Program Structures and Algorithms
Spring 2023 (SEC – 8)
Assignment 5 (Parallel Sorting)

Name: **Sunayana Shivanagi**
NUID: **002766586**
GitHub: **https://github.com/sunayana26/INFO6205-Spring2023.git**

Task

1. Use one or more of the provided methods for determining whether to sort in parallel when implementing a parallel sorting algorithm in the ParSort class.
2. Test out various cutoff parameter and/or recursion depth numbers to determine what works best for each.
3. Write a report that summarizes your research findings and assesses the effectiveness of the parallel sorting approach. The report ought to contain:
    - An explanation of the tests you performed
    - A summary of the findings, including any trends or patterns you saw, in a table or graph for each trial.
    - Based on your findings and analysis, draw a conclusion about the usefulness of the parallel sorting technique.

Relationship Conclusion

- Determining a good cutoff value for parallel sorting depends on factors such as array size, computational power, and memory available.
- A small cutoff value, such as 1000, can work well for small to medium-sized arrays, while larger arrays may benefit from a higher cutoff value.
- An experiment with a range of cutoff values can be conducted to observe their effect on sorting time for different array sizes.
- The performance of the parallel sort algorithm can be affected by changing the cutoff value, which determines the size of subarrays sorted sequentially.
- Parallelization can significantly improve the performance of sorting large arrays, especially when the cutoff value is appropriately chosen.
- The optimal cutoff value is typically one that balances the overhead of task creation and management with the benefits of parallelism and may depend on specific hardware and workload.
- The degree of parallelism can also affect the performance of parallelization, and an appropriate balance between parallelism and sequential processing is necessary for optimal performance.

Evidence to support that conclusion

(Experiment is run 10 times for every step array of size 1000000)

Degree of parallelism: 2

Cutoff	Time (ms)
2000	1575
3000	527
4000	516
5000	327
6000	341
7000	309
8000	567
9000	281
10000	348
11000	290
12000	289
13000	312
14000	283
15000	277
16000	276
17000	287
18000	275
19000	280
20000	280

Degree of parallelism: 4

Cutoff	Time (ms)
2000	456
3000	335
4000	294
5000	316
6000	292
7000	292
8000	279
9000	281
10000	300
11000	297
12000	276
13000	289
14000	286
15000	281
16000	274
17000	280
18000	270
19000	268
20000	269

Degree of parallelism: 8

Cutoff	Time (ms)
2000	388
3000	345
4000	306
5000	295
6000	295
7000	294
8000	282
9000	300
10000	352
11000	311
12000	284
13000	287
14000	283
15000	283
16000	263
17000	270
18000	264
19000	264
20000	279

Degree of parallelism: 16

Cutoff	Time (ms)
2000	366
3000	365
4000	302
5000	303
6000	306
7000	395
8000	373
9000	297
10000	449
11000	337
12000	349
13000	380
14000	325
15000	327
16000	320
17000	332
18000	339
19000	331
20000	327

