Parallel

19:43 запуск программы

19:45 создание массива на 250000 элементов

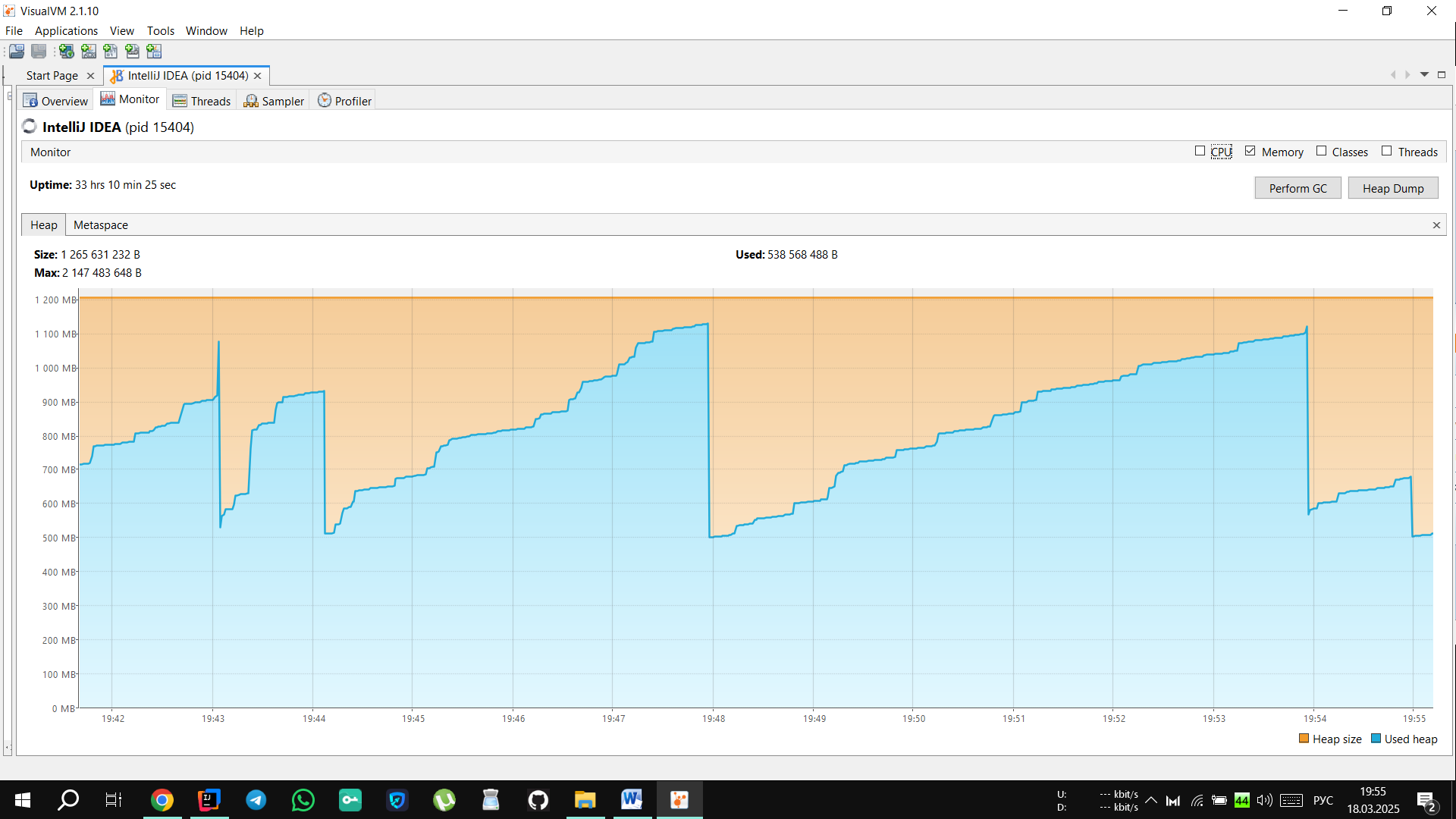
19:47 сортировка слиянием

19:49 сортировка вставкой

19:51 сортировка пузырьком

лог:

[0.015s][info][gc] Using Parallel  
[0.265s][info][gc] GC(0) Pause Young (Allocation Failure) 3M->1M(11M) 4.126ms  
[0.509s][info][gc] GC(1) Pause Young (Allocation Failure) 4M->3M(11M) 4.329ms  
[12.632s][info][gc] GC(2) Pause Young (Allocation Failure) 6M->3M(11M) 2.907ms



