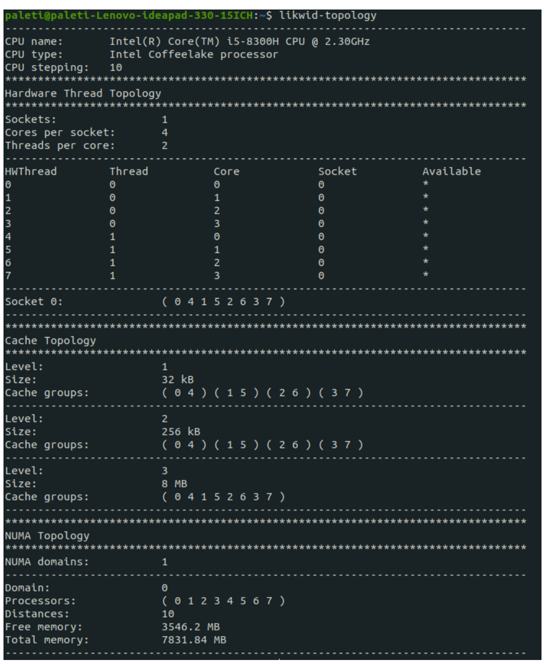
Paleti Krishnasai
CED18I039

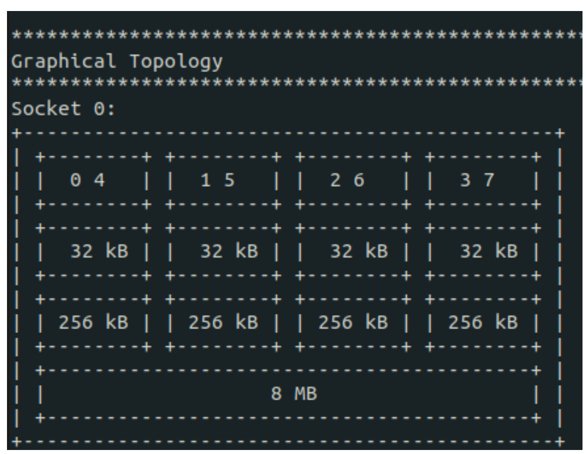
HPC LAB - Block based Matrix Multiplication

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: OpenMP
Problem: Block based matrix multiplication
Date: 2nd September 2021

Hardware Configuration:
PU NAME: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Number of Sockets: 1
Cores per Socket: 4
Threads per core: 2
L1d cache: 128 KiB
L1i cache: 128 KiB
L2 cache: 1 MiB
L3 cache: 8 MiB



```
paleti@paleti-Lenovo-ideapad-330-15ICH:~$ likwid-topology
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
CPU type:       Intel Coffeelake processor
CPU stepping:   10
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                1
Cores per socket:       4
Threads per core:       2
--------------------------------------------------------------------------------
HWThread        Thread          Core            Socket          Available
0               0               0               0               *
1               0               1               0               *
2               0               2               0               *
3               0               3               0               *
4               1               0               0               *
5               1               1               0               *
6               1               2               0               *
7               1               3               0               *
--------------------------------------------------------------------------------
Socket 0:               ( 0 4 1 5 2 6 3 7 )
--------------------------------------------------------------------------------
********************************************************************************
Cache Topology
********************************************************************************
Level:                  1
Size:                   32 kB
Cache groups:           ( 0 4 ) ( 1 5 ) ( 2 6 ) ( 3 7 )
--------------------------------------------------------------------------------
Level:                  2
Size:                   256 kB
Cache groups:           ( 0 4 ) ( 1 5 ) ( 2 6 ) ( 3 7 )
--------------------------------------------------------------------------------
Level:                  3
Size:                   8 MB
Cache groups:           ( 0 4 1 5 2 6 3 7 )
--------------------------------------------------------------------------------
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:           1
--------------------------------------------------------------------------------
Domain:                 0
Processors:             ( 0 1 2 3 4 5 6 7 )
Distances:              10
Free memory:            3546.2 MB
Total memory:           7831.84 MB
--------------------------------------------------------------------------------
```



```
*************************************************
Graphical Topology
*************************************************
Socket 0:
+---------------------------------------------+
| +--------+ +--------+ +--------+ +--------+ |
| |  0 4   | |  1 5   | |  2 6   | |  3 7   | |
| +--------+ +--------+ +--------+ +--------+ |
| +--------+ +--------+ +--------+ +--------+ |
| |  32 kB | |  32 kB | |  32 kB | |  32 kB | |
| +--------+ +--------+ +--------+ +--------+ |
| +--------+ +--------+ +--------+ +--------+ |
| | 256 kB | | 256 kB | | 256 kB | | 256 kB | |
| +--------+ +--------+ +--------+ +--------+ |
| +-----------------------------------------+ |
| |                  8 MB                   | |
| +-----------------------------------------+ |
+---------------------------------------------+
```