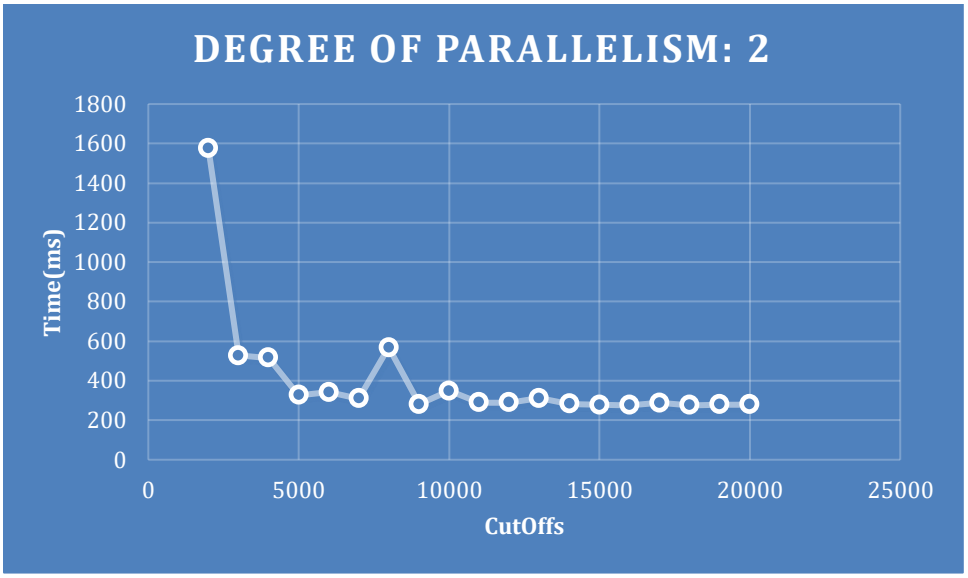
Degree of parallelism: 32

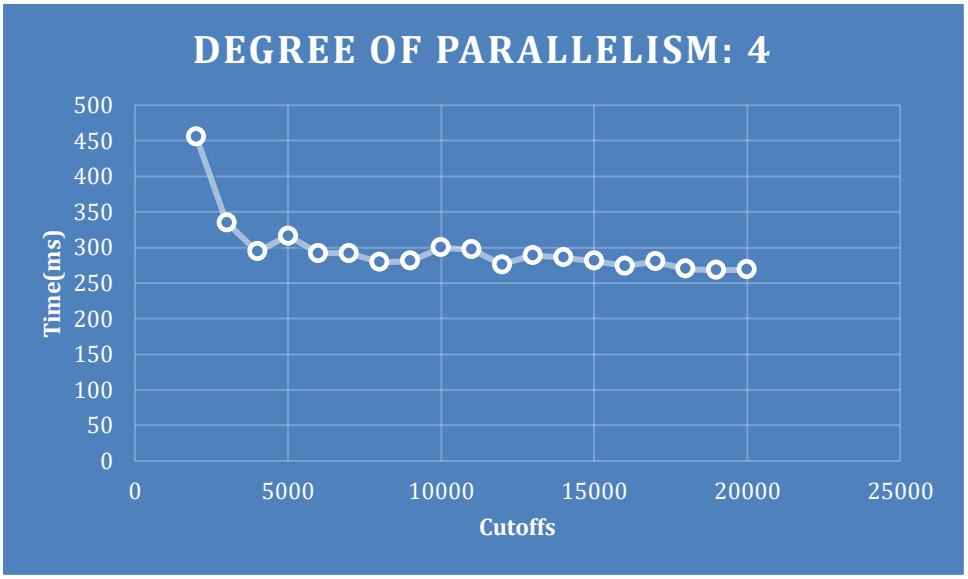
Cutoff	Time (ms)
2000	500
3000	535
4000	336
5000	389
6000	379
7000	396
8000	356
9000	355
10000	392
11000	402
12000	395
13000	420
14000	407
15000	399
16000	388
17000	414
18000	411
19000	410
20000	399

