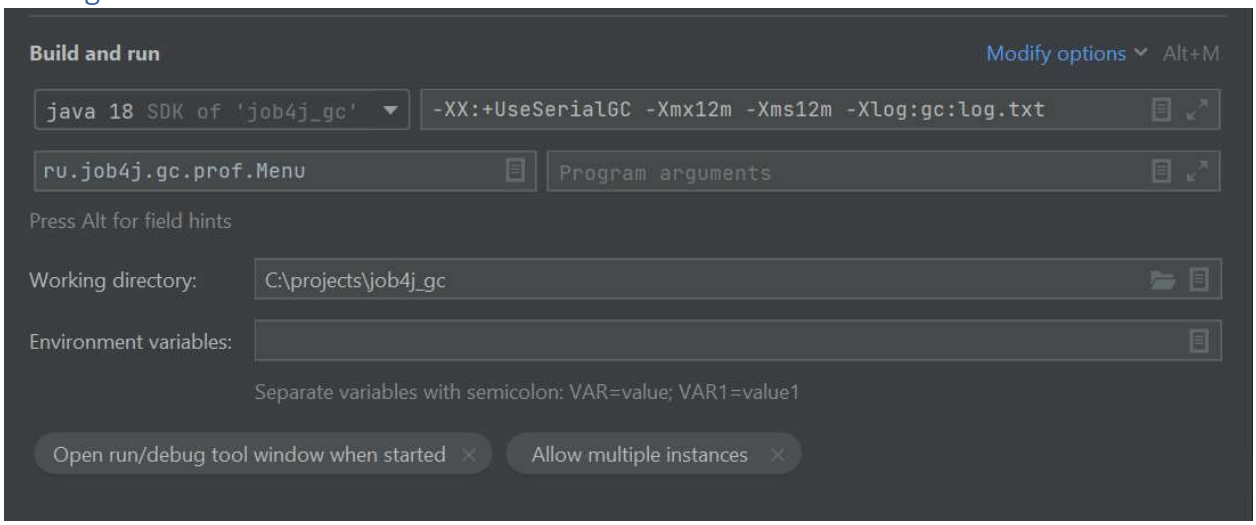
Contents

Serial

Settings



Total Heap = 12 MBt
Reserved = Xmx – xms = 0 MBt
Eden capacity = 3328 KBt
Survivor1 capacity = Survivor2 capacity = 384 KBt
Old Generation capacity = 8192 KBt

-jstat was run in parallel with VisualVM to see data move between different parts of the heap

Java internal process

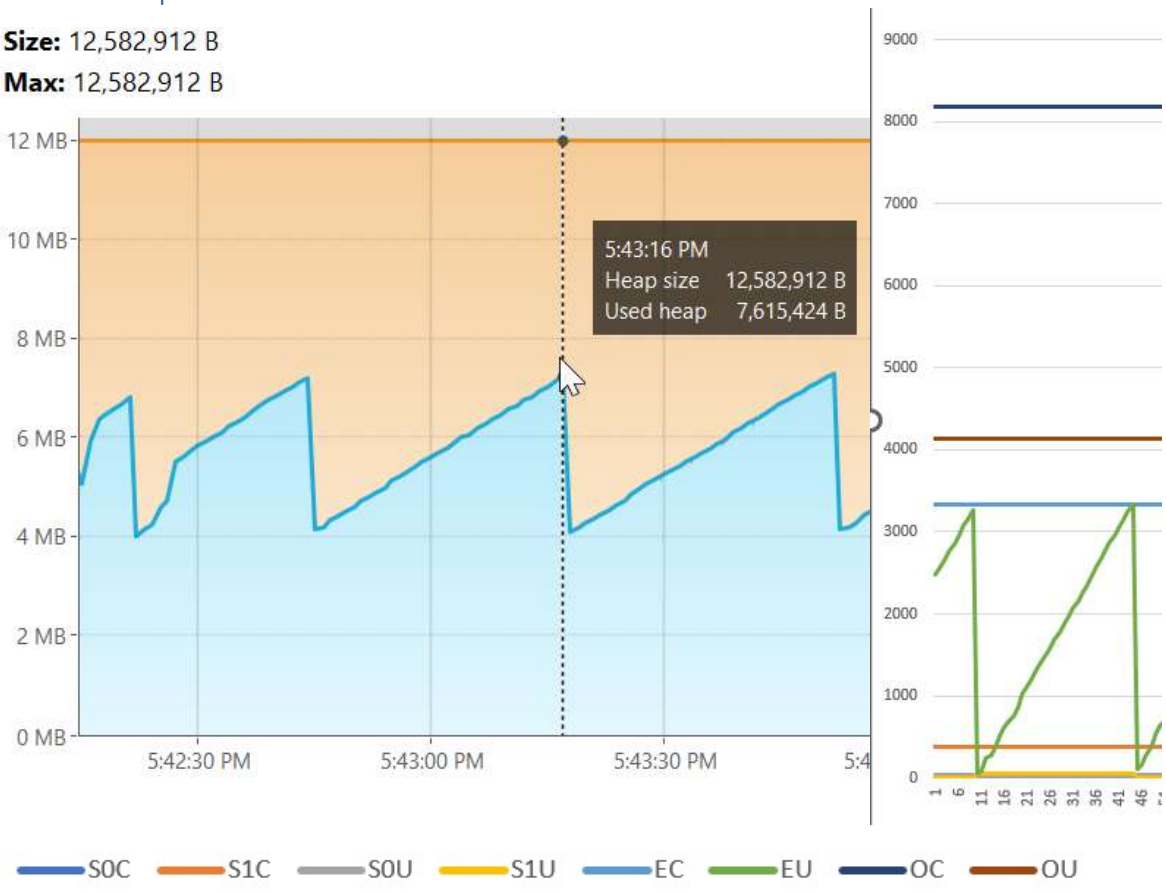
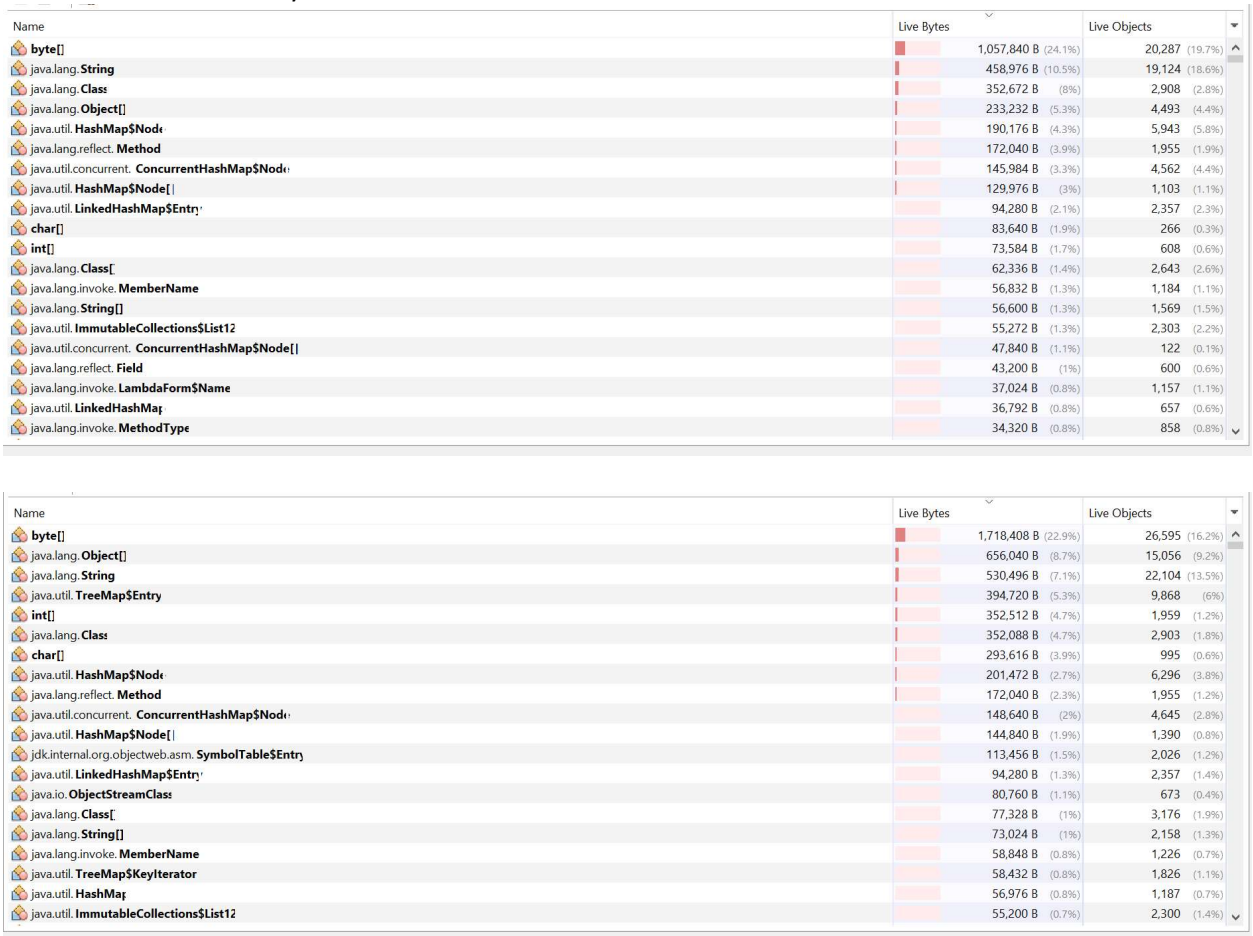


Figure 1 – Visual VM on the right top. Data from jstat on the left + history at the bottom

Before data addition, I've noticed that some processes already run in our application. Heap is filling with data periodically (every 35 seconds):

- Some data 4136 KBt from 8192 KBt are already occupy old generation
- Eden space is filling with bytes from 0 to 3328 KBt every 35 seconds. When Eden Used = Eden Capacity, the garbage collector delete the data. Most of the data are stored somewhere in java in byte[] array
- From 30 to 60 bytes data are toss from one survivor to another



Name	Live Bytes	Live Objects
byte[]	1,057,840 B (24.1%)	20,287 (19.7%)
java.lang.String	458,976 B (10.5%)	19,124 (18.6%)
java.lang.Class	352,672 B (8%)	2,908 (2.8%)
java.lang.Object[]	233,232 B (5.3%)	4,493 (4.4%)
java.util.HashMap$Node	190,176 B (4.3%)	5,943 (5.8%)
java.lang.reflect.Method	172,040 B (3.9%)	1,955 (1.9%)
java.util.concurrent.ConcurrentHashMap$Node	145,984 B (3.3%)	4,562 (4.4%)
java.util.HashMap$Node[]	129,976 B (3%)	1,103 (1.1%)
java.util.LinkedHashMap$Entry	94,280 B (2.1%)	2,357 (2.3%)
char[]	83,640 B (1.9%)	266 (0.3%)
int[]	73,584 B (1.7%)	608 (0.6%)
java.lang.Class[]	62,336 B (1.4%)	2,643 (2.6%)
java.lang.invoke.MemberName	56,832 B (1.3%)	1,184 (1.1%)
java.lang.String[]	56,600 B (1.3%)	1,569 (1.5%)
java.util.ImmutableCollections$List12	55,272 B (1.3%)	2,303 (2.2%)
java.util.concurrent.ConcurrentHashMap$Node[]	47,840 B (1.1%)	122 (0.1%)
java.lang.reflect.Field	43,200 B (1%)	600 (0.6%)
java.lang.invoke.LambdaForm$Name	37,024 B (0.8%)	1,157 (1.1%)
java.util.LinkedHashMap	36,792 B (0.8%)	657 (0.6%)
java.lang.invoke.MethodType	34,320 B (0.8%)	858 (0.8%)

Name	Live Bytes	Live Objects
byte[]	1,718,408 B (22.9%)	26,595 (16.2%)
java.lang.Object[]	656,040 B (8.7%)	15,056 (9.2%)
java.lang.String	530,496 B (7.1%)	22,104 (13.5%)
java.util.TreeMap$Entry	394,720 B (5.3%)	9,868 (6%)
int[]	352,512 B (4.7%)	1,959 (1.2%)
java.lang.Class	352,088 B (4.7%)	2,903 (1.8%)
char[]	293,616 B (3.9%)	995 (0.6%)
java.util.HashMap$Node	201,472 B (2.7%)	6,296 (3.8%)
java.lang.reflect.Method	172,040 B (2.3%)	1,955 (1.2%)
java.util.concurrent.ConcurrentHashMap$Node	148,640 B (2%)	4,645 (2.8%)
java.util.HashMap$Node[]	144,840 B (1.9%)	1,390 (0.8%)
jdk.internal.org.objectweb.asm.SymbolTable$Entry	113,456 B (1.5%)	2,026 (1.2%)
java.util.LinkedHashMap$Entry	94,280 B (1.3%)	2,357 (1.4%)
java.io.ObjectStreamClass	80,760 B (1.1%)	673 (0.4%)
java.lang.Class[]	77,328 B (1%)	3,176 (1.9%)
java.lang.String[]	73,024 B (1%)	2,158 (1.3%)
java.lang.invoke.MemberName	58,848 B (0.8%)	1,226 (0.7%)
java.util.TreeMap$KeyIterator	58,432 B (0.8%)	1,826 (1.1%)
java.util.HashMap	56,976 B (0.8%)	1,187 (0.7%)
java.util.ImmutableCollections$List12	55,200 B (0.7%)	2,300 (1.4%)

Figure 2 – While running in background mode, java collects some bytes/ objects/ string and classes in Eden. The garbage collector erase this data when there is no free space in Eden

Merge Sort

Description

250000 int values had been added to sort using merge algorithm
250000 * 4 bytes = 1000000 bytes = 1000 KBt

After adding these values:

- In line with VisualVM used heap increased from 4928784 bytes to 5985360 bytes. The difference equals to 1056576 which is about 1000000 bytes
- In line with VisualVM sampler snapshot, int arrays increased up to 1185152 bytes
- In line with jstat, used Eden increased from 695 to 1776.8. Difference equals to 1137 kBt, which is about 1 MBt. Used old generation didn't changed

After running Merge Sort:

- In line with VisualVM, used heap increased for 2.5 MBt
- In line with jstat, garbage collector was running: Eden had been free for 783 KBt; some data was moved into survivor 1 from survivor 0 and Eden; The most of the data (about 3 MBt) was moved into Old Generation. The heap in total increased for 2.3 MBt

- Int[] array had been increased in 5 times. Looks reasonable. Every time when we call merge() method, new int[] array is created
- CPU wasn't loaded too much – only for 1.9%
- Merge Sort start - 17:44:11.210902200, end - 17:44:11.292715400

After adding these values

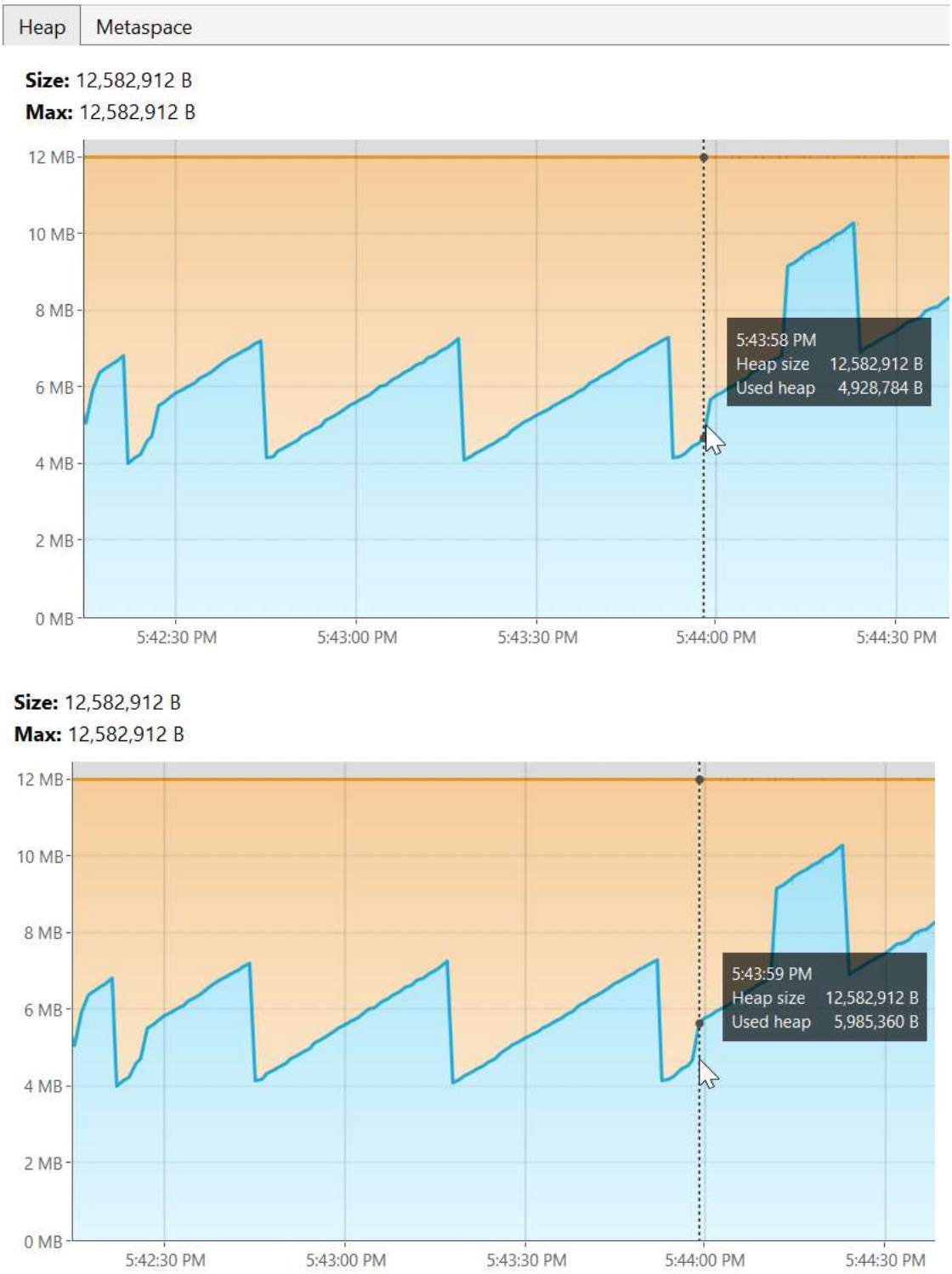


Figure 3 – 250000 values had been added. In line with VisualVM used heap increased from 4928784 bytes to 5985360 bytes. The difference equals to 1056576 which is about 1000000 bytes