запуск программы

сортировка пузырьком

BubbleSort Начало сортировки - 19:51:04.320793600

BubbleSort Конец сортировки - 19:52:45.326896800

сортировка вставкой

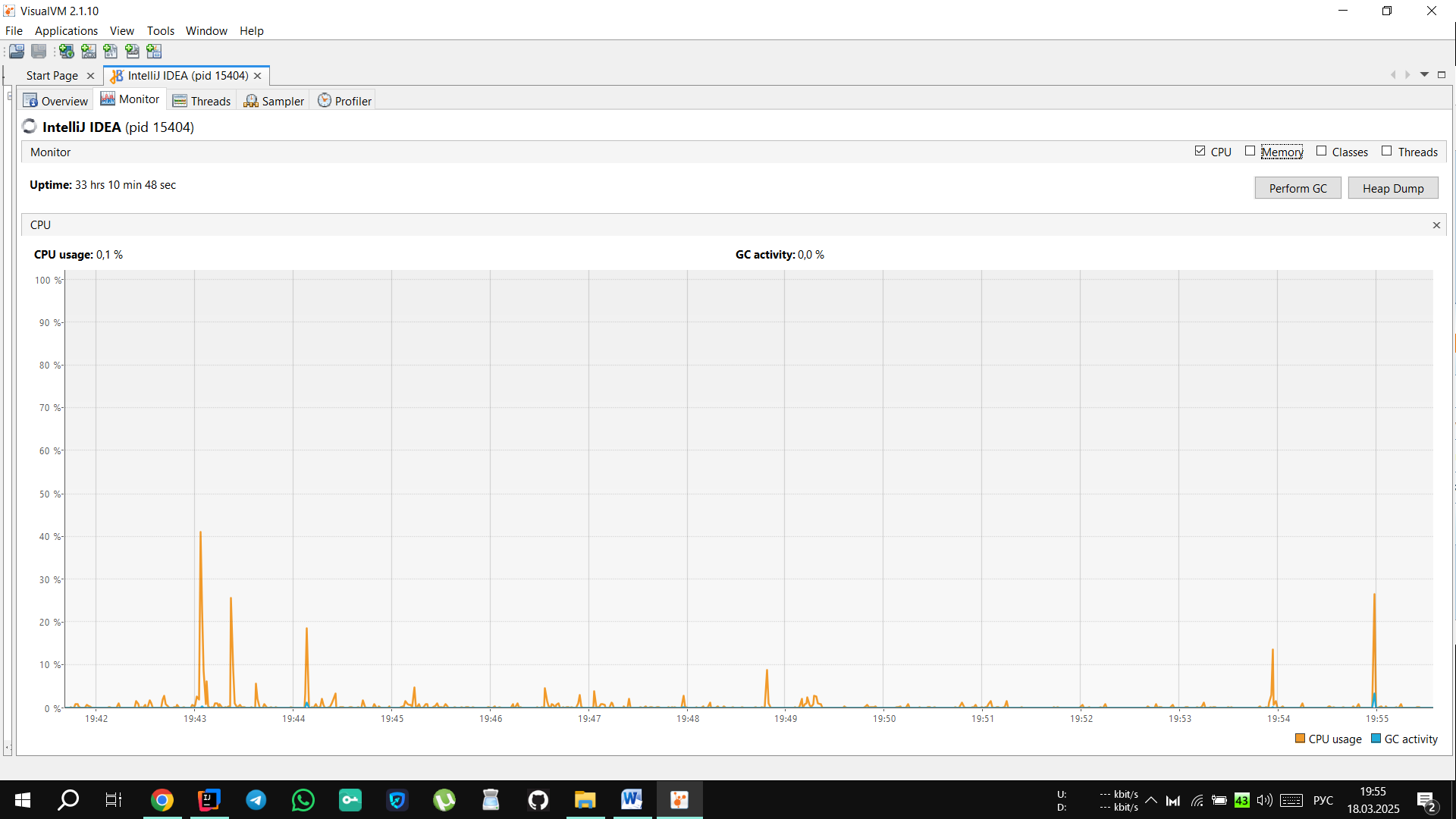
InsertSort Начало сортировки - 19:49:08.923307100

