CSE4001- PARALLEL AND DISTRIBUTED COMPUTING LAB

Name: Arya Adarsh

Reg. No: 19BCE1556

Programming Environment: OpenMP

Problem: Prefix sum problem, Calculation of PI using critical construct, single construct

Date: 22-09-2021

Hardware Configuration:

$ lscpu

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          1
On-line CPU(s) list:             0
Thread(s) per core:              1
Core(s) per socket:              1
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           142
Model name:                      Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Stepping:                        10
CPU MHz:                         1800.002
BogoMIPS:                        3600.00
Hypervisor vendor:               KVM
Virtualization type:             full
L1d cache:                       32 KiB
L1i cache:                       32 KiB
L2 cache:                        256 KiB
L3 cache:                        6 MiB
NUMA node0 CPU(s):               0
Vulnerability Itlb multihit:     KVM: Mitigation: VMX unsupported
Vulnerability L1tf:              Mitigation; PTE Inversion
Vulnerability Mds:               Mitigation; Clear CPU buffers; SMT Host state
```

$

Task 1: Prefix Sum Problem:

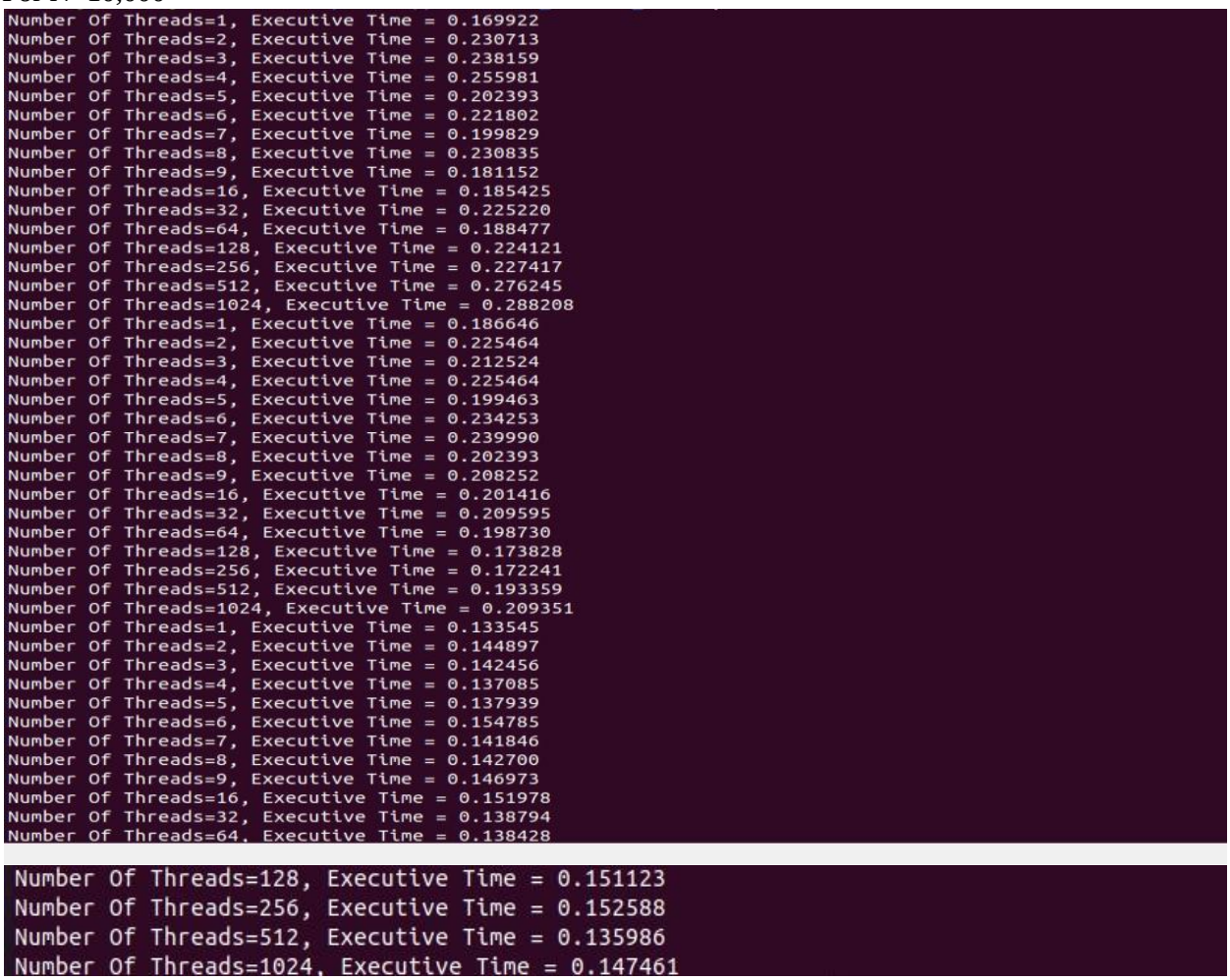
CODE:

