Effective Java Heap Memory Analysis on Enterprise-Scale
SAP Memory Analyzer

Vedran Lerenc, SAP NetWeaver Java Server Technology
October 11th 2007
Memory 101

-XX:+HeapDumpOnOutOfMemoryError

Minor Garbage Collection

Full Garbage Collection

Memory utilization trend

Out Of Memory
1. Memory 101
   1.1. Java Memory Classification
   1.2. HPROF Binary Heap Dump Content
   1.3. How To Acquire A Heap Dump

2. Analysis Techniques
   2.1. Shallow Size vs. Retained Size
   2.2. Dominator Tree
   2.3. Aggregation Patterns
   2.4. 4-Step Approach To Finding Issues

3. Summary
   3.1. Critical Problems
   3.2. Lessons Learned
   3.3. Announcement
Java Memory Classification (Very Simplified View)
HPROF Binary Heap Dump Content

**All Objects**
Class, fields, primitive values and references

**All Classes**
ClassLoader, name, super class, static fields

**All ClassLoaders**
Defined classes

**Garbage Collection Roots**
Objects defined to be reachable by the JVM
A heap dump contains a **snapshot of objects that are alive** at one point in time.

A full GC is triggered before the heap dump is written.

A heap dump cannot **not** answer

- who and where objects have been created.
- which objects have been garbage collected.
How To Acquire A Heap Dump

Available in 1.4.2_12 and 5.0_7 and 6.0 upwards
-XX:+HeapDumpOnOutOfMemoryError

Alternatives to get it on demand
-XX:+HeapDumpOnCtrlBreak
jmap -dump:format=b,file=<filename.hprof> <pid>
JConsole
SAP Memory Analyzer / JVMMON / (MMC)

<table>
<thead>
<tr>
<th>Vendor / Release</th>
<th>VM Parameter</th>
<th>Sun Tools</th>
<th>SAP Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun, HP 1.4.2_12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2.5.0_07</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (Only Solaris and Linux)</td>
</tr>
<tr>
<td>1.6.0_06</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (Not working with Java AS)</td>
</tr>
<tr>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Any 1.5.0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (Only Solaris and Linux)</td>
</tr>
</tbody>
</table>
Disclaimer:

- Class, class loader and thread names have been changed to protect the identity of our customers.
- Any similarities to actual disasters that you have witnessed, either in real or virtual reality, are purely coincidental.
1. Memory 101
   1.1. Java Memory Classification
   1.2. HPROF Binary Heap Dump Content
   1.3. How To Acquire A Heap Dump

2. Analysis Techniques
   2.1. Shallow Size vs. Retained Size
   2.2. Dominator Tree
   2.3. Aggregation Patterns
   2.4. 4-Step Approach To Finding Issues

3. Summary
   3.1. Critical Problems
   3.2. Lessons Learned
   3.3. Announcement
Shallow Size vs. Retained Size

class X
{
    static
}

LinkedList
LinkedList$Entry
SomeEntry
String
char[]
Determine Retained Size via GC Simulation

1. Remove all references to object X
2. Mark all objects which are still reachable from the GC Roots
3. The unmarked objects constitute the retained set of object X
Demo

SAP Memory Analyzer

Disclaimer:
- Class, class loader and thread names have been changed to protect the identity of our customers.
- Any similarities to actual disasters that you have witnessed, either in real or virtual reality, are purely coincidental.
X dominates Y if all paths from the roots to Y run through X
Dominator Tree – Benefit #1
Retained Set and Size is just a Sub-Tree

Retained Set
→ Retained Size
Immediate Dominator Shows the Closest Responsible for Keeping an Analyzed Object Alive

Dominator Tree – Benefit #2
Quickly Find the Greedy Memory Pigs
Quick approximation of the Retained Size for a set of objects is done by summing up the distinct top dominators in the set.
Dominator Tree – Benefit #4
Biggest Distinct Object Graphs

Top-level Dominators
Show Biggest Distinct Objects

→ Easy Grouping by Class, ClassLoader
Disclaimer:

- Class, class loader and thread names have been changed to protect the identity of our customers.
- Any similarities to actual disasters that you have witnessed, either in real or virtual reality, are purely coincidental.
Aggregation Patterns

1. Look For A Property
2. Group Objects By It
3. Inspect Big Memory Chunks
4. Identify Systematic Activity

Examples:
- Arrays by Length
- Strings by Value

“It’s like looking for the butterfly swallowed by the hurricane it has caused!”
Multiple Paths shows common sections near the GC roots

Group Dominator Tree by class to find big groups of distinct object graphs

Group in Top Dominators
Group Along Shortest Paths
Group Referrers by Class

(list)
(header)
(entries)
(payload)
SAP Memory Analyzer

Disclaimer:
- Class, class loader and thread names have been changed to protect the identity of our customers.
- Any similarities to actual disasters that you have witnessed, either in real or virtual reality, are purely coincidental.
4-Step Approach To Finding Issues

- Get an Overview
  - Total heap size
  - Total number of objects, classes and class loaders
  - Class Histogram

- Find Big Chunks

- Inspect Content

- Identify Holders
1. Memory 101
   1.1. Java Memory Classification
   1.2. HPROF Binary Heap Dump Content
   1.3. How To Acquire A Heap Dump

2. Analysis Techniques
   2.1. Shallow Size vs. Retained Size
   2.2. Dominator Tree
   2.3. Aggregation Patterns
   2.4. 4-Step Approach To Finding Issues

3. Summary
   3.1. Critical Problems
   3.2. Lessons Learned
   3.3. Announcement
Critical Problems

Heap
- Inefficient data structures (e.g. badly used collections, keeping XML DOM,...)
- Caches (i.e. unknown entry size, different competing caches,...)

Perm
- Model/Proxy-driven class generation
- „Leaking“ loaders

In General
- Real size of objects not apparent to programmer
- No application/user quota
Lessons Learned

Inspecting heap dumps of enterprise scale needs new ideas and approaches
- Performance is crucial
- Functionality is even more important to tell apart noise from real problems

It’s not about leaks, it’s about footprint
- It’s relatively easy to find the leak
- But it’s really difficult to get down general memory consumption of a component

Analysis can be automated (Expert System)
- Instead of figuring it out yourself, let the tool do the job for you
  That’s our current focus!
Announcement

Today: SAP Memory Analyzer is Free Software.

Soon: SAP Memory Analyzer goes Open Source!

Eclipse Technology Project

"Handles large heap dumps easily"
"I am pretty impressed"
"Very nice UI"
"Performance is great"
"Quickly isolated problem"
"Very Cool"
"Quite impressed"
Thank you!

Vedran Lerenc, SAP NetWeaver Java Server Technology
October 11th 2007

WIKI & Download @ www.sdn.sap.com/irj/sdn/wiki?path=/display/Java/Java+Memory+Analysis
… or just Google for “SAP Memory Analyzer”