Paleti Krishnasai
CED18I039

HPC LAB - Dot Product MPI

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: MPI
Problem:Dot Product

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

No of nodes: 12 (4 for each as written in the machine file).

Paleti Krishnasai
CED18I039

Serial Code:

```c
#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#include <time.h>
#define SIZE 1200000

int main(int argc, char *argv[])

{

    srand(time(0));

    double start,end;

    MPI_Init(&argc,&argv);

    start=MPI_Wtime();

    int i = 0, n = 120000;

    double a[n],b[n], sum = 0;

    for(i=0;i<120000;i++)

    {

        a[i] = (i+1)*4.37 ;

        b[i] = (i+1)*5.36 ;

    }

    for(i=0;i<120000;i++)
```

```
    {

        sum += a[i]*b[i];

    }

    end=MPI_Wtime();

    printf("\nTime= %f",end-start);

    return 0;

}
```

Parallel Code:

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int myid, numprocs;
    double start,end;
    int  namelen;
    int ARRAY_SIZE = 10000;
    int vector1[ARRAY_SIZE+50];
    int vector2[ARRAY_SIZE+50];
    long int vector3[ARRAY_SIZE],part_sum=0,sum=0;
    int i, j;
    int s, s0;
    //double totalTime;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    start=MPI_Wtime();
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
MPI_Get_processor_name(processor_name,&namelen);
    //fprintf(stderr,"Process %d on %s\n", myid, processor_name);
    fflush(stderr);
```

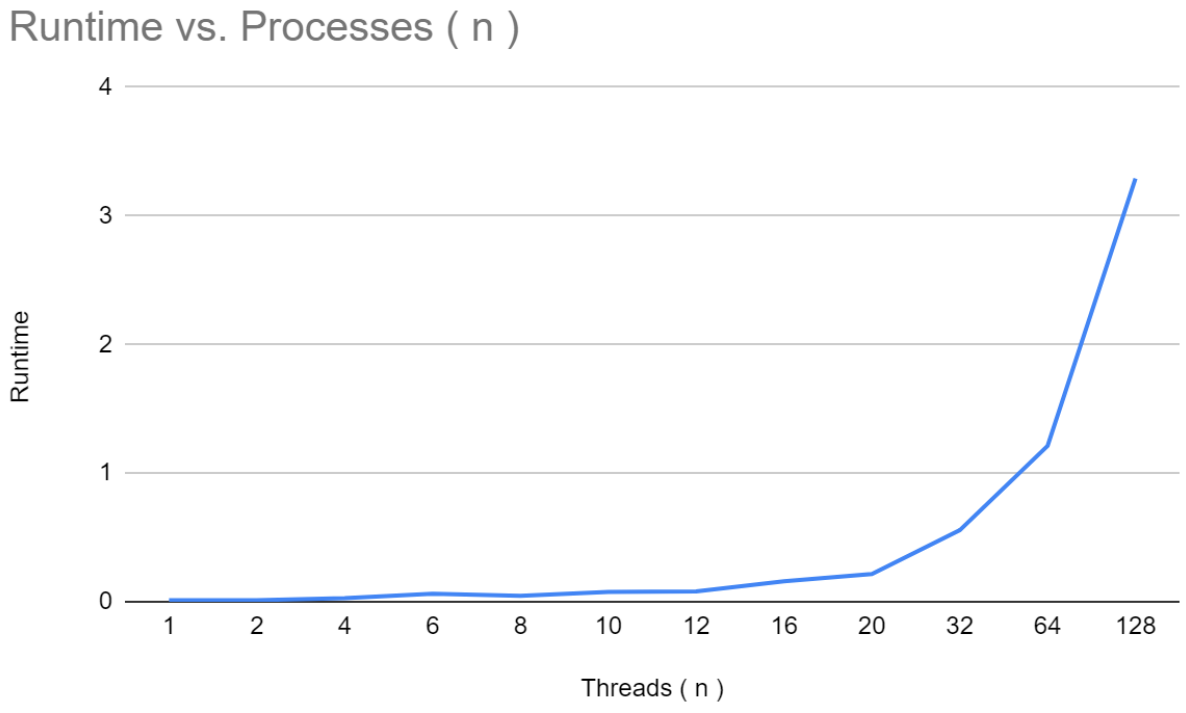
```
 // Vector 1 Reading
    for (i=0; i < ARRAY_SIZE; i++)
    vector1[i] = (i+1)*4.37;
 // Vector 2 Reading
    for (i=0; i < ARRAY_SIZE; i++)
    vector2[i] = (i+1)*5.36;

if(myid == 0)
{
    MPI_Bcast(&ARRAY_SIZE, 1, MPI_INT, 0, MPI_COMM_WORLD);
    s = (int) floor(ARRAY_SIZE/numprocs);
    s0 = ARRAY_SIZE%numprocs;
    int vector1Receive[s];
    int vector2Receive[s];
    long int vector3Receive[s];
    if (s0 != 0)
    {
        s = s + 1;
        for(i=0; i < ((s * numprocs) - ARRAY_SIZE); i++)
        {
            vector1[ARRAY_SIZE + i] = 0;
            vector2[ARRAY_SIZE + i] = 0;
        }
    }
    MPI_Bcast(&s, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(vector1, s, MPI_INT, vector1Receive, s, MPI_INT, 0,
MPI_COMM_WORLD);
    MPI_Scatter(vector2, s, MPI_INT, vector2Receive, s, MPI_INT, 0,
MPI_COMM_WORLD);
    for(i=0; i<s; i++)
    {
        vector3Receive[i] = vector1Receive[i] * vector2Receive[i];
        part_sum += vector3Receive[i];
    }
    MPI_Gather(vector3Receive, s, MPI_LONG, vector3, s, MPI_LONG, 0,
MPI_COMM_WORLD);

