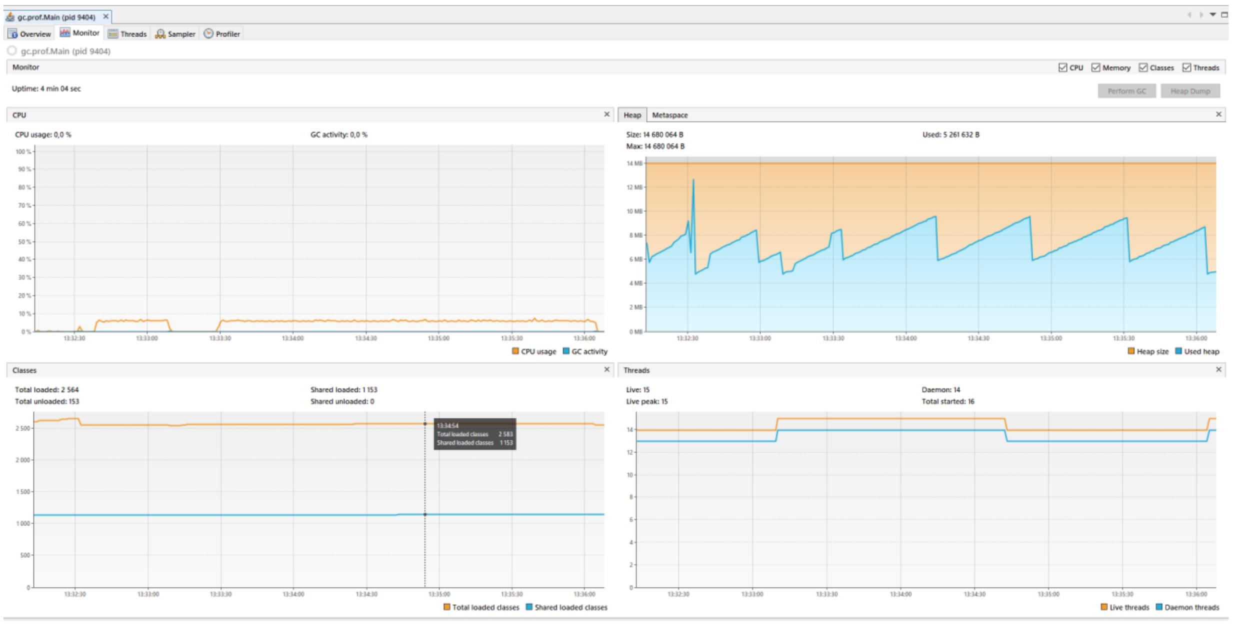
SerialGC

В ходе эксперимента было использовано два размера кэша. По итогу получили:

1. 12 Мб. хипа мало для серийного ГК. Судя по последней записи в логе Major очистка уже не происходила ([25.638s][info][gc] GC(26) Pause Full (Allocation Failure) 8M->8M(11M) 11.115ms), отсюда и место закончилось, так как удалять нечего было, все объекты уже «повзрослели» и возмужали настолько, что не влезли в 12 метров хипа.
2. 14 Мб. Хватило и на кривой графика мы видим, что минимальный размер хипа у нас должен быть величиной чуть более 12Мб.
3. Три итерации сортировки (22Мб.) + сохранение скринов заняли примерно 5-6 минут.



