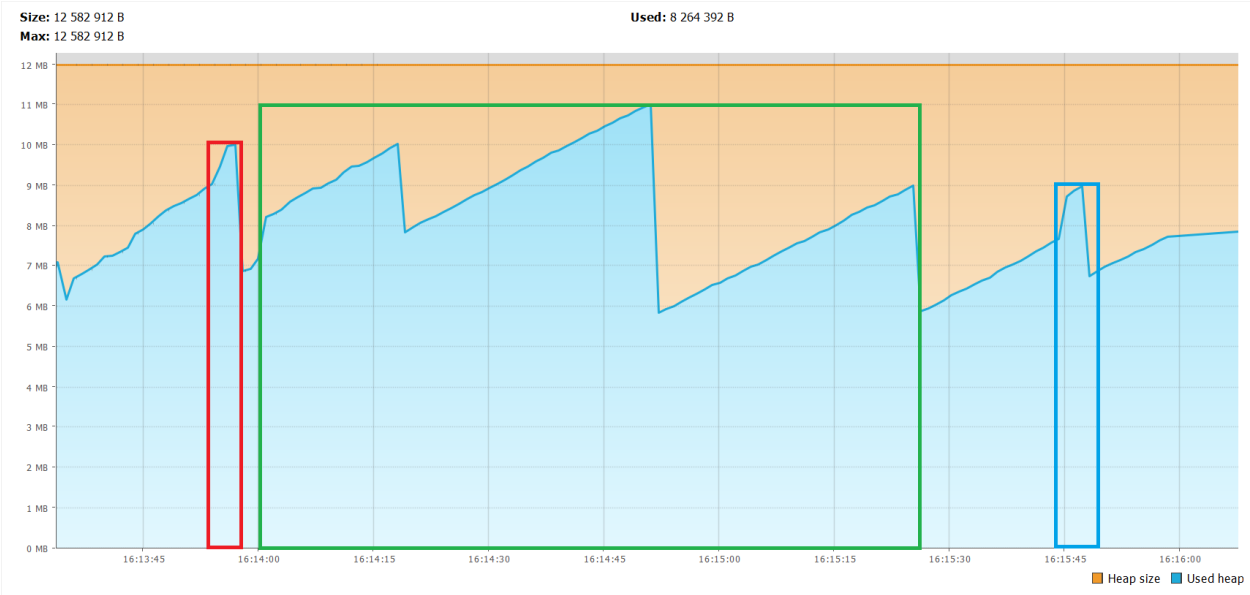
Serial GC



Участки на графике:

- Красный – сортировка слиянием (16:13:54 – 16:13:54)
- Зеленый – сортировка пузырьком (16:14:00 – 16:15:17)
- Синий – сортировка вставками (16:15:45 – 16:15:56)

Из графика видно, что сортировка пузырьком, по сравнению с сортировкой вставками, при сопоставимой оценке сложности $O(n^2)$ заняла значительно больше времени – 77 секунд против 11 секунд для одних и тех же исходных данных.

Именно для пузырьковой сортировки на графике проявились три пиковых значения занимаемой памяти в хипе, соответствующие Pause Full событиям в логах (ниже выделены красным), вызвавшим значительную задержку в выполнении алгоритма.

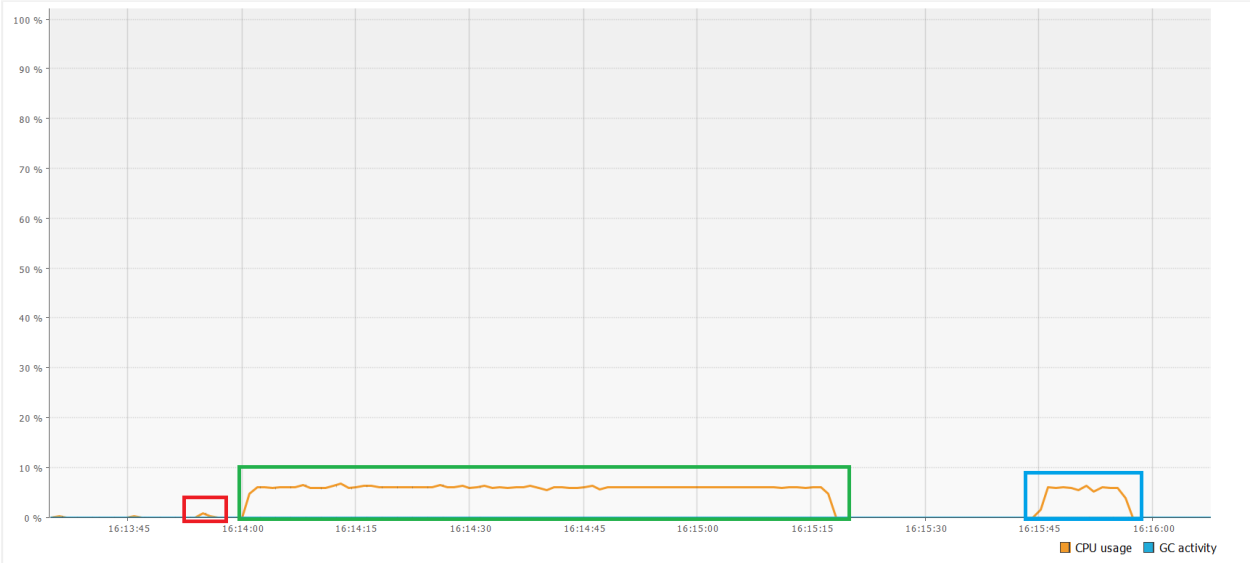
Лог GC:

```
[0.006s][info][gc] Using Serial
[1.046s][info][gc] GC(0) Pause Young (Allocation Failure) 3M->1M(11M) 3.125ms
```

