**PRACTICAL # 07**

**OBJECT:**

Calculating factorial of a number in parallel

**THEORY:**

The traditional and slow algorithm to calculate factorial of number is the iterative or recursive method. For a number N, this method works as follows:

Factorial = N\*(N-1)\*(N-2) \* … \* 1

We can however, easily design a parallel version of this algorithm by easily dividing the number into equal parts and computing individual factorials. Finally the individual factorials can be multiplied together to get the final value.

For example, we need to calculate factorial for 100, and we have two computing units available. We can divide this number and assign to the two computing units as:

f1 = 1\*2\*… 50

f2 = 51\*52\* … 100

Finally the factorial of 100 becomes:

F = f1\*f2

**CODE:**

The program to calculate factorial of a number in parallel is given below:

*#include <mpi.h>*

*#include <stdio.h>*

*#include <string.h>*

*#define BUFSIZE 1*

*#define TEMPBUFSIZE 1*

*#define TAG 0*

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

*{*

*int buff[] = {12};*

*int temp[] = {0};*

*int numprocs, workerProcs, myid, i;*

*int factorial = 0;*

*MPI\_Status stat;*

*/\* MPI programs start with MPI\_Init; all 'N' processes exist thereafter \*/*

*MPI\_Init(&argc, &argv);*

*/\* find out how big the SPMD world is \*/*

*MPI\_Comm\_size(MPI\_COMM\_WORLD, &numprocs);*

*/\* and this processes' rank is \*/*

*MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);*

*workerProcs = numprocs-1;*

*/\* At this point, all programs are running equivalently, the rank*

*distinguishes the roles of the programs in the SPMD model, with*

*rank 0 often used specially...*

*\*/*

*if(myid == 0)*

*{*

*printf("%d: We have %d processors\n", myid, numprocs);*

*//distribute the data to different processes.*

*for(i=1; i<numprocs; i++)*

*{*

*MPI\_Send(buff, BUFSIZE, MPI\_INT, i, TAG, MPI\_COMM\_WORLD);*

*/\* buf - initial address of send buffer (choice)*

*count - number of elements in send buffer (nonnegative integer)*

*datatype - datatype of each send buffer element (handle)*

*dest - rank of destination (integer)*

*tag - message tag (integer)*

*comm - communicator (handle)\*/*

*}*

*//Receive the data from worker processes and calculate overall factorial*

*for(i=1; i<numprocs; i++)*

*{*

*MPI\_Recv(temp, TEMPBUFSIZE, MPI\_INT, i, TAG, MPI\_COMM\_WORLD, &stat);*

*factorial += temp[0];*

*}*

*printf("\n Total Factorial = %d\n", factorial);*

*}*

*else*

*{*

*/\* receive from process with rank 0: \*/*

*int src = 0, k, sum=1;*

*int buffer[1];*

*MPI\_Recv(buff, BUFSIZE, MPI\_INT, src/\*0\*/, TAG, MPI\_COMM\_WORLD, &stat);*

*printf("Processor %d reporting for duty. ", myid);*

*int myPartialNum = buff[0]/workerProcs;*

*for(k=((myid-1)\*myPartialNum)+1; k<=myid\*myPartialNum; k++){*

*printf("k= %d\n", k);*

*//printf("arr[i] = %d ", buff[BUFSIZE/workerProcs\*(myid-1)+ k]);*

*sum\*=k;*

*}*

*printf("Partial factorial from processor %d = %d \n", myid, sum);*

*temp[0] = sum;*

*//MPI\_Send(buff, BUFSIZE, MPI\_INT, 0, TAG, MPI\_COMM\_WORLD);*

*MPI\_Send(temp, TEMPBUFSIZE, MPI\_INT, 0, TAG, MPI\_COMM\_WORLD);*

*}*

*/\* MPI programs end with MPI Finalize; this is a weak synchronization point \*/*

*MPI\_Finalize();*

*return 0;*

*}*

In the program above, the magic happens at the lines:

int myPartialNum = buff[0]/workerProcs;

for(k=((myid-1)\*myPartialNum)+1; k<=myid\*myPartialNum; k++){

The myPartialNum variable is the number of elements for which a process will calculate the factorial.

Next, the for loop makes sure that each process gets its particular range of numbers to calculate factorial for.

**ACTIVITY**

Execute the program in OpenMPI and observe the output.

**REVIEW QUESTIONS**

1. How do you divide a factorial problem in parallel?

1. What problem can occur if you calculate factorial of a very large number?