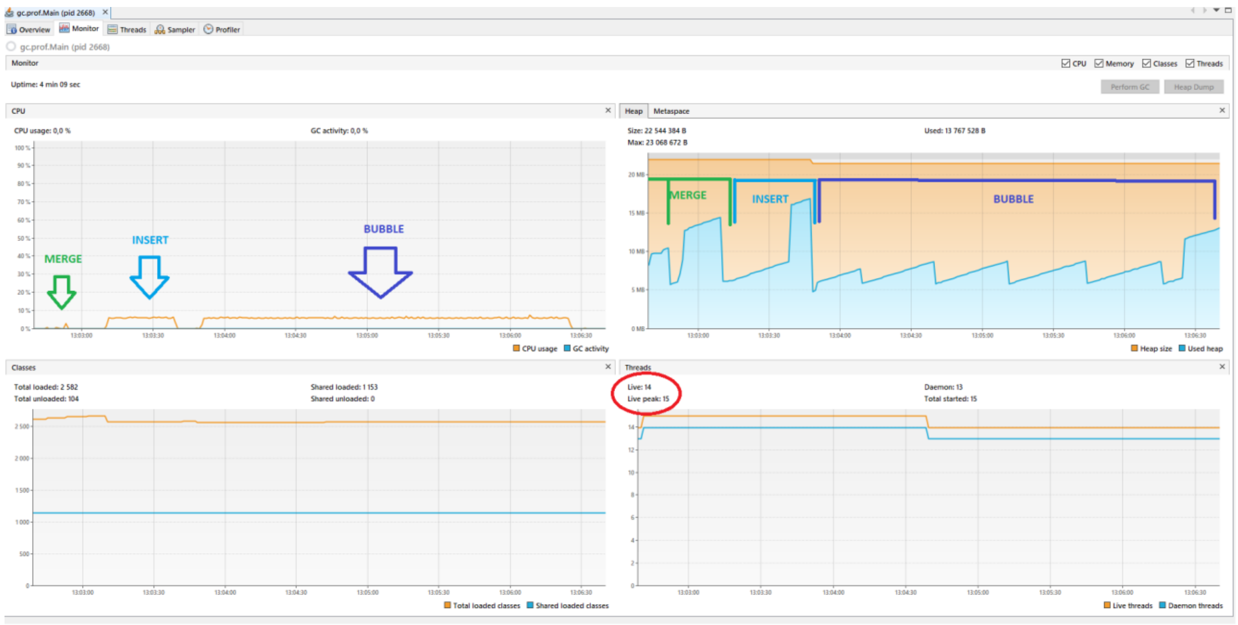
```
[0.010s][info][gc] Using Serial
[1.067s][info][gc] GC(0) Pause Young (Allocation Failure) 3M->1M(13M) 5.576ms
[4.658s][info][gc] GC(1) Pause Young (Allocation Failure) 5M->2M(13M) 4.692ms
[4.783s][info][gc] GC(2) Pause Young (Allocation Failure) 6M->3M(13M) 3.853ms
[4.832s][info][gc] GC(3) Pause Young (Allocation Failure) 7M->3M(13M) 2.308ms
[5.065s][info][gc] GC(4) Pause Young (Allocation Failure) 7M->4M(13M) 4.315ms
[9.857s][info][gc] GC(5) Pause Young (Allocation Failure) 8M->5M(13M) 9.940ms
[26.919s][info][gc] GC(6) Pause Young (Allocation Failure) 9M->6M(13M) 6.482ms
[28.497s][info][gc] GC(7) Pause Young (Allocation Failure) 10M->7M(13M) 1.488ms
[28.530s][info][gc] GC(8) Pause Young (Allocation Failure) 11M->7M(13M) 1.237ms
[28.543s][info][gc] GC(9) Pause Young (Allocation Failure) 11M->7M(13M) 1.523ms
[28.550s][info][gc] GC(10) Pause Young (Allocation Failure) 11M->8M(13M) 1.167ms
[28.556s][info][gc] GC(11) Pause Young (Allocation Failure) 11M->11M(13M) 0.113ms
[28.587s][info][gc] GC(12) Pause Full (Allocation Failure) 11M->6M(13M) 31.065ms
[28.600s][info][gc] GC(13) Pause Young (Allocation Failure) 10M->7M(13M) 1.272ms
[28.613s][info][gc] GC(14) Pause Young (Allocation Failure) 10M->7M(13M) 0.651ms
[28.645s][info][gc] GC(15) Pause Young (Allocation Failure) 10M->8M(13M) 1.416ms
[28.671s][info][gc] GC(16) Pause Young (Allocation Failure) 12M->12M(13M) 0.055ms
[28.688s][info][gc] GC(17) Pause Full (Allocation Failure) 12M->9M(13M) 16.900ms
[28.689s][info][gc] GC(18) Pause Young (Allocation Failure) 11M->11M(13M) 0.056ms
[28.703s][info][gc] GC(19) Pause Full (Allocation Failure) 11M->8M(13M) 13.933ms
