Experiment :3

//Aim :. Develop a distributed system, to find sum of N elements in an array by distributing N/n elements to n number of processors MPI or OpenMP. Demonstrate by displaying the intermediate sums calculated at different processors.

[**https://www.open-mpi.org/software/ompi/v4.1/**](https://www.open-mpi.org/software/ompi/v4.1/)

**https://edu.itp.phys.ethz.ch/hs12/programming\_techniques/openmpi.pdf**

**Installation of OPENMPI**

1. Download openmpi-4.1.4.tar.bz2 from http://www.open-mpi.org

2. Goto the terminal (Command prompt)

3. update using

sudo apt-get update

sudo apt install gcc {if not already installed}

4. Goto the directory which contains the downloaded file (download folder)

5. Extract the files using (download folder)

tar -jxf openmpi-4.1.4.tar.bz2

6. The directory openmpi-4.1.4 is created

7. Go into the source directory through terminal

Command prompt > cd /home/it/download/openmpi-1.4.4

7. Configure, compile and install by executing the following commands

Command prompt//home/it/download/openmpi-1.4.4 > ./configure --prefix=$HOME/opt/openmpi

make all

make install

8. Now openmpi folder is created in ‘opt‘ folder of Home directory.

9. Now the folder LP5 can be deleted (optional)

10. Update the PATH and LD\_LIBRARY\_PATH environment variable using

echo "export PATH=\$PATH:\$HOME/it/opt/openmpi/bin" >> $HOME/it.bashrc

echo "export LD\_LIBRARY\_PATH=\$LD\_LIBRARY\_PATH:\$HOME/it/opt/openmpi/lib">>$HOME/it.bashrc

11. Compile the program using

mpicc name of the program

example : mpicc hello1.c

12. Execute the program using

mpirun -np N ./a.out

example : mpirun -np 2 ./a.out

example : mpirun -np 3 ./a.out

example : mpirun -np 4 ./a.out

/\*\*\*\*\*\* The cores 2 and will be in waiting mode … Press Ctrl+z to end the execution \*\*\*\*\*\*\*/

Assignment program: Add 20 numbers in an array using 4 cores

C:\>cd /opt/openmpi/bin

C:\opt/openmpi/bin> edit program\_name.c

Example :

C:\opt/openmpi/bin> edit add.c

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

#include "mpi.h"

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

{

int rank, size;

int num[20]; //N=20, n=4

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

for(int i=0;i<20;i++)

num[i]=i+1;

if(rank == 0){

int s[4];

printf("Distribution at rank %d \n", rank);

for(int i=1;i<4;i++)

MPI\_Send(&num[i\*5], 5, MPI\_INT, i, 1, MPI\_COMM\_WORLD); //N/n i.e. 20/4=5

int sum=0, local\_sum=0;

for(int i=0;i<5;i++)

{

local\_sum=local\_sum+num[i];

}

for(int i=1;i<4;i++)

{

MPI\_Recv(&s[i], 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

printf("local sum at rank %d is %d\n", rank,local\_sum);

sum=local\_sum;

for(int i=1;i<4;i++)

sum=sum+s[i];

printf("final sum = %d\n\n",sum);

}

else

{

int k[5];

MPI\_Recv(k, 5, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

int local\_sum=0;

for(int i=0;i<5;i++)

{

local\_sum=local\_sum+k[i];

}

printf("local sum at rank %d is %d\n", rank, local\_sum);

MPI\_Send(&local\_sum, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}

Distribution at rank 0

local sum at rank 1 is 40

local sum at rank 2 is 65

local sum at rank 3 is 90

local sum at rank 0 is 15

final sum = 210

/\*\*\*\*\*\* students can be asked to take dynamic values for N, n and array \*\*\*\*\*\*\*\*\*\*\*\*/

**Run : Compile the program using**

C:mpicc name of the program

example : opt/openmpi/bin> mpicc add.c

12. Execute the program using

opt/openmpi/bin> mpirun -np N ./a.out

example : opt/openmpi/bin> mpirun -np 2 ./a.out

example : opt/openmpi/bin> mpirun -np 3 ./a.out

example : opt/openmpi/bin> mpirun -np 4 ./a.out

