Week2 作業-GC總結

第 3 课作业实践

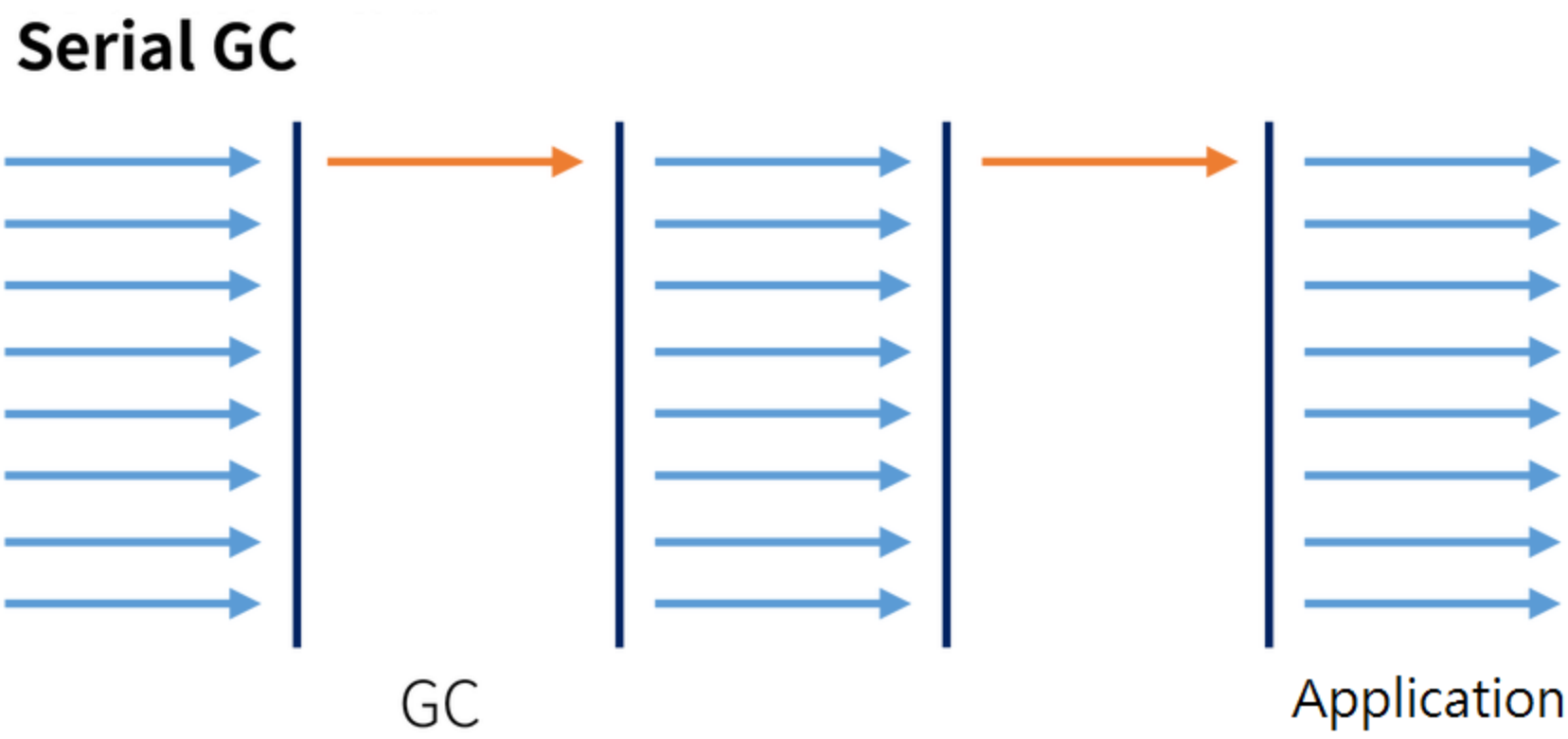
1、使用 GCLogAnalysis.java 自己演练一遍串行/并行/CMS/G1的案例。
2、使用压测工具（wrk或sb），演练gateway-server-0.0.1-SNAPSHOT.jar示例。
3、(选做)如果自己本地有可以运行的项目，可以按照2的方式进行演练。

根据上述自己对于1和2的演示，写一段对于不同GC的总结，提交到github。

Serial GC

特點 ：

1. Young Gen : `mark-copy` 算法 ; Old Gen : `mark-sweep-compact` 算法
2. 兩者都是 單線程GC，都會觸發STW(停止所有應用線程)
3. 不適合做服務器



指令 ：

```
java -XX:+UseSerialGC -Xms512m -Xmx512m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps demo/jvm0204/GCLogAnalysis
```

日誌內容 ：

```
Java HotSpot(TM) 64-Bit Server VM (25.241-b07) for bsd-amd64 JRE (1.8.0_241-b07), built on Dec 11 2019 02:29:59 by "java_re"
```

```
with gcc 4.2.1 (Based on Apple Inc. build 5658)
(LLVM build 2336.11.00)Memory: 4k page, physical 16777216k(1138280k free)



CommandLine flags:
-XX:InitialHeapSize=536870912 -XX:MaxHeapSize=536870912
-XX:+PrintGC -XX:+PrintGCDateStamps -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+UseCompressedClassPointers
-XX:+UseCompressedOops -XX:+UseSerialGC

2021-01-14T22:24:56.781-0800: 0.152:
[GC (Allocation Failure)
2021-01-14T22:24:56.781-0800: 0.152:
[DefNew: 139230K->17471K(157248K), 0.0287046 secs]
139230K->47874K(506816K),
 0.0288280 secs]
[Times: user=0.01 sys=0.01, real=0.03 secs]

...

2021-01-14T22:24:57.676-0800: 1.046:
[GC (Allocation Failure) 2021-01-14T22:24:57.676-0800: 1.046:
[DefNew: 139776K->139776K(157248K), 0.0000197 secs]
2021-01-14T22:24:57.676-0800: 1.046:
[Tenured: 333127K->347113K(349568K), 0.0453274 secs]
472903K->347113K(506816K),
[Metaspace: 2591K->2591K(1056768K)],
 0.0454504 secs]
[Times: user=0.05 sys=0.00, real=0.05 secs]


Heap
 def new generation   total 157248K, used 6147K [0x00000007a0000000, 0x00000007aaaa0000, 0x00000007aaaa0000)
  eden space 139776K,   4% used [0x00000007a0000000, 0x00000007a0600c50, 0x00000007a8880000)
  from space 17472K,   0% used [0x00000007a9990000, 0x00000007a9990000, 0x00000007aaaa0000)
  to   space 17472K,   0% used [0x00000007a8880000, 0x00000007a8880000, 0x00000007a9990000)
 tenured generation   total 349568K, used 347113K [0x00000007aaaa0000, 0x00000007c0000000, 0x00000007c0000000)
   the space 349568K,  99% used [0x00000007aaaa0000, 0x00000007bfd9a7a8, 0x00000007bfd9a800, 0x00000007c0000000)
 Metaspace       used 2598K, capacity 4486K, committed 4864K, reserved 1056768K
  class space    used 279K, capacity 386K, committed 512K, reserved 1048576K
```