    //for(i=0; i<ARRAY_SIZE; i++)
        //printf("%ld\n", vector3[i]);
}
```

```
else
{
    MPI_Bcast(&ARRAY_SIZE, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&s, 1, MPI_INT, 0, MPI_COMM_WORLD);
    int vector1Receive[s];
    int vector2Receive[s];
    long int vector3Receive[s];
    MPI_Scatter(vector1, s, MPI_INT, vector1Receive, s, MPI_INT, 0,
MPI_COMM_WORLD);
    MPI_Scatter(vector2, s, MPI_INT, vector2Receive, s, MPI_INT, 0,
MPI_COMM_WORLD);
    for(i=0; i<s; i++)
    {
        vector3Receive[i] = vector1Receive[i] * vector2Receive[i];
        part_sum += vector3Receive[i];
    }
    MPI_Gather(vector3Receive, s, MPI_LONG, vector3, s, MPI_LONG, 0,
MPI_COMM_WORLD);

}
MPI_Reduce(&part_sum, &sum, 1, MPI_LONG, MPI_SUM, 0, MPI_COMM_WORLD);
    if (myid == 0)
{
        printf("\ndot product = %ld \n", sum);

    end=MPI_Wtime();
        printf("\nTime_hello= %f",end-start);
}
MPI_Finalize();
}
```

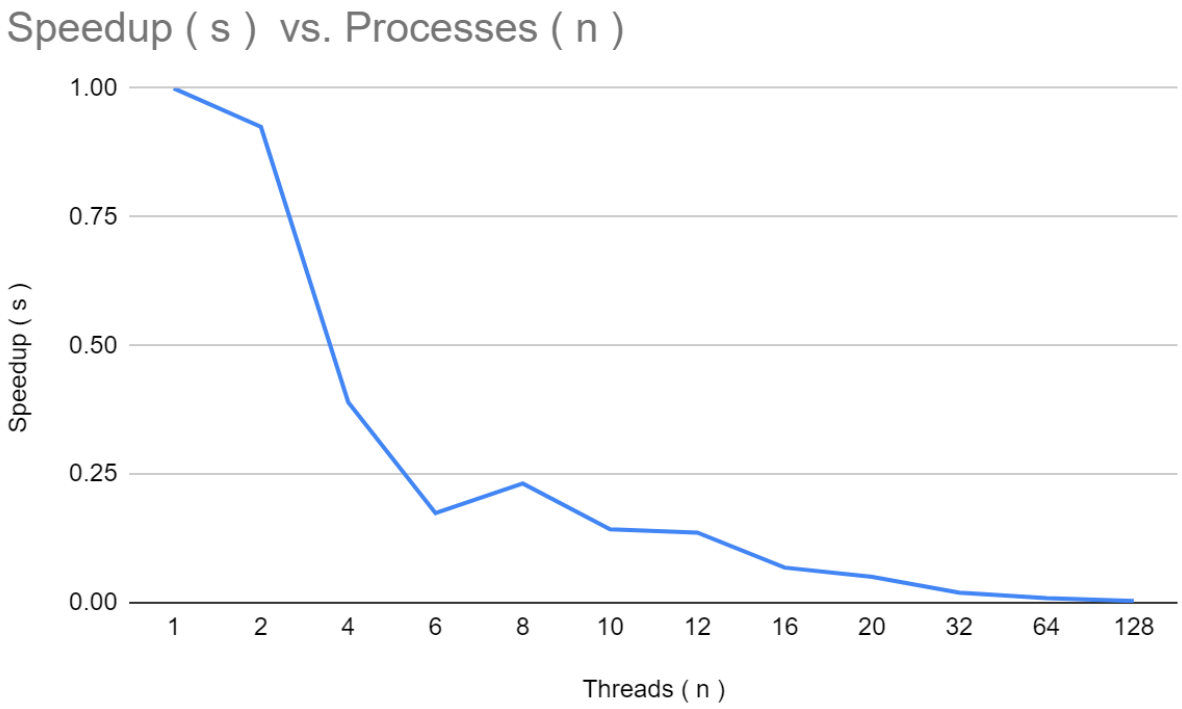
Paleti Krishnasai
CED18I039

Observations :

Processes (n)	Runtime	Speedup (s)
1	0.010808	1
2	0.011686	0.9248673627
4	0.02773	0.3897583844
6	0.062029	0.1742410808
8	0.046634	0.2317622336
10	0.07572	0.1427363973
12	0.079346	0.1362135457
16	0.158487	0.06819486772
20	0.214793	0.05031821335
32	0.555407	0.0194596035
64	1.212303	0.008915262933
128	3.290684	0.003284423542



Paleti Krishnasai
CED18I039



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

- Since MPI is a distributed memory architecture, the communication overhead between nodes causes the parallel code to run slower compared to serial code (running in 1 node or only in master).