InsertSort Конец сортировки - 19:49:13.622396400

сортировка слиянием

MergeSort Начало сортировки - 19:47:03.141275500

MergeSort Конец сортировки - 19:47:03.216254700



запуск программы

сортировка пузырьком

BubbleSort Начало сортировки - 19:51:04.320793600

BubbleSort Конец сортировки - 19:52:45.326896800

сортировка вставкой

InsertSort Начало сортировки - 19:49:08.923307100

InsertSort Конец сортировки - 19:49:13.622396400

сортировка слиянием

MergeSort Начало сортировки - 19:47:03.141275500

MergeSort Конец сортировки - 19:47:03.216254700

В логе видно, что сборки происходят довольно часто (первая сборка через 0.265 секунды, вторая через 0.509 секунды, и третья через 12.632 секунды). Это может указывать на то, что приложение испытывает давление на память, и объекты в молодом поколении не успевают «выживать» достаточно долго, чтобы перейти в старое поколение. Время, затраченное на сборки, относительно невелико (в пределах 4.126 до 4.329 миллисекунд), что означает, что сборщик мусора работает эффективно, но частые вызовы могут указывать на необходимость оптимизации использования памяти.

G1

23:01 запуск программы

23:02 создание массива на 250000 элементов

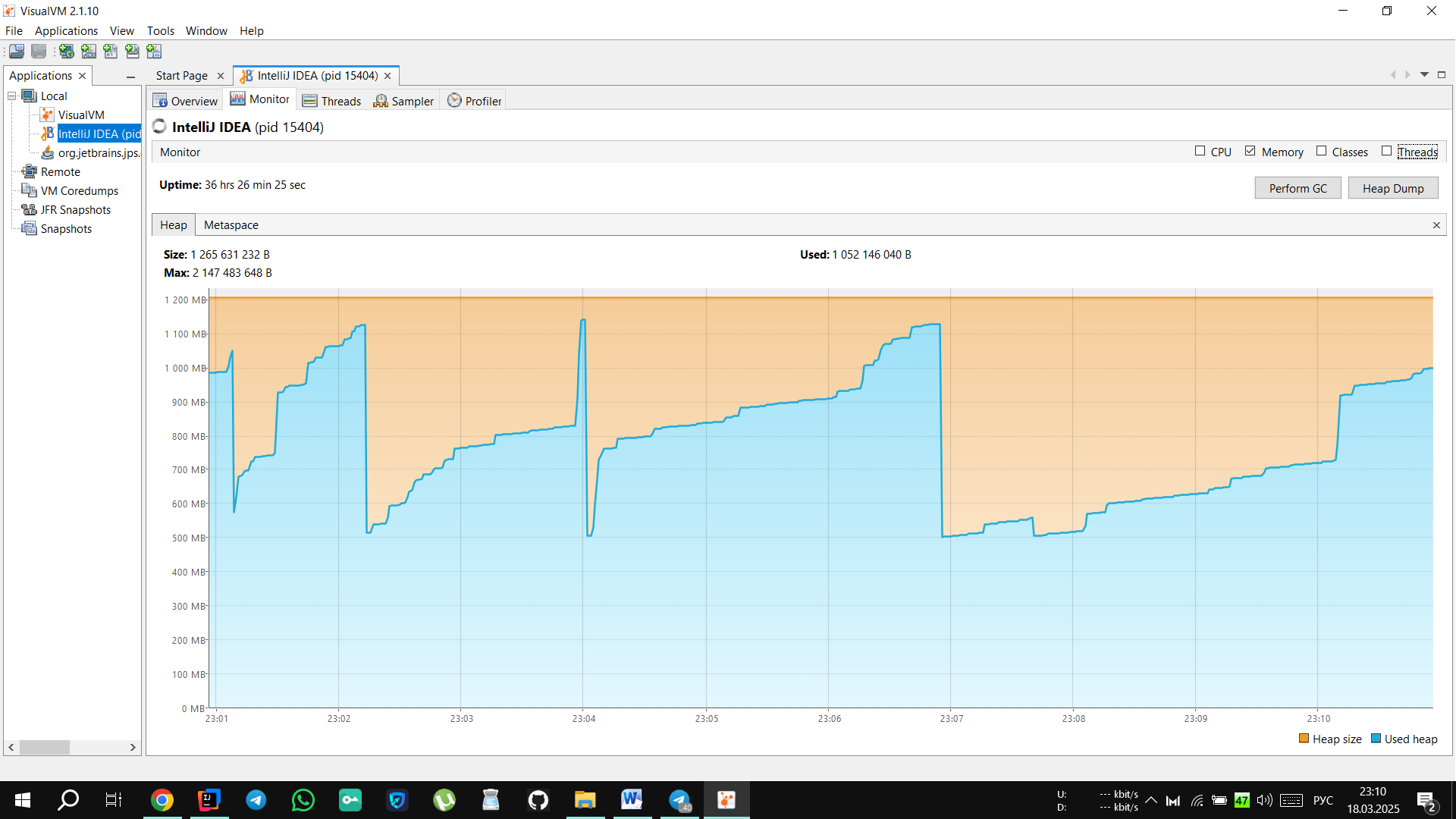
23:04 сортировка слиянием

23:06 сортировка вставкой

23:08 сортировка пузырьком

лог:

[0.014s][info][gc] Using G1  
[0.283s][info][gc] GC(0) Pause Young (Normal) (G1 Evacuation Pause) 4M->2M(12M) 6.586ms  
[0.415s][info][gc] GC(1) Pause Young (Normal) (G1 Evacuation Pause) 3M->3M(12M) 3.932ms  
[0.501s][info][gc] GC(2) Pause Young (Normal) (G1 Evacuation Pause) 4M->3M(12M) 2.151ms  
[0.782s][info][gc] GC(3) Pause Young (Normal) (G1 Evacuation Pause) 5M->3M(12M) 3.305ms  
[10.657s][info][gc] GC(4) Pause Young (Normal) (G1 Evacuation Pause) 4M->4M(12M) 2.837ms  
[89.859s][info][gc] GC(5) Pause Young (Normal) (G1 Evacuation Pause) 6M->3M(12M) 0.905ms  
[170.002s][info][gc] GC(6) Pause Young (Normal) (G1 Evacuation Pause) 6M->4M(12M) 0.988ms  
[221.763s][info][gc] GC(7) Pause Young (Concurrent Start) (G1 Humongous Allocation) 6M->4M(12M) 1.839ms  
[221.772s][info][gc] GC(8) Concurrent Undo Cycle  
[221.773s][info][gc] GC(8) Concurrent Undo Cycle 0.235ms  
[221.781s][info][gc] GC(9) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 1.182ms  
[221.785s][info][gc] GC(10) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 1.515ms  
[221.785s][info][gc] GC(11) Concurrent Mark Cycle  
[221.793s][info][gc] GC(11) Pause Remark 6M->6M(12M) 3.340ms  
[221.795s][info][gc] GC(11) Pause Cleanup 6M->6M(12M) 0.047ms  
[221.795s][info][gc] GC(11) Concurrent Mark Cycle 10.152ms  
[221.809s][info][gc] GC(12) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 8M->6M(12M) 1.367ms  
[221.819s][info][gc] GC(13) Pause Young (Mixed) (G1 Evacuation Pause) 7M->5M(12M) 2.049ms  
[221.829s][info][gc] GC(14) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->5M(12M) 1.296ms  
[221.829s][info][gc] GC(15) Concurrent Mark Cycle  
[221.831s][info][gc] GC(16) Pause Young (Normal) (G1 Evacuation Pause) 6M->5M(12M) 0.937ms  
[221.834s][info][gc] GC(17) Pause Young (Normal) (G1 Evacuation Pause) 6M->6M(12M) 1.012ms  
[221.836s][info][gc] GC(15) Pause Remark 6M->6M(12M) 1.273ms  
[221.838s][info][gc] GC(18) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.938ms  
[221.838s][info][gc] GC(15) Pause Cleanup 6M->6M(12M) 0.028ms  
[221.838s][info][gc] GC(15) Concurrent Mark Cycle 9.814ms  
[221.840s][info][gc] GC(19) Pause Young (Prepare Mixed) (G1 Evacuation Pause) 7M->6M(12M) 0.900ms  
[221.844s][info][gc] GC(20) Pause Young (Mixed) (G1 Evacuation Pause) 6M->5M(12M) 2.241ms  
[221.847s][info][gc] GC(21) Pause Young (Concurrent Start) (G1 Evacuation Pause) 6M->5M(12M) 0.685ms  
[221.847s][info][gc] GC(22) Concurrent Mark Cycle  
[221.850s][info][gc] GC(23) Pause Young (Normal) (G1 Evacuation Pause) 6M->6M(12M) 0.758ms  
[221.851s][info][gc] GC(24) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.350ms  
