## Summary

The total number of logs output per test is 1e+6. (1000000 logs with no rendering and output )

Here are the LinkedListMemAppender VS ArrayListMemAppender Performance result summary table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Appender | List | Layout | MaxSize | Time Before Maxsize  (ms) | Time(ms)  After  Maxsize  (ms) | Total  Time  (ms) | Peek  Memory Usage  (MB) | Final Memoty Usage  (MB) |
| MemAppender | ArrayList | Pattern | 1e+5 | 28 | 9002 | 9030 | 82.2 | 82.2 |
| MemAppender | ArrayList | Pattern | 2e+5 | 53 | 16235 | 16288 | 102.3 | 73.2 |
| MemAppender | ArrayList | Pattern | 5e+5 | 73 | 25032 | 25105 | 139.8 | 83 |
| MemAppender | LinkedList | Pattern | 1e+5 | 29 | 192 | 221 | 54 | 54 |
| MemAppender | LinkedList | Pattern | 2e+5 | 77 | 186 | 263 | 101.5 | 101.5 |
| MemAppender | LinkedList | Pattern | 5e+5 | 96 | 417 | 513 | 107.5 | 107.5 |

More Details and the screenshorts are inSection ***JConsole Screen Shorts*.**

From this table we can find that,

1. Before the number of logs reaches the maxSize of MemAppender, the ArrayListMemAppender and LinkedListMemAppender has a similar time consumption,.But after maxSize, the MemAppender with LinkedList has a better performance than with ArrayList.

The reason is that, the appender needs to invoke the remove() method of the List to discard the first log element. The remove() method of ArrayList will cause element shift whose time complexity is about O(N), while LinkedList has only 1 element operation whose time complexity is O(1).

1. The LinkedListMemAppender has a better space performance than ArrayListMemAppender.

The reason is that the ArrayList is expanded by 1.5 times the length of the original array each time. The grow of ArrayList algorithm may like this:

***int oldCapacity = elementData.length;***

***int newCapacity = oldCapacity + (oldCapacity >> 1); // newCapcity= oldCapacity+ oldCapacity/2***

So the space growth of ArrayList is not linear. The LinkedList LinkedList adds only one element at the tail of the list, so its space growth is linear.

Here are the Layout Performance with MemAppender, FileAppender and ConsoleAppender result summary table: (output 1e+6 logs )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Appender | List | Layout | maxSize | Total time  (ms) | Peak memory  (MB) | Final memory  (MB) |
| MemAppender | ArrayList | Pattern | 1e+6 | 6778 | 461.6 | 396.2 |
| MemAppender | ArrayList | Velocity | 1e+6 | 18870 | 1177.7 | 1177.7 |
| MemAppender | LinkedList | Pattern | 1e+6 | 7789 | 487.1 | 413.1 |
| MemAppender | LinkedList | Velocity | 1e+6 | 18156 | 1214.0 | 1214.0 |
| FileAppender | N/A | Pattern | N/A | 2228 | 57.5 | 35.9 |
| FileAppender | N/A | Velocity | N/A | 13567 | 191.8 | 9.7 |
| ConsoleAppender | N/A | Pattern | N/A | 3437 | 59. 4 | 55.4 |
| ConsoleAppender | N/A | Velocity | N/A | 15524 | 66.4 | 60.5 |

More Details and the screenshorts are inSection ***JConsole Screen Shorts*.**

From this table we can find that,

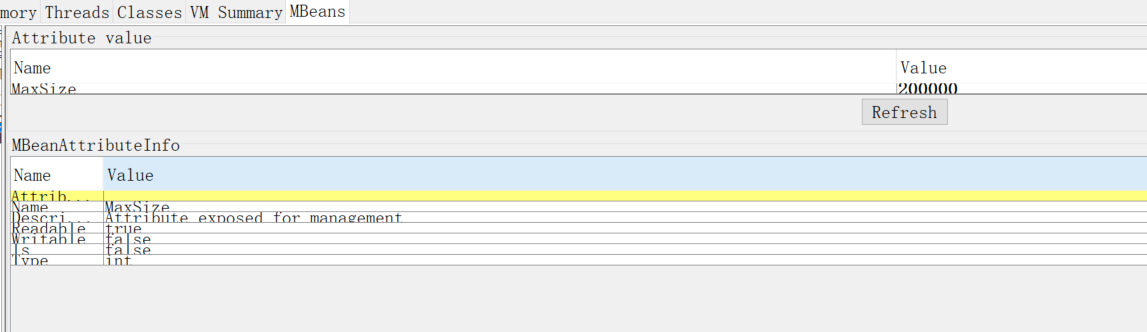
1. Time performance of VelocityLayout is worse than that of PatternLayout
2. Space performance of VelocityLayout is worse than that of PatternLayout

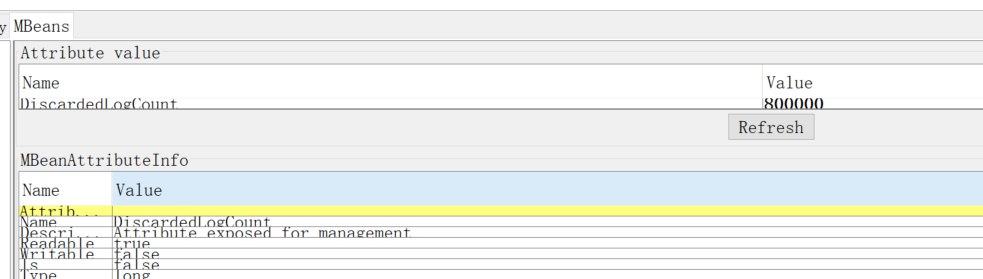
Note that, FileAppenderLogger and ConsoleAppenderLogger will instantly output and display once a log is received. But our MemAppender needs to invoke the printLog() methods to format all of the loggingEvent Objects and then print them. So it will consume more time and space to output logs.

