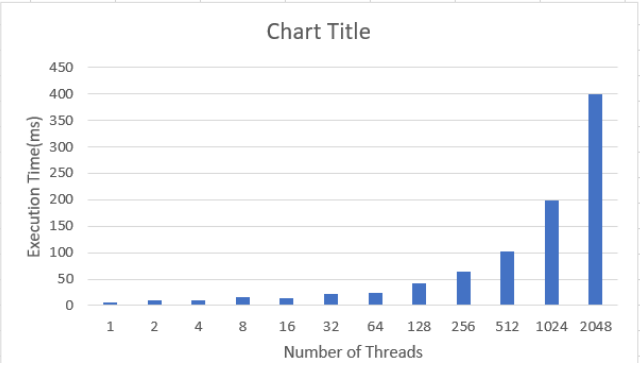
Multi Thread Mean Graph

Number of Threads	1	2	4	8	16	32	64	128	256	512	1024	2048
Execution Time(ms)	6.381	9.599	10.923	16.184	14.789	22.896	24.761	42.561	63.619	103.171	199.453	399.538



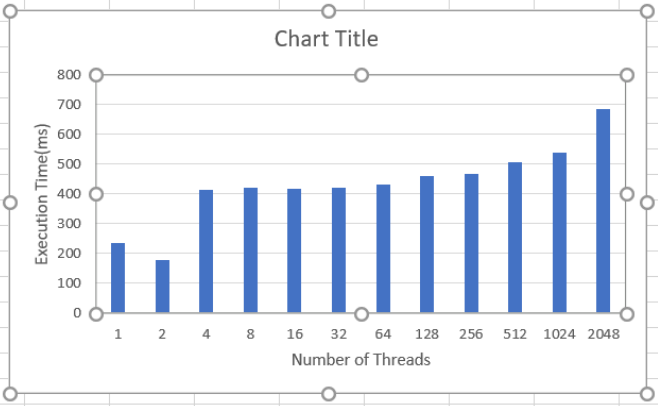
Proof of Code working:

```
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 1
Number of processors: 8
Temporal mean value of thread n is ...
5002.019383430481
Time elapsed: 6.96ms
The global mean value is ... 5002.019383430481
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 2
Number of processors: 8
Temporal mean value of thread n is ...
5000.948331832886 5003.090435028076
Time elapsed: 8.884ms
The global mean value is ... 5002.019383430481
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 4
Number of processors: 8
Temporal mean value of thread n is ...
4996.885913848877 5005.0107498168945 5000.335964202881 5005.8449058532715
Time elapsed: 11.224ms
The global mean value is ... 5002.019383430481
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 8
Number of processors: 8
Temporal mean value of thread n is ...
4995.028823852539 4998.743003845215 5015.597785949707 4994.423713684082 5003.888244628906 4996.7836837768555 5003.21764
3737793 5008.47216796875
Time elapsed: 13.826ms
The global mean value is ... 5002.019383430481
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 16
Number of processors: 8
Temporal mean value of thread n is ...
4984.044342041016 5006.0133056640625 4996.079345703125 5001.406661987305 5003.930252075195 5027.265319824219 4986.04638
671875 5002.801040649414 5020.112518310547 4987.663970947266 5000.725036621094 4992.842330932617 5003.167785644531 5003
.267501831055 5010.052230834961 5006.892105102539
Time elapsed: 17.545ms
The global mean value is ... 5002.019383430481
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MeanThread input.txt 32
Number of processors: 8
Temporal mean value of thread n is ...
5003.316009521484 4964.772674560547 4988.2056884765625 5023.8209228515625 4988.783782958984 5003.374908447266 4991.0971
98486328 5011.716125488281 5004.753814697266 5003.106689453125 5037.984466552734 5016.546173095703 4983.513488769531 49
88.579284667969 4994.291198730469 5011.310882568359 5028.301239013672 5011.923797607422 4989.389678955078 4985.93826293
9453 5007.778228759766 4993.671844482422 4988.2335205078125 4997.451141357422 5001.25341796875 5005.0821533203125 5004.
036285400391 5002.498718261719 5006.61083984375 5013.493621826172 5007.197021484375 5006.587188720703
Time elapsed: 14.366ms
The global mean value is ... 5002.019383430481
```

For Mean, the operations are not resource intensive (O(n) complexity) enough to warrant using threading. As a result, the overhead of creating threads outweigh the gain from parallel computation. Moreover, when number of thread >8, the process is pseudo-parallel as the thread number exceeds the number of cores, hence the execution time continues to increase due to time cost of thread creation.

Multi Thread Median Graph

Number of Threads	1	2	4	8	16	32	64	128	256	512	1024	2048
Execution Time(ms)	234	180	414	422	416	420	431	461	469	506	538	685




```
Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 4
The Median value is: 5002
Running time is 414 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 16
The Median value is: 5002
Running time is 416 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 32
The Median value is: 5002
Running time is 420 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 64
The Median value is: 5002
Running time is 431 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 128
The Median value is: 5002
Running time is 461 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 256
The Median value is: 5002
Running time is 469 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 512
The Median value is: 5002
Running time is 506 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 1024
The Median value is: 5002
Running time is 538 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 2048
The Median value is: 5002
Running time is 685 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 1
The Median value is: 5002
Running time is 234 milliseconds

Vincentcent1@DESKTOP-SLF4L6L:/mnt/d/Dropbox/Computer Systems Engineering/Code/2_Lab$ java MedianThread input.txt 2
The Median value is: 5002
Running time is 180 milliseconds
```

For Median, since we need to sort the list first, the time complexity is O(nlogn). Moreover, by using mergeSort, we are making a full use of the parallel thread since mergeSort can work individually before merging the sorted arrays. This explains the reduction in execution time between one and two cores. The speed can be further increased if the merging of the subarrays from the thread are done in parallel as well. But since the merging is not done in parallell, from 2 thread to 4 thread,

there is an increase in execution time due to the additional overhead of sequential merging of the 4 subarrays. The gradual increases from 4 threads onwards can be explained by the thread creation overhead and the sequential merging. It is a curious case that using one thread is faster than 4 threads since the one thread also needed to do sequential merging within itself. But I think this can be explained by the thread creation overhead.

Considering the main bulk of the computation cost is in the merging of the arrays, it makes sense that as number of thread increases, more sequential merging needs to be done and hence, the increase in execution time. Also, thread creation overhead plays a part in the increase as well.