Delft University of Technology

Parallel Algorithms & Parallel Computers – IN4026
Lab Assignment Report

Name: Papadopoulos Charalampos **Student Number:** 4192141

Assignment A – Prefix/suffix minima

In this assignment we have to calculate the prefix/suffix minima for a N-size array. The algorithm which was chosen replaces a variable every time that finds a smaller one. That way we get the prefix/suffix minima for each element. The algorithm used is shown below.

```
for (m=0; m<N; m++){                                    O(1)
      if(m==0)
        prefix[m] = A[m];
      else{
        temp = A[m];
        for(n=0; n<m; n++){                             O(N)
          if(A[n] < temp)
            temp = A[n];
        }
        prefix[m] = temp;
     }
}
```

The time complexity is O(N).

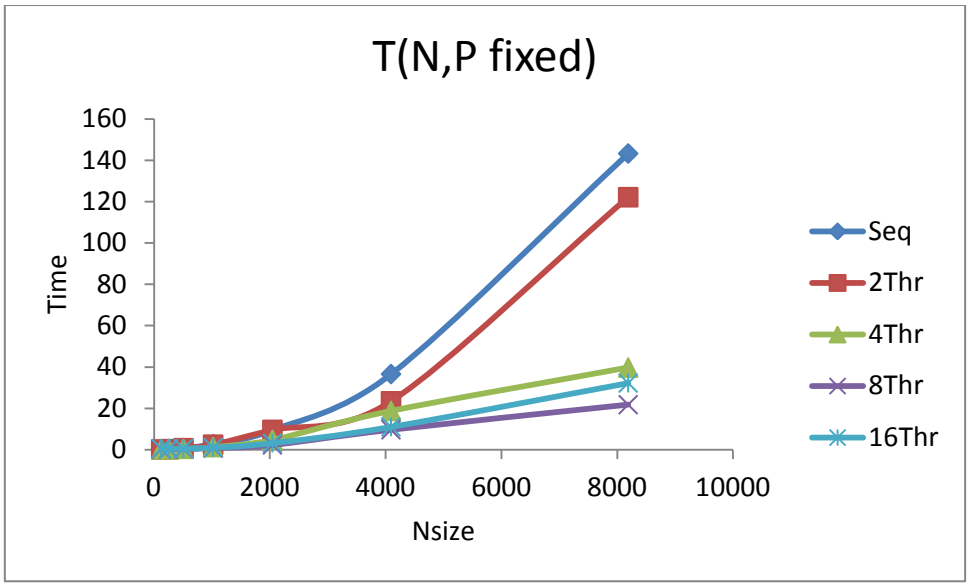
We used the online testbed in order to test the performance of the above algorithm for both pthread and openmp implementations. The results are shown below:

