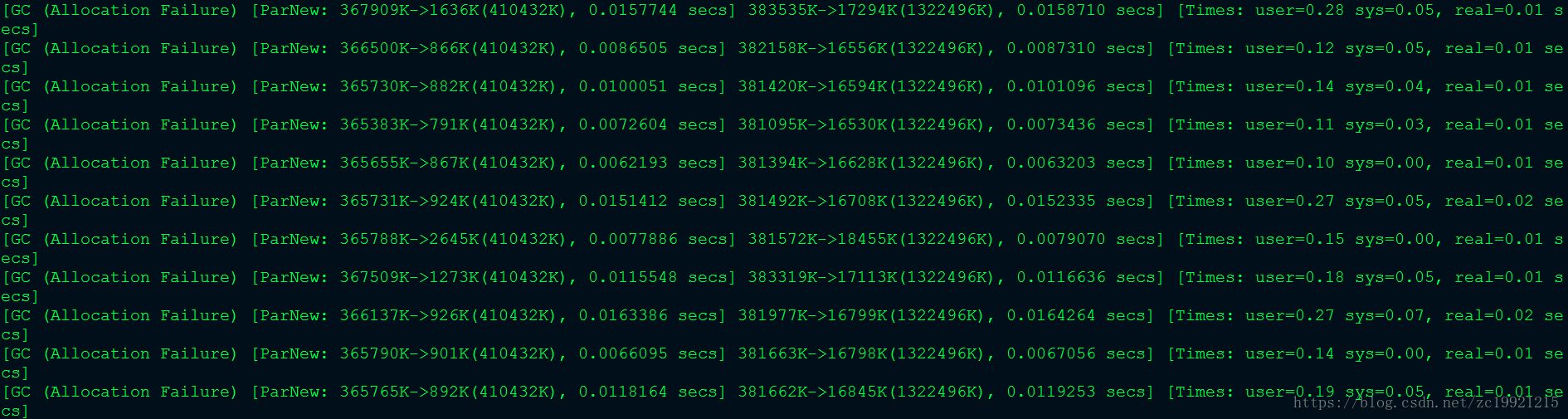
新生代GC

日前查看某个程序的日志，发现一直在报GC相关的信息，不确定这样的信息是代表正确还是不正确，所以正好借此机会再复习下GC相关的内容：



以其中一行为例来解读下日志信息：

[GC (Allocation Failure) [ParNew: 367523K->1293K(410432K), 0.0023988 secs] 522739K->156516K(1322496K), 0.0025301 secs] [Times: user=0.04 sys=0.00, real=0.01 secs]

GC：

表明进行了一次垃圾回收，前面没有Full修饰，表明这是一次Minor GC ,注意它不表示只GC新生代，并且现有的不管是新生代还是老年代都会STW。

Allocation Failure：

表明本次引起GC的原因是因为在年轻代中没有足够的空间能够存储新的数据了。

ParNew：

表明本次GC发生在年轻代并且使用的是ParNew垃圾收集器。ParNew是一个Serial收集器的多线程版本，会使用多个CPU和线程完成垃圾收集工作（默认使用的线程数和CPU数相同，可以使用-XX：ParallelGCThreads参数限制）。该收集器采用复制算法回收内存，期间会停止其他工作线程，即Stop The World。

367523K->1293K(410432K)：单位是KB

三个参数分别为：GC前该内存区域(这里是年轻代)使用容量，GC后该内存区域使用容量，该内存区域总容量。

0.0023988 secs：

该内存区域GC耗时，单位是秒

522739K->156516K(1322496K)：

三个参数分别为：堆区垃圾回收前的大小，堆区垃圾回收后的大小，堆区总大小。

0.0025301 secs：

该内存区域GC耗时，单位是秒

[Times: user=0.04 sys=0.00, real=0.01 secs]：

分别表示用户态耗时，内核态耗时和总耗时

分析下可以得出结论：

该次GC新生代减少了367523-1293=366239K

Heap区总共减少了522739-156516=366223K

366239 – 366223 =16K，说明该次共有16K内存从年轻代移到了老年代，可以看出来数量并不多，说明都是生命周期短的对象，只是这种对象有很多。

我们需要的是尽量避免Full GC的发生，让对象尽可能的在年轻代就回收掉，所以这里可以稍微增加一点年轻代的大小，让那17K的数据也保存在年轻代中。

老年代GC

249.893: [GC (Allocation Failure) 249.893: [DefNew: 78656K->78656K(78656K), 0.0000398 secs]249.893: [Tenured: 160389K->174783K(174784K), 0.7500708 secs] 239045K->186968K(253440K), [Metaspace: 42704K->42704K(44416K)], 0.7502810 secs] [Times: user=0.73 sys=0.02, real=0.75 secs]

Tenured: 160389K->174783K(174784K), 0.7500708 secs

三个参数分别为：GC前该内存区域(这里是老年代)使用容量，GC后该内存区域使用容量，该内存区域总容量。

参考：

https://blog.csdn.net/zc19921215/article/details/83029952

https://segmentfault.com/a/1190000013509330