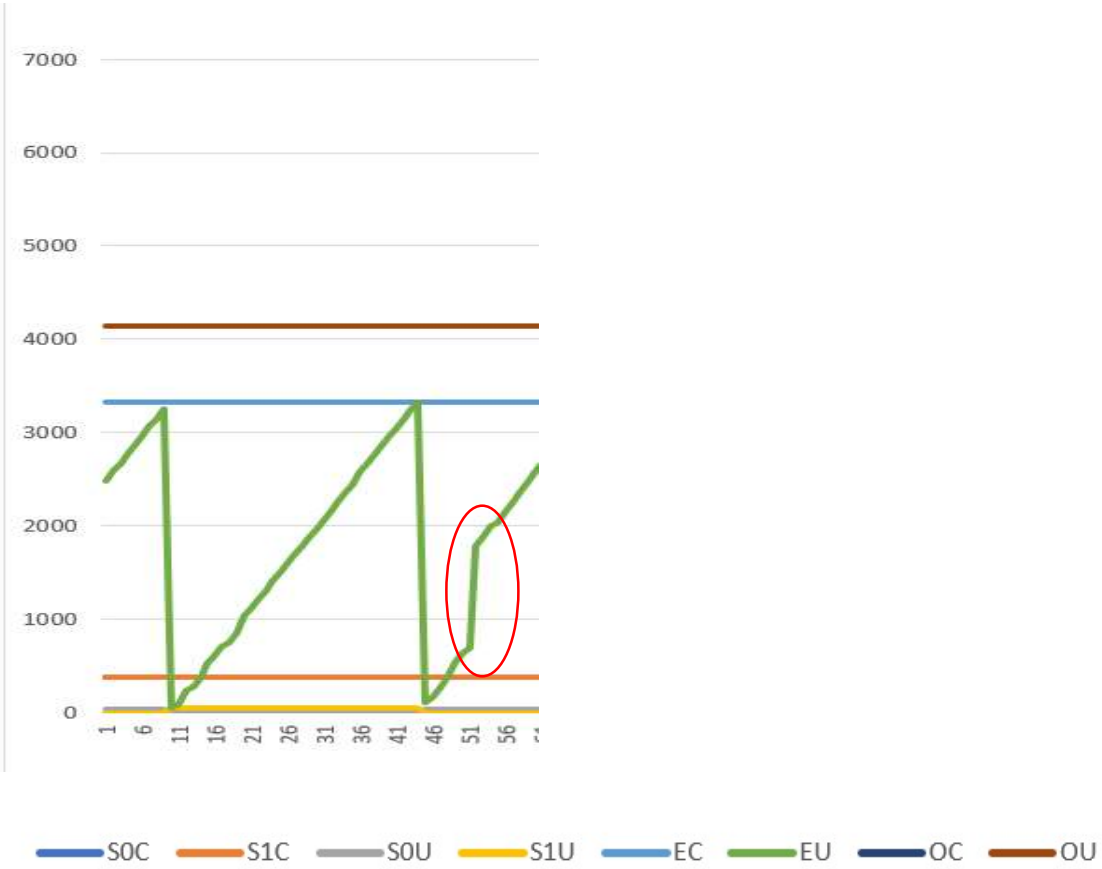


Figure 4 – 250000 values had been added. In line with jstat, used Eden increased from 695 to 1776.8. Difference equals to 1137 kBt, which is about 1 MBt

Name	Live Bytes	Live Objects
byte[]	1,301,160 B (20.1%)	22,693 (18.4%)
int[]	1,185,152 B (18.3%)	1,037 (0.8%)
java.lang.String	485,664 B (7.5%)	20,236 (16.4%)
java.lang.Class	352,784 B (5.4%)	2,909 (2.4%)
java.lang.Object[]	350,240 B (5.4%)	7,023 (5.7%)
java.util.HashMap$Node	195,552 B (3%)	6,111 (4.9%)
char[]	184,408 B (2.8%)	613 (0.5%)
java.lang.reflect.Method	172,040 B (2.7%)	1,955 (1.6%)
java.util.TreeMap$Entry	170,640 B (2.6%)	4,266 (3.5%)
java.util.concurrent.ConcurrentHashMap$Node	146,880 B (2.3%)	4,590 (3.7%)
java.util.HashMap$Node[]	136,984 B (2.1%)	1,238 (1%)
java.util.LinkedHashMap$Entry	94,280 B (1.5%)	2,357 (1.9%)
java.lang.Class[]	64,728 B (1%)	2,719 (2.2%)
java.lang.String[]	64,424 B (1%)	1,850 (1.5%)
java.lang.invoke.MemberName	56,976 B (0.9%)	1,187 (1%)
java.util.ImmutableCollections$List12	55,272 B (0.9%)	2,303 (1.9%)
java.util.concurrent.ConcurrentHashMap$Node[]	49,760 B (0.8%)	146 (0.1%)
java.lang.reflect.Field	43,200 B (0.7%)	600 (0.5%)
java.io.ObjectStreamClass	42,240 B (0.7%)	352 (0.3%)
java.util.HashMap	42,096 B (0.6%)	877 (0.7%)

Figure 5 – 250000 values had been added. In line with VisualVM sampler snapshot, int arrays increased up to 1185152 bytes

After running Merge Sort

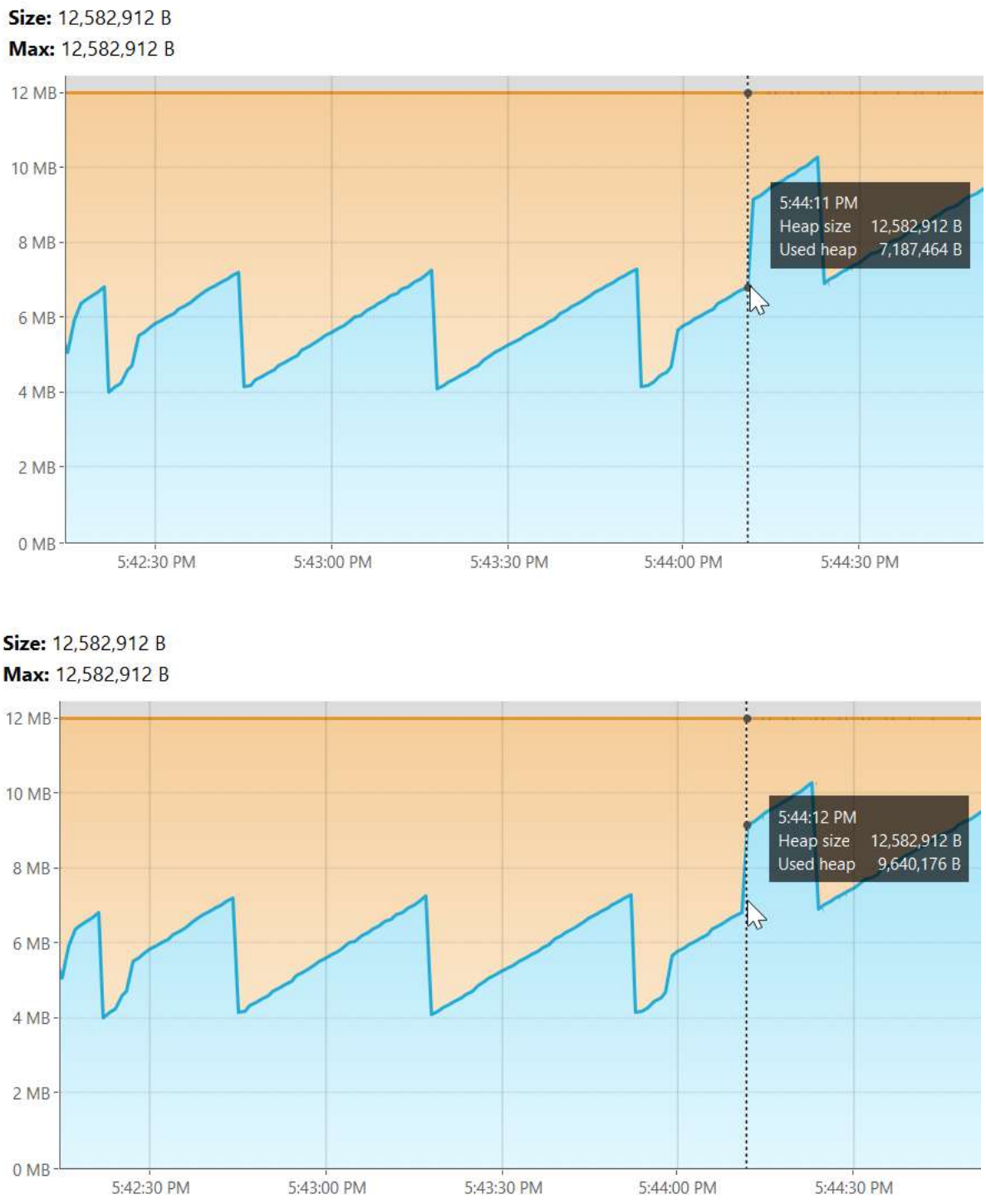


Figure 6 – Merge sort had been started. In line with VisualVM, used heap increased for 2.5 MBt

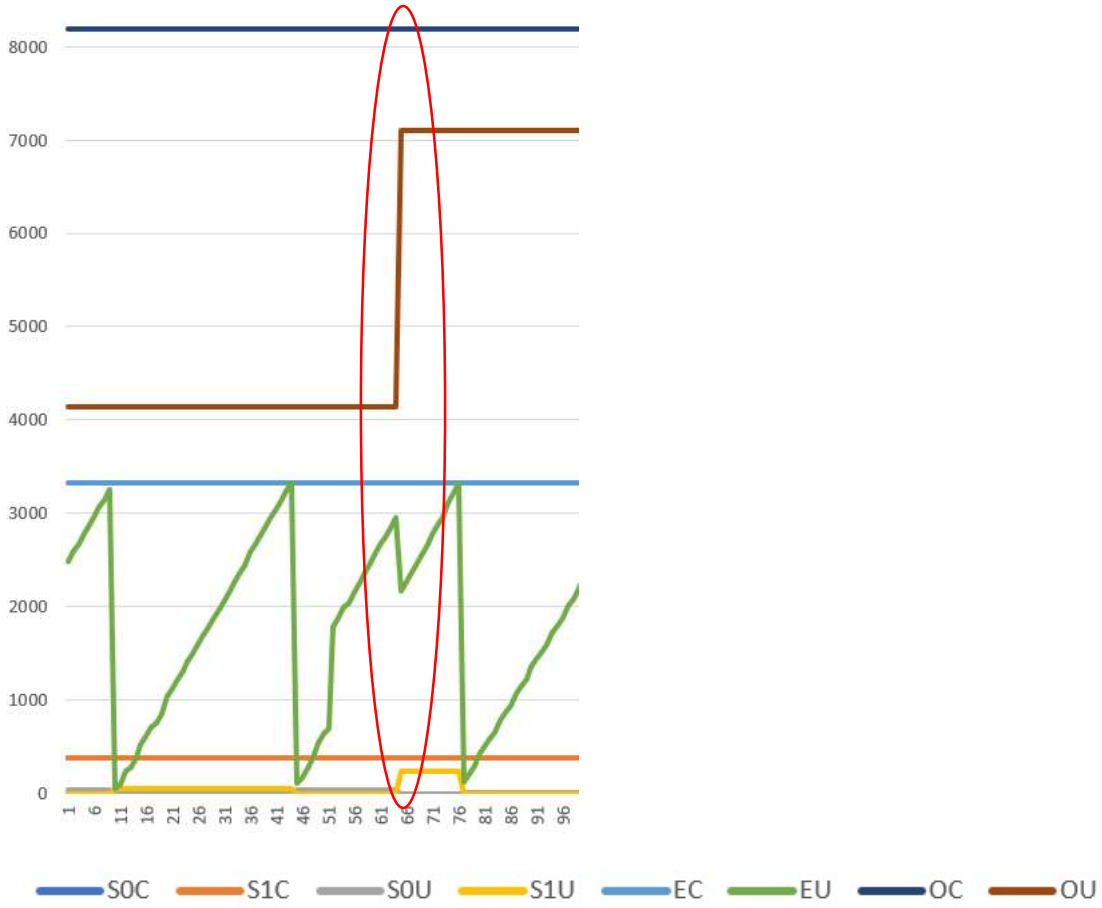


Figure 7 – Merge sort had been run. In line with jstat, garbage collector was running: Eden had been free for 783 KBt; some data was moved into survivor 1 from survivor 0 and Eden; The most of the data (about 3 MBt) was moved into Old Generation. The heap in total increased for 2.3 MBt

Name	Live Bytes	Live Objects
int[]	5,428,600 B (54.6%)	3,649 (3.3%)
byte[]	1,112,512 B (11.2%)	20,820 (18.9%)
java.lang.**String**	465,048 B (4.7%)	19,377 (17.6%)
java.lang.**Class**	353,024 B (3.5%)	2,911 (2.6%)
java.lang.**Object[]**	255,848 B (2.6%)	4,988 (4.5%)
java.util.**HashMap$Node**	191,392 B (1.9%)	5,981 (5.4%)
java.lang.reflect.**Method**	172,040 B (1.7%)	1,955 (1.8%)
java.util.concurrent.**ConcurrentHashMap$Node**	146,240 B (1.5%)	4,570 (4.1%)
java.util.**HashMap$Node[]**	131,560 B (1.3%)	1,133 (1%)
char[]	106,728 B (1.1%)	345 (0.3%)
java.util.**LinkedHashMap$Entry**	94,280 B (0.9%)	2,357 (2.1%)
java.lang.**Class[**	62,840 B (0.6%)	2,659 (2.4%)
java.lang.**String[]**	58,280 B (0.6%)	1,628 (1.5%)
java.lang.invoke.**MemberName**	56,976 B (0.6%)	1,187 (1.1%)
java.util.**ImmutableCollections$List12**	55,320 B (0.6%)	2,305 (2.1%)
java.util.concurrent.**ConcurrentHashMap$Node[]**	48,160 B (0.5%)	126 (0.1%)
java.util.**TreeMap$Entry**	47,520 B (0.5%)	1,188 (1.1%)
java.lang.reflect.**Field**	43,200 B (0.4%)	600 (0.5%)
java.lang.invoke.**LambdaForm$Name**	37,024 B (0.4%)	1,157 (1.1%)
java.util.**LinkedHashMap**	36,792 B (0.4%)	657 (0.6%)

Figure 8 – Merge sort had been started. Int[] array had been increased in 5 times. Looks reasonable. Every time when we call merge() method, new int[] array is created

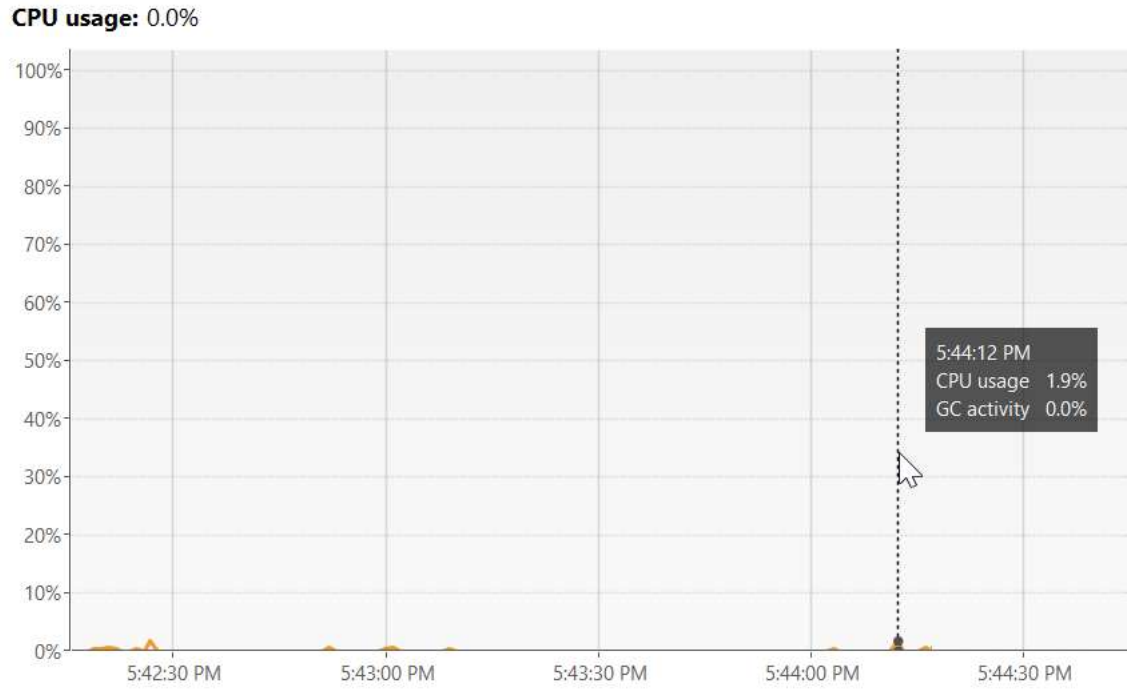


Figure 9 – Merge sort had been started. CPU wasn't loaded too much – only for 1.9%

Bubble Sort

Description

200000 int values had been added to sort using bubble algorithm.
200000 * 4 bytes = 800000 bytes = 800 KBt

After adding these values:

- In line with VisualVM used heap increased from 8 564 456 bytes to 9 464 424 bytes. The difference equals to 899 968 which is about 800 000 bytes
- In line with jstat, used Eden increased from 1344.9 to 2226.9. Difference equals to 882 kBt, which is about 0.8 MBt. Old Generation usage left the same.
- In line with VisualVM Sampler, int[] array increased about for 0.8 MBt

Before running Merge Sort:

- Eden become filled with data, the garbage collector free Eden and moved some data to Old Generation

After running Merge Sort:

- In line with VisualVM, used heap increased for 0.8 MBt because of data.getClone() method in BubbleSort.class/sort()
- In line with jstat, 0.87 MBt had been added into Eden MBt because of data.getClone() method in BubbleSort.class/sort()
- There was big garbage clean up during bubble sorting. 11 MBt from 12 was occupied with files
- In line with jstat. After some time from bubble sort startup and when Eden got filled with data, big cleanup happened. The heapfreeratio got below 10% (8.7%). Garbage collector decided to remove data from both Young and Old Generation. The application had been paused for 15 ms. 2.3 MBt was removed from Old generation. Total heap used space had been reduced from 11.2 MBt down to 5.6 MBt
- In line with the log file. The garbage collector # 22 did the job. The application was stopped for 13.6 ms. Used heap had been reduced from 10 down to 4 MBt
- Bubble Sort start - 17:45:37.816269900, end - 17:46:49.387821800

After adding these values

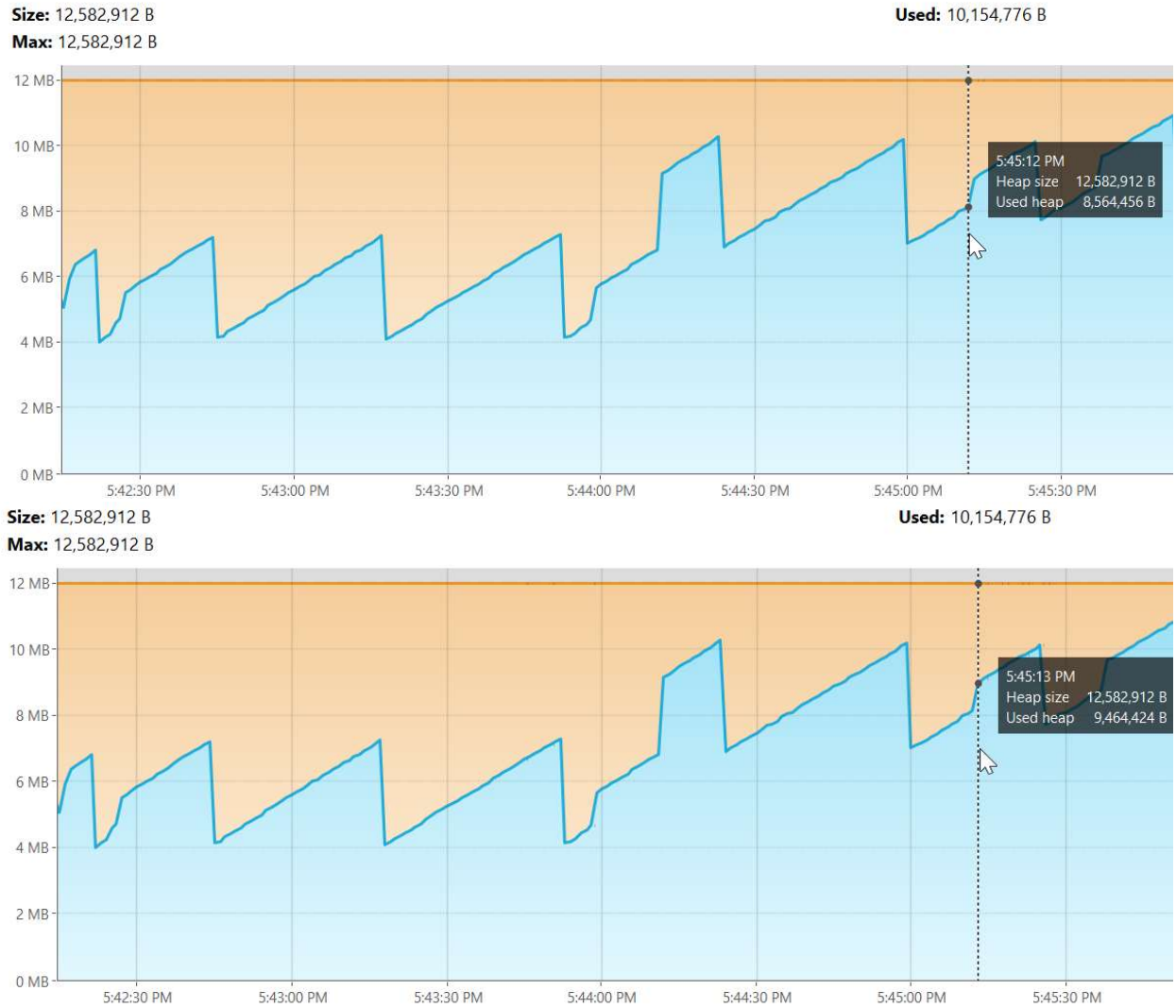


Figure 10 - 200000 values had been added. In line with VisualVM used heap increased from 8 564 456 bytes to 9 464 424 bytes. The difference equals to 899 968 which is about 800 000 bytes

