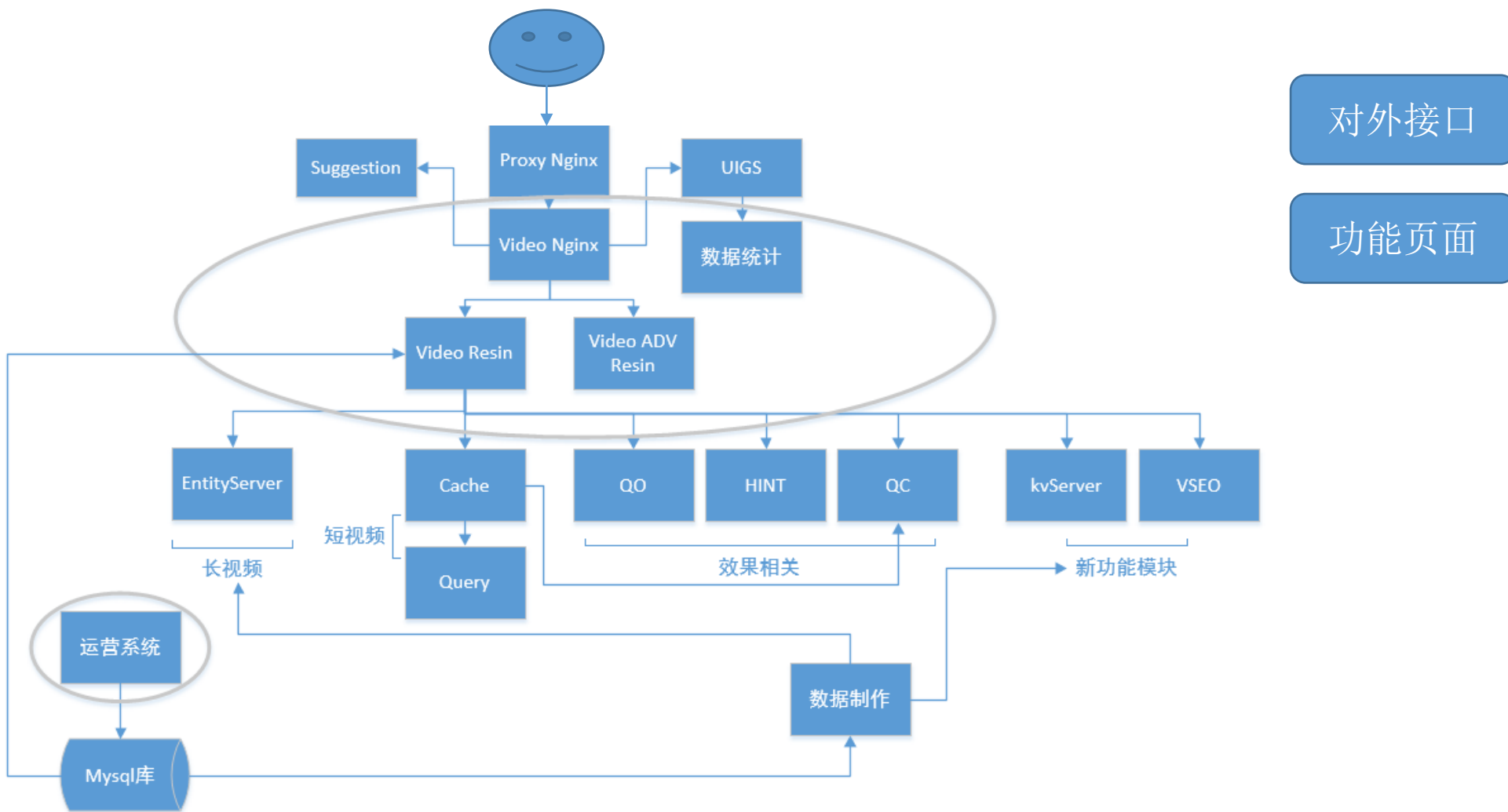


# Java性能优化经验以及Java内存数据库实践

前端接口响应时间优化经验和视频广告服务实现

穆琼

# 视频搜索线上服务架构



为您找到“奔跑吧第二季”的相关作品：5部综艺



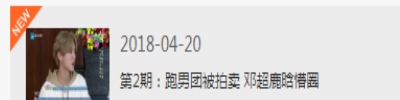
## 奔跑吧第二季 综艺

地 区：内地 类 型：真人秀

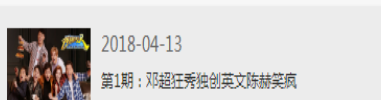
主持人：邓超 Angelababy 李晨 陈赫 郑恺

简介：《奔跑吧第二季》在联合国维也纳总部和奥地利国家旅游局的支持下，来到联合国的维也纳总部进行学习访问，积极响应并致力于带动更多人在节目中了解联合国的可... 更多▼

腾讯视频 爱奇艺



2018-04-20  
第2期：跑男团被拍卖 邓超鹿晗情困



2018-04-13  
第1期：邓超狂秀独创英文陈赫笑疯

查看更多

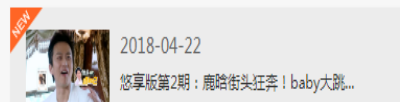


## 奔跑吧第二季粉丝悠享版 综艺

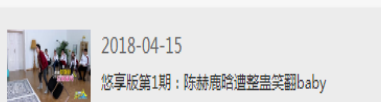
地 区：其他 类 型：户外真人秀

简介：《奔跑吧2粉丝悠享版》是浙江卫视推出的大型户外竞技真人秀节目衍生的会员节目，内包含《跑男来了》和其他正片没有的未播精彩内容，节目邀请邓超、李晨、杨颖... 更多▼

腾讯视频



2018-04-22  
悠享版第2期：鹿晗街头狂奔！baby大跳...



2018-04-15  
悠享版第1期：陈赫鹿晗遭整蛊笑翻baby

查看更多

综合排序 最新发布 时长 ▼ 发布 ▼ 品质 ▼ 来源 ▼



奔跑吧第二季 20180420

2018-04-20

腾讯



奔跑吧第二季 20180413

2018-04-12

腾讯



《奔跑吧第二季》：兄弟团启动演唱会，邓超鹿晗邀约...

2018-04-25

优酷



奔跑吧第二季第一期，跑男团猜外国人名字，我看了三...

2018-04-21

优酷



《奔跑吧第二季》：从杨颖和钟楚曦抢被子，就可以看...

2018-04-26

优酷



《奔跑吧第二季》：昆凌自曝睡觉抢周杰伦被子，心疼...

2018-04-26

优酷



《奔跑吧第二季》：陈赫徒手掰苹果，指甲都要断了！...

2018-04-25

优酷



《奔跑吧第二季》：邓超陈赫为抢心仪女生，决定闹掰！

2018-04-25

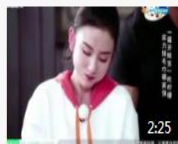
优酷



《奔跑吧第二季》：陈赫情商究竟有多高？黄渤胡歌...

2018-04-25

优酷



《奔跑吧第二季》：宋祖儿为什么参加奔跑？网友：想...

2018-04-25

优酷



《奔跑吧第二季》：郑恺自曝为baby穿了五季增高鞋...

2018-04-26

优酷



《奔跑吧第二季》：陈赫说自己像周杰伦？昆凌这样...

2018-04-26

优酷



《奔跑吧第二季》：这段没播？昆凌“弹指神功”把跑男...

2018-04-26

优酷



奔跑吧第二季：从来没有觉得跑男撕名牌声音，如此清...

2018-04-25

优酷



《奔跑吧第二季》：迪丽热巴不能再参加跑男，真的是...

2018-04-23

优酷



《奔跑吧第二季》：邓超说只要不遇到李晨就行，可惜...

2018-04-24

优酷



《奔跑吧第二季》：钟楚曦一口一瓣柠檬，吃得李晨三...

2018-04-24

优酷



《奔跑吧第二季》：节目没这段？angelababy、李...

2018-04-24

优酷



《奔跑吧第二季》：宋祖儿可爱又美丽，迷倒跑男团...

2018-04-23

优酷

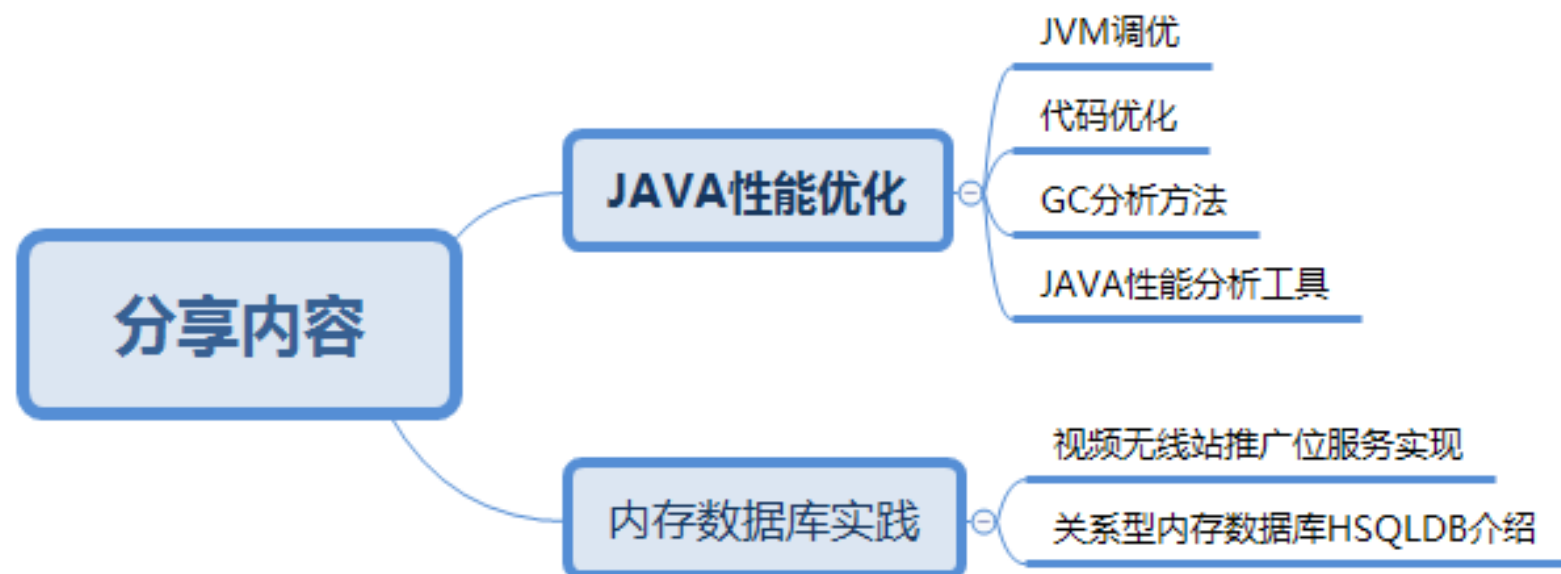


《奔跑吧第二季》：鹿晗一见到蓝盈莹，就提出这样的...

