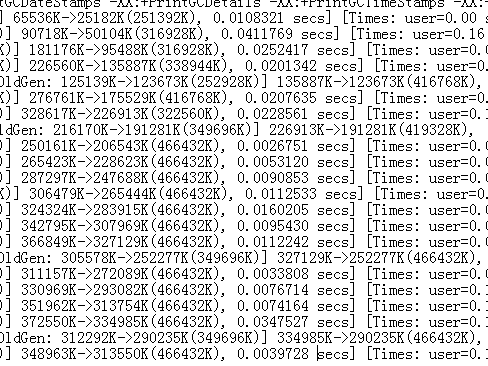
1. 串行GC

Xms XmX配置大小相同的情况下，随着其配置的增大GC会出现以下特征：

1. 初始YGC与初始FGC会延后，这主要是由于堆空间的扩大导致其占满时刻被延后
2. 出现OOM的时刻被延后
3. 并行GC
4. 在Xms Xmx 均为128M的情况下，并行YGC有10次，串行YGC有8次，在并行GC线程数配置合理的前提下，并行GC相比串行GC可以提升一定的性能
5. 在FGC的时间计时中出现了[user:0.11 sys:0.00 real:0.01]实际耗时小于用户态+内核态的耗时，在没有设置并行GC线程数的情况下，利用java -XX:+PrintFlagsFinal –version命令找到并行线程参数XX:ParallelGCThread，发现与CPU内核数相同为8，所以系统实际耗时（0.11+0.00）/ 8 = 0.01375≈0.01与日志记录相符
6. 在Xms Xmx均为128M的情况下，运行出现了OOM，另外在运行多次FGC后，老年代空间不足，无法容纳新生代的对象，所以FGC无法使新生代的空间归0
7. 如果不配置Xms,当前系统内存为16GB，初始堆大小为16GB/64约为256MB，从日志可以看出堆内存占用量不断上升



1. CMS
2. CMS算法中，Initial Mark与Final Remark阶段均会STW
3. CMSInitiatingOccupancyFraction控制触发第一次CMS回收时老年代空间的占用率，默认为92%；UseCMSInitiatingOccupancyOnly参数如果为false，除第一次CMS回收，后续回收的空间占用率则自动调整
4. Xmx Xms设置为512M时，在并发清理阶段，出现了Concurrent Mode Failure错误，表示并发模式失败，说明在CMS阶段老年代的空间不足以容纳应用产生的垃圾。此时CMS回收器将退化至serial-old 回收器，从图中可见消耗用时用了0.07s，相比并行GC，从日志上看，FGC回收耗时用了0.04s，退化程序较严重。



1. 设置为4G后，程序进行了3次YGC，相同Xms Xmx参数的并行GC只触发了1次YGC，区别原因在于两种回收策略默认的新生代大小不同：CMS 中MaxNewSize 与NewSize均为697933824，而并行GC中新生代大小为4G/3，大于CMS新生代大小，所以YGC频率较小。