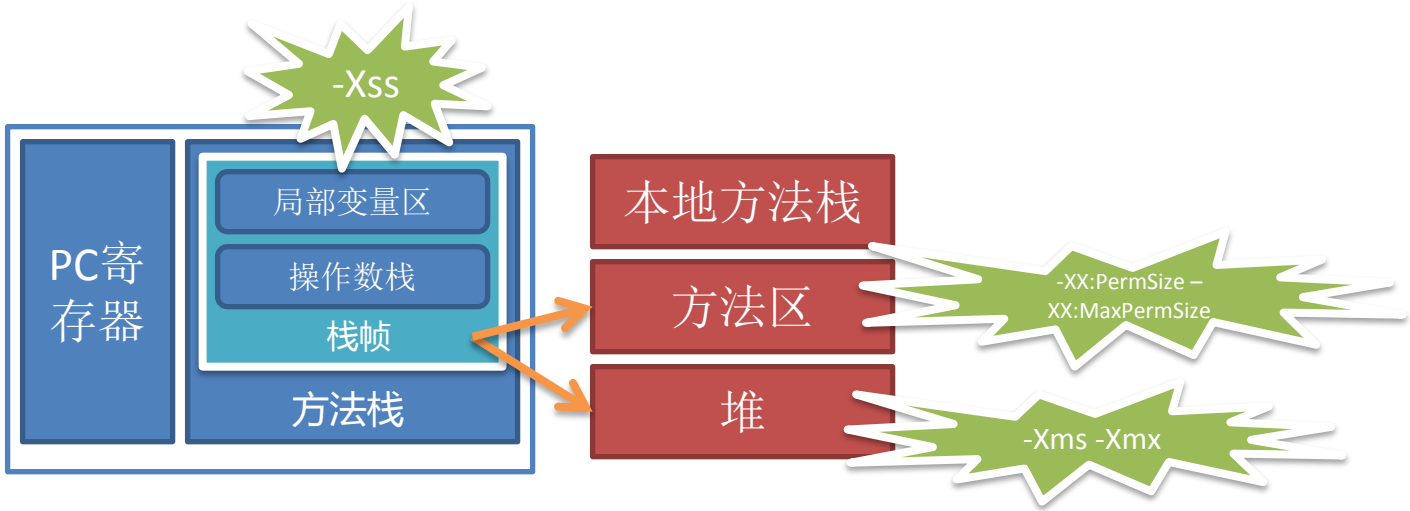
# Java内存管理问题案例分享

毕玄

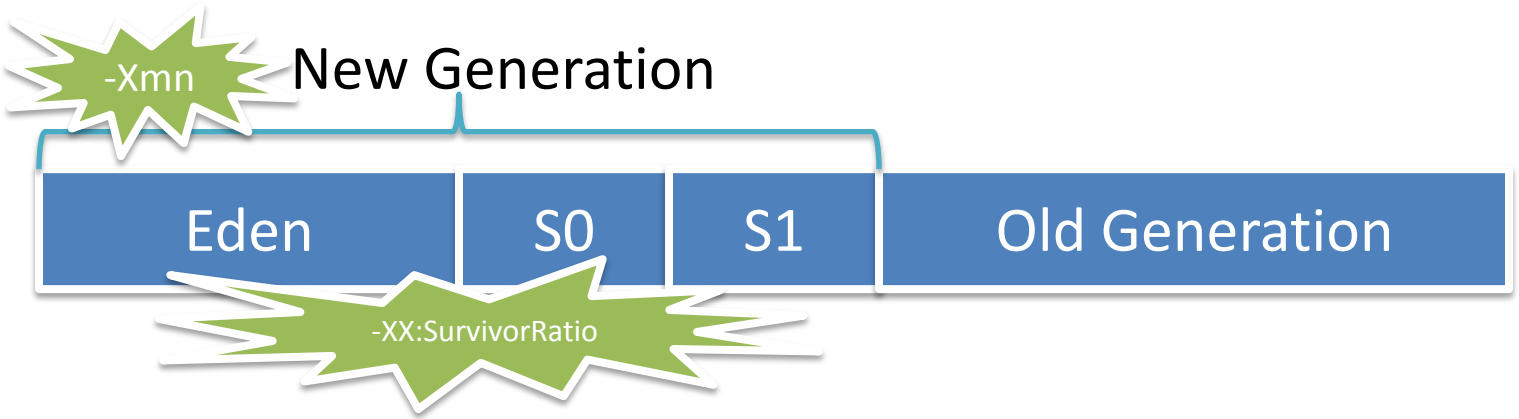
2012-03

- **Java Runtime Data Area**




备注：在Sun JDK中本地方法栈和方法栈是同一个，因此也可用-Xss控制

- **Oracle JDK Java Heap划分**



- **Oracle JDK内存管理常用参数**
  - -Xms –Xmx –Xmn –XX:PermSize –XX:MaxPermSize
  - -XX:SurvivorRatio
  - -XX:+UseParallelGC || -XX:+UseParallelOldGC
    - -XX:MaxTenuringThreshold -XX:ParallelGCThreads
    - -XX:-UseAdaptiveSizePolicy
  - -XX:+UseConcMarkSweepGC
    - -XX:CMSInitiatingOccupancyFraction -XX:+UseCMSInitiatingOccupancyOnly