[28.922s][info][gc] GC(20) Pause Young (Allocation Failure) 12M->12M(13M) 0.148ms
[28.948s][info][gc] GC(21) Pause Full (Allocation Failure) 12M->4M(13M) 25.324ms
[55.877s][info][gc] GC(22) Pause Young (Allocation Failure) 8M->5M(13M) 1.675ms
```

```
[64.981s][info][gc] GC(23) Pause Young (Allocation Failure) 8M->6M(13M) 1.504ms
[64.987s][info][gc] GC(24) Pause Young (Allocation Failure) 9M->9M(13M) 2.017ms
[64.988s][info][gc] GC(25) Pause Young (Allocation Failure) 11M->11M(13M) 0.126ms
[65.014s][info][gc] GC(26) Pause Full (Allocation Failure) 11M->7M(13M) 25.397ms
[65.885s][info][gc] GC(27) Pause Young (Allocation Failure) 11M->11M(13M) 0.129ms
[65.928s][info][gc] GC(28) Pause Full (Allocation Failure) 11M->4M(13M) 42.717ms
[89.930s][info][gc] GC(29) Pause Young (Allocation Failure) 8M->5M(13M) 2.068ms
[128.946s][info][gc] GC(30) Pause Young (Allocation Failure) 9M->5M(13M) 0.901ms
[167.962s][info][gc] GC(31) Pause Young (Allocation Failure) 9M->5M(13M) 1.617ms
[208.950s][info][gc] GC(32) Pause Young (Allocation Failure) 9M->5M(13M) 1.582ms
[240.778s][info][gc] GC(33) Pause Young (Allocation Failure) 9M->6M(13M) 1.599ms
[240.782s][info][gc] GC(34) Pause Young (Allocation Failure) 7M->6M(13M) 1.433ms
[240.786s][info][gc] GC(35) Pause Young (Allocation Failure) 9M->9M(13M) 1.754ms
[240.787s][info][gc] GC(36) Pause Young (Allocation Failure) 11M->11M(13M) 0.121ms
[240.820s][info][gc] GC(37) Pause Full (Allocation Failure) 11M->7M(13M) 32.758ms
[240.972s][info][gc] GC(38) Pause Young (Allocation Failure) 11M->11M(13M) 0.130ms
[241.010s][info][gc] GC(39) Pause Full (Allocation Failure) 11M->4M(13M) 38.417ms,
```