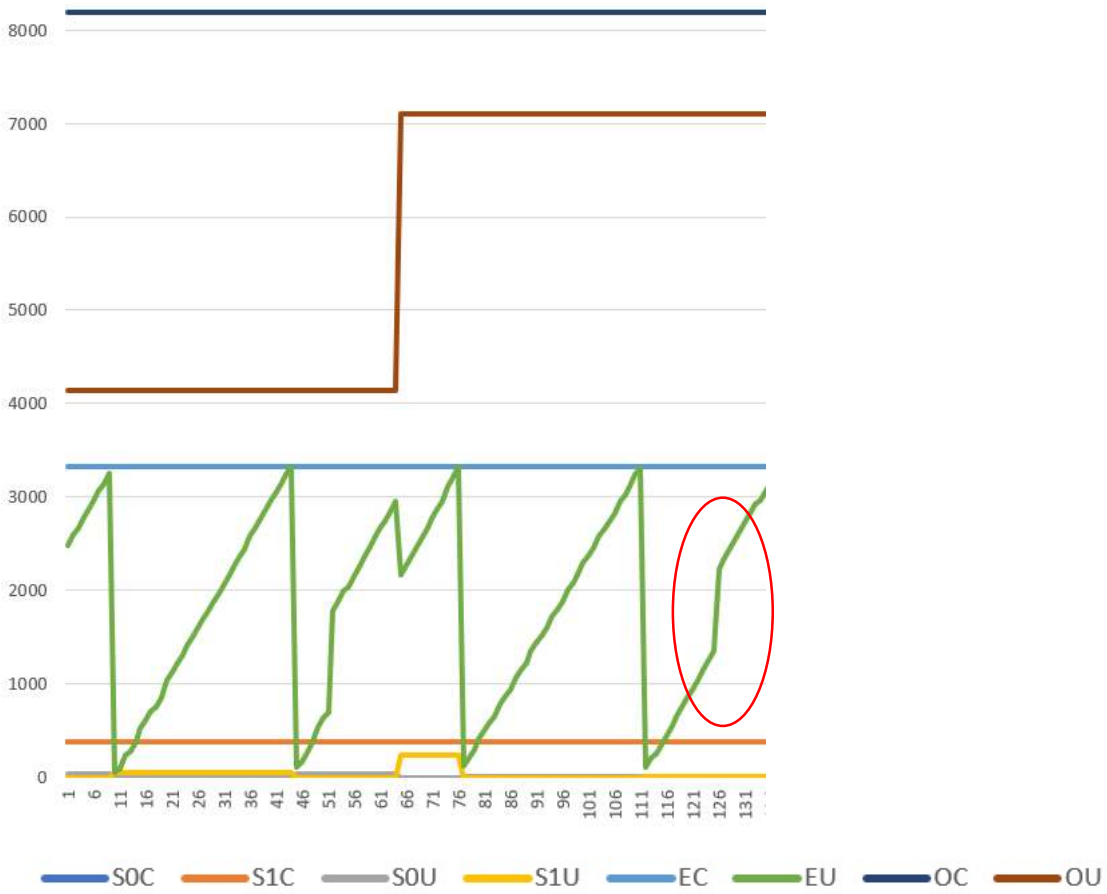
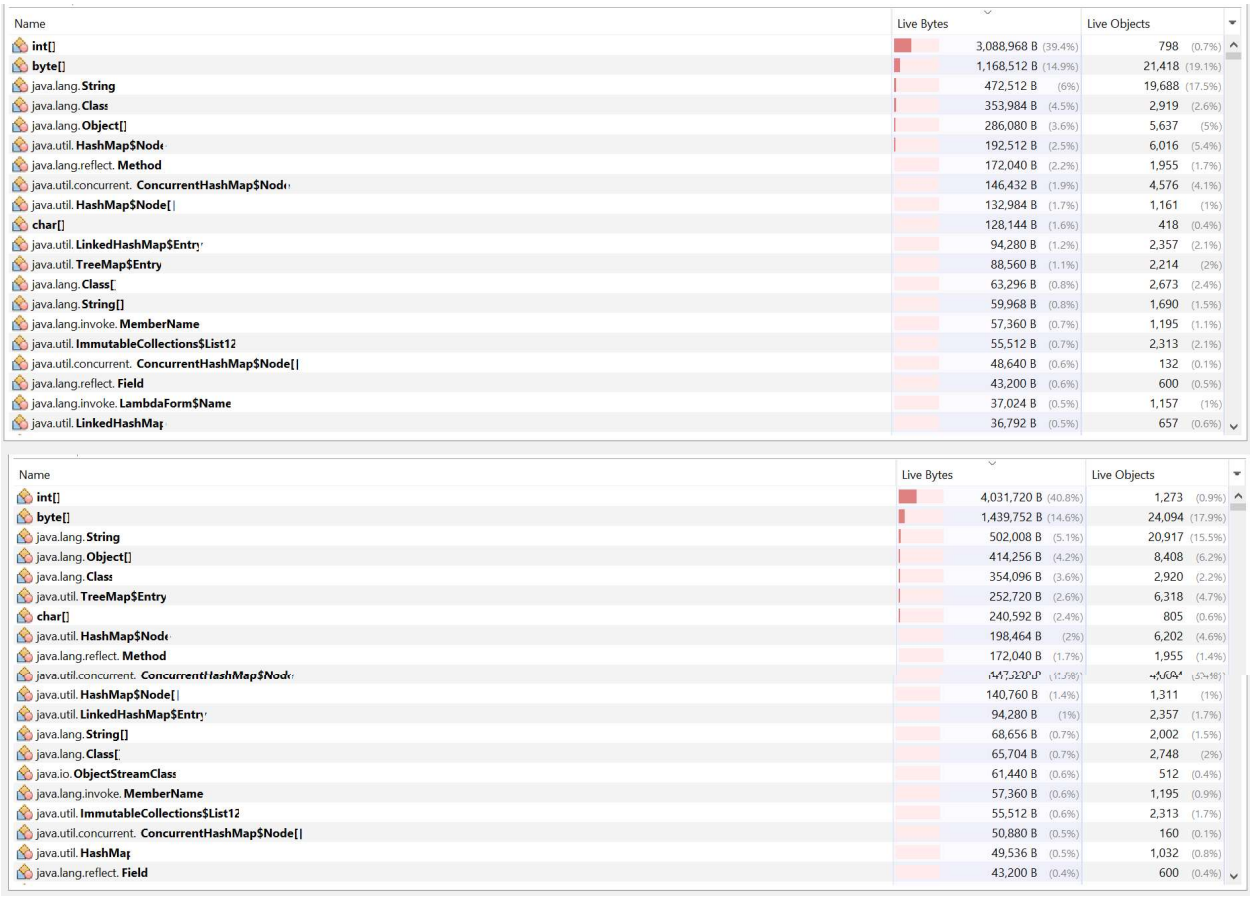


Figure 11 - 200000 values had been added. In line with jstat, used Eden increased from 1344.9 to 2226.9. Difference equals to 882 kBt, which is about 0.8 MBt. Old Generation usage left the same.



Name	Live Bytes	Live Objects
int[]	3,088,968 B (39.4%)	798 (0.7%)
byte[]	1,168,512 B (14.9%)	21,418 (19.1%)
java.lang.String	472,512 B (6%)	19,688 (17.5%)
java.lang.Class	353,984 B (4.5%)	2,919 (2.6%)
java.lang.Object[]	286,080 B (3.6%)	5,637 (5%)
java.util.HashMap$Node	192,512 B (2.5%)	6,016 (5.4%)
java.lang.reflect.Method	172,040 B (2.2%)	1,955 (1.7%)
java.util.concurrent.ConcurrentHashMap$Node	146,432 B (1.9%)	4,576 (4.1%)
java.util.HashMap$Node[]	132,984 B (1.7%)	1,161 (1%)
char[]	128,144 B (1.6%)	418 (0.4%)
java.util.LinkedHashMap$Entry	94,280 B (1.2%)	2,357 (2.1%)
java.util.TreeMap$Entry	88,560 B (1.1%)	2,214 (2%)
java.lang.Class[]	63,296 B (0.8%)	2,673 (2.4%)
java.lang.String[]	59,968 B (0.8%)	1,690 (1.5%)
java.lang.invoke.MemberName	57,360 B (0.7%)	1,195 (1.1%)
java.util.ImmutableCollections$List12	55,512 B (0.7%)	2,313 (2.1%)
java.util.concurrent.ConcurrentHashMap$Node[]	48,640 B (0.6%)	132 (0.1%)
java.lang.reflect.Field	43,200 B (0.6%)	600 (0.5%)
java.lang.invoke.LambdaForm$Name	37,024 B (0.5%)	1,157 (1%)
java.util.LinkedHashMap	36,792 B (0.5%)	657 (0.6%)

Name	Live Bytes	Live Objects
int[]	4,031,720 B (40.8%)	1,273 (0.9%)
byte[]	1,439,752 B (14.6%)	24,094 (17.9%)
java.lang.String	502,008 B (5.1%)	20,917 (15.5%)
java.lang.Object[]	414,256 B (4.2%)	8,408 (6.2%)
java.lang.Class	354,096 B (3.6%)	2,920 (2.2%)
java.util.TreeMap$Entry	252,720 B (2.6%)	6,318 (4.7%)
char[]	240,592 B (2.4%)	805 (0.6%)
java.util.HashMap$Node	198,464 B (2%)	6,202 (4.6%)
java.lang.reflect.Method	172,040 B (1.7%)	1,955 (1.4%)
java.util.concurrent.ConcurrentHashMap$Node	[illegible]	[illegible]
java.util.HashMap$Node[]	140,760 B (1.4%)	1,311 (1%)
java.util.LinkedHashMap$Entry	94,280 B (1%)	2,357 (1.7%)
java.lang.String[]	68,656 B (0.7%)	2,002 (1.5%)
java.lang.Class[]	65,704 B (0.7%)	2,748 (2%)
java.io.ObjectStreamClass	61,440 B (0.6%)	512 (0.4%)
java.lang.invoke.MemberName	57,360 B (0.6%)	1,195 (0.9%)
java.util.ImmutableCollections$List12	55,512 B (0.6%)	2,313 (1.7%)
java.util.concurrent.ConcurrentHashMap$Node[]	50,880 B (0.5%)	160 (0.1%)
java.util.HashMap	49,536 B (0.5%)	1,032 (0.8%)
java.lang.reflect.Field	43,200 B (0.4%)	600 (0.4%)

Figure 12 - 200000 values had been added. In line with VisualVM Sampler, int[] array increased about for 0.8 MBt

Before running Merge Sort

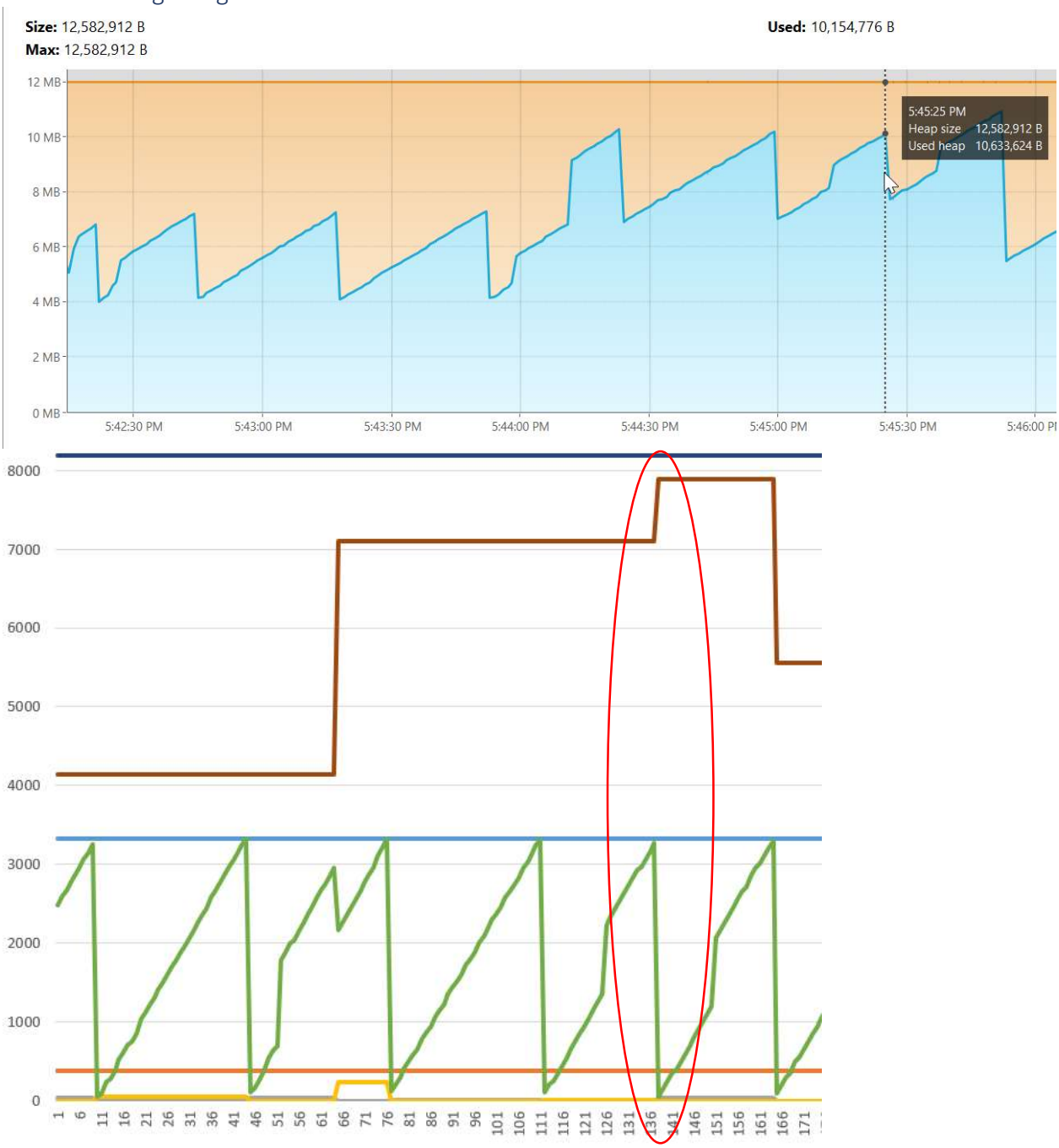


Figure 13 – When Eden become filled with data, the garbage collector free Eden and moved some data to Old Generation and to another survivor

After running Merge Sort

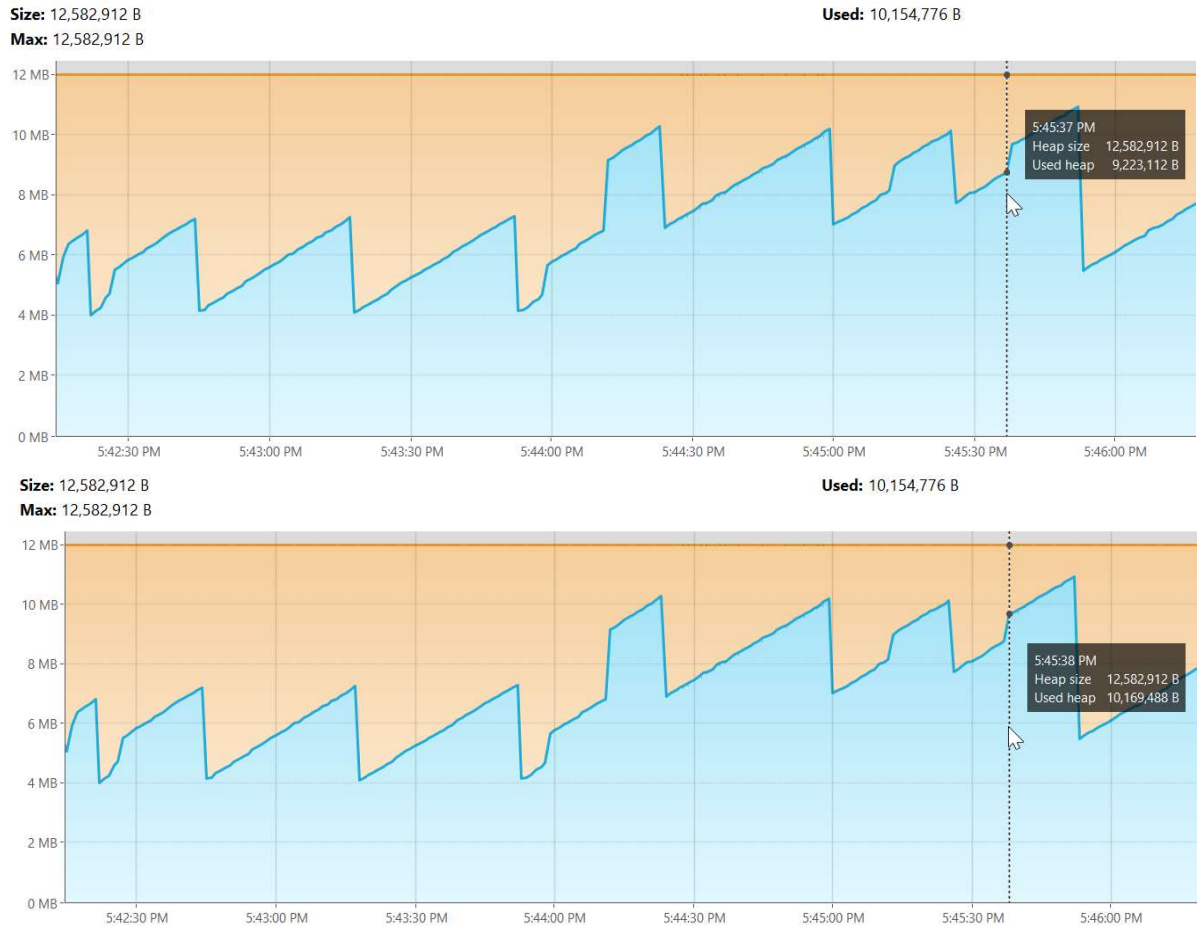


Figure 14 - Bubble sort had been started. In line with VisualVM, used heap increased for 0.8 MBt because of data.getClone() method in BubbleSort.class/sort()