[221.854s][info][gc] GC(22) Pause Remark 6M->6M(12M) 2.270ms  
[221.854s][info][gc] GC(22) Pause Cleanup 6M->6M(12M) 0.004ms  
[221.854s][info][gc] GC(22) Concurrent Mark Cycle 7.572ms  
[221.856s][info][gc] GC(25) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.375ms  
[221.857s][info][gc] GC(26) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 0.394ms  
[221.857s][info][gc] GC(27) Concurrent Mark Cycle  
[221.859s][info][gc] GC(28) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.380ms  
[221.862s][info][gc] GC(29) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.396ms  
[221.863s][info][gc] GC(27) Pause Remark 6M->6M(12M) 1.280ms  
[221.863s][info][gc] GC(27) Pause Cleanup 6M->6M(12M) 0.002ms  
[221.863s][info][gc] GC(27) Concurrent Mark Cycle 5.821ms  
[221.865s][info][gc] GC(30) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.351ms  
[221.867s][info][gc] GC(31) Pause Young (Concurrent Start) (G1 Evacuation Pause) 7M->6M(12M) 0.439ms  
[221.867s][info][gc] GC(32) Concurrent Mark Cycle  
[221.869s][info][gc] GC(33) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.379ms  
[221.870s][info][gc] GC(34) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.314ms  
[221.873s][info][gc] GC(32) Pause Remark 7M->7M(12M) 1.208ms  
[221.873s][info][gc] GC(32) Pause Cleanup 7M->7M(12M) 0.003ms  
[221.873s][info][gc] GC(32) Concurrent Mark Cycle 5.877ms  
[221.874s][info][gc] GC(35) Pause Young (Normal) (G1 Evacuation Pause) 7M->6M(12M) 0.382ms  
[221.877s][info][gc] GC(36) Pause Young (Concurrent Start) (G1 Humongous Allocation) 8M->6M(12M) 0.565ms  
[221.877s][info][gc] GC(37) Concurrent Mark Cycle  
[221.882s][info][gc] GC(37) Pause Remark 7M->7M(12M) 1.372ms  
[221.882s][info][gc] GC(37) Pause Cleanup 7M->7M(12M) 0.004ms  
[221.882s][info][gc] GC(37) Concurrent Mark Cycle 4.993ms  
[246.046s][info][gc] GC(38) Pause Young (Normal) (G1 Evacuation Pause) 8M->5M(12M) 0.812ms  
[321.014s][info][gc] GC(39) Pause Young (Normal) (G1 Evacuation Pause) 7M->5M(12M) 0.827ms  
[335.870s][info][gc] GC(40) Pause Young (Concurrent Start) (G1 Humongous Allocation) 5M->5M(12M) 0.864ms  
[335.870s][info][gc] GC(41) Concurrent Undo Cycle  
[335.870s][info][gc] GC(41) Concurrent Undo Cycle 0.107ms  
[452.493s][info][gc] GC(42) Pause Young (Normal) (G1 Evacuation Pause) 9M->5M(12M) 0.936ms  
[456.505s][info][gc] GC(43) Pause Young (Concurrent Start) (G1 Humongous Allocation) 5M->5M(12M) 0.794ms  
[456.505s][info][gc] GC(44) Concurrent Undo Cycle  
[456.505s][info][gc] GC(44) Concurrent Undo Cycle 0.115ms  
[575.535s][info][gc] GC(45) Pause Young (Normal) (G1 Evacuation Pause) 9M->5M(12M) 0.646ms



