# GUJARAT TECHNOLOGICAL UNIVERSITY

**Master of Engineering**
**Subject Code: 3720220**
**Semester – II**
**Subject Name: HPC Architecture and ECO system**

**Type of course: Elective**

**Prerequisite:** Computer Architecture and Organization, Data Structure

**Rationale:** Algorithmic processing performed in High Performance Computing environments impacts the lives of billions of people, and planning for exascale computing presents significant power challenges to the industry. The objective of the course is to impart in depth knowledge of parallel architectures and different shared/distributed memory architectures in support to exascale computing. MPI and OpenMP are discussed along with their applications.

**Teaching and Examination Scheme:**

| Teaching Scheme | | | Credits | Examination Marks | | | | Total Marks |
|---|---|---|---|---|---|---|---|---|
| | | | | Theory Marks | | Practical Marks | | |
| L | T | P | C | ESE (E) | PA (M) | ESE (V) | PA (I) | |
| 3 | 0 | 2 | 4 | 70 | 30 | 30 | 20 | 150 |

**Content:**

| Sr. No. | Content | Total Hrs |
|---|---|---|
| **1** | Overview of parallel system organization | 4 |
| **2** | Examples of Scientific Computing; Parallel Languages, Embarrassingly parallel problems; problem decomposition, graph partitioning and load balancing | 12 |
| **3** | Introduction to message passing and MPI programming | 7 |
| **4** | Introduction to shared memory and OpenMP programming | 7 |
| **5** | Recent trends in OpenMP and MPI programming, application areas of scientific computing. | 9 |
| **6** | Performance analysis tools; HPCToolkit, OpenSpeedShop,<br>Debugging and Profiling tools; GDB, PAPI, mpiP, ompP, Valgrind, MPI Program Profiler | 9 |
| | **Total** | **48** |

**Reference Books:**


1. 1. Parallel Programming for Multicore and Cluster Systems by Thomas Rauber and Gudula Runger