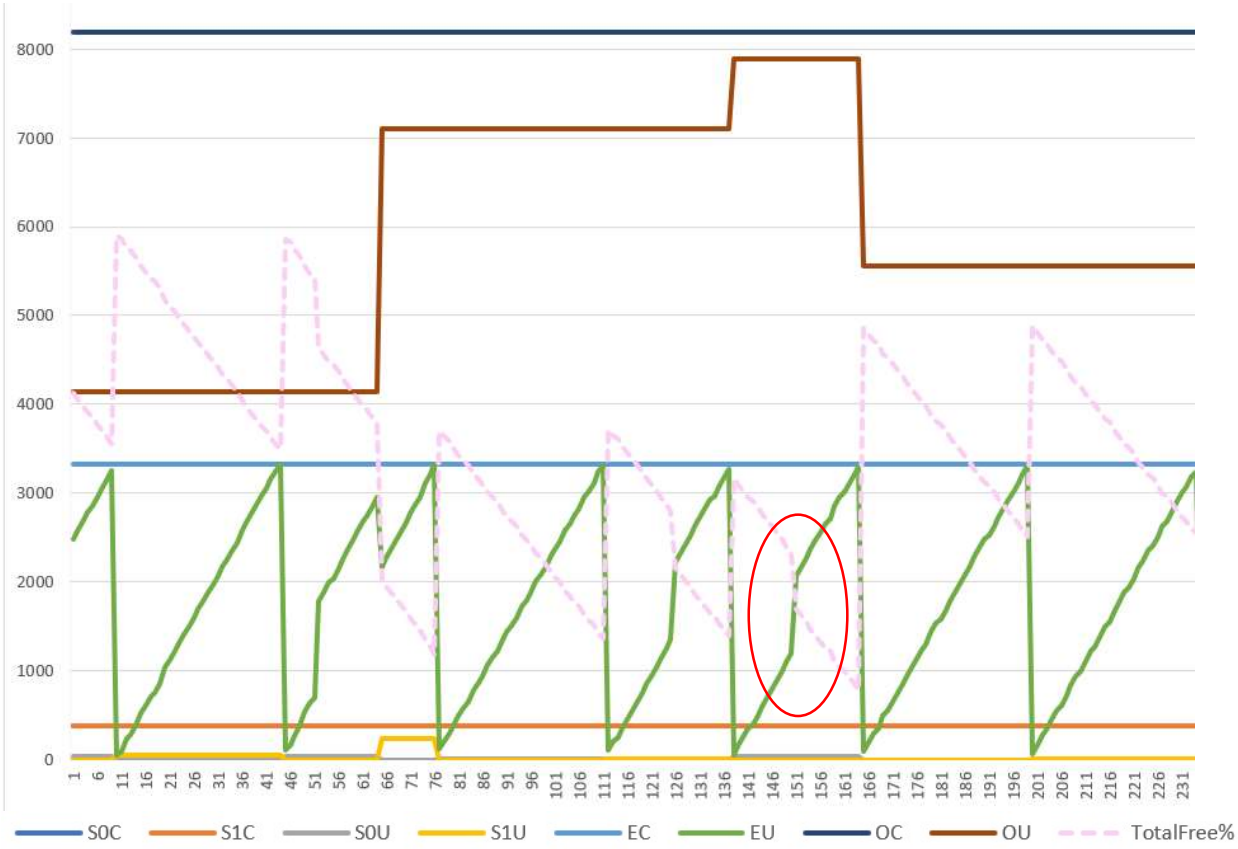


Figure 15 - Bubble sort had been run. In line with jstat, 0.87 MBt had been added into Eden MBt because of data.getClone() method in BubbleSort.class/sort()

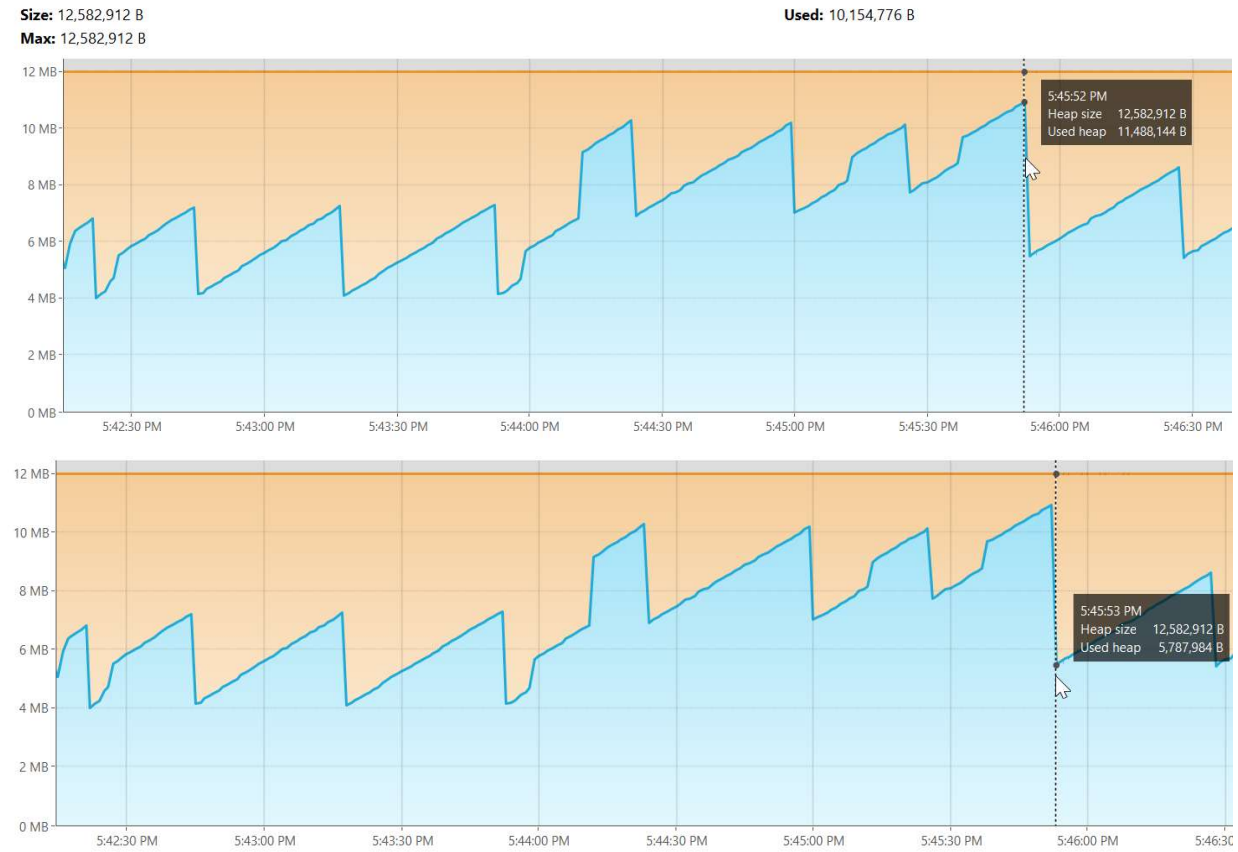


Figure 16 – In line with VisualVM. There was big garbage clean up during bubble sorting. 11 MBt from 12 was occupied with files

Figure 17 – In line with jstat. After some time from bubble sort startup and when Eden got filled with data, big cleanup happened. The heapfreeratio got below 10% (8.7%). Garbage collector decided to remove data from both Young and Old Generation. The application had been paused for 15 ms. 2.3 MBt was removed from Old generation. Total heap used space had been reduced from 11.2 MBt down to 5.6 MBt

```
[177.314s][info][gc] GC(15) Pause Young (Allocation Failure) 9M->6M(11M) 0.610ms
[177.319s][info][gc] GC(16) Pause Young (Allocation Failure) 9M->6M(11M) 0.516ms
[177.323s][info][gc] GC(17) Pause Young (Allocation Failure) 9M->6M(11M) 0.272ms
[177.327s][info][gc] GC(18) Pause Young (Allocation Failure) 10M->6M(11M) 0.283ms
[177.331s][info][gc] GC(19) Pause Young (Allocation Failure) 10M->7M(11M) 0.458ms
[177.335s][info][gc] GC(20) Pause Young (Allocation Failure) 9M->7M(11M) 0.681ms
[201.455s][info][gc] GC(21) Pause Young (Allocation Failure) 10M->10M(11M) 0.039ms
[201.469s][info][gc] GC(22) Pause Full (Allocation Failure) 10M->4M(11M) 13.586ms
[227.483s][info][gc] GC(23) Pause Young (Allocation Failure) 8M->5M(11M) 0.762ms
[262.518s][info][gc] GC(24) Pause Young (Allocation Failure) 8M->5M(11M) 0.341ms
[289.539s][info][gc] GC(25) Pause Young (Allocation Failure) 8M->6M(11M) 0.388ms
```

Figure 18 – In line with the log file. The garbage collector # 22 did the job. The application was stopped for 13.6 ms. Used heap had been reduced from 10 down to 4 MBt

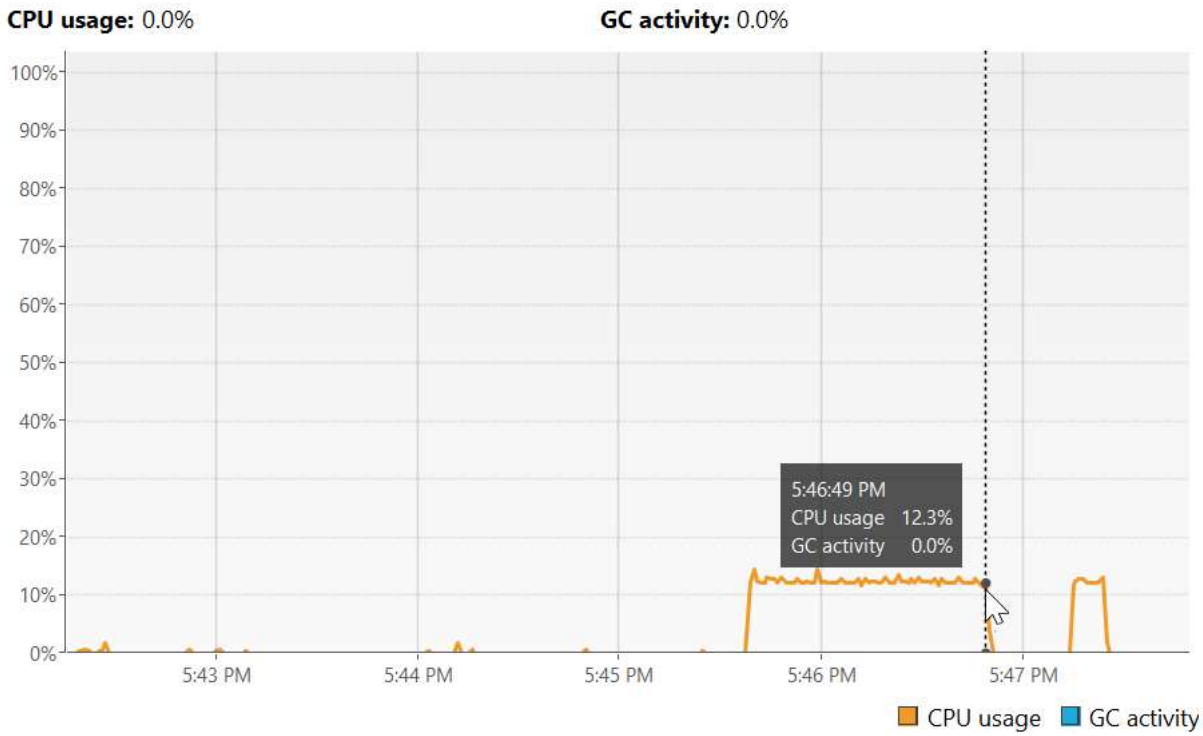


Figure 19 – Bubble sort was running for 1 minute. The CPU average usage was 12.3%

Insert Sort

Description

200000 int values had been added to sort using insert algorithm.
200000 * 4 bytes = 800000 bytes = 800 KBt

After adding these values:

- In line with VisualVM used heap increased from 6 015 760 bytes to 6 959 232 bytes. The difference equals to 944 KBt
- In line with jstat, used Eden increased from 413 to 1273.6. Difference equals to 860 kBt, which is about 0.8 MBt. Old Generation usage left the same.

After running InsertSort:

- In line with VisualVM, used heap increased for 0.8 MBt because of the same method - data.getClone() in BubbleSort.class/sort()
- In line with jstat, 0.84 MBt had been added into Eden MBt because of the same method - data.getClone() in BubbleSort.class/sort()
- Last Garbage Collector #25 moved 1.5 MBt data from Young generation into Old and empty young generation.
- In line with VisualVM, average CPU usage was about 13.2%. The insert sort worked only for 10 seconds
- InsertSort Sort start - 17:47:14.132656400, end - 17:47:24.276586200

After adding these values

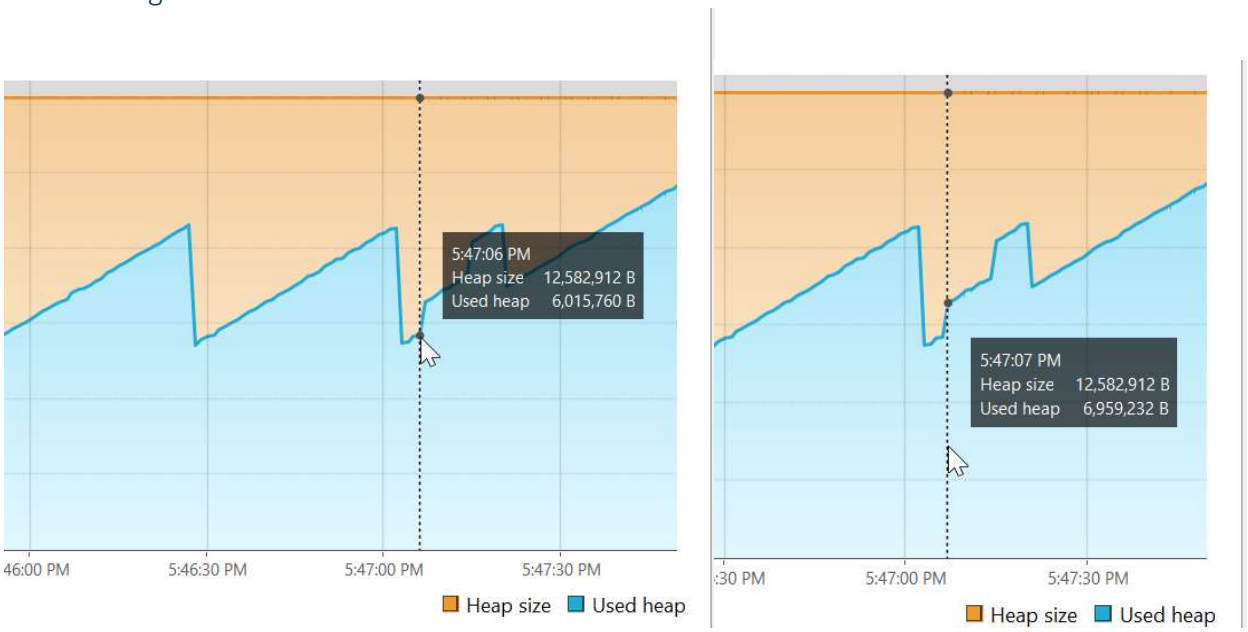


Figure 20 - 200000 values had been added. In line with VisualVM used heap increased from 6 015 760 bytes to 6 959 232 bytes. The difference equals to 944 KBt

Figure 21 - 200000 values had been added. In line with jstat, used Eden increased from 413 to 1273.6. Difference equals to 860 kBt, which is about 0.8 MBt. Old Generation usage left the same.

After running InsertSort

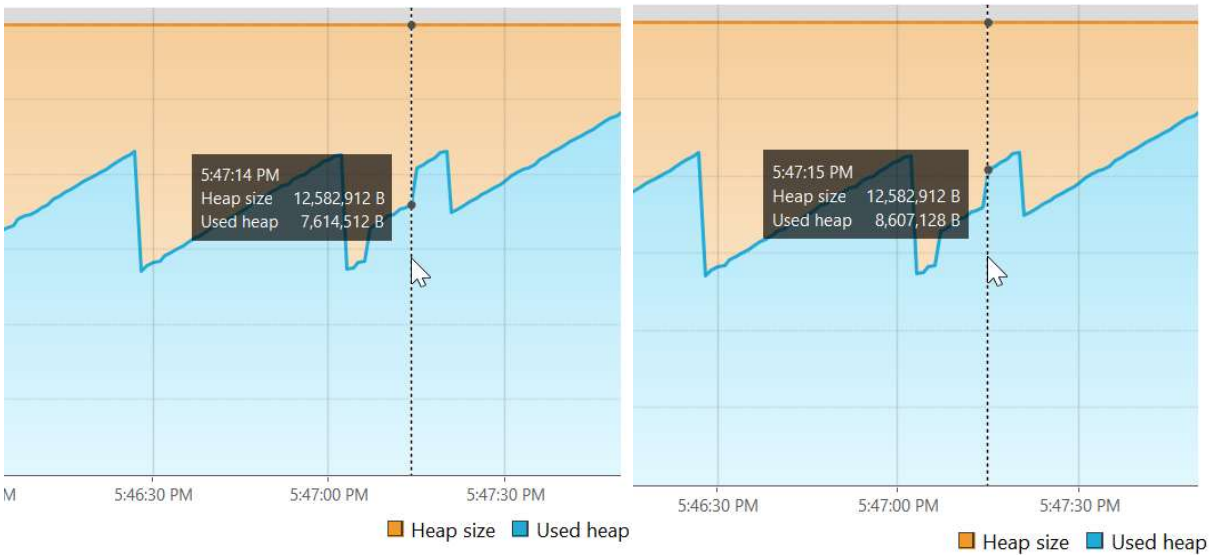


Figure 22 - Insert sort had been started. In line with VisualVM, used heap increased for 0.8 MBt because of the same method - data.getClone() in BubbleSort.class/sort()

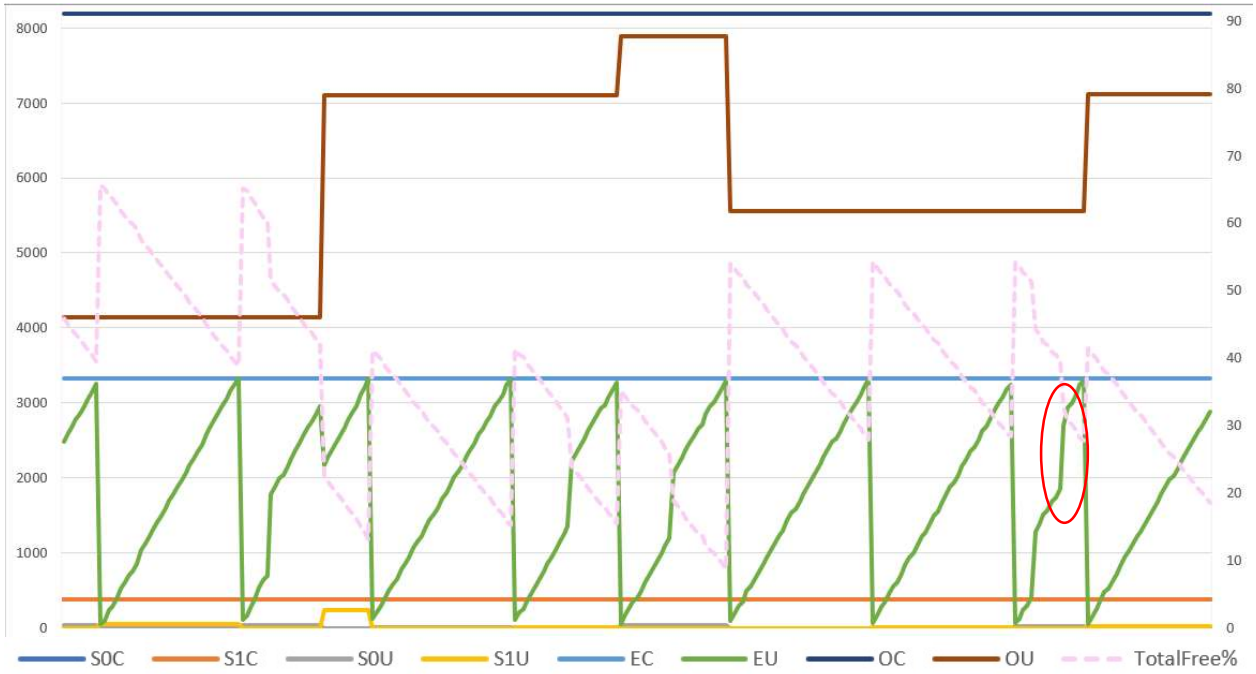


Figure 23 - Insert sort had been run. In line with jstat, 0.84 MBt had been added into Eden MBt because of the same method - data.getClone() in BubbleSort.class/sort()