```
#include <stdio.h>
#include <omp.h>

#define N 10000

int main() {
   int array[N],CHUNK=10000;
   int t[16] = {1,2,3,4,5,6,7,8,9,16,32,64,128,256,512,1024};
   // Initalize the array
for (int i = 0; i < N; i++)
   {
      array[i] = i + 1;
   }
   for (int k = 0; k < 16; k++)
   {
      omp_set_num_threads(t[k]);
      int prefix_array[N] = {0};
      int i, j;
      float startT = omp_get_wtime();
#pragma omp parallel for shared(prefix_array, array) private(i, j) schedule(dynamic, CHUNK)
for (int i = 0; i < N; i++)
         for (int j = 0; j <= i; j++)
         {
            prefix_array[i] += array[j];
         }
      float endT = omp_get_wtime();
float exectime = endT - startT;
      printf("Number Of Threads=%d, Executive Time = %f\n",t[k],exectime);
   }
   for (int k = 0; k < 16; k++)
   {
      omp_set_num_threads(t[k]);
int prefix_array[N] = {0};
      int i, j;
      float startT = omp_get_wtime();
#pragma omp parallel for shared(prefix_array, array) private(i, j) schedule(static, CHUNK)
for (int i = 0; i < N; i++)          for (int j = 0; j <= i; j++)
         {
            prefix_array[i] += array[j];
         }
      float endT = omp_get_wtime();
float exectime = endT - startT;
      printf("Number Of Threads=%d, Executive Time = %f\n",t[k],exectime);
   }
   for (int k = 0; k < 16; k++)
   {
      omp_set_num_threads(t[k]);
      int prefix_array[N] = {0};
```

