# Профилирование Garbage Collector - Parallel

Общий график Heap, на котором выделены сортировки:

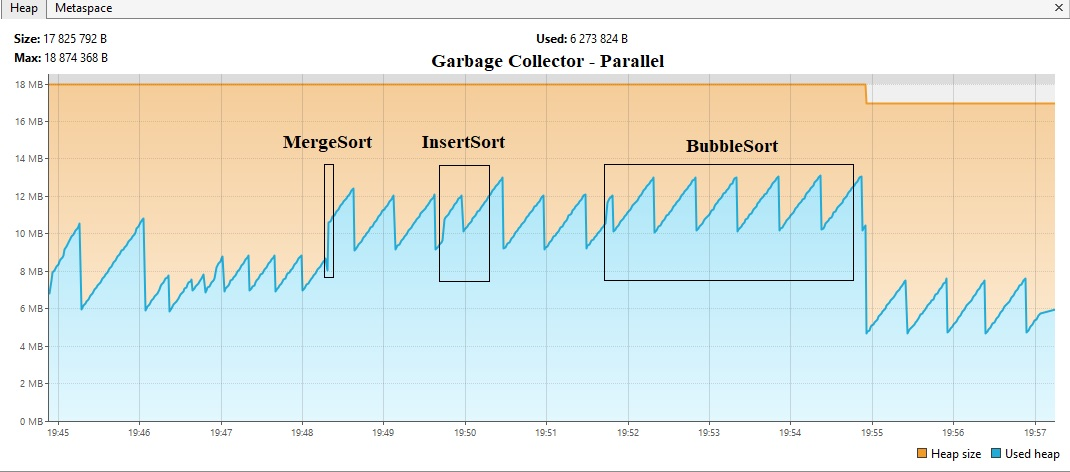
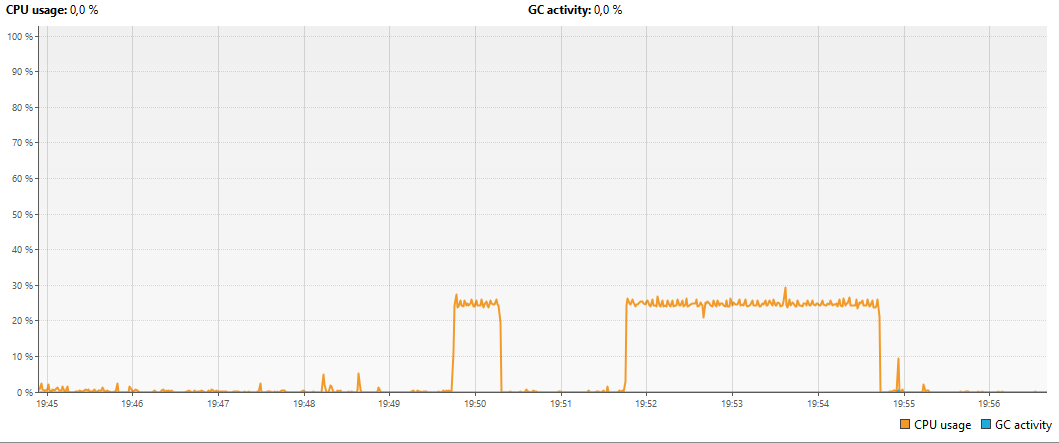


График загрузки процессора на каждой сборке (название сборок см. на графике выше). Сортировка пузырьком потребовала больше всего ресурсов процессора:



Нагрузка на процессор составила около 30%.

Введите количество элементов

250000

Массив создан и заполнен случайными значениями от 1 до 250000

**MergeSort**

MergeSort Начало сортировки - 19:48:18.276827100

MergeSort Конец сортировки - 19:48:18.381760300

**Вывод**: на сортировку MergeSort затрачено **105 мс.** Used heap увеличился в 1,5 раза (с 9 Mb до 13 Mb). График стабилен. Идут малые сборки.

**InsertSort**

InsertSort Начало сортировки - 19:49:43.903218700

InsertSort Конец сортировки - 19:50:17.171442700

**Вывод**: на сортировку InsertSort затрачено **33 сек**. Used heap практически не изменился. График стабилен. Идут малые сборки.