  - -XX:+DisableExplicitGC –XX:+ExplicitGCInvokesConcurrent

# 基本知识

- **内存管理问题排查常用参数或工具**
  - -XX:+PrintGCDateStamps –XX:+PrintGCDetails -Xloggc
  - -XX:+HeapDumpOnOutOfMemoryError
  - -XX:+HeapDumpBeforeFullGC
  - -XX:+PrintFlagsFinal（JDK6u21+）
  - for CMS GC
    - -XX:+PrintPromotionFailure -XX:+CMSDumpAtPromotionFailure
  - jinfo
  - jstat
  - jmap
  - MAT
  - btrace
  - google perf-tools

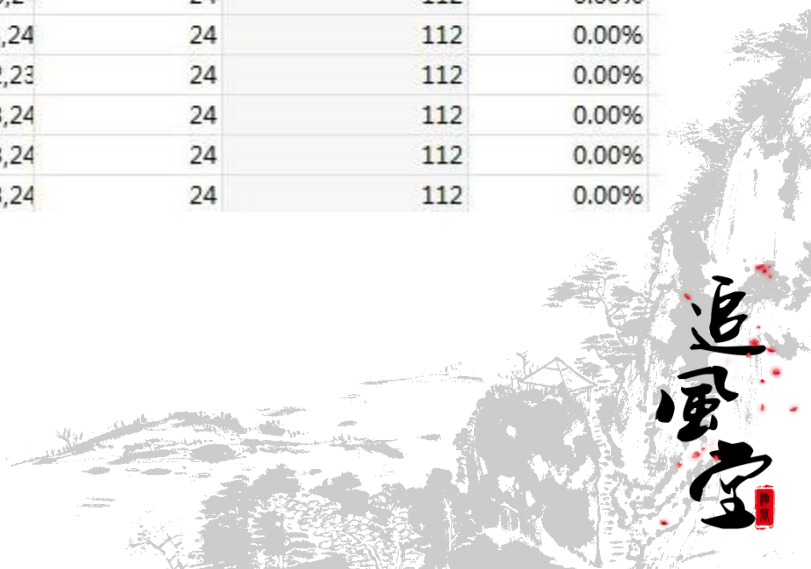
# 可能会碰到的问题

- **OOM**
  - java.lang.OutOfMemoryError: {$reason}
- **Full GC频繁**
- **CMS GC**
  - promotion failed
  - concurrent mode failure