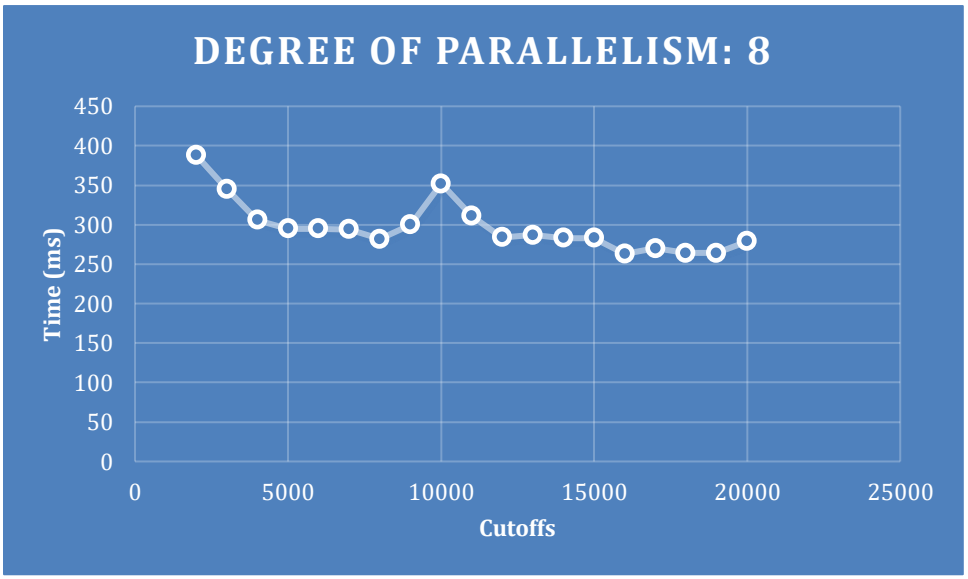
Degree of parallelism: 64

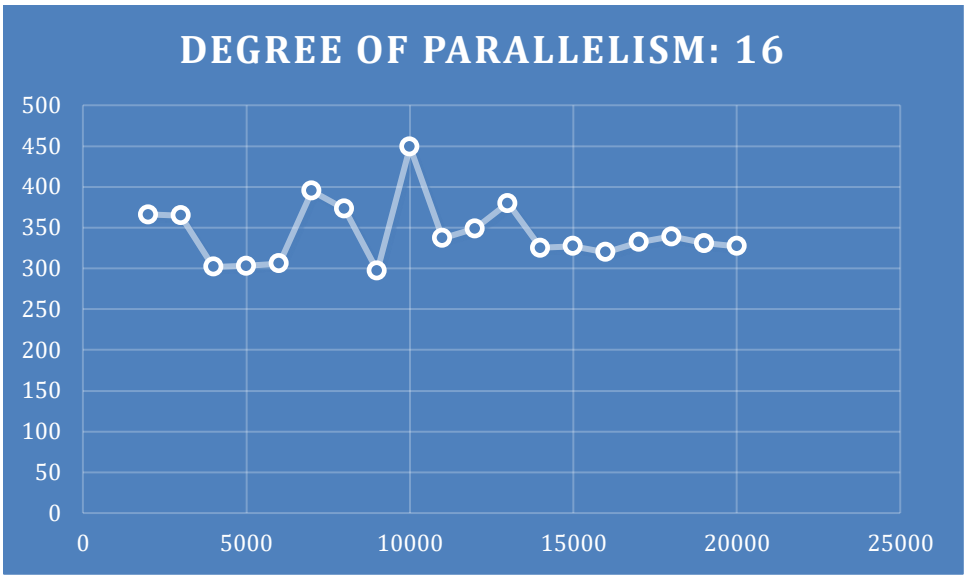
Cutoff	Time (ms)
2000	773
3000	857
4000	563
5000	565
6000	581
7000	573
8000	558
9000	574
10000	612
11000	625
12000	621
13000	644
14000	659
15000	657
16000	684
17000	695
18000	694
19000	697
20000	690