**BubbleSort**

BubbleSort Начало сортировки - 19:51:44.511655

BubbleSort Конец сортировки - 19:54:42.606299300

**Вывод**: на сортировку BubbleSort затрачено около **3 мин**. В момент сортировки Used heap увеличился на 15% (около 14 Mb). По окончании сортировки прошла Full (полная) сборка и Used heap уменьшился на 43%, составив 8 Mb. Незначительно уменьшился размер HEAP.

Лог:

[0.026s][info][gc] Using Parallel  
[4.227s][info][gc] GC(0) Pause Young (Allocation Failure) 5M->2M(17M) 6.884ms  
[4.530s][info][gc] GC(1) Pause Young (Allocation Failure) 7M->3M(17M) 4.307ms  
[4.657s][info][gc] GC(2) Pause Young (Allocation Failure) 8M->4M(17M) 4.989ms  
[5.386s][info][gc] GC(3) Pause Young (Allocation Failure) 9M->5M(17M) 6.532ms  
[27.563s][info][gc] GC(4) Pause Young (Allocation Failure) 10M->5M(17M) 2.863ms  
[74.590s][info][gc] GC(5) Pause Young (Allocation Failure) 10M->5M(14M) 2.573ms  
[93.864s][info][gc] GC(6) Pause Young (Allocation Failure) 7M->5M(16M) 2.606ms  
[110.204s][info][gc] GC(7) Pause Young (Allocation Failure) 7M->5M(16M) 1.623ms  
[119.636s][info][gc] GC(8) Pause Young (Allocation Failure) 7M->6M(16M) 0.926ms  
[133.643s][info][gc] GC(9) Pause Young (Allocation Failure) 8M->6M(16M) 0.800ms  
[152.655s][info][gc] GC(10) Pause Young (Allocation Failure) 8M->6M(16M) 0.846ms  
[171.669s][info][gc] GC(11) Pause Young (Allocation Failure) 8M->6M(16M) 0.370ms  
[191.681s][info][gc] GC(12) Pause Young (Allocation Failure) 8M->6M(16M) 0.358ms  
[209.839s][info][gc] GC(13) Pause Young (Allocation Failure) 8M->6M(16M) 0.350ms  
[209.845s][info][gc] GC(14) Pause Young (Allocation Failure) 8M->7M(16M) 0.575ms  
[209.855s][info][gc] GC(15) Pause Young (Allocation Failure) 9M->7M(16M) 0.956ms  
[209.876s][info][gc] GC(16) Pause Young (Allocation Failure) 9M->8M(16M) 0.412ms  
[209.898s][info][gc] GC(17) Pause Young (Allocation Failure) 10M->8M(16M) 0.368ms  
[209.903s][info][gc] GC(18) Pause Young (Allocation Failure) 10M->8M(16M) 0.437ms  
[209.907s][info][gc] GC(19) Pause Young (Allocation Failure) 10M->8M(16M) 0.320ms  
[209.911s][info][gc] GC(20) Pause Young (Allocation Failure) 10M->8M(16M) 0.316ms  
[209.915s][info][gc] GC(21) Pause Young (Allocation Failure) 10M->8M(16M) 0.385ms  