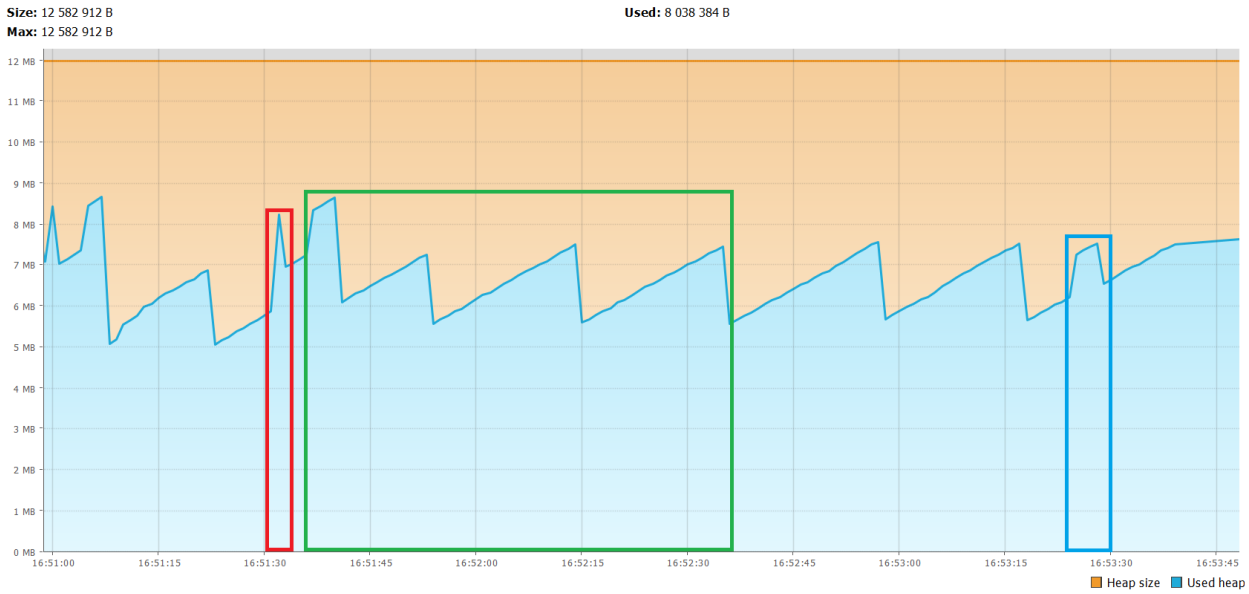
```
[20.680s][info][gc] GC(1) Pause Young (Allocation Failure) 5M->3M(11M) 2.725ms
[20.728s][info][gc] GC(2) Pause Young (Allocation Failure) 6M->3M(11M) 1.667ms
[20.766s][info][gc] GC(3) Pause Young (Allocation Failure) 6M->4M(11M) 2.684ms
[20.785s][info][gc] GC(4) Pause Young (Allocation Failure) 7M->4M(11M) 1.335ms
[21.054s][info][gc] GC(5) Pause Young (Allocation Failure) 8M->5M(11M) 2.687ms
[21.687s][info][gc] GC(6) Pause Young (Allocation Failure) 8M->6M(11M) 2.651ms
[41.533s][info][gc] GC(7) Pause Young (Allocation Failure) 9M->6M(11M) 1.957ms
[41.539s][info][gc] GC(8) Pause Young (Allocation Failure) 9M->7M(11M) 1.100ms
[41.571s][info][gc] GC(9) Pause Young (Allocation Failure) 10M->10M(11M) 0.030ms
[41.584s][info][gc] GC(10) Pause Full (Allocation Failure) 10M->5M(11M) 12.898ms
[41.590s][info][gc] GC(11) Pause Young (Allocation Failure) 9M->6M(11M) 0.434ms
[41.593s][info][gc] GC(12) Pause Young (Allocation Failure) 9M->6M(11M) 0.193ms
[41.597s][info][gc] GC(13) Pause Young (Allocation Failure) 9M->6M(11M) 0.186ms
[41.600s][info][gc] GC(14) Pause Young (Allocation Failure) 10M->10M(11M) 0.017ms
[41.609s][info][gc] GC(15) Pause Full (Allocation Failure) 10M->6M(11M) 9.088ms
[41.612s][info][gc] GC(16) Pause Young (Allocation Failure) 9M->6M(11M) 0.147ms
[41.615s][info][gc] GC(17) Pause Young (Allocation Failure) 10M->6M(11M) 0.145ms
[45.205s][info][gc] GC(18) Pause Young (Allocation Failure) 10M->6M(11M) 0.807ms
```

```
[66.103s][info][gc] GC(19) Pause Young (Allocation Failure) 10M->7M(11M) 1.193ms
[98.483s][info][gc] GC(20) Pause Young (Allocation Failure) 11M->11M(11M) 0.057ms
[98.497s][info][gc] GC(21) Pause Full (Allocation Failure) 11M->5M(11M) 14.131ms
[132.556s][info][gc] GC(22) Pause Young (Allocation Failure) 9M->5M(11M) 0.410ms
[155.568s][info][gc] GC(23) Pause Young (Allocation Failure) 9M->6M(11M) 0.748ms
```