Minor GC日誌分析

```
2021-01-14T22:24:56.781-0800: 0.152:
[GC (Allocation Failure)
2021-01-14T22:24:56.781-0800: 0.152:
[DefNew: 139230K->17471K(157248K), 0.0287046 secs]
139230K->47874K(506816K),
0.0288280 secs]
[Times: user=0.01 sys=0.01, real=0.03 secs]
```

日誌解析 ：

- `DefNew` ： 表示 - 年輕代使用單線程、標記-複製、STW 垃圾收集器。
- `139230K → 17471K` GC之前和之後使用量，總空間 `157248` (Young Gen)
- `139230K → 47874K` GC之前和之後使用量，總空間 `506816` (Heap)
- `0.0288280 secs` ： GC 事件持續時間
- `[Times: user=0.01 sys=0.01, real=0.03 secs]` 此次GC事件持續時間。
  `user` ： GC線程消耗的CPU時間
  `sys` ： 系統調用和系統等待事件消耗時間
  `real` ： 程式暫停的時間
  – 因串行垃圾收集器(Serial Garbage Collector)使用單線程，所以 `real = user + sys`，0.03秒(30毫秒)

分析 ：

- GC前後，年輕代使用量 `139230K → 17471K` ，減少 `121,919K`
- 堆內存總量 `139230K → 47874K` ，減少 `91,356 K`
- 老年代提升到老年代占用了 `121919K - 91356K = 30574K` 空間

- GC後，老年代使用量 `47874K - 17471K  = 30403K`

Full GC 日誌分析

```
2021-01-14T22:24:57.676-0800: 1.046:
[GC (Allocation Failure) 2021-01-14T22:24:57.676-0800: 1.046:
[DefNew: 139776K->139776K(157248K), 0.0000197 secs]
2021-01-14T22:24:57.676-0800: 1.046:
[Tenured: 333127K->347113K(349568K), 0.0453274 secs]
472903K->347113K(506816K),
[Metaspace: 2591K->2591K(1056768K)],
 0.0454504 secs]
[Times: user=0.05 sys=0.00, real=0.05 secs]
```

日誌解析 ：

1. `[DefNew: 139776K->139776K(157248K), 0.0000197 secs]` ，GC回收時間 0.0000197
2. `Tenured` 清理老年代GC名稱，Tenured 表示單線程的STW 垃圾回收器，算法為 `mark-sweep-compact`。
3. `[Tenured: 333127K->347113K(349568K), 0.0453274 secs]` ： `333127K->347113K(349568K)` GC前後老年代的使用量，及空間大小。 `0.0453274 secs` 清理老年代所花的時間。
4. `472903K->347113K(506816K)` ： GC前後整個 Heap 使用狀況
5. `[Metaspace: 2591K->2591K(1056768K)], 0.0454504 secs]`
6. `[Times: user=0.05 sys=0.00，real=0.05 secs]`， GC持續時間，串行GC只能使用單線程，real = user + system 。 50毫秒的時間暫停。