запуск программы

сортировка пузырьком

BubbleSort Начало сортировки - 23:08:06.176443

BubbleSort Конец сортировки - 23:09:47.878870800

сортировка вставкой

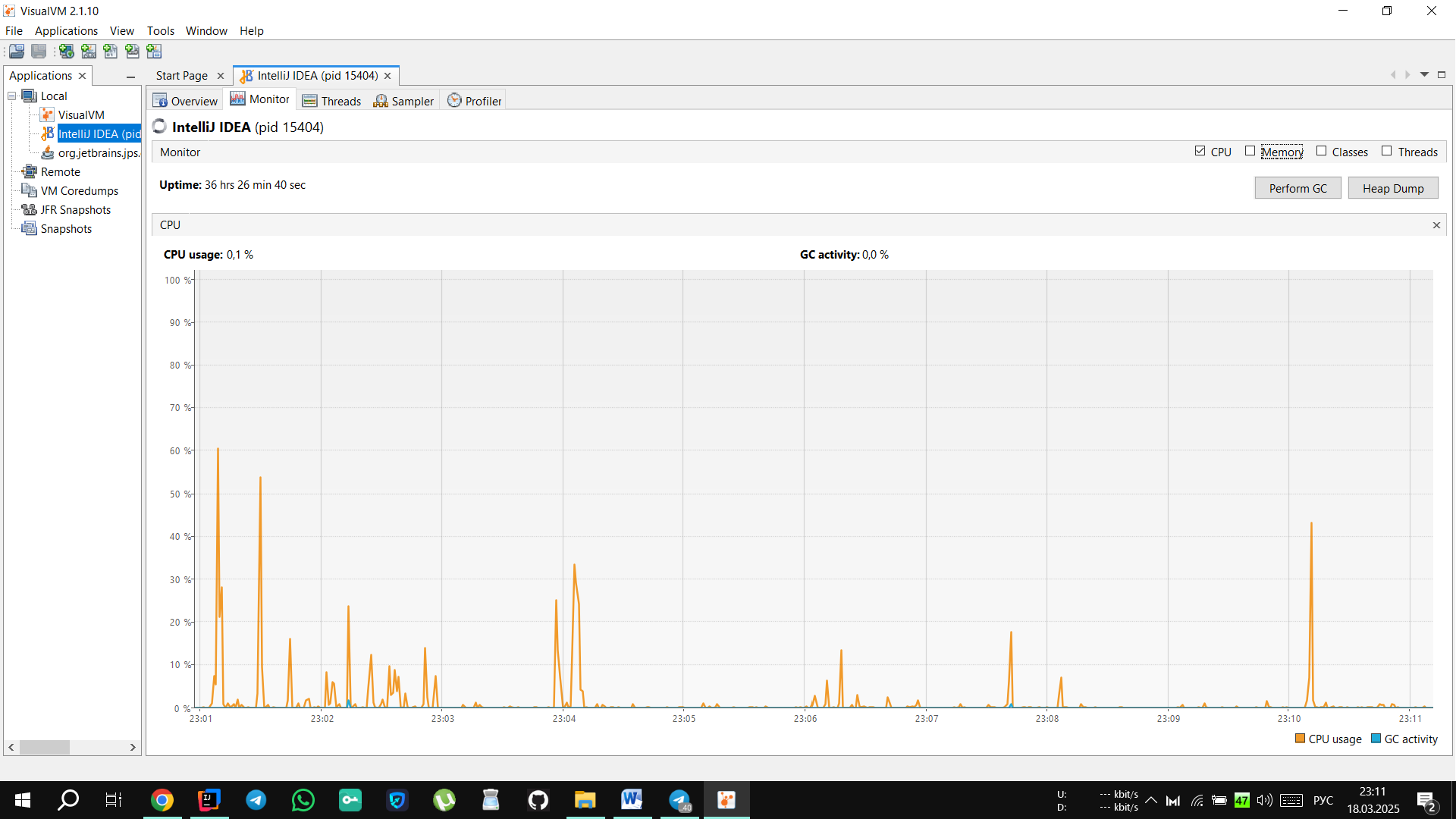
InsertSort Начало сортировки - 23:06:04.271492100

InsertSort Конец сортировки - 23:06:09.733049300

сортировка слиянием

MergeSort Начало сортировки - 23:04:08.829647700

MergeSort Конец сортировки - 23:04:08.943662900



сортировка пузырьком

BubbleSort Начало сортировки - 23:08:06.176443

BubbleSort Конец сортировки - 23:09:47.878870800

сортировка вставкой

InsertSort Начало сортировки - 23:06:04.271492100

InsertSort Конец сортировки - 23:06:09.733049300

сортировка слиянием

MergeSort Начало сортировки - 23:04:08.829647700

MergeSort Конец сортировки - 23:04:08.943662900

запуск программы

Судя по логу сборки "молодого поколения" происходят довольно часто, что указывает на высокую скорость создания объектов в приложении.

Время пауз сборки мусора в основном невелико (в пределах 0.5-6.5 мс), что свидетельствует о хорошей эффективности G1 в управлении памятью и минимизации задержек. Это важно для поддержки производительности приложения.

В большинстве случаев использование памяти снижается после сборок, что положительно сказывается на управлении памятью. Однако есть моменты, когда память остается на том же уровне, что может указывать на необходимость более глубокого анализа.

