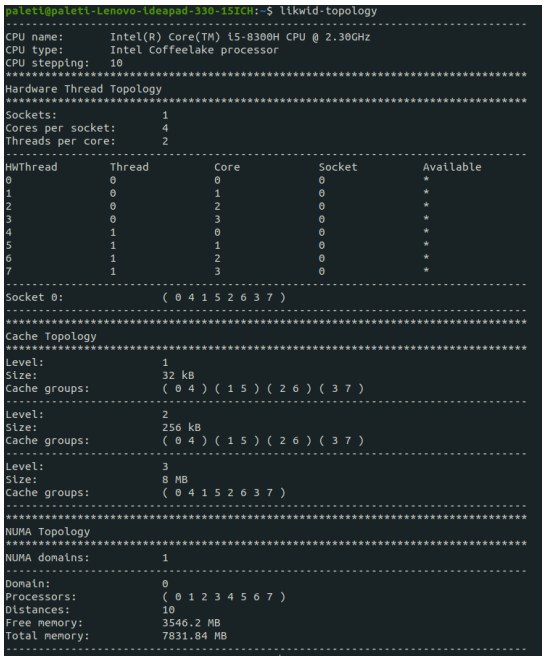
Paleti Krishnasai
CED18I039

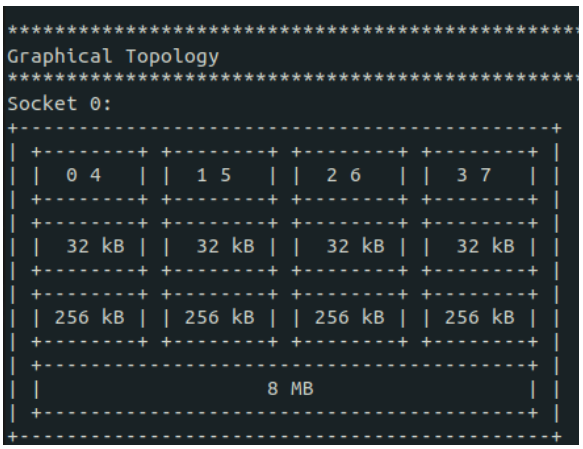
HPC LAB - Matrix Addition MPI

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: MPI
Problem: Matrix Addition
Date: 21-10-2021

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

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

Paleti Krishnasai
CED18I039

Serial Code:

```
#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>



#define ROWS 20                 /* number of rows  */

#define COLUMNS 20                 /* number of columns */

#define MASTER 0                /* taskid of first task */

#define FROM_MASTER 1           /* setting a message type */

#define FROM_WORKER 2           /* setting a message type */



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

{

    double start,end;

    MPI_Init(&argc,&argv);

    start=MPI_Wtime();

    int i,j,k;

    long double a[ROWS][COLUMNS],           /* matrix A to be Added */

    b[COLUMNS][COLUMNS],           /* matrix B to be Added */

    c[ROWS][COLUMNS];

    //printf("Initializing arrays...\n");
```

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

        for (j=0; j<COLUMNS; j++)

            a[i][j]= (i+j)*1.6584;

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

        for (j=0; j<COLUMNS; j++)

            b[i][j]= (i+j)*4.2367;

    for (k=0; k<COLUMNS; k++)            // Matrix Addition

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

         {

            c[i][k] = a[i][k] + b[i][k];

         }

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

      {

         printf("\n");

         for (j=0; j<COLUMNS; j++)

            printf("%6.2Lf   ", c[i][j]);}
printf("\nFinished.\n");

    end=MPI_Wtime();

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

    return 0;
}
```

Paleti Krishnasai
CED18I039

Parallel Code :

```
#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>



#define ROWS 200                /* number of rows  */

#define COLUMNS 200               /* number of columns */

#define MASTER 0               /* taskid of first task */

#define FROM_MASTER 1          /* setting a message type */

#define FROM_WORKER 2          /* setting a message type */



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

{

int   numtasks,              /* number of tasks in partition */

   taskid,                /* a task identifier */

   numworkers,            /* number of worker tasks */

   source,                /* task id of message source */

   dest,                  /* task id of message destination */

   mtype,                 /* message type */

   rows,                  /* rows of matrix A sent to each worker */

   averow, extra, offset, /* used to determine rows sent to each worker */
```

```
i, j, k, rc;           /* misc */

long double a[ROWS][COLUMNS],           /* matrix A to be Added */

b[COLUMNS][COLUMNS],           /* matrix B to be Added */

c[ROWS][COLUMNS];           /* result matrix C */

MPI_Status status;

double start,end;




MPI_Init(&argc,&argv);

start=MPI_Wtime();

MPI_Comm_rank(MPI_COMM_WORLD,&taskid);

MPI_Comm_size(MPI_COMM_WORLD,&numtasks);

if (numtasks < 2 ) {

   printf("Need at least two MPI tasks. Quitting...\n");