2018-04-24

优酷

# 目录



# 前端接口性能优化——现象

- 现象：部分请求耗时较高，以内部VR接口为例

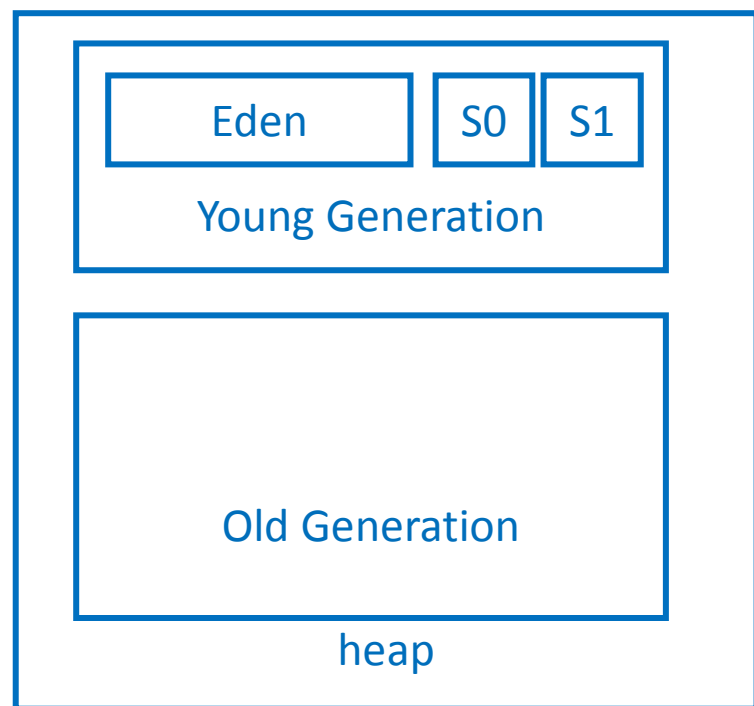
现象	
Resin cost不高	Nginx cost 高
Resin log中cache cost高	Cache cost不高



