HPC LAB - Matrix Addition CUDA

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: CUDA (collab)
Problem: Matrix Addition

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                   | |
| +-----------------------------------------+ |
+---------------------------------------------+
```

CUDA code:

```
//Mat Add
%%cu
#include <stdio.h>
#include <stdlib.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<unistd.h>
//#define Threads 2
#define n_size 100000
__global__ void mul(float *a, float *b, float *c) {
int index=threadIdx.x+blockIdx.x*blockDim.x;
c[index]=a[index]*b[index];
}
void random_init(float a[],int ch)
{
  srand(time(NULL));
  if(ch==0)
  {
      for(int i=0;i<n_size;i++)
      {
          a[i]=((float)rand()/(float)(RAND_MAX)) * 5.0;
      }
  }
  else
  {
      for(int i=0;i<n_size;i++)
      {
            a[i]=(i+1);
      }
  }
 }
int main() {
 float a[n_size], b[n_size],c[n_size];
cudaEvent_t start, end;
// host copies of variables a, b & c
float *d_a, *d_b, *d_c;
// device copies of variables a, b & c
```

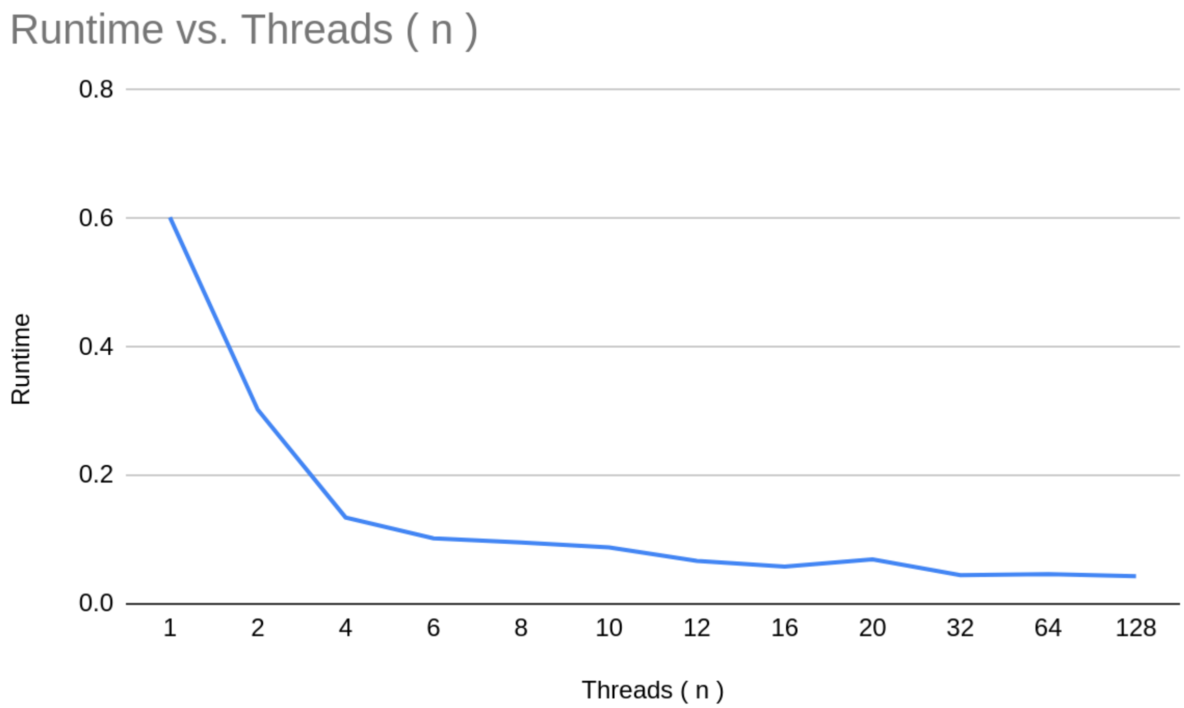
```
int size = n_size*sizeof(float);
// Allocate space for device copies of a, b, c
cudaMalloc((void **)&d_a, size);
cudaMalloc((void **)&d_b, size);
cudaMalloc((void **)&d_c, size);
// Create Event for time
cudaEventCreate(&start);
cudaEventCreate(&end);
// Setup input values
random_init(a,0);
random_init(b,0);
// Copy inputs to device
cudaMemcpy(d_a, &a, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_b, &b, size, cudaMemcpyHostToDevice);
int Thread[]={1,2,4,6,8,10,12,16,20,32,64,128,150};
int thread_arr_size=13;
for(int i=0;i<thread_arr_size;i++)
{       sleep(1);
      int Threads=Thread[i];
      cudaEventRecord(start);
      // Launch add() kernel on GPU
      mul<<<n_size/Threads,Threads>>>(d_a, d_b, d_c);
      cudaEventRecord(end);
      cudaEventSynchronize(end);
      float time = 0;
      cudaEventElapsedTime(&time, start, end);
      // Copy result back to host
      cudaError err = cudaMemcpy(&c, d_c, size, cudaMemcpyDeviceToHost);
      if(err!=cudaSuccess) {
          printf("CUDA error copying to Host: %s\n",
cudaGetErrorString(err));
      }
          int flag=0;
      for(int i=0;i<n_size;i++)
      {   //printf("Result[%d]=%f\n",i+1,c[i]);
          if(c[i]!=(a[i]*b[i]))
          {
              flag=1;
          break;
          }
```

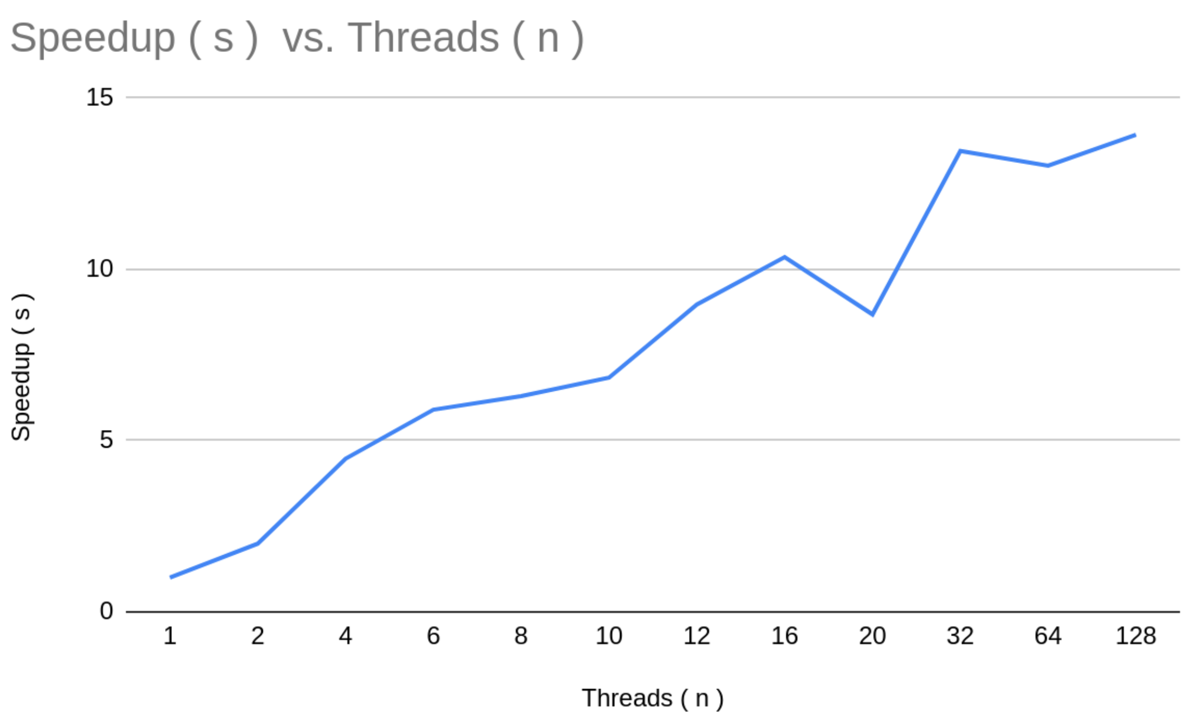
```
      }
      if(flag==0)
      {
          printf("Program Executed as Expected\n");
          printf("Time Taken by the program for %d 
Threads=%f\n",Threads,time);
          //printf("%f\n",time);
      }
      else
      {
          printf("Vector Addition hasnt been done properly,Mismatch in 
Values!!!\n");
      }