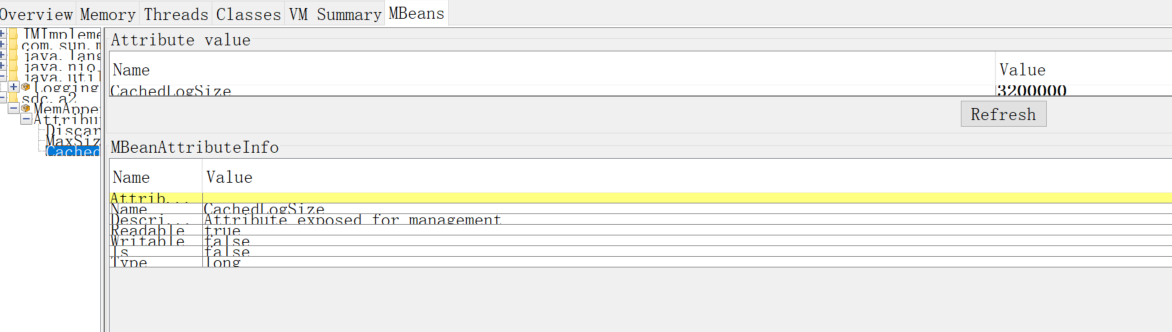
## MemAppenderMBean for JMX

I have added a MBean interface into the MemAppender Class so that We can monitor each MemAppender instace in JConsole:

The Monitored Properties are the maxSize, DiscaredLogCount, and CachedLogSize(characters)







## JConsole Screen Shorts.

-------testAppenderPerformance1XWithArrayListMemAppenderPatternLayout() -------

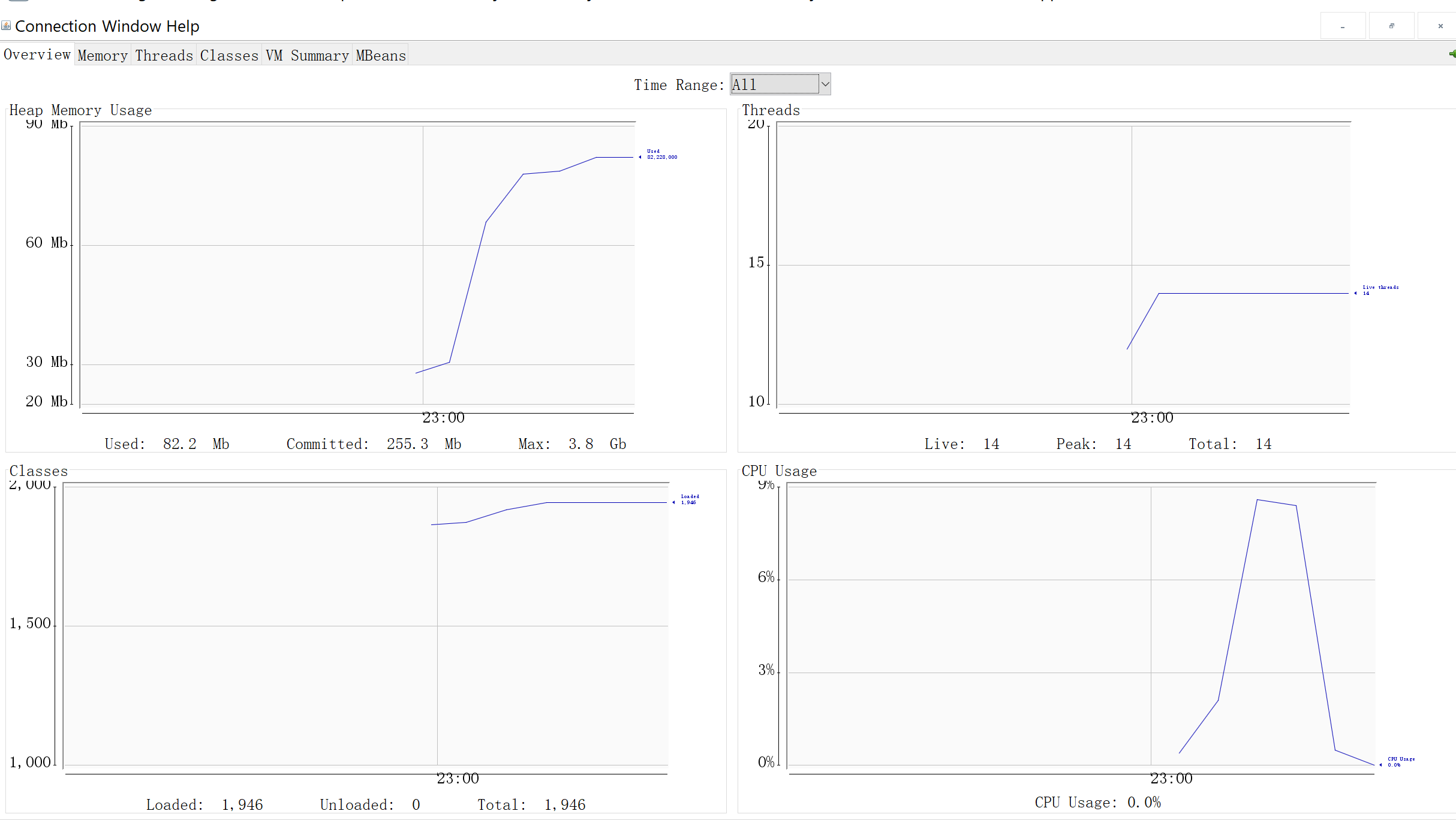
Insert 1000000 Before reach the MaxSize 100000 Time Consuming: 28