- 代码优化：简化VR请求处理逻辑，效果不明显
- 对照resin GC log，某些时刻GC停顿时间（stopped time）较长

# 前端接口性能优化——JVM调优

- JVM调参，调优目标：减少慢查询比例，保证系统整体性能平稳
- GC策略：分代收集，ParNew+CMS收集器



*-Xmx: 最大堆大小*

*-Xms: 初始堆大小*

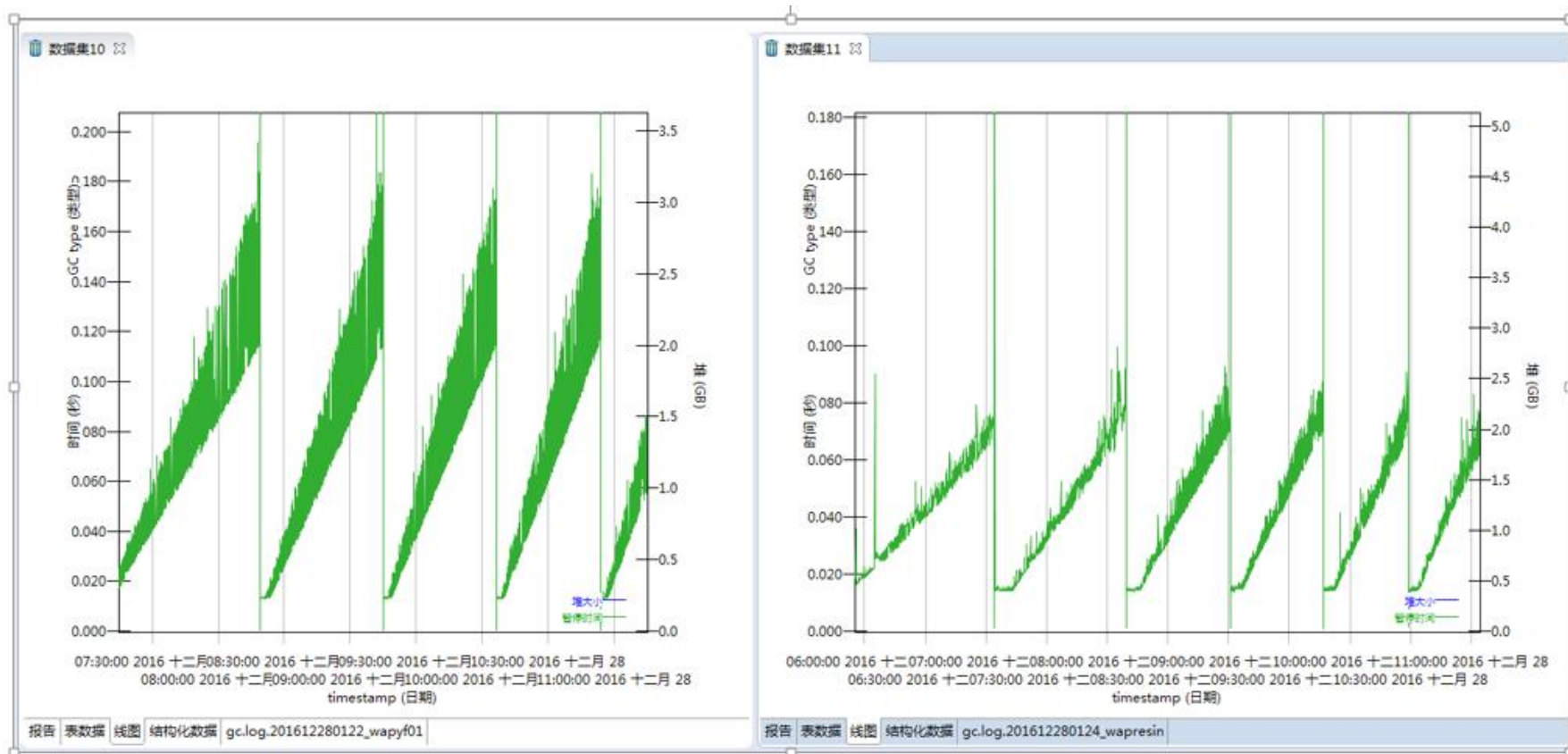
*-XX:MaxTenuringThreshold: 最大复制年龄*

*-XX:CMSInitiatingOccupancyFraction: 触发CMS回收老年代堆空间使用率*

*-XX:SurvivorRatio: Eden区与Survivor区的大小比例*

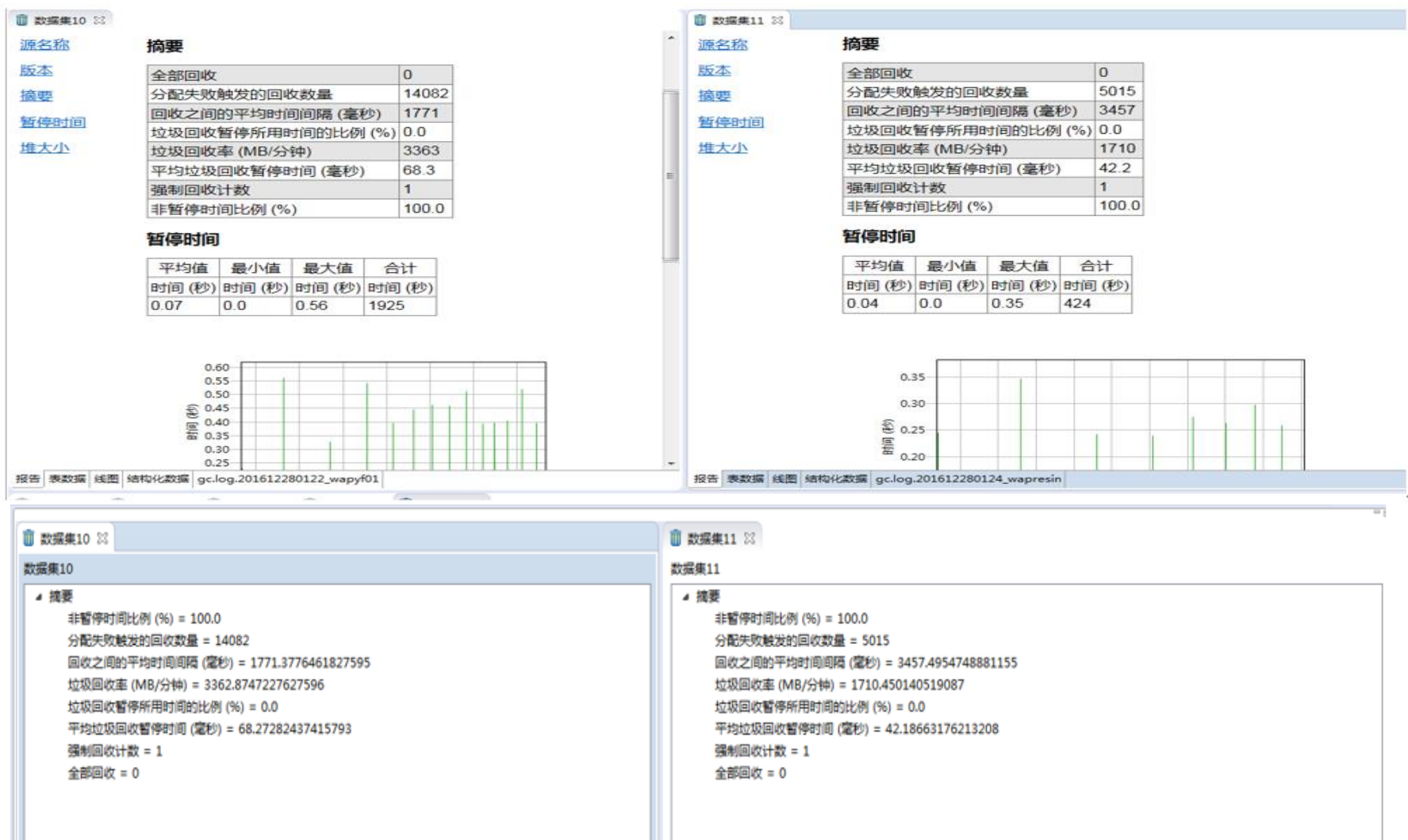
# 前端接口性能优化

- 调大内存、调整比例加快major gc的频率，使gc stop time减少





# 前端接口性能优化





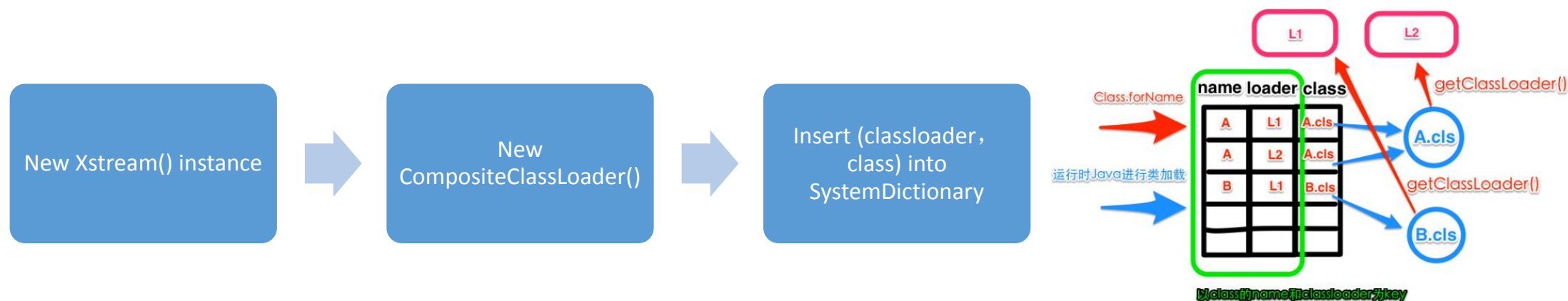
# 前端接口性能优化

- 新站上线，压力测试现象
  - ✓ 只对页面请求打压平均cost和gc log正常，较平稳
  - ✓ 对VR接口+页面请求打压young gc暂停时间逐渐变长，不正常