分析 ：

1. GC後老年代使用率 ： `347113K` / `349568K` = 9%

▼ 為何比起前面年輕代的 GC 快增加一倍 ？

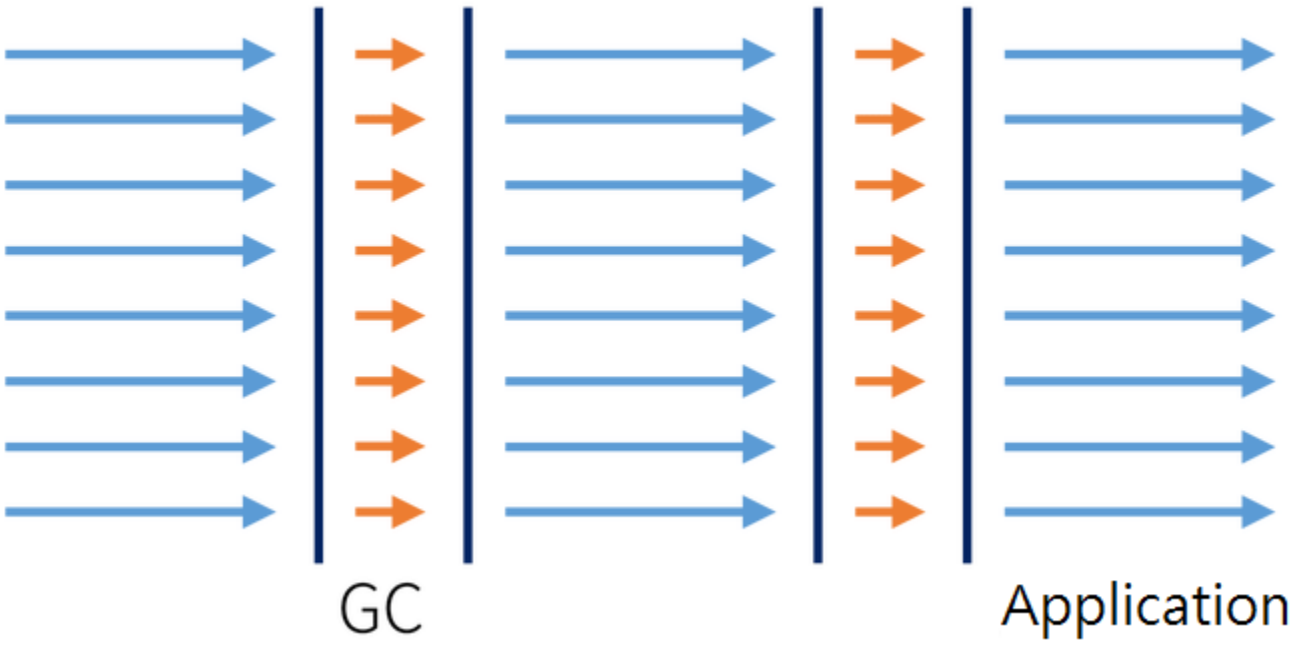
GC時間，與GC後存的對象總數量關係最大。

Parallel GC

特性 ：

1. Young Gen : `mark-copy` 算法 ; Old Gen : `mark-sweep-compact` 算法
2. 年輕和老年代回收都會觸發STW，暫停所有線程，執行GC
3. 跑 `mark-copy` 、 `mark-sweep-compact` 階段都是多線程 ，稱 "Rarallel"。通過多個GC線程並行執行方式，能使JVM在多CPU平台上的GC時間大幅減少。

參數 -XX:ParallelGCThreads=X 可設定GC線程數量，默認為CPU內核數。



指令 ：

```
java -XX:+UseParallelGC -Xms512m -Xmx512m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps demo/jvm0204/G
CLogAnalysis
```

日誌 ：

```
Java HotSpot(TM) 64-Bit Server VM (25.241-b07) for bsd-amd64 JRE (1.8.0_241-b07), built on Dec 11 2019 02:29:59 by "jav
a_re" with gcc 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)
Memory: 4k page, physical 16777216k(497952k free)


CommandLine flags:
-XX:InitialHeapSize=536870912 -XX:MaxHeapSize=536870912
-XX:+PrintGC -XX:+PrintGCDateStamps -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+UseCompressedClassPointers
-XX:+UseCompressedOops -XX:+UseParallelGC


2021-01-15T08:16:35.498-0800: 0.160:
[GC (Allocation Failure)
[PSYoungGen: 131584K->21497K(153088K)]
131584K->44219K(502784K), 0.0153968 secs]
[Times: user=0.02 sys=0.07, real=0.02 secs]


# TODO 為何 Heap總數會變動
2021-01-15T08:16:35.676-0800: 0.338:
[GC (Allocation Failure)
[PSYoungGen: 153054K->21495K(80384K)]
344999K->252182K(430080K), 0.0158088 secs]
[Times: user=0.04 sys=0.06, real=0.02 secs]



2021-01-15T08:16:36.412-0800: 1.074:
[Full GC (Ergonomics)
[PSYoungGen: 60416K->0K(116736K)]
[ParOldGen: 329230K->332627K(349696K)] 389646K->332627K(466432K),
[Metaspace: 2591K->2591K(1056768K)], 0.0357423 secs]
[Times: user=0.26 sys=0.00, real=0.03 secs]


Heap
 PSYoungGen      total 116736K, used 2695K [0x00000007b5580000, 0x00000007c0000000, 0x00000007c0000000)
  eden space 60416K, 4% used [0x00000007b5580000,0x00000007b5821d08,0x00000007b9080000)
  from space 56320K, 0% used [0x00000007b9080000,0x00000007b9080000,0x00000007bc780000)
  to   space 57856K, 0% used [0x00000007bc780000,0x00000007bc780000,0x00000007c0000000)
 ParOldGen       total 349696K, used 332627K [0x00000007a0000000, 0x00000007b5580000, 0x00000007b5580000)
  object space 349696K, 95% used [0x00000007a0000000,0x00000007b44d4ef0,0x00000007b5580000)
 Metaspace       used 2598K, capacity 4486K, committed 4864K, reserved 1056768K
  class space    used 279K, capacity 386K, committed 512K, reserved 1048576K
```

Minor GC日誌

```
2021-01-15T08:16:35.498-0800: 0.160:
[GC (Allocation Failure)
[PSYoungGen: 131584K->21497K(153088K)]
131584K->44219K(502784K), 0.0153968 secs]
[Times: user=0.02 sys=0.07, real=0.02 secs]
```

分析 ：

1. [PSYoungGen: 131584K->21497K(153088K)] - GC後年輕代使用率 21497K/153088K =14%
2. 131584K→44219K(502784K) - GC後Heap 使用率為 44219K / 502784K = 8%
3. [Times: user=0.02 sys=0.07, real=0.02 secs] - 因為並不是所有操作過程都能全部並行，所以在 Parallel GC中， real 約等於 user + sys / GC 線程數 ，測試電腦為四核，默認為4個GC線程數。

解析 ：

1. 透過分析 3 會發現如果使用 Serial GC可能會暫停 90毫秒，但ParallelGC只暫停20毫秒，性能有大幅提升。
2. 老年使用量 153088K-31584K = 121504K
3. 此次Young Gen使用量減少 131584K - 21497K = 110087K ； 總的內存使用量減少 131584K→44219K=87365K

Full GC日誌

```
2021-01-15T08:16:36.412-0800: 1.074:
[Full GC (Ergonomics)
[PSYoungGen: 60416K->0K(116736K)]
[ParOldGen: 329230K->332627K(349696K)]
389646K->332627K(466432K),
[Metaspace: 2591K->2591K(1056768K)],
0.0357423 secs]
[Times: user=0.26 sys=0.00, real=0.03 secs]
```