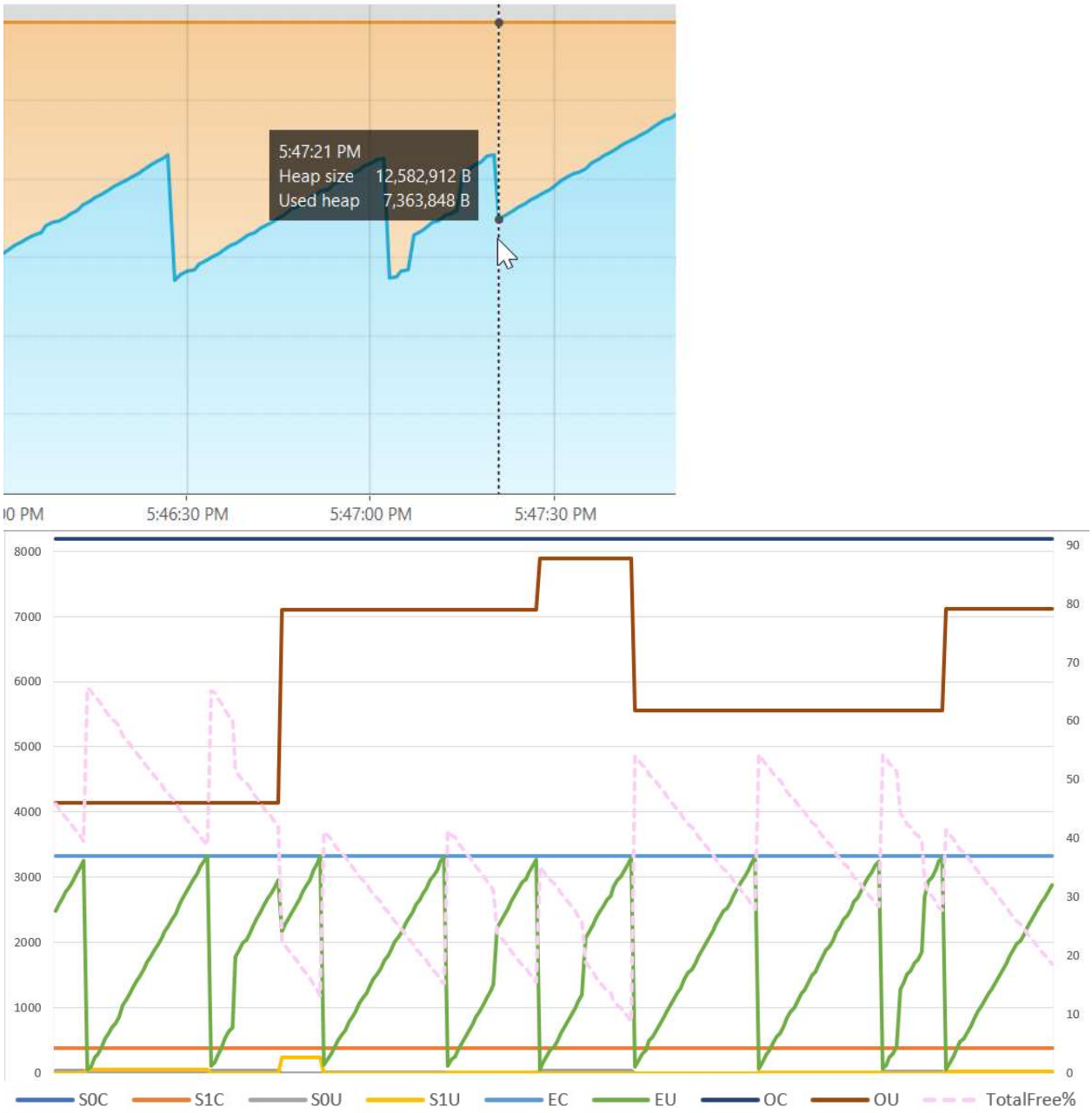


Figure 24 – Last Garbage Collector #25 moved 1.5 MBt data from Young generation into Old and empty young generation.

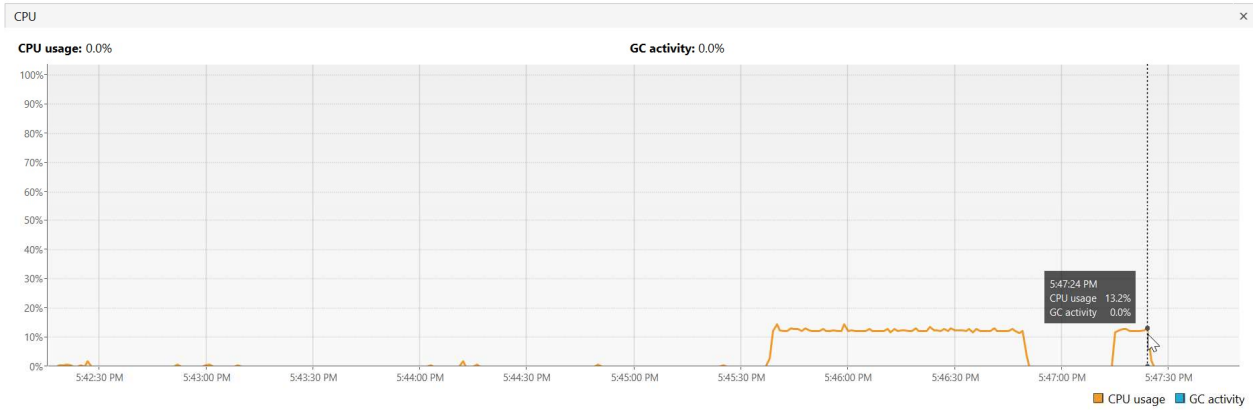


Figure 25 – In line with VisualVM, average CPU usage was about 13.2%. The insert sort worked only for 10 seconds

Parallel

Description

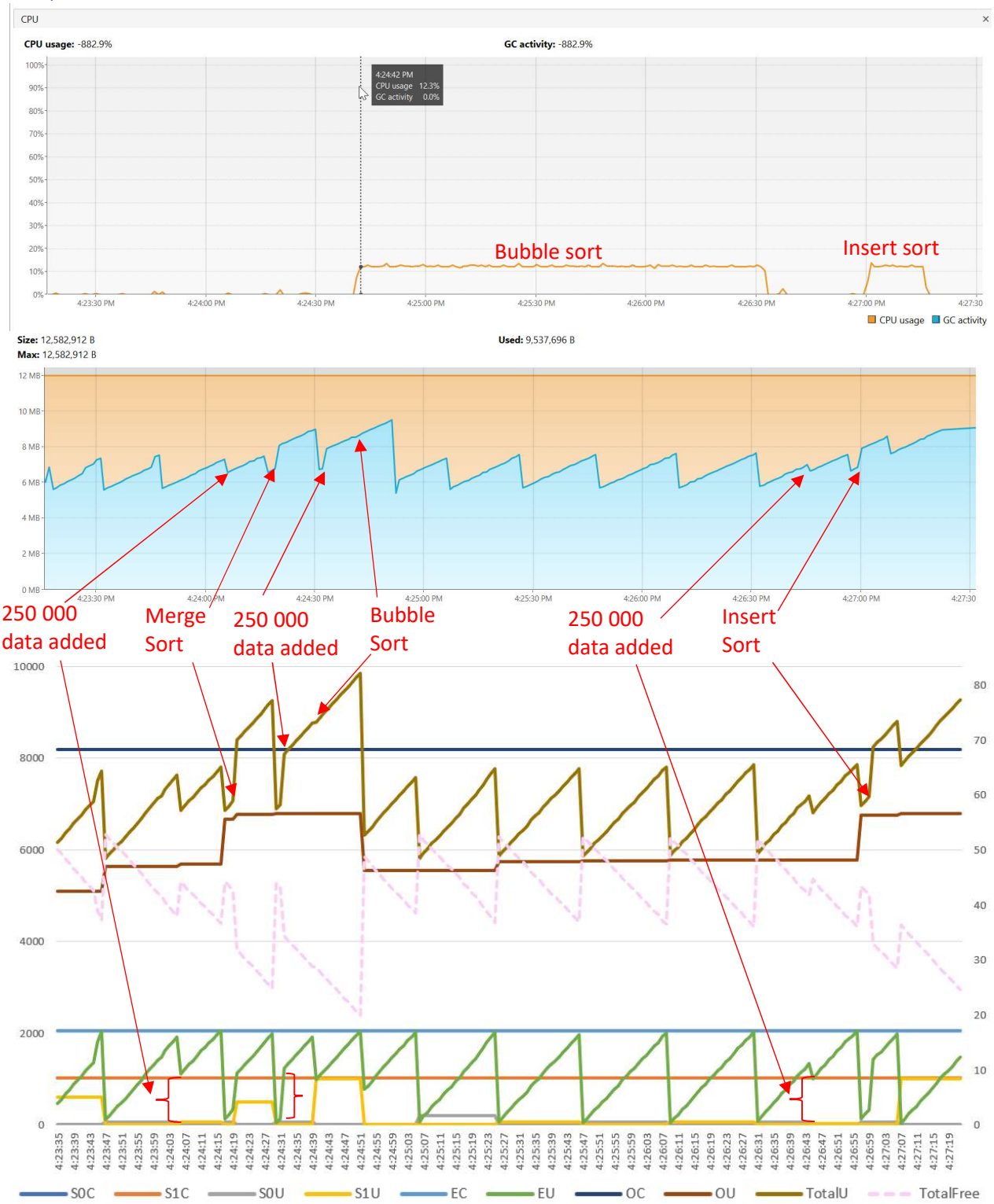
16:24:05 – Array with 250 000 integers had been added. Eden became overfilled. Garbage collector deleted all data from Eden “Pause Young (Allocation Failure)” and put 1MBt array
16:24:16 – Data had been moved from Young generation to Old
16:24:19 – Merge Sort. Every time when we call merge() method, new int[] array is created. New arrays are required new space. Data was put into Survivor, Eden and into Old Generation
16:24:32 - Array with 250 000 integers had been added. Eden became overfilled. Garbage collector put 1MBt array into Eden
16:24:40 – Bubble Sort creates array copy. Arry copy had been put into survivor
16:24:52 – Total used heap became equal 9.8 MBt from 12 MBt which is 81% usage. Garbage collector launched Full Pause and deleted 3.5 MBt data from the heap
16:26:45 - Array with 250 000 integers had been added. Eden became overfilled. Garbage collector deleted all data from Eden “Pause Young (Allocation Failure)” and put 1MBt array
16:26:27 – Garbage collector could store all data in Eden. Some data had been moved into Old Generation
16:26:59 - Insert Sort creates array copy. Arry copy had been put into Eden
16:27:05 - Garbage collector could store all data in Eden. Some data had been moved into Old Generation

Commands

```
2024-04-07T16:23:10.715945900
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T16:24:05.950058700
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

2
2
MergeSort Sort start - 16:24:19.730244300
2024-04-07T16:24:19.731239600
MergeSort Sort end - 16:24:19.814050600
2024-04-07T16:24:19.814050600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T16:24:32.483466700
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

3
3
BubbleSort Sort start - 16:24:40.598345300
2024-04-07T16:24:40.598345300
BubbleSort Sort end - 16:26:32.134579600
2024-04-07T16:26:32.135577200
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
```

```
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T16:26:45.499088600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

4
4
InsertSort Sort start - 16:26:59.884351200
2024-04-07T16:26:59.884351200
InsertSort Sort end - 16:27:15.533909900
2024-04-07T16:27:15.533909900
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

6
6
The End
2024-04-07T16:27:23.174232400
```

Graphs


CPU
CPU usage: -882.9%
GC activity: -882.9%
4:24:42 PM
CPU usage 12.3%
GC activity 0.0%
Bubble sort
Insert sort
CPU usage
GC activity
Size: 12,582,912 B
Max: 12,582,912 B
Used: 9,537,696 B
250 000 data added
Merge Sort
250 000 data added
Bubble Sort
250 000 data added
Insert Sort
S0C
S1C
S0U
S1U
EC
EU
OC
OU
TotalU
TotalFree

Logs

[0.027s][info][gc] Using Parallel
[0.893s][info][gc] GC(0) Pause Young (Allocation Failure) 3M->1M(11M) 4.962ms
[4.088s][info][gc] GC(1) Pause Young (Allocation Failure) 4M->2M(11M) 2.475ms
[4.176s][info][gc] GC(2) Pause Young (Allocation Failure) 5M->2M(11M) 1.788ms
[4.224s][info][gc] GC(3) Pause Young (Allocation Failure) 5M->3M(11M) 2.346ms
[4.253s][info][gc] GC(4) Pause Young (Allocation Failure) 6M->4M(11M) 1.571ms
[5.314s][info][gc] GC(5) Pause Young (Allocation Failure) 7M->4M(10M) 3.047ms
[5.518s][info][gc] GC(6) Pause Young (Allocation Failure) 6M->5M(11M) 2.179ms
[6.776s][info][gc] GC(7) Pause Young (Allocation Failure) 7M->5M(11M) 1.523ms
[20.650s][info][gc] GC(8) Pause Young (Allocation Failure) 7M->5M(11M) 1.251ms
[36.662s][info][gc] GC(9) Pause Young (Allocation Failure) 7M->5M(11M) 1.118ms
[55.374s][info][gc] GC(10) Pause Young (Allocation Failure) 7M->5M(11M) 0.416ms
[66.664s][info][gc] GC(11) Pause Young (Allocation Failure) 7M->6M(11M) 0.852ms
[69.165s][info][gc] GC(12) Pause Young (Allocation Failure) 8M->7M(11M) 0.472ms
[69.184s][info][gc] GC(13) Pause Full (Ergonomics) 7M->5M(11M) 19.307ms
[69.199s][info][gc] GC(14) Pause Young (Allocation Failure) 7M->5M(11M) 0.406ms
[69.210s][info][gc] GC(15) Pause Young (Allocation Failure) 7M->5M(11M) 0.248ms
[69.213s][info][gc] GC(16) Pause Young (Allocation Failure) 7M->5M(11M) 0.262ms
[69.216s][info][gc] GC(17) Pause Young (Allocation Failure) 7M->5M(11M) 0.226ms
[69.219s][info][gc] GC(18) Pause Young (Allocation Failure) 7M->5M(11M) 0.218ms
[69.223s][info][gc] GC(19) Pause Young (Allocation Failure) 7M->5M(11M) 0.217ms
[69.226s][info][gc] GC(20) Pause Young (Allocation Failure) 7M->6M(11M) 0.287ms
[69.229s][info][gc] GC(21) Pause Young (Allocation Failure) 8M->6M(11M) 0.302ms
[69.231s][info][gc] GC(22) Pause Young (Allocation Failure) 8M->6M(11M) 0.208ms
[69.234s][info][gc] GC(23) Pause Young (Allocation Failure) 8M->6M(11M) 0.256ms
[69.237s][info][gc] GC(24) Pause Young (Allocation Failure) 8M->6M(11M) 0.311ms
[69.240s][info][gc] GC(25) Pause Young (Allocation Failure) 8M->6M(11M) 0.315ms
[69.243s][info][gc] GC(26) Pause Young (Allocation Failure) 8M->7M(11M) 0.293ms
[79.695s][info][gc] GC(27) Pause Young (Allocation Failure) 9M->6M(11M) 0.383ms
[90.030s][info][gc] GC(28) Pause Young (Allocation Failure) 8M->7M(11M) 0.346ms
[101.684s][info][gc] GC(29) Pause Young (Allocation Failure) 9M->8M(11M) 1.002ms
[101.696s][info][gc] GC(30) Pause Full (Ergonomics) 8M->5M(11M) 11.684ms
[115.717s][info][gc] GC(31) Pause Young (Allocation Failure) 7M->5M(11M) 0.603ms
[135.729s][info][gc] GC(32) Pause Young (Allocation Failure) 7M->5M(11M) 0.576ms
[156.740s][info][gc] GC(33) Pause Young (Allocation Failure) 7M->5M(11M) 0.347ms
[178.749s][info][gc] GC(34) Pause Young (Allocation Failure) 7M->5M(11M) 0.710ms
[200.756s][info][gc] GC(35) Pause Young (Allocation Failure) 7M->5M(11M) 0.327ms
[214.927s][info][gc] GC(36) Pause Young (Allocation Failure) 7M->5M(11M) 0.259ms
[226.741s][info][gc] GC(37) Pause Young (Allocation Failure) 7M->6M(11M) 0.377ms
[236.784s][info][gc] GC(38) Pause Young (Allocation Failure) 8M->7M(11M) 0.463ms

CMS

Description

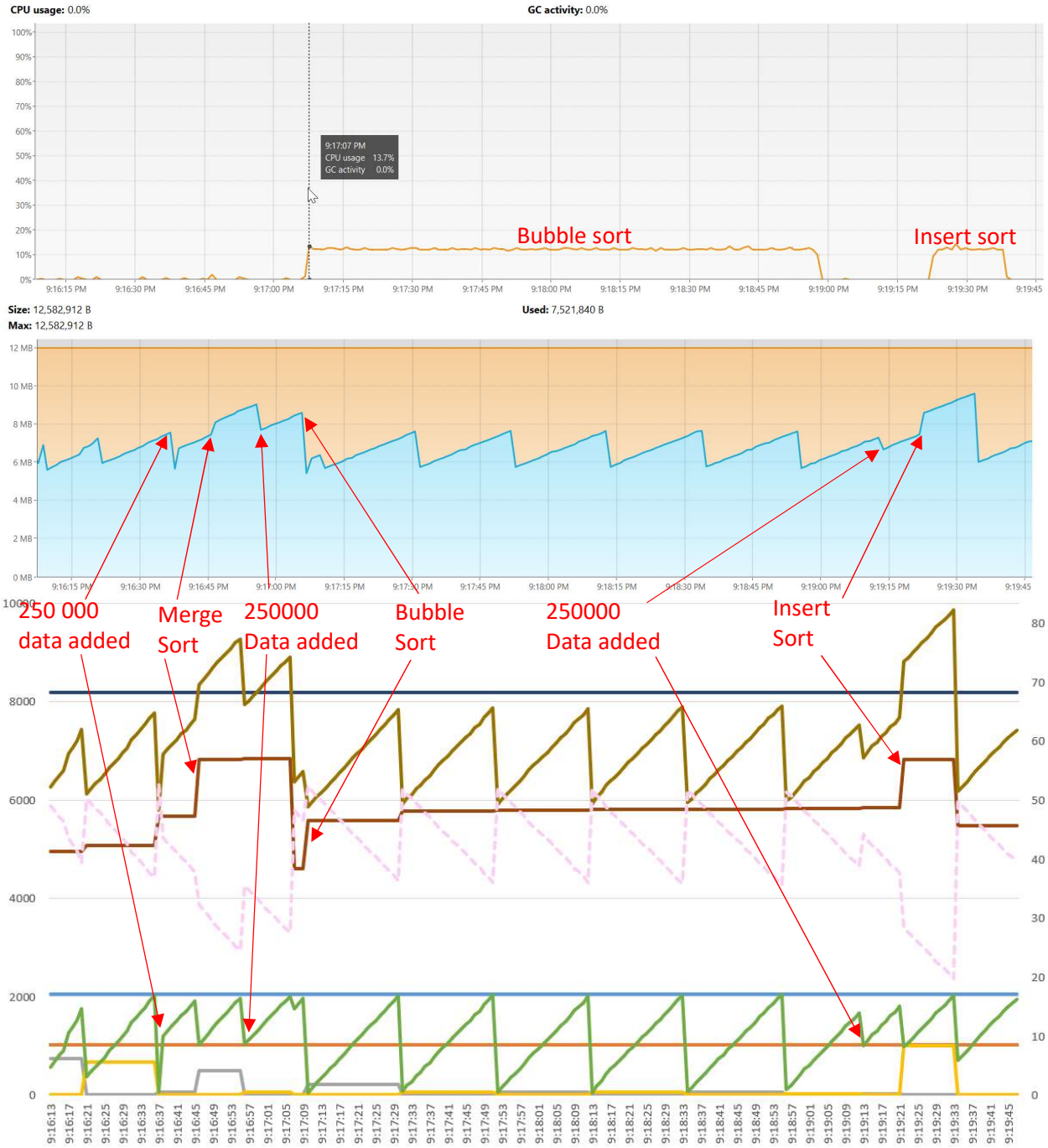
21:16:36 – Pause Young (Allocation Failure). Eden space clean up. Data had been moved from survivor to old generation.
21:16:38 – Array with 250 000 integers had been added. Integer array had been put into Eden space
21:16:46 – Merge Sort. Eden clean up. 1MBt data had put into Eden and 1.1 MBt into Old Generation. 0.5 MBt data had been put into Survivor
21:16:46 - Pause Full (Ergonomics) 7M->4M. Garbage collector clean some data
21:16:55 – Array with 250 000 integers had been added. 1MBt data had been put into Eden
21:17:06 - Pause Full (Ergonomics) 8M->4M(11M). 2.2 MBt had been deleted from Old Generation
21:17:06 – Bubble Sort. 1.7 MBt had been put into Eden. The Sort had been completed at 21:18:57
21:17:09 – Pause Young (Allocation Failure). 1MBt had been moved from Eden to Old Generation
21:19:13 – Array with 250 000 integers had been added. 1MBt had been added to Eden
21:19:21 – Insert Sort. 3 MBt (1MBt each) had been added to Eden/ Survivor/Old Generation
21:19:33 - Pause Full (Ergonomics) 8M->5M(11M). 1.3 MBt had been deleted from Old Generation and 1.3 MBt from Edene. 1MBt deleted from Survivor.