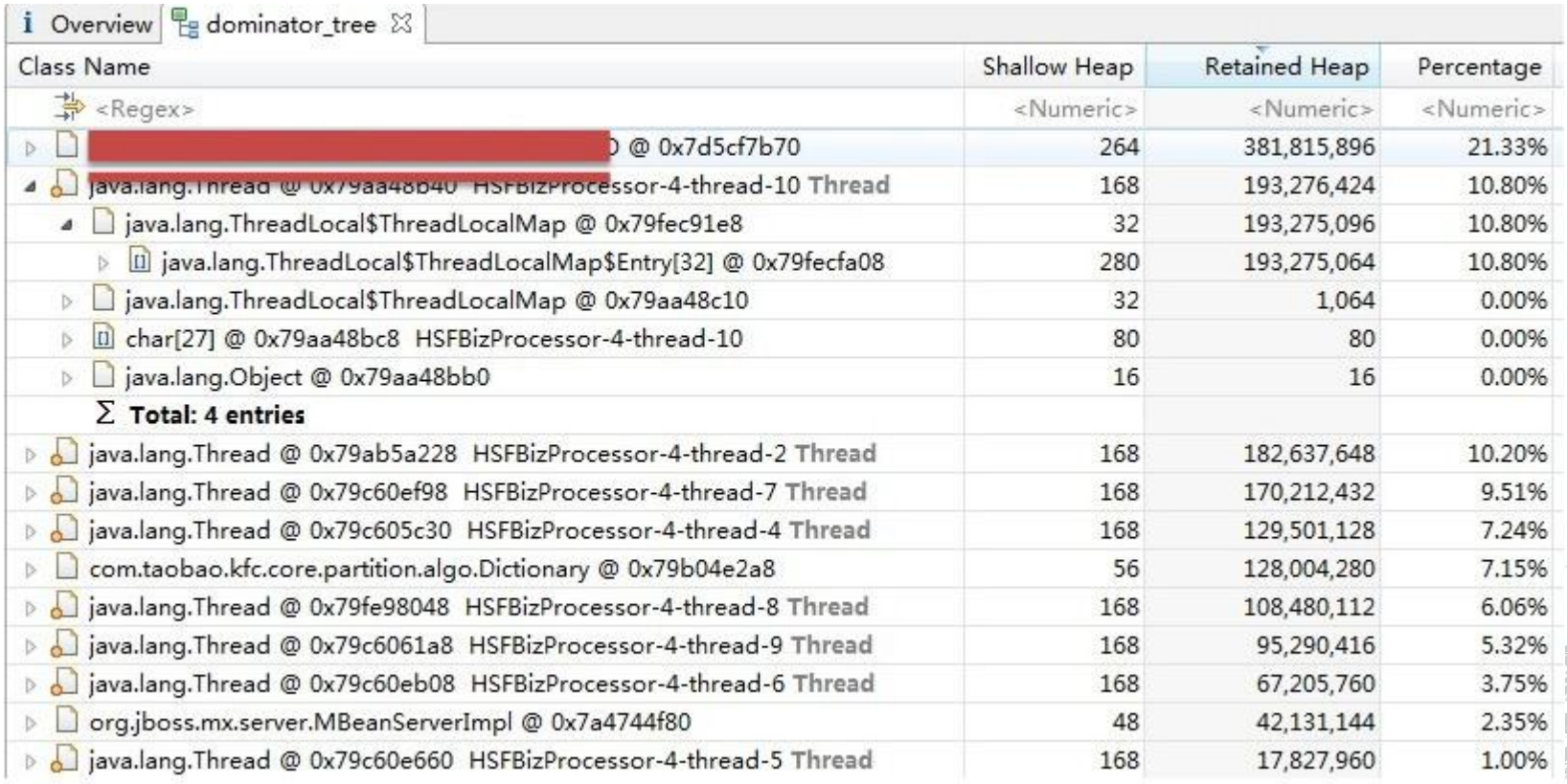
# OOM案例一

Overview | dominator_tree

| Class Name | Shallow Heap | Retained Heap | Percentage |
|---|---|---|---|
| <Regex> | <Numeric> | <Numeric> | <Numeric> |
| java.lang.Thread @ 0x5aead990 ajp-0.0.0.0-8009-111 **Thread** | 104 | 1,086,816,976 | 86.27% |
| com.taobao[redacted] @ 0x6c103760 | 16 | 1,086,773,392 | 86.27% |
| java.util.ArrayList @ 0x6c103730 | 24 | 1,086,773,376 | 86.27% |
| java.lang.Object[11451103] @ 0x8c38ffa8 | 45,804,424 | 1,086,773,352 | 86.27% |
| java.lang.String @ 0x9fedbe78 2,7,9,11,12,15,17,20,21,22,24,25,26 | 24 | 112 | 0.00% |
| java.lang.String @ 0x67bc19b0 2,7,8,10,11,12,13,14,17,21,22,23,24 | 24 | 112 | 0.00% |
| java.lang.String @ 0x67bc19c8 1,7,8,10,11,12,13,14,17,21,22,23,24 | 24 | 112 | 0.00% |
| java.lang.String @ 0x6194e980 1,5,8,10,11,12,14,17,19,20,21,22,23 | 24 | 112 | 0.00% |
| java.lang.String @ 0x67bc1a38 5,6,8,10,11,12,13,14,17,21,22,23,24 | 24 | 112 | 0.00% |
| java.lang.String @ 0x67bc1a50 4,6,8,10,11,12,13,14,17,21,22,23,24 | 24 | 112 | 0.00% |
| java.lang.String @ 0x67bc1a68 3,6,8,10,11,12,13,14,17,21,22,23,24 | 24 | 112 | 0.00% |



