How to analyze an OutOfMemory issue

WebSphere Application Server Level2 Support team

WASSDK,103/WOTSDK,103

Yang Jie

Agenda

- Java memory: heap and native
- What can trigger OutOfMemory issue?
- How to gather data to analyze OutOfMemory issue?
- Analyze OutOfMemory issue

How to analyze OutOfMemory(OOM) issue

- Java memory: heap and native
- What can trigger OOM issue?
- How to gather data to analyze OOM issue?
- Analyze OOM issue

Type of OOM

- An OutOfMemoryError exception is generated when a particular resource is exhausted. The Java run time throws an OutOfMemoryError exception for these resources:
  - Java heap

    The memory area used for the allocation and management of Java objects created by the application.
  - Native heap

    The memory area used for the Java run time itself, for the allocation of resources to back some Java objects, and for the allocation of memory by JNI code.
-

How to analyze OOM issue

- Java memory: heap and native
- What can trigger OOM issue?
- How to gather data to analyze OOM issue?
- Analyze OOM issue

What can trigger OOM

- Java Heap OutOfMemory
  - Java Heap Exhaustion
  - Excessive GC activity
- Native Heap OutOfMemory
  - Native Heap Exhaustion
  - Native Heap Fragmentation
  - Classloader exhaustion
  - OS resource restriction
  - OS resource consumption
-

How to analyze OOM issue

- Java memory: heap and native
- What can trigger OOM issue?
- How to gather data to analyze OOM issue?
- Analyze OOM issue

Collecting data(IBM JDK)

"MustGather: Out of Memory errors on AIX, Linux, or Windows"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21138587

"MustGather: Native Memory Issues on AIX"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21405353

"MustGather: Native Memory Issues on Linux"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21138462

"MustGather: Native Out Of Memory on Windows"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21313578

Collecting data(non-IBM JDK)

"MustGather: Out of Memory exceptions on HP-UX"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21164686

"MustGather: Out of Memory errors on Solaris - Heap Leak"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21145349

"MustGather: Out of Memory errors on Solaris - Native Leak"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21104470

Collecting data

- Javadumps (javacore.txt files IBM-JDK only)
- Heapdumps (heapdump.phd files)
- -verbose:gc output
  - Rotate gc log configuration:

    -Xverbosegclog:<path to file><filename, X, Y>

    the verboseGC output is redirected to X files, each containing verboseGC output from Y GC cycles.

    For example: -Xverbosegclog:/opt/websphere/logs/gc#.log,100,1000
- Process size monitoring output
- WSAS server log directory

How to analyze OOM issue

- Java memory: heap and native
- What can trigger OOM issue?
- How to gather data to analyze OOM issue?
- Analyze OOM issue
  - First need to decide if heap or native OOM issue
  - Heap OOM
    - Leak? Fragmentation? Workload high?
  - Native OOM
    - Known possible native OOM issue and solution.

Decide if native or heap OOM

- Use the last garbage collection cycle before the OutOfMemoryError exception in the verbose:gc output to determine whether the OutOfMemoryError exception was caused by exhaustion of the Java heap or by excessive garbage collection.
  - Heap exhaustion

```
<gc type="global" id="13" totalid="74" intervalms="0.030">
    <compaction movecount="12133" movebytes="262762572" reason="compact to meet allocation" />
    <refs_cleared soft="0" weak="0" phantom="0" />
    <finalization objectsqueued="0" />
    <timesms mark="21.701" sweep="9.644" compact="590.093" total="621.504" />
    <nursery freebytes="785168" totalbytes="786432" percent="99" />
    <tenured freebytes="4435616" totalbytes="267386880" percent="1" />
  </gc>
  <tenured freebytes="4435616" totalbytes="267386880" percent="1" />
  <time totalms="745.990" />
</af>
```

  - Exccessive gc

```
<warning details="excessive gc activity detected" />
```

Decide if native or heap OOM (continued)

- In JAVACORE, if find following information, then that means a Native OOM issue.

"java/lang/OutOfMemoryError": "Failed to fork OS thread" received

"java/lang/OutOfMemoryError": "Failed to create a thread: retVal

Decide if native or heap OOM (continued)

- If no gc log, we can analyze javacore(IBM-JDK only)
  - Check maxheap, free heap, allocated heap from JAVACORE

2CIUSERARG -Xmx15360m

1STHEAPFREE Bytes of Heap Space Free: 1a96a8

1STHEAPALLOC Bytes of Heap Space Allocated: 10000000

  - Analyze GC History in JAVACORE
    - These terms indicate Java heap exhaustion:

J9AllocateIndexableObject() returning NULL!

J9AllocateObject() returning NULL!

    - Excessive GC can also cause OOM.

Forcing J9AllocateIndexableObject() to fail because of excessive GC

J9AllocateIndexableObject() to fail because of excessive GC

How to resolve a Java Heap OOM issue

- Java Heap used
  - Analyze heapdump

  Tools: HeapAnalyzer, Memory Analyzer

    - Memory Leak
    - Workload too high
- Java Heap fragmentation
  - Configure to make large object can be allocated in large object area – can only delay the issue.
  - Correct code not to request large object

"How to identify the Java stack of a thread making an allocation request larger than a certain size" (ALLOCATION_THRESHOLD)

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21236523

How to resolve a Java Heap OOM issue(continued)

- Heap fragmentation

If heap OOM caused by large object request, then analyze the calling thread. The calling thread can be determined by using ALLOCATION_THRESHOLD or -xdump (noted on page previous).

If last gc cycle which triggered OOM issue requests large size of memory, then analyze javacore to find the current thread as first step:

Current Thread Details

"main" (TID:0x000C4600, sys_thread_t:0x0003696C, state:R, native ID: 0x00000E48) prio=5

at java/lang/StringBuffer.ensureCapacityImpl(StringBuffer.java:397)

at java/lang/StringBuffer.append(StringBuffer.java:246)

If can't decide if native or heap OOM issue

- If there is no verbosegc log, it is not apparent if it is a native or java heap OOM issue, clear the logs, enable verbosegc and wait for the issue to re-occur

"Enabling verbose garbage collection (verbosegc) in WebSphere Application Server"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21114927

Tools to analyze gc log

- Use analyze tool to analyze verbosegc log
  - JAVA GC and Memory Visualizer

    http://publib.boulder.ibm.com/infocenter/javasdk/tools/topic/com.ibm.java.doc.igaa/_1vg00011e17d8ea-1163a087e6c-7ff6_1005.html

  - PMAT

Analyze native OOM issue

- If there is no problem in the GC cycle just before the javacore, then the issue is a native one.
- Check the size of the Java process. If it is close to the maximum process size for your platform the native heap has become exhausted.
- In log files, if find following information, then that means a Native OOM issue

JVMCI015:OutOfMemoryError, cannot create anymore threads due to memory or resource constraints

JVMDBG001: malloc failed to allocate n bytes

JVMDBG004: calloc failed to allocate an array of

JVMCL052: Cannot allocate memory in initializeHeap for heap segment

- If WSAS is 32 bit, please reference following table

Operating system	Additional options	Maximum process address space
AIX	-Xmx < 2.3 GB	2.75 GB
	2.3 GB <= -Xmx < 3 GB	3 GB
	-Xmx >= 3 GB	3.25 GB
Linux	None	3 GB
	Hugemem Kernel	4 GB
Windows	None	2 GB
	/3GB	3 GB

Analyze native OOM issue

- Known Native OOM issue and solution

"Troubleshooting native memory issues"

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21373312

  - Check the maximum heap size
  - Potential native memory use in WebSphere Application Server thread pools
  - WebContainer DirectByteBuffer use
  - AIO Native Transport

Thank you!