ParallelGC

В ходе эксперимента было использовано два размера кэша (на самом деле было еще пару шагов в +2МБ). По итогу получили:

1. 12 Мб. хипа мало для параллельного ГК. Судя по последней записи в логе Major очистка уже не происходила ([179.118s][info][gc] GC(21) Pause Full (Allocation Failure) 6M->6M(11M) 26.986ms), отсюда и место закончилось, так как удалять нечего было, все объекты уже «повзрослели» и возмужали настолько, что не влезли в 12 метров хипа.
2. 22 Мб. Хватило и на кривой графика мы видим, что минимальный размер хипа у нас должен быть величиной 18Мб.
3. Три итерации сортировки (22Мб.) + сохранение скринов заняли примерно 4 минуты.



Uptime: 4 min 09 sec

CPU usage: 0,0 %
GC activity: 0,0 %

Size: 22 544 384 B
Max: 23 068 672 B
Used: 13 767 528 B

Total loaded: 2 582
Total unloaded: 104
Shared loaded: 1 153
Shared unloaded: 0

Live: 14
Live peak: 15
Daemon: 13
Total started: 15

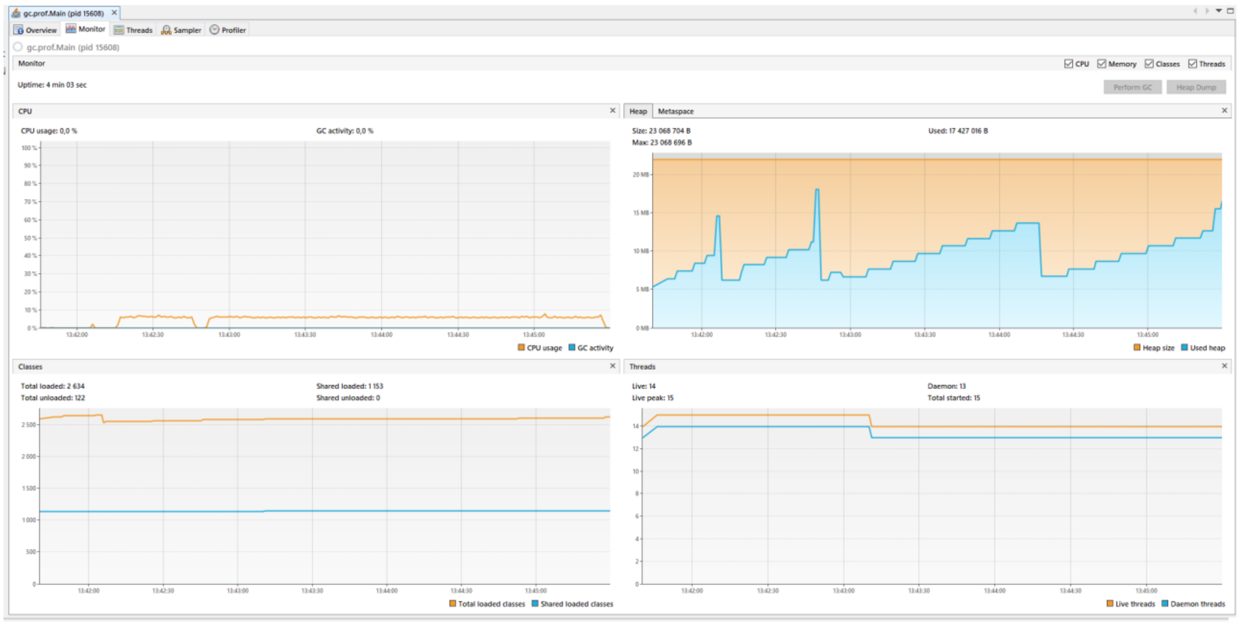
```
[0.008s][info][gc] Using Parallel
[3.434s][info][gc] GC(0) Pause Young (Allocation Failure) 6M->2M(21M) 4.995ms
[3.596s][info][gc] GC(1) Pause Young (Allocation Failure) 8M->3M(21M) 3.415ms
[3.815s][info][gc] GC(2) Pause Young (Allocation Failure) 9M->4M(21M) 3.571ms
[16.685s][info][gc] GC(3) Pause Young (Allocation Failure) 10M->5M(21M) 5.981ms
[22.454s][info][gc] GC(4) Pause Young (Allocation Failure) 11M->7M(21M) 2.863ms
[22.494s][info][gc] GC(5) Pause Young (Allocation Failure) 13M->8M(18M) 4.730ms
[22.501s][info][gc] GC(6) Pause Young (Allocation Failure) 11M->8M(20M) 2.002ms
[22.508s][info][gc] GC(7) Pause Young (Allocation Failure) 11M->8M(20M) 1.641ms
```

```
[22.513s][info][gc] GC(8) Pause Young (Allocation Failure) 11M->8M(20M) 0.357ms
[22.519s][info][gc] GC(9) Pause Young (Allocation Failure) 11M->8M(20M) 0.512ms
[22.524s][info][gc] GC(10) Pause Young (Allocation Failure) 11M->9M(20M) 0.394ms
[22.531s][info][gc] GC(11) Pause Young (Allocation Failure) 12M->9M(20M) 0.610ms
[22.603s][info][gc] GC(12) Pause Full (Ergonomics) 16M->9M(20M) 46.866ms
[39.669s][info][gc] GC(13) Pause Full (Ergonomics) 15M->6M(20M) 22.464ms
[68.043s][info][gc] GC(14) Pause Young (Allocation Failure) 9M->6M(20M) 0.917ms
[76.736s][info][gc] GC(15) Pause Full (Ergonomics) 16M->4M(19M) 60.148ms
[97.700s][info][gc] GC(16) Pause Young (Allocation Failure) 7M->5M(19M) 0.510ms
[128.718s][info][gc] GC(17) Pause Young (Allocation Failure) 8M->5M(19M) 0.776ms
[159.728s][info][gc] GC(18) Pause Young (Allocation Failure) 8M->5M(19M) 0.657ms
[192.741s][info][gc] GC(19) Pause Young (Allocation Failure) 8M->5M(19M) 0.851ms
[225.729s][info][gc] GC(20) Pause Young (Allocation Failure) 8M->5M(19M) 0.697ms
[234.019s][info][gc] GC(21) Pause Young (Allocation Failure) 7M->6M(19M) 0.427ms
[234.734s][info][gc] GC(22) Pause Young (Allocation Failure) 15M->11M(19M) 0.724ms
```