[209.920s][info][gc] GC(22) Pause Young (Allocation Failure) 10M->8M(16M) 0.600ms  
[209.924s][info][gc] GC(23) Pause Young (Allocation Failure) 10M->8M(16M) 0.323ms  
[209.928s][info][gc] GC(24) Pause Young (Allocation Failure) 10M->8M(16M) 0.300ms  
[209.933s][info][gc] GC(25) Pause Young (Allocation Failure) 10M->9M(15M) 0.427ms  
[209.937s][info][gc] GC(26) Pause Young (Allocation Failure) 11M->9M(15M) 0.330ms  
[209.941s][info][gc] GC(27) Pause Young (Allocation Failure) 10M->9M(16M) 0.286ms  
[229.705s][info][gc] GC(28) Pause Young (Allocation Failure) 12M->9M(16M) 0.329ms  
[259.721s][info][gc] GC(29) Pause Young (Allocation Failure) 12M->9M(16M) 0.526ms  
[289.741s][info][gc] GC(30) Pause Young (Allocation Failure) 12M->9M(16M) 0.371ms  
[309.753s][info][gc] GC(31) Pause Young (Allocation Failure) 12M->10M(16M) 0.814ms  
[339.767s][info][gc] GC(32) Pause Young (Allocation Failure) 13M->9M(16M) 0.370ms  
[369.789s][info][gc] GC(33) Pause Young (Allocation Failure) 12M->9M(16M) 0.371ms  
[400.807s][info][gc] GC(34) Pause Young (Allocation Failure) 12M->9M(16M) 0.365ms  
[420.821s][info][gc] GC(35) Pause Young (Allocation Failure) 12M->10M(16M) 0.483ms  
[451.221s][info][gc] GC(36) Pause Young (Allocation Failure) 13M->10M(16M) 0.735ms  
[481.845s][info][gc] GC(37) Pause Young (Allocation Failure) 13M->10M(16M) 0.387ms  
[512.267s][info][gc] GC(38) Pause Young (Allocation Failure) 13M->10M(16M) 0.286ms  
[542.873s][info][gc] GC(39) Pause Young (Allocation Failure) 13M->10M(16M) 0.399ms  
[573.885s][info][gc] GC(40) Pause Young (Allocation Failure) 13M->10M(16M) 0.305ms  
[603.962s][info][gc] GC(41) Pause Young (Allocation Failure) 13M->10M(16M) 0.389ms  
[606.808s][info][gc] GC(42) Pause Young (Heap Dump Initiated GC) 10M->10M(15M) 0.370ms  
[606.844s][info][gc] GC(43) Pause Full (Heap Dump Initiated GC) 10M->4M(15M) 35.933ms  
[636.984s][info][gc] GC(44) Pause Young (Allocation Failure) 7M->4M(16M) 0.519ms  
[666.999s][info][gc] GC(45) Pause Young (Allocation Failure) 7M->4M(16M) 0.342ms  
[695.388s][info][gc] GC(46) Pause Young (Allocation Failure) 7M->4M(16M) 0.294ms  
[725.037s][info][gc] GC(47) Pause Young (Allocation Failure) 7M->4M(16M) 0.344ms