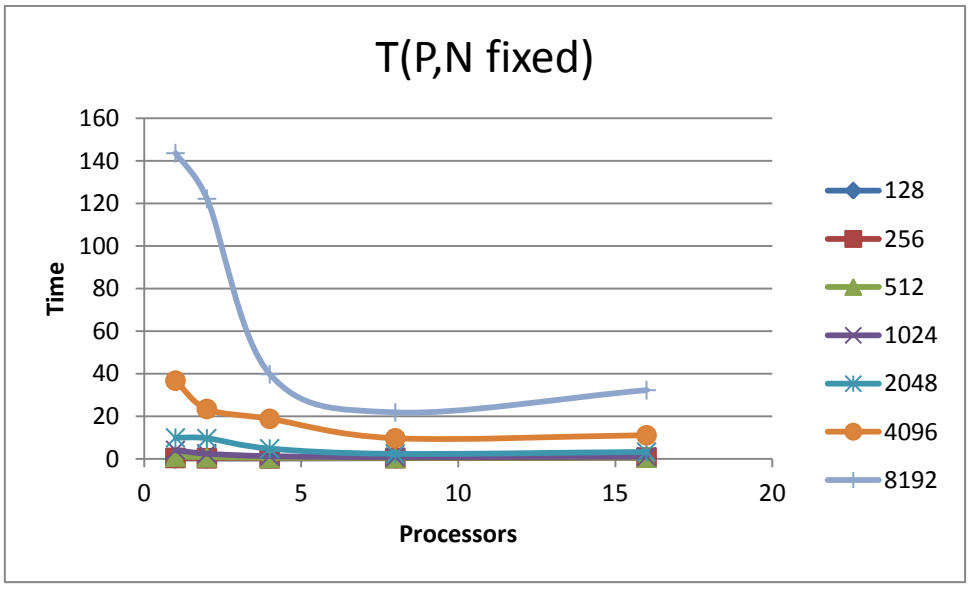
Nsize	Iterations	Seq	1Thr	2Thr	4Thr	8Thr	16Thr
128	1000	0,092764	0,132255	0,088334	0,112708	0,263166	0,565882
256	1000	0,274130	0,37757	0,213085	0,169922	0,269927	0,525003
512	1000	0,792148	0,952321	0,682286	0,376397	0,373519	0,540864
1024	1000	2,656997	3,992184	2,384869	1,235895	0,738879	0,987292
2048	1000	9,461020	9,859274	9,544925	4,734284	2,274523	3,172909
4096	1000	36,553595	36,695863	23,386846	18,75676	9,598136	11,031533
8192	1000	143,263395	143,551423	122,111649	39,724467	21,753398	32,217022

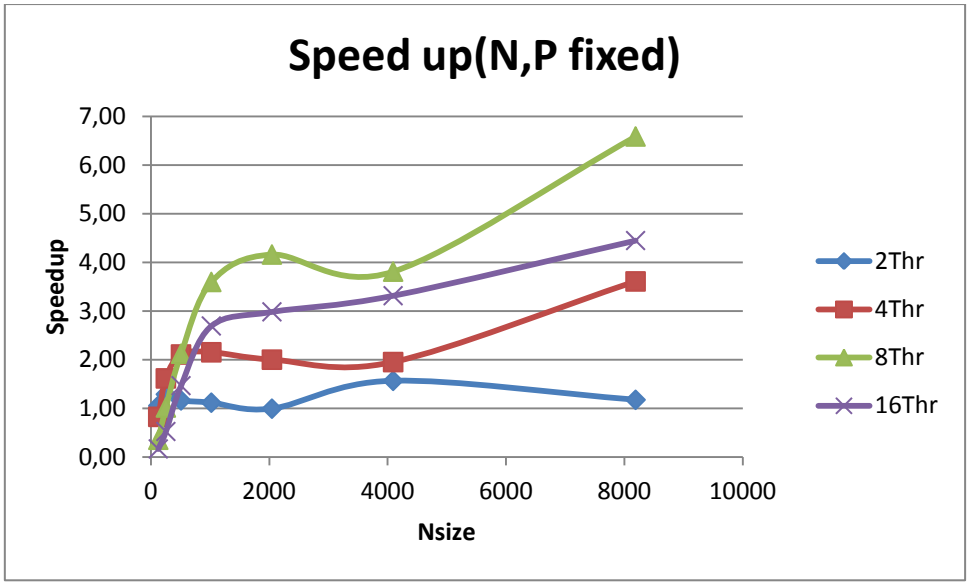
We can compute the following speed ups:

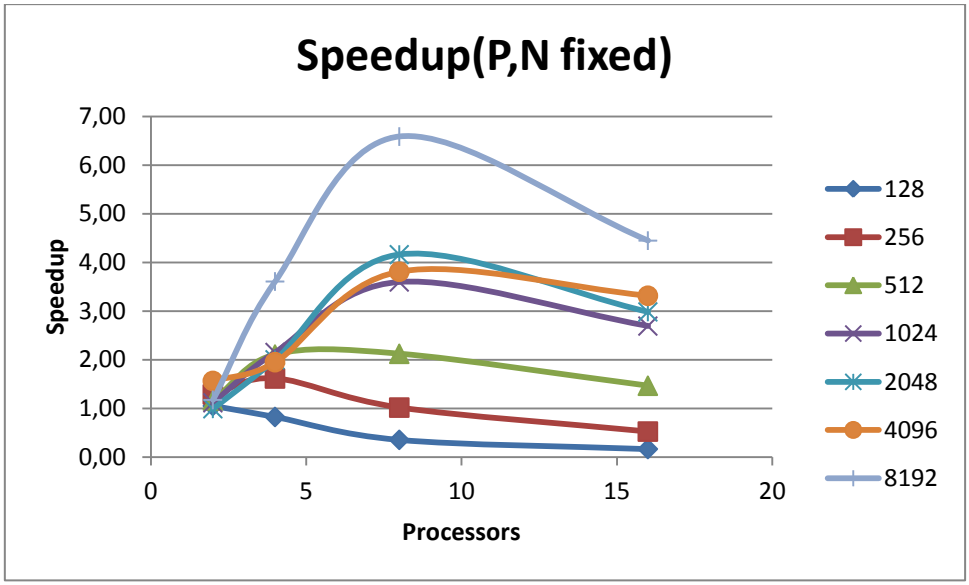
Nsize	2Thr	4Thr	8Thr	16Thr
128	1,05	0,82	0,35	0,16
256	1,29	1,61	1,02	0,52
512	1,16	2,10	2,12	1,46
1024	1,11	2,15	3,60	2,69
2048	0,99	2,00	4,16	2,98
4096	1,56	1,95	3,81	3,31
8192	1,17	3,61	6,59	4,45

We get the following charts.









As a conclusion we could observe from the graphs that it is not true that the more threads we have, the faster the execution. Higher parallelism sometimes leads to slower performance. The factors that affect the overall performance are the overhead of parallelism (thread creation, synchronization, etc), the structure of the algorithm that is parallelized and the computer architecture of the system that is tested.

Assignment B – Simple merge

In this exercise we have to merge two arrays A and B with size N. The algorithm that we use is the same that is presented in the lecture's slides and is based on the recursive binary search.

```
for ( j=0 ; j <n ; j++){                                                    O(N)
        if(A[j] < B[0])
                rankA_in_AB[j] = j;
        else if(A[j] > B[k-1])
                rankA_in_AB[j] = k+j;
        else
                rankA_in_AB[j] = Search(B, A[j], 0, k) + j;                 O(logN)
        C[rankA_in_AB[j]] = A[j];
}
```

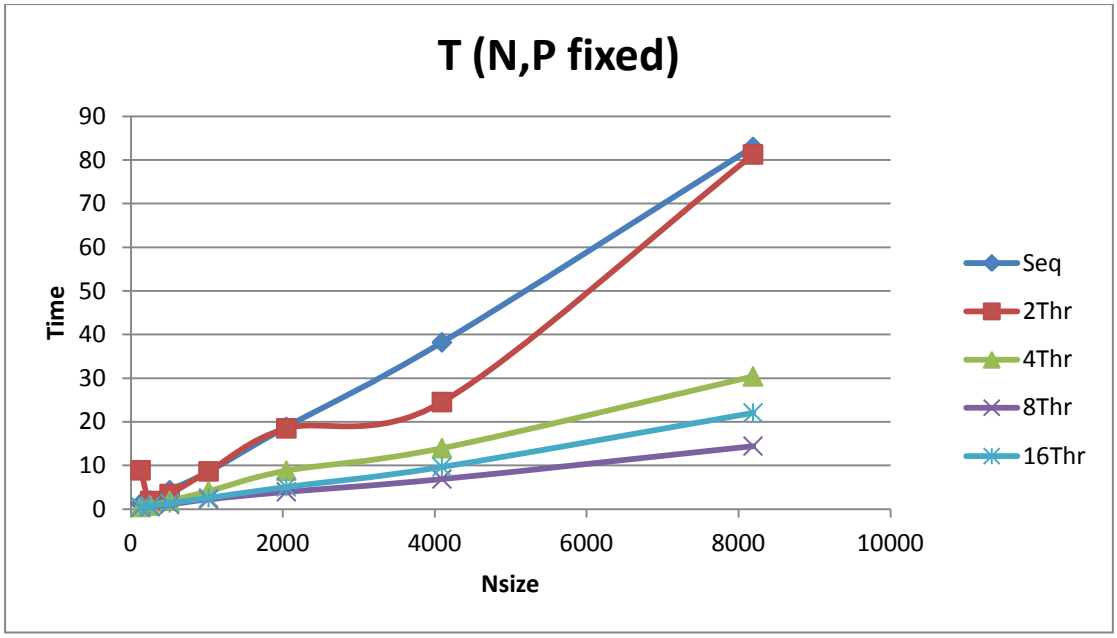
The time complexity is O(NlogN). The pthread and openmp implementations can be found in the delivered code. The template code was used to give us the following results:

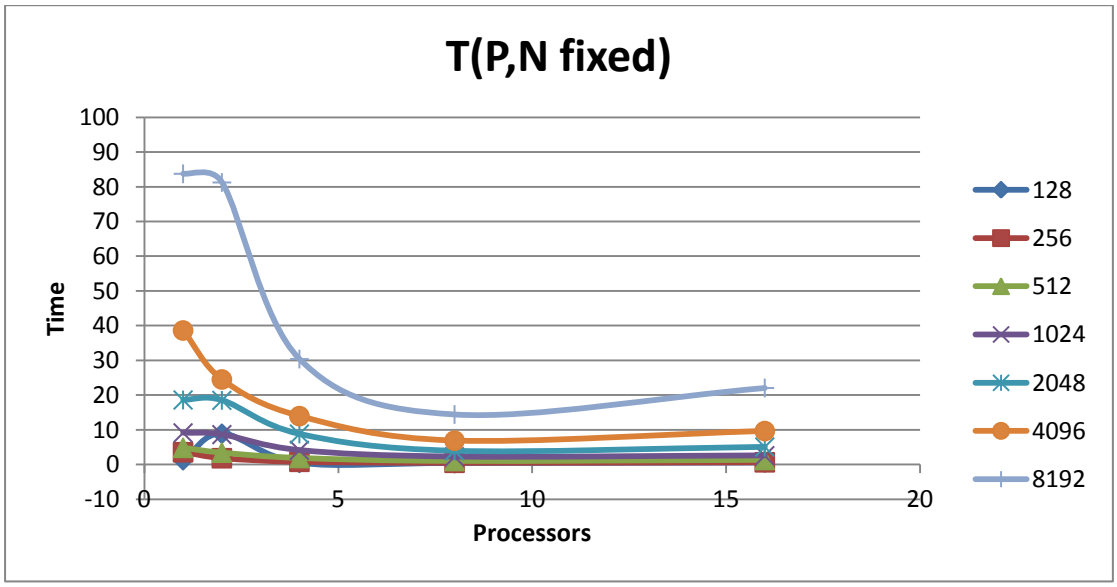
Nsize	Iterations	Seq	1Thr	2Thr	4Thr	8Thr	16Thr
128	1000	1,198839	1,272473	8,900600	0,500770	0,418473	0,600392
256	1000	1,981547	3,570586	1,845646	0,814499	0,535546	0,680362
512	1000	4,329292	4,844096	3,447675	1,912211	1,047754	1,349210
1024	1000	8,590491	9,143011	8,649840	4,123844	2,259516	2,584165
2048	1000	18,827921	18,616900	18,512599	8,812937	3,959173	5,079825
4096	1000	38,205463	38,635299	24,530225	13,972135	6,900450	9,666590
8192	1000	82,986105	83,770269	81,272228	30,413187	14,449036	22,067805