G1GC

В ходе эксперимента было использовано два размера кэша. По итогу получили:

1. 12 Мб. хипа мало для серийного ГК. Аналогично с parallelGC
2. 22 Мб. хватило и на кривой графика мы видим, что минимальный размер хипа у нас должен быть величиной чуть более 18-19Мб.
3. Три итерации сортировки (22Мб.) + сохранение скринов заняли примерно 3- 4 минуты.
4. Кривая графика сильно изменилась в сравнении с SerialGC и ParallelGC. В данном случае видно как используемый хип нарастает, а затем Major-очистка происходит.



Uptime: 4 min 03 sec

CPU usage: 0,0 %   GC activity: 0,0 %

Size: 23 068 704 B
Max: 23 068 696 B
Used: 17 427 016 B

Total loaded: 2 634
Total unloaded: 122
Shared loaded: 1 153
Shared unloaded: 0

Live: 14
Live peak: 15
Daemon: 13
Total started: 15

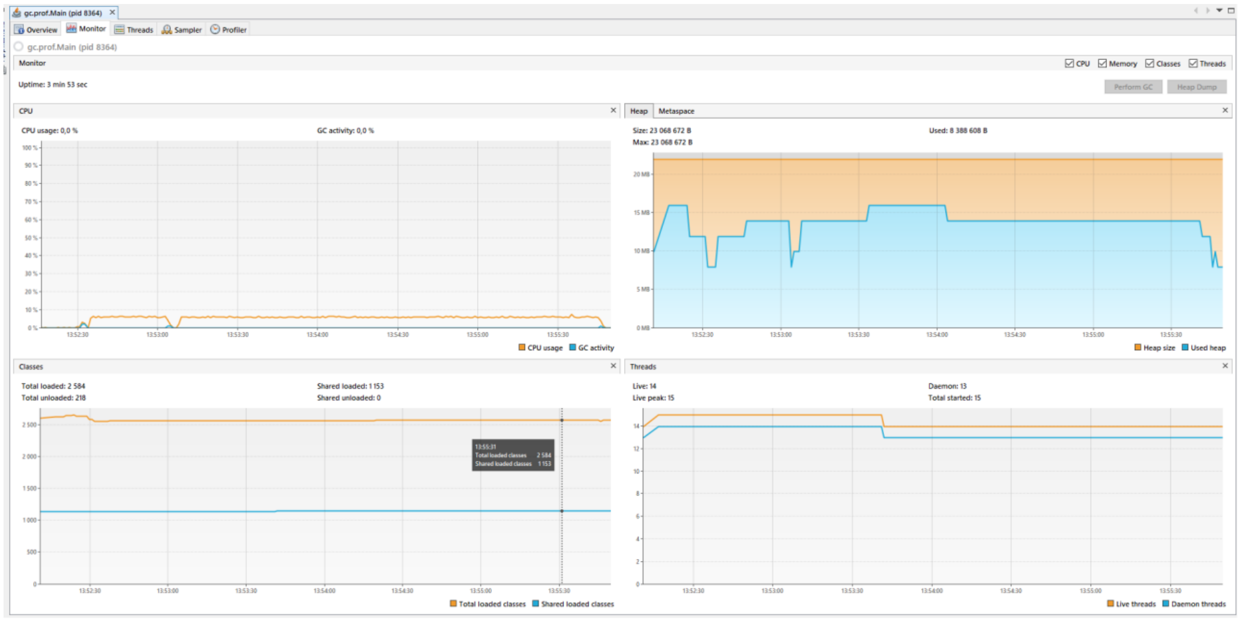
```
[0.010s][info][gc] Using G1
[8.388s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 9M->2M(22M) 6.944ms
[8.525s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 8M->3M(22M) 4.681ms
[13.518s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 10M->4M(22M) 5.292ms
[38.596s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 13M->7M(22M) 3.368ms
[38.639s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 13M->7M(22M) 2.227ms
[38.653s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 14M->7M(22M) 1.909ms
[38.666s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 14M->8M(22M) 2.288ms
[38.690s][info][gc] GC(7) Pause Young (Concurrent Start) (G1 Humongous Allocation) 14M->9M(22M) 1.409ms
```

```
[38.690s][info][gc] GC(8) Concurrent Mark Cycle
[38.704s][info][gc] GC(8) Pause Remark 11M->11M(22M) 9.928ms
[38.706s][info][gc] GC(8) Pause Cleanup 11M->11M(22M) 0.100ms
