Program Structures and Algorithms

Spring 2024

NAME: Siddharth Dumbre
NUID: 002247119
GITHUB LINK: https://github.com/dumbresi/Info-6205-spring2024

Task: Parallel Sorting

For small arrays , the overhead of parallelism may outweigh the benefits. In such cases, it's better to use a sequential sorting algorithm. Therefore, a cutoff number of around 100-200 might be suitable for these cases.
For larger arrays, where the benefits of parallelism are more significant, a higher cutoff number can be chosen. This reduces the overhead of creating and managing parallel tasks. A cutoff number in the range of 500-1000 might be appropriate for arrays of moderate size.
It's also essential to consider the hardware characteristics of the system. If the system has a large number of CPU cores and good parallel processing capabilities, a higher cutoff number can be chosen to take advantage of parallelism effectively.
Empirical testing with different cutoff values on representative input sizes can help determine the optimal cutoff number for a specific application and environment.

Keeping Array length constant

Number of threads	Array Length	Total Time (ms)
2	2M	89606
4	2M	72071
6	2M	48911
8	2M	48919
12	2M	50796
16	2M	48521

Varying the array length

Number of threads	Array Length	Total Time (ms)
4	2M	72071
4	1M	32490
4	0.5M	23774
4	0.25M	11567

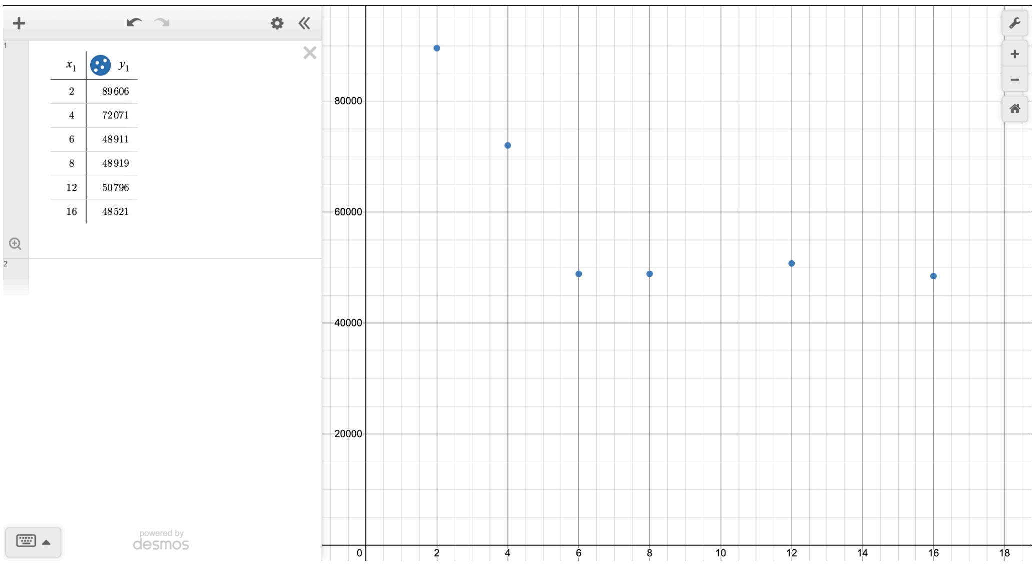
Array Length vs Sorting time when number of threads=8

Cutoff	Time
210000	1906
220000	1222
230000	1255
240000	1226
250000	1237
260000	867
270000	818
280000	820
290000	842
300000	841
310000	1001
320000	1448
330000	862
340000	833
350000	864
360000	825