Insert 1000000 After reach the MaxSize 100000 Time Consuming: 9002

Insert 1000000 TotalTime: 9030

Peek: 82.2 Mb

Final: 82.2 Mb



-------testAppenderPerformance2XWithArrayListMemAppenderPatternLayout() -------

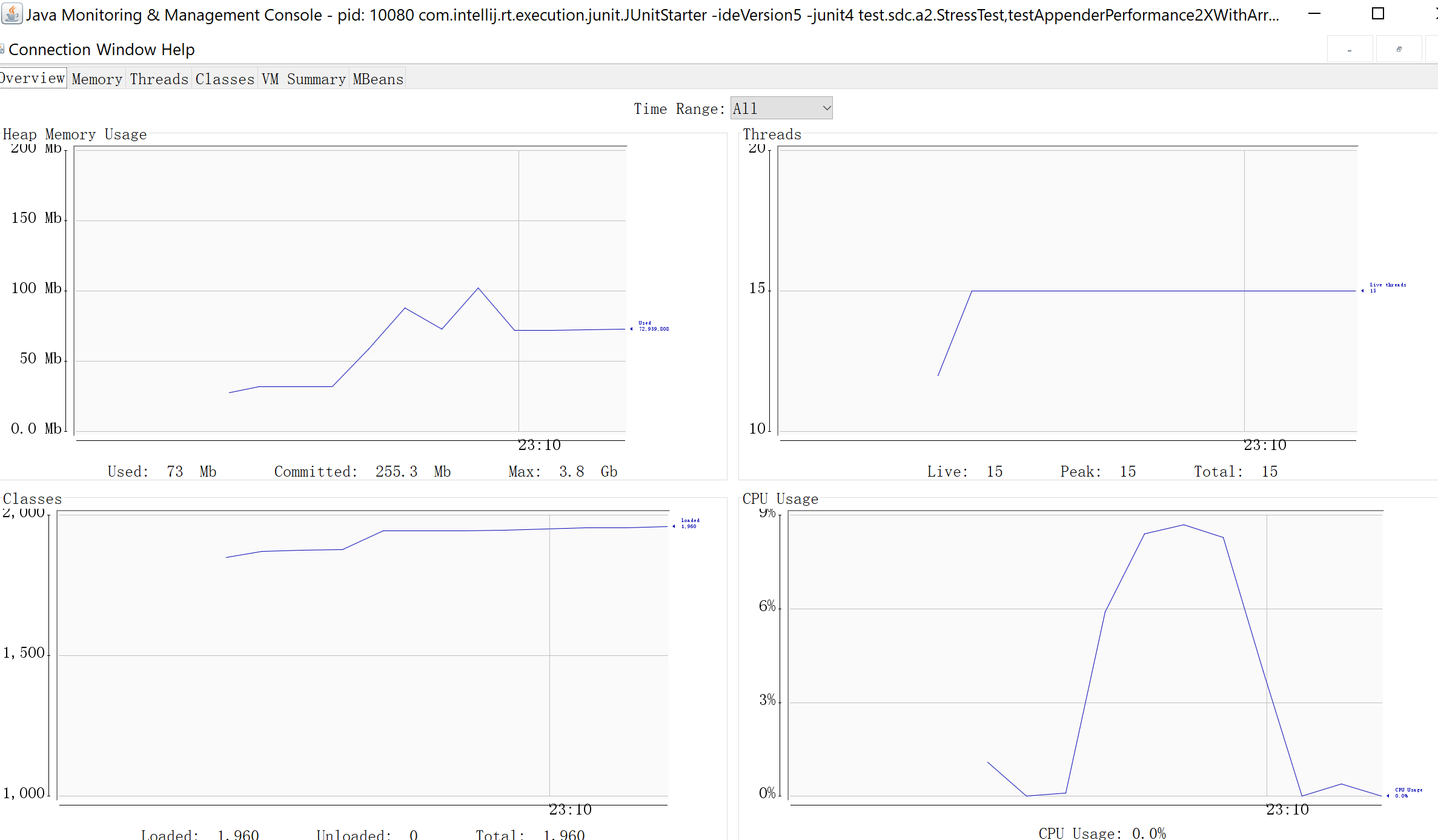
Insert 1000000 Before reach the MaxSize 200000 Time Consuming: 53

Insert 1000000 After reach the MaxSize 200000 Time Consuming: 16235

Insert 1000000 TotalTime: 16288

Peek:102.3MB

Final:73.2



-------testAppenderPerformance5XWithArrayListMemAppenderPatternLayout() -------

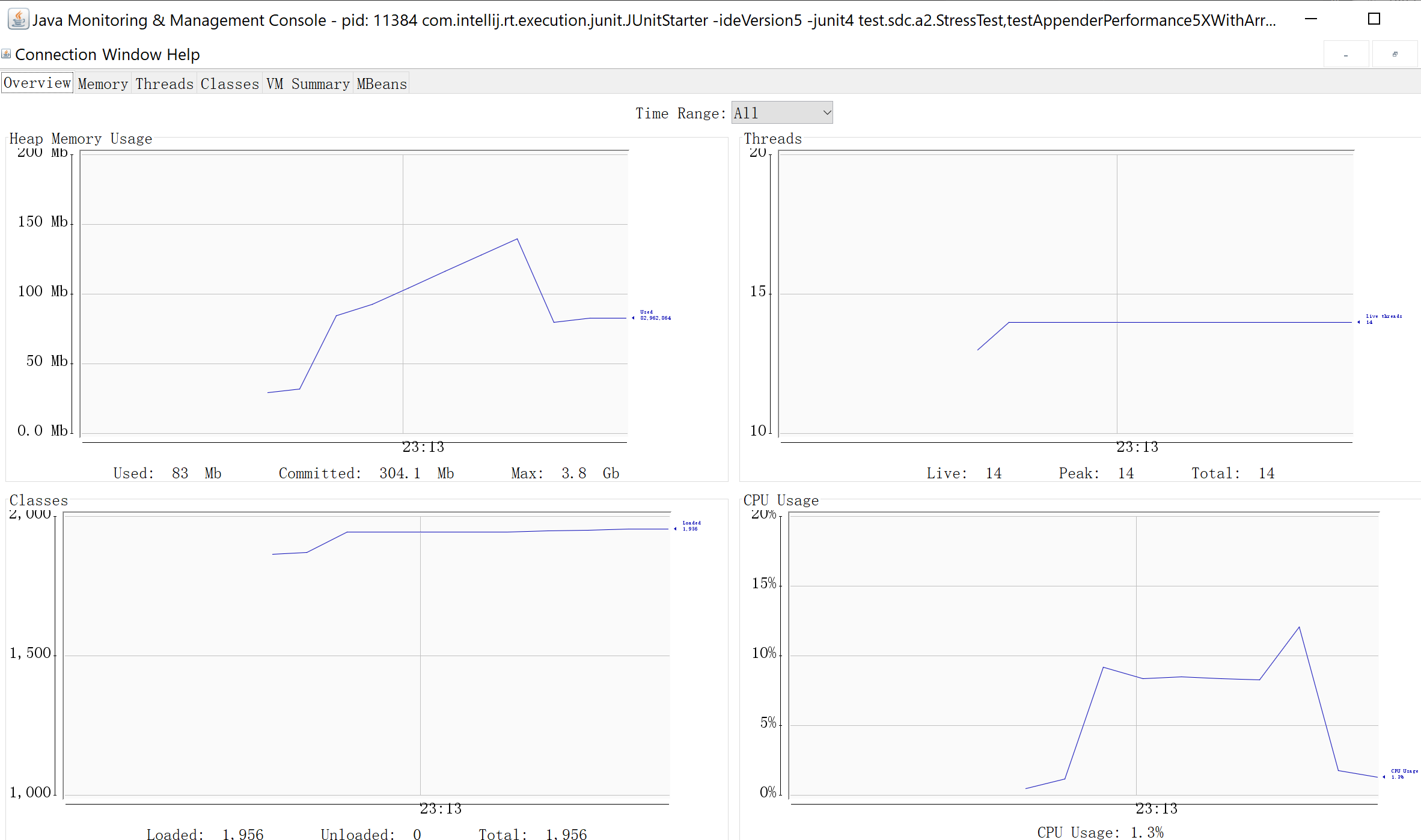
Insert 1000000 Before reach the MaxSize 500000 Time Consuming: 73

