University of Hyderabad, Department of Computer and Information Sciences

CS726-Parallel Computing

Max. Marks: 60 **Date: November 18, 2010**

Duration: 3 Hrs.

Note:
1. Attempt <u>**all**</u> questions, each carry equal marks
2. Each question contains **3** sub questions. Attempt **any and only 2** questions
3. Answer sub questions of a question before attempting the next question
4. In case of any doubt, mention your assumptions in the answer-book and proceed for your answers

Q.1 (A) What is Gustafson's Law? With an example show how it overcomes the limitations of Amdhal's law.

(B) Explain the functioning of the Shuffle Exchange Network in terms of criteria used to define networks capabilities.

(C) Propose a parallel algorithm for sorting ***n*** integers using Bucket Sort, where numbers can be repeated more than once in the given input.

Q.2 (A) Explain the working behavior of the following program segment and write your comments. Assume that there is no syntax error in the program segment.

```
#include <omp.h>
int  a, b, i, tid;
float x;
main ()  {
  omp_set_dynamic(0);
#pragma omp parallel private(b,tid)
  {
  tid = omp_get_thread_num();
  a = tid;
  b = tid;
  x = 1.1 * tid +1.0;
  printf("Thread %d:   a,b,x= %d %d %f\n",tid,a,b,x);
  }
#pragma omp parallel private(tid)
  {
  tid = omp_get_thread_num();
  printf("Thread %d:   a,b,x= %d %d %f\n",tid,a,b,x);
  }
}
```

(B) What is PRAM model of Computation? How does it achieve concurrent write operations?

(C) Propose a row wise Matrix Vector multiplication program for distributed multi computer system using constructs of MPI.

Q.3 (A) Propose a PRAM algorithm for merging two sorted list. What is the parallel time and processor's complexity of your algorithm?

(B) Using Foster's design methodology, writes a parallel program for finding the value of ***pi*** using numerical integration.

(C) List down advantages and disadvantages of using asymmetrical multi-computers.

Q.4 (A) Explain the working behavior of the following program segment and write your comments when

i. it is run as mpirun –np 4 *filename*
ii. it is run as mpirun –np 5 *filename*

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

int main (int argc, char *argv[])
{
int numtasks, rank, sendcount, recvcount, source;
float sendbuf[SIZE][SIZE] = {
  {1.0, 2.0, 3.0, 4.0},
  {5.0, 6.0, 7.0, 8.0},
  {9.0, 10.0, 11.0, 12.0},
  {13.0, 14.0, 15.0, 16.0}  };
float recvbuf[SIZE];

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

if (numtasks == SIZE) {
  source = 1;
  sendcount = SIZE;
  recvcount = SIZE;
  MPI_Scatter(sendbuf,sendcount,MPI_FLOAT,recvbuf,recvcount,
             MPI_FLOAT,source,MPI_COMM_WORLD);

  printf("rank= %d  Results: %f %f %f %f\n",rank,recvbuf[0],
         recvbuf[1],recvbuf[2],recvbuf[3]);
  }
else
  printf("Must specify %d processors. Terminating.\n",SIZE);

MPI_Finalize();
}
```

(B) Explain how Cache Coherency problem is handled in the Non Uniform Memory Access Machines?

(C) Compare work sharing constructs of OpenMP based on their working.

Q. 5(A) What is cluster? What are the common models? How do you classify them?

(B) Write a **MPI/pthread** program for the following situation: **"N** threads are to be created, each taking numbers from the list to add to their partial sums. When all numbers have been taken, the threads can add their partial results to a shared location **sum**. A shared loacation **global_index** can be used by each thread to select next element of **a[ ]**."

(C) Explain the OpenMP programming model for shared memory programming.

------Best of Luck------

```
#include <omp.h>

int  a, b, i, tid;
float x;

main ()  {

  omp_set_dynamic(0);

#pragma omp parallel private(b,tid)
  {
  tid = omp_get_thread_num();
  a = tid;
  b = tid;
  x = 1.1 * tid +1.0;
  printf("Thread %d:   a,b,x= %d %d %f\n",tid,a,b,x);
  }

#pragma omp parallel private(tid)
  {
  tid = omp_get_thread_num();
  printf("Thread %d:   a,b,x= %d %d %f\n",tid,a,b,x);
  }
}
Output:

1st Parallel Region:
Thread 0:   a,b,x= 0 0 1.000000
Thread 2:   a,b,x= 2 2 3.200000
Thread 3:   a,b,x= 3 3 4.300000
Thread 1:   a,b,x= 1 1 2.100000
************************************
Master thread doing serial work here
************************************
2nd Parallel Region:
Thread 0:   a,b,x= 0 0 1.000000
Thread 3:   a,b,x= 3 0 4.300000
Thread 1:   a,b,x= 1 0 2.100000
Thread 2:   a,b,x= 2 0 3.200000
```

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

int main (int argc, char *argv[])
{
int numtasks, rank, sendcount, recvcount, source;
float sendbuf[SIZE][SIZE] = {
  {1.0, 2.0, 3.0, 4.0},
  {5.0, 6.0, 7.0, 8.0},
  {9.0, 10.0, 11.0, 12.0},
  {13.0, 14.0, 15.0, 16.0}  };
float recvbuf[SIZE];

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

if (numtasks == SIZE) {
  source = 1;
  sendcount = SIZE;
  recvcount = SIZE;
  MPI_Scatter(sendbuf,sendcount,MPI_FLOAT,recvbuf,recvcount,
             MPI_FLOAT,source,MPI_COMM_WORLD);

  printf("rank= %d  Results: %f %f %f %f\n",rank,recvbuf[0],
         recvbuf[1],recvbuf[2],recvbuf[3]);
  }
else
  printf("Must specify %d processors. Terminating.\n",SIZE);

MPI_Finalize();

}
```

```
rank= 1  Results: 5.000000 6.000000 7.000000 8.000000
rank= 0  Results: 1.000000 2.000000 3.000000 4.000000
rank= 3  Results: 13.000000 14.000000 15.000000 16.000000
rank= 2  Results: 9.000000 10.000000 11.000000 12.000000
```