```
        int i, j;
        float startT = omp_get_wtime();
for (int i = 0; i < N; i++)
            for (int j = 0; j <= i; j++)
            {
                prefix_array[i] += array[j];
            }
        float endT = omp_get_wtime();
float exectime = endT - startT;
        printf("Number Of Threads=%d, Executive Time = %f\n",t[k],exectime);
    }
return 0; }
```

OUTPUT:

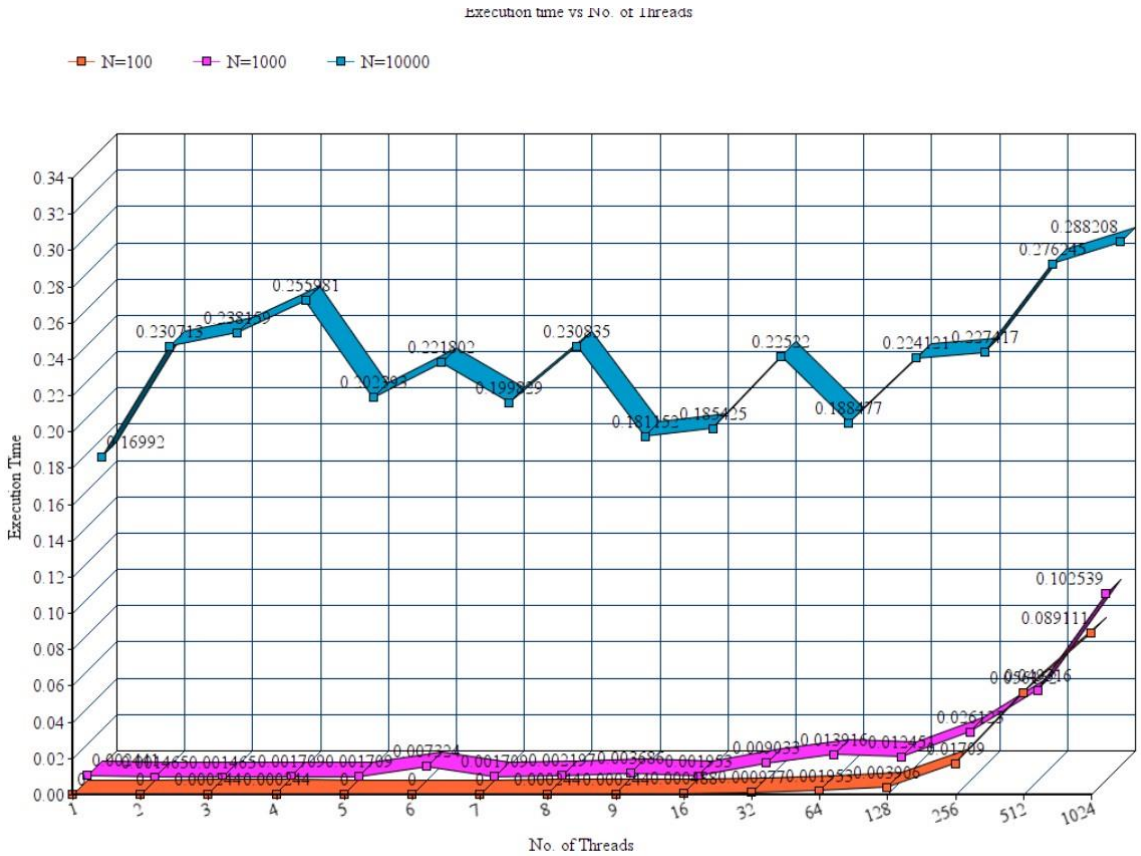
For N=10,000



Execution Time:

Static N=10000	Static N=1000	Static N=100	Dynamic N=10000	Dynamic N=1000	Dynamic N=100	Default N=10000	Default N=1000	Default N=100
0.186646	0.001221	0.000000	0.169922	0.002441	0.000000	0.133545	0.001221	0.000000
0.225464	0.084473	0.013428	0.230713	0.001465	0.000000	0.144897	0.000977	0.000244
0.212524	0.003418	0.067871	0.238159	0.001465	0.000244	0.142456	0.002686	0.000000
0.225464	0.001709	0.000244	0.255981	0.001709	0.000244	0.137085	0.001221	0.000000
0.199463	0.001953	0.000000	0.202393	0.001709	0.000000	0.137939	0.000977	0.000000
0.234253	0.002441	0.000244	0.221802	0.007324	0.000000	0.154785	0.001221	0.000244
0.239990	0.002441	0.000244	0.199829	0.001709	0.000000	0.141846	0.008545	0.000000
0.202393	0.003174	0.000244	0.230835	0.002197	0.000244	0.142700	0.001221	0.000000
0.208252	0.003418	0.000244	0.181152	0.003686	0.000244	0.146973	0.001221	0.000000
0.201416	0.001953	0.000488	0.185425	0.001953	0.000488	0.151978	0.001221	0.000000
0.209595	0.003174	0.000977	0.225220	0.009033	0.000977	0.138794	0.001221	0.000000
0.198730	0.013428	0.001709	0.188477	0.013916	0.001953	0.138428	0.001221	0.000000
0.173828	0.013916	0.008789	0.224121	0.012451	0.003906	0.151123	0.001221	0.000000
0.172241	0.031250	0.022949	0.227417	0.026123	0.017090	0.152588	0.001221	0.000000
0.193359	0.072998	0.046631	0.276245	0.049316	0.056152	0.135986	0.002197	0.000000
0.209351	0.107178	0.099365	0.288208	0.102539	0.089111	0.147461	0.001221	0.000000

Execution Time vs No. of threads:
For Dynamic:



Task2: Pi calculation using Critical Construct:

CODE:

```
#include<stdio.h>
#include<omp.h>
#include <math.h> #include<stdlib.h>
#define PI 3.1415926538837211
int main()
{
        int         N, i,CHUNK=10000,k;
        float        sum, x, tsum, h, psum, sumt,starttime,endtime,exectime;

        printf("Enter number of intervals\n");
scanf("%d", &N);

        if (N <= 0) {
                printf("Number of intervals should be positive integer\n");
                exit(1);
        }
        sum = 0.0;
        h = 1.0 / N;
        int t[16] = {1,2,3,4,5,6,7,8,9,16,32,64,128,256,512,1024};
for(k=0;k<16;k++)
        {
        starttime=omp_get_wtime();
#pragma omp parallel for private(x) shared(sumt) schedule(static,CHUNK)
for (i = 1; i < N + 1; i = i + 1)

{
                x = h * (i - 0.5);




                sumt = sumt + 4.0 / (1 + x * x);
        }
        #pragma omp critical
        psum = sumt * h;



#pragma omp critical
        sum = sum + psum;
endtime=omp_get_wtime();       exectime =
endtime - starttime;
        printf("Number Of Threads=%d, Executive Time = %f\n",t[k],exectime);
}
}
```

OUTPUT:

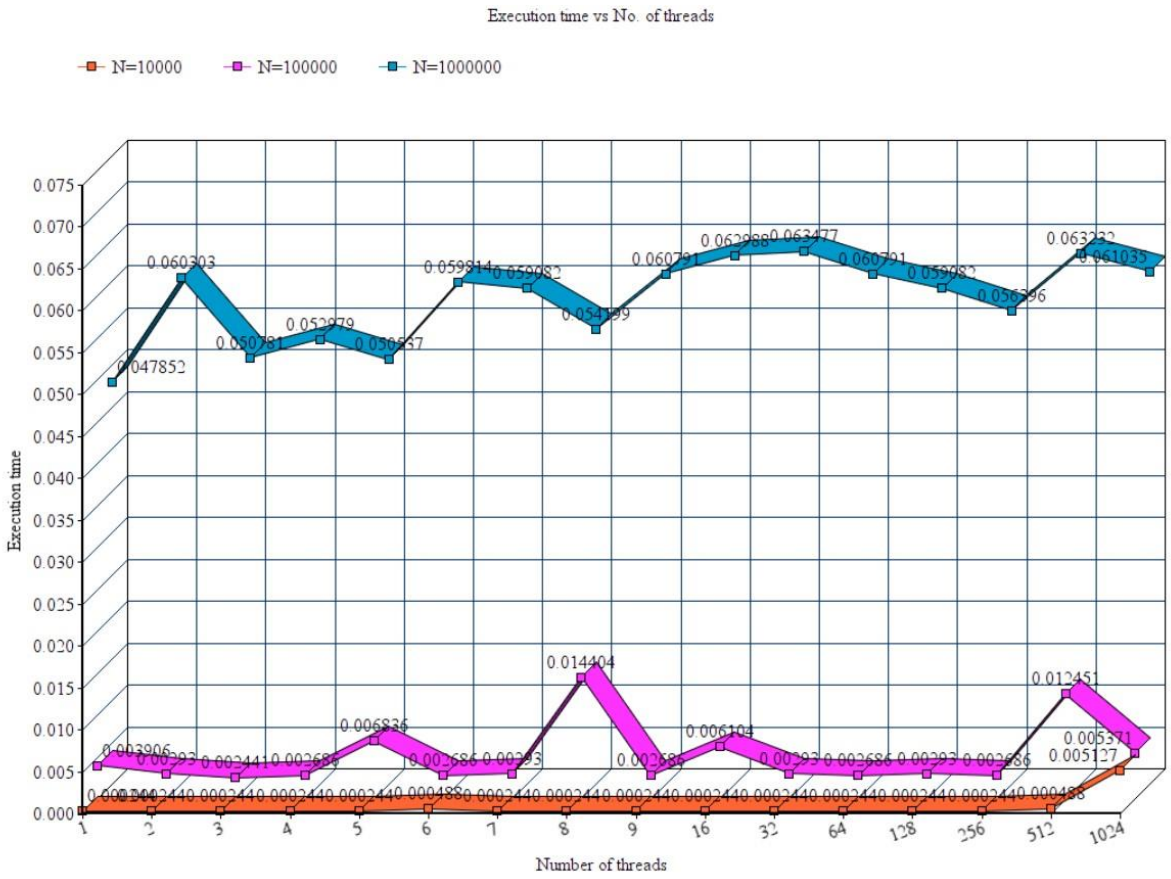
FOR STATIC N=10,000