[38.706s][info][gc] GC(8) Concurrent Mark Cycle 16.535ms
[38.714s][info][gc] GC(9) Pause Young (Prepare Mixed) (G1 Humongous Allocation) 16M->10M(22M) 0.703ms
[40.597s][info][gc] GC(10) Pause Young (Mixed) (G1 Evacuation Pause) 15M->6M(22M) 4.292ms
[78.373s][info][gc] GC(11) Pause Young (Concurrent Start) (G1 Humongous Allocation) 13M->8M(22M) 2.348ms
[78.373s][info][gc] GC(12) Concurrent Undo Cycle
[78.374s][info][gc] GC(12) Concurrent Undo Cycle 0.129ms
[78.377s][info][gc] GC(13) Pause Young (Concurrent Start) (G1 Humongous Allocation) 10M->9M(22M) 1.479ms
[78.377s][info][gc] GC(14) Concurrent Mark Cycle
[78.384s][info][gc] GC(14) Pause Remark 18M->18M(22M) 3.772ms
[78.387s][info][gc] GC(14) Pause Cleanup 18M->18M(22M) 0.037ms
[78.387s][info][gc] GC(14) Concurrent Mark Cycle 9.784ms
[80.616s][info][gc] GC(15) Pause Young (Prepare Mixed) (G1 Preventive Collection) 19M->6M(22M) 2.484ms
[90.137s][info][gc] GC(16) Pause Young (Mixed) (G1 Evacuation Pause) 8M->6M(22M) 4.622ms
[169.653s][info][gc] GC(17) Pause Young (Normal) (G1 Evacuation Pause) 14M->6M(22M) 2.048ms
[239.691s][info][gc] GC(18) Pause Young (Concurrent Start) (G1 Humongous Allocation) 16M->8M(22M) 2.166ms
[239.691s][info][gc] GC(19) Concurrent Undo Cycle
[239.691s][info][gc] GC(19) Concurrent Undo Cycle 0.143ms
[239.695s][info][gc] GC(20) Pause Young (Concurrent Start) (G1 Humongous Allocation) 11M->9M(22M) 1.764ms
[239.695s][info][gc] GC(21) Concurrent Mark Cycle
[239.707s][info][gc] GC(21) Pause Remark 15M->15M(22M) 6.178ms
[239.710s][info][gc] GC(21) Pause Cleanup 15M->15M(22M) 0.071ms
[239.711s][info][gc] GC(21) Concurrent Mark Cycle 15.421ms
```

UseZGC

В ходе эксперимента было использовано два размера кэша. По итогу получили:

1. 12 Мб. хипа мало для ZGC.
2. 22 Мб. Хватило и на кривой графика мы видим, что минимальный размер хипа у нас должен быть величиной 16Мб.
3. Три итерации сортировки (22Мб.) + сохранение скринов заняли примерно 4 минуты.
4. Графическая кривая сильно отличается от предыдущих. Хип используется почти всегда на пределе и очистка мусора происходит системно в определенное время.



```
gc.prof.Main (pid 8364)
Monitor
Uptime: 3 min 53 sec

CPU
CPU usage: 0,0 %          GC activity: 0,0 %

Heap
Size: 23 068 672 B        Used: 8 388 608 B
Max: 23 068 672 B

Classes
Total loaded: 2 584       Shared loaded: 1 153
Total unloaded: 218       Shared unloaded: 0

13:55:31
Total loaded classes    2 584
Shared loaded classes   1 153

Threads
Live: 14                  Daemon: 13
Live peak: 15             Total started: 15
```

```
[0.012s][info][gc] Using The Z Garbage Collector
[0.148s][info][gc] GC(0) Garbage Collection (Warmup) 4M(18%)->4M(18%)
[0.244s][info][gc] GC(1) Garbage Collection (Warmup) 6M(27%)->4M(18%)
[9.752s][info][gc] GC(2) Garbage Collection (Warmup) 12M(55%)->8M(36%)
[9.854s][info][gc] GC(3) Garbage Collection (Allocation Rate) 12M(55%)->10M(45%)
[9.963s][info][gc] GC(4) Garbage Collection (Allocation Rate) 14M(64%)->8M(36%)
[10.060s][info][gc] GC(5) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[10.157s][info][gc] GC(6) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[10.267s][info][gc] GC(7) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[10.378s][info][gc] GC(8) Garbage Collection (Allocation Rate) 8M(36%)->10M(45%)
[10.470s][info][gc] GC(9) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[10.566s][info][gc] GC(10) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[10.670s][info][gc] GC(11) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[10.769s][info][gc] GC(12) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[11.370s][info][gc] GC(13) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[11.467s][info][gc] GC(14) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[11.570s][info][gc] GC(15) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[14.954s][info][gc] GC(16) Garbage Collection (Proactive) 14M(64%)->10M(45%)
[28.083s][info][gc] GC(17) Garbage Collection (Proactive) 14M(64%)->8M(36%)
[33.283s][info][gc] GC(18) Garbage Collection (Proactive) 12M(55%)->10M(45%)
[34.527s][info][gc] Allocation Stall (main) 30.391ms
[34.529s][info][gc] GC(19) Garbage Collection (Allocation Stall) 22M(100%)->12M(55%)
[34.556s][info][gc] Allocation Stall (main) 12.730ms
[34.558s][info][gc] GC(20) Garbage Collection (Allocation Rate) 16M(73%)->20M(91%)
[34.590s][info][gc] Allocation Stall (main) 21.671ms
[34.592s][info][gc] GC(21) Garbage Collection (Allocation Stall) 22M(100%)->14M(64%)
[34.632s][info][gc] Allocation Stall (main) 21.941ms
[34.635s][info][gc] GC(22) Garbage Collection (Allocation Stall) 22M(100%)->18M(82%)
[34.663s][info][gc] Allocation Stall (main) 22.535ms
[34.667s][info][gc] GC(23) Garbage Collection (Allocation Stall) 22M(100%)->20M(91%)
[34.691s][info][gc] Allocation Stall (main) 26.396ms
[34.693s][info][gc] GC(24) Garbage Collection (Allocation Stall) 20M(91%)->20M(91%)
[34.765s][info][gc] GC(25) Garbage Collection (Allocation Rate) 20M(91%)->18M(82%)
[34.871s][info][gc] GC(26) Garbage Collection (Allocation Rate) 18M(82%)->8M(36%)
[34.958s][info][gc] GC(27) Garbage Collection (Allocation Rate) 8M(36%)->10M(45%)
[35.056s][info][gc] GC(28) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[35.158s][info][gc] GC(29) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.256s][info][gc] GC(30) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.353s][info][gc] GC(31) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.452s][info][gc] GC(32) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.573s][info][gc] GC(33) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[35.673s][info][gc] GC(34) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.768s][info][gc] GC(35) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
