Parallel Computation

MPI vs OpenMP: A Short Introduction Plus Comparison (Parallel Computing)

MPI Vs OpenMP : A Short Introduction Plus Comparison

Here i will talk briefly about OpenMP and MPI (OpenMPI ,MPICH, HP-MPI) for parallel programming or parallel computing . (*Many a times one can easily confuse OpenMP with OpenMPI or vice versa. OpenMPI is a particular API of MPI whereas OpenMP is shared memory standard available with compiler*). This is intended for user who are new to parallel programming or parallel computation and is thinking of using OpenMP or MPI for their applications or learning. This will introduce them to with differences as well advantages of both. I will not go in development history of these two and changes that they have gone but will focus on differences at present form i.e with OpenMP 4 and MPI 3.

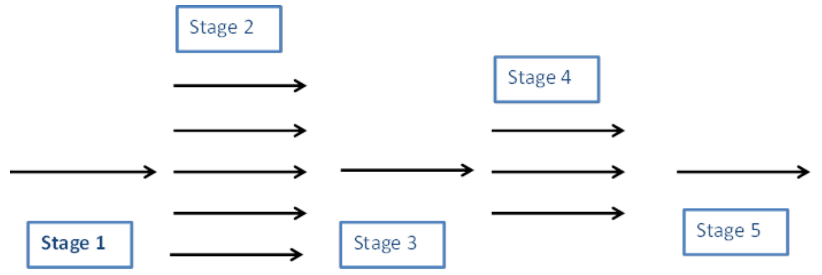
A Brief about MPI & OpenMP

1. MPI stands for *Message Passing Interface* . These are available as API(Application programming interface) or in library form for C,C++ and FORTRAN.

- Different MPI's API are available in market i.e OpenMPI,MPICH,HP-MPI.Intel MPI, etc. Whereas many are freely available like OpenMPI, MPICH etc , other like Intel MPI comes with license i.e you need to pay for it .
- One can use any one of above to parallelize programs . MPI standards maintain that all of these APIs provided by different vendors or groups follow similar standards, so all functions or subroutines in all different MPI API follow similar functionality as well arguments.

- The difference lies in implementation that can make some MPIs API to be more efficient than other. Many commercial CFD-Packages gives user option to select between different MPI API. However HP-MPI as well Intel MPIs are considered to be more efficient in performance.
- When MPI was developed, it was aimed at distributed memory system but now focus is both on distributed as well shared memory system. However it does not mean that with MPI , one cannot run program on shared memory system, it just that earlier, we could not take advantage of shared memory but now we can with latest MPI 3

2. OpenMP stand for *Open Multiprocessing* . OpenMP is basically an add on in compiler. It is available in gcc (gnu compiler) , Intel compiler and with other compilers.
   - OpenMP target shared memory systems i.e where processor shared the main memory.
   - OpenMP is based on thread approach . It launches a single process which in turn can create *n* number of thread as desired. It is based on what is called "fork and join method" i.e depending on particular task it can launch desired number of thread as directed by user.

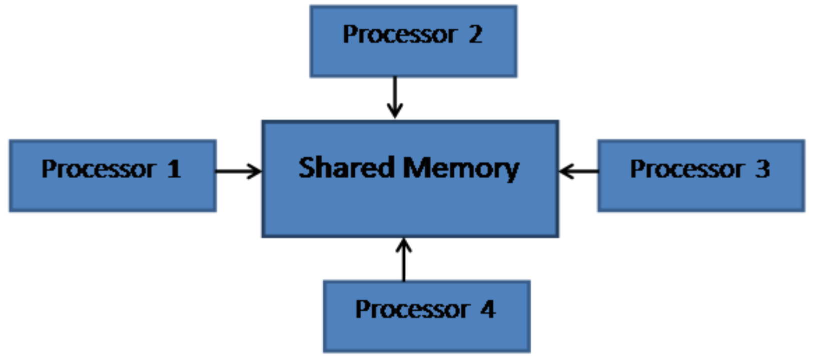


Fork and Join Model of OpenMP : Different stage of program show different number of thread

- Programming in OpenMP is relatively easy and involve adding *pragma* directive . User need to tell number of thread it need to use. (Note that launching more thread than number of processing unit available can actually slow down the whole program)

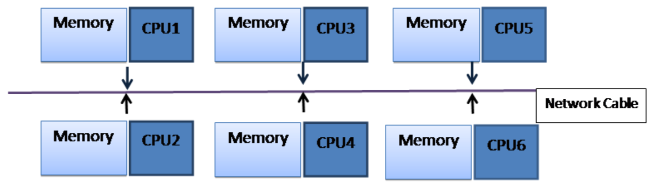
What is Shared Memory and Distributed Memory

1) Shared memory is one where all processors can see whole of the memory that is available . Simple example is your desktop computer or laptop , where all processing units can see all the memory of system



Shared Memory : Processor 1 ,2 3 4 can see whole memory

2) Distributed memory system is one where processor can see limited memory i.e two desktop computer connected in network . They can only see memory available to them only not of other



Distributed Memory System: CPU can see only limited memory of their own

What is Process and Thread

1) **Process**: An executing instance of program .It has distinct address space . It different from other executing instance of program in way that it has separate resources .