# 前端接口性能优化

- 接口和页面最大的区别：接口生成xml
- Xstream使用
- 每次请求都进行new Xstream会发生什么？？？



# 前端接口性能优化

- CMS GC在开启CMSClassUnloadingEnabled的情况下是可能对类做卸载动作的，此时会对SystemDictionary进行清理

*<jvm-arg>-XX:+CMSClassUnloadingEnabled</jvm-arg>*

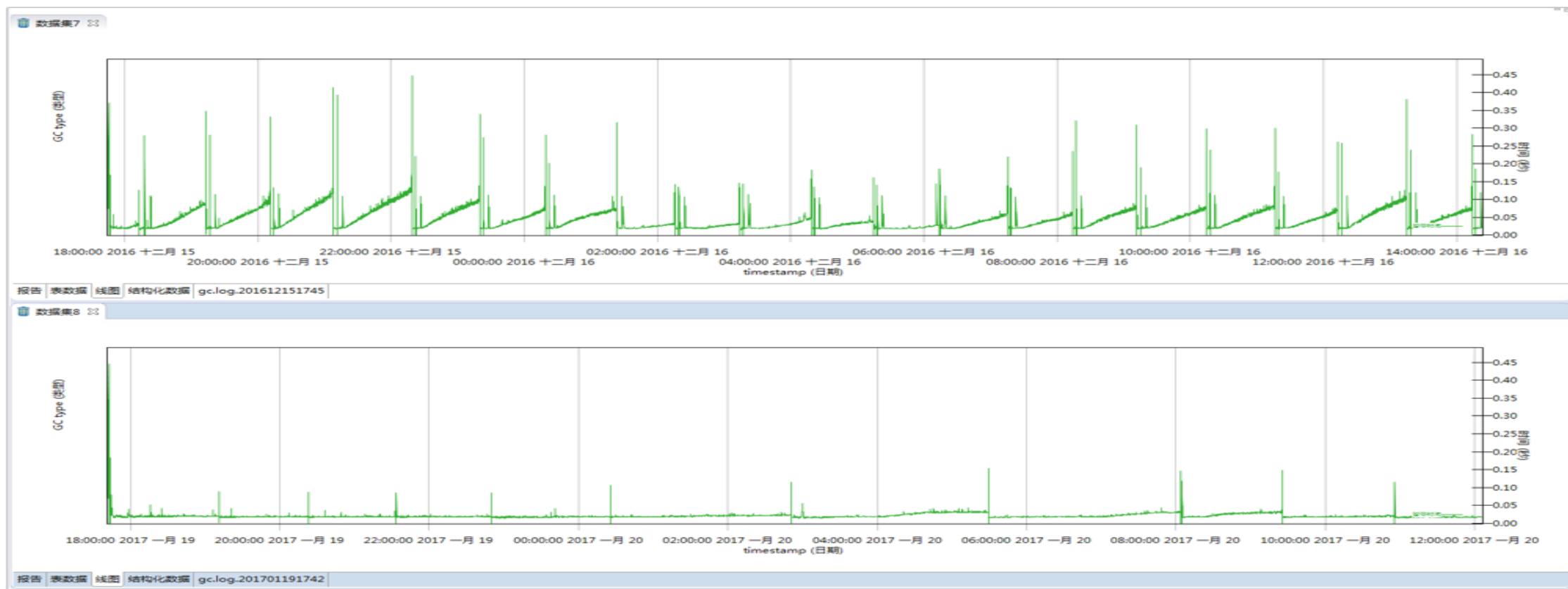
- 因此，CMS GC后GC暂停时间会降下来

# 前端接口性能优化

- 修改xml序列化实现
- 为每个类实现toxml()接口
- 更灵活、更易控制

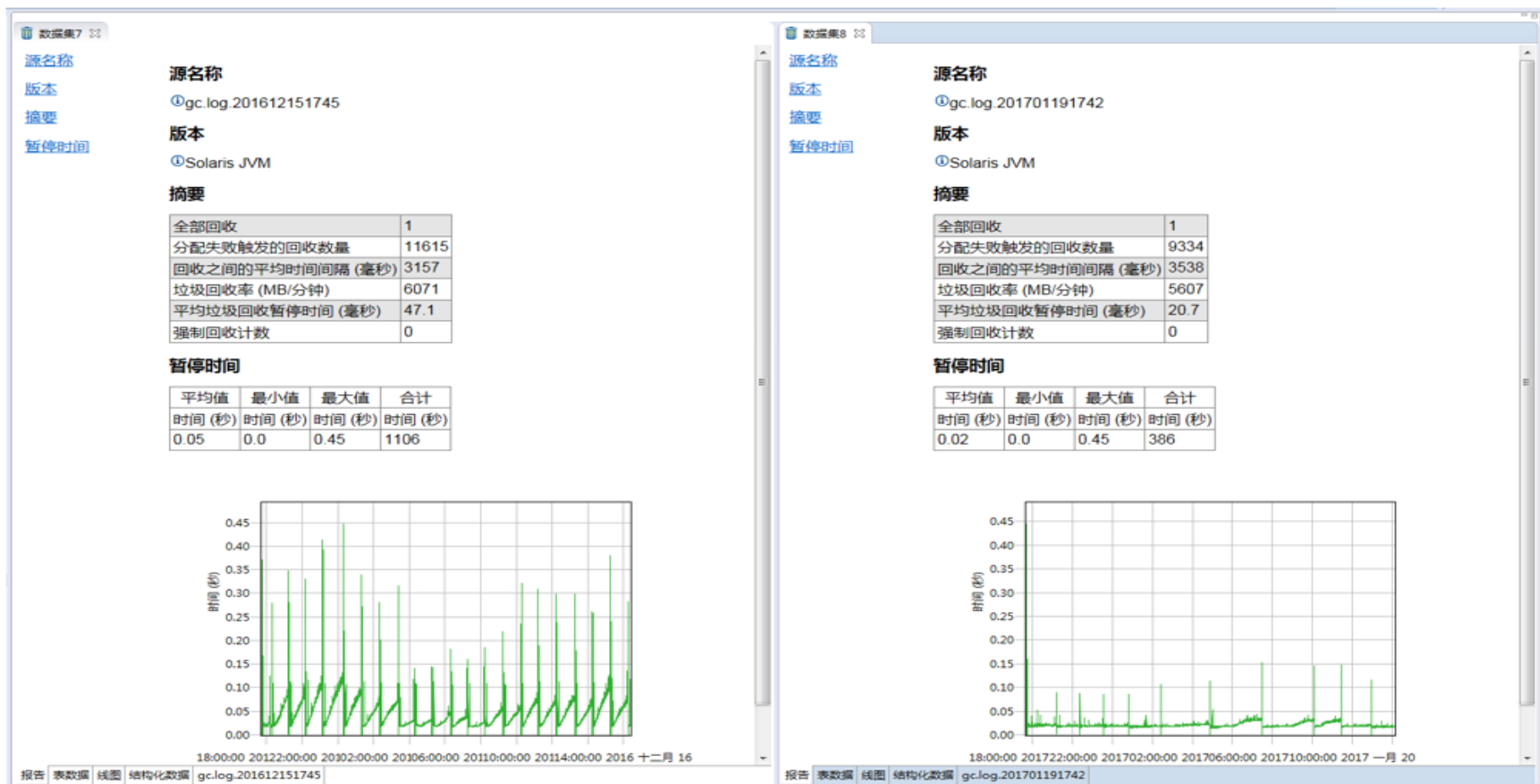
# 前端接口性能优化效果

- GC stop time变平稳



# 前端接口性能优化效果

- GC stop time变短



# 前端接口性能优化效果

- Resin cost大于300ms请求比例减少**70%+**: 0.0098341248794---》0.00220929164945





# 前端接口性能优化总结

- 第三方类库最佳实践
- 充分考虑请求量增长时代码的性能
- 注重代码，选择合适的GC算法

# 前端接口性能优化总结

- 开启GC log

- XX:+PrintGCDetails : 输出GC日志

- XX:+PrintGCDateStamps: 输出GC的时间

- XX:+PrintTenuringDistribution : 打印年龄情况

- XX:+PrintGCApplicationStoppedTime : 打印GC暂停时间

- XX:+PrintGCApplicationConcurrentTime : 打印GC之间应用程序运行时间

- XX:+PrintHeapAtGC : 打印GC时的堆信息

# 前端接口性能优化总结

## • GC log分析

GC次数 →

回收前年轻代使用情况 →

CMS老年代使用情况 →

持久代perm区使用情况 →

GC后年轻代变化 →

```
2018-04-19T16:37:30.944+0800: 1318375.432: Application time: 5.2945260 seconds
{Heap before GC invocations=327587 (full 41):
 par new generation total 460800K, used 416139K [0x0000000660000000, 0x000000067f400000, 0x000000067f400000)
  eden space 409600K, 100% used [0x0000000660000000, 0x0000000679000000, 0x0000000679000000)
  from space 51200K, 12% used [0x0000000679000000, 0x0000000679662df0, 0x000000067c200000)
  to   space 51200K,  0% used [0x000000067c200000, 0x000000067c200000, 0x000000067f400000)
 concurrent mark-sweep generation total 5779456K, used 3410207K [0x000000067f400000, 0x00000007e0000000, 0x00000007e0000000)
 concurrent-mark-sweep perm gen total 262144K, used 55032K [0x00000007e0000000, 0x00000007f0000000, 0x0000000800000000)
2018-04-19T16:37:30.948+0800: 1318375.436: [GC2018-04-19T16:37:30.948+0800: 1318375.436: [ParNew
Desired survivor size 26214400 bytes, new threshold 6 (max 6)
- age  1:    2457008 bytes,    2457008 total
- age  2:     526144 bytes,    2983152 total
- age  3:     431464 bytes,    3414616 total
- age  4:     434136 bytes,    3848752 total
- age  5:     362800 bytes,    4211552 total
- age  6:     337704 bytes,    4549256 total
: 416139K->6367K(460800K), 0.0130830 secs] 3826347K->3416900K(6240256K), 0.0133440 secs] [Times: user=0.05 sys=0.00, real=0.02 secs]
Heap after GC invocations=327588 (full 41):
 par new generation total 460800K, used 6367K [0x0000000660000000, 0x000000067f400000, 0x000000067f400000)
  eden space 409600K,  0% used [0x0000000660000000, 0x0000000660000000, 0x0000000679000000)
  from space 51200K, 12% used [0x000000067c200000, 0x000000067c837c58, 0x000000067f400000)
  to   space 51200K,  0% used [0x0000000679000000, 0x0000000679000000, 0x000000067c200000)
 concurrent mark-sweep generation total 5779456K, used 3410533K [0x000000067f400000, 0x00000007e0000000, 0x00000007e0000000)
 concurrent-mark-sweep perm gen total 262144K, used 55032K [0x00000007e0000000, 0x00000007f0000000, 0x0000000800000000)
}
2018-04-19T16:37:30.961+0800: 1318375.449: Total time for which application threads were stopped: 0.0175360 seconds
2018-04-19T16:37:36.476+0800: 1318380.965: Application time: 5.5153500 seconds
```

允许进入存活区的最大空间

Young gc用户时间、系统时间、实际暂停时间

# 前端接口性能优化总结

- GC log分析工具

1. Eclipse 插件: GCMV

2. 在线GC log分析: <http://gceasy.io/>

Total GC stats

Total GC count ?	312
Total reclaimed bytes ?	n/a
Total GC time ?	5 sec 170 ms
Avg GC time ?	17 ms
GC avg time std dev	8 ms
GC min/max time	10 ms / 110 ms
GC Interval avg time ?	25 sec 491 ms

Minor GC stats

Minor GC count	312
Minor GC reclaimed ?	202.71 gb
Minor GC total time	5 sec 170 ms
Minor GC avg time ?	17 ms
Minor GC avg time std dev	8 ms
Minor GC min/max time	10 ms / 110 ms
Minor GC Interval avg ?	25 sec 491 ms

Full GC stats

Full GC Count	0
Full GC reclaimed ?	n/a
Full GC total time	n/a
Full GC avg time ?	n/a
Full GC avg time std dev	n/a
Full GC min/max time	n/a
Full GC Interval avg ?	n/a

# Java内置监控工具

- JPS命令：列出系统上运行的所有java进程，支持-q,-m,-l,-v,-V等参数，可以输出进程PID、主类的包名、JVM参数等

```
[@bjzw_90_181 ~]# jps
24389 Jps
4647 Resin
3767 WatchdogManager
10136 Resin
10088 WatchdogManager
```

```
[@bjzw_90_181 ~]# jps -l
4647 com.caucho.server.resin.Resin
3767 com.caucho.boot.WatchdogManager
10136 com.caucho.server.resin.Resin
10088 com.caucho.boot.WatchdogManager
26152 sun.tools.jps.Jps
```

```
[@bjzw_90_181 ~]# jps -v
24368 Jps -Dapplication.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.111-1.b15.el7_2.x86_64 -Xms8m
4647 Resin -Xms10g -Xmx10g -Dfile.encoding=GBK -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSClassUnloadingEnabled -XX:NewSize=2000M -XX:MaxNewSize=2000M -XX:SurvivorRatio=1 -XX:CMSInitiatingOccupancyFraction=70 -XX:MaxTenuringThreshold=5 -XX:PermSize=200M -XX:+CMSScavengeBeforeRemark -XX:+UseNUMA -XX:+CMSParallelRemarkEnabled -XX:ParallelGCThreads=8 -XX:ParallelCMSThreads=8 -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCApplicationConcurrentTime -XX:+PrintHeapAtGC -Xdebug -Xrunjdwp:transport=dt_socket,address=9988,server=y,suspend=n -Dresin.server=1 -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djava.system.class.loader=com.caucho.loader.SystemClassLoader -Djavax.management.builder.initial=com.caucho.jmx.MBeanServerBuilderImpl -Djava.awt.headless=true -Dresin.home=/usr/local/resin_video/ -Xss1m -Xrs -Dresin.watchdog=video -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djavax.management.builder.initial=com.caucho.jmx.MB
3767 WatchdogManager -Dresin.watchdog=video -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djavax.management.builder.initial=com.caucho.jmx.MBeanServerBuilderImpl -Djava.awt.headless=true -Dresin.home=/usr/local/resin_video/ -Dresin.root=/usr/local/resin_video/ -Xrs -Xss256k -Xmx32m
10136 Resin -Xmx6144m -Xms6144m -Xmn500m -Xss512k -Dfile.encoding=GBK -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:ParallelGCThreads=4 -XX:+CMSClassUnloadingEnabled -XX:CMSInitiatingOccupancyFraction=70 -XX:PermSize=256m -XX:MaxPermSize=512m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCApplicationConcurrentTime -XX:+PrintHeapAtGC -Xdebug -Xrunjdwp:transport=dt_socket,address=7889,server=y,suspend=n -Dresin.server=1 -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djava.system.class.loader=com.caucho.loader.SystemClassLoader -Djavax.management.builder.initial=com.caucho.jmx.MBeanServerBuilderImpl -Djava.awt.headless=true -Dresin.home=/usr/local/resin_videowap2/ -Xrs -Dresin.watchdog=videowap2 -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djavax.management.builder.initial=com.caucho.jmx.MBeanServerBuilderImpl -Djava.awt.headless=true -Dresin.home=/usr/local/resin_videowap2/ -Dresin.root=/usr/local/resin_videowap2/ -Xloggc:log
10088 WatchdogManager -Dresin.watchdog=videowap2 -Djava.util.logging.manager=com.caucho.log.LogManagerImpl -Djavax.management.builder.initial=com.caucho.jmx.MBeanServerBuilderImpl -Djava.awt.headless=true -Dresin.home=/usr/local/resin_videowap2/ -Dresin.root=/usr/local/resin_videowap2/ -Xrs -Xss256k -Xmx32m
```

# Java内置监控工具

- jstack: 监控堆栈信息
- 定位线程阻塞、死锁等问题
- 常见线程状态包括:

- a) RUNNABLE
- b) BLOCKED
- c) WAITING
- d) TIMED\_WAITING

```
"pool-6-thread-28" #1232 prio=5 os_prio=0 tid=0x00007f1f5173e800 nid=0x5fe9 runnable [0x00007f24e9fe2000]
java.lang.Thread.State: RUNNABLE
    at com.yourkit.runtime.Callback.yjpMethodEntry0(Native Method)
    at com.yourkit.runtime.Callback.yjpMethodEntry(Callback.java:180)
    at java.nio.HeapByteBuffer.get(HeapByteBuffer.java:139)
    at org.apache.mina.core.buffer.AbstractIoBuffer.get(AbstractIoBuffer.java:591)
    at com.sohu.searchhub.http.HttpMessage.findByteArray(HttpMessage.java:37)
    at com.sohu.searchhub.http.HttpMessage.isMessageComplete(HttpMessage.java:72)
    at com.sohu.searchhub.http.HttpResponseDecoder.decodeable(HttpResponseDecoder.java:20)
    at org.apache.mina.filter.codec.demux.DemuxingProtocolDecoder.doDecode(DemuxingProtocolDecoder.java:144)
    at org.apache.mina.filter.codec.CumulativeProtocolDecoder.decode(CumulativeProtocolDecoder.java:178)
    at org.apache.mina.filter.codec.ProtocolCodecFilter.messageReceived(ProtocolCodecFilter.java:241)
    - locked <0x0000000673a53468> (a org.apache.mina.filter.codec.ProtocolCodecFilters$ProtocolDecoderOutputImpl)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain.callNextMessageReceived(DefaultIoFilterChain.java:434)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain.access$1200(DefaultIoFilterChain.java:46)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain$EntryImpl$1.messageReceived(DefaultIoFilterChain.java:796)
    at org.apache.mina.core.filterchain.IoFilterEvent.fire(IoFilterEvent.java:75)
    at org.apache.mina.core.session.IoEvent.run(IoEvent.java:63)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.runTask(OrderedThreadPoolExecutor.java:780)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.runTasks(OrderedThreadPoolExecutor.java:772)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.run(OrderedThreadPoolExecutor.java:714)
    at java.lang.Thread.run(Thread.java:745)

"pool-6-thread-27" #1231 prio=5 os_prio=0 tid=0x00007f1f79bc0000 nid=0x5f57 runnable [0x00007f24dd5d6000]
java.lang.Thread.State: RUNNABLE
    at com.yourkit.runtime.Callback.yjpMethodEntry0(Native Method)
    at com.yourkit.runtime.Callback.yjpMethodEntry(Callback.java:180)
    at java.nio.HeapByteBuffer.get(HeapByteBuffer.java:139)
    at org.apache.mina.core.buffer.AbstractIoBuffer.get(AbstractIoBuffer.java:591)
    at com.sohu.searchhub.http.HttpMessage.findByteArray(HttpMessage.java:37)
    at com.sohu.searchhub.http.HttpMessage.findByteArray(HttpMessage.java:24)
    at com.sohu.searchhub.http.HttpMessage.isMessageComplete(HttpMessage.java:61)
    at com.sohu.searchhub.http.HttpResponseDecoder.decodeable(HttpResponseDecoder.java:20)
    at org.apache.mina.filter.codec.demux.DemuxingProtocolDecoder.doDecode(DemuxingProtocolDecoder.java:144)
    at org.apache.mina.filter.codec.CumulativeProtocolDecoder.decode(CumulativeProtocolDecoder.java:178)
    at org.apache.mina.filter.codec.ProtocolCodecFilter.messageReceived(ProtocolCodecFilter.java:241)
    - locked <0x0000000672e4e6e0> (a org.apache.mina.filter.codec.ProtocolCodecFilters$ProtocolDecoderOutputImpl)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain.callNextMessageReceived(DefaultIoFilterChain.java:434)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain.access$1200(DefaultIoFilterChain.java:46)
    at org.apache.mina.core.filterchain.DefaultIoFilterChain$EntryImpl$1.messageReceived(DefaultIoFilterChain.java:796)
    at org.apache.mina.core.filterchain.IoFilterEvent.fire(IoFilterEvent.java:75)
    at org.apache.mina.core.session.IoEvent.run(IoEvent.java:63)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.runTask(OrderedThreadPoolExecutor.java:780)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.runTasks(OrderedThreadPoolExecutor.java:772)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.run(OrderedThreadPoolExecutor.java:714)
    at java.lang.Thread.run(Thread.java:745)

"pool-6-thread-26" #1230 prio=5 os_prio=0 tid=0x00007f24ec792000 nid=0x4c4c waiting on condition [0x00007f24ddbdc000]
java.lang.Thread.State: TIMED_WAITING (parking)
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <0x0000000632e76a30> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
    at java.util.concurrent.LinkedBlockingQueue.poll(LinkedBlockingQueue.java:467)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.fetchSession(OrderedThreadPoolExecutor.java:741)
    at org.apache.mina.filter.executor.OrderedThreadPoolExecutor$Worker.run(OrderedThreadPoolExecutor.java:694)
    at java.lang.Thread.run(Thread.java:745)
```



# Java内置监控工具

- jmap: 监控内存情况

1. 无任何参数，VM上载入的每个共享对象，start地址，映射大小，以及共享对象文件的完整路径

```
1 0x0000000000400000 11K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/bin/java
2 0x000000032d720000 153K /lib64/ld-2.12.so
3 0x000000032d760000 22K /lib64/libdl-2.12.so
4 0x000000032d7a0000 1881K /lib64/libc-2.12.so
5 0x000000032d7e0000 142K /lib64/libpthread-2.12.so
6 0x000000032d820000 46K /lib64/librt-2.12.so
7 0x000000032d8a0000 585K /lib64/libm-2.12.so
8 0x000000032e260000 46K /usr/lib64/libnuma.so.1
9 0x0000000351d60000 106K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/lib/amd64/jli/libjli.so
10 0x00007fcfbeab8000 89K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libnio.so
11 0x00007fcfbf2ca000 112K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libnet.so
12 0x00007fcfbf4e1000 44K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libmanagement.so
13 0x00007fcfbf6e9000 30K /usr/local/resin_video/libexec64/libresin_os.so
14 0x00007fcfbf8f1000 36K /usr/local/resin_video/libexec64/libresin.so
15 0x00007fd044e51000 120K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libzip.so
16 0x00007fd0450a1000 214K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libjava.so
17 0x00007fd0452cc000 63K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/libverify.so
18 0x00007fd0455db000 14877K /usr/lib/jvm/java-1.7.0-oracle-1.7.0.79.x86_64/jre/lib/amd64/server/libjvm.so
```



# Java内置监控工具

## 2. -finalizerinfo: 打印等待被回收对象的信息

```
[@bjzw_90_181 trunk_IntelliJ]# jmap -finalizerinfo 27321
Attaching to process ID 27321, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 24.79-b02
Number of objects pending for finalization: 0
```

## 3. -heap: 打印heap的概要信息，GC使用的算法，heap的配置及使用情况

```
Number of objects pending for finalization: 0
[@bjzw_90_181 trunk_IntelliJ]# jmap -heap 27321
Attaching to process ID 27321, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 24.79-b02

using parallel threads in the new generation.
using thread-local object allocation.
Concurrent Mark-Sweep GC

Heap Configuration:
  MinHeapFreeRatio = 40
  MaxHeapFreeRatio = 70
  MaxHeapSize      = 10737418240 (10240.0MB)
  NewSize          = 2097152000 (2000.0MB)
  MaxNewSize       = 2097152000 (2000.0MB)
  OldSize          = 5439488 (5.1875MB)
  NewRatio         = 2
  SurvivorRatio    = 1
  PermSize         = 209715200 (200.0MB)
  MaxPermSize      = 209715200 (200.0MB)
  G1HeapRegionSize = 0 (0.0MB)

Heap Usage:
New Generation (Eden + 1 Survivor Space):
  capacity = 1398145024 (1333.375MB)
  used     = 276219864 (263.42378997802734MB)
  free     = 1121925160 (1069.9512100219727MB)
  19.75616686813742% used
Eden Space:
  capacity = 699138048 (666.75MB)
  used     = 272109416 (259.5037612915039MB)
  free     = 427028632 (407.2462387084961MB)
  38.920699106337295% used
From Space:
  capacity = 699006976 (666.625MB)
  used     = 4110448 (3.9200286865234375MB)
  free     = 694896528 (662.7049713134766MB)
  0.5880410555444872% used
To Space:
  capacity = 699006976 (666.625MB)
  used     = 0 (0.0MB)
  free     = 699006976 (666.625MB)
  0.0% used
concurrent mark-sweep generation:
  capacity = 8640266240 (8240.0MB)
  used     = 5995575744 (5717.826599121094MB)
  free     = 2644690496 (2522.1734008789062MB)
  69.39109950389677% used
Perm Generation:
  capacity = 209715200 (200.0MB)
  used     = 54811576 (52.27239227294922MB)
  free     = 154903624 (147.72760772705078MB)
  26.13619613647461% used

29569 interned Strings occupying 3123632 bytes.
```