```

```
[35.872s][info][gc] GC(36) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[35.973s][info][gc] GC(37) Garbage Collection (Allocation Rate) 8M(36%)->10M(45%)
[36.068s][info][gc] GC(38) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[36.174s][info][gc] GC(39) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[36.272s][info][gc] GC(40) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[36.368s][info][gc] GC(41) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[36.473s][info][gc] GC(42) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[38.767s][info][gc] GC(43) Garbage Collection (Proactive) 12M(55%)->12M(55%)
[50.053s][info][gc] GC(44) Garbage Collection (Proactive) 16M(73%)->14M(64%)
[66.903s][info][gc] Allocation Stall (main) 23.254ms
[66.903s][info][gc] Relocation Stall (main) 0.019ms
[66.905s][info][gc] GC(45) Garbage Collection (Allocation Stall) 22M(100%)->16M(73%)
[66.930s][info][gc] Allocation Stall (main) 21.714ms
[66.932s][info][gc] GC(46) Garbage Collection (Allocation Stall) 20M(91%)->18M(82%)
[66.967s][info][gc] GC(47) Garbage Collection (Allocation Rate) 18M(82%)->16M(73%)
[67.081s][info][gc] GC(48) Garbage Collection (Allocation Rate) 16M(73%)->8M(36%)
[67.173s][info][gc] GC(49) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[67.253s][info][gc] GC(50) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[67.353s][info][gc] GC(51) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[67.462s][info][gc] GC(52) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[67.569s][info][gc] GC(53) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[67.675s][info][gc] GC(54) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[68.166s][info][gc] GC(55) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[68.275s][info][gc] GC(56) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[68.373s][info][gc] GC(57) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[68.469s][info][gc] GC(58) Garbage Collection (Allocation Rate) 10M(45%)->10M(45%)
[68.575s][info][gc] GC(59) Garbage Collection (Allocation Rate) 12M(55%)->10M(45%)
[71.275s][info][gc] GC(60) Garbage Collection (Proactive) 14M(64%)->14M(64%)
[97.053s][info][gc] GC(61) Garbage Collection (Proactive) 18M(82%)->14M(64%)
[127.060s][info][gc] GC(62) Garbage Collection (Proactive) 18M(82%)->12M(55%)
[159.058s][info][gc] GC(63) Garbage Collection (Proactive) 16M(73%)->12M(55%)
[192.062s][info][gc] GC(64) Garbage Collection (Proactive) 16M(73%)->12M(55%)
[225.070s][info][gc] GC(65) Garbage Collection (Proactive) 16M(73%)->12M(55%)
[228.968s][info][gc] Allocation Stall (main) 40.416ms
[228.973s][info][gc] Allocation Stall (main) 1.372ms
[228.973s][info][gc] GC(66) Garbage Collection (Allocation Stall) 22M(100%)->22M(100%)
[229.069s][info][gc] Allocation Stall (C1 CompilerThread0) 32.113ms
[229.069s][info][gc] Allocation Stall (RMI TCP Connection(2)-192.168.238.1) 34.893ms
[229.071s][info][gc] GC(67) Garbage Collection (Allocation Stall) 22M(100%)->10M(45%)
[229.184s][info][gc] GC(68) Garbage Collection (Allocation Stall) 10M(45%)->8M(36%)
[229.255s][info][gc] GC(69) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[229.365s][info][gc] GC(70) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[229.453s][info][gc] GC(71) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[229.568s][info][gc] GC(72) Garbage Collection (Allocation Rate) 8M(36%)->8M(36%)
[229.663s][info][gc] GC(73) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[230.475s][info][gc] GC(74) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
[230.665s][info][gc] GC(75) Garbage Collection (Allocation Rate) 10M(45%)->8M(36%)
```