График использования CPU:



Parallel GC



Участки на графике:

- Красный – сортировка слиянием (16:51:31 – 16:51:31)
- Зеленый – сортировка пузырьком (16:51:36 – 16:52:52)
- Синий – сортировка вставками (16:53:24 – 16:53:36)

В данном примере пузырьковая сортировка снова выполнилась медленнее всех – за 76 секунд (что практически совпадает с результатом для Serial GC).

Однако в этот раз не произошло пауз «Pause Full» ни для одного из алгоритмов.

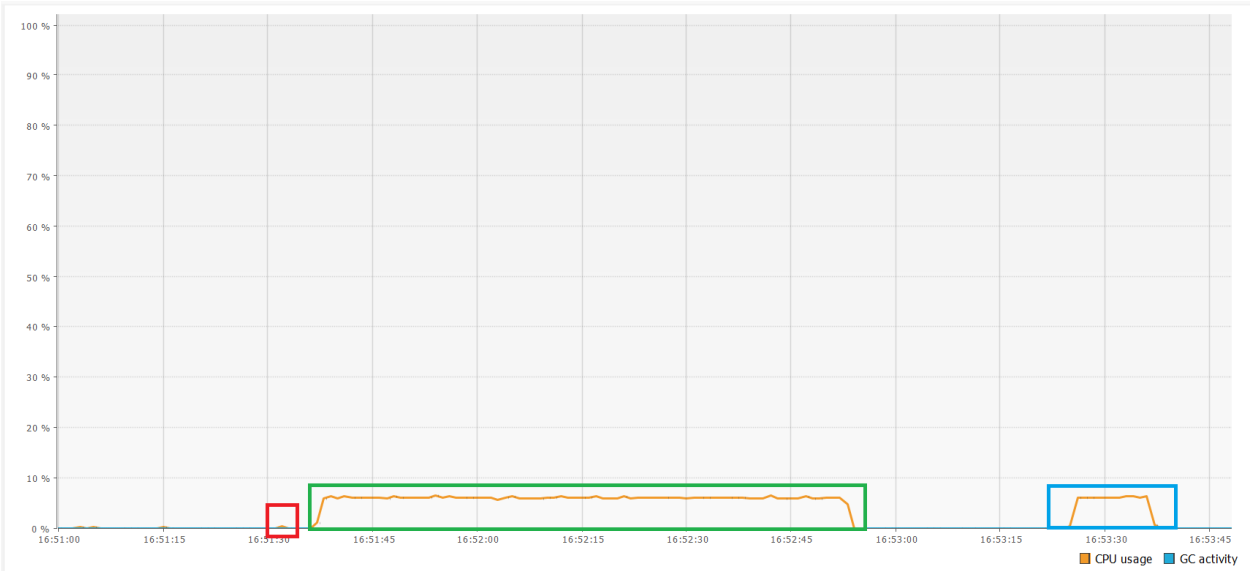
В самом худшем случае для пузырьковой сортировки, размер занимаемой памяти в хипе был около 9Мб против 11Мб в пике для случая с Serial GC.