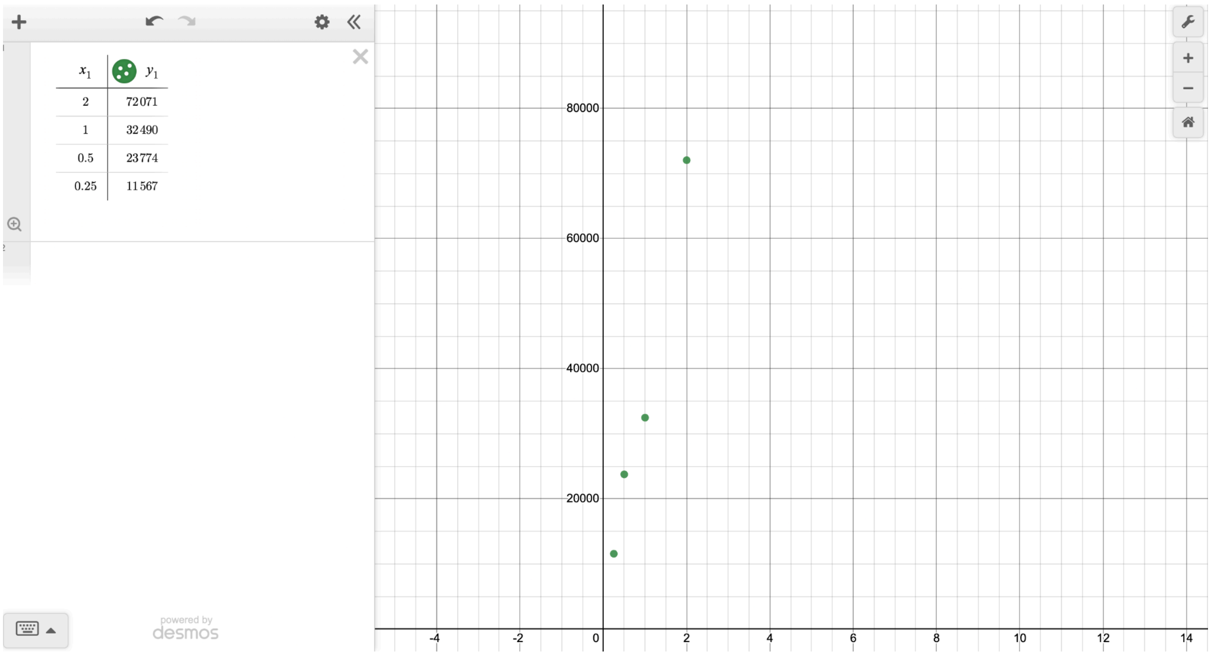
370000	828
380000	823
390000	827
400000	826
410000	848
420000	861
430000	827
440000	835
450000	839
460000	824
470000	839
480000	825
490000	815
500000	822

Output screenshot:

Varying the number of threads



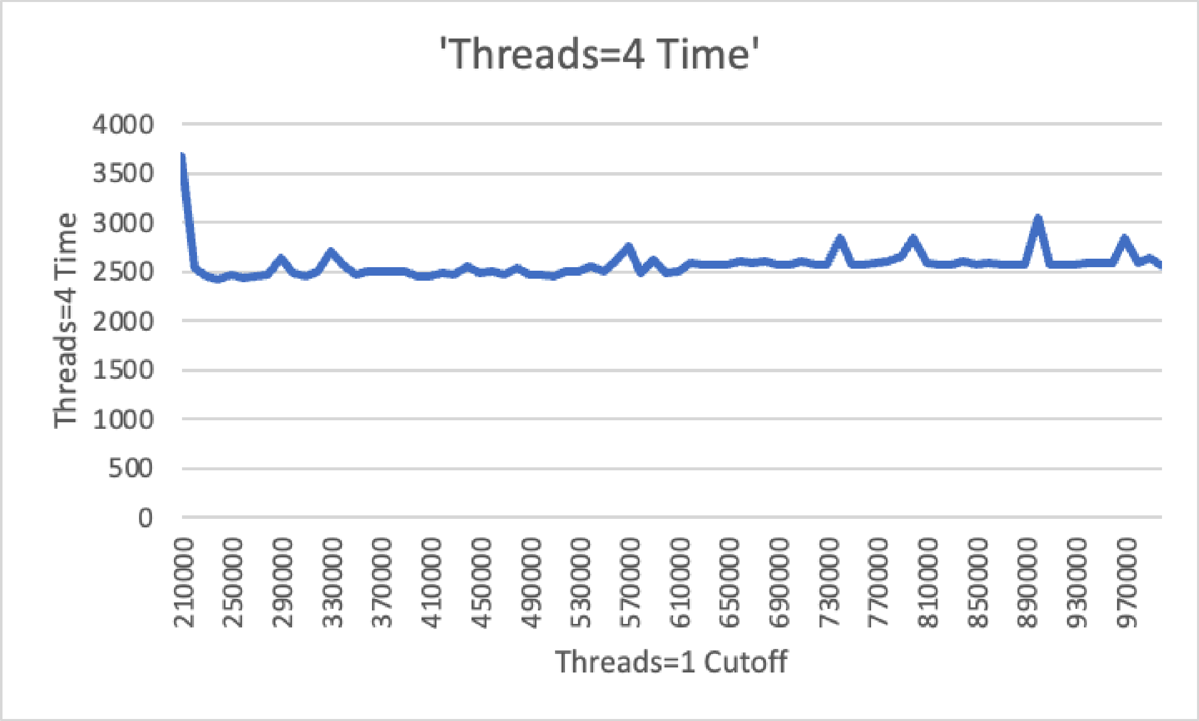
x_1	y_1
2	89606
4	72071
6	48911
8	48919
12	50796
16	48521