# OOM案例二

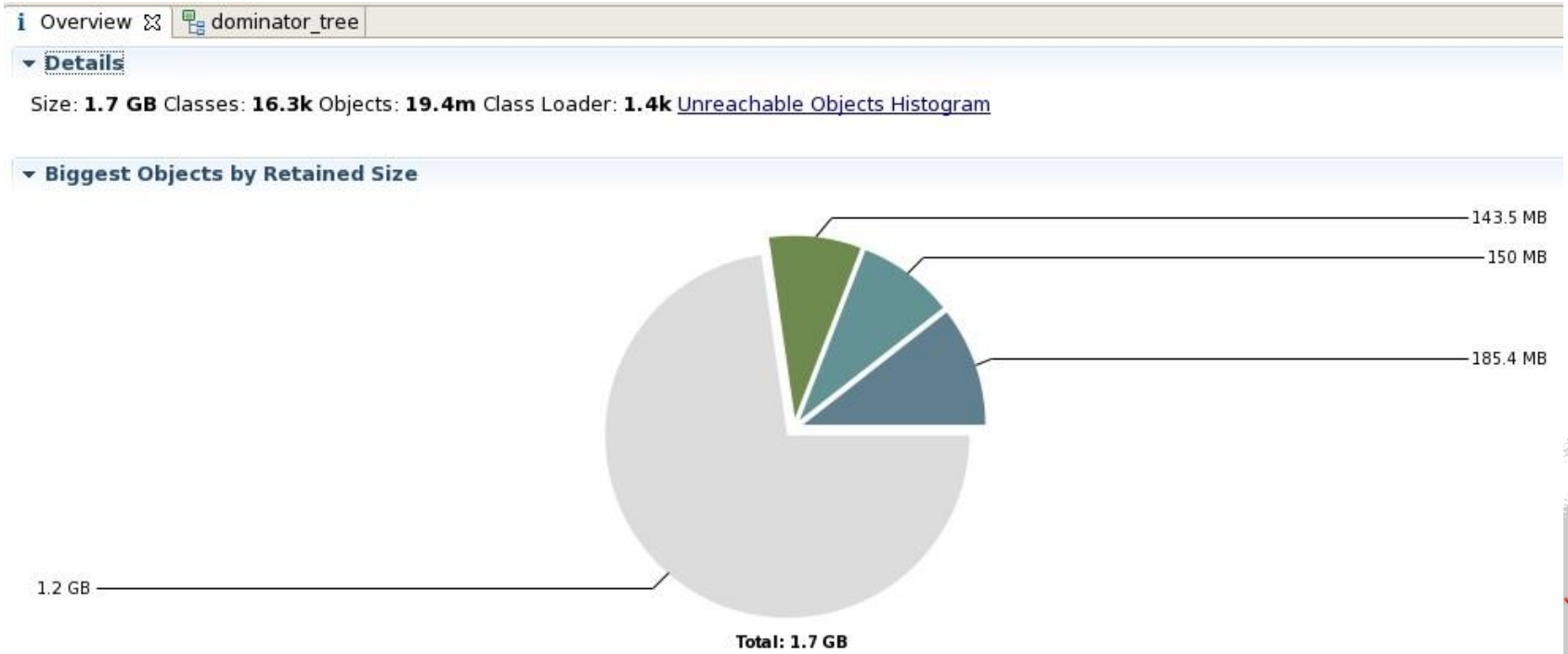


Overview | dominator_tree

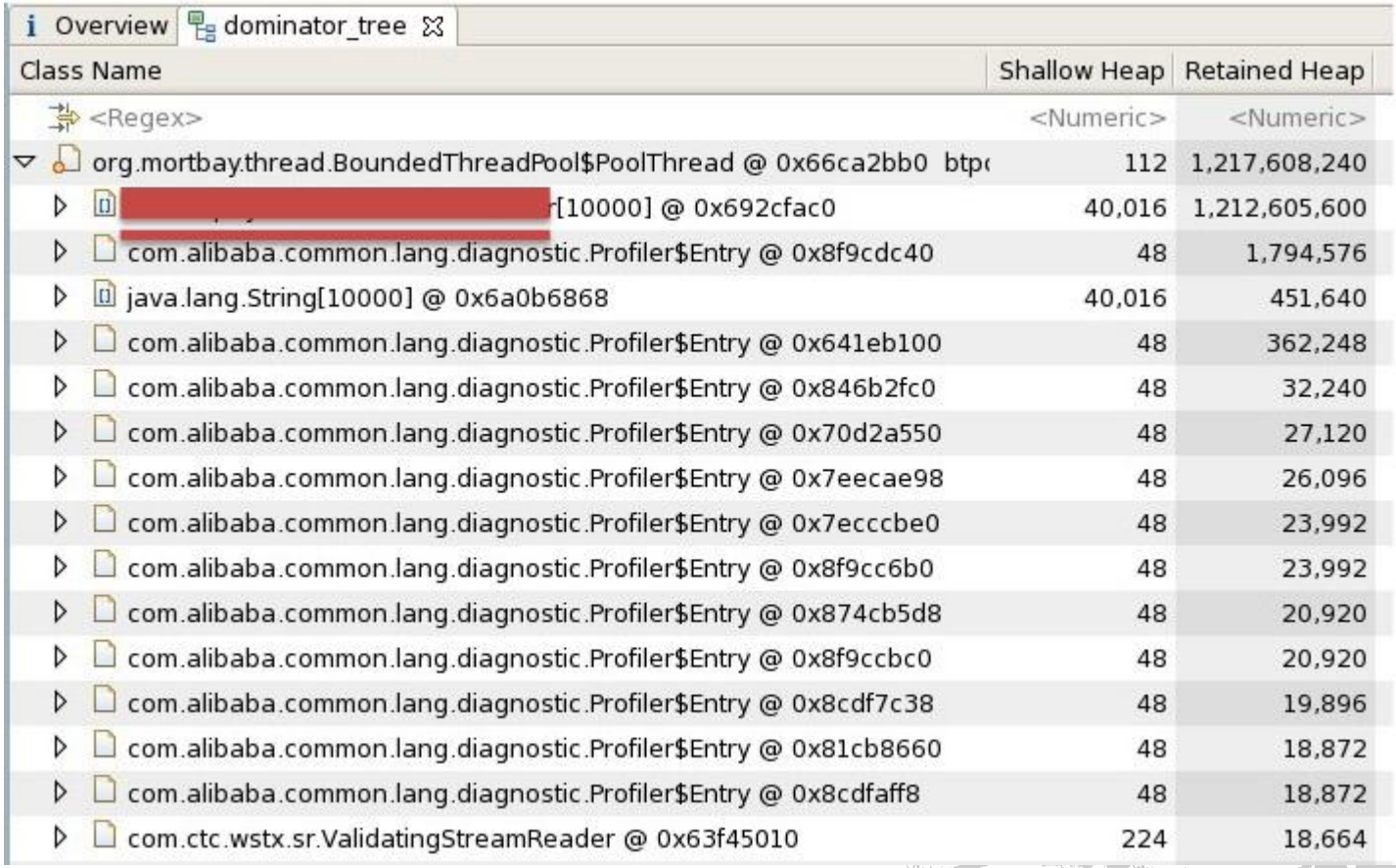
| Class Name | Shallow Heap | Retained Heap | Percentage |
|---|---|---|---|
| <Regex> | <Numeric> | <Numeric> | <Numeric> |
| [redacted] @ 0x7d5cf7b70 | 264 | 381,815,896 | 21.33% |
| java.lang.Thread @ 0x79aa48b40 HSFBizProcessor-4-thread-10 Thread | 168 | 193,276,424 | 10.80% |
| java.lang.ThreadLocal$ThreadLocalMap @ 0x79fec91e8 | 32 | 193,275,096 | 10.80% |
| java.lang.ThreadLocal$ThreadLocalMap$Entry[32] @ 0x79fecfa08 | 280 | 193,275,064 | 10.80% |
| java.lang.ThreadLocal$ThreadLocalMap @ 0x79aa48c10 | 32 | 1,064 | 0.00% |
| char[27] @ 0x79aa48bc8 HSFBizProcessor-4-thread-10 | 80 | 80 | 0.00% |
| java.lang.Object @ 0x79aa48bb0 | 16 | 16 | 0.00% |
| Σ **Total: 4 entries** | | | |