Commands

```
2024-04-07T21:16:01.762541800
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T21:16:38.077639400
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

2
2
MergeSort Sort start - 21:16:46.144120100
2024-04-07T21:16:46.144120100
MergeSort Sort end - 21:16:46.232911200
2024-04-07T21:16:46.232911200
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T21:16:55.731862800
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

3
3
BubbleSort Sort start - 21:17:06.538772800
2024-04-07T21:17:06.538772800
BubbleSort Sort end - 21:18:57.528025100
2024-04-07T21:18:57.528025100
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
```

```
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-07T21:19:13.340730
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

4
4
InsertSort Sort start - 21:19:21.961001700
2024-04-07T21:19:21.961001700
InsertSort Sort end - 21:19:37.807685400
2024-04-07T21:19:37.807685400
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

6
6
The End
2024-04-07T21:19:46.816309800
```

Graphs


CPU usage: 0.0%
GC activity: 0.0%
9:17:07 PM
CPU usage 13.7%
GC activity 0.0%
Bubble sort
Insert sort
Size: 12,582,912 B
Max: 12,582,912 B
Used: 7,521,840 B
250 000
data added
Merge
Sort
250000
Data added
Bubble
Sort
250000
Data added
Insert
Sort

Logs

[0.023s][info][gc] Using Parallel
[0.764s][info][gc] GC(0) Pause Young (Allocation Failure) 3M->1M(11M) 3.848ms
[4.043s][info][gc] GC(1) Pause Young (Allocation Failure) 4M->2M(11M) 1.855ms
[4.136s][info][gc] GC(2) Pause Young (Allocation Failure) 5M->2M(11M) 1.822ms
[4.182s][info][gc] GC(3) Pause Young (Allocation Failure) 5M->3M(11M) 2.552ms
[4.214s][info][gc] GC(4) Pause Young (Allocation Failure) 6M->4M(11M) 1.440ms
[5.833s][info][gc] GC(5) Pause Young (Allocation Failure) 7M->4M(10M) 2.560ms
[5.952s][info][gc] GC(6) Pause Young (Allocation Failure) 6M->5M(11M) 2.855ms
[7.051s][info][gc] GC(7) Pause Young (Allocation Failure) 7M->5M(11M) 1.574ms
[19.093s][info][gc] GC(8) Pause Young (Allocation Failure) 7M->5M(11M) 1.327ms
[35.044s][info][gc] GC(9) Pause Young (Allocation Failure) 7M->5M(11M) 1.283ms
[44.521s][info][gc] GC(10) Pause Young (Allocation Failure) 7M->7M(11M) 1.502ms
[44.537s][info][gc] GC(11) Pause Full (Ergonomics) 7M->4M(11M) 16.346ms
[44.541s][info][gc] GC(12) Pause Young (Allocation Failure) 6M->5M(11M) 0.967ms
[44.545s][info][gc] GC(13) Pause Young (Allocation Failure) 7M->5M(11M) 0.349ms
[44.560s][info][gc] GC(14) Pause Young (Allocation Failure) 7M->5M(11M) 0.309ms
[44.574s][info][gc] GC(15) Pause Young (Allocation Failure) 7M->5M(11M) 0.263ms
[44.578s][info][gc] GC(16) Pause Young (Allocation Failure) 7M->5M(11M) 0.355ms
[44.582s][info][gc] GC(17) Pause Young (Allocation Failure) 7M->5M(11M) 0.300ms
[44.585s][info][gc] GC(18) Pause Young (Allocation Failure) 7M->5M(11M) 0.313ms
[44.588s][info][gc] GC(19) Pause Young (Allocation Failure) 7M->6M(11M) 0.314ms
[44.591s][info][gc] GC(20) Pause Young (Allocation Failure) 8M->6M(11M) 0.308ms
[44.595s][info][gc] GC(21) Pause Young (Allocation Failure) 8M->6M(11M) 0.313ms
[44.598s][info][gc] GC(22) Pause Young (Allocation Failure) 8M->6M(11M) 0.355ms
[44.601s][info][gc] GC(23) Pause Young (Allocation Failure) 8M->6M(11M) 0.322ms
[44.603s][info][gc] GC(24) Pause Young (Allocation Failure) 8M->6M(11M) 0.293ms
[44.606s][info][gc] GC(25) Pause Young (Allocation Failure) 8M->7M(11M) 0.280ms
[54.061s][info][gc] GC(26) Pause Young (Allocation Failure) 9M->6M(11M) 0.431ms
[64.915s][info][gc] GC(27) Pause Young (Allocation Failure) 8M->8M(11M) 0.524ms
[64.928s][info][gc] GC(28) Pause Full (Ergonomics) 8M->4M(11M) 12.947ms
[68.074s][info][gc] GC(29) Pause Young (Allocation Failure) 6M->5M(11M) 0.674ms
[89.098s][info][gc] GC(30) Pause Young (Allocation Failure) 7M->5M(11M) 0.526ms
[110.127s][info][gc] GC(31) Pause Young (Allocation Failure) 7M->5M(11M) 0.358ms
[131.152s][info][gc] GC(32) Pause Young (Allocation Failure) 7M->5M(11M) 0.453ms
[152.181s][info][gc] GC(33) Pause Young (Allocation Failure) 7M->5M(11M) 0.358ms
[174.087s][info][gc] GC(34) Pause Young (Allocation Failure) 7M->5M(11M) 0.364ms
[191.713s][info][gc] GC(35) Pause Young (Allocation Failure) 7M->5M(11M) 0.318ms
[200.337s][info][gc] GC(36) Pause Young (Allocation Failure) 7M->7M(11M) 0.443ms
[212.243s][info][gc] GC(37) Pause Young (Allocation Failure) 9M->8M(11M) 0.524ms
[212.261s][info][gc] GC(38) Pause Full (Ergonomics) 8M->5M(11M) 17.628ms

G1

Settings

Total Heap = 12 MBt
Reserved = Xmx – xms = 0 MBt
Eden capacity = 3072 KBt
Survivor1 capacity = 0
Survivor2 capacity = 1024 KBt
Old Generation capacity = 8192 KBt

-jstat was run in parallel with VisualVM to see data move between different parts of the heap

Merge Sort

Description

G1 collection is different from Serial/ Parallel and CMS:

- Each heap part (old/ Eden/ survivor) is defragged into small scattered pieces.
- Sometimes old generation pieces could be combined to store humongous data
- Garbage collector could change young/old ratio
- G1 works constantly in a cycle (review Figure 26 page#33):
    - **5:56:55 … 5:56:56; 5:57:38**. G1 Evacuation Pause – it moves data from Eden to Survivor or to Old generation. It implements short but often pauses
    - **5:57:01** Concurrent Mark Cycle – actually, it is the start of the cycle. Garbage collector marks the live data/objects. It finishes with two special stop the word pauses: Remark and Cleanup
    - **5:57:01** Remark – it completes marking process and calculates data to delete it from heap
    - **5:57:01** Cleanup – it determines if data removal from heap is required. If removal is required, the cleanup will be followed by Prepared Mixed pause
    - **5:57:08** Prepared Mixed – it calculates how much data (objects from both generations) should be deleted based on statistics to be in determined limits (e.g. pause time)
    - **5:57:12** Mixed – Garbage collector removes the garbage. Review Graphs and logs bellow. Mixed pause happened several times: successful - **5:57:12**; and non successful - **5:58:21; 5:58:22**. I've added a lot of data to get out of memory exception. Garbage collector could cope with so much data.
    - The cycle repeats

I've added data (250 000 1MBt and then 500 000 2MBt) and started Merge sort two times to see G1 work and get out of memory exception (Figure 27 page#33, Figure 28 page#34 and Logs page#35):

- **5:57:27** I've added 250 000 data (1MBt), used heap increased for 2 MBt (both Eden and Old generation were increased for 1 MBt evenly). The same as for previous the array int[] increased its' size in the Sampler
- **5:57:42** I've initiated Merge Sort. Used heap spiked from 5.5 MBt to 8.5 MBt and got down to 5.9 MBt. In line with jstat, old generation increased from 5.5 to 5.9 MBt. CPU usage wasn't significant (Figure 29 page#34)
- **5:58:10** I've added 500 000 data (2 MBt). Used heap increased for 3 MBt (from 6 MBt to 8 MBt). All data went into Old Generation space.
- **5:58:21** I've initiated Merge Sort. Used heap started to fill. Garbage collector couldn't cope with so much data. It initiated Full Pause. The application had been stopped with exception in thread – OutOfMemoryError: Heap space.

Commands

```
2024-04-05T17:56:37.861285600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

2
2
MergeSort Sort start - 17:57:42.882404400
2024-04-05T17:57:42.882404400
2024-04-05T17:57:42.977867100
MergeSort Sort end - 17:57:42.977867100
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
500000
 Array had been created and completed with random values from 1 to 500000
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.
```

```
2
2
MergeSort Sort start - 17:58:21.804466800
2024-04-05T17:58:21.805433100
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at ru.job4j.gc.prof.MergeSort.merge(MergeSort.java:31)
        at ru.job4j.gc.prof.MergeSort.sort(MergeSort.java:21)
        at ru.job4j.gc.prof.MergeSort.sortMerge(MergeSort.java:13)
        at ru.job4j.gc.prof.MergeSort.sort(MergeSort.java:8)
        at ru.job4j.gc.prof.Menu.main(Menu.java:48)
```

Graphs

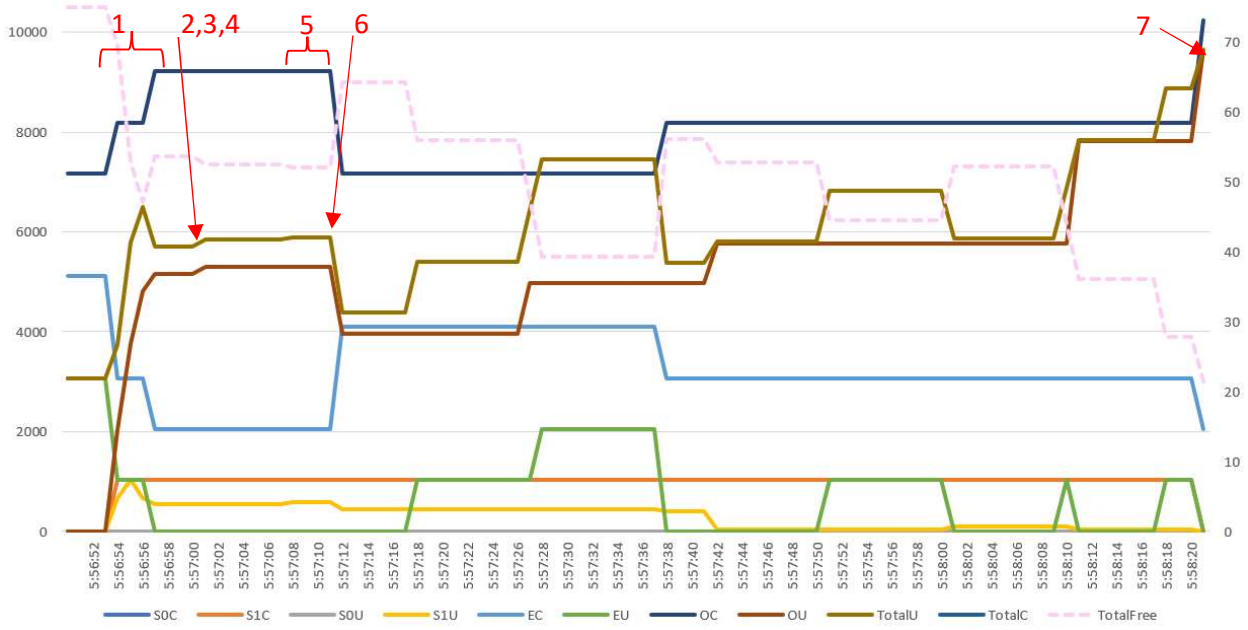


Figure 26 – JSTAT. G1 cycle. 1 - G1 Evacuation Pause; 2 – Concurrent start/ Concurrent Mark Cycle; 3 – Remark; 4 – Cleanup; 5 – Premix preparation; 6 Mixed cleanup; 7 – Full Pause G1 Compaction Pause

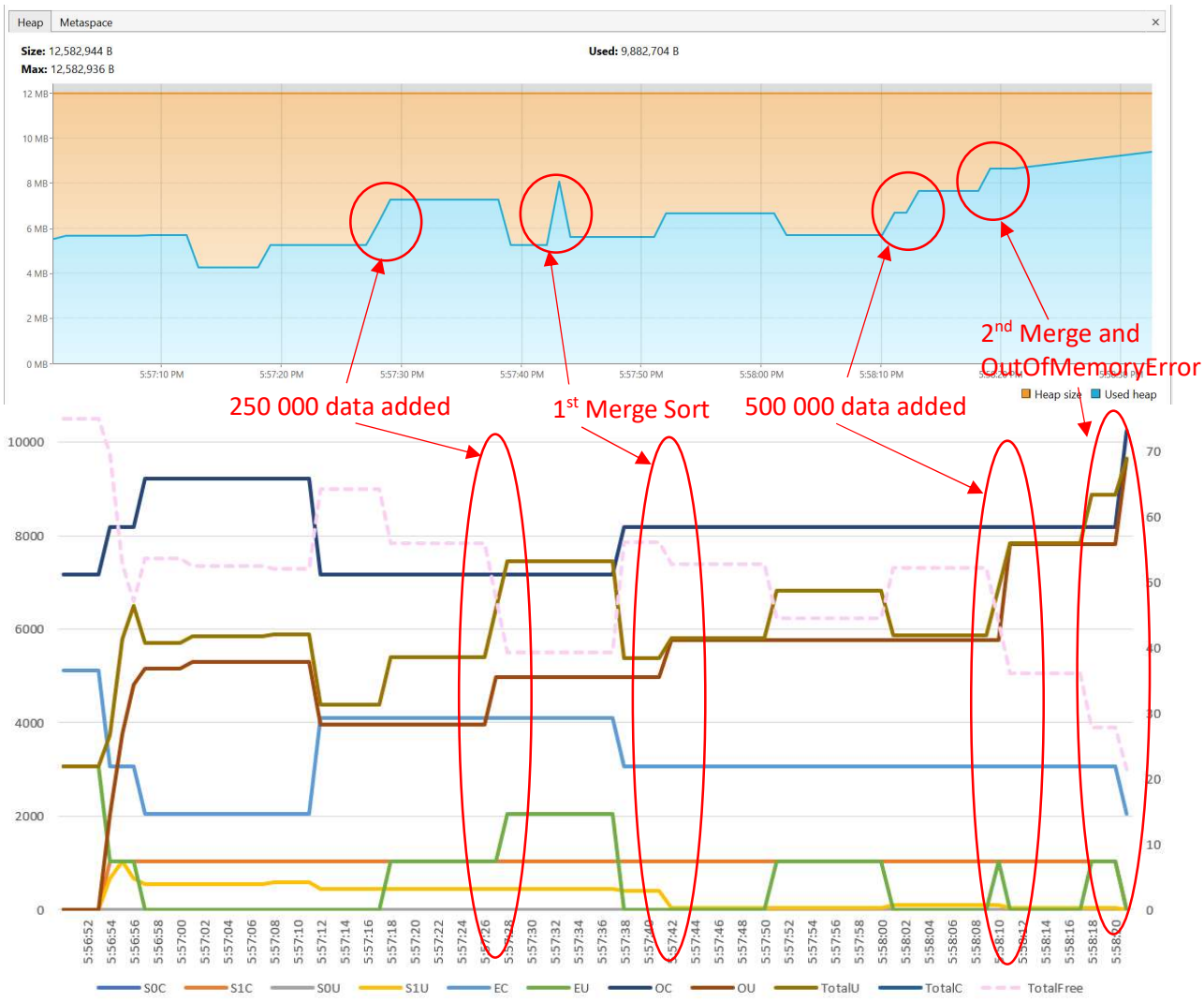


Figure 27 – JSTAT. 5:56:55 – G1 moves data from EDEN to Survivor and to Old Generation. Old/Young generation ratio is increasing.

