Assignment 10 Report

High Performance Computing

Pratik Mitra

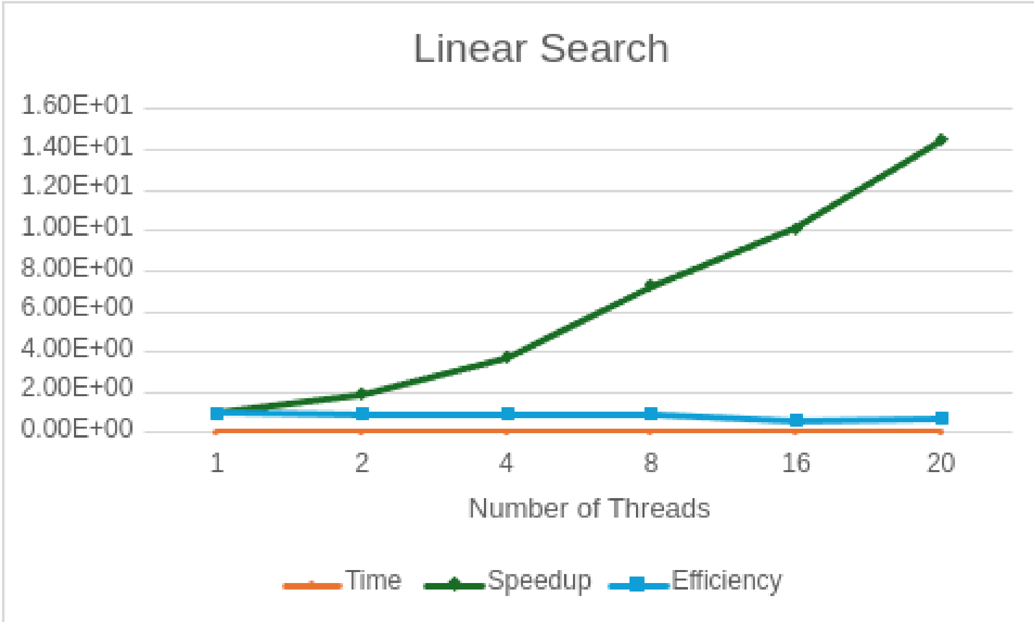
James Smith

2. OpenMP Parallelization of the linear search algorithm (Assignment 10).

Linear Search

Name or # of Threads	Time(s)	Speedup	Efficiency
1	3.94E-02	1	1
2	2.07E-02	1.907370851	0.953685426
4	1.06E-02	3.713527971	0.928381993
8	5.44E-03	7.24135513	0.905169391
16	3.90E-03	10.10556575	0.631597859
20	2.73E-03	14.44030764	0.722015382

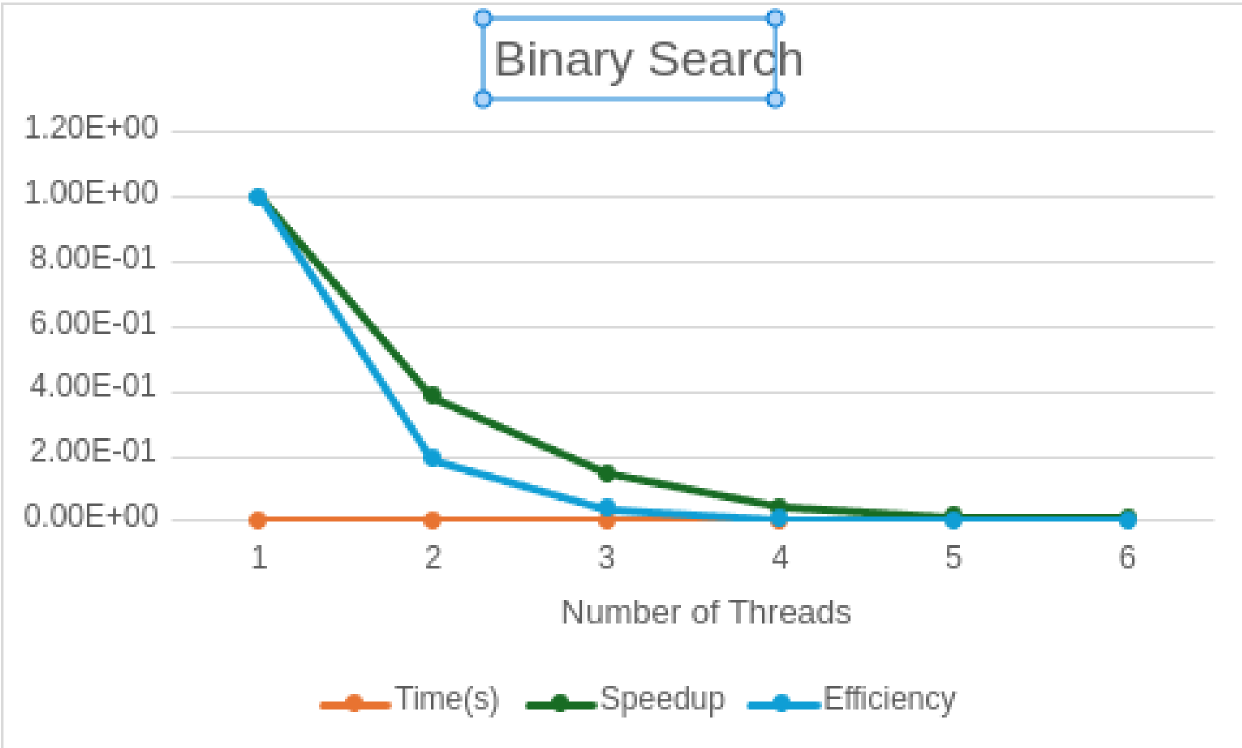
The table illustrates that increasing the number of threads significantly reduces computation time, with a notable peak speedup of approximately 10.1 using 16 threads. Increasing threads to 20 continues to decrease time but with diminishing efficiency, suggesting an optimal thread count exists where performance gains stabilize.



Binary Search

Name or # of Threads	Time(s)	Speedup	Efficiency
1	5.00E-06	1.00E+00	1
2	1.30E-05	3.85E-01	0.192307692
4	3.40E-05	1.47E-01	0.036764706
8	1.21E-04	4.13E-02	0.005165289
16	3.83E-04	1.31E-02	0.000815927
20	6.30E-04	7.94E-03	0.000396825

The performance table for binary search demonstrates a counterintuitive trend where increasing the number of threads leads to longer execution times, from just 5.00E-06 seconds with one thread to 6.30E-04 seconds with 20 threads. This results in a significant decrease in both speedup and efficiency as threads increase, highlighting the inefficiency of threading for a non-parallelized, inherently sequential process like binary search. The data underscores the importance of proper thread management to avoid negatively impacting the performance of sequential sections of a program.


Binary Search
1.20E+00
1.00E+00
8.00E-01
6.00E-01
4.00E-01
2.00E-01
0.00E+00
1
2
3
4
5
6
Number of Threads
Time(s)
Speedup
Efficiency