| java.lang.Thread @ 0x79ab5a228 HSFBizProcessor-4-thread-2 Thread | 168 | 182,637,648 | 10.20% |
| java.lang.Thread @ 0x79c60ef98 HSFBizProcessor-4-thread-7 Thread | 168 | 170,212,432 | 9.51% |
| java.lang.Thread @ 0x79c605c30 HSFBizProcessor-4-thread-4 Thread | 168 | 129,501,128 | 7.24% |
| com.taobao.kfc.core.partition.algo.Dictionary @ 0x79b04e2a8 | 56 | 128,004,280 | 7.15% |
| java.lang.Thread @ 0x79fe98048 HSFBizProcessor-4-thread-8 Thread | 168 | 108,480,112 | 6.06% |
| java.lang.Thread @ 0x79c6061a8 HSFBizProcessor-4-thread-9 Thread | 168 | 95,290,416 | 5.32% |
| java.lang.Thread @ 0x79c60eb08 HSFBizProcessor-4-thread-6 Thread | 168 | 67,205,760 | 3.75% |
| org.jboss.mx.server.MBeanServerImpl @ 0x7a4744f80 | 48 | 42,131,144 | 2.35% |
| java.lang.Thread @ 0x79c60e660 HSFBizProcessor-4-thread-5 Thread | 168 | 17,827,960 | 1.00% |

