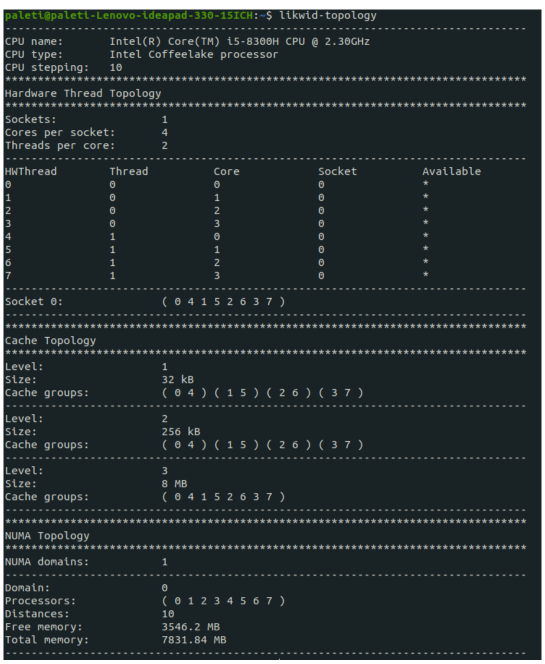
Paleti Krishnasai
CED18I039

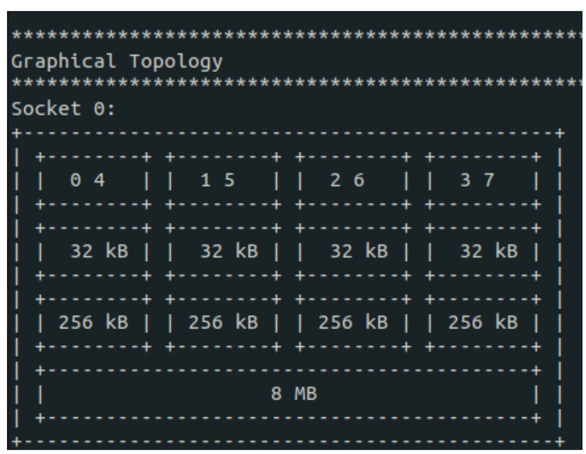
HPC LAB - Vector dot product

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: OpenMP
Problem: Vector dot product
Date: 26th August 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 <time.h>

#define n 200000

int main()
{
   double a[n],b[n], sum=0.0,random_a,random_b;
   float startTime, endTime,execTime;
   int i;
   srand(time(0));
   startTime = omp_get_wtime();
   {
       for(i=0;i<n;i++)
       {   random_a = rand();
           random_b = rand();
           a[i] = i * random_a;
           b[i] = i * random_b;
           for(int j=0;j<n;j++)
           sum = sum + a[i] * b[i];
       }
   }
   endTime = omp_get_wtime();

   execTime = endTime - startTime;
   printf("%f \n",execTime);
   return(0);
}
```

Paleti Krishnasai
CED18I039

Paralle Code : [REDUCTION]

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

#define n 200000

int main()
{
   double a[n],b[n], sum=0.0,random_a,random_b;
   float startTime, endTime,execTime;
   int i;
   srand(time(0));
   startTime = omp_get_wtime();
   #pragma omp parallel private (i) shared (a,b) reduction(+:sum)
   {
       #pragma omp for
       for(i=0;i<n;i++)
       {   random_a = rand();
           random_b = rand();
           a[i] = i * random_a;
           b[i] = i * random_b;
           for(int j=0;j<n;j++)
           sum = sum + a[i] * b[i];
       }
   }
   endTime = omp_get_wtime();

   execTime = endTime - startTime;
   printf("%f \n",execTime);
   return(0);
}
```

Paleti Krishnasai
CED18I039

Compilation and Execution: [REDUCTION]

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

gcc -O0 -fopenmp Vector_dot_product_mp_reduction.c

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

./a.out

Observations : [REDUCTION]

Threads (n)	Runtime	Speedup (s)	Parallelization Fraction
1	168.352539	1	
2	85.917969	1.959456688	0.9793089013
4	47.551758	3.540406203	0.9567287132
6	39.470703	4.265253117	0.9186567908
8	39.235352	4.290837992	0.8765089039
10	37.852539	4.447589077	0.8612878716
12	38.336914	4.391395171	0.8424893864
16	35.928711	4.68573835	0.8390255593
20	38.045898	4.42498529	0.8147479455
32	38.611328	4.360185151	0.7955116813
64	35.791016	4.703765297	0.7999028405
128	37.900391	4.441973673	0.7809762243

Speed up can be found using the following formula,
S(n)=T(1)/T(n)
where, **S(n)** = Speedup for thread count ‘n’
T(1) = Execution Time for Thread count ‘1’ (serial code)
T(n) = Execution Time for Thread count ‘n’ (serial code)