Insert 1000000 After reach the MaxSize 500000 Time Consuming: 25032

Insert 1000000 TotalTime: 25105

Peek: 139.8MB

Final:83MB



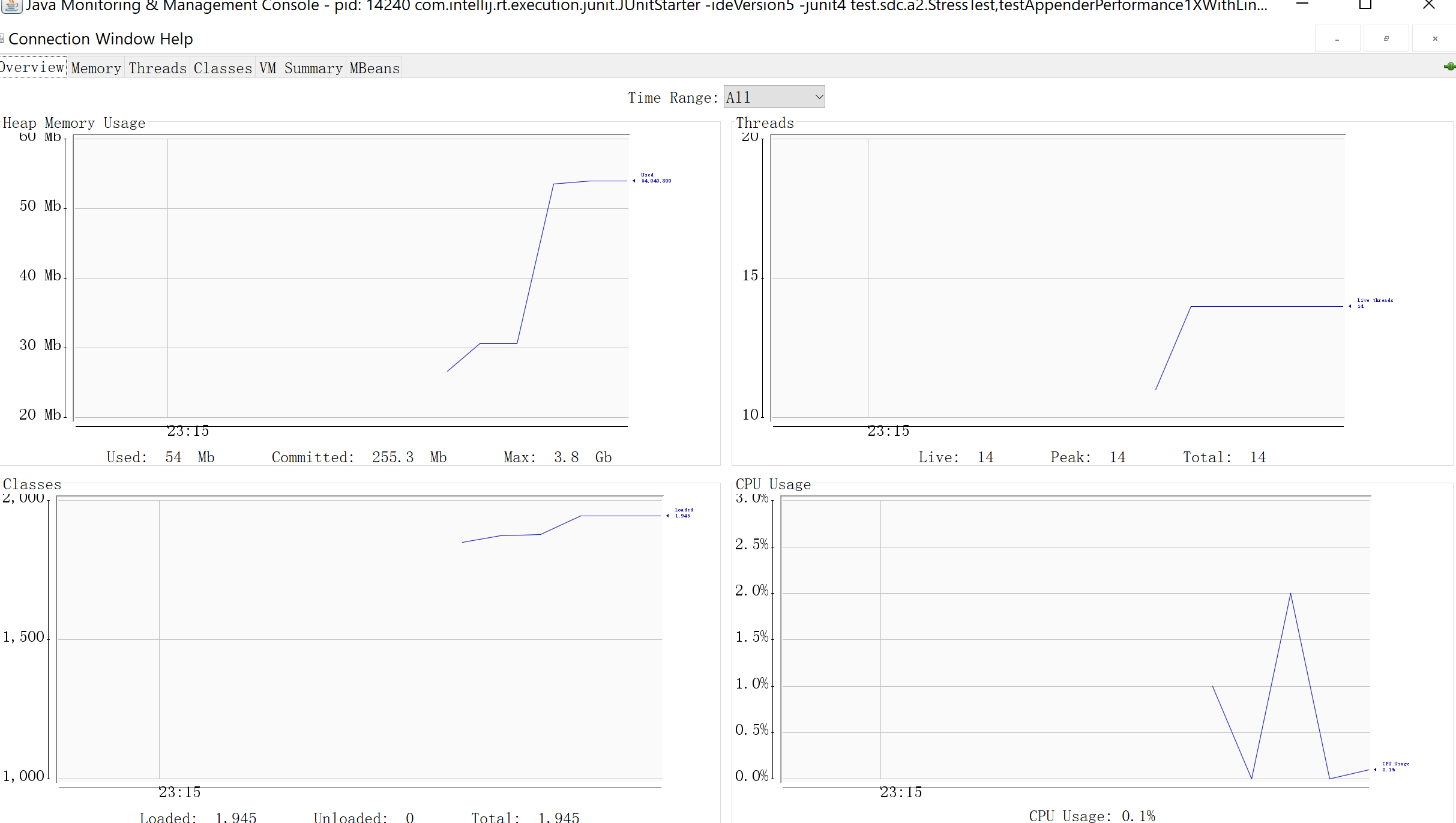
-------testAppenderPerformance1XWithLinkedListMemAppenderPatternLayout() -------

Insert 1000000 Before reach the MaxSize 100000 Time Consuming: 29

Insert 1000000 After reach the MaxSize 100000 Time Consuming: 192

Insert 1000000 TotalTime: 221

Peak &Final :54MB



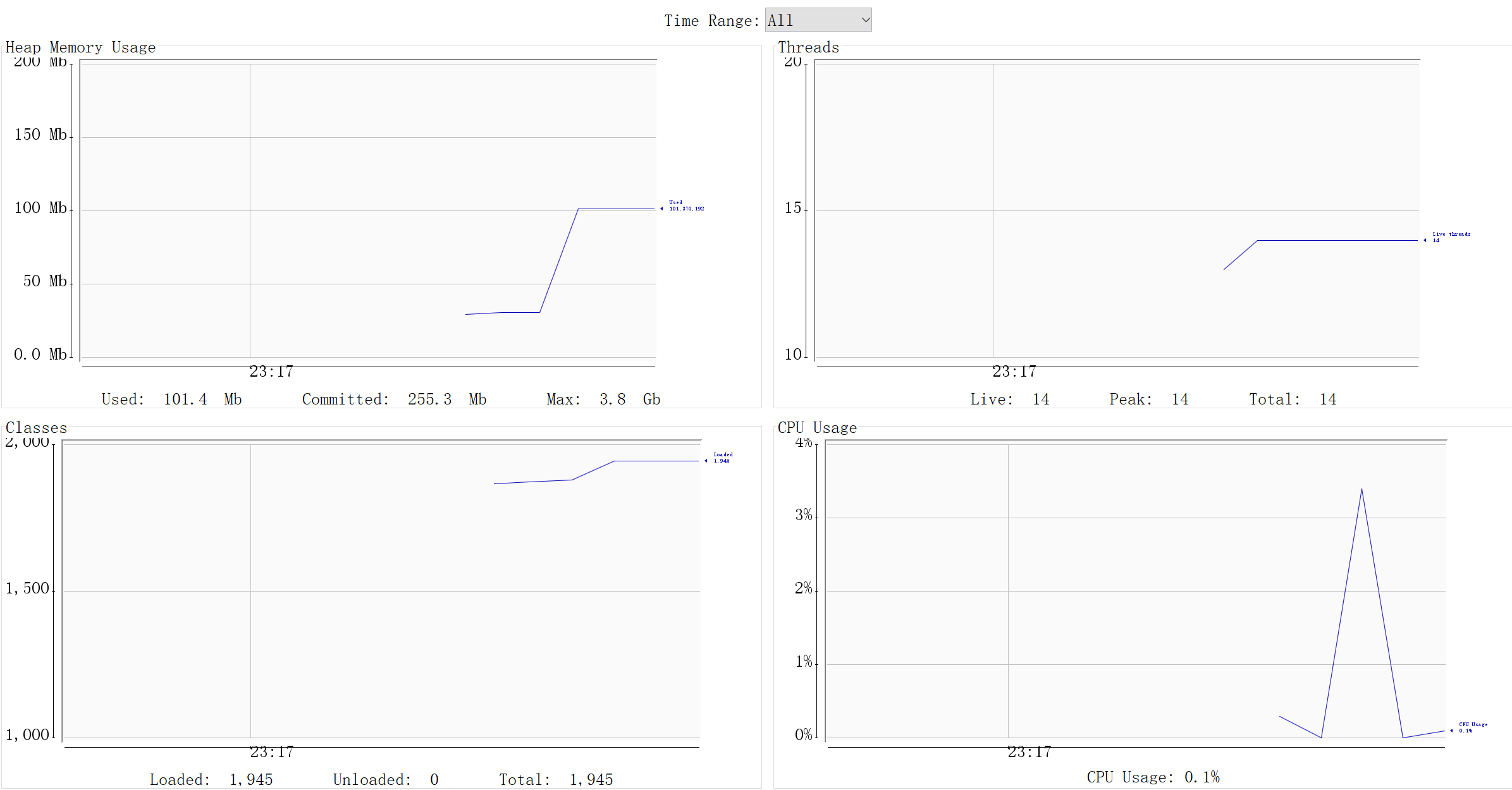
-------testAppenderPerformance2XWithLinkedListMemAppenderPatternLayout() -------