# OOM案例三

-Xms4g –Xmx4g –Xmn2560m



# OOM案例四



Overview | dominator_tree

| Class Name | Shallow Heap | Retained Heap |
|---|---|---|
| <Regex> | <Numeric> | <Numeric> |
| org.mortbay.thread.BoundedThreadPool$PoolThread @ 0x66ca2bb0 btp( | 112 | 1,217,608,240 |
| [redacted][10000] @ 0x692cfac0 | 40,016 | 1,212,605,600 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x8f9cdc40 | 48 | 1,794,576 |
| java.lang.String[10000] @ 0x6a0b6868 | 40,016 | 451,640 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x641eb100 | 48 | 362,248 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x846b2fc0 | 48 | 32,240 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x70d2a550 | 48 | 27,120 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x7eecae98 | 48 | 26,096 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x7ecccbe0 | 48 | 23,992 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x8f9cc6b0 | 48 | 23,992 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x874cb5d8 | 48 | 20,920 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x8f9ccbc0 | 48 | 20,920 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x8cdf7c38 | 48 | 19,896 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x81cb8660 | 48 | 18,872 |
| com.alibaba.common.lang.diagnostic.Profiler$Entry @ 0x8cdfaff8 | 48 | 18,872 |
| com.ctc.wstx.sr.ValidatingStreamReader @ 0x63f45010 | 224 | 18,664 |

# OOM案例五

```
#
# An unexpected error has been detected by Java Runtime Environment:
#
# java.lang.OutOfMemoryError: requested 20971520 bytes for GrET in /BUILD_AREA/jdk6_07/hotspot/src/share/vm/utilities/growableArray.
cpp. Out of swap space?
#
#  Internal Error (allocation.inline.hpp:42), pid=4567, tid=1042095008
#  Error: GrET in /BUILD_AREA/jdk6_07/hotspot/src/share/vm/utilities/growableArray.cpp
#
# Java VM: Java HotSpot(TM) Server VM (10.0-b23 mixed mode linux-x86)
# If you would like to submit a bug report, please visit:
#   http://java.sun.com/webapps/bugreport/crash.jsp
#
```

```
---------------  S Y S T E M  ---------------

OS:Red Hat Enterprise Linux AS release 4 (Nahant Update 8)

uname:Linux 2.6.9-89.ELxenU #1 SMP Mon Apr 20 10:56:05 EDT 2009 i686
libc:glibc 2.3.4 NPTL 2.3.4
rlimit: STACK 10240k, CORE infinity, NPROC 65664, NOFILE 65535, AS infinity
load average:2.77 0.69 0.22

CPU:total 4 (8 cores per cpu, 2 threads per core) family 6 model 10 stepping 5, cmov, cx8, fxsr, mmx, sse, sse2, sse3, ssse3, ht

Memory: 4k page, physical 4194436k(24548k free), swap 2096472k(2096304k free)
```