Лог GC:

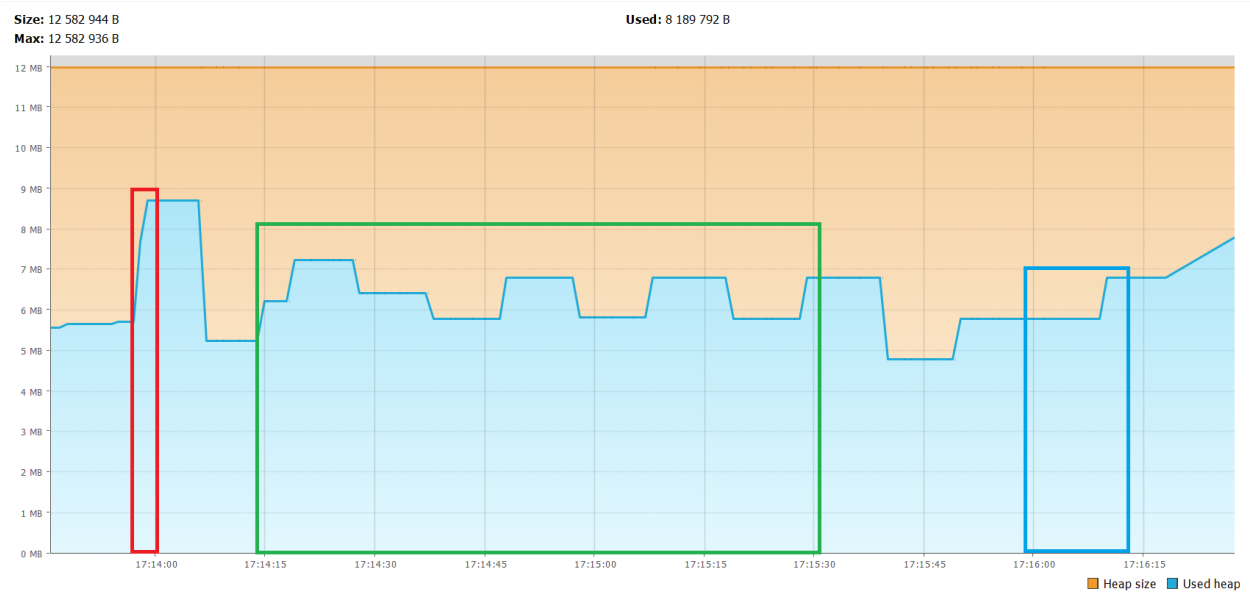
```
[0.007s][info][gc] Using Parallel
[1.039s][info][gc] GC(0) Pause Young (Allocation Failure) 3M-
```

```
>1M(11M) 2.029ms
[3.089s][info][gc] GC(1) Pause Young (Allocation Failure) 4M-
>2M(11M) 1.568ms
[3.146s][info][gc] GC(2) Pause Young (Allocation Failure) 5M-
>2M(11M) 1.346ms
[3.180s][info][gc] GC(3) Pause Young (Allocation Failure) 5M-
>3M(11M) 1.925ms
[3.196s][info][gc] GC(4) Pause Young (Allocation Failure) 6M-
>5M(11M) 2.462ms
[3.473s][info][gc] GC(5) Pause Young (Allocation Failure) 8M-
>6M(10M) 2.459ms
[3.574s][info][gc] GC(6) Pause Young (Allocation Failure) 8M-
>6M(11M) 1.946ms
[4.620s][info][gc] GC(7) Pause Young (Allocation Failure) 8M-
>6M(11M) 1.202ms
[15.634s][info][gc] GC(8) Pause Young (Allocation Failure) 8M-
>6M(11M) 1.293ms
```

График использования CPU



G1 GC



Участки на графике:

- Красный – сортировка слиянием (17:13:57 – 17:13:57)
- Зеленый – сортировка пузырьком (17:14:14 – 17:15:30)
- Синий – сортировка вставками (17:15:58 – 17:16:09)

Как видно из графика и таймкодов, время выполнения каждого алгоритма не изменилось.

Однако, очевидным является более экономное использование памяти каждым из алгоритмов.

Так, для G1 худшим случаем по памяти оказался алгоритм сортировки слиянием – чуть менее 9Мб в пике из доступных 12Мб. Как-раз в районе этого пика последовательно наступили события «Concurrent Mark Cycle», «Pause Remark», «Pause Cleanup», «Concurrent Mark Cycle», «Pause Young (Prepare Mixed)» (В логе выделено красным).

Аналогичные последовательности событий возникли и в пиковых значениях используемой памяти для двух других алгоритмов.