# Java内置监控工具

4. **-histo[:live]**: 打印每个class的实例数目,内存占用,类全名信息。

```
num      #instances      #bytes  class name
-----
1:        2504703      1784600392  [C
2:        413092       871722624  [B
3:        2483152       79460864  java.util.HashMap$Node
4:        1776960       71078400  java.util.HashMap$KeyIterator
5:         80643       62629936  [I
6:        2446742       58721808  java.lang.String
7:        1023817       49143216  java.util.HashMap
8:        523328       43552280  [Ljava.util.HashMap$Node;
9:        890230       40661816  [Lorg.jdom.Content;
10:       890228       35609120  org.jdom.Element
11:       995756       31864192  java.util.LinkedList$ListItr
12:       890230       28487360  org.jdom.ContentList
13:       890228       28487296  org.jdom.AttributeList
14:       827512       19860288  org.jdom.Text
15:       774426       18586224  org.jdom.CDATA
16:       754669       18112056  java.util.LinkedList$Node
17:      1020901       16334416  java.util.HashSet
18:       11073       11515920  [Lcom.sohu.vcsearch.util.Aho_Corasick.State;
19:       11072       11514880  [Lcom.sohu.vcsearch.util.Aho_Corasick.State;
20:       283696       9078272  com.sohu.vcsearch.util.Aho_Corasick.State
21:       283678       9077696  com.sohu.vcsearch.util.Aho_Corasick.State
22:       526762       8428192  java.util.HashMap$KeySet
23:       263285       6318840  com.sohu.vcsearch.util.Aho_Corasick.SparseEdgeList$Cons
24:       263269       6318456  com.sohu.vcsearch.util.Aho_Corasick.SparseEdgeList$Cons
25:       79049       4540024  [Ljava.lang.String;
26:       272623       4361968  com.sohu.vcsearch.util.Aho_Corasick.SparseEdgeList
27:       272606       4361696  com.sohu.vcsearch.util.Aho_Corasick.SparseEdgeList
28:       144725       3473400  java.util.Collections$UnmodifiableCollection$1
29:       38228       2446592  java.util.regex.Matcher
30:       45079       2163792  com.sun.tools.javac.file.ZipFileIndex$Entry
31:       84007       2016168  com.sohu.vcsearch.blacklist.BlackItem
32:       120583       1929328  org.apache.mina.transport.socket.nio.NioProcessor$IoSessionIterator
33:       108734       1739744  java.lang.Object
34:       38509       1540360  com.caucho.vfs.FilePath
35:        109       1535184  [Lcom.caucho.util.LruCache$CacheItem;
36:       11705       1514152  [Ljava.lang.Object;
37:       38646       1236672  java.io.File
38:         1       1048592  [Lcom.caucho.util.LongKeyLruCache$CacheItem;
39:       39391       945384  com.caucho.util.CharBuffer
40:       7671       857024  java.lang.Class
41:       34485       827640  com.sohu.vcsearch.blacklist.BlackItem
42:       33390       801360  java.util.ArrayList
43:       24644       788608  java.util.ArrayList$Itr
44:       17274       690960  java.lang.ref.Finalizer
45:       12072       676032  java.util.concurrent.ConcurrentHashMap$EntryIterator
46:       7400       651200  java.lang.reflect.Method
47:       23697       568728  com.caucho.env.thread.ThreadPool$ThreadNode
48:       16409       525088  java.io.FileDescriptor
49:       16385       524320  java.io.FileInputStream
50:       8322       466032  com.sohu.vcsearch.live.ChannelGuideItem
51:       8151       456456  com.sohu.vcsearch.live.ChannelGuideItem
52:       6035       434520  com.caucho.vfs.ReadStream
53:       13376       428032  java.util.concurrent.ConcurrentHashMap$Node
54:       8826       353040  java.util.LinkedHashMap$Entry
55:      10909       349088  java.util.LinkedList
```

# Java内置监控工具

5. **-dump:[live,]format=b,file=<filename>:** 将堆信息以二进制形式dump到filename文件中。若指定live, 只dump还存活的对象。配合jhat或其他工具进行内存分析

6. **-permstat:** 打印java堆持久代的classloader信息统计。在java8后不再支持

# Java 内置监控工具

- **jhat**: `jhat [ options ] <heap-dump-file>`, java堆分析工具，可以将堆中的对象以html的形式显示出来，包括对象的数量，大小等等，支持对象查询语言

[illegible]

← → ↺ 🏠 ⓘ 10.134.115.38:7000

### All Classes (excluding platform)

## Package <Arrays>

```

class |lcom.caucho.bam.mailbox.MailboxWorker; [0x5be789818]
class |lcom.caucho.bam.query.QueryManager$QueryItem; [0x5be78b750]
class |lcom.caucho.bytecode.AnnotationObject; [0x5c1ba3008]
class |lcom.caucho.bytecode.JClass; [0x5c1ba2f38]
class |lcom.caucho.bytecode.Type; [0x5c1ba2fa0]
class |lcom.caucho.cloud.topology.CloudCluster; [0x5bd5fdcd8]
class |lcom.caucho.cloud.topology.CloudPod; [0x5bde6ed40]
class |lcom.caucho.cloud.topology.CloudServer; [0x5bde74190]
class |lcom.caucho.cloud.topology.TriadOwner; [0x5c1ba3620]
class |lcom.caucho.config.inject.QualifierBinding; [0x5bd5834d0]
class |lcom.caucho.config.program.Arg; [0x5b5d7bda8]
class |lcom.caucho.config.program.BeanArg; [0x5bd57be10]
class |lcom.caucho.config.program.ConfigProgram; [0x5bd59ae90]
class |lcom.caucho.config.reflect.BaseType$ClassInfo; [0x5bd4c0858]
class |lcom.caucho.config.reflect.BaseType; [0x5bd4dbfd0]
class |lcom.caucho.db.block.Block; [0x5be7859b0]
class |lcom.caucho.db.block.BlockStore; [0x5be785ae8]
class |lcom.caucho.db.sql.Data; [0x5be789678]
class |lcom.caucho.db.sql.Expr; [0x5be789338]
class |lcom.caucho.db.sql.FromItem; [0x5be785bb8]
class |lcom.caucho.db.sql.ParamExpr; [0x5be785b50]
class |lcom.caucho.db.sql.Query$SqlInRow; [0x5be785c88]
class |lcom.caucho.db.sql.RowLiteralExpr; [0x5be7892d0]
class |lcom.caucho.db.sql.SetItem; [0x5c1ba31a8]
class |lcom.caucho.db.table.Column$ColumnType; [0x5be7895a8]
class |lcom.caucho.db.table.Column; [0x5be7893a0]
class |lcom.caucho.db.table.Constraint; [0x5be785c20]
class |lcom.caucho.db.table.TableIterator; [0x5bdab31c0]
class |lcom.caucho.distcache.AbstractCache$Persistent; [0x5bd4c50d0]
class |lcom.caucho.distcache.AbstractCache$Scope; [0x5bd4c5138]
class |lcom.caucho.el.Expr$CoeerceType; [0x5bd8b5648]
class |lcom.caucho.el.Expr; [0x5c1ba32e0]
class |lcom.caucho.el.ExpressionFactoryImpl$CoeerceType; [0x5be7856d8]
class |lcom.caucho.el.Marshall; [0x5c1ba3418]
class |lcom.caucho.env.deploy.DeployActionHandler; [0x5be04ab40]
class |lcom.caucho.env.deploy.DeployController; [0x5be04ac78]
class |lcom.caucho.env.deploy.DeployControllerApi; [0x5be04aac10]

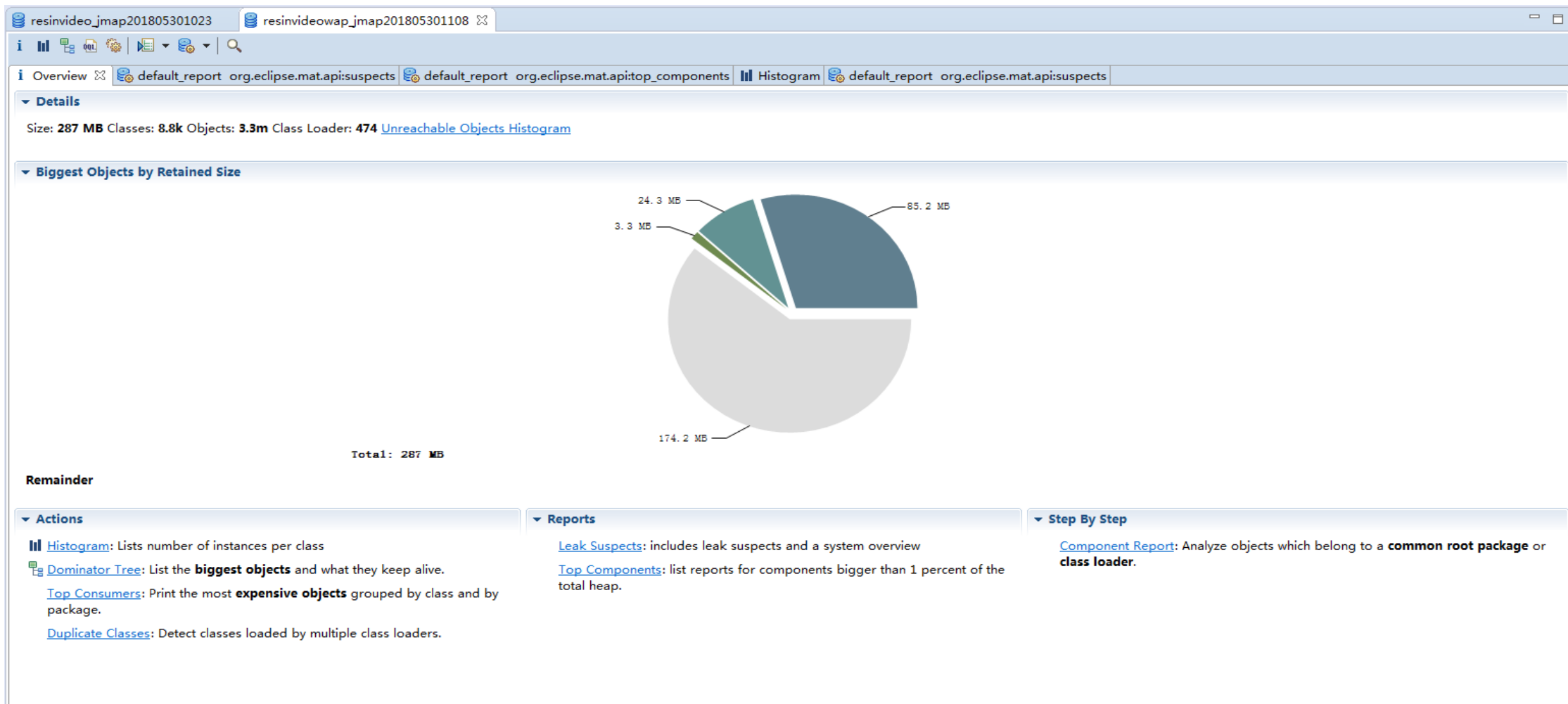
```