ZGC

13:09 запуск программы

13:11 создание массива на 250000 элементов

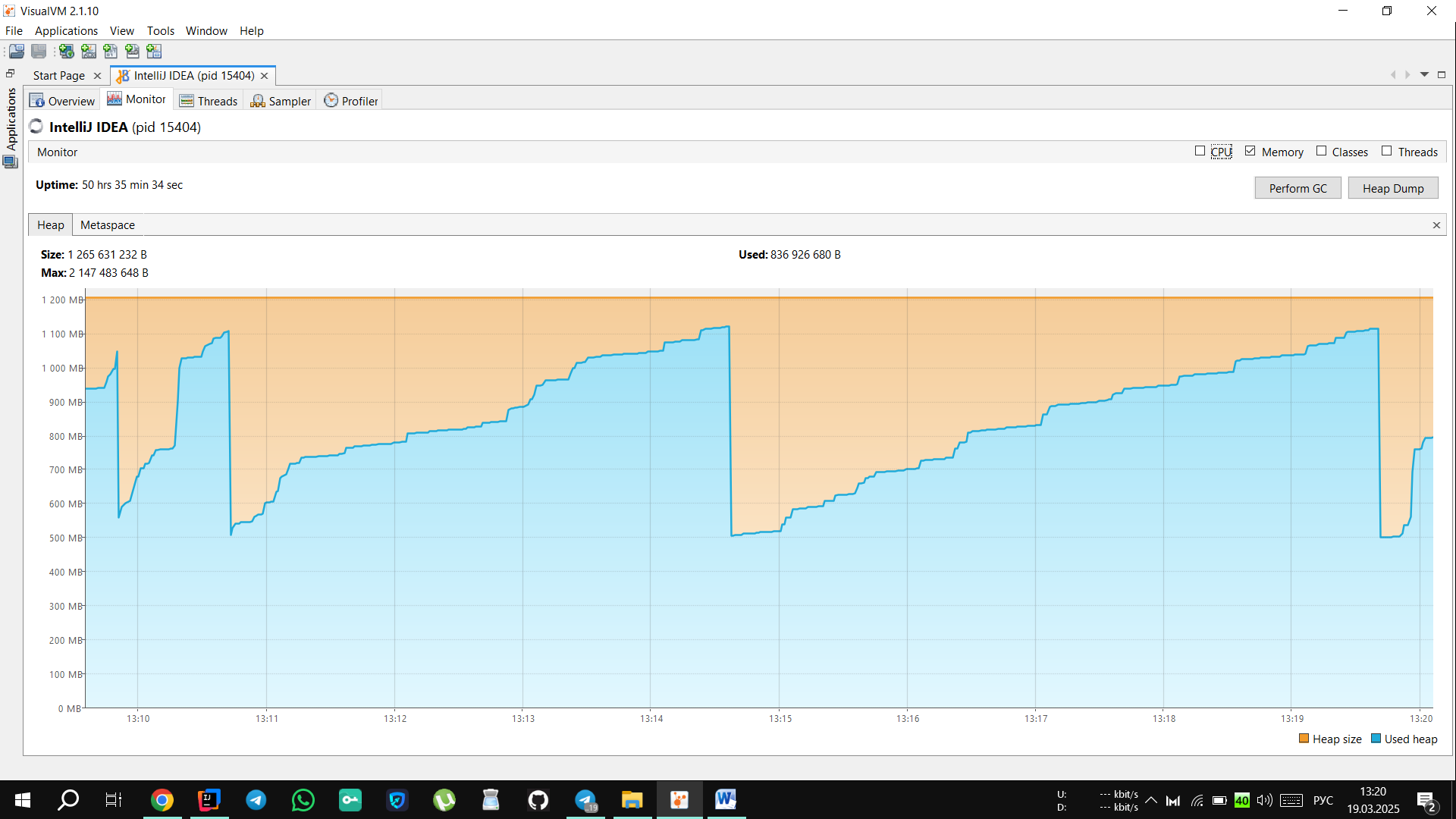
13:13 сортировка слиянием

13:15 сортировка вставкой

13:17 сортировка пузырьком

лог:

[0.106s][info][gc] Using The Z Garbage Collector  
[3.130s][info][gc] GC(0) Garbage Collection (Warmup) 6M(12%)->4M(8%)  
[186.526s][info][gc] GC(1) Garbage Collection (Warmup) 26M(52%)->16M(32%)  
[186.628s][info][gc] GC(2) Garbage Collection (Warmup) 26M(52%)->6M(12%)  
[426.511s][info][gc] GC(3) Garbage Collection (Proactive) 12M(24%)->10M(20%)