Лог GC:

```
[0.006s][info][gc] Using G1
[14.394s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 5M->2M(12M) 2.990ms
[14.425s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 4M->3M(12M) 1.873ms
[14.439s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 4M->3M(12M) 0.738ms
[14.476s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 5M->3M(12M) 1.285ms
[14.509s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 5M->3M(12M) 1.110ms
[14.521s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 5M->5M(12M) 1.598ms
[14.530s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 1.605ms
[14.535s][info][gc] GC(7) Pause Young (Concurrent Start) (G1 Evacuation Pause) 6M->6M(12M) 1.007ms
[14.544s][info][gc] GC(8) Concurrent Mark Cycle
[14.556s][info][gc] GC(8) Pause Remark 6M->6M(12M) 1.645ms
[14.557s][info][gc] GC(8) Pause Cleanup 6M->6M(12M) 0.027ms
[14.558s][info][gc] GC(8) Concurrent Mark Cycle 13.716ms
[14.585s][info][gc] GC(9) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 7M->5M(12M) 1.789ms
[14.713s][info][gc] GC(10) Pause Young (Mixed) (G1 Evacuation Pause) 6M->5M(12M) 1.474ms
[14.747s][info][gc] GC(11) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 1.079ms
[14.800s][info][gc] GC(12) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 1.350ms
[14.843s][info][gc] GC(13) Pause Young (Concurrent Start) (G1 Evacuation Pause) 6M->5M(12M) 1.245ms
[14.843s][info][gc] GC(14) Concurrent Mark Cycle
[14.846s][info][gc] GC(14) Pause Remark 5M->5M(12M) 1.264ms
[14.847s][info][gc] GC(14) Pause Cleanup 6M->6M(12M) 0.022ms
```