   MPI_Abort(MPI_COMM_WORLD, rc);

   exit(1);

}

numworkers = numtasks-1;
```

```
// ------------------------------------------- master task
-----------------------------------------------------//



   if (taskid == MASTER)

   {

      //printf("mpi_Matrix Addition has started with %d
tasks.\n",numtasks);

      //printf("Initializing arrays...\n");

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

         for (j=0; j<COLUMNS; j++)

            a[i][j]= (i+j)*1.6584;

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

         for (j=0; j<COLUMNS; j++)

            b[i][j]= (i+j)*4.2367;



      /* Send matrix data to the worker tasks */

      averow = ROWS/numworkers;

      extra = ROWS%numworkers;

      offset = 0;

      mtype = FROM_MASTER;

      for (dest=1; dest<=numworkers; dest++)
```

```
      {

         rows = (dest <= extra) ? averow+1 : averow;

         //printf("Sending %d rows to task %d
offset=%d\n",rows,dest,offset);

         MPI_Send(&offset, 1, MPI_INT, dest, mtype, MPI_COMM_WORLD);

         MPI_Send(&rows, 1, MPI_INT, dest, mtype, MPI_COMM_WORLD);

         MPI_Send(&a[offset][0], rows*COLUMNS, MPI_LONG_DOUBLE, dest,
mtype,MPI_COMM_WORLD);

         MPI_Send(&b[offset][0], rows*COLUMNS, MPI_LONG_DOUBLE, dest,
mtype, MPI_COMM_WORLD);

         offset = offset + rows;

      }



//       ------------------- Receive results from worker tasks
------------------------



     mtype = FROM_WORKER;

      for (i=1; i<=numworkers; i++)

      {

         source = i;

         MPI_Recv(&offset, 1, MPI_INT, source, mtype, MPI_COMM_WORLD,
&status);
```

```
         MPI_Recv(&rows, 1, MPI_INT, source, mtype, MPI_COMM_WORLD,
&status);

         MPI_Recv(&c[offset][0], rows*COLUMNS, MPI_LONG_DOUBLE, source,
mtype,

                  MPI_COMM_WORLD, &status);

         printf("Received results from task %d\n",source);

      }



      // Print results




printf("-------------------------------------------------------\n");

      printf("Result Matrix:\n");

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

      {

         printf("\n");

         for (j=0; j<COLUMNS; j++)

            printf("%6.2Lf   ", c[i][j]);

      }


printf("\n-------------------------------------------------------\n");

      printf ("Done.\n");
```

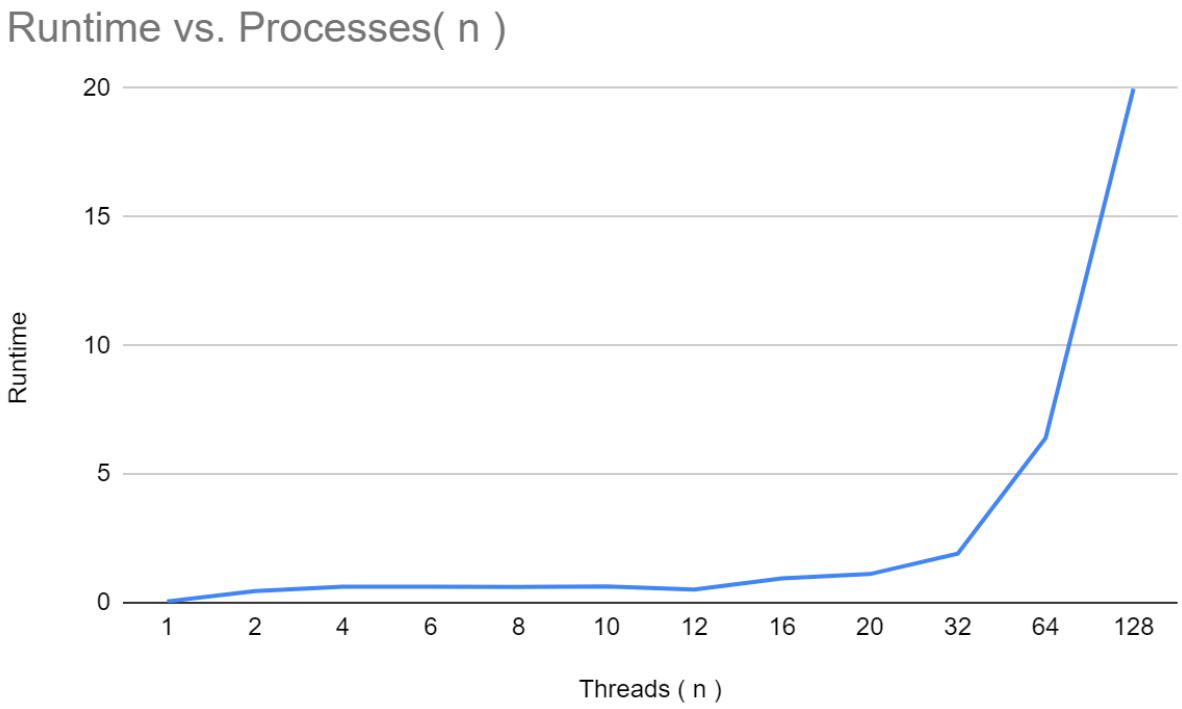
```
      end=MPI_Wtime();