# OOM案例五

```
[                5 oomcases]$ ~/google-perftools/bin/pprof [                    ]java --text oomcase.hprof_9402.0008.heap
Using local file [                    ]/java.
Using local file oomcase.hprof_9402.0008.heap.
Total: 800.3 MB
   798.9  99.8%  99.8%    798.9  99.8% zcalloc
     1.0   0.1%  99.9%      1.0   0.1% os::malloc
     0.2   0.0% 100.0%      0.2   0.0% readCEN
     0.2   0.0% 100.0%    799.0  99.8% Java_java_util_zip_Deflater_init
```

Then to find who use Deflater.init

# OOM类型

- **java.lang.OutOfMemoryError: {$reason}**
  - GC overhead limit exceeded
  - Java Heap Space
  - Unable to create new native thread
  - PermGen Space
  - Direct buffer memory
  - request {} bytes for {}. Out of swap space?

# OOM产生的影响

- **GC overhead limit exceeded**

  **Java Heap Space**

  - 未处理异常会造成线程退出
  - 线程调度慢
  - 系统响应慢或无响应
    - because gc频繁

- **Out of Swap**

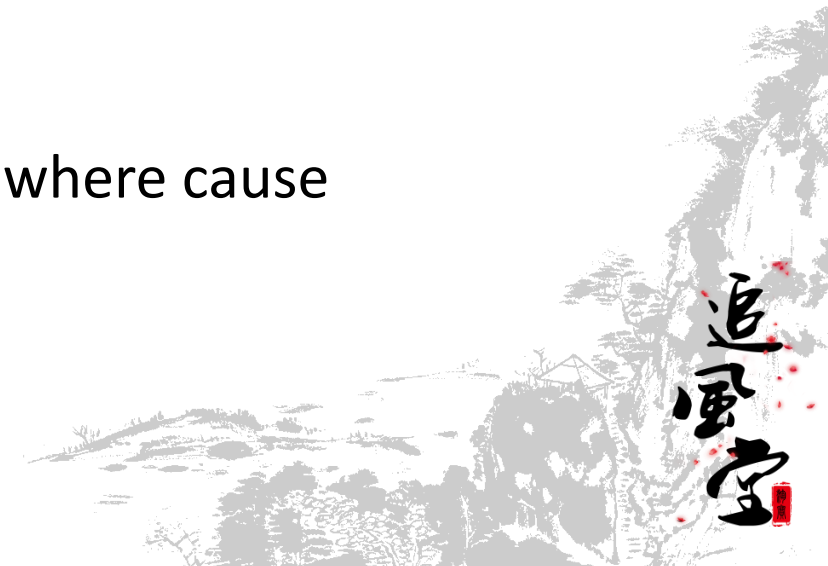
  - Java进程crash

# OOM解决方法

- **GC overhead limit exceeded**

  **Java Heap Space**

  - monitor app log & gc log
  - heap dump or jmap –histo|-dump when oom
  - mat analyze heap dump
  - app developer or btrace to find where cause



# OOM解决方法

- **Out of swap**
  - crash log
  - google-perftools
  - then btrace to find where cause
  - ps: 也可以先尝试下把-XX:+DisableExplicitGC去掉，原因见此link。

# OOM友好的代码

- 慎用ThreadLocal；
- 限制Collection/StringBuilder等的大小；
- 限制提交的请求的大小，尤其是批量处理；
- 限制数据库返回的数据的大小；
- 避免死循环。

# 可能会碰到的问题

- ~~OOM~~
  - ~~java.lang.OutOfMemoryError: {$reason}~~
- **Full GC频繁**
- CMS GC
  - promotion failed
  - concurrent mode failure