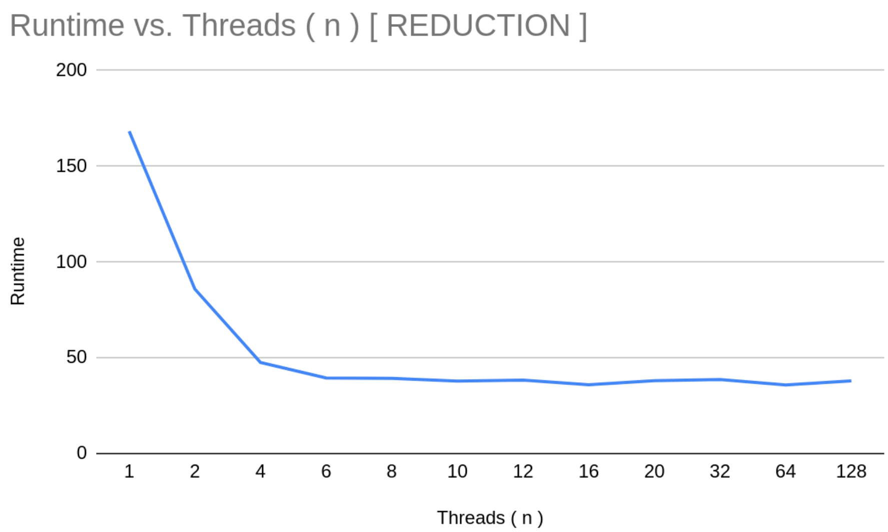
Parallelization Fraction can be found using the following formula,
S(n)=1/((1 - p) + p/n)
where, **S(n)** = Speedup for thread count ‘n’
n = Number of threads
p = Parallelization fraction

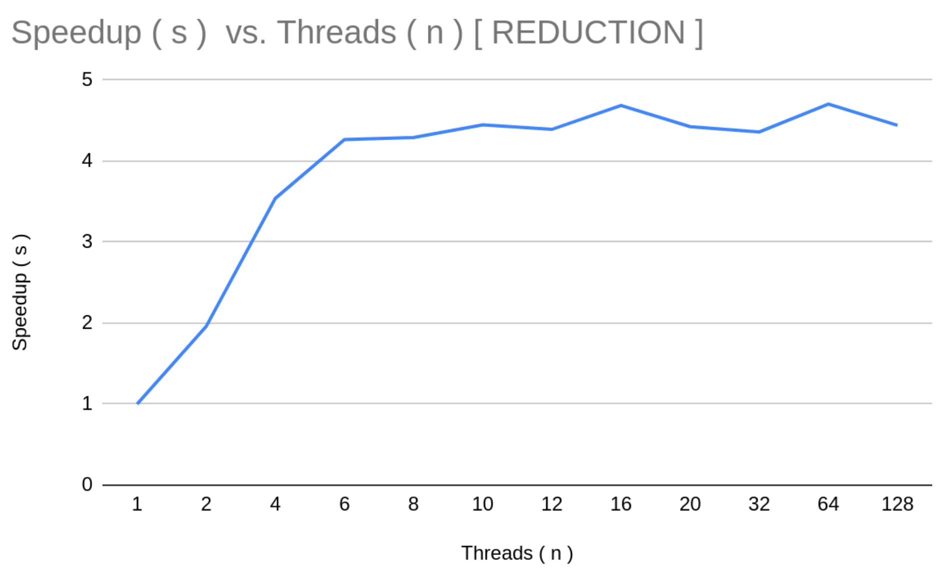
Paleti Krishnasai
CED18I039

Assumption:

Following extra for loop is added to increase the number of operations in the parallel region to visualize the effect of multi-threading in the vector dot product.

for(int j=0;j<n;j++) , where n = 200000





Paleti Krishnasai
CED18I039

Parallel Code : [CRITICAL SECTION]

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

#define n 200000

int main()
{
   double a[n],b[n], privatesum, sum=0.0,random_a,random_b;
   float startTime, endTime,execTime;
   int i;
   srand(time(0));
   startTime = omp_get_wtime();
   #pragma omp parallel private (i,privatesum) shared (a,b, sum)
   {
       privatesum=0;
       #pragma omp for
       for(i=0;i<n;i++)
       {   random_a = rand();
           random_b = rand();
           a[i] = i * random_a;
           b[i] = i * random_b;
           for(int j=0;j<n;j++)
           privatesum = privatesum + a[i] * b[i];
       }
       #pragma omp critical
       {
           sum = sum + privatesum;
       }
   }
   endTime = omp_get_wtime();