2) **Thread** is subset of process. A process can have *n* number of threads as desired . Every thread of process share its all resources i.e data as well address space of process that created it . Thread has to be part of some process.   It cannot be independent .

MPI	OpenMP
1 . Available from different vendor and can be compiled in desired platform with desired compiler. One can use any of MPI API i.e MPICH, OpenMPI or other	**1** .OpenMP are hooked with compiler so with gnu compiler and with Intel compiler one have specific implementation. User is at liberty with changing compiler but not with openmp implementation.
2. MPI support C,C++ and FORTRAN	**2**.OpenMP support C,C++ and FORTRAN
3.OpenMPI one of API for MPI is providing provisional support for Java	**3**.Few projects try to replicate openmp for Java.
4. MPI target both distributed as well shared memory system	**4**.OpenMP target only shared memory system
5.Based on both process and thread based approach .(Earlier it was mainly process based parallelism but now with MPI 2 and 3 thread based parallelism is there too. Usually a process can contain more than 1 thread and call MPI subroutine as desired	**5.**Only thread based parallelism.

6. Overhead for creating process is one time	**6.** Depending on implementation threads can be created and joined for particular task which add overhead
7.There are overheads associated with transferring message from one process to another	**7.**No such overheads, as thread can share variables
8. Process in MPI has private variable only, no shared variable	**8**. In OpenMP , threads have both private as well shared variable
9.Data racing is not there if not using any thread in process .	**9**. Data racing is inherent in OpenMP model
10. Compilation of MPI program require 1. Adding header file : #include "mpi.h" 2. compiler as:(in linux) *mpic++ mpi.cxx -o mpiExe* (User need to set environment variable PATH and LD_LIBRARY_PATH to MPI as OpenMPI installed folder or binaries) (For Linux)	**10**. Need to add omp.h and then can directly compile code with -fopenmp in Linux environment *g++ -fopenmp openmp.cxx -o openmpExe*

11 . Running MPI program . a) User need to make sure that bin and library folder from MPI installation are included in environmental variable PATH and LD_LIBRARY_PATH. b) For running executable from command line ,user need to supply following command and specify number of processor as in example below it is four . **mpirun -np 4 mpiExe**	**11**. User can launch executable *openmpExe* in normal way **./openmpExe**

Sample MPI program

```
#include <iostream>
#include <mpi.h>

/*******************************************************************************
This is a simple hello world program. Each processor print its id
****************************************************************/
using namespace std;
int main(int argc,char** argv)
{
   int myid, numprocs;

   MPI_Init(&argc,&argv);
```

```
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);


        /* output  my rank */
    cout<<"Hello from "<<myid<<endl;
    MPI_Finalize();

}
```

Command to run executable with name a.out in Linux = **mpirun -np 4   a.out**

Output

```
   Hello from 1
   Hello from 0
   Hello from 2
   Hello from 3
```

Sample OpenMP Program

```
#include<iostream>
#include<omp.h>
using namespace std;
/***********************************************************************
Sample OpenMP program which at stage 1 has 4 threads and at stage 2 has 2 threads
*************************************************************/
int main()
{
#pragma omp parallel  num_threads(4) //*create 4 threads and region inside it will be executed by
all   threads . */
{
  #pragma omp critical//allow one thread at a time to access below statement
  cout<<" Thread Id  in OpenMP stage 1=  "<<omp_get_thread_num()<< endl;
}  //here all thread get merged into one thread id
cout<<"I am alone"<<endl;

#pragma omp parallel num_threads(2)//create two threads
{
  cout<<" Thread Id  in OpenMP stage 2=  "<<omp_get_thread_num()<<  endl;;
}

}
```

Command to run executable with name a.out on Linux : /a.out

Output

```
    Thread Id  in OpenMP stage 1= 2
     Thread Id  in OpenMP stage 1=0
     Thread Id  in OpenMP stage 1=3
     Thread Id  in OpenMP stage 1= 1
     I am alone
     Thread Id  in OpenMP stage 2= 1
     Thread Id  in OpenMP stage 2=0
```

Summary

MPI and OpenMP have its own advantages and limitations . OpenMP is relatively easy to implement and involves few pragma directives to achieve desired tasks. OpenMP can be used in recursive function as well i.e as traversing in binary tree. However it suffers from problem of memory limitations for memory intensive calculations.
MPI usually serve those problem well which involve large memory. With MPI 3 , shared memory advantage can be utilized within MPI too. Also one can use OpenMP with MPI i.e for shared memory in targeted platform OpenMP can be used whereas for distributed one, MPI can be used.

Author
Pawan Ghildiyal

Google