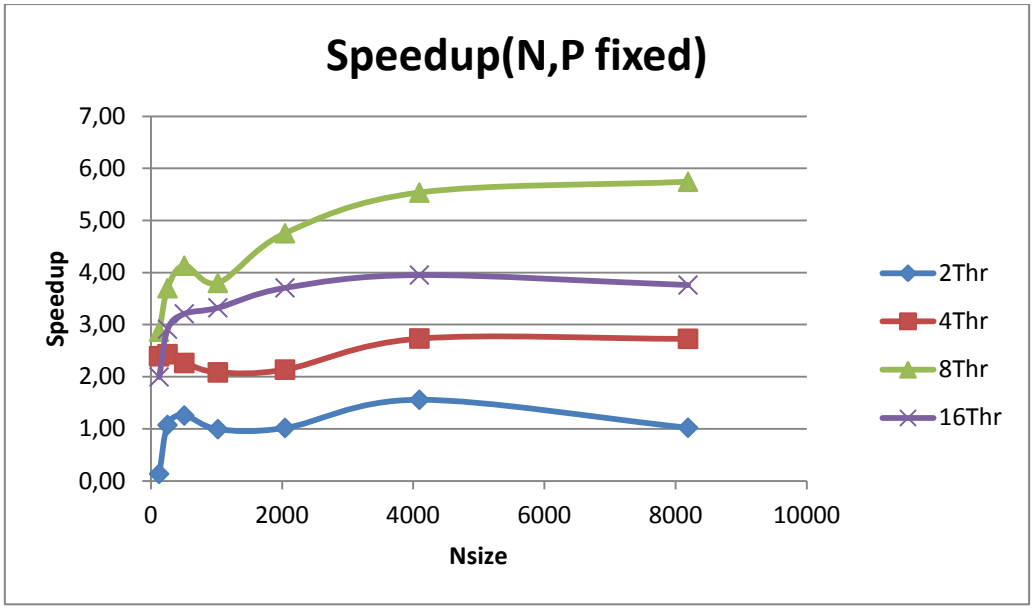
Based on the above results we computed the following speedups:

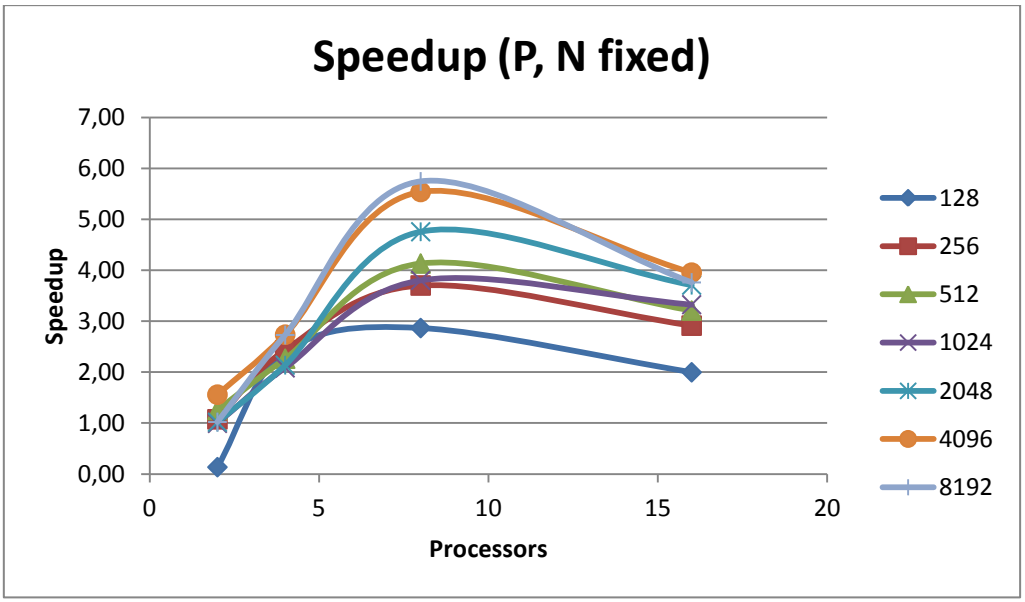
Nsize	2Thr	4Thr	8Thr	16Thr
128	0,13	2,39	2,86	2,00
256	1,07	2,43	3,70	2,91
512	1,26	2,26	4,13	3,21
1024	0,99	2,08	3,80	3,32
2048	1,02	2,14	4,76	3,71
4096	1,56	2,73	5,54	3,95
8192	1,02	2,73	5,74	3,76