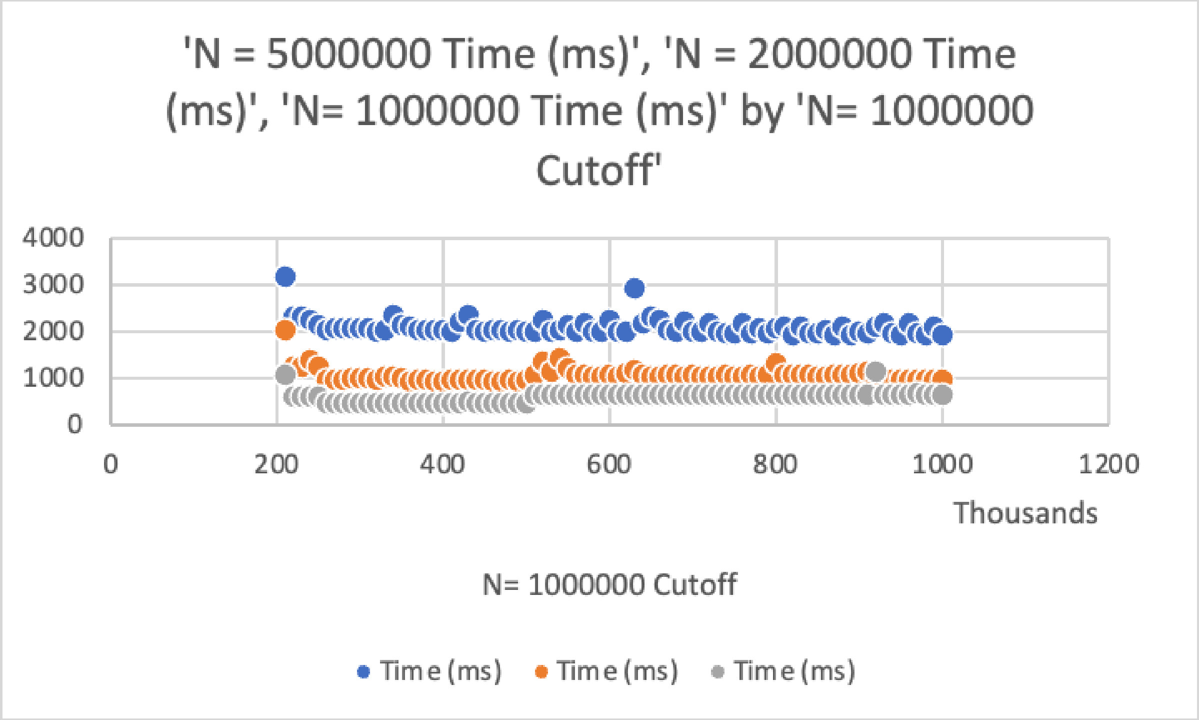


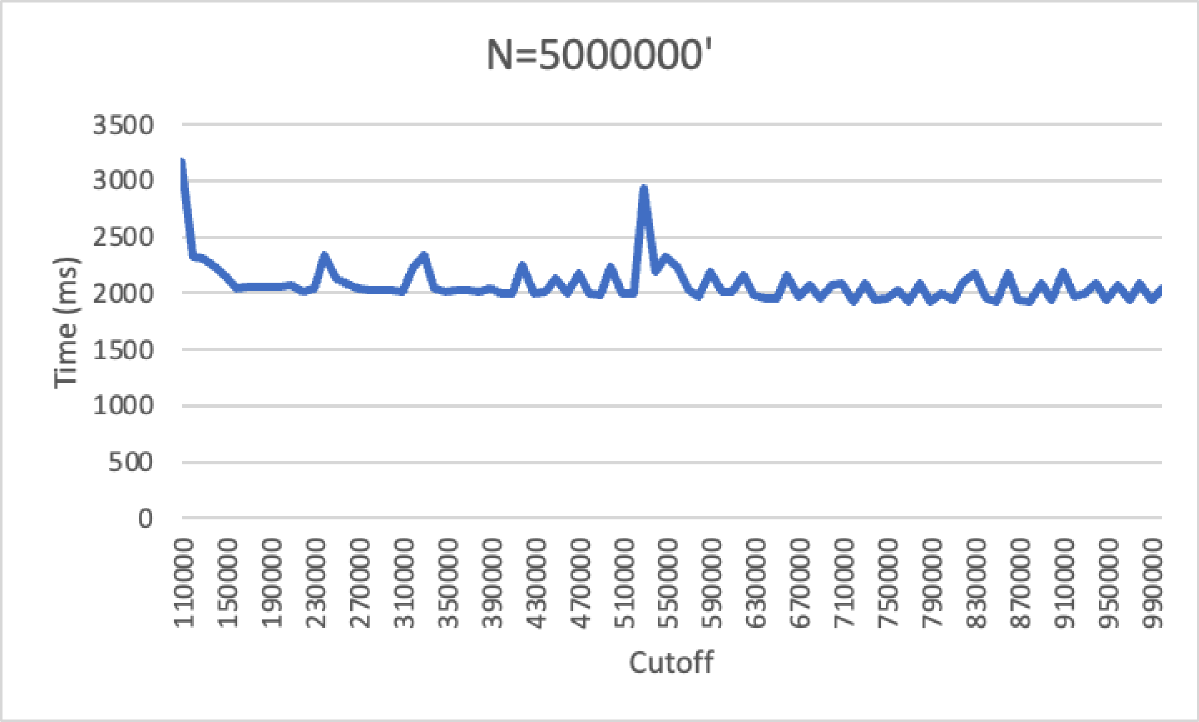
x_1	y_1
2	72071
1	32490
0.5	23774
0.25	11567