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

   }




// ------------------------------------------------ worker task
-----------------------------------------------------------//



   if (taskid > MASTER)

   {

      mtype = FROM_MASTER;

      MPI_Recv(&offset, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD,
&status);

      MPI_Recv(&rows, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD, &status);

      MPI_Recv(&a, rows*COLUMNS, MPI_LONG_DOUBLE, MASTER, mtype,
MPI_COMM_WORLD, &status);

      MPI_Recv(&b, rows*COLUMNS, MPI_LONG_DOUBLE, MASTER, mtype,
MPI_COMM_WORLD, &status);



      for (k=0; k<COLUMNS; k++)         // Matrix Addition

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

         {
```

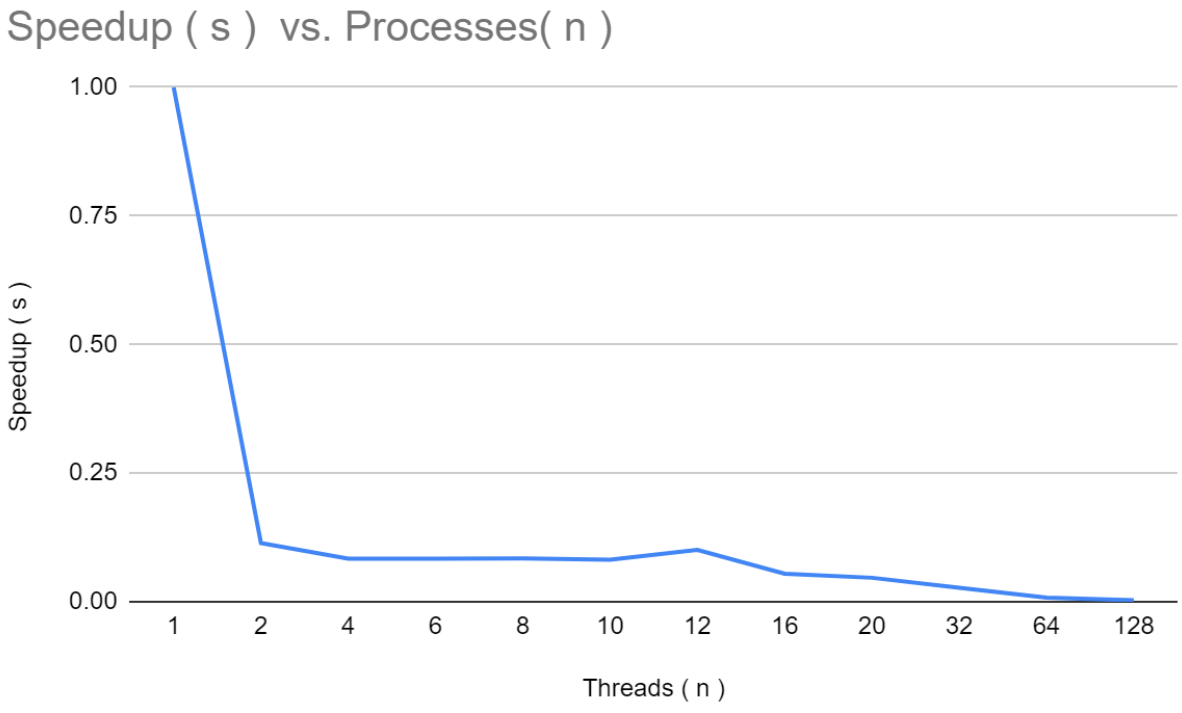
```
            c[i][k] = a[i][k] + b[i][k];

         }

      mtype = FROM_WORKER;

      MPI_Send(&offset, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD);

      MPI_Send(&rows, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD);

      MPI_Send(&c, rows*COLUMNS, MPI_LONG_DOUBLE, MASTER, mtype,
MPI_COMM_WORLD);

   }

   MPI_Finalize();

}
```

Observations:

Processes(n)	Runtime	Speedup (s)
1	0.051765	1
2	0.453605	0.1141191124
4	0.619022	0.08362384536
6	0.619272	0.08359008642
8	0.613912	0.08431990253
10	0.632624	0.08182585548
12	0.514897	0.10053467
16	0.949612	0.05451173743
20	1.112871	0.04651482517
32	1.909993	0.02710219357
64	6.402997	0.008084495432
128	19.987483	0.002589870871

Paleti Krishnasai
CED18I039





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)