**Анализ Лога:**

Наблюдаются частые, но кратковременные задержки времени на Young сборку мусора, длительностью от 0.3 мс до 1.0 мс. Длительные Young паузы отмечены вначале работы приложения – до 7 мс.

Единственная значительная пауза, длительностью 36 мс, была вызвана Full сборкой мусора.

# Профилирование Garbage Collector – G1

Общий график Heap, на котором выделены сортировки:

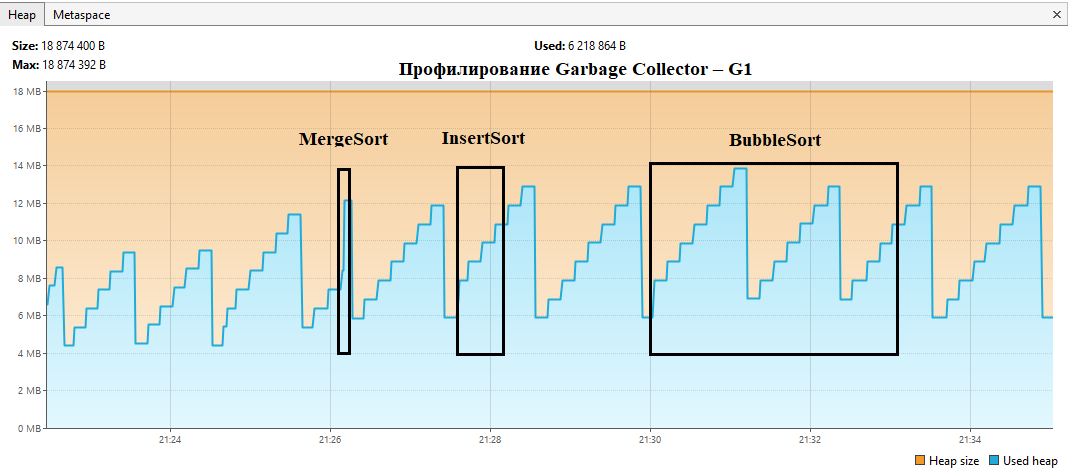
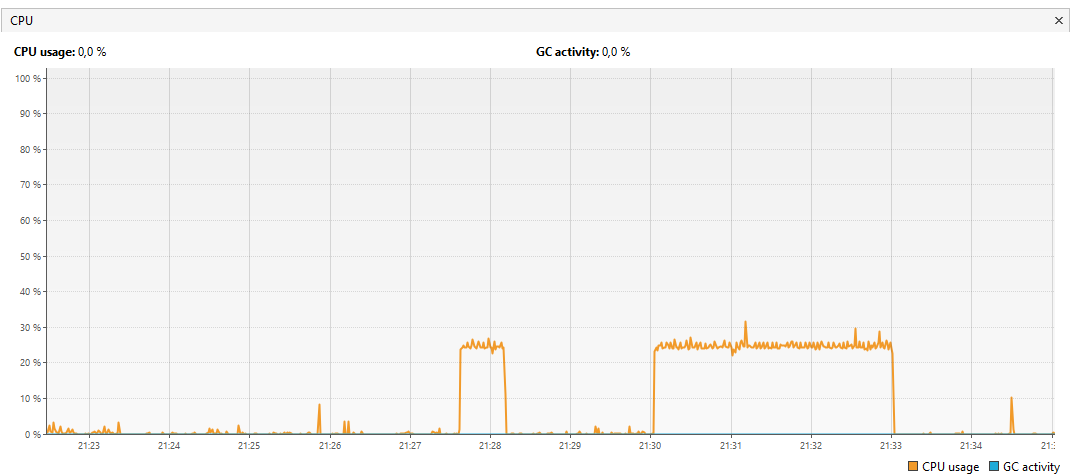


График загрузки процессора на каждой сборке (название сборок см. на графике выше). Сортировка пузырьком потребовала больше всего ресурсов процессора:



Нагрузка на процессор составила около 30%.

Введите количество элементов

250000

Массив создан и заполнен случайными значениями от 1 до 250000

**MergeSort**

MergeSort Начало сортировки - 21:26:09.873562900

MergeSort Конец сортировки - 21:26:09.969500800

**Вывод**: на сортировку MergeSort затрачено **96 мс** против **105 мс** для Parallel**.** Used heap увеличился на 7% (с 12 Mb до 12,8 Mb) против 50% для Parallel. График стабилен. Идут малые сборки.

**InsertSort**

InsertSort Начало сортировки - 21:27:36.749899400

InsertSort Конец сортировки - 21:28:10.259010500

**Вывод**: на сортировку InsertSort затрачено **34 сек**. Used heap увеличился на 12%. График стабилен. Идут малые сборки.

**BubbleSort**

BubbleSort Начало сортировки - 21:30:01.884760

BubbleSort Конец сортировки - 21:33:00.861627100

**Вывод**: на сортировку BubbleSort затрачено около **3 мин**. В моменте Used heap увеличился на 8% (около 14 Mb). По окончании сортировки отсутствовала Full (полная) сборка и Used heap остался неизменным, составив 13 Mb. Размер HEAP не изменился. Более стабильный график Used heap, относительно Parallel.

Лог:

[0.050s][info][gc] Using G1  
[5.659s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 8M->2M(18M) 7.133ms  
[5.846s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 7M->3M(18M) 4.197ms  
[6.372s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 8M->3M(18M) 2.800ms  
[17.874s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 9M->4M(18M) 5.271ms  
[71.894s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 10M->4M(18M) 3.343ms  
[130.023s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 10M->4M(18M) 0.843ms  
[197.652s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 12M->5M(18M) 1.073ms  
[228.775s][info][gc] GC(7) Pause Young (Normal) (G1 Evacuation Pause) 12M->6M(18M) 1.017ms  
[228.820s][info][gc] GC(8) Pause Young (Normal) (G1 Evacuation Pause) 12M->6M(18M) 0.835ms  
[228.834s][info][gc] GC(9) Pause Young (Normal) (G1 Evacuation Pause) 12M->6M(18M) 0.819ms  
[228.846s][info][gc] GC(10) Pause Young (Normal) (G1 Evacuation Pause) 12M->7M(18M) 0.919ms  
[235.094s][info][gc] GC(11) Pause Young (Normal) (G1 Evacuation Pause) 14M->5M(18M) 0.697ms  
[304.136s][info][gc] GC(12) Pause Young (Normal) (G1 Evacuation Pause) 12M->5M(18M) 0.684ms  
[372.172s][info][gc] GC(13) Pause Young (Normal) (G1 Evacuation Pause) 13M->5M(18M) 0.655ms  
[452.289s][info][gc] GC(14) Pause Young (Normal) (G1 Evacuation Pause) 13M->5M(18M) 0.628ms  
[531.322s][info][gc] GC(15) Pause Young (Normal) (G1 Evacuation Pause) 14M->6M(18M) 0.706ms  
[601.349s][info][gc] GC(16) Pause Young (Normal) (G1 Evacuation Pause) 13M->6M(18M) 0.733ms  
[670.815s][info][gc] GC(17) Pause Young (Normal) (G1 Evacuation Pause) 13M->5M(18M) 0.733ms  
[752.433s][info][gc] GC(18) Pause Young (Normal) (G1 Evacuation Pause) 13M->5M(18M) 0.704ms

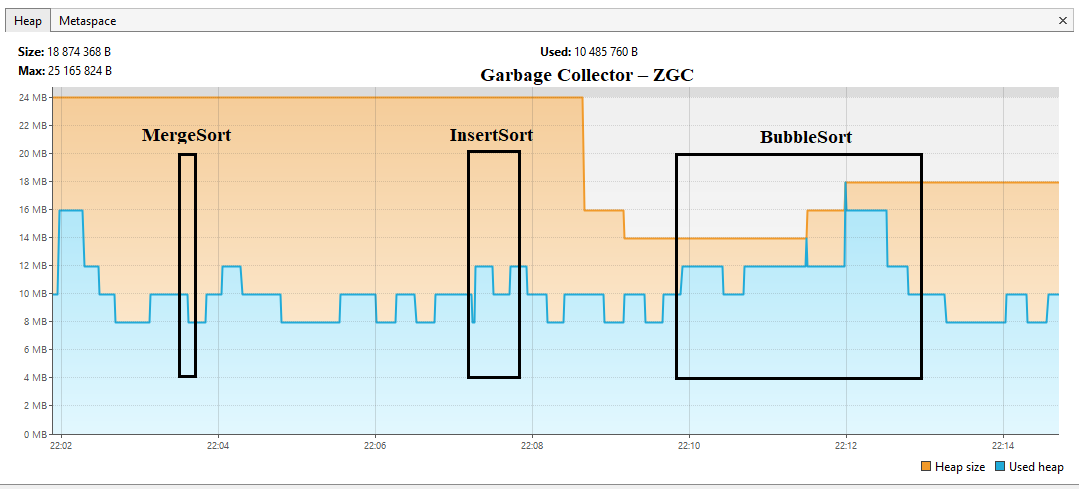
**Анализ Лога:**

Наблюдаются более редкие (по сравнению с Parallel), кратковременные задержки времени на Young сборку мусора, длительностью от 0.6 мс до 1.1 мс. Длительные Young паузы отмечены вначале работы приложения – до 7 мс.

Full сборка мусора не вызывалась.

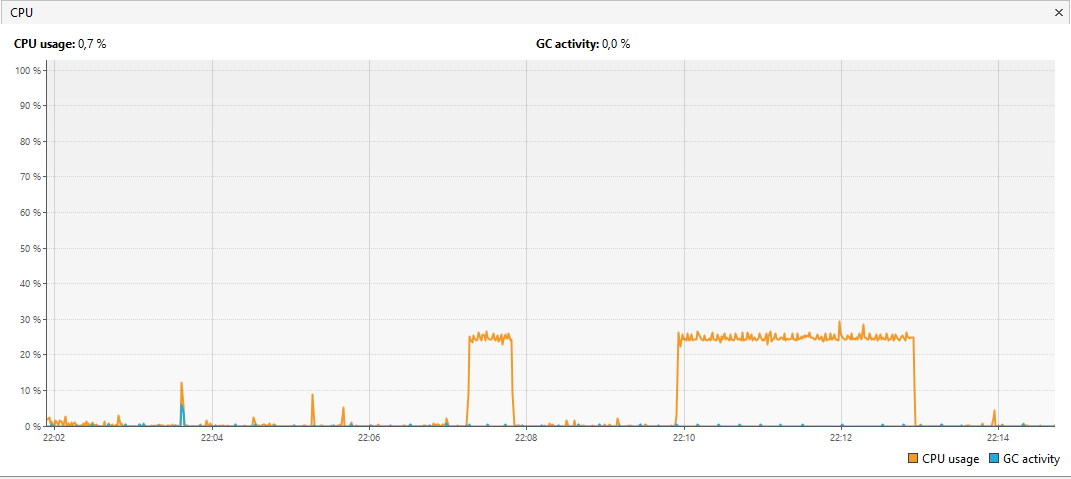
# Профилирование Garbage Collector – ZGC