Paleti Krishnasai
CED18I039

Serial Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <bits/stdc++.h>

#define N 1000

using namespace std;

double a[N][N], b[N][N], c[N][N];
int main()
{
   srand(time(0));

   for(int i=0; i<N; i+=1)
   {
       for(int j=0; j<N; j+=1)
       {
           a[i][j] = rand();
           b[i][j] = rand();
       }
   }
   int blocks;
   for(int i=1;i<10;i++)
   {
       blocks = pow(2,i);

       float startTime, endTime, execTime;

       startTime = omp_get_wtime();

       for(int jj=0; jj<N; jj+=blocks) {
           for(int kk=0; kk<N; kk+=blocks) {
               for(int i=0; i<N; i+=1) {
                   for(int j=jj; j<min(jj+blocks, N); j+=1) {
                       double r = 0;
                       for(int k=kk; k<min(kk+blocks, N); k+=1) {
                           r += a[i][k] * b[k][j];
                       }
```

```
                        c[i][j] += r;
                    }
                }
            }
        }

        endTime = omp_get_wtime();
        execTime = endTime - startTime;
        cout << "Exec time for " << blocks << " blocks is: " << execTime <<
endl;
    }
    return 0;
}
```

Parallel Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <bits/stdc++.h>

#define N 1000

using namespace std;

double a[N][N], b[N][N], c[N][N];
int main()
{
    srand(time(0));

    for(int i=0; i<N; i+=1)
    {
        for(int j=0; j<N; j+=1)
        {
            a[i][j] = rand();
            b[i][j] = rand();
        }
    }
    int blocks;
    for(int i=1;i<10;i++)
    {
```

```
        blocks = pow(2,i);

        float startTime, endTime, execTime;

        startTime = omp_get_wtime();

        #pragma omp for
        for(int jj=0; jj<N; jj+=blocks) {
            for(int kk=0; kk<N; kk+=blocks) {
                for(int i=0; i<N; i+=1) {
                    for(int j=jj; j<min(jj+blocks, N); j+=1) {
                        double r = 0;
                        for(int k=kk; k<min(kk+blocks, N); k+=1) {
                            r += a[i][k] * b[k][j];
                        }
                        c[i][j] += r;
                    }
                }
            }
        }

        endTime = omp_get_wtime();
        execTime = endTime - startTime;
        cout << "Exec time for " << blocks << " blocks is: " << execTime <<
endl;
    }
    return 0;
}
```

Compilation and Execution:

To enable OpenMP environment, use *-fopenmp* flag while compiling using gcc.
-O0 flag is used to disable compiler optimizations.

gcc -O0 -fopenmp Block_Matrix_Mul.c

Then,
export OMP_NUM_THREADS= no of threads for parallel execution.

./a.out

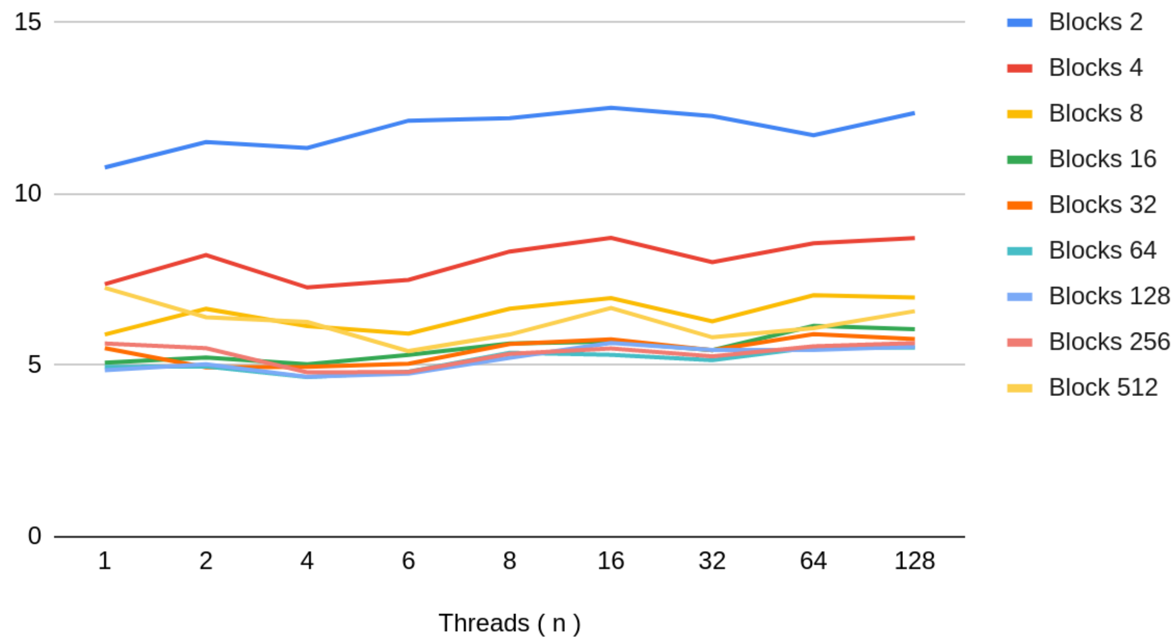
Paleti Krishnasai
CED18I039

Observations :