Insert 1000000 Before reach the MaxSize 200000 Time Consuming: 77

Insert 1000000 After reach the MaxSize 200000 Time Consuming: 186

Insert 1000000 TotalTime: 263

Peak & Final :101.5MB



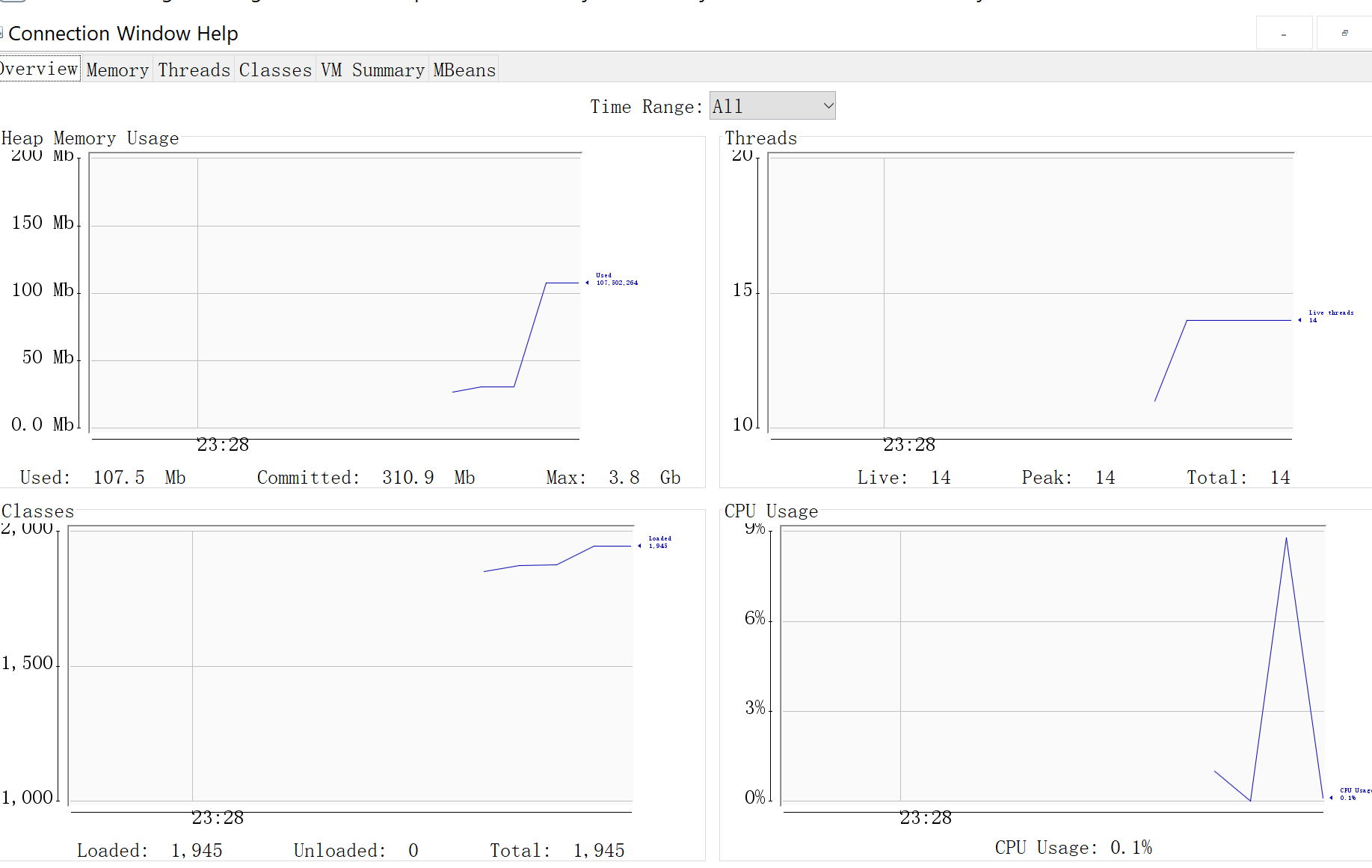
-------testAppenderPerformance5XWithLinkedListMemAppenderPatternLayout() -------

