**CSE–312 PARALLEL PROGRAMMING LAB**

**[ 0 0 3 1 ]**

**Week 1**

1. Study of working environment of Visual studio for MPI and OpenCL programming. Write a simple C++ program to learn how to use Visual studio environment.
2. Study of MPI functions. Write MPI program to display message "Hello world" by all the processes.

**Week 2**

1. Write a MPI program using standard mode send and receive. The program consists of two processes. Process 0 sends a message to the Process1. Process1 receives the message and sends it back.
2. Write an MPI program if the process is the master, then send a message, to each of the slaves and if the process is a slave, then receive the message.
3. Write MPI program to send a number from master process to its slaves. Let slaves print them as they receive these numbers.
4. Write an MPI program to add an array of size N using 2 processors.
5. Modify the above program to handle errors.

**Week 3**

1. Write a MPI program using synchronous send. The program does exactly the same as the program with standard mode send, but the communication is synchronous. The sender process sends a message to the receiver. This receives the message and sends it back. Both processes use synchronous send operations (MPI\_Send).
2. Write a MPI program to investigate the amount of time required for execution by each process.
3. Write an MPI program to read n elements of the array. If even processor, find square of the corresponding elements. If odd processor, find cube of corresponding elements.
4. Write an MPI program to read elements of an array. Search a number in this array using 2 processes.

**Week 4**

1. Write a MPI program using buffered send. The program consists of two processes. Process 0 sends a message to the receiver. This receives the message and sends it back, but now with a buffered send.
2. Write an MPI program to read n elements in root process. Send one element to corresponding process using scatter.
3. Write an MPI program to scatter one element to each process and send it to root process using gather.
4. Write an MPI program to read n\*m elements in root process where n is the number of processes and m is the number of elements expected to be scattered to each slaves including root. Send m elements to corresponding process using scatter.
5. Write an MPI program to read n\*m elements in root process. Send m elements to each process (use scatter), find average of m elements in each process, send these average to root (use gather). Find the total average in root process.

**Week 5**

1. Write an MPI program to read a number in the root process. Broadcast this value to all the processes and let each process find its square. Calculate the total sum of these squares using reduce and display the result in the root process.
2. Write an MPI program to read n\*m elements in the root process where n is the number of processes and m is the number of elements. Find the sum of m elements using n processes. Calculate the total sum using reduce and display the result in the root process.
3. Write an MPI program using n processes to find 1! + 2! +…..+n!.

**Week 6**

1. Write an MPI program to read a 3 x 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using 3 processes.
2. Write an MPI program to calculate π-value by integrating f(x) = 4 /(1+x2) . Area under the curve is divided into rectangles and the rectangles are distributed to the processors.

**Week 7**

Study of OpenCL APIs. Write an OpenCL program to initialise the GPU and test the initialization.

1. Write an OpenCL program to copy a string.
2. Write an OpenCL program to copy a string in parallel.
3. Write an OpenCL program to convert in parallel an uppercase string to lowercase.

**Week 8**

1. Write an OpenCL program for vector-vector addition.
2. Write an OpenCL program to find the square of each element in a vector.
3. Write an OpenCL program to find the square of each element of an array and add the respective elements of the original array.
4. Write an OpenCL program to convert each element of an array in to its equivalent binary value.

Give bench mark for the following:

1. Time taken with sequential code
2. Time taken with parallel computation

**Week 9**

1. Write an OpenCL program to copy a string n times parallelly.
2. Write an OpenCL program for matrix-matrix multiplication.

Give bench mark for the following:

* + 1. Time taken with sequential code
    2. Time taken with parallel computation

**Week 10**

1. Write an OpenCL program to read n words of a string and reverse each word of it parallelly.
2. Write an OpenCL program to read a string and reverse it parallelly.
3. Write an OpenCL program for scalar-matrix multiplication.

Give bench mark for the following:

1. Time taken with sequential code
2. Time taken with parallel computation

**Week 11**

1. Write an OpenCL program to sort N elements of a given array using selection sort method.
2. Write an OpenCL program to read a string and sort it.
3. Write an OpenCL program to read *n* words of a string and sort the words in parallel.

Give bench mark for the following:

1. Time taken with sequential code
2. Time taken with parallel computation

**Week 12**

i) Write an OpenCL program to compute the Transpose of given matrix.

Give bench mark for the following:

1. Time taken with sequential code
2. Time taken with parallel computation

ii)Write an OpenCL program to sum the row and column elements of a matrix.

iii)Write an OpenCL program to calculate π-value.

**References:**

1. Benedict R. Gaster, Lee Howes, David R, Perhaad Mistry, Dana Schaa, “Heterogeneous Computing with OpenCL” Morgan Kaufmann, 2012.
2. V.Rajaraman, C. Siva Ram Murthy, “ Parallel Computers Architecture and Programming” Prentice-Hall India, 2000.
3. Michael J Quinn, “Parallel Computing: Theory and Practice” Tata McGraw Hill, 2nd Edition,1994.
4. Michael J Quinn, “Parallel Programming   
   in C with MPI and OpenMP”, Tata McGraw Hill, 2011.
5. <http://developer.amd.com/sdks/AMDAPPSDK/documentation/pages/TutorialOpenCL>
6. http://developer.download.nvidia.com/compute/cuda/3\_0/sdk/website/OpenCL/website/samples.html

**ADDITIONAL PROGRAMS:**

* 1. Write an MPI program to read n elements of the array. If even processor, find square of the corresponding elements. If odd processor, find cube of corresponding elements.
  2. Write an MPI program to print “hello” by processes whose address id is a perfect square.
  3. Write an MPI program to add an array of size N using n processors.
  4. Write an MPI program to read elements of an array. Search a number in this array using n processes.
  5. Write an MPI program to find substring in a string.