解析:

1. Full GC ： 本次GC清理年輕代和老年代，Ergonomics 觸發GC原因，表示JVM內部環境認為此時可以進行一次GC。
2. [PSYoungGen: 60416K->0K(116736K)] ： 採用 mark-copy ，60416K → 0 K 一般 Full GC 中年輕代都是這樣。
3. [ParOldGen: 329230K->332627K(349696K)] ： 算法 ： mark-sweep-compact ，老年代使用率 329230K/349696K = 94% ； GC後使用率 332627K/349696K = 95%
4. 389646K->332627K(466432K) :
5. [Metaspace: 2591K->2591K(1056768K)] ： 發現Meta區沒有被回收。

總結 ：

1. 解析3會發現GC後使用率反而上升，当前要晋升old gen的对象大于目前old gen剩馀的空间，所以这边才触发FullGC，而不是YoungGC。老年代里的对象几乎没有被回收掉，但新生代又要晋升活对象上来，导致ＧＣ后使用量就上升。

   ```
   2021-01-15T08:16:36.270-0800: 0.932:
   [GC (Allocation Failure) --
   [PSYoungGen: 102631K->102631K(116736K)]
   415937K->452250K(466432K), 0.0099321 secs]
   ```

```
[Times: user=0.04 sys=0.03, real=0.01 secs]
(前一次 MinorGC，因老年代使用量達到很高，所以觸發Full GC)
```

2. Full GC要更關注老年代的使用量有沒有下降，如果GC後內存不怎麼下降，使用率還增高，代表系統有問題了。

CMS

特性：

1. 名稱為 "Mostly Concurrent Mark and Sweep Garbage Collector" (最大併發-標記-清除-垃圾收集器)
2. 年輕代 `mark-copy (標記-複製)` ； 老年代主要使用並發 `mark-sweep(標記-清除)`
3. 默認使用的併發線程數等於CPU內核數的 `1/4`
4. 針對老年代進行回收，不回收年輕代
5. 每次 GC 時間較短

指令 ：

```
java -XX:+UseConcMarkSweepGC -Xms512m -Xmx512m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps demo/jvm0204/GCLogAnalysis
```

日誌 ：

```
Java HotSpot(TM) 64-Bit Server VM (25.241-b07) for bsd-amd64 JRE (1.8.0_241-b07), built on Dec 11 2019 02:29:59 by "java_re" with gcc 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)
Memory: 4k page, physical 16777216k(1552176k free)

CommandLine flags:
-XX:InitialHeapSize=536870912 -XX:MaxHeapSize=536870912
-XX:MaxNewSize=178958336 -XX:MaxTenuringThreshold=6
-XX:NewSize=178958336 -XX:OldPLABSize=16
-XX:OldSize=357912576 -XX:+PrintGC
-XX:+PrintGCDateStamps -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+UseCompressedClassPointers
-XX:+UseCompressedOops -XX:+UseConcMarkSweepGC -XX:+UseParNewGC


2021-01-16T16:09:57.686-0800: 0.856:
[GC (Allocation Failure)
  2021-01-16T16:09:57.686-0800: 0.856:
  [ParNew: 139776K->17470K(157248K), 0.0188443 secs]
  449293K->360823K(506816K), 0.0189319 secs]
[Times: user=0.14 sys=0.02, real=0.02 secs]


2021-01-16T16:09:57.705-0800: 0.875:
[GC (CMS Initial Mark)
  [1 CMS-initial-mark: 343352K(349568K)]
  363742K(506816K), 0.0001870 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.705-0800: 0.876:
  [CMS-concurrent-mark-start]
2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-mark: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-preclean: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.878:
```

```
  [CMS-concurrent-abortable-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-abortable-preclean: 0.000/0.000 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.878:
  [GC (CMS Final Remark)
    [YG occupancy: 28068 K (157248 K)]
    2021-01-16T16:09:57.707-0800: 0.878:
      [Rescan (parallel) , 0.0003623 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [weak refs processing, 0.0000174 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [class unloading, 0.0002188 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub symbol table, 0.0003117 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub string table, 0.0002084 secs]
      [1 CMS-remark: 343352K(349568K)]
    371420K(506816K), 0.0011940 secs]
[Times: user=0.01 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.709-0800: 0.879:
[CMS-concurrent-sweep-start]
2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-sweep: 0.001/0.001 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-reset-start]
2021-01-16T16:09:57.710-0800: 0.880:
[CMS-concurrent-reset: 0.000/0.000 secs]
[Times: user=0.00 sys=0.00, real=0.01 secs]



Heap
 par new generation   total 157248K, used 5613K [0x00000007a0000000, 0x00000007aaaa0000, 0x00000007aaaa0000)
  eden space 139776K,   4% used [0x00000007a0000000, 0x00000007a057b7e8, 0x00000007a8880000)
  from space 17472K,   0% used [0x00000007a9990000, 0x00000007a9990000, 0x00000007aaaa0000)
  to   space 17472K,   0% used [0x00000007a8880000, 0x00000007a8880000, 0x00000007a9990000)
 concurrent mark-sweep generation total 349568K, used 335632K [0x00000007aaaa0000, 0x00000007c0000000, 0x00000007c00
00000)
 Metaspace       used 2598K, capacity 4486K, committed 4864K, reserved 1056768K
  class space    used 279K, capacity 386K, committed 512K, reserved 1048576K
```

MinorGC

日誌:

```
2021-01-16T16:09:57.686-0800: 0.856:
[GC (Allocation Failure)
  2021-01-16T16:09:57.686-0800: 0.856:
  [ParNew: 139776K->17470K(157248K), 0.0188443 secs]
  449293K->360823K(506816K), 0.0189319 secs]
  [Times: user=0.14 sys=0.02, real=0.02 secs]
```

解析:

1. GC (Allocation Failure) : 小型GC; Allocation Failure 表示觸發 GC的原因為年輕代可用空間不足，新對象內存分配失敗。
2. `[ParNew: 139776K->17470K(157248K), 0.0188443 secs]` : ParNew 為GC名稱，表示在年輕代使用: mark-copy 垃圾收集器，
3. 449293K->360823K(506816K), 0.0189319 secs] : Heap內存使用量
4. [Times: user=0.14 sys=0.02, real=0.02 secs] : 應用程式實際暫停時間 `real ~= (user + sys) / GC線程數`

分析 :

1. 年輕代使用量為 `139776K / 157248K =  88.8%` ; 堆內存使用量為 449293K/506816K=88.6%。

2. 老年代使用率 ： (449293K - 139776K) / (506816K-157248K) =(309517/349,568) = 88.5%
3. GC後，年輕代使用量為 17470K ~= 11% ，下降 122306K，堆內存使用量 360823K ~= 71% ，下降 88470K，年輕代提升到老年代的內存量(兩個下降值相減) ： 122306K - 88470K = 33836K
4. 老年代空間 ： 506816K - 157248K = 349568K; 老年代使用率 ： 343353K/349568K = 98%

FullGC

日誌 ：

```
2021-01-16T16:09:57.705-0800: 0.875:
[GC (CMS Initial Mark)
  [1 CMS-initial-mark: 343352K(349568K)]
  363742K(506816K), 0.0001870 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.705-0800: 0.876:
  [CMS-concurrent-mark-start]
2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-mark: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-preclean: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-abortable-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-abortable-preclean: 0.000/0.000 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.707-0800: 0.878:
  [GC (CMS Final Remark)
    [YG occupancy: 28068 K (157248 K)]
    2021-01-16T16:09:57.707-0800: 0.878:
      [Rescan (parallel) , 0.0003623 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [weak refs processing, 0.0000174 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [class unloading, 0.0002188 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub symbol table, 0.0003117 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub string table, 0.0002084 secs]
      [1 CMS-remark: 343352K(349568K)]
    371420K(506816K), 0.0011940 secs]
[Times: user=0.01 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.709-0800: 0.879:
[CMS-concurrent-sweep-start]
2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-sweep: 0.001/0.001 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]


2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-reset-start]
2021-01-16T16:09:57.710-0800: 0.880:
[CMS-concurrent-reset: 0.000/0.000 secs]
[Times: user=0.00 sys=0.00, real=0.01 secs]
```

流程階段 ：

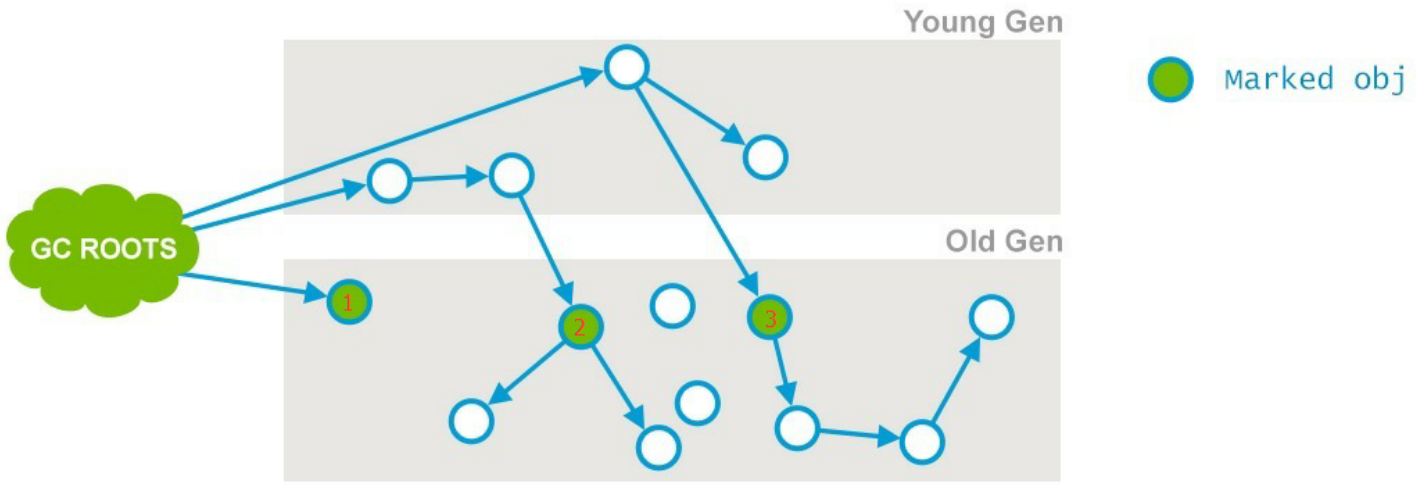
1. Initial Mark(初始標記)
2. Concurrent Mark(併發標記)
3. Concurrent Preclean(並發預先清理)

4. Concurrent Abortable Preclean(可取消的併發預清理)
5. Final Remark(最終標記)
6. Concurrent Sweep(併發清除)
7. **Concurrent Reset(併發重置)**

Initial Mark(初始標記)