Общий график Heap, на котором выделены сортировки:



Пиковый скачек Used heap до 16 Mb в момент запуска приложения.

График загрузки процессора на каждой сборке (название сборок см. на графике выше). Сортировка пузырьком потребовала больше всего ресурсов процессора:



Нагрузка на процессор составила около 30%.

Введите количество элементов

250000

Массив создан и заполнен случайными значениями от 1 до 250000

**MergeSort**

MergeSort Начало сортировки - 22:03:36.052268200

MergeSort Конец сортировки - 22:03:36.227162

**Вывод**: на сортировку MergeSort затрачено **175 мс** против **96 мс** для G1**.** Used heap пиково увеличился на 20% (с 10,5 Mb до 12,5 Mb), затем снова вернулся на уровень 10,5 Mb. График стабилен. Идут малые сборки.

**InsertSort**

InsertSort Начало сортировки - 22:07:15.017031600

InsertSort Конец сортировки - 22:07:48.775062200

**Вывод**: на сортировку InsertSort затрачено **33 сек**. Used heap пиково увеличился на 20% (с 10,5 Mb до 12,5 Mb), затем снова вернулся на уровень 10,5 Mb. График стабилен. Идут малые сборки.

**BubbleSort**

BubbleSort Начало сортировки - 22:09:54.344658800

BubbleSort Конец сортировки - 22:12:54.953782

**Вывод**: на сортировку BubbleSort затрачено около **3 мин**. Heap size обвалился с 24 Mb до 14,5 Mb, затем восстановился до уровня 18 Mb. Used heap пиково увеличился на 20% (с 12,5 Mb до 16 Mb), затем снова вернулся на уровень 10,5 Mb. По окончании сортировки отсутствовала Full (полная) сборка и Used heap остался неизменным, составив 10,5 Mb.

Лог:

[0.029s][info][gc] Using The Z Garbage Collector  
[0.409s][info][gc] GC(0) Garbage Collection (Warmup) 4M(17%)->4M(17%)  
[1.127s][info][gc] GC(1) Garbage Collection (Warmup) 6M(25%)->6M(25%)  
[5.324s][info][gc] GC(2) Garbage Collection (Warmup) 8M(33%)->6M(25%)  
[5.674s][info][gc] GC(3) Garbage Collection (Allocation Rate) 14M(58%)->10M(42%)  
[5.758s][info][gc] GC(4) Garbage Collection (Allocation Rate) 12M(50%)->8M(33%)  
[5.824s][info][gc] GC(5) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[5.910s][info][gc] GC(6) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[6.035s][info][gc] GC(7) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[6.145s][info][gc] GC(8) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[6.230s][info][gc] GC(9) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[6.322s][info][gc] GC(10) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[9.710s][info][gc] GC(11) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[28.708s][info][gc] GC(12) Garbage Collection (Proactive) 14M(58%)->8M(33%)  
[40.714s][info][gc] GC(13) Garbage Collection (Proactive) 12M(50%)->6M(25%)  
[52.721s][info][gc] GC(14) Garbage Collection (Proactive) 10M(42%)->6M(25%)  
[65.709s][info][gc] GC(15) Garbage Collection (Proactive) 10M(42%)->6M(25%)  
[76.711s][info][gc] GC(16) Garbage Collection (Proactive) 10M(42%)->6M(25%)  
[80.212s][info][gc] GC(17) Garbage Collection (Proactive) 10M(42%)->8M(33%)  
[90.714s][info][gc] GC(18) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[103.816s][info][gc] GC(19) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[108.832s][info][gc] GC(20) Garbage Collection (Proactive) 14M(58%)->14M(58%)  
[108.902s][info][gc] Allocation Stall (main) 24.513ms  
[108.905s][info][gc] GC(21) Garbage Collection (Allocation Stall) 24M(100%)->14M(58%)  
[108.942s][info][gc] Allocation Stall (main) 21.971ms  
[108.950s][info][gc] GC(22) Garbage Collection (Allocation Stall) 24M(100%)->18M(75%)  
[109.024s][info][gc] GC(23) Garbage Collection (Allocation Rate) 18M(75%)->8M(33%)  
[109.112s][info][gc] GC(24) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.217s][info][gc] GC(25) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.309s][info][gc] GC(26) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.403s][info][gc] GC(27) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.511s][info][gc] GC(28) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.603s][info][gc] GC(29) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[109.720s][info][gc] GC(30) Garbage Collection (Allocation Rate) 10M(42%)->10M(42%)  
[110.119s][info][gc] GC(31) Garbage Collection (Allocation Rate) 10M(42%)->8M(33%)  
[110.211s][info][gc] GC(32) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[110.315s][info][gc] GC(33) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[110.408s][info][gc] GC(34) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[110.517s][info][gc] GC(35) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[110.609s][info][gc] GC(36) Garbage Collection (Allocation Rate) 8M(33%)->8M(33%)  
[122.805s][info][gc] GC(37) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[134.822s][info][gc] GC(38) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[149.826s][info][gc] GC(39) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[164.825s][info][gc] GC(40) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[179.812s][info][gc] GC(41) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[194.805s][info][gc] GC(42) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[209.804s][info][gc] GC(43) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[224.806s][info][gc] GC(44) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[237.813s][info][gc] GC(45) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[252.822s][info][gc] GC(46) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[267.912s][info][gc] GC(47) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[282.920s][info][gc] GC(48) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[297.914s][info][gc] GC(49) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[310.922s][info][gc] GC(50) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[326.215s][info][gc] GC(51) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[328.514s][info][gc] GC(52) Garbage Collection (Proactive) 12M(50%)->10M(42%)  
[343.010s][info][gc] GC(53) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[356.028s][info][gc] GC(54) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[369.009s][info][gc] GC(55) Garbage Collection (Proactive) 14M(58%)->8M(33%)  
[384.008s][info][gc] GC(56) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[397.016s][info][gc] GC(57) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[412.114s][info][gc] GC(58) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[427.216s][info][gc] GC(59) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[443.118s][info][gc] GC(60) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[459.113s][info][gc] GC(61) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[473.116s][info][gc] GC(62) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[487.126s][info][gc] GC(63) Garbage Collection (Proactive) 12M(50%)->10M(42%)  
[503.119s][info][gc] GC(64) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[519.115s][info][gc] GC(65) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[535.113s][info][gc] GC(66) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[550.233s][info][gc] GC(67) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[566.114s][info][gc] GC(68) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[582.228s][info][gc] GC(69) Garbage Collection (Proactive) 14M(58%)->12M(50%)  
[612.225s][info][gc] GC(70) Garbage Collection (Proactive) 16M(67%)->12M(50%)  
[644.219s][info][gc] GC(71) Garbage Collection (Proactive) 16M(67%)->10M(42%)  
[659.319s][info][gc] GC(72) Garbage Collection (Proactive) 14M(58%)->10M(42%)  
[673.206s][info][gc] GC(73) Garbage Collection (Proactive) 14M(58%)->8M(33%)  
[688.315s][info][gc] GC(74) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[704.213s][info][gc] GC(75) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[719.316s][info][gc] GC(76) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[735.208s][info][gc] GC(77) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[750.313s][info][gc] GC(78) Garbage Collection (Proactive) 12M(50%)->8M(33%)  
[766.319s][info][gc] GC(79) Garbage Collection (Proactive) 12M(50%)->8M(33%)

**Анализ Лога:**

Задержки времени отсутствуют. Выполнил всю работу параллельно с работой приложения. Более требователен по ресурсу Heap.