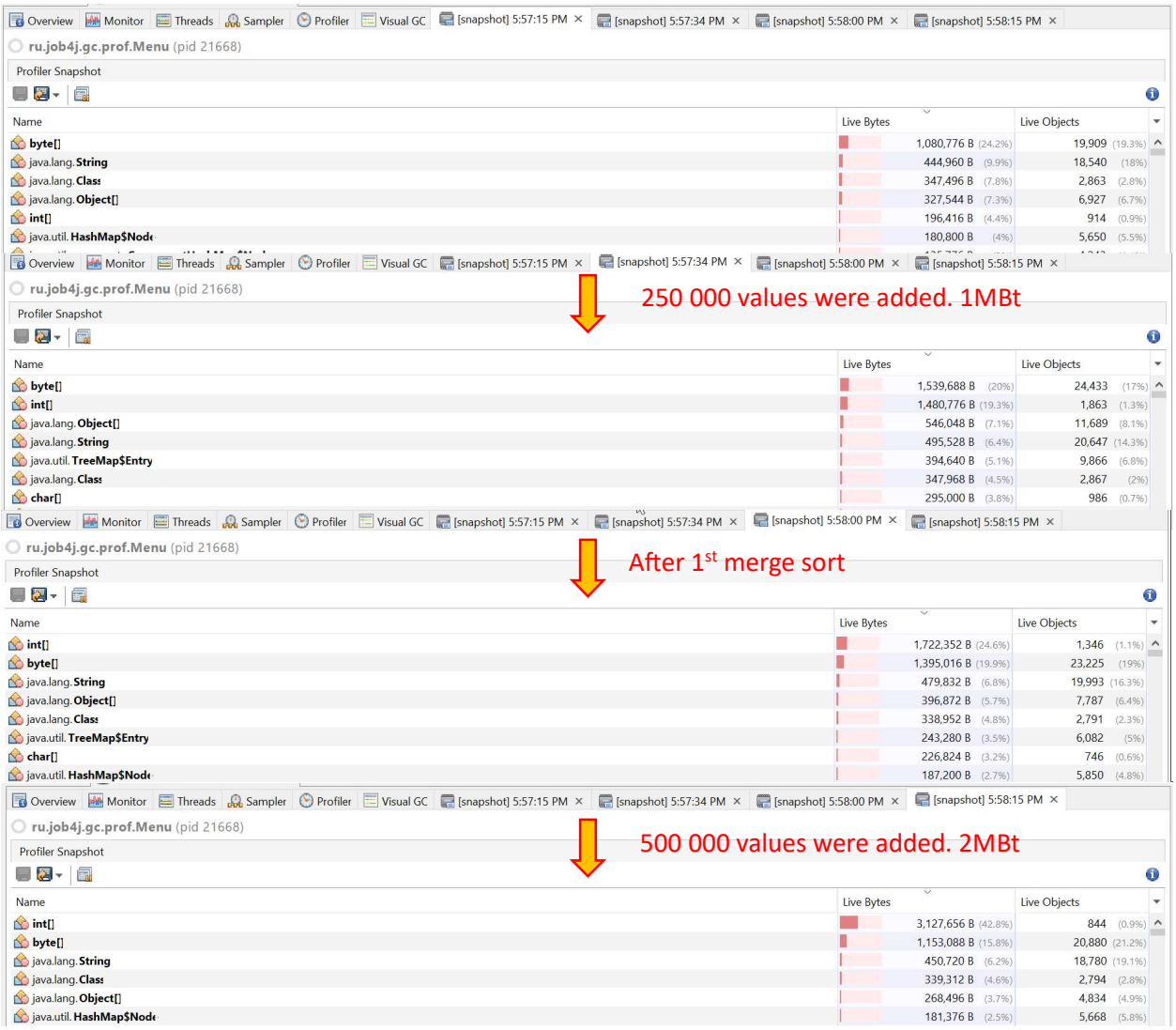


Figure 28 – Snapshots

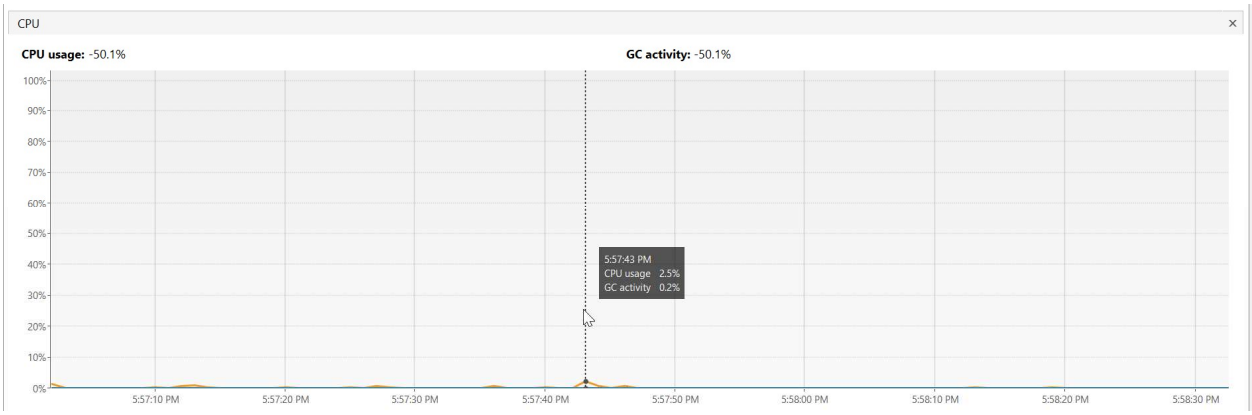


Figure 29 – CPU usage wasn't significant during merge sort = 2.5% only

Logs

- [0.022s][info][gc] Using G1
- [17.310s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 4M->1M(12M) 4.763ms
- [17.350s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 3M->2M(12M) 2.616ms
- [17.406s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 4M->2M(12M) 1.660ms
- [17.474s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 4M->2M(12M) 2.368ms
- [17.495s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 4M->3M(12M) 1.966ms
- [17.513s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 5M->4M(12M) 2.475ms
- [18.587s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 6M->4M(12M) 1.568ms
- [18.648s][info][gc] GC(7) Pause Young (Normal) (G1 Evacuation Pause) 5M->4M(12M) 1.313ms
- [18.699s][info][gc] GC(8) Pause Young (Normal) (G1 Evacuation Pause) 5M->5M(12M) 2.232ms
- [18.814s][info][gc] GC(9) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 1.675ms
- [18.852s][info][gc] GC(10) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 1.684ms
- [19.916s][info][gc] GC(11) Pause Young (Normal) (G1 Evacuation Pause) 7M->5M(12M) 1.785ms
- [23.931s][info][gc] GC(12) Pause Young (Concurrent Start) (G1 Evacuation Pause) 6M->5M(12M) 2.273ms
- [23.931s][info][gc] GC(13) Concurrent Mark Cycle
- [23.937s][info][gc] GC(13) Pause Remark 5M->5M(12M) 2.375ms
- [23.939s][info][gc] GC(13) Pause Cleanup 5M->5M(12M) 0.097ms
- [23.940s][info][gc] GC(13) Concurrent Mark Cycle 9.088ms
- [30.928s][info][gc] GC(14) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 6M->5M(12M) 1.585ms
- [35.079s][info][gc] GC(15) Pause Young (Mixed) (G1 Evacuation Pause) 6M->4M(12M) 2.658ms
- [60.950s][info][gc] GC(16) Pause Young (Normal) (G1 Evacuation Pause) 8M->5M(12M) 1.531ms
- [65.165s][info][gc] GC(17) Pause Young (Concurrent Start) (G1 Humongous Allocation) 5M->5M(12M) 1.774ms
- [65.165s][info][gc] GC(18) Concurrent Undo Cycle
- [65.165s][info][gc] GC(18) Concurrent Undo Cycle 0.145ms
- [65.173s][info][gc] GC(19) Pause Young (Normal) (G1 Evacuation Pause) 9M->6M(12M) 1.677ms
- [65.189s][info][gc] GC(20) Pause Young (Concurrent Start) (G1 Evacuation Pause) 8M->6M(12M) 1.750ms
- [65.189s][info][gc] GC(21) Concurrent Mark Cycle
- [65.196s][info][gc] GC(21) Pause Remark 6M->6M(12M) 3.297ms
- [65.199s][info][gc] GC(21) Pause Cleanup 7M->7M(12M) 0.069ms
- [65.200s][info][gc] GC(21) Concurrent Mark Cycle 10.577ms
- [65.205s][info][gc] GC(22) Pause Young (Normal) (G1 Evacuation Pause) 8M->6M(12M) 1.635ms
- [65.209s][info][gc] GC(23) Pause Young (Concurrent Start) (G1 Evacuation Pause) 8M->6M(12M) 1.617ms
- [65.209s][info][gc] GC(24) Concurrent Mark Cycle
- [65.213s][info][gc] GC(25) Pause Young (Normal) (G1 Evacuation Pause) 8M->6M(12M) 1.720ms
- [65.216s][info][gc] GC(26) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.223ms
- [65.220s][info][gc] GC(24) Pause Remark 7M->7M(12M) 2.888ms
- [65.222s][info][gc] GC(27) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.177ms
- [65.224s][info][gc] GC(28) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.041ms
- [65.225s][info][gc] GC(24) Pause Cleanup 6M->6M(12M) 0.085ms
- [65.225s][info][gc] GC(24) Concurrent Mark Cycle 16.558ms
- [65.227s][info][gc] GC(29) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.010ms
- [65.229s][info][gc] GC(30) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 0.801ms
- [65.229s][info][gc] GC(31) Concurrent Mark Cycle
- [65.232s][info][gc] GC(32) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.607ms

- [65.233s][info][gc] GC(33) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.561ms
- [65.235s][info][gc] GC(34) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.427ms
- [65.239s][info][gc] GC(31) Pause Remark 7M->7M(12M) 2.654ms
- [65.241s][info][gc] GC(35) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.649ms
- [65.242s][info][gc] GC(31) Pause Cleanup 7M->7M(12M) 0.089ms
- [65.243s][info][gc] GC(31) Concurrent Mark Cycle 13.162ms
- [65.244s][info][gc] GC(36) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.667ms
- [65.246s][info][gc] GC(37) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 0.680ms
- [65.246s][info][gc] GC(38) Concurrent Mark Cycle
- [65.248s][info][gc] GC(39) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.420ms
- [65.249s][info][gc] GC(40) Pause Young (Normal) (G1 Evacuation Pause) 7M->7M(12M) 0.379ms
- [65.251s][info][gc] GC(41) Pause Young (Normal) (G1 Evacuation Pause) 8M->7M(12M) 0.405ms
- [65.254s][info][gc] GC(38) Pause Remark 7M->7M(12M) 2.415ms
- [65.255s][info][gc] GC(42) Pause Young (Normal) (G1 Evacuation Pause) 8M->7M(12M) 0.635ms
- [65.257s][info][gc] GC(38) Pause Cleanup 7M->7M(12M) 0.093ms
- [65.258s][info][gc] GC(38) Concurrent Mark Cycle 11.748ms
- [65.391s][info][gc] GC(43) Pause Young (Normal) (G1 Evacuation Pause) 9M->5M(12M) 0.484ms
- [83.969s][info][gc] GC(44) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->5M(12M) 1.052ms
- [83.969s][info][gc] GC(45) Concurrent Mark Cycle
- [83.976s][info][gc] GC(45) Pause Remark 5M->5M(12M) 3.086ms
- [83.978s][info][gc] GC(45) Pause Cleanup 5M->5M(12M) 0.081ms
- [83.979s][info][gc] GC(45) Concurrent Mark Cycle 9.925ms
- [94.489s][info][gc] GC(46) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->5M(12M) 1.157ms
- [94.490s][info][gc] GC(47) Concurrent Mark Cycle
- [94.498s][info][gc] GC(47) Pause Remark 7M->7M(12M) 3.279ms
- [94.500s][info][gc] GC(47) Pause Cleanup 7M->7M(12M) 0.096ms
- [94.500s][info][gc] GC(47) Concurrent Mark Cycle 10.770ms
- [104.087s][info][gc] GC(48) Pause Young (Concurrent Start) (G1 Humongous Allocation) 8M->6M(12M) 1.263ms
- [104.088s][info][gc] GC(49) Concurrent Mark Cycle
- [104.090s][info][gc] GC(50) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.726ms
- [104.092s][info][gc] GC(51) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.888ms
- [104.095s][info][gc] GC(52) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.871ms
- [104.097s][info][gc] GC(49) Pause Remark 9M->9M(12M) 1.960ms
- [104.099s][info][gc] GC(53) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.796ms
- [104.101s][info][gc] GC(54) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.782ms
- [104.102s][info][gc] GC(49) Pause Cleanup 9M->9M(12M) 0.069ms
- [104.102s][info][gc] GC(49) Concurrent Mark Cycle 14.780ms
- [104.104s][info][gc] GC(55) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.818ms
- [104.107s][info][gc] GC(56) Pause Young (Concurrent Start) (G1 Preventive Collection) 9M->8M(12M) 0.875ms
- [104.107s][info][gc] GC(57) Concurrent Mark Cycle

- [104.109s][info][gc] GC(58) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.656ms
- [104.111s][info][gc] GC(59) Pause Young (Normal) (G1 Preventive Collection) 9M->9M(12M) 0.683ms
- [104.113s][info][gc] GC(60) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.643ms
- [104.116s][info][gc] GC(57) Pause Remark 9M->9M(12M) 2.599ms
- [104.118s][info][gc] GC(61) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.847ms
- [104.120s][info][gc] GC(62) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.693ms
- [104.121s][info][gc] GC(57) Pause Cleanup 9M->9M(12M) 0.111ms
- [104.121s][info][gc] GC(57) Concurrent Mark Cycle 14.498ms
- [104.123s][info][gc] GC(63) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.870ms
- [104.126s][info][gc] GC(64) Pause Young (Concurrent Start) (G1 Preventive Collection) 10M->9M(12M) 0.919ms
- [104.126s][info][gc] GC(65) Concurrent Mark Cycle
- [104.128s][info][gc] GC(66) Pause Young (Normal) (G1 Preventive Collection) (Evacuation Failure) 10M->9M(12M) 0.699ms
- [104.129s][info][gc] GC(67) Pause Young (Normal) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.516ms
- [104.148s][info][gc] GC(68) Pause Full (G1 Compaction Pause) 10M->8M(12M) 18.823ms
- [104.149s][info][gc] GC(65) Concurrent Mark Cycle 22.799ms
- [104.151s][info][gc] GC(69) Pause Young (Normal) (G1 Evacuation Pause) 9M->8M(12M) 0.855ms
- [104.153s][info][gc] GC(70) Pause Young (Concurrent Start) (G1 Preventive Collection) 9M->8M(12M) 0.702ms
- [104.153s][info][gc] GC(71) Concurrent Mark Cycle
- [104.155s][info][gc] GC(72) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.785ms
- [104.157s][info][gc] GC(73) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.646ms
- [104.159s][info][gc] GC(74) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.648ms
- [104.161s][info][gc] GC(71) Pause Remark 8M->8M(12M) 1.878ms
- [104.164s][info][gc] GC(75) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.658ms
- [104.165s][info][gc] GC(71) Pause Cleanup 9M->9M(12M) 0.097ms
- [104.166s][info][gc] GC(76) Pause Young (Prepare Mixed) (G1 Preventive Collection) 9M->8M(12M) 0.671ms
- [104.166s][info][gc] GC(71) Concurrent Mark Cycle 13.477ms
- [104.169s][info][gc] GC(77) Pause Young (Mixed) (G1 Preventive Collection) (Evacuation Failure) 9M->8M(12M) 1.673ms
- [104.171s][info][gc] GC(78) Pause Young (Concurrent Start) (G1 Preventive Collection) 9M->8M(12M) 0.808ms
- [104.171s][info][gc] GC(79) Concurrent Mark Cycle
- [104.173s][info][gc] GC(80) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.653ms
- [104.175s][info][gc] GC(81) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.591ms

- [104.177s][info][gc] GC(82) Pause Young (Normal) (G1 Preventive Collection) 8M->8M(12M) 0.480ms
- [104.179s][info][gc] GC(79) Pause Remark 9M->9M(12M) 1.774ms
- [104.181s][info][gc] GC(83) Pause Young (Normal) (G1 Preventive Collection) 9M->9M(12M) 0.629ms
- [104.183s][info][gc] GC(79) Pause Cleanup 9M->9M(12M) 0.075ms
- [104.183s][info][gc] GC(79) Concurrent Mark Cycle 11.288ms
- [104.183s][info][gc] GC(84) Pause Young (Prepare Mixed) (G1 Preventive Collection) 10M->9M(12M) 0.481ms
- [104.186s][info][gc] GC(85) Pause Young (Mixed) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 1.646ms
- [104.188s][info][gc] GC(86) Pause Young (Concurrent Start) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.729ms
- [104.188s][info][gc] GC(88) Concurrent Mark Cycle
- [104.204s][info][gc] GC(87) Pause Full (G1 Compaction Pause) 10M->8M(12M) 15.340ms
- [104.204s][info][gc] GC(88) Concurrent Mark Cycle 15.517ms
- [104.206s][info][gc] GC(89) Pause Young (Normal) (G1 Evacuation Pause) 9M->8M(12M) 0.783ms
- [104.208s][info][gc] GC(90) Pause Young (Concurrent Start) (G1 Preventive Collection) 9M->8M(12M) 0.975ms
- [104.208s][info][gc] GC(91) Concurrent Mark Cycle
- [104.210s][info][gc] GC(92) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.698ms
- [104.212s][info][gc] GC(93) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.651ms
- [104.214s][info][gc] GC(94) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.653ms
- [104.217s][info][gc] GC(91) Pause Remark 9M->9M(12M) 2.006ms
- [104.219s][info][gc] GC(95) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 0.638ms
- [104.222s][info][gc] GC(96) Pause Young (Normal) (G1 Preventive Collection) 9M->8M(12M) 1.257ms
- [104.222s][info][gc] GC(91) Pause Cleanup 8M->8M(12M) 0.082ms
- [104.222s][info][gc] GC(91) Concurrent Mark Cycle 14.076ms
- [104.224s][info][gc] GC(97) Pause Young (Prepare Mixed) (G1 Preventive Collection) 9M->8M(12M) 0.604ms
- [104.228s][info][gc] GC(98) Pause Young (Mixed) (G1 Preventive Collection) (Evacuation Failure) 9M->8M(12M) 2.866ms
- [104.230s][info][gc] GC(99) Pause Young (Concurrent Start) (G1 Preventive Collection) 9M->9M(12M) 0.733ms
- [104.230s][info][gc] GC(100) Concurrent Mark Cycle
- [104.232s][info][gc] GC(101) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.660ms
- [104.234s][info][gc] GC(102) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.621ms
- [104.236s][info][gc] GC(103) Pause Young (Normal) (G1 Preventive Collection) (Evacuation Failure) 10M->9M(12M) 0.580ms
- [104.239s][info][gc] GC(100) Pause Remark 9M->9M(12M) 1.911ms
- [104.241s][info][gc] GC(104) Pause Young (Normal) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.611ms
- [104.242s][info][gc] GC(100) Pause Cleanup 10M->10M(12M) 0.071ms
- [104.242s][info][gc] GC(100) Concurrent Mark Cycle 11.842ms