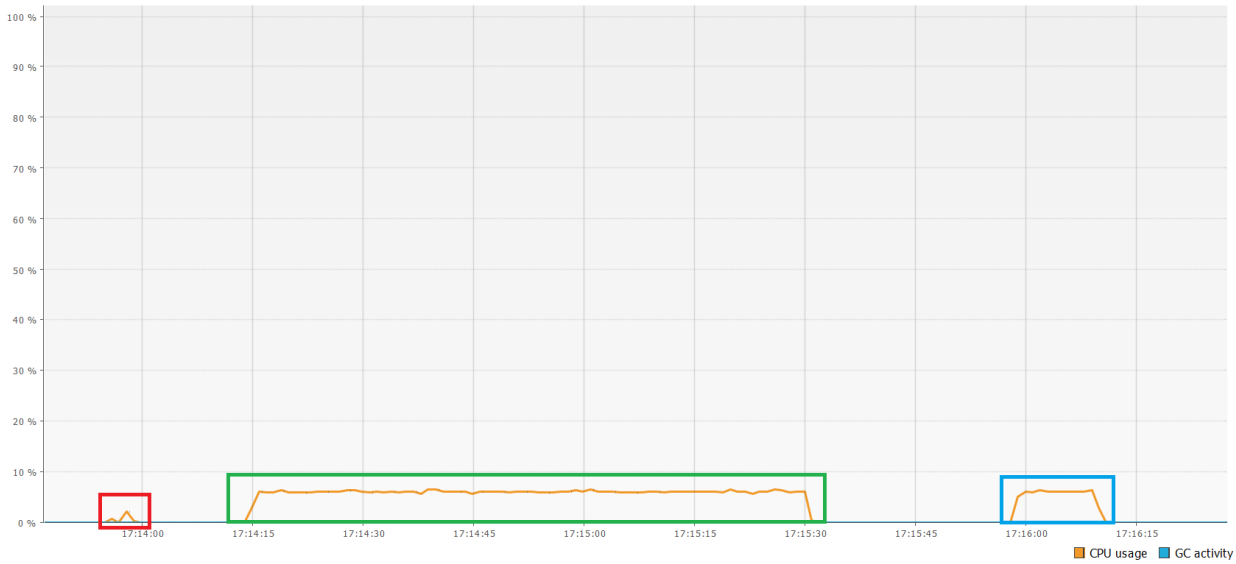
[14.847s][info][gc] GC(14) Concurrent Mark Cycle 4.402ms
[14.862s][info][gc] GC(15) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 6M->6M(12M) 1.100ms
[14.883s][info][gc] GC(16) Pause Young (Mixed) (G1 Evacuation Pause) 7M->5M(12M) 1.722ms
[15.910s][info][gc] GC(17) Pause Young (Concurrent Start) (G1 Evacuation Pause) 6M->5M(12M) 1.126ms
[15.910s][info][gc] GC(18) Concurrent Mark Cycle
[15.915s][info][gc] GC(18) Pause Remark 5M->5M(12M) 1.648ms
[15.916s][info][gc] GC(18) Pause Cleanup 5M->5M(12M) 0.029ms
[15.916s][info][gc] GC(18) Concurrent Mark Cycle 5.797ms
[22.918s][info][gc] GC(19) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 6M->5M(12M) 1.161ms
[25.977s][info][gc] GC(20) Pause Young (Mixed) (G1 Evacuation Pause) 7M->6M(12M) 2.088ms
[25.979s][info][gc] GC(21) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 1.042ms
[25.979s][info][gc] GC(22) Concurrent Mark Cycle
[25.982s][info][gc] GC(23) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.025ms
[25.988s][info][gc] GC(22) Pause Remark 6M->6M(12M) 3.743ms
[25.989s][info][gc] GC(22) Pause Cleanup 6M->6M(12M) 0.029ms
[25.989s][info][gc] GC(22) Concurrent Mark Cycle 9.836ms
[25.994s][info][gc] GC(24) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 7M->6M(12M) 0.933ms
[26.004s][info][gc] GC(25) Pause Young (Mixed) (G1 Evacuation Pause) 7M->6M(12M) 2.112ms
[26.011s][info][gc] GC(26) Pause Young (Concurrent Start) (G1 Evacuation Pause) 8M->6M(12M) 0.836ms
[26.011s][info][gc] GC(27) Concurrent Mark Cycle
[26.013s][info][gc] GC(28) Pause Young (Normal) (G1 Evacuation Pause) 8M->6M(12M) 0.666ms
[26.016s][info][gc] GC(27) Pause Remark 7M->7M(12M) 1.989ms
[26.018s][info][gc] GC(27) Pause Cleanup 8M->8M(12M) 0.035ms
[26.018s][info][gc] GC(27) Concurrent Mark Cycle 6.993ms
[26.018s][info][gc] GC(29) Pause Young (Prepare Mixed) (G1

```
Evacuation Pause) 8M->6M(12M) 0.723ms
[26.021s][info][gc] GC(30) Pause Young (Mixed) (G1 Evacuation
Pause) 7M->6M(12M) 1.722ms
[26.024s][info][gc] GC(31) Pause Young (Concurrent Start) (G1
Evacuation Pause) 7M->6M(12M) 0.466ms
[26.024s][info][gc] GC(32) Concurrent Mark Cycle
[26.029s][info][gc] GC(32) Pause Remark 7M->7M(12M) 2.713ms
[26.030s][info][gc] GC(33) Pause Young (Normal) (G1 Evacuation
Pause) 8M->6M(12M) 0.451ms
[26.030s][info][gc] GC(32) Pause Cleanup 6M->6M(12M) 0.021ms
[26.030s][info][gc] GC(32) Concurrent Mark Cycle 6.777ms
[26.031s][info][gc] GC(34) Pause Young (Normal) (G1 Evacuation
Pause) 7M->6M(12M) 0.409ms
[26.033s][info][gc] GC(35) Pause Young (Concurrent Start) (G1
Evacuation Pause) 7M->6M(12M) 0.324ms
[26.033s][info][gc] GC(36) Concurrent Mark Cycle
[26.034s][info][gc] GC(37) Pause Young (Normal) (G1 Evacuation
Pause) 7M->6M(12M) 0.283ms
[26.038s][info][gc] GC(36) Pause Remark 7M->7M(12M) 2.652ms
[26.038s][info][gc] GC(38) Pause Young (Normal) (G1 Evacuation
Pause) 7M->6M(12M) 0.260ms
[26.039s][info][gc] GC(36) Pause Cleanup 7M->7M(12M) 0.018ms
[26.039s][info][gc] GC(36) Concurrent Mark Cycle 6.653ms
[26.040s][info][gc] GC(39) Pause Young (Prepare Mixed) (G1
Evacuation Pause) 7M->6M(12M) 0.208ms
[26.042s][info][gc] GC(40) Pause Young (Mixed) (G1 Evacuation
Pause) 7M->6M(12M) 0.984ms
[26.043s][info][gc] GC(41) Pause Young (Concurrent Start) (G1
Evacuation Pause) 7M->6M(12M) 0.322ms
[26.043s][info][gc] GC(42) Concurrent Mark Cycle
[26.047s][info][gc] GC(42) Pause Remark 8M->8M(12M) 1.614ms
[26.048s][info][gc] GC(43) Pause Young (Normal) (G1 Evacuation
Pause) 8M->6M(12M) 0.344ms
[26.048s][info][gc] GC(42) Pause Cleanup 6M->6M(12M) 0.018ms
[26.048s][info][gc] GC(42) Concurrent Mark Cycle 5.061ms
[26.049s][info][gc] GC(44) Pause Young (Normal) (G1 Evacuation
```