сортировка пузырьком

BubbleSort Начало сортировки - 13:17:02.705398800

BubbleSort Конец сортировки - 13:19:51.196267

сортировка вставкой

InsertSort Начало сортировки - 13:15:03.245602300

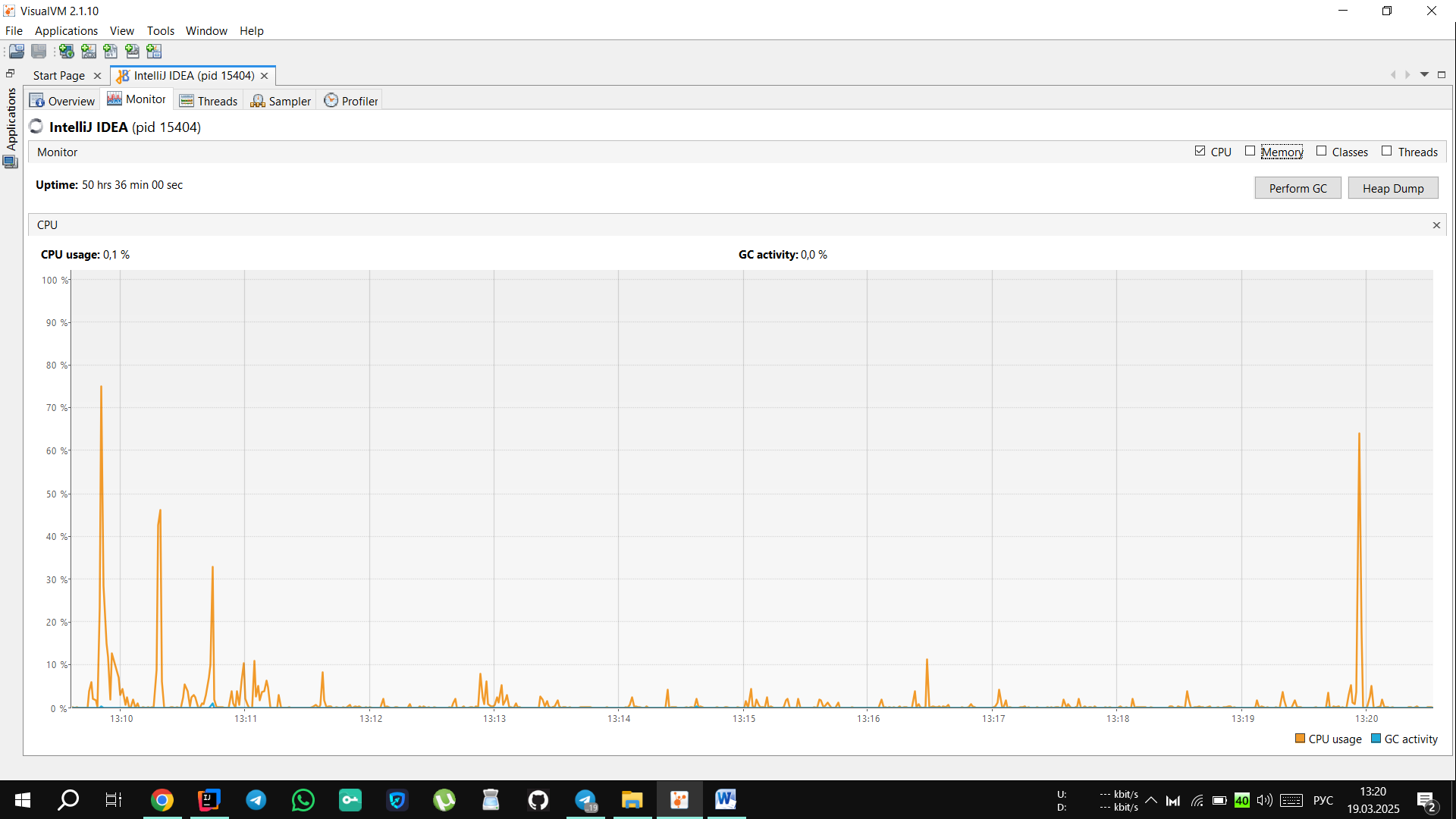
InsertSort Конец сортировки - 13:15:10.659806300

сортировка слиянием

MergeSort Начало сортировки - 13:13:02.718630800

MergeSort Конец сортировки - 13:13:02.832621200

запуск программы



сортировка пузырьком

BubbleSort Начало сортировки - 13:17:02.705398800

BubbleSort Конец сортировки - 13:19:51.196267

сортировка вставкой

InsertSort Начало сортировки - 13:15:03.245602300

InsertSort Конец сортировки - 13:15:10.659806300

сортировка слиянием

MergeSort Начало сортировки - 13:13:02.718630800

MergeSort Конец сортировки - 13:13:02.832621200

запуск программы

ZGC демонстрирует хорошую эффективность в управлении памятью, так как большинство сборок приводят к значительному освобождению памяти. Это особенно заметно в сборках GC(1) и GC(2), где освобождается значительное количество памяти за короткий промежуток времени.

Сборки происходят с разным интервалом. Первые две сборки (GC(0) и GC(1)) происходят довольно быстро, что может указывать на активное создание объектов в начале работы приложения. В то время как GC(3) происходит значительно позже и с меньшим освобождением памяти, что может указывать на стабилизацию работы приложения.

Наблюдается колебание в использовании памяти, что может быть нормальным для приложений с динамическим выделением памяти.