Using the data presented above we plotted the following graphs:


T (N,P fixed)
Time
Nsize
0
10
20
30
40
50
60
70
80
90
0
2000
4000
6000
8000
10000
Seq
2Thr
4Thr
8Thr
16Thr


T(P,N fixed)
Time
Processors
-10
0
10
20
30
40
50
60
70
80
90
100
0
5
10
15
20
128
256
512
1024
2048
4096
8192





The conclusions are the same as before. Higher parallelism sometimes leads to slower performance. The factors that affect the overall performance are the overhead of parallelism (thread creation, synchronization, etc), the structure of the algorithm that is parallelized and the computer architecture of the system that is tested.

Assignment C – List Ranking (Pointer Jumping)

We are asked to compute the distance of each element of a linked list L to the last node of the list. A simple algorithm was chosen in order to have also simplified parallel versions. The algorithm uses the standard pointer jumping algorithm as presented in the course lectures with known time complexity O(logN).

```
for(n=0; n<N; n++){
        S_matrix[n] = L_matrix[n];
        W_matrix[n] = 0;
}
for(n=0; n<N; n++){
        while(S_matrix[n]!=0){
                W_matrix[n] += W_matrix[S_matrix[n]-1] + 1;
                S_matrix[n] = S_matrix[S_matrix[n]-1];
        }
}
```

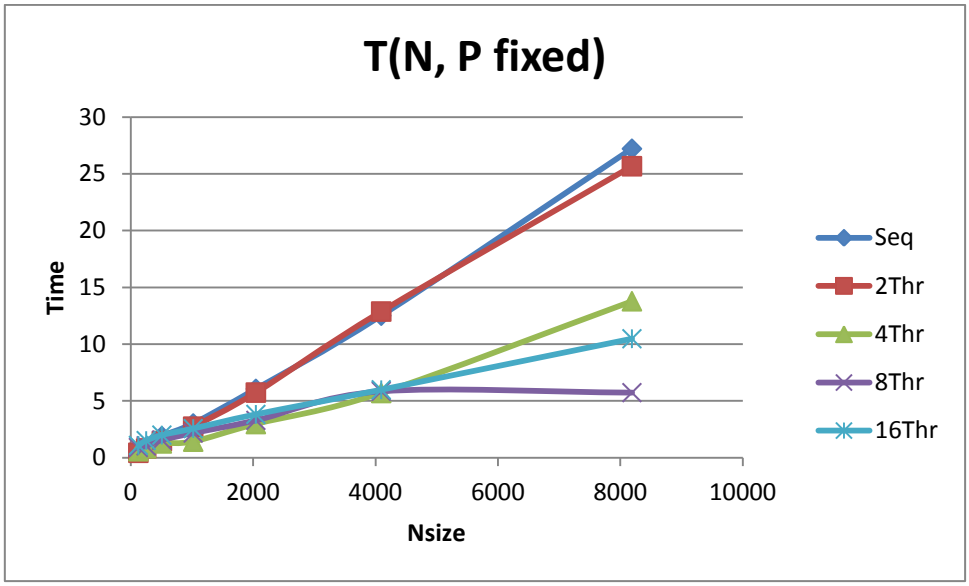
The time complexity is O(logN). The pthread and openmp implementations can be found in the delivered code. The template code was used to give us the following results:

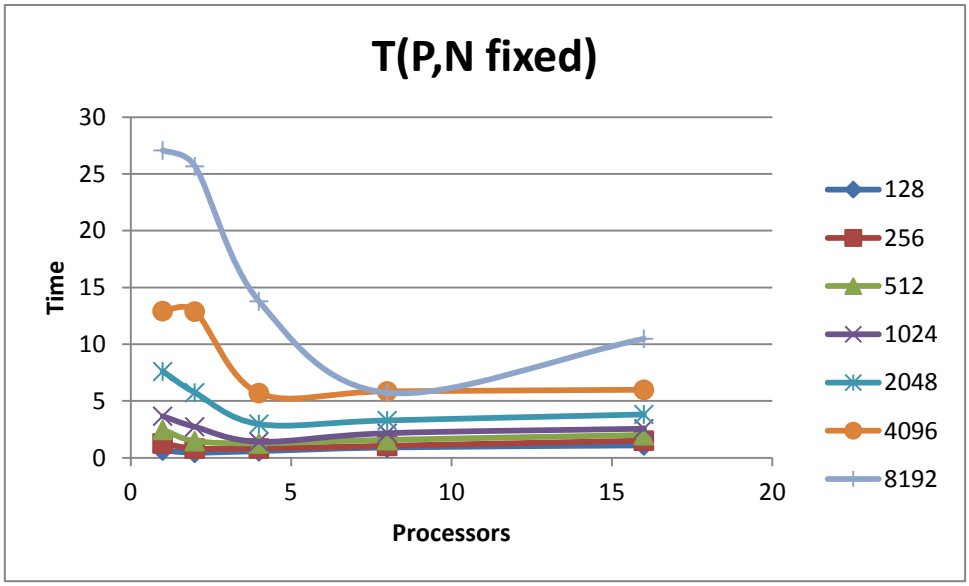
Nsize	Iterations	Seq	1Thr	2Thr	4Thr	8Thr	16Thr
128	1000	0,576131	0,631843	0,434716	0,571233	0,895217	1,083345
256	1000	0,852491	1,266152	0,815147	0,822822	1,048739	1,507989
512	1000	1,894588	2,458866	1,460466	1,259745	1,561194	1,993463
1024	1000	3,012491	3,663594	2,732426	1,415682	2,170480	2,564053
2048	1000	6,059144	7,598995	5,736356	2,966235	3,289474	3,816218
4096	1000	12,511886	12,918181	12,875621	5,699568	5,846194	5,985614
8192	1000	27,215080	27,075899	25,673963	13,783082	5,730388	10,479467

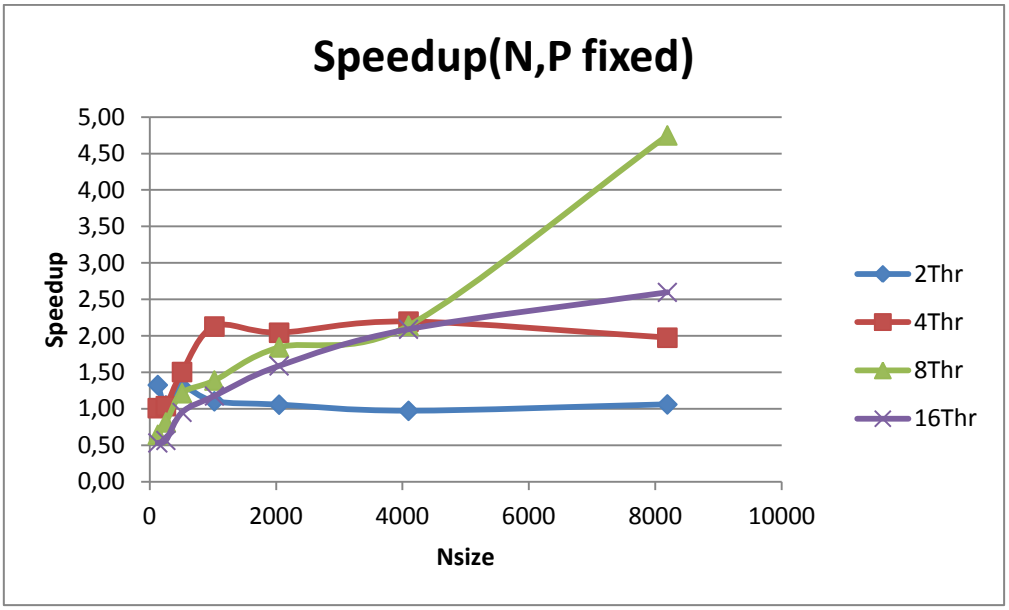
Based on the above results we computed the following speedups:

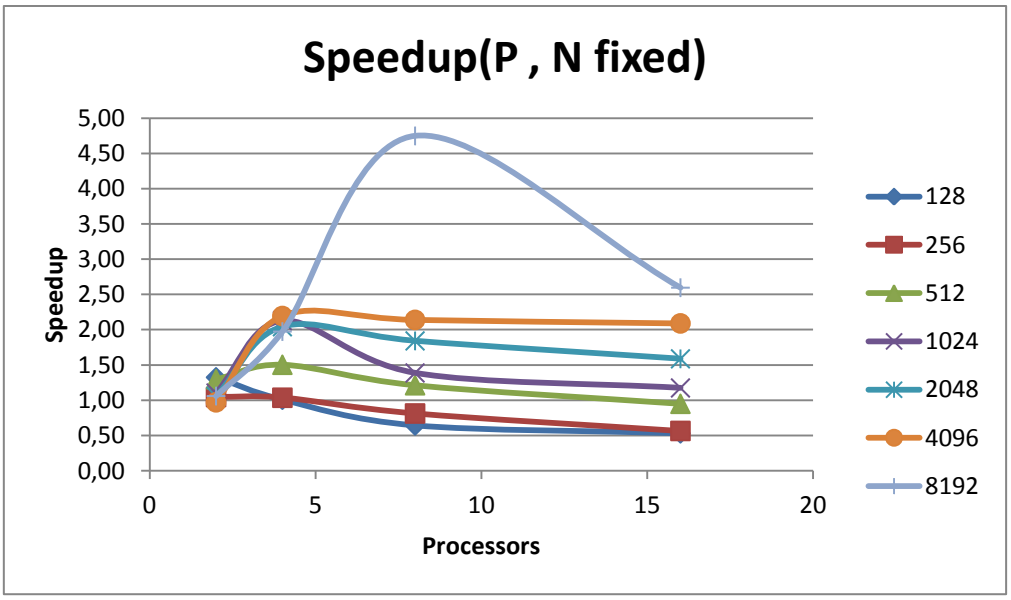
Nsize	2Thr	4Thr	8Thr	16Thr
128	1,33	1,01	0,64	0,53
256	1,05	1,04	0,81	0,57
512	1,30	1,50	1,21	0,95
1024	1,10	2,13	1,39	1,17
2048	1,06	2,04	1,84	1,59
4096	0,97	2,20	2,14	2,09
8192	1,06	1,97	4,75	2,60

Using the data presented above we plotted the following graphs:


T(N, P fixed)
Time
Nsize
0
5
10
15
20
25
30
0
2000
4000
6000
8000
10000
Seq
2Thr
4Thr
8Thr
16Thr


T(P,N fixed)
Time
Processors
0
5
10
15
20
25
30
0
5
10
15
20
128
256
512
1024
2048
4096
8192





The conclusions are the same as before. Higher parallelism sometimes leads to slower performance. The factors that affect the overall performance are the overhead of parallelism (thread creation, synchronization, etc), the structure of the algorithm that is parallelized and the computer architecture of the system that is tested.