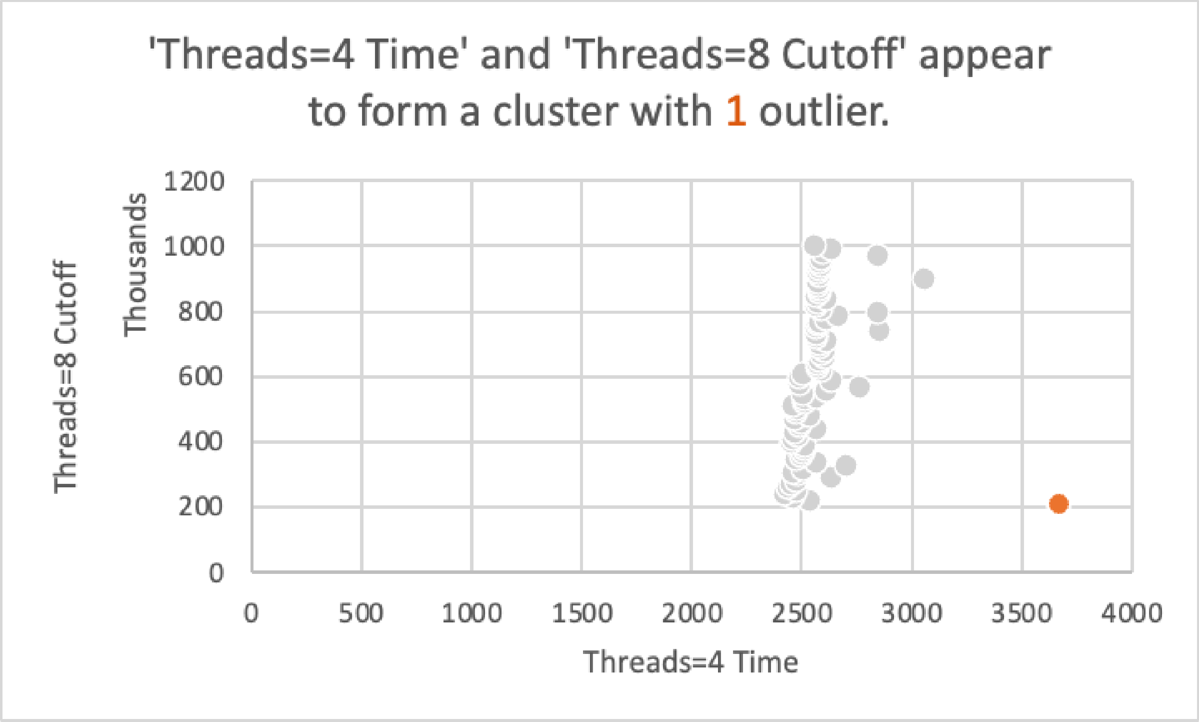
Array length vs time in ms

Graphs for different observations:
(Refer to the spreadsheet for data)

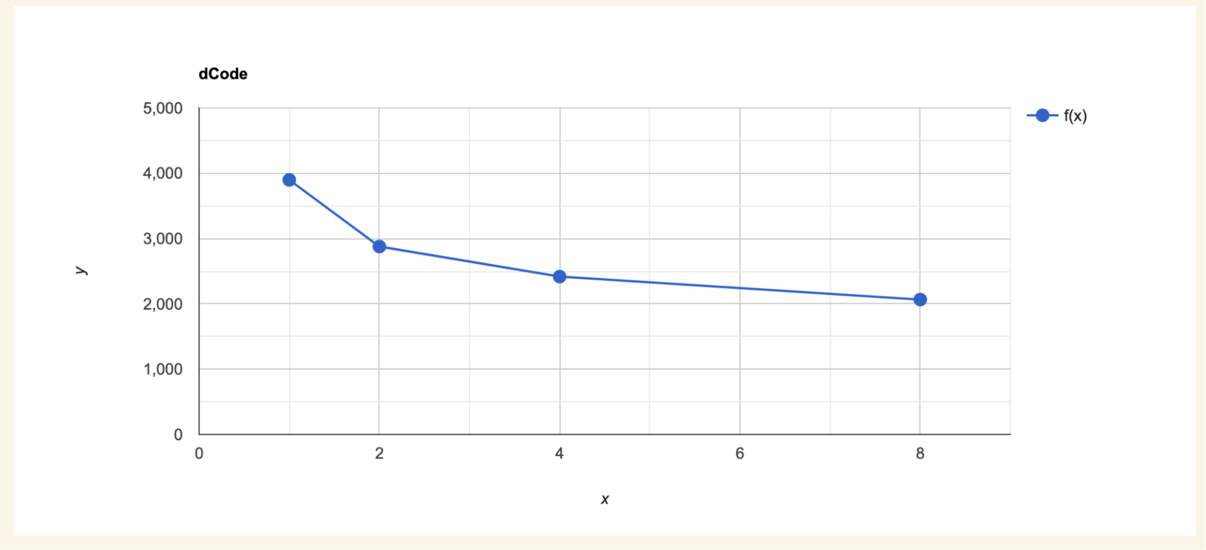





N=5000000'
Time (ms)
3500
3000
2500
2000
1500
1000
500
0
110000
150000
190000
230000
270000
310000
350000
390000
430000
470000
510000
550000
590000
630000
670000
710000
750000
790000
830000
870000
910000
950000
990000
Cutoff


'Threads=4 Time' and 'Threads=8 Cutoff' appear to form a cluster with 1 outlier.
Threads=8 Cutoff
Thousands
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
3000
3500
4000
Threads=4 Time

Minimum time for different number of threads



Conculsion:

Different computation times were measured for different array lengths(N=5000000,2000000,1000000), and for different number of threads for a constant array length(Threads=1,2,4,8).

It is observed that as the number of thread increase, the minimum time of computation decreases. If the number of thread is 2, then the minimum time taken is approximately at cutoff N/2, for thread 4, the minimum time taken is for cutoff N/4. So as the number of thread keeps on increasing, the appropriate cutoff value is at N/(No of threads). Furthermore, as the threads increase, the time of computation decreases, until a certain threshold is reached, beyond that value, the time stays steady and no drastic change is observed.