## Other Queries

- [All classes including platform](#)
- [Show all members of the rootset](#)
- [Show instance counts for all classes \(including platform\)](#)
- [Show instance counts for all classes \(excluding platform\)](#)
- [Show heap histogram](#)
- [Show finalizer summary](#)
- [Execute Object Query Language \(OQL\) query](#)

# Java性能分析工具

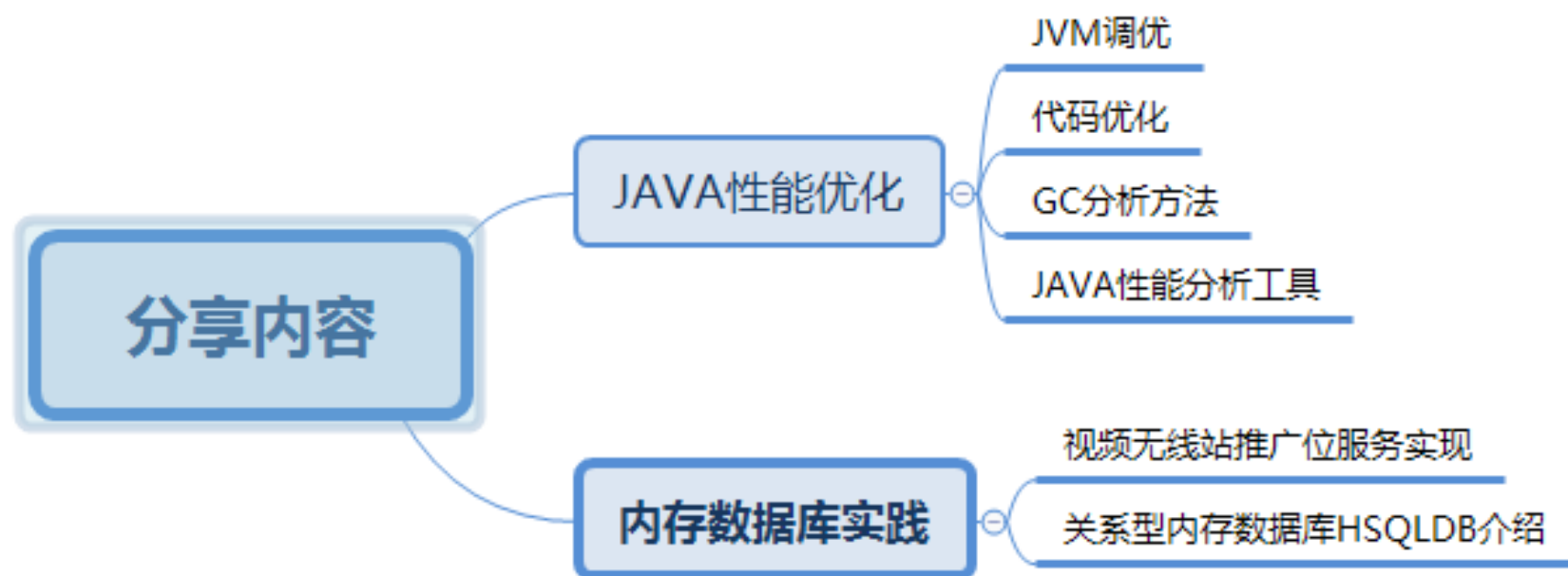
- Eclipse插件Memory Analysis Tool (MAT)



# Java性能分析工具

resinvideo_jmap201805301023 resinvideowap_jmap201805301108				
i Overview default_report org.eclipse.mat.api:suspects default_report org.eclipse.mat.api:top_components Histogram def				
Class Name	Objects	Shallow Heap	Retained Heap	
<Regex>	<Numeric>	<Numeric>	<Numeric>	
char[]	818,124	163,415,528	>= 163,415,5...	
byte[]	138,256	51,385,904	>= 51,385,904	
java.lang.String	605,457	14,530,968	>= 62,477,208	
com.caucho.server.distcache.MnodeValue	65,537	7,340,144	>= 7,340,152	
com.caucho.util.LruCache\$CacheItem	146,079	7,011,792	>= 115,388,6...	
com.caucho.server.dispatch.Invocation	65,536	6,291,456	>= 60,062,240	
com.caucho.util.CharBuffer	205,838	4,940,112	>= 35,288,448	
com.caucho.util.LruCache\$CacheItem[]	119	3,707,248	>= 3,707,248	
com.caucho.server.dispatch.FilterFilterChain	131,408	3,153,792	>= 4,733,264	
com.caucho.server.webapp.WebAppFilterChain	65,536	3,145,728	>= 7,864,208	
com.sun.tools.javac.file.ZipFileIndex\$Entry	55,315	2,655,120	>= 7,452,472	
com.caucho.server.distcache.FileCacheEntry	65,536	2,621,440	>= 17,465,256	
java.lang.Object	147,194	2,355,104	>= 2,355,104	
com.caucho.server.http.InvocationKey	65,803	2,105,696	>= 24,520,568	
java.lang.Object[]	18,877	1,681,632	>= 11,749,336	
java.util.concurrent.atomic.AtomicLong	66,634	1,599,216	>= 1,599,240	
com.caucho.server.dispatch.ServletFilterChain	65,536	1,572,864	>= 1,572,872	
java.util.concurrent.atomic.AtomicBoolean	74,887	1,198,192	>= 1,198,216	
java.nio.ByteBuffer[]	271	1,114,352	>= 1,114,352	
java.util.concurrent.atomic.AtomicReference	66,051	1,056,816	>= 8,397,088	
com.caucho.util.HashKey	65,539	1,048,624	>= 4,194,304	
java.util.HashMap\$Node	24,112	771,584	>= 3,039,448	
java.lang.reflect.Method	8,055	708,840	>= 1,459,632	
java.util.concurrent.ConcurrentHashMap\$Node	19,879	636,128	>= 3,937,008	
com.caucho.server.session.SessionImpl	8,192	589,824	>= 1,114,152	
com.caucho.util.CharBuffer[]	532	553,280	>= 23,998,024	
java.util.ArrayList	22,021	528,504	>= 4,473,088	
com.caucho.util.LongKeyLruCache\$CacheItem[]	1	524,304	>= 524,304	
java.util.HashMap\$Node[]	4,540	521,488	>= 4,428,792	
java.util.LinkedHashMap\$Entry	11,899	475,960	>= 2,100,136	

# 目录





# 视频无线站推广位服务实现

## • 背景

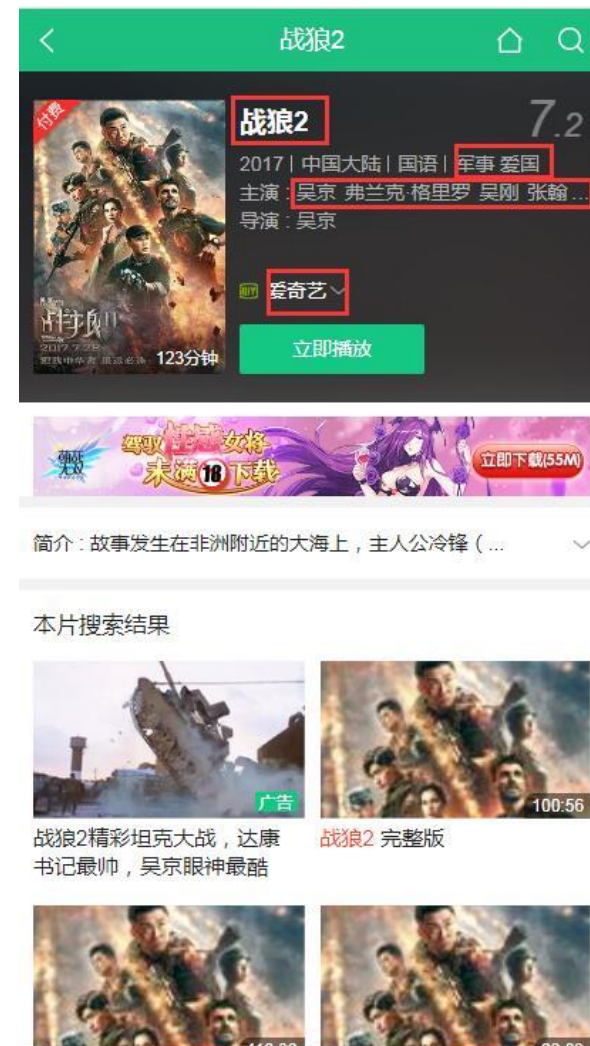
无线站内推广位由产品运营进行配置

## • 痛点

- 1、**参数多**：页面、位置、用户地域、设备等多个参数
- 2、**逻辑复杂**：优先级+权重确定本次请求展现广告
- 3、**功能独立**：与现有视频业务逻辑相对独立