   execTime = endTime - startTime;
   printf("%f \n",execTime);
   return(0);
}
```

Paleti Krishnasai
CED18I039

Compilation and Execution: [CRITICAL SECTION **]**

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

gcc -O0 -fopenmp Vector_dot_product_mp_CS.c

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

./a.out

Observations : [CRITICAL SECTION **]**

Threads (n)	Runtime	Speedup (s)	Parallelization Fraction
1	168.984375	1	
2	84.609375	1.997229917	0.9986130374
4	45.898438	3.681702088	0.9711819964
6	41.824219	4.040347412	0.9029958373
8	40.910156	4.130621624	0.8661779292
10	38.152344	4.42920034	0.860250679
12	38.195312	4.424217689	0.8443323699
16	35.628906	4.742901031	0.8417691494
20	37.048828	4.561126063	0.8218483109
32	38.923828	4.341412027	0.7944879431
64	37.830078	4.46693171	0.7884522533
128	39.447266	4.283804485	0.7725985686

Speed up can be found using the following formula,
S(n)=T(1)/T(n)
where, **S(n)** = Speedup for thread count ‘n’
T(1) = Execution Time for Thread count ‘1’ (serial code)
T(n) = Execution Time for Thread count ‘n’ (serial code)

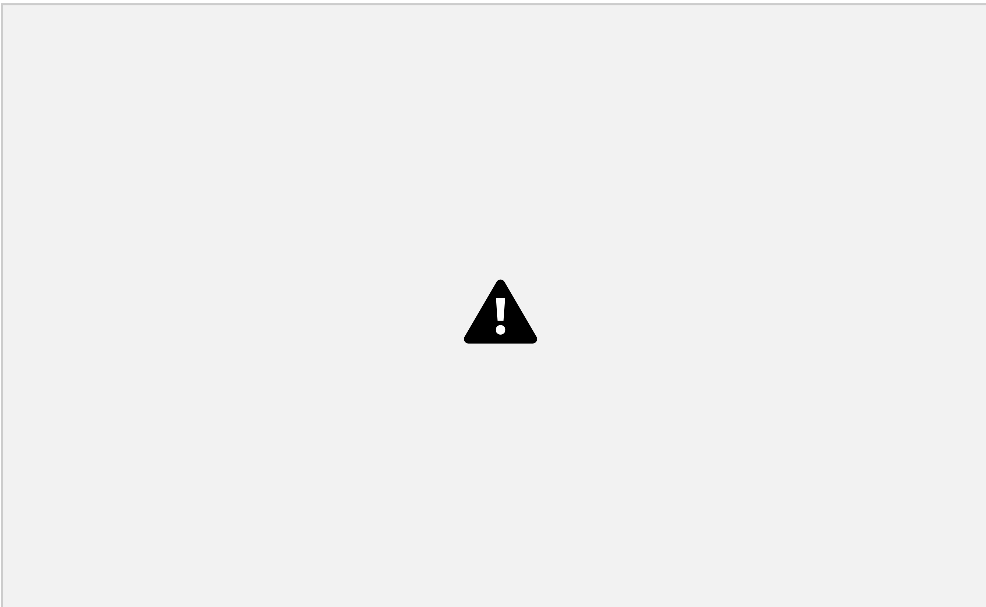
Parallelization Fraction can be found using the following formula,
S(n)=1/((1 - p) + p/n)
where, **S(n)** = Speedup for thread count ‘n’
n = Number of threads
p = Parallelization fraction

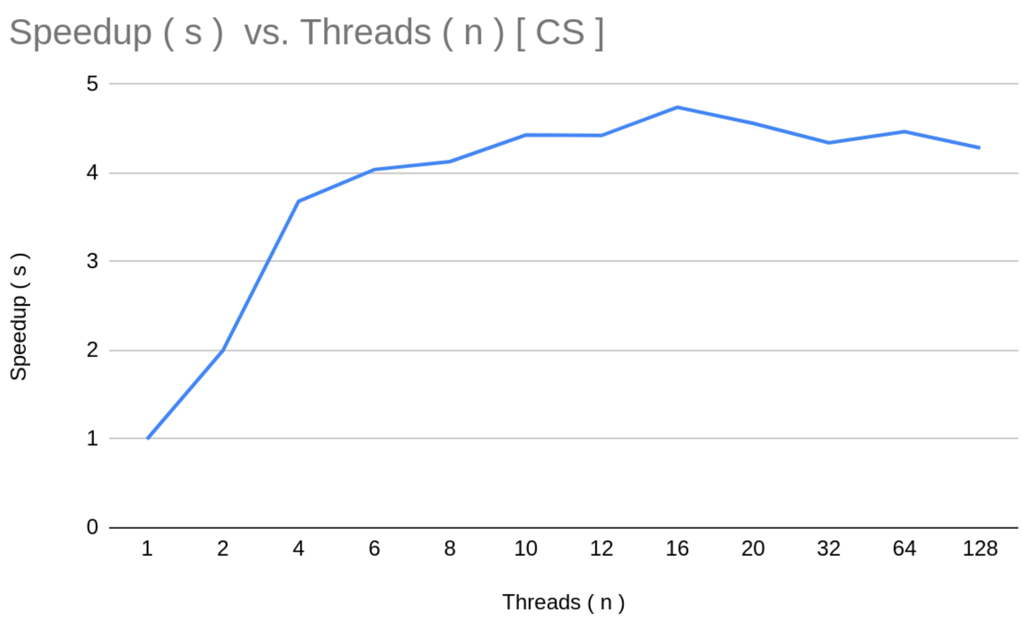
Paleti Krishnasai
CED18I039

Assumption:

Following extra for loop is added to increase the number of operations in the parallel region to visualize the effect of multi-threading in the vector dot product.

for(int j=0;j<n;j++) , where n = 200000





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

- At thread count 16 maximum speedup is observed in both methods with small fluctuations till 128 .
- The Runtime is least at thread count 16 for both methods.