該階段會使用 `STW` 暫停，標記所有存活對象。

1. 標記老年代所有`GC Roots`對象
2. 標記年輕代中活著的對象引用到的老年代的對象



日誌 ：

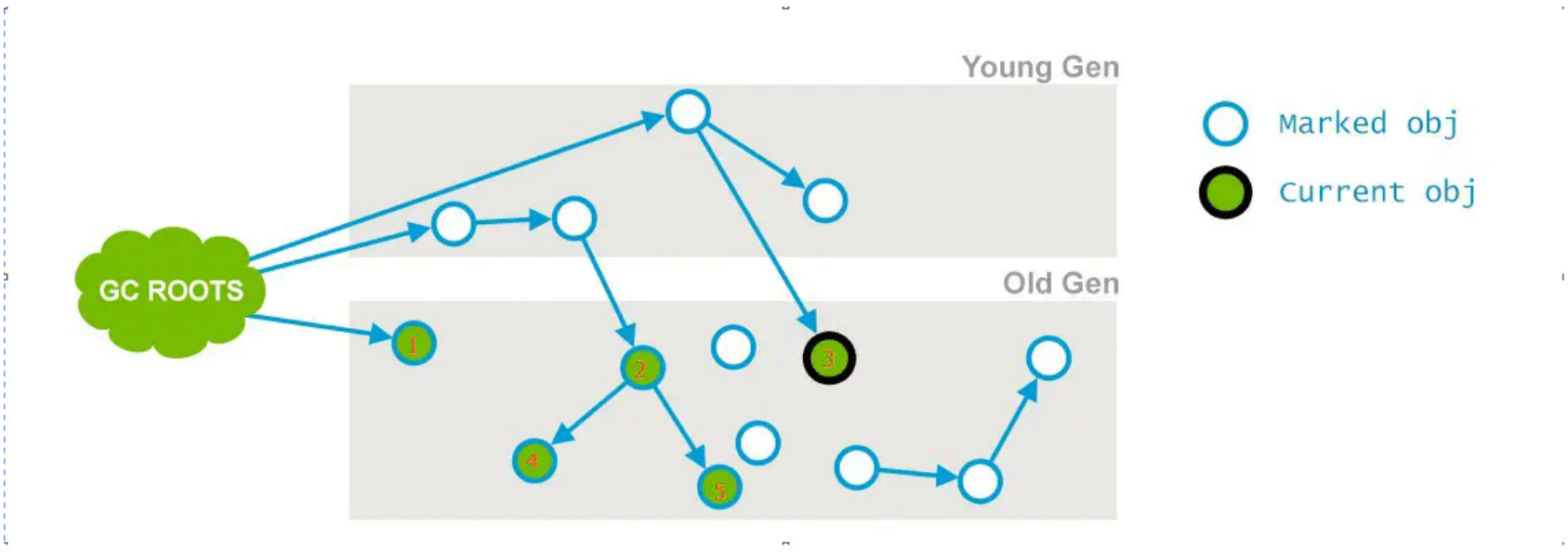
```
2021-01-16T16:09:57.705-0800: 0.875:
[GC (CMS Initial Mark)
[1 CMS-initial-mark: 343352K(349568K)]
  363742K(506816K), 0.0001870 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]
```

解析 ：

1. CMS Initial Mark ： 該階段標示所有 "GC Root"
2. [1 CMS-initial-mark: 343352K(349568K)] ： 老年代使用量及總空間
3. 363742K(506816K), 0.0001870 secs ： 內存使用量、總大小及消耗時間。
4. [Times: user=0.00 sys=0.00, real=0.00 secs] ： 暫停時間

Concurrent Mark(併發標記)

1. 該階段併發運行
2. 運行期間會發生年輕代的對象晉升到老年代、或者是直接在老年代分配對象、或者更新老年代對向的飲用關係等。
   → 這些對象所在的 Card 標示為 `Dirty`



日誌 ：

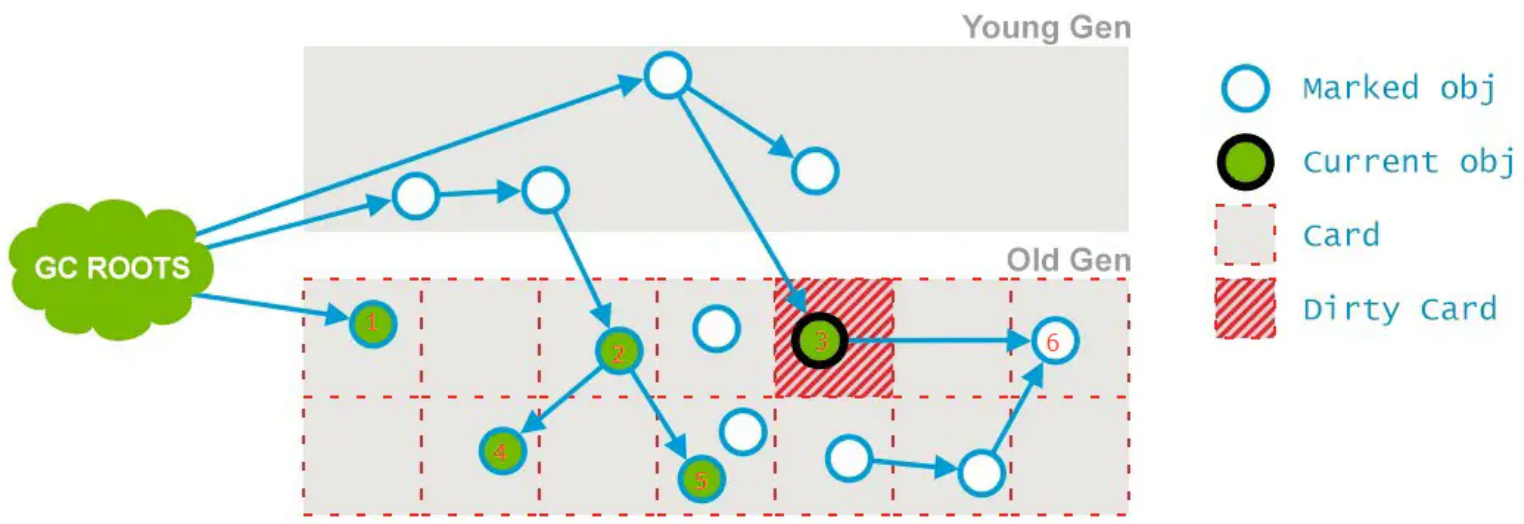
```
2021-01-16T16:09:57.705-0800: 0.876:
  [CMS-concurrent-mark-start]
2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-mark: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]
```