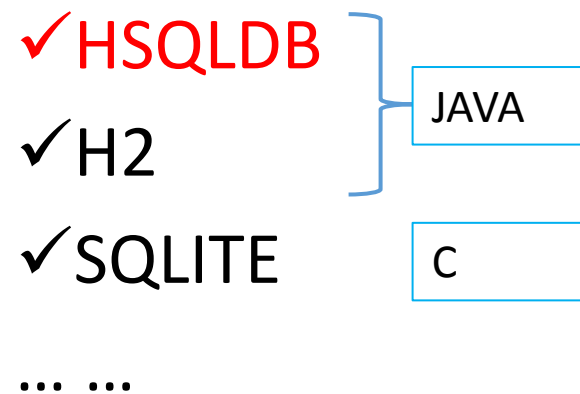
## • 解决方案

- 1、搭建**独立**resin模块承接推广位服务
- 2、关系型**内存数据库**：SQL贴合业务、内存访问的速度优势



# 视频无线站推广位服务实现

- 内存数据库方案选型



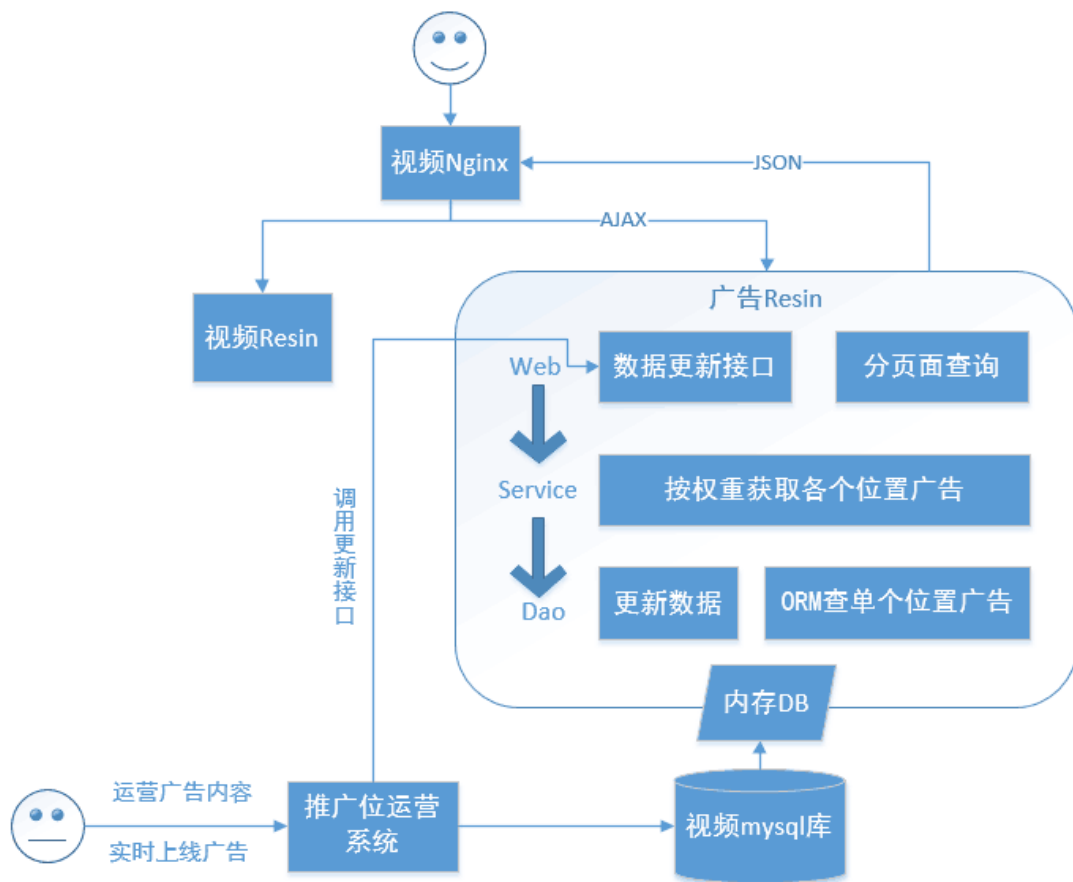
HSQLDB性能好，  
具备Hibernate原生dialect支持

Speed comparison of JPA database retrieval operations (normalized score, higher is better)

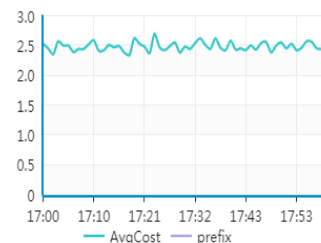
Retrieval Size =>	Few Entities		Many Entities		Average Score	
	Hibernate H2 server	Hibernate HSQLDB embedded	Hibernate H2 server	Hibernate HSQLDB embedded	Hibernate H2 server	Hibernate HSQLDB embedded
Basic Person Test	3.1	24.8	6.0	16.4	4.6	20.6
Element Collection Test	1.0	11.3	1.1	10.8	1.0	11.0
Inheritance Test	2.9	21.7	7.6	21.0	5.2	21.3
Indexing Test	2.6	11.9	7.0	21.5	4.8	16.7
Graph (Binary Tree) Test	0.42	3.6	1.1	4.7	0.78	4.1
Multithreading Test	6.5	19.8	10.5	21.5	8.5	20.6
All Tests	2.8	15.5	5.6	16.0	4.2	15.7

The results above show that in general **Hibernate with HSQLDB embedded** is much more efficient than **Hibernate with H2 server** in retrieving JPA entity objects from the database. Comparing the normalized speed of Hibernate with H2 database server (4.2) to the normalized speed of Hibernate with HSQLDB embedded database (15.7) reveals that in these tests, Hibernate with HSQLDB embedded is **3.7 times faster** than Hibernate with H2 server.

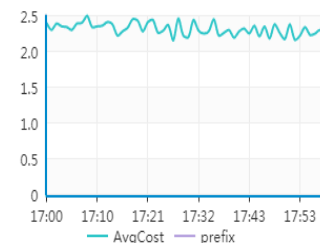
# 视频无线站推广位服务实现



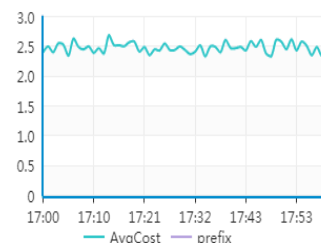
cache004.video.sjs.ted



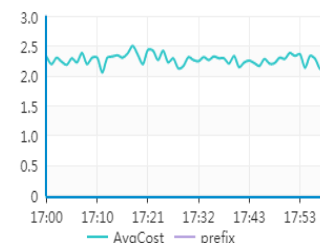
cache005.video.sjs.ted



cache009.video.sjs.ted



cache010.video.sjs.ted



cache014.video.sjs.ted

cache015.video.sjs.ted

平均cost2.5ms左右

日均请求量600w+

广告resin性能

# 内存数据库——HSQLDB

- Java编写的关系数据库管理系统，hibernate原生方言支持
- 提供基于磁盘和内存的表
- 支持嵌入模式和服务器模式运行

# HSQldb——运行模式

- 嵌入模式

- ✓ 内存：随JVM关闭数据也丢失 *`jdbc:hsqldb:mem:mymemdb`*
- ✓ 文件： *`jdbc:hsqldb:file:/opt/db/testdb`*
- ✓ Java resource： *`jdbc:hsqldb:res:org.my.path.resdb`*

- 服务器模式

- ✓ 可在应用之外被访问 *`jdbc:hsqldb:(hsql/http)://localhost/xdb`*

# HSQldb——表格类型

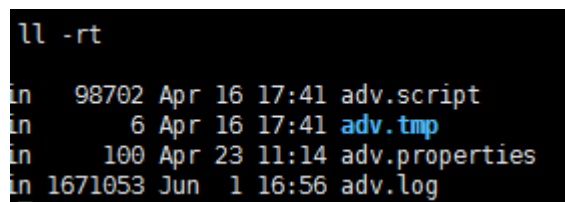
- 临时表（TEMPORARY table）

- ✓ 只存活在session的生命周期

- ✓ 有两种类型： GLOBAL TEMPORARY和LOCAL TEMPORARY

- 持久化表

- ✓ 保存结构



```
ll -rt
in  98702 Apr 16 17:41 adv.script
in   6 Apr 16 17:41  adv.tmp
in  100 Apr 23 11:14 adv.properties
in 1671053 Jun  1 16:56 adv.log
```

**Memory table:** 默认，全部数据在内存中。打开数据库时读取数据并插入，关闭时被保存

**Cached table:** 部分数据或索引在内存中。启动时较快。访问速度稍慢

**Text table:** 用csv或其他文件类型保存数据。可以对文件数据进行SQL操作

# HSQLDB——查询优化

- HSQLDB通过使用索引优化查询，支持范围和相等查询，包括 IS NULL和NOT NULL条件。
- IN 查询、AND查询和OR查询也可以使用索引



# HSQLDB ——创建和关闭数据库

- 数据库连接建立时，对应地址的空数据库会被创建
- SHUTDOWN 命令会关闭数据库，数据库不会被JDBC显式关闭
- 在连接配置中加入shutdown=true参数，表示最后一个连接关闭时数据库被关闭。

# HSQldb —— 一些常用参数

name	default	Description
ifexists	false	只有当数据库已经存在时连接。对mem:和file:数据库有影响。置为true则不会在连接时创建新库
create	true	与ifexists对应
shutdown	false	最后一个连接断开时关闭数据库
readonly	false	只读库
hsqldb.cache_rows	50000	cached table内存中的最大行数
hsqldb.cache_size	10000	cached table内存中的缓存大小，单位KB，范围100KB-4GB
hsqldb.lock_file	true	默认时，会在读写数据库时创建一个.lock文件，设为false则不会创建该文件
hsqldb.log_data	true	记录数据变化。当数据库意外崩溃后的恢复不是必要，可置为false
hsqldb.log_size	50	checkpoint发生的log文件大小，单位为MB。checkpoint会重写.script文件并清除.log文件

# HSQldb —— 一些常用参数

name	default	description
hsqldb.defrag_limit	0	checkpoint发生时，.data文件的浪费空间会被计算。若浪费空间超过限制，会进行磁盘碎片整理。范围0-100，表示.data文件的百分比
hsqldb.script_format	0	压缩.script文件，如果置为3，则.script文件以压缩格式存储，压缩格式的文件不可读
hsqldb.write_delay	true	设置为true，则延迟执行log文件的fsync写操作
hsqldb.write_delay_millis	500	延迟执行log文件的fsync写操作的时间

- 数据量不大，性能要求较高的场景

谢谢！