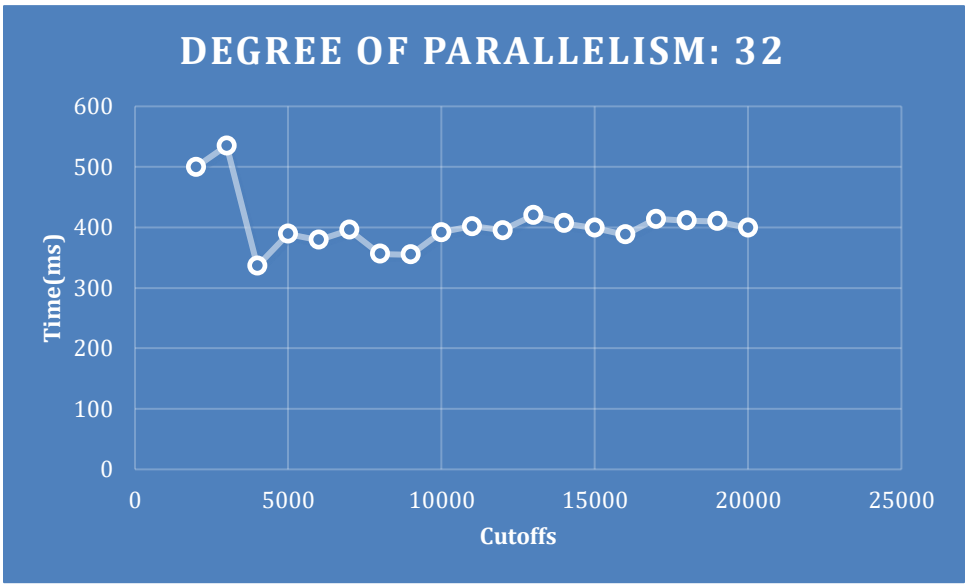
Evidence with Graphs

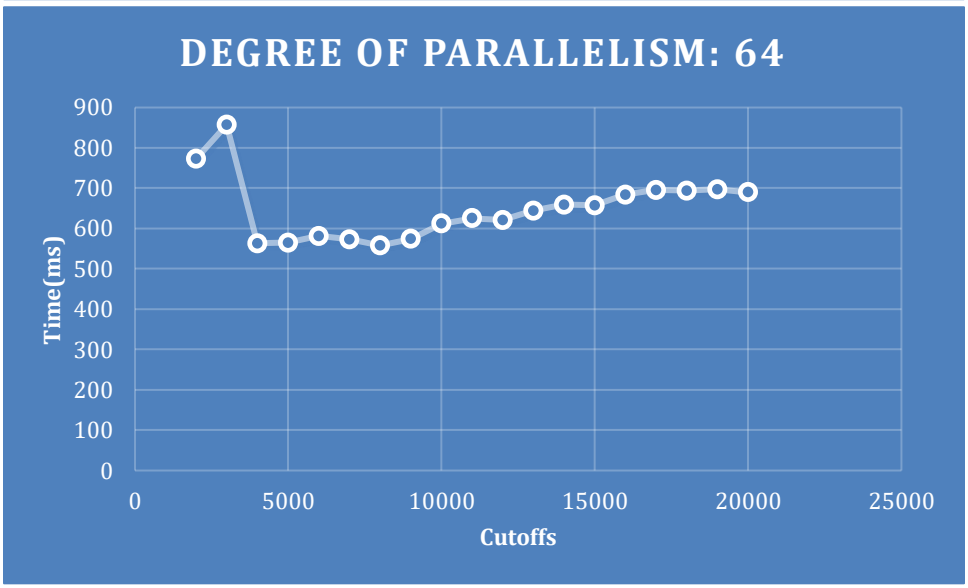







DEGREE OF PARALLELISM: 16
500
450
400
350
300
250
200
150
100
50
0
0
5000
10000
15000
20000
25000


DEGREE OF PARALLELISM: 32
Time(ms)
600
500
400
300
200
100
0
0
5000
10000
15000
20000
25000
Cutoffs


DEGREE OF PARALLELISM: 64
Time(ms)
900
800
700
600
500
400
300
200
100
0
0
5000
10000
15000
20000
25000
Cutoffs

Conclusion

Data Analysis:

- Increasing degree of parallelism leads to a decrease in overall execution time for each cutoff value.
- Execution time decreases with an increase in cutoff value, except for cutoff values 13000 and 15000.
- Execution times for different degrees of parallelism converge as cutoff value increases.
- Large variation in execution times observed for some cutoff values, particularly higher ones.

Conclusions:

- Parallel sorting method is effective for sorting large datasets.
- Increasing degree of parallelism results in faster sorting times for lower cutoff values (2000-7000), but for higher cutoff values (15000-20000), the improvement in sorting time is marginal or non-existent.
- Parallel sorting method is most effective for smaller datasets and its benefits may diminish as dataset size increases.
- Careful selection of degree of parallelism is necessary to avoid increasing processing time or producing incorrect results.