解析 ：

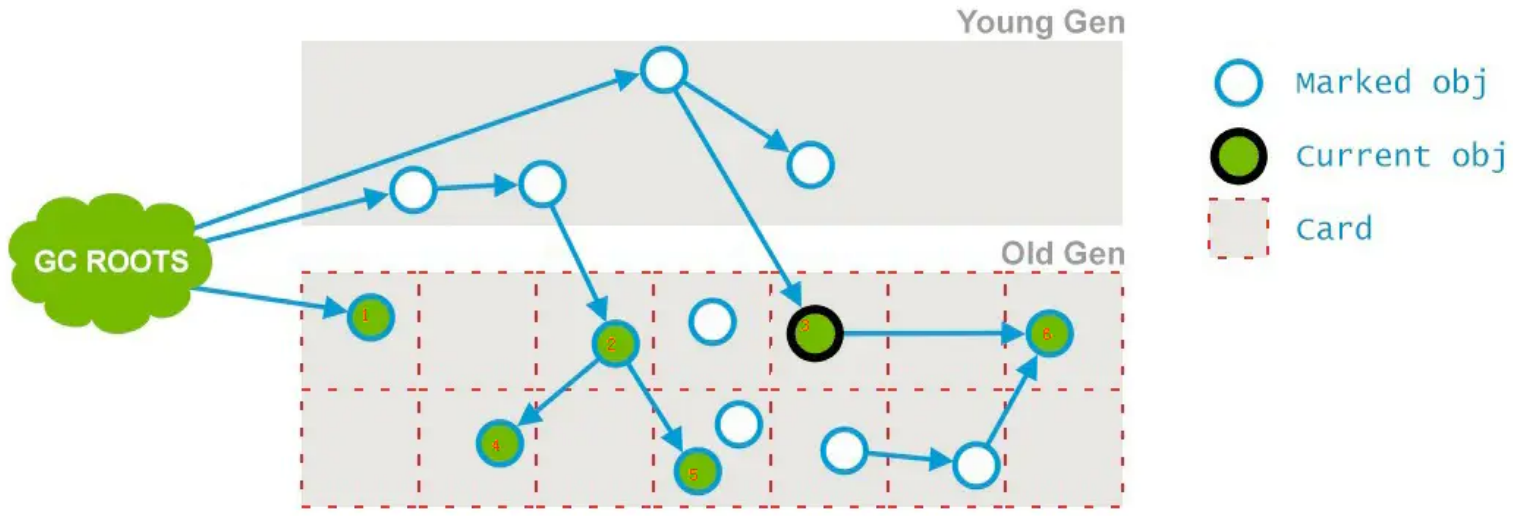
1. CMS-concurrent-mark ： 併發標記
2. 0.001/0.001 secs ： GC線程消耗時間 和 實際消耗時間
3. [Times: user=0.00 sys=0.00, real=0.00 secs] - 併發大概執行時間

Concurrent Preclean(並發預先清理)

該階段處理上一階段因引用關係改變導致沒有標記到的存活對象。這邊會掃描所有邊際為Dirty 的Card。



在併發清理階段，節點3 指向 6，會把節點3Card 設為 Dirty。



最後將 6標記為存活

日誌 ：

```
2021-01-16T16:09:57.707-0800: 0.877:
  [CMS-concurrent-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-preclean: 0.001/0.001 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]
```

解析 ：

1. CMS-concurrent-preclean ： 並發預先清理
2. 0.001/0.001 secs - GC線程運行時間 和 實際占用時間。
3. [Times: user=0.00 sys=0.00, real=0.00 secs]

Concurrent Abortable Preclean(可終止的併發預清理)

這個階段嘗試著去承擔下一個階段Final Remark階段足夠多的工作。這個階段持續的時間依賴好多的因素，由於這個階段是重複的做相同的事情直到發生aboart的條件（比如：重複的次數、多少量的工作、持續的時間等等）之一才會停止。

日誌 ：

```
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-abortable-preclean-start]
2021-01-16T16:09:57.707-0800: 0.878:
  [CMS-concurrent-abortable-preclean: 0.000/0.000 secs]
  [Times: user=0.00 sys=0.00, real=0.00 secs]
```

解析 ：

1. CMS-concurrent-abortable-preclean ： 可取消的併發預清理
2. 0.000/0.000 secs ： GC線程運行時間 和 實際占用時間。
3. [Times: user=0.00 sys=0.00, real=0.00 secs] ： 併發時持續時間

Final Remark(最終標記)

這個階段會導致第二次STW，該階段的任務是完成標記整個年老代的所有的存活對象。

日誌 ：

```
2021-01-16T16:09:57.707-0800: 0.878:
  [GC (CMS Final Remark)
```

```
    [YG occupancy: 28068 K (157248 K)]
    2021-01-16T16:09:57.707-0800: 0.878:
      [Rescan (parallel) , 0.0003623 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [weak refs processing, 0.0000174 secs]
    2021-01-16T16:09:57.708-0800: 0.878:
      [class unloading, 0.0002188 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub symbol table, 0.0003117 secs]
    2021-01-16T16:09:57.708-0800: 0.879:
      [scrub string table, 0.0002084 secs]
      [1 CMS-remark: 343352K(349568K)]
    371420K(506816K), 0.0011940 secs]
[Times: user=0.01 sys=0.00, real=0.00 secs]
```

解析 ：

1. CMS Final Remark ： 最終標記
2. YG occupancy: 28068 K (157248 K) ： 年輕代使用量和總量
3. [Rescan (parallel) , 0.0003623 secs] ： 已並行方式，將 程序暫停後進行重新掃描(Rescan)，完成存活對想標記。花費 0.0003623 秒。
4. weak refs processing, 0.0000174 secs ： 處理old區的弱引用總時間
5. class unloading, 0.0002188 secs ： 卸載不使用的類總時間
6. scrub symbol table, 0.0003117 secs ： 清理符號表，及持有class級別 metadata的符號表(symbol tables)
7. scrub string table, 0.0002084 secs ： 清理內聯字串對應的 string tables
8. [1 CMS-remark: 343352K(349568K)] ： GC後，老年代使用量和總
9. 371420K(506816K), 0.0011940 secs] ： GC後，Heap使用量和總量
10. [Times: user=0.01 sys=0.00, real=0.00 secs] ： GC 持續時間