Pause) 7M->6M(12M) 0.253ms
[26.050s][info][gc] GC(45) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 0.350ms
[26.050s][info][gc] GC(46) Concurrent Mark Cycle
[26.054s][info][gc] GC(46) Pause Remark 8M->8M(12M) 1.653ms
[26.055s][info][gc] GC(46) Pause Cleanup 8M->8M(12M) 0.020ms
[26.055s][info][gc] GC(46) Concurrent Mark Cycle 4.844ms
[35.905s][info][gc] GC(47) Pause Young (Normal) (G1 Preventive Collection) 9M->5M(12M) 0.785ms
[43.458s][info][gc] GC(48) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->5M(12M) 0.859ms
[43.458s][info][gc] GC(49) Concurrent Mark Cycle
[43.464s][info][gc] GC(49) Pause Remark 6M->6M(12M) 2.929ms
[43.465s][info][gc] GC(49) Pause Cleanup 6M->6M(12M) 0.026ms
[43.465s][info][gc] GC(49) Concurrent Mark Cycle 6.786ms
[56.917s][info][gc] GC(50) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 8M->6M(12M) 0.813ms
[66.928s][info][gc] GC(51) Pause Young (Mixed) (G1 Evacuation Pause) 7M->5M(12M) 1.505ms
[85.961s][info][gc] GC(52) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->5M(12M) 0.692ms
[85.961s][info][gc] GC(53) Concurrent Mark Cycle
[85.965s][info][gc] GC(53) Pause Remark 5M->5M(12M) 1.928ms
[85.966s][info][gc] GC(53) Pause Cleanup 5M->5M(12M) 0.019ms
[85.966s][info][gc] GC(53) Concurrent Mark Cycle 5.152ms
[106.980s][info][gc] GC(54) Pause Young (Normal) (G1 Evacuation Pause) 7M->5M(12M) 0.711ms
[128.000s][info][gc] GC(55) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->4M(12M) 0.836ms
[128.001s][info][gc] GC(56) Concurrent Mark Cycle
[128.005s][info][gc] GC(56) Pause Remark 4M->4M(12M) 2.069ms
[128.006s][info][gc] GC(56) Pause Cleanup 4M->4M(12M) 0.017ms
[128.006s][info][gc] GC(56) Concurrent Mark Cycle 5.405ms
[147.200s][info][gc] GC(57) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->4M(12M) 1.028ms
[147.200s][info][gc] GC(58) Concurrent Mark Cycle

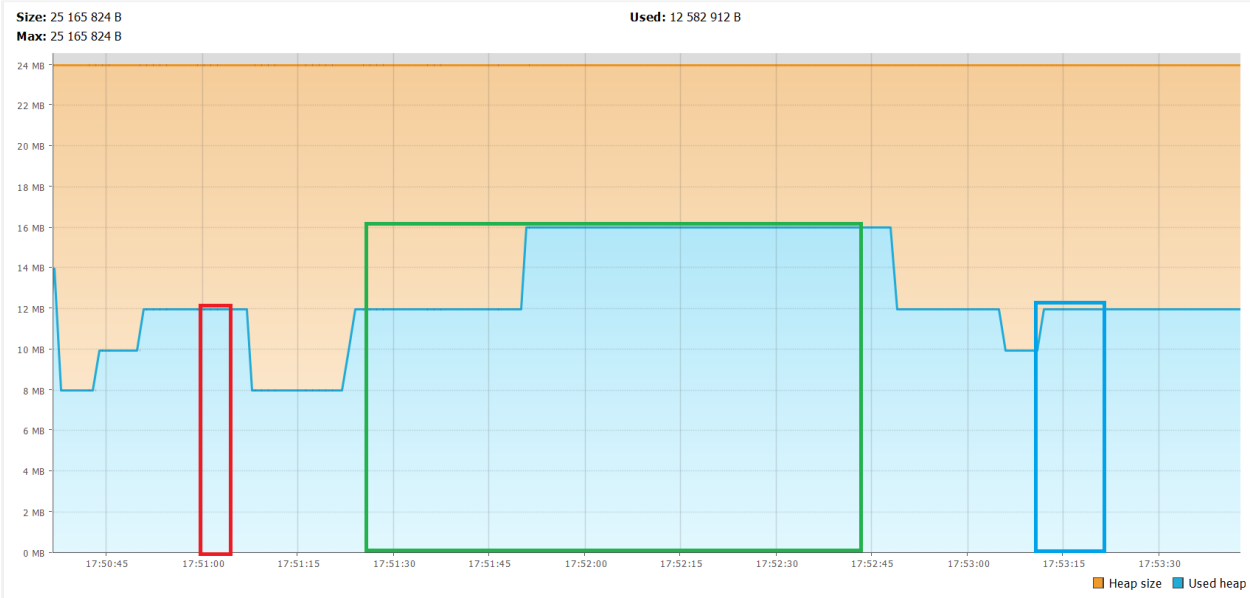
```
[147.205s][info][gc] GC(58) Pause Remark 5M->5M(12M) 2.283ms
[147.206s][info][gc] GC(58) Pause Cleanup 5M->5M(12M) 0.023ms
[147.206s][info][gc] GC(58) Concurrent Mark Cycle 5.954ms
```

График использования CPU



ZGC

Для данного сборщика мусора размер хипа был увеличен до 24Мб, т.к. при выделенных 12Мб приложение падает с OutOfMemoryError.