- [104.243s][info][gc] GC(105) Pause Young (Prepare Mixed) (G1 Evacuation Pause) (Evacuation Failure) 11M->11M(12M) 0.609ms
- [104.257s][info][gc] GC(106) Pause Full (G1 Compaction Pause) 11M->9M(12M) 14.111ms
- [104.260s][info][gc] GC(107) Pause Young (Concurrent Start) (G1 Preventive Collection) 10M->9M(12M) 1.055ms
- [104.260s][info][gc] GC(108) Concurrent Mark Cycle
- [104.262s][info][gc] GC(109) Pause Young (Normal) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.523ms
- [104.264s][info][gc] GC(110) Pause Young (Normal) (G1 Evacuation Pause) (Evacuation Failure) 11M->11M(12M) 0.766ms
- [104.278s][info][gc] GC(111) Pause Full (G1 Compaction Pause) 11M->9M(12M) 13.973ms
- [104.278s][info][gc] GC(108) Concurrent Mark Cycle 18.366ms
- [104.280s][info][gc] GC(112) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.794ms
- [104.282s][info][gc] GC(113) Pause Young (Concurrent Start) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.860ms
- [104.282s][info][gc] GC(115) Concurrent Mark Cycle
- [104.297s][info][gc] GC(114) Pause Full (G1 Compaction Pause) 10M->9M(12M) 14.350ms
- [104.297s][info][gc] GC(115) Concurrent Mark Cycle 14.525ms
- [104.300s][info][gc] GC(116) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.945ms
- [104.302s][info][gc] GC(117) Pause Young (Concurrent Start) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.965ms
- [104.302s][info][gc] GC(119) Concurrent Mark Cycle
- [104.318s][info][gc] GC(118) Pause Full (G1 Compaction Pause) 10M->9M(12M) 16.378ms
- [104.319s][info][gc] GC(119) Concurrent Mark Cycle 16.573ms
- [104.321s][info][gc] GC(120) Pause Young (Normal) (G1 Preventive Collection) 10M->9M(12M) 0.812ms
- [104.323s][info][gc] GC(121) Pause Young (Concurrent Start) (G1 Evacuation Pause) (Evacuation Failure) 10M->10M(12M) 0.819ms
- [104.323s][info][gc] GC(122) Concurrent Mark Cycle
- [104.325s][info][gc] GC(123) Pause Young (Normal) (G1 Humongous Allocation) (Evacuation Failure) 10M->10M(12M) 0.716ms
- [104.340s][info][gc] GC(124) Pause Full (G1 Compaction Pause) 10M->9M(12M) 14.913ms
- [104.340s][info][gc] GC(122) Concurrent Mark Cycle 17.275ms
- [104.343s][info][gc] GC(125) Pause Young (Concurrent Start) (G1 Humongous Allocation) 10M->10M(12M) 0.561ms
- [104.343s][info][gc] GC(126) Concurrent Mark Cycle
- [104.343s][info][gc] GC(127) Pause Young (Normal) (G1 Humongous Allocation) 10M->10M(12M) 0.349ms
- [104.358s][info][gc] GC(128) Pause Full (G1 Compaction Pause) 10M->9M(12M) 14.921ms
- [104.375s][info][gc] GC(129) Pause Full (G1 Compaction Pause) 9M->9M(12M) 17.275ms
- [104.376s][info][gc] GC(126) Concurrent Mark Cycle 32.974ms

Bubble and Insert Sorts

Description

- **05:17:40** – 250000 integer values had been added (1MBt). G1 garbage collector put everything into Old Generation into humongous memory space (see humongous allocation in the log)
- **05:17:55** – Bubble sort had been initiated. G1 garbage collector put again everything into Old Generation into humongous memory space (see humongous allocation in the log). Concurrent Mark Cycle/ Remark and Cleanup had been initiated
- **05:18:16** – Garbage collector initiated prepared mixed to calculate data which should be deleted based on mark and statistics
- **05:18:26** – Mixed garbage collection had been initiated. The trash had been collected
- **05:18:46** - Concurrent Mark Cycle/ Remark/ Cleanup/ Prepare Mixed and Mixed were repeated
- **05:19:54** – Bubble sort had been completed
- **05:19:59** - Pause Young (Normal) (G1 Evacuation Pause) had been initiated by G1. Garbage after bubble sort had been completed. Old/ Young generation ratio had been changed
- **05:20:05** - 250000 integer values had been added (1MBt). G1 garbage collector put everything into Old Generation into humongous memory space (see humongous allocation in the log)
- **05:20:16** – Insert sort had been initiated. G1 garbage collector put again everything into Old Generation into humongous memory space (see humongous allocation in the log). It last for 15 seconds. No significant changes was noted during insert sort

Commands

```
2024-04-06T17:17:02.513921100
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-06T17:17:40.269984500
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

3
3
BubbleSort Sort start - 17:17:55.686004800
2024-04-06T17:17:55.686004800
BubbleSort Sort end - 17:19:54.421950500
2024-04-06T17:19:54.422953400
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
```

```
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-06T17:20:06.656144
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

4
4
InsertSort Sort start - 17:20:16.294623200
2024-04-06T17:20:16.294623200
InsertSort Sort end - 17:20:32.276191400
2024-04-06T17:20:32.276191400
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

6
6
The End
2024-04-06T17:20:41.256511600
```

Graphs

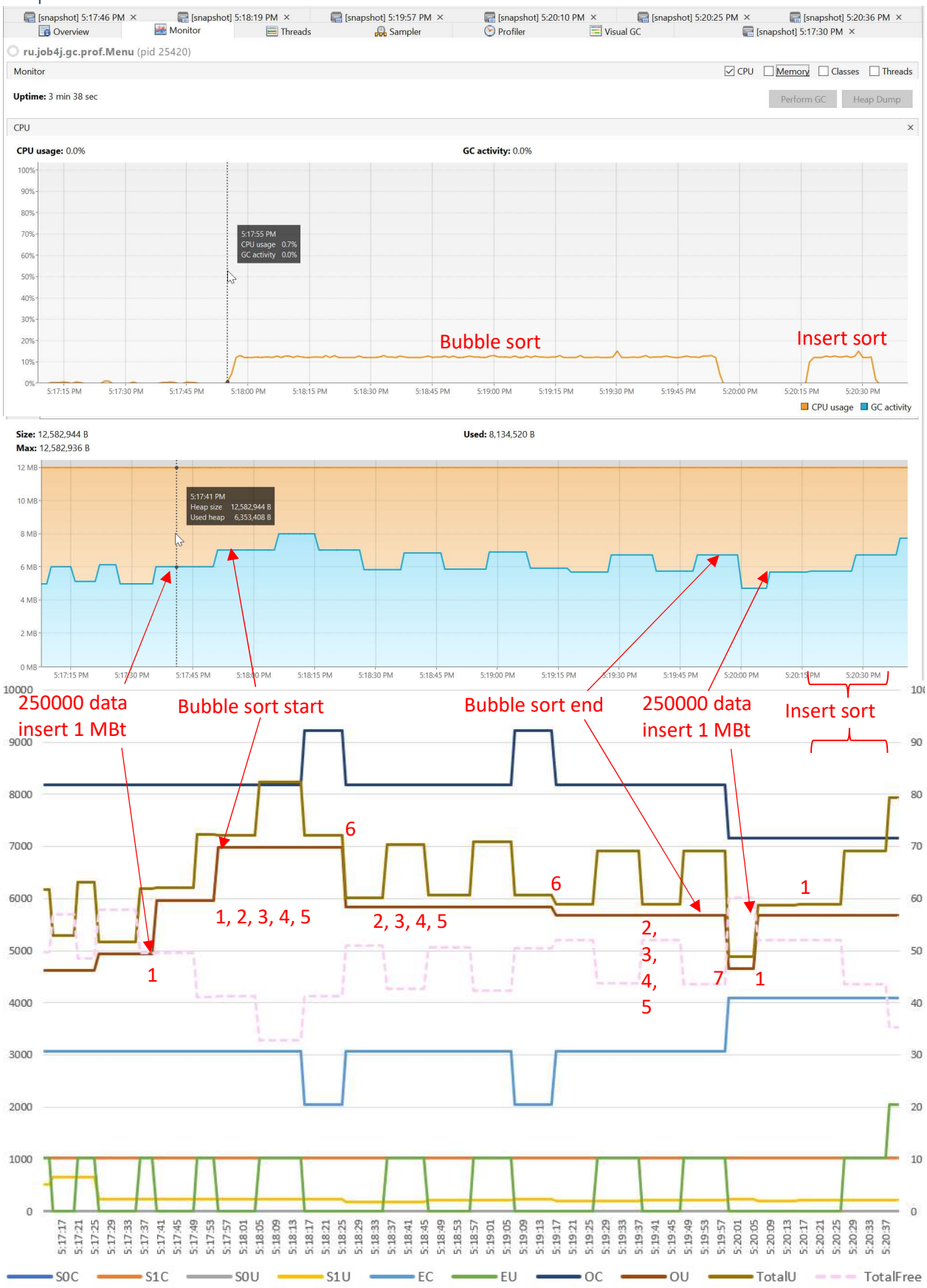


Figure 30 – 1 humongous allocation; 2 Concurrent Mark Cycle; 3 remark; 4 cleanup; 5 prepare mixed; 6 mixed (old and young) data deletion from the heap; 7 Young pause (G1 Evacuation Pause) - it moves data between Eden, Survivor and to Old generation.

Logs

[0.025s][info][gc] Using G1
[3.933s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 4M->1M(12M) 4.560ms
[3.994s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 3M->2M(12M) 2.945ms
[4.061s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 4M->2M(12M) 1.236ms
[4.124s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 4M->2M(12M) 1.728ms
[4.148s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 4M->3M(12M) 1.993ms
[4.169s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 5M->4M(12M) 2.154ms
[5.339s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 6M->4M(12M) 1.289ms
[5.453s][info][gc] GC(7) Pause Young (Normal) (G1 Evacuation Pause) 6M->4M(12M) 1.709ms
[5.612s][info][gc] GC(8) Pause Young (Normal) (G1 Evacuation Pause) 6M->4M(12M) 2.996ms
[5.651s][info][gc] GC(9) Pause Young (Normal) (G1 Evacuation Pause) 5M->5M(12M) 1.455ms
[12.690s][info][gc] GC(10) Pause Young (Normal) (G1 Evacuation Pause) 7M->5M(12M) 1.877ms
[23.692s][info][gc] GC(11) Pause Young (Normal) (G1 Evacuation Pause) 7M->5M(12M) 1.842ms
[37.896s][info][gc] GC(12) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->5M(12M) 1.427ms
[37.896s][info][gc] GC(13) Concurrent Undo Cycle
[37.896s][info][gc] GC(13) Concurrent Undo Cycle 0.176ms
[53.327s][info][gc] GC(14) Pause Young (Concurrent Start) (G1 Humongous Allocation) 7M->6M(12M) 1.326ms
[53.327s][info][gc] GC(15) Concurrent Mark Cycle
[53.335s][info][gc] GC(15) Pause Remark 7M->7M(12M) 3.317ms
[53.338s][info][gc] GC(15) Pause Cleanup 7M->7M(12M) 0.107ms
[53.338s][info][gc] GC(15) Concurrent Mark Cycle 10.970ms
[73.704s][info][gc] GC(16) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 9M->7M(12M) 1.777ms
[83.758s][info][gc] GC(17) Pause Young (Mixed) (G1 Evacuation Pause) 8M->5M(12M) 2.733ms
[103.782s][info][gc] GC(18) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->5M(12M) 1.325ms
[103.782s][info][gc] GC(19) Concurrent Mark Cycle
[103.788s][info][gc] GC(19) Pause Remark 5M->5M(12M) 1.892ms
[103.790s][info][gc] GC(19) Pause Cleanup 5M->5M(12M) 0.058ms
[103.790s][info][gc] GC(19) Concurrent Mark Cycle 7.997ms
[124.807s][info][gc] GC(20) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 7M->5M(12M) 1.615ms
[134.820s][info][gc] GC(21) Pause Young (Mixed) (G1 Evacuation Pause) 6M->5M(12M) 3.651ms
[155.843s][info][gc] GC(22) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->5M(12M) 1.999ms
[155.843s][info][gc] GC(23) Concurrent Mark Cycle
[155.854s][info][gc] GC(23) Pause Remark 5M->5M(12M) 3.230ms
[155.858s][info][gc] GC(23) Pause Cleanup 5M->5M(12M) 0.080ms
[155.858s][info][gc] GC(23) Concurrent Mark Cycle 15.296ms
[176.865s][info][gc] GC(24) Pause Young (Normal) (G1 Evacuation Pause) 7M->4M(12M) 1.625ms
[184.292s][info][gc] GC(25) Pause Young (Concurrent Start) (G1 Humongous Allocation) 5M->4M(12M) 1.653ms
[184.292s][info][gc] GC(26) Concurrent Undo Cycle
[184.292s][info][gc] GC(26) Concurrent Undo Cycle 0.119ms
[193.936s][info][gc] GC(27) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->4M(12M) 1.741ms
[193.936s][info][gc] GC(28) Concurrent Undo Cycle
[193.936s][info][gc] GC(28) Concurrent Undo Cycle 0.197ms

ZGC

Description

15:37:05 - Array with 250 000 integers had been added. Nothing noted in logs and in VisualVM
15:37:13 – Merge Sort is started. Nothing noted in logs and in VisualVM
15:37:21 - Array with 250 000 integers had been added. Nothing noted in logs and in VisualVM
15:37:33 - Bubble Sort is started. Nothing noted in logs and in VisualVM
15:39:23 - Garbage Collection (Warmup) - it gathers statistics and adapts its behavior to optimize garbage collection performance. Used heap increased from 0 to 75 MBt
15:39:36 – Array with 250 000 integers had been added. Nothing noted in logs and in VisualVM
15:39:45 - Insert Sort is started. Nothing noted in logs and in VisualVM
//I've decided to add 10000000 integers to see a difference
15:40:24 - Array with 10 000 000 integers had been added. Nothing noted in logs and in VisualVM
15:40:39 – Merge set for 10 M elements had been started. Used heap increased from 75 MBt to 180 MBt. After two warm up, ZGC announced Allocation Rate – which should be allocation rate of new data
15:40:42 – Garbage collector announced Proactive – run ZGC proactively. Used heap decreased from 180 MBt down to 52 MBt
15:41:12 - Array with 10 000 000 integers had been added. Nothing noted in logs and in VisualVM
15:41:23 – Insert sort is started. Used heap increased up to 102 MBt. Garbage collector announced Proactive – run ZGC proactively.
//Sorting took too much time. I've stopped the application

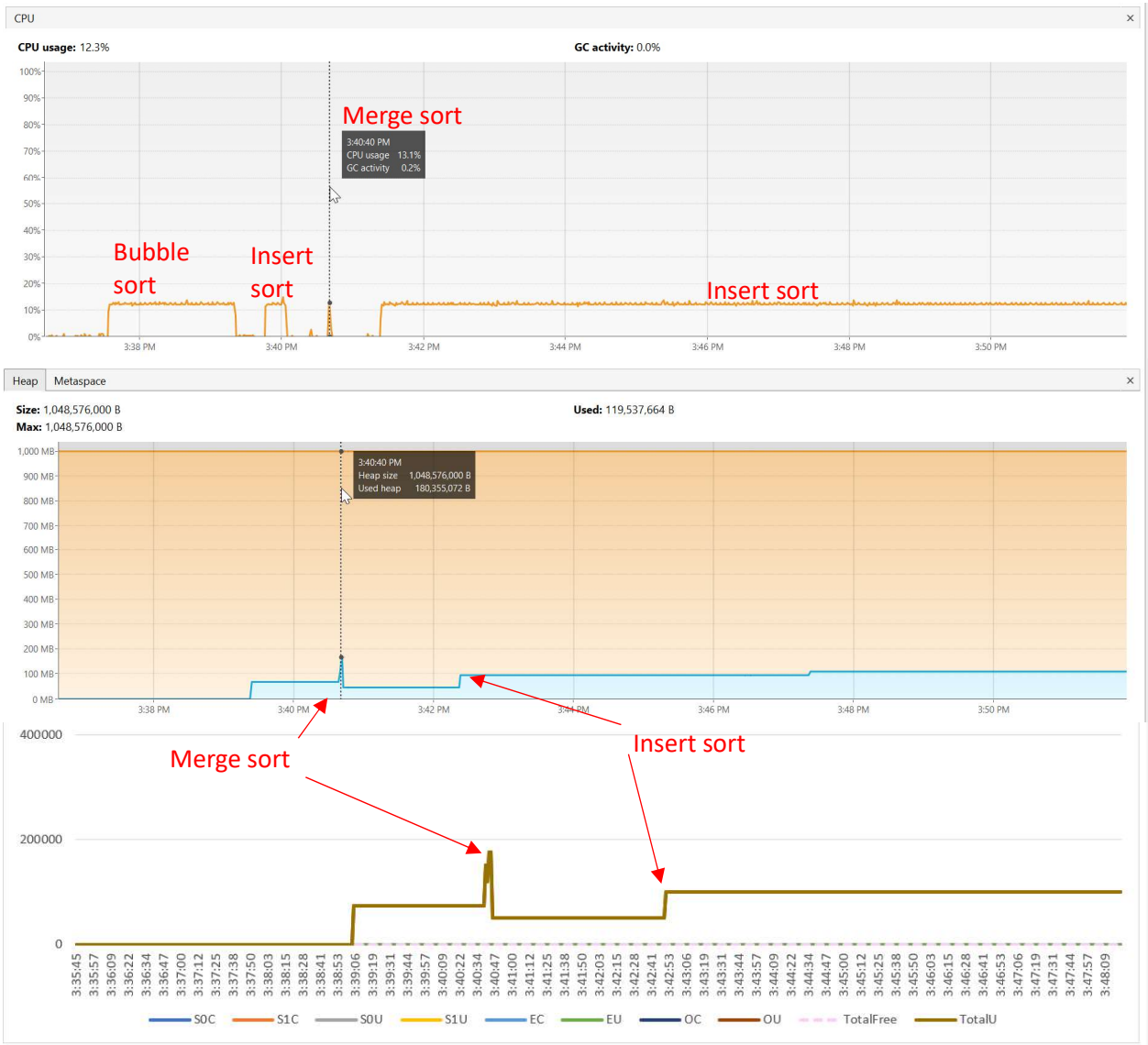
Commands

```
2024-04-08T15:36:34.720198200
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-08T15:37:05.676331600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

2
2
MergeSort Sort start - 15:37:13.316236800
2024-04-08T15:37:13.316236800
MergeSort Sort end - 15:37:13.382761700
2024-04-08T15:37:13.382761700
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-08T15:37:21.711210500
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

3
3
BubbleSort Sort start - 15:37:33.031819500
2024-04-08T15:37:33.032818300
BubbleSort Sort end - 15:39:20.526762600
2024-04-08T15:39:20.526762600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
```

```
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
250000
 Array had been created and completed with random values from 1 to 250000
2024-04-08T15:39:36.637874300
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

4
4
InsertSort Sort start - 15:39:45.977203900
2024-04-08T15:39:45.977203900
InsertSort Sort end - 15:40:03.849381100
2024-04-08T15:40:03.850378700
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
 Enter number of elements -
10000000
 Array had been created and completed with random values from 1 to 10000000
2024-04-08T15:40:24.414009600
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

2
2
MergeSort Sort start - 15:40:39.058894900
2024-04-08T15:40:39.058894900
MergeSort Sort end - 15:40:41.108569900
2024-04-08T15:40:41.108569900
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

1
1
```

```
 Enter number of elements -
10000000
 Array had been created and completed with random values from 1 to 10000000
2024-04-08T15:41:12.236966900
   Enter 1 to create an array.
   Enter 2 to sort using merge_sort.
   Enter 3 to sort using bubble_sort.
   Enter 4 to sort using insert_sort.
   Enter another number for exit.

4
4
InsertSort Sort start - 15:41:23.401029
2024-04-08T
```

Graphs


CPU
CPU usage: 12.3%
GC activity: 0.0%
Merge sort
Bubble sort
Insert sort
Insert sort
Heap
Metaspace
Size: 1,048,576,000 B
Max: 1,048,576,000 B
Used: 119,537,664 B
Merge sort
Insert sort
S0C
S1C
S0U
S1U
EC
EU
OC
OU
TotalFree
TotalU

Logs

```
[0.097s][info][gc] Using The Z Garbage Collector
[168.474s][info][gc] GC(0) Garbage Collection (Warmup) 100M(10%)->26M(3%)
[244.756s][info][gc] GC(1) Garbage Collection (Warmup) 228M(23%)->136M(14%)
[245.059s][info][gc] GC(2) Garbage Collection (Warmup) 338M(34%)->116M(12%)
[246.262s][info][gc] GC(3) Garbage Collection (Allocation Rate) 898M(90%)->162M(16%)
[247.946s][info][gc] GC(4) Garbage Collection (Proactive) 410M(41%)->46M(5%)
[347.563s][info][gc] GC(5) Garbage Collection (Proactive) 146M(15%)->90M(9%)
[647.655s][info][gc] GC(6) Garbage Collection (Proactive) 130M(13%)->94M(9%)
```