where <option> is one of:
                         to print same info as Solaris pmap
    <none>
                         to print java heap summary
    -heap
    -histo[:live]
                         to print histogram of java object heap; if the "live"
                         suboption is specified, only count live objects
                         to print permanent generation statistics
    -permstat
    -finalizerinfo
                         to print information on objects awaiting finalization
    -dump:<dump-options> to dump java heap in hprof binary format
                         dump-options:
                                        dump only live objects; if not specified,
                           live
                                        all objects in the heap are dumped.
                                        binary format
                           format=b
                           file=<file> dump heap to <file>
                         Example: imap -dump:live.format=b.file=heap.bin <pid>
                         force. Use with -dump:<dump-options> <pid> or -histo
                         to force a heap dump or histogram when <pid> does not
                         respond. The "live" suboption is not supported
                         in this mode.
    -h I -help
                         to print this help message
    -J<flag>
                         to pass (flag) directly to the runtime system
cmlee@linux:~l{1005}
```

jmap (Windows)

```
Command Prompt
                                                                       C:\Documents and Settings\sang>jmap
Usage:
    jmap -histo <pid>
      (to connect to running process and print histogram of java object heap
    jmap -dump:<dump-options> <pid>
      (to connect to running process and dump java heap)
    dump-options:
      format=b
                  binary default
      file=<file> dump heap to <file>
    Example:
                   jmap -dump:format=b,file=heap.bin <pid>
C:\Documents and Settings\sang>
```

HPROF

- Uses the JVMTI to get information from a Java VM
- Data capture includes
 - > CPU usages, heap dump, thread states
- Start at commandline
 - > java -Xrunhprof:<options> MyJavaApp
 - > Includes format=b for jhat analysis
- Useful options
 - heap=all displays all heap info
 - > cpu=sample sample active thread
- Dump on application exit or CTRL-\

jhat

- Allows you to interactively work with a memory snapshot captured by jmap
 - > Use jmap -dump:format=b,file=heap_dump_file
- Use jhat to 'mount' heap file
 - > Hosted on a web server
 - Access through a standard browser
- Shows the following (standard query)
 - > All classes
 - Object on the heap
 - Instances
 - Objects reachable from root set

jhat Object Query Language

- Develop custom query with object query language
 - SQL like, uses JavaScript for expression in from and where clause
 - A set of built-in functions like heap, referrers, reachables, sizeof, etc.
- Use to answer questions like
 - > Find all String instances that are over 1K in size select s from j.l.Sring s where s.count >= 1024
 - > Find all URL instances that is referenced by 2 or more objects

select u from j.n.URL u where count(referrers(u)) > 2

Note: some queries takes a long time to run on large heaps

Visual Tool - jconsole

- Bundle with JDK
 - Solution > Graphical console that enables you monitor and manage Java applications
 - > API to create your own plugin to jconsole
- Provides information on
 - Memory usage and GC activities
 - > Threads, thread stack trace, locks
 - > Objects pending finalization
 - > Runtime information such as uptime, CPU time
 - > JVM information such as classpath, properties, command line arguments, etc.

Visual Tool – visualvm

- Open source project
 - > Based on NetBeans platform, uses the update center
 - Current plugin includes VisualGC, jconsole, thread dump analyzer, profiler
 - > Runs only on JDK 6 and later
 - http://visualvm.dev.java.net

When to Use Them over NetBeans Profiler?

Usecases

- Use jps to find which PID you want to dump
- Use jmap to generate a heap dump that can be opened with the NetBeans profiler
- Use OQL capability of jhat
 - Other features of jhat are now addressed by HeapWalker
- Use jstack when full attach of the profiler is too much of a hassle
 - > Useful in the field

Thank you!



Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"