PADP ASSIGNMENT REPORT

Topic:- MPI Synchronization

Name:- P. Siddarth Shankar

USN:- 1RV15CS107

Subject:- PADP

MPI SYNCHRONIZATION

* Many parallel algorithms require that none process proceeds before all the processes have reached the same state at certain points of a program.
* A synchronization point among processes means that all processes must reach a point in their code before they can all begin executing again.
* MPI has a special function that is dedicated to synchronizing processes. This function is:- MPI\_Barrier(MPI\_Comm communicator)

where:

|  |  |
| --- | --- |
| MPI\_Comm | is an MPI predefined structure for communicators, and |
| comm | is a communicator. |

* The name of the function is quite descriptive - the function forms a barrier, and no processes in the communicator can pass the barrier until all of them call the function
* When MPI\_BARRIER returns, all processes are synchronized at that point

IMPLEMENTATION

In the program given below, the barrier function in the for loop will ensure that all processes are synchronized when passing through the for loop.

#include <stdio.h>

#include <mpi.h>

int main (int nargs, char\*\* args)

{

int size, my\_rank,i;

MPI\_Init (&nargs, &args);

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

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

for (i=0; i<size; i++) {

MPI\_Barrier (MPI\_COMM\_WORLD);

if (i==my\_rank) {

printf("Hello world, I’ve rank %d out of %d procs.\n",

my\_rank, size);

fflush (stdout);

}

}

MPI\_Finalize ();

return 0;

}

REFERENCES

1) Alex Aravind, ‘Barrier Synchronization: Simplified, Generalized, and Solved without Mutual Exclusion’, IEEE, 2018