```
Enter number of intervals
10000
The value of PI is      3.141591
error is        0.0000018202197807
Number Of Threads=1, Executive Time = 0.000366
The value of PI is      9.424780
error is        6.2831872380840519
Number Of Threads=2, Executive Time = 0.000244
The value of PI is      18.849567
error is        15.7079747594463566
Number Of Threads=3, Executive Time = 0.000244
The value of PI is      31.415968
error is        28.2743752874004599
Number Of Threads=4, Executive Time = 0.000366
The value of PI is      47.123962
error is        43.9823697484600302
Number Of Threads=5, Executive Time = 0.000244
The value of PI is      65.973595
error is        62.8320020116436240
Number Of Threads=6, Executive Time = 0.000732
The value of PI is      87.964867
error is        84.8232739842998740
Number Of Threads=7, Executive Time = 0.000366
The value of PI is      113.097778
error is        109.9561856664287802
Number Of Threads=8, Executive Time = 0.000244
The value of PI is      141.372330
error is        138.2307370580303427
Number Of Threads=9, Executive Time = 0.000244
The value of PI is      172.788620
error is        169.6470273412334677
Number Of Threads=16, Executive Time = 0.000488
The value of PI is      207.346649
error is        204.2050565160381552
Number Of Threads=32, Executive Time = 0.000488
The value of PI is      245.046417
error is        241.9048245824444052
Number Of Threads=64, Executive Time = 0.000244
The value of PI is      285.887909
error is        282.7463162816631552
Number Of Threads=128, Executive Time = 0.000366
The value of PI is      329.871155
```

Execution Time:

	Static N=10000	Static N=100000	Static N=1000000	Dynamic N=10000	Dynamic N=100000	Dynamic N=1000000	Default N=10000	Default N=100000	Default N=1000000
1	0.000366	0.003174	0.059570	0.000244	0.003906	0.047852	0.000000	0.001709	0.026123
2	0.000244	0.008423	0.052246	0.000244	0.002930	0.060303	0.000000	0.001709	0.037598
3	0.000244	0.019897	0.066162	0.000244	0.002441	0.050781	0.000244	0.001953	0.043945
4	0.000366	0.003540	0.062744	0.000244	0.002686	0.052979	0.000244	0.001709	0.034424
5	0.000244	0.007324	0.063232	0.000244	0.006836	0.050537	0.000000	0.001709	0.039062
6	0.000732	0.003174	0.068115	0.000488	0.002686	0.059814	0.000244	0.001953	0.044434
7	0.000366	0.006958	0.058838	0.000244	0.002930	0.059082	0.000244	0.001709	0.042236
8	0.000244	0.010986	0.058838	0.000244	0.014404	0.054199	0.000244	0.004883	0.032715
9	0.000244	0.006714	0.059570	0.000244	0.002686	0.060791	0.000000	0.001709	0.038330
16	0.000488	0.005371	0.060547	0.000244	0.006104	0.062988	0.000244	0.001953	0.036865
32	0.000488	0.002808	0.060059	0.000244	0.002930	0.063477	0.000244	0.009277	0.031982
64	0.000244	0.002930	0.047852	0.000244	0.002686	0.060791	0.000000	0.001709	0.031494
128	0.000366	0.009399	0.058105	0.000244	0.002930	0.059082	0.000000	0.001709	0.032715
256	0.000244	0.007324	0.066650	0.000244	0.002686	0.056396	0.000244	0.001709	0.038330
512	0.005249	0.002808	0.058105	0.000488	0.012451	0.063232	0.000244	0.001709	0.038818
1024	0.000366	0.013184	0.051758	0.005127	0.005371	0.061035	0.000244	0.001709	0.037842

Execution Time vs No. of threads:
For dynamic:



Task3: Pi calculation using single construct

CODE:

```
#include<stdio.h>
#include<omp.h>
#include <math.h> #include<stdlib.h>
#define PI 3.1415926538837211
int main()
{
        int         N, i,CHUNK=10000,k;
        float        sum, x, tsum, h, psum, sumt,starttime,endtime,exectime;

        printf("Enter number of intervals\n");
scanf("%d", &N);

        if (N <= 0) {
                printf("Number of intervals should be positive integer\n");
                exit(1);
        }
        sum = 0.0;
        h = 1.0 / N;
        int t[16] = {1,2,3,4,5,6,7,8,9,16,32,64,128,256,512,1024};
for(k=0;k<16;k++)
        {
        starttime=omp_get_wtime();
#pragma omp parallel for private(x) shared(sumt) schedule(static,CHUNK)
for (i = 1; i < N + 1; i = i + 1)

{
                x = h * (i - 0.5);




                sumt = sumt + 4.0 / (1 + x * x);
        }
        #pragma omp single
        psum = sumt * h;



#pragma omp single
        sum = sum + psum;
endtime=omp_get_wtime();       exectime =
endtime - starttime;
        printf("Number Of Threads=%d, Executive Time = %f\n",t[k],exectime);
}
}
```

OUTPUT:

FOR DYNAMIC N = 10,000

```
Enter number of intervals
10000
Number Of Threads=1, Executive Time = 0.000244
Number Of Threads=2, Executive Time = 0.000244
Number Of Threads=3, Executive Time = 0.000000
Number Of Threads=4, Executive Time = 0.000244
Number Of Threads=5, Executive Time = 0.000244
Number Of Threads=6, Executive Time = 0.000244
Number Of Threads=7, Executive Time = 0.000000
Number Of Threads=8, Executive Time = 0.000244
Number Of Threads=9, Executive Time = 0.000244
Number Of Threads=16, Executive Time = 0.000000
Number Of Threads=32, Executive Time = 0.000244
Number Of Threads=64, Executive Time = 0.000244
Number Of Threads=128, Executive Time = 0.000244
Number Of Threads=256, Executive Time = 0.000000
Number Of Threads=512, Executive Time = 0.000244
Number Of Threads=1024, Executive Time = 0.000244
```

Execution Time:

	Static N=10000	Static N=100000	Static N=1000000	Dynamic N=10000	Dynamic N=100000	Dynamic N=1000000	Default N=10000	Default N=100000	Default N=1000000
1	0.000244	0.001709	0.021240	0.000244	0.001709	0.026123	0.00000	0.001709	0.026123
2	0.000000	0.001953	0.053711	0.000244	0.001709	0.041748	0.00000	0.001709	0.037598
3	0.000244	0.001709	0.038574	0.000000	0.001709	0.043701	0.000244	0.001953	0.043945
4	0.000244	0.001709	0.042725	0.000244	0.001709	0.043945	0.000244	0.001709	0.034424
5	0.000244	0.001953	0.037354	0.000244	0.001709	0.042480	0.000000	0.001709	0.039062
6	0.000000	0.001953	0.037842	0.000244	0.001953	0.039062	0.000244	0.001953	0.044434
7	0.000244	0.007080	0.049316	0.000000	0.013428	0.042236	0.000244	0.001709	0.042236
8	0.000244	0.001709	0.044189	0.000244	0.001953	0.044678	0.000244	0.004883	0.032715
9	0.000244	0.001709	0.044922	0.000244	0.002930	0.035156	0.000000	0.001709	0.038330
16	0.000000	0.001953	0.028564	0.000000	0.001709	0.036865	0.000244	0.001953	0.036865
32	0.000244	0.001953	0.042969	0.000244	0.001953	0.031250	0.000244	0.009277	0.031982
64	0.000244	0.020508	0.039551	0.000244	0.001709	0.037598	0.000000	0.001709	0.031494
128	0.000000	0.001953	0.035889	0.000244	0.001953	0.037109	0.000000	0.001709	0.032715
256	0.000244	0.001709	0.037598	0.000000	0.001709	0.036133	0.000244	0.001709	0.038330
512	0.000244	0.001709	0.036621	0.000244	0.001709	0.038574	0.000244	0.001709	0.038818
1024	0.000244	0.001709	0.039307	0.000244	0.001709	0.034912	0.000244	0.001709	0.037842

Execution time vs No. of threads

For static