Участки на графике:

- Красный – сортировка слиянием (17:51:07 – 17:51:07)
- Зеленый – сортировка пузырьком (17:51:23 – 17:52:42)
- Синий – сортировка вставками (17:53:11 – 17:53:21)

Из графика видно, что распределение памяти стало более равномерным. В отличие от «пилы» с явно выраженными частыми пиками и падениями в предыдущих примерах, в данном примере каждое увеличение размера памяти выводит занятый объем на плато. После прохождения максимальных значений высвобождение памяти происходит практически мгновенно.

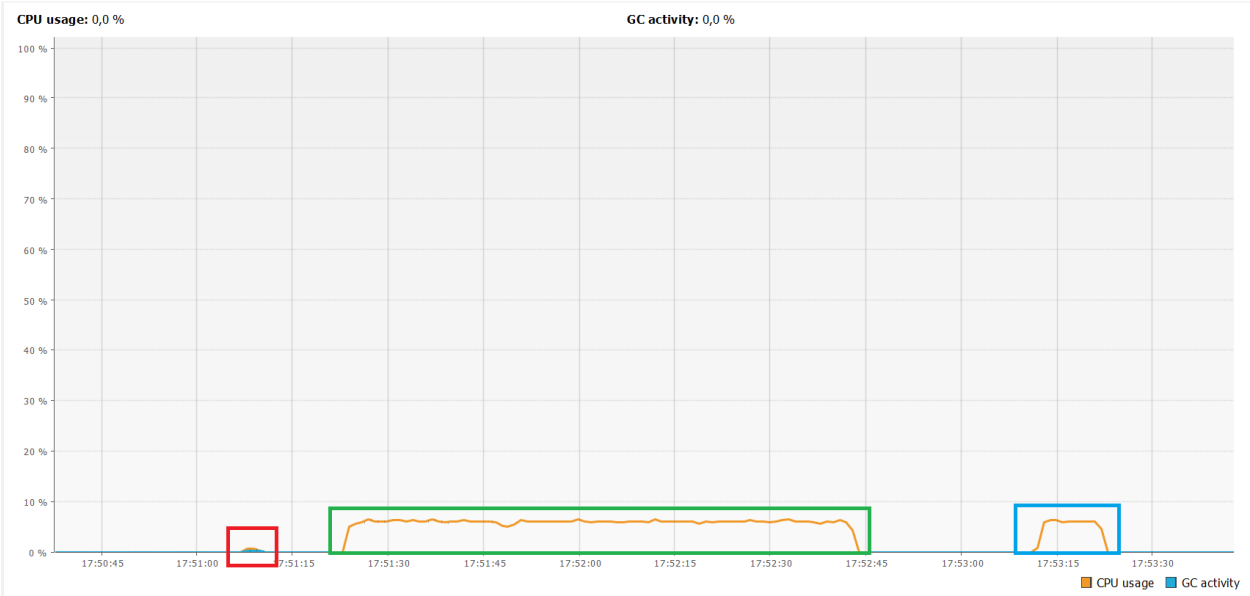
Из лога видно, что на протяжении выполнения всех трех алгоритмов не возникло ни одной паузы.

Лог GC:

```
[0.007s][info][gc] Using The Z Garbage Collector
```

```
    [0.134s][info][gc] GC(0) Garbage Collection (Warmup) 4M(17%)->2M(8%)
    [2.444s][info][gc] GC(1) Garbage Collection (Warmup) 6M(25%)->6M(25%)
    [2.543s][info][gc] GC(2) Garbage Collection (Warmup) 14M(58%)->10M(42%)
    [2.646s][info][gc] GC(3) Garbage Collection (Allocation Rate) 14M(58%)->6M(25%)
    [2.832s][info][gc] GC(4) Garbage Collection (Allocation Rate) 8M(33%)->6M(25%)
    [2.940s][info][gc] GC(5) Garbage Collection (Allocation Rate) 8M(33%)->6M(25%)
    [3.033s][info][gc] GC(6) Garbage Collection (Allocation Rate) 8M(33%)->6M(25%)
    [4.438s][info][gc] GC(7) Garbage Collection (Proactive) 10M(42%)->8M(33%)
    [5.350s][info][gc] GC(8) Garbage Collection (Proactive) 14M(58%)->10M(42%)
```

График использования CPU



Здесь стоит отметить, что для алгоритма сортировки слиянием в данном случае активность Z сборщика мусора возросла по сравнению с другими сборщиками мусора.