# Full GC频繁案例

- **大部分Case**
  - 内存占用过多未释放，但又还不到OOM
- **一个特殊的Case**
  - 一个GC频繁的Case

# Full GC频繁造成的影响

- 应用响应慢

# Full GC频繁解决方法

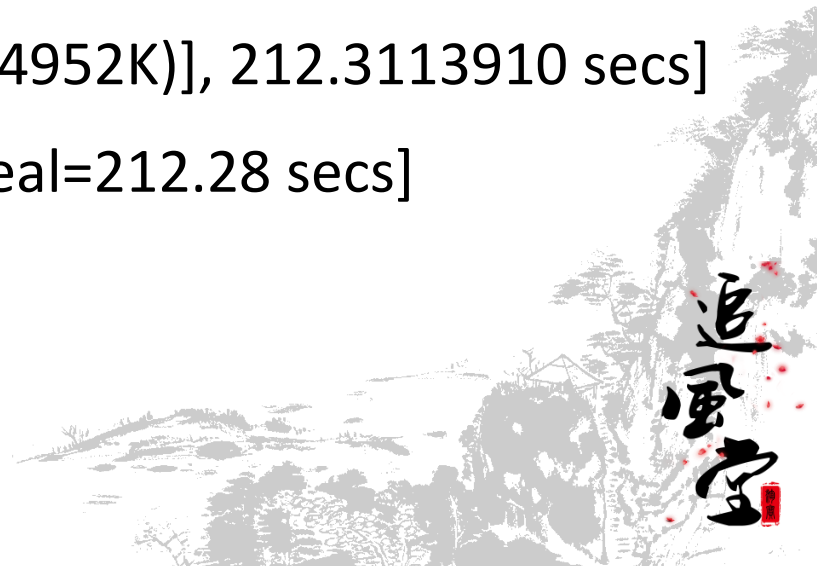
- **根据gc log判断是不是内存占用过多造成的；**
  - 如是则可以加上-XX:+HeapDumpBeforeFullGC或写个脚本，在FGC频繁时dump下内存；
  - 如不是则需要用pstack+源码来看下具体是什么原因触发的。
    - 如是淘宝版的JDK，则可加上-XX:+PrintGCReason

# 可能会碰到的问题

- ~~OOM~~
  - ~~java.lang.OutOfMemoryError: {$reason}~~
- ~~Full GC频繁~~
- **CMS GC**
  - promotion failed
  - concurrent mode failure

- **promotion failed案例一**
  - 363439.937: [Full GC 363439.938: [ParNew (promotion failed): 523840K->523840K(523840K), 0.3404490 secs] 1690606K->1712399K(1834560K), 0.3412880 secs]
  - 这个案例中淘宝版JDK立下了巨大功劳
    - ==WARNNING== allocating large array: thread_id[0x00002aaac2d85800], thread_name[ajp-0.0.0.0-8009-48], array_size[782611408 bytes], array_length[195652847 elememts]

- **promotion failed案例二**
  - 1768.432: [GC 1768.432: [ParNew (promotion failed): 1572827K->1572827K(1572864K), 0.2845480 secs]
  - 1972.208: [CMS: 10545160K->5853155K(14680064K), 8.5355320 secs] 12105682K->5853155K(16252928K), [CMS Perm : 20907K->20873K(34952K)], 212.3113910 secs] [Times: user=53.38 sys=22.45, real=212.28 secs]
  - 悲催的CMS GC碎片问题



- **空口讲四个**
  - promotion failed案例三
    - 触发比率过高…
  - concurrent mode failure案例一
    - 触发比率过高…
  - concurrent mode failure案例二
    - 旧生代小
  - concurrent mode failure案例三
    - 碰到的一个permgen造成问题的case

# CMS GC问题造成的影响

- CMS GC问题触发Serial Full GC，从而导致应用响应慢...

# CMS GC问题解决方法

- **看gc log，判断是否由于触发比率过高或旧生代过小造成，如是则调整相应的参数；**
- **如不是，则继续**
  - promotion failed
    - 如为内存用完的情况，则dump内存分析；
    - 如为cms gc碎片问题，暂时只能定时执行下jmap –histo:live；
  - concurrent mode failure
    - 同promotion failed...

# 可能会碰到的问题

- OOM
  - java.lang.OutOfMemoryError: {$reason}
- Full GC频繁
- CMS GC
  - promotion failed
  - concurrent mode failure

# 你可能还会碰到

- **Young GC速度非常慢**
  - 存活对象小的情况下也在几十毫秒以上；
- **CMS GC Concurrent-Mark阶段造成了应用暂停；**
- **…**

# References

- 记录贴：碰到的一些Java问题
- perftools查看堆内存并解决hbase内存溢出
- BTrace使用简介