Concurrent Sweep(併發清除)

通過以上的標記，清除那些沒有標記的對象，並且回收空間。

日誌 ：

```
2021-01-16T16:09:57.709-0800: 0.879:
[CMS-concurrent-sweep-start]
2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-sweep: 0.001/0.001 secs]
[Times: user=0.00 sys=0.00, real=0.00 secs]
```

解析 ：

1. CMS-concurrent-sweep ： 併發清除老年代中所有未標記的對象。
2. 0.001/0.001 secs ： 該階段持續時間和實際佔用時間

Concurrent Reset(併發重置)

重新設置CMS算法內部的數據結構，準備下一個CMS生命週期的使用。

日誌 ：

```
2021-01-16T16:09:57.709-0800: 0.880:
[CMS-concurrent-reset-start]
2021-01-16T16:09:57.710-0800: 0.880:
[CMS-concurrent-reset: 0.000/0.000 secs]
[Times: user=0.00 sys=0.00, real=0.01 secs]
```

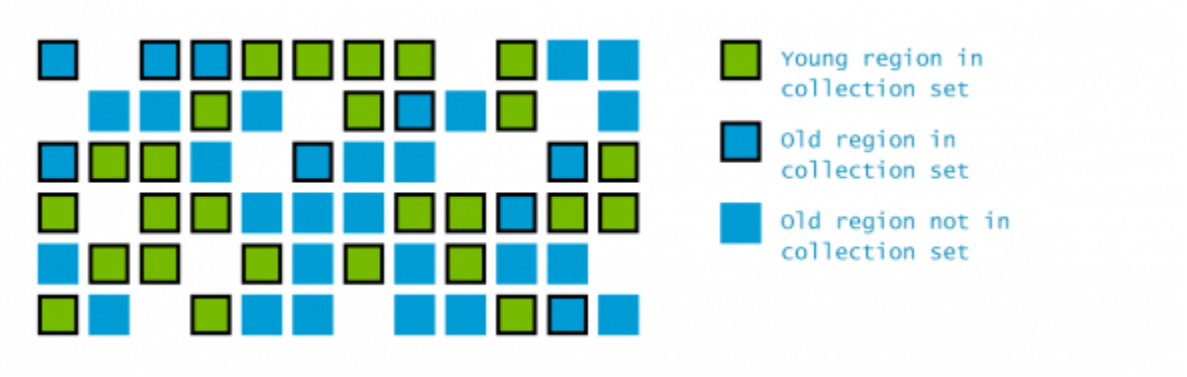
解析 ：

1. CMS-concurrent-reset ： 重置 CMS GC內部資料結構，為下次GC循環做準備
2. 0.000/0.000 secs ： 持續和實際佔用時間

G1 GC

特點 ：

1. 堆不分 年輕代 和 老年代，劃分成多個可存放的對象的小塊堆區域(smaller heap regions)
2. 每個區塊都可代表 Eden 、存活區、老年區
3. Eden & Survivor 合起來就是 年輕代
4. 在高堆內存下，吞吐量大，停頓時間小
5. 可指定最大停頓時間
6. 不會產生內存碎片



指令 ：

```
java -XX:+UseG1GC -Xms512m -Xmx512m -Xloggc:gc.demo.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps demo/jvm0204/GCLo
gAnalysis
```

日誌 ：

```
Java HotSpot(TM) 64-Bit Server VM (25.241-b07) for bsd-amd64 JRE (1.8.0_241-b07), built on Dec 11 2019 02:29:59 by
"java_re" with gcc 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)
Memory: 4k page, physical 16777216k(1127436k free)


CommandLine flags:
-XX:InitialHeapSize=536870912 -XX:MaxHeapSize=536870912
-XX:+PrintGC -XX:+PrintGCDateStamps
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+UseCompressedClassPointers -XX:+UseCompressedOops
-XX:+UseG1GC

2021-01-18T22:21:55.883-0800: 0.108:
  [GC pause (G1 Evacuation Pause) (young), 0.0037390 secs]
  [Parallel Time: 3.1 ms, GC Workers: 8]
......
```

```
   [Code Root Fixup: 0.0 ms]
   [Code Root Purge: 0.0 ms]
   [Clear CT: 0.1 ms]
   [Other: 0.5 ms]
......
[
 Eden: 25.0M(25.0M)->0.0B(21.0M)
 Survivors: 0.0B->4096.0K
 Heap: 33.9M(512.0M)->13.4M(512.0M)
]
[Times: user=0.01 sys=0.01, real=0.00 secs]



2021-01-18T22:21:56.139-0800: 0.364: [GC pause (G1 Humongous Allocation) (young) (to-space exhausted), 0.0036994 sec
s]
[Parallel Time: 1.2 ms, GC Workers: 8]
....
[
 Eden: 281.0M(297.0M)->0.0B(54.0M)
 Survivors: 10.0M->0.0B
 Heap: 451.5M(512.0M)->360.0M(512.0M)]
[Times: user=0.01 sys=0.01, real=0.01 secs]


2021-01-18T22:21:56.168-0800: 0.393:
[GC pause (G1 Evacuation Pause) (mixed), 0.0026751 secs]
[Parallel Time: 1.9 ms, GC Workers: 8]
....
[Eden: 18.0M(18.0M)->0.0B(21.0M) Survivors: 7168.0K->4096.0K Heap: 406.6M(512.0M)->341.3M(512.0M)]
[Times: user=0.01 sys=0.01, real=0.01 secs]


Heap
 garbage-first heap   total 524288K, used 385060K [0x00000007a0000000, 0x00000007a0101000, 0x00000007c0000000)
  region size 1024K, 5 young (5120K), 4 survivors (4096K)
 Metaspace       used 2598K, capacity 4486K, committed 4864K, reserved 1056768K
  class space    used 279K, capacity 386K, committed 512K, reserved 1048576K
```

(日誌分析或詳細總結後續會自主補上)