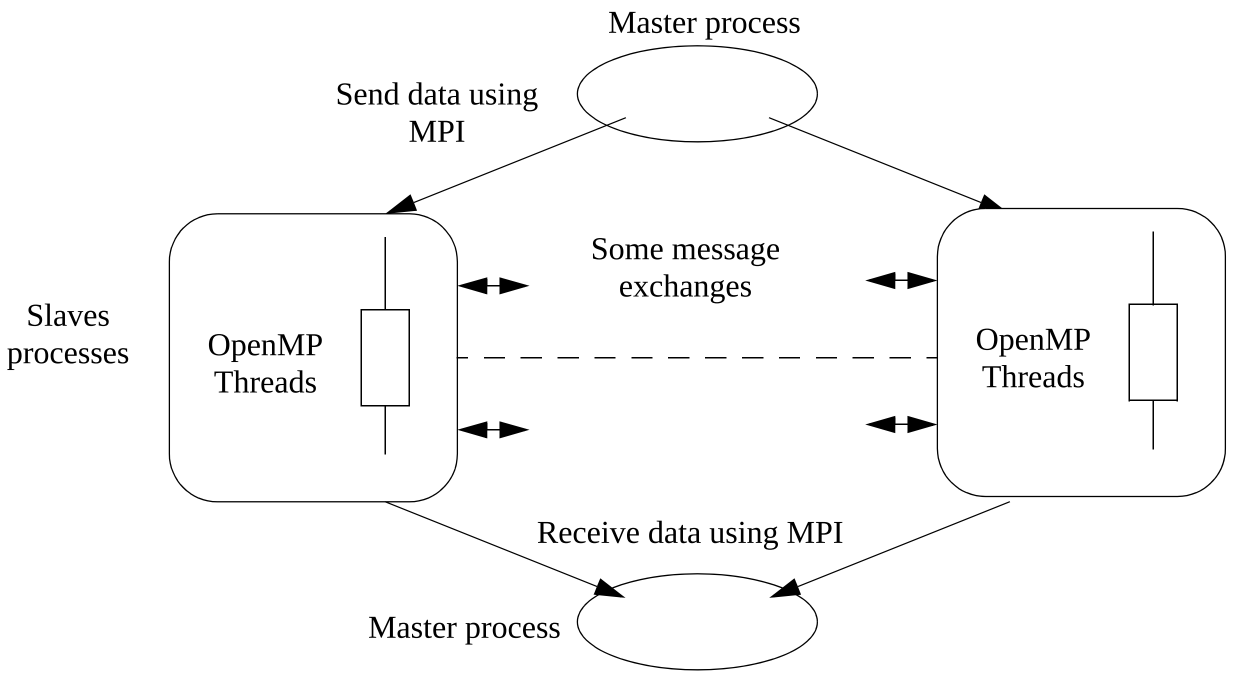
# ITCS 4145/5145 Parallel Computing
# Assignment 4
# Combining MPI message passing and OpenMP threads

**B. Wilkinson March 13, 2012**

In this assignment, you are to write a C program that has both MPI message-passing routines and OpenMP threads. This is a form of “hybrid programming” and extremely relevant for clusters of multiprocessor and multicore computer systems. The message-passing routines are used to pass messages between computer systems and the threads are run using the multiple cores on each system. The overall structure of the program is illustrated below:



Write and test a hybrid MPI/OpenMP program that executes on the coit-grid cluster demontrating the structure above. The actual computation to be performed is for you to decide but it must achieve a purpose. Possible computations of sufficient complexity are:

- Sorting numbers
- Finding prime numbers
- Searching a sequence for a pattern

## Sample MPI/OpenMP program and compiling

A sample MPI/OpenMP program is given below:

```
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <stdlib.h>
#include "mpi.h"

#define CHUNKSIZE   10
#define N       100

void openmp_code(){
   int nthreads, tid, i, chunk;
   float a[N], b[N], c[N];
   for (i=0; i < N; i++)
      a[i] = b[i] = i * 1.0;             // initialize arrays

   chunk = CHUNKSIZE;

   #pragma omp parallel shared(a,b,c,nthreads,chunk) private(i,tid) {
   tid = omp_get_thread_num();
   if (tid == 0){
      nthreads = omp_get_num_threads();
      printf("Number of threads = %d\n", nthreads);
   }

   //  printf("Thread %d starting...\n",tid);

   #pragma omp for schedule(dynamic,chunk)

   for (i=0; i<N; i++){
      c[i] = a[i] + b[i];
   //  printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);
   }
   }  /* end of parallel section */
}

main(int argc, char **argv ) {
   char message[20];
   int i,rank, size, type=99;
   MPI_Status status;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD,&size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   if(rank == 0) {
      strcpy(message, "Hello, world");
      for (i=1; i<size; i++)
      MPI_Send(message, 13, MPI_CHAR, i, type, MPI_COMM_WORLD);
   }
   else
      MPI_Recv(message, 20, MPI_CHAR, 0, type, MPI_COMM_WORLD, &status);

   if( 1){ //* if this machine has multiple cores
      openmp_code(); //run the mpi code
   }
   printf( "Message from process =%d : %.13s\n", rank,message);
   MPI_Finalize();
}
```

It can also be found on the course home page in the assignment section. Compile with:

```
mpicc -o mpi_out mpi_test.c -fopenmp
```

Note this command invokes the regular cc compiler not the Intel icc compiler.

## Submission

Produce a document that describes the purpose of your computation and your code. Show that the program executed successfully on the coit-grid servers. Include screen shots in the document. Provide insightful conclusions. Submit a single pdf file to Moodle by the due date (see home page). Include your code. Grading will depend upon report and work. Greater expectations from graduate students.
***It is very important that you work independently. Copied work will not be accepted.***