}
// Cleanup
cudaFree(d_a);
cudaFree(d_b);
cudaFree(d_c);
return 0;
}
```

Observations:
The function add() is executed for an array of size ‘N’ on GPU for “N/T” times for ‘T’ Threads.

Blocks	Threads (n)	Runtime	Speedup (s)	Parallelization Fraction
N	1	0.60192	1	
N/2	2	0.302368	1.990686845	0.9953216374
N/4	4	0.13456	4.473246136	1.03526493
N/6	6	0.101984	5.902102291	0.9966826156
N/8	8	0.095584	6.297288249	0.9613731298
N/10	10	0.088032	6.837513631	0.948608896
N/12	12	0.067072	8.974236641	0.9693489923
N/16	16	0.058112	10.35792952	0.9636859826
N/20	20	0.069312	8.684210526	0.9314194577
N/32	32	0.044736	13.45493562	0.9555384061
N/64	64	0.046208	13.02631579	0.9378868046
N/128	128	0.0432	13.93333333	0.9355385601

Graphical Inference:





Inference:

From the Graph, it is observed that the speedup increases steadily in an up and down manner till 16 threads, and the highest Speedup is for 128 threads.

Runtime Decreases drastically from 1 to 4 threads and steadily decreases from then on till 128, with a minor rise at threads = 20.