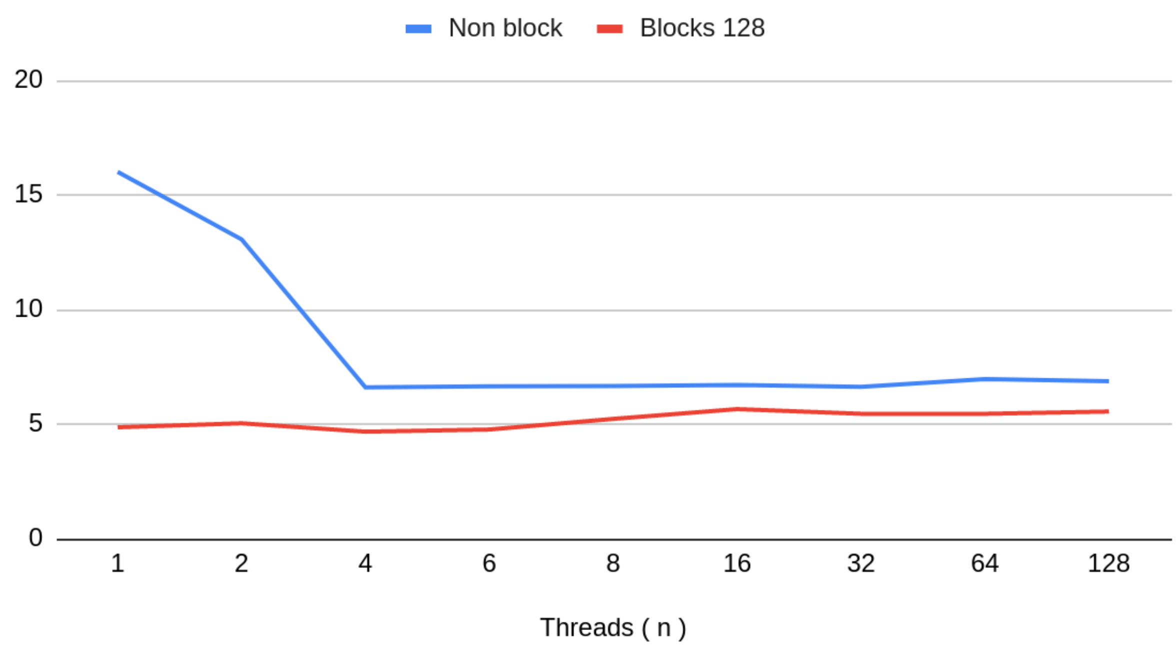
Threads (n)	Non block	Blocks 2	Blocks 4	Blocks 8	Blocks 16	Blocks 32	Blocks 64	Blocks 128	Blocks 256	Block 512	min
1	16.001953	10.7773	7.37012	5.90234	5.07715	5.50293	4.95801	4.8584	5.63965	7.26562	4.8584
2	13.056641	11.5186	8.2207	6.65039	5.22949	4.94629	4.96973	5.0332	5.50293	6.40527	4.94629
4	6.598633	11.3457	7.27441	6.14453	5.03223	4.95508	4.65918	4.65699	4.79688	6.26953	4.65699
6	6.643555	12.1416	7.49805	5.93066	5.30371	5.0498	4.81641	4.76172	4.80957	5.41797	4.76172
8	6.657227	12.2148	8.32324	6.65234	5.63867	5.62695	5.36914	5.22559	5.33887	5.90332	5.22559
16	6.702148	12.5176	8.7207	6.96289	5.69629	5.76465	5.30957	5.65234	5.49414	6.67383	5.30957
32	6.621094	12.2734	8.01074	6.28418	5.44141	5.44629	5.15332	5.44824	5.26562	5.82031	5.15332
64	6.96582	11.7178	8.56543	7.05078	6.15625	5.91113	5.5293	5.44727	5.55762	6.08887	5.44727
128	6.86622	12.3652	8.71777	6.98438	6.05664	5.77246	5.51758	5.54688	5.65234	6.58398	5.51758
	global min	4.65699									

Paleti Krishnasai
CED18I039

Blocking



Non block and Blocks 128



Paleti Krishnasai
CED18I039

Inference:

(Note: Execution time, graph, and inference will be based on hardware configuration)

- The runtime time follows a trend of decreasing until 6 threads for most blocks, after which it increases and flatlines.
- The optimal number of blocks is 128 since it has the maximum number of local minimum runtime of every row(among all other blocks).
- Thus, for all values of threads, it would give the most number of minimum runtimes when compared to other blocking values.
- From the graphs, it is observed that 128 blocks and 4 threads are optimum.