Insert 1000000 Before reach the MaxSize 500000 Time Consuming: 96

Insert 1000000 After reach the MaxSize 500000 Time Consuming: 417

Insert 1000000 TotalTime: 513

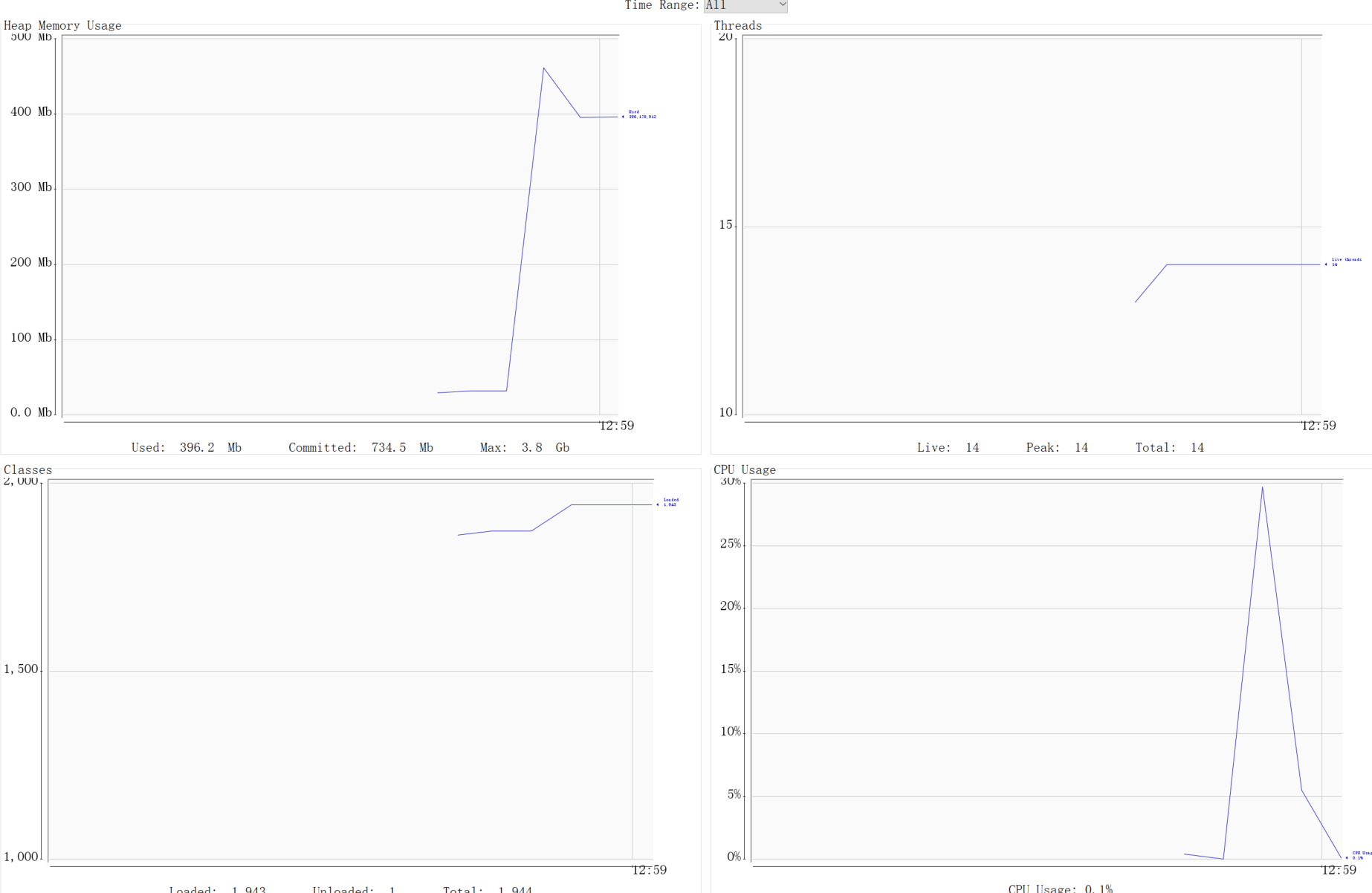
Peak & Final :107.5MB



-------testLayoutPerformanceWithArrayListMemAppenderPatternLayout() -------"

output 1000000 logs TotalTime: 6778

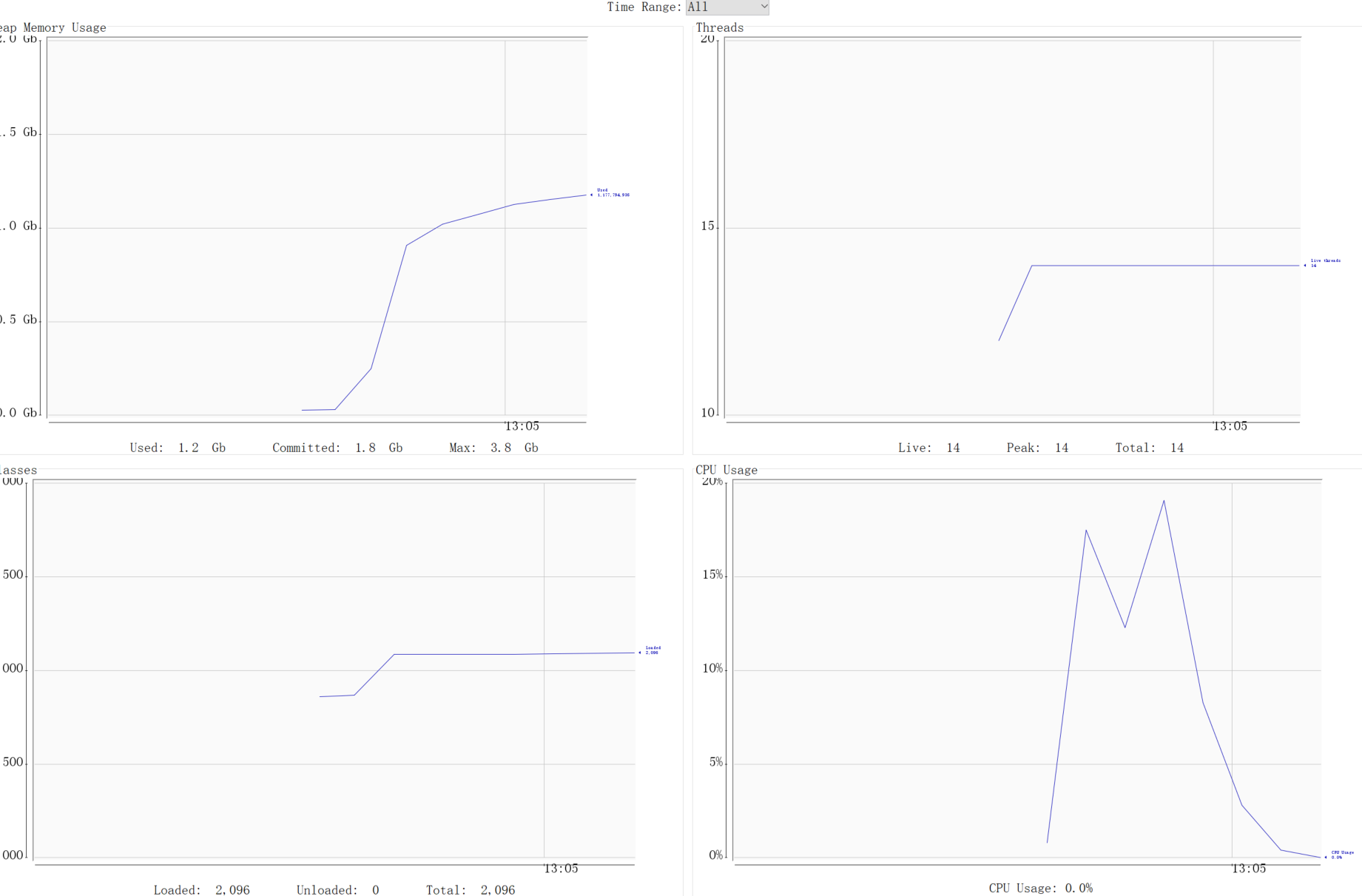
peak 461.6M final 396.2 MB



-------testLayoutPerformanceWithArrayListMemAppenderVelocityLayout() -------

output 1000000 logs TotalTime: 18870

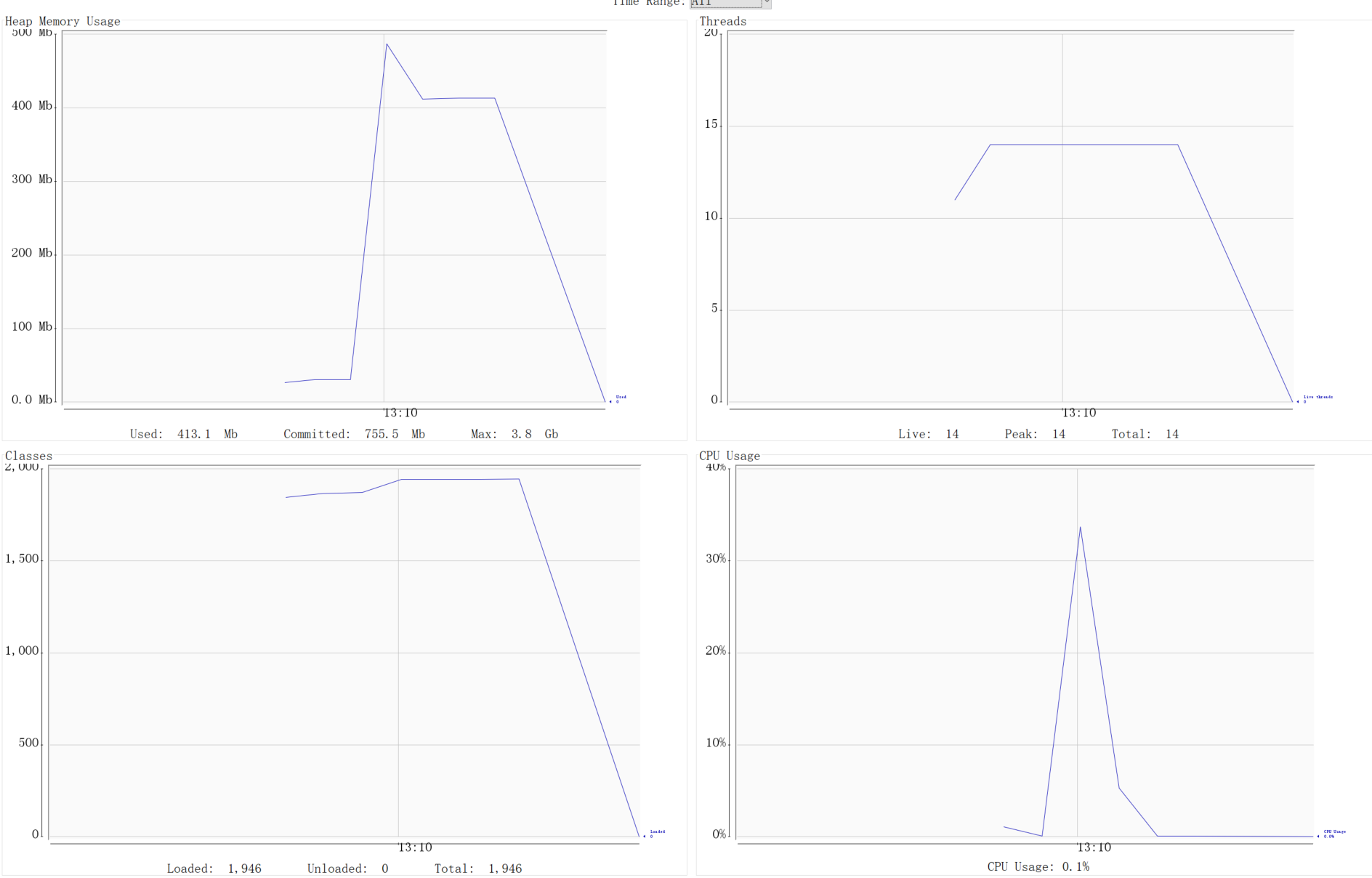
peak final 1177.7MB



-------testLayoutPerformanceWithLinkedListMemAppenderPatternLayout ()-------

output 1000000 logs TotalTime: 7789

peak: 487.1 final 413.1 MB



"-------testLayoutPerformanceWithLinkedListMemAppenderVelocityLayout() -------"

output 1000000 logs TotalTime: 18156

peak final 1214.0 MB



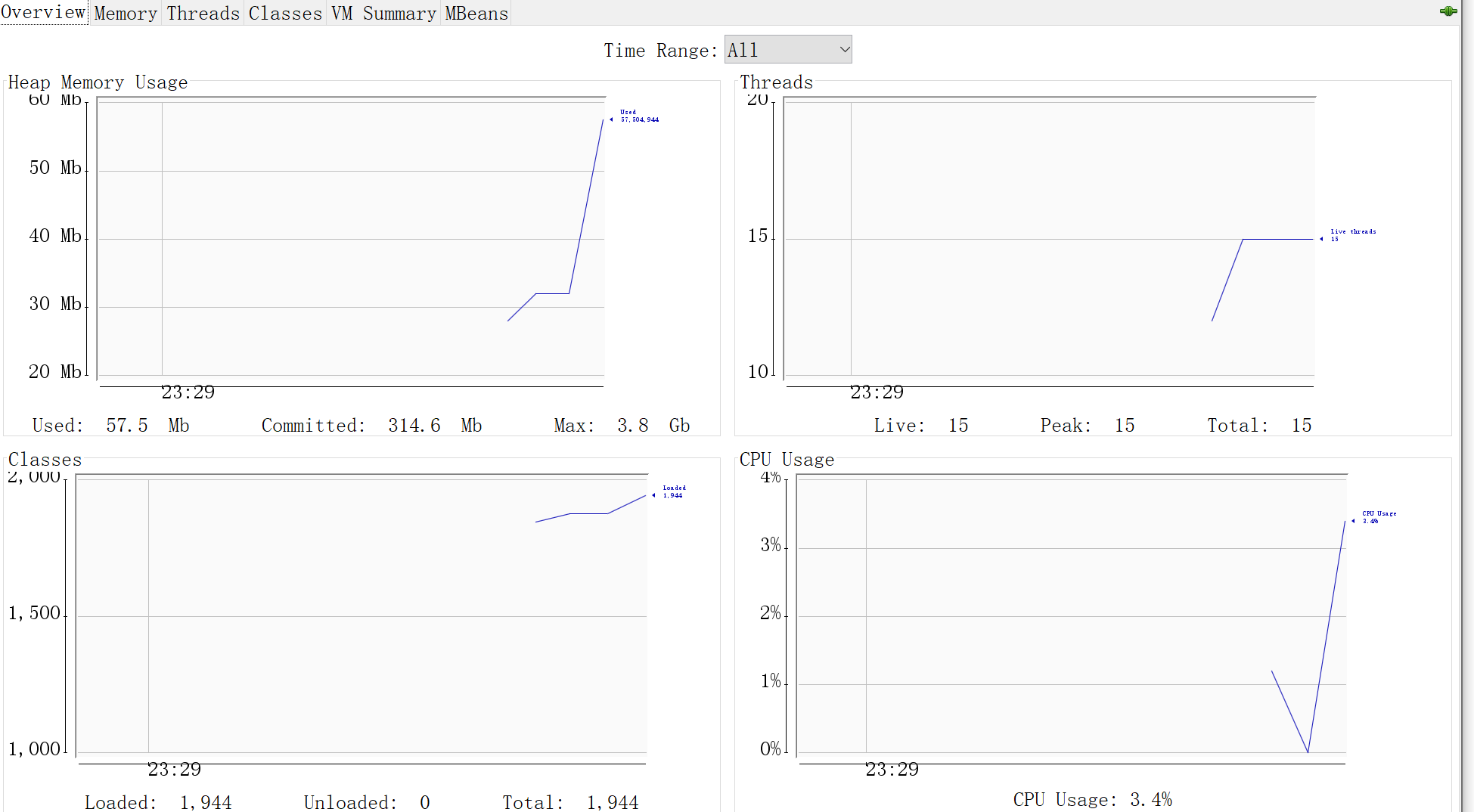
-------testLayoutPerformanceWithFileAppenderPatternLayout()() -------

Insert 1000000 Before reach the MaxSize 1000000 Time Consuming: 2228

Insert 1000000 After reach the MaxSize 1000000 Time Consuming: 0

Insert 1000000 TotalTime: 2228

Peak:57.5MB final 35.9 MB



-------testLayoutPerformanceWithFileAppenderVelocityLayout() -------

output 1000000 logs TotalTime: 13567

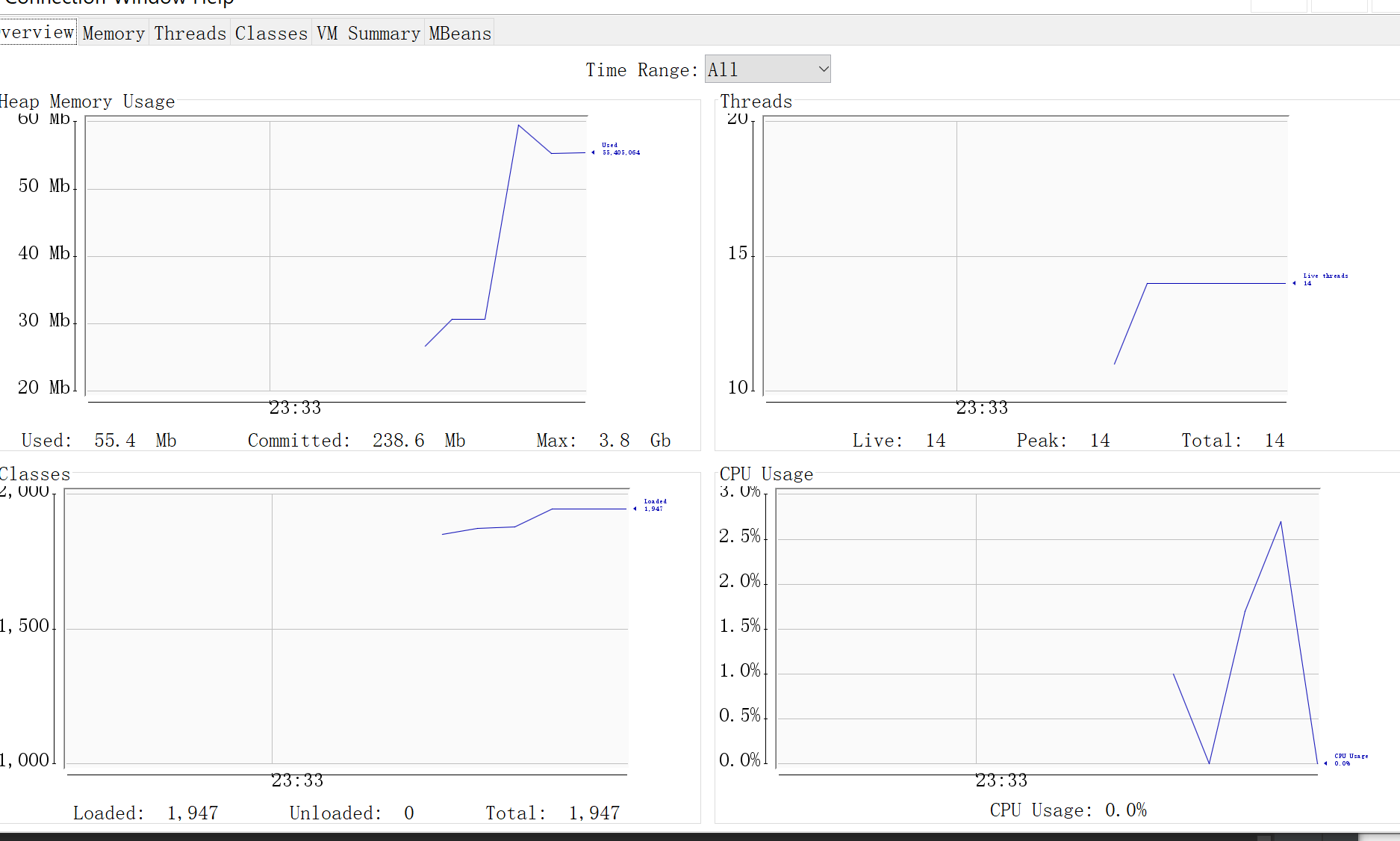
Peak: 191.8 MB Final 9.7 MB



------------testLayoutPerformanceWithConsoleAppenderPatternLayout()----------

output 1000000 logs TotalTime: 3437

Peak 59. 4 MB Final 55.4 MB



-------testLayoutPerformanceWithConsoleAppenderVelocityLayout() -------

output 1000000 logs TotalTime: 15524

Peak 66.4 MB Final 60.5 MB

