# JAVA□□□□□□□□□

□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□




## □□□□□□□□□□

### □□□□ Top 3□□

1. **Tomcat Java□□ (PID 31037)**□
   - □□□□□ 5.26GB (32.5%)
   - JVM□□□ `-Xms2048m -Xmx4096m` (□□□ 2-4GB)
   - □□ RSS□□□□□□□□□□□
     * □□□□□□ (□□ `-XX:MaxDirectMemorySize=1024m`□□ )
     * □□ JNI□□□□□□□□□

2. **MySQL (PID 1129)**□
   - □□□□□ 2.53GB (15.6%)
   - □□□□ (88460□□ )□□□□□
     ```bash
     mysqladmin variables | grep buffer_pool
     mysqladmin status | grep -E 'Threads|Memory'
     ```

3. **shun-sync□□ (PID 5522)**□
   - □□□□□ 1.35GB (8.3%)
   - JVM□□□ `-Xms1024m -Xmx1024m` (□□□□□ )

## 内存问题诊断

### 异常现象确认
1. **物理内存**:
   - 总内存: 16GB
   - 已用: 15.97GB (99.8%)
   - 空闲: 186MB (1.2%)
   - 缓存: 2.3GB (偏低)

2. **交换空间**:
   - 总量: 4GB
   - 使用量: 1.3MB (0.03%)
   - 缓存比例: 10.5% (正常)

3. **内存分布**:
   - Active内存: 12.4GB (其中匿名页 11.1GB)
   - Inactive内存: 2.8GB

## 根本原因分析

1. **Tomcat内存泄漏**:
   - RSS(5.26GB) > Xmx(4GB) + MaxDirectMemory(1GB)
   - 可能原因:
     * JNI库内存泄漏
     * 未受JVM管控的本地内存分配
     * 线程栈过多(`-Xss`设置过大)

2. **swappiness配置不当**:
   - 当前配置可能为0, 导致系统宁愿触发OOM也不使用swap
   - 验证命令:
     ```bash
     cat /proc/sys/vm/swappiness
     ```

3. **MySQL内存配置**:
   - 未优化的InnoDB缓冲池可能导致内存碎片

## 紧急处理方案

### 立即释放内存
```bash
# 清理页面缓存(临时方案)
sync; echo 1 > /proc/sys/vm/drop_caches

# 重启最耗内存的Tomcat(需安排维护窗口)
systemctl restart tomcat
```

### JVM□□□□
```bash
# □□  JDK□□
apt-get install -y openjdk-8-jdk

# □□  Tomcat□□□□
jcmd 31037 VM.native_memory summary
```

```
root@shuncom-01:~# jcmd 31037 VM.native_memory summary
31037:

Native Memory Tracking:

Total: reserved=6165979KB, committed=4780715KB
-                 Java Heap (reserved=4194304KB, committed=3899392KB)
                            (mmap: reserved=4194304KB, committed=3899392KB)

-                     Class (reserved=1280500KB, committed=260596KB)
                            (classes #37630)
                            (malloc=10740KB #71177)
                            (mmap: reserved=1269760KB, committed=249856KB)

-                    Thread (reserved=186976KB, committed=186976KB)
                            (thread #179)
                            (stack: reserved=182984KB, committed=182984KB)
                            (malloc=588KB #896)
                            (arena=3405KB #356)

-                      Code (reserved=273364KB, committed=203492KB)
                            (malloc=23764KB #19623)
                            (mmap: reserved=249600KB, committed=179728KB)

-                        GC (reserved=159054KB, committed=158478KB)
                            (malloc=5810KB #1350)
                            (mmap: reserved=153244KB, committed=152668KB)

-                  Compiler (reserved=615KB, committed=615KB)
                            (malloc=484KB #857)
                            (arena=131KB #3)

-                  Internal (reserved=19846KB, committed=19846KB)
                            (malloc=19814KB #46359)
                            (mmap: reserved=32KB, committed=32KB)

-                    Symbol (reserved=44112KB, committed=44112KB)
                            (malloc=40268KB #288206)
                            (arena=3844KB #1)

-    Native Memory Tracking (reserved=7077KB, committed=7077KB)
                            (malloc=300KB #4658)
                            (tracking overhead=6777KB)

-               Arena Chunk (reserved=131KB, committed=131KB)
                            (malloc=131KB)
```

jmap -histo:live 31037 | head -20

```
root@shuncom-01:~# jmap -histo:live 31037 | head -20

 num     #instances         #bytes  class name
----------------------------------------------
   1:        550669      379776624  [C
   2:         25173      171052792  [B
   3:        546122       13106928  java.lang.String
   4:        235518        9420720  java.util.LinkedHashMap$Entry
   5:        102432        8554088  [Ljava.lang.Object;
   6:        116858        8413776  org.apache.jasper.compiler.Node$TemplateText
   7:         94211        8290568  java.lang.reflect.Method
   8:         96381        7375384  [Ljava.util.HashMap$Node;
   9:        222013        7104416  java.util.concurrent.ConcurrentHashMap$Node
  10:        107867        6040552  java.util.LinkedHashMap
  11:        124835        4993400  org.apache.jasper.compiler.Mark
  12:         39744        4404984  java.lang.Class
  13:         19634        4289136  [I
  14:        120823        3866336  java.util.HashMap$Node
  15:        224151        3586416  java.lang.Integer
  16:          9596        2736488  [Ljava.util.concurrent.ConcurrentHashMap$Node;
  17:        101173        2428152  java.util.ArrayList
```

```

## □□□□□□

1. **Tomcat□□  **□
   ```diff
   - -XX:MaxDirectMemorySize=1024m
   + -XX:MaxDirectMemorySize=512m
   + -XX:NativeMemoryTracking=summary
   + -XX:+AlwaysPreTouch
   ```

2. **MySQL□□  **□
   ```ini
   # /etc/mysql/my.cnf
   [mysqld]
   innodb_buffer_pool_size = 1G
   innodb_buffer_pool_instances = 4
   ```

3. **□□□□□□      **□
   ```bash
   # □□  OOM killer□□
   echo 'vm.panic_on_oom=2' >> /etc/sysctl.conf
   echo 'vm.overcommit_memory=2' >> /etc/sysctl.conf
   sysctl -p
   ```

4. **□□□□    **□
   ```bash
   # □□□□□□
   apt-get install -y sysstat valgrind
```
```

sar -r 60 -o /var/log/sa/memlog &
```

## □□□□

□□□□□□□
- □□□□□ <90%
- Swap□□□ 1-5%
- □□□□ >20%
- □□□□ RSS ≈ Xmx + 30%

# □□ `jcmd` □ `jmap` □□□□□ Tomcat □□ (PID 31037)

# □□□□□□□□□□□□□□□□□□□□

# □□□□□□□

---

## **1. □□□□□□ **
### **1.1 Native Memory □□ **
- **□ Reserved**: 6,165,979 KB (~6.1GB)
- **□ Committed**: 4,780,715 KB (~4.78GB)

| □□□□ | Reserved (KB) | Committed (KB) | □□□ |
|---|---|---|---|
| Java Heap | 4,194,304 | 3,899,392 | □□ Xmx □□ (4GB) |
| Class Metadata | 1,280,500 | 260,596 | **MetaSpace □□□ ** |
| Thread Stacks | 186,976 | 186,976 | **179 □□□□□□□□□ ** |
| Code Cache | 273,364 | 203,492 | **JIT □□□□□□□□ ** |
| GC □□ | 159,054 | 158,478 | G1GC □□□□ |
| Symbol Storage | 44,112 | 44,112 | □□□□□□□□ |

### **1.2 Heap □□□□ (`jmap -histo`)**
- **Top □□□□□□ **□
  - `char[]` (`[C`): **379MB**
  - `byte[]` (`[B`): **171MB**
  - `String`: **13MB**
  - `LinkedHashMap$Entry`: **9.4MB**
```

  - `Object[]` (`[Ljava.lang.Object;`): **8.5MB**

**问题点 **:
- **大量 `char[]` 和 `byte[]` 占用 **, 可能是:
  - **未优化的字符串处理 ** (如日志、 JSON 解析)
  - **缓存未限制大小 ** (如 HTTP Session 缓存)
  - **内存泄漏 ** (如静态集合未清理)

---

## **2. 问题诊断 **
### **2.1 MetaSpace 占用高 **
- **Class Metadata 占用 260MB**, 加载了 **37,630 个类 **。
- **可能原因 **:
  - **动态类加载 ** (如 JSP 编译、反射生成类)
  - **未限制 MetaSpace** (默认无上限, 可能无限增长)

### **2.2 线程栈占用过大 **
- **179 个线程 **, 每个线程栈占用约 **1MB** (默认 `-Xss1m`)。
- **总栈内存 **: 179 × 1MB ≈ **179MB**。
- **可能原因 **:
  - **线程池配置过大 ** (如 Tomcat `maxThreads=200`)
  - **线程泄漏 ** (如未正确关闭线程池)

### **2.3 Direct Memory 泄漏 **
- `-XX:MaxDirectMemorySize=1024m` 设置了 **1GB 堆外内存 **。
- **NIO Buffer 或 Netty 等框架可能泄漏 Direct Memory**。

---

## **3. 优化方案 **
### **3.1 JVM 参数优化 **
```diff
# 修改 Tomcat 启动参数 ( catalina.sh 或 setenv.sh)
- -Xms2048m -Xmx4096m -XX:MaxDirectMemorySize=1024m
+ -Xms2048m -Xmx4096m
+ -XX:MaxDirectMemorySize=512m # 限制 Direct Memory
+ -XX:MaxMetaspaceSize=256m # 避免 MetaSpace 无限增长
+ -Xss256k # 减少线程栈大小
+ -XX:+UseG1GC # 改用 G1 垃圾回收器
+ -XX:NativeMemoryTracking=detail # 持续监控 Native Memory
```

### **3.2 检查线程泄漏 **
```bash
# 查看 Tomcat 线程数
ps -L -p 31037 | wc -l
```

```
# 查看线程栈
jstack 31037 > /tmp/jstack.log
grep "pool-" /tmp/jstack.log | wc -l
```

**优化建议**:
- 调整 `server.xml` 中的 `maxThreads`（如从 200 → 100）。
- 检查是否有自定义线程池未正确关闭。

### **3.3 检查 MetaSpace 泄漏**
```bash
# 查看加载的类数量
jcmd 31037 GC.class_stats | head -20

# 检查是否有动态生成的类
jcmd 31037 VM.class_hierarchy | grep Generated
```

**优化建议**:
- 限制 `-XX:MaxMetaspaceSize=256m`，防止无限增长。
- 检查是否有频繁的 JSP 热部署。

### **3.4 检查 Direct Memory 泄漏**
```bash
# 查看 NIO Buffer 分配
jcmd 31037 VM.native_memory detail | grep "Java Heap\|Direct"
```

**优化建议**:
- 如果使用 Netty，检查 `ByteBuf` 是否正确释放。
- 限制 `-XX:MaxDirectMemorySize=512m`。

---

## **4. 监控与验证**
### **4.1 监控内存变化**
```bash
# 每 5 秒打印内存使用
watch -n 5 'ps -p 31037 -o %mem,rss,comm && jcmd 31037 VM.native_memory summary'
```

### **4.2 触发 Full GC 观察内存回收**
```bash
# 手动触发 Full GC
jcmd 31037 GC.run
```

### **4.3 检查 OOM 风险**
```bash
```

# □□  GC □□□□□□  -Xloggc□
tail -f /usr/local/apache-tomcat-8.0.39/logs/gc.log
```

---

## **5. □□  **
- **□□□□  **□
  - **MetaSpace □□□  **□ 260MB□□□□□  `MaxMetaspaceSize`□
  - **□□□□□□  **□ 179MB□□□□□  `-Xss` □ `maxThreads`□
  - **Direct Memory □□□□  **□□□□□□□  NIO Buffer□
  - **Heap □ `char[]` □ `byte[]` □□□  **□□□□□□□□□□□□□□□□

- **□□□□□  **□
  - **RSS □□□□  1-2GB**□□  5.26GB → 3.5GB □□□□
  - **□□  OOM □□  **□□□□□□□□□

□□□  **□□□□□  Tomcat** □□□□  JVM □□□□□□□□□□□□

root@shuncom-01:/usr/local/apache-tomcat-8.0.39/bin# jcmd 31037 VM.native_memory detail | grep "Java Heap\|Direct"
-                 Java Heap (reserved=4194304KB, committed=3646976KB)
[0x00000006c0000000 - 0x00000007c0000000] reserved 4194304KB for Java Heap from
root@shuncom-01:/usr/local/apache-tomcat-8.0.39/bin# jcmd 31037 VM.class_hierarchy | grep Generated
java.io.IOException: Command failed in target VM
        at sun.tools.attach.LinuxVirtualMachine.execute(LinuxVirtualMachine.java:224)
        at sun.tools.attach.HotSpotVirtualMachine.executeCommand(HotSpotVirtualMachine.java:217)
        at sun.tools.attach.HotSpotVirtualMachine.executeJCmd(HotSpotVirtualMachine.java:199)
        at sun.tools.jcmd.JCmd.executeCommandForPid(JCmd.java:150)
        at sun.tools.jcmd.JCmd.main(JCmd.java:129)
root@shuncom-01:/usr/local/apache-tomcat-8.0.39/bin# jcmd 31037 GC.class_stats | head -20

31037:
GC.class_stats command requires -XX:+UnlockDiagnosticVMOptions


??????????????? Tomcat ?? (PID 31037) ? □□□□□□  ? □□□□  ?

---

## **1. □□□□  **
### **1.1 Java Heap □□□□  **
- **□□  **□ `-Xmx4096m` (4GB)
- **□□  Commit**□ 3.64GB□□□□□□□□  OOM □□□
- **□□  **□ Heap □□□□□□□□□□□
  - □□□□□□□  HTTP Session □□□
```

- 大对象未及时释放（如 `char[]` 占 379MB）

### **1.2 Direct Memory**
- **未检测到 Direct Buffer 泄漏 **（`grep "Direct"` 无输出）
- 但 `-XX:MaxDirectMemorySize=1024m` 仍可能被 NIO 或第三方库（如 Netty）使用，需保持限制。

### **1.3 MetaSpace**
- **无法直接获取类统计 **（需 `-XX:+UnlockDiagnosticVMOptions`）
- **间接证据 **：`Class Metadata` 占用 260MB，加载 37,630 个类，可能存在：
- 动态类加载（如 JSP 编译）
- 反射生成类（如 Spring AOP）

---

## **2. 最终优化方案 **
### **2.1 修改 Tomcat JVM 参数 **
编辑 `catalina.sh` 或 `setenv.sh`，替换原有参数：
` ` ` bash
JAVA_OPTS="
-Xms2048m -Xmx4096m
-XX:MaxMetaspaceSize=256m # 限制 MetaSpace 增长
-XX:MaxDirectMemorySize=512m # 限制堆外内存
-Xss256k # 减少线程栈大小
-XX:+UseG1GC # 改用 G1 垃圾回收器
-XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:/usr/local/apache-tomcat-8.0.39/logs/gc.log -XX:NativeMemoryTracking=detail # 持续监控 Native Memory
"
` ` `

### **2.2 检查线程泄漏 **
` ` ` bash
# 查看线程数
ps -L -p 31037 | wc -l

# 如果线程数 > 150，调整 Tomcat 的 maxThreads
sed -i 's/maxThreads="200"/maxThreads="100"/g' /usr/local/apache-tomcat-8.0.39/conf/server.xml
` ` `

### **2.3 检查缓存问题 **
` ` ` bash
# 检查 Tomcat 的 Session 超时设置（单位：分钟）
grep -A 1 "<session-config>" /usr/local/apache-tomcat-8.0.39/conf/web.xml
` ` `

**建议 **：确保 `<session-timeout>30</session-timeout>`（默认 30 分钟）。

### **2.4 重启 Tomcat 并监控 **
` ` ` bash

```
# 重启 Tomcat
/usr/local/apache-tomcat-8.0.39/bin/shutdown.sh
/usr/local/apache-tomcat-8.0.39/bin/startup.sh

# 监控内存变化
watch -n 5 'ps -p $(pgrep -f "Bootstrap start") -o %mem,rss,comm'
```

---

## **3. 预期优化效果**

| 指标 | 优化前 | 优化后目标 |
|---|---|---|
| **RSS 内存** | 5.26GB | 3.5-4GB |
| **线程栈内存** | 179MB (179 threads) | 25MB (100 threads) |
| **MetaSpace** | 260MB (无限制) | ≤256MB |
| **Direct Memory** | 可能泄漏 (1GB) | ≤512MB |

---

## **4. 后续维护建议**

1. **定期检查 GC 日志**:
   ```bash
   tail -f /usr/local/apache-tomcat-8.0.39/logs/gc.log | grep -E "Full GC|OOM"
   ```

2. **设置内存报警** (如 Prometheus + Grafana):
   - 报警规则: `RSS > 3.5GB` 持续 10 分钟。

3. **升级 JDK**:
   - 当前使用 **JDK 1.8.0_111** (旧版，存在已知内存问题)，建议升级到 **1.8.0_381+**。

---

## **5. 紧急情况处理**

如果优化后仍出现 **OOM**:

1. **快速 Dump 内存**:
   ```bash
   jmap -dump:live,format=b,file=/tmp/tomcat_heap.hprof 31037
   ```
2. **分析内存泄漏**:
   - 使用 [Eclipse MAT](https://www.eclipse.org/mat/) 分析 `tomcat_heap.hprof`。

---

**结论**: 通过限制 MetaSpace、减少线程栈大小、优化 JVM 参数，**预计可降低 1-2GB 内存占用**，显著减少 OOM 风险。建议在低峰期实施并持续监控。

Revision #2
Created 2025-08-02 02:35:21 UTC by Admin
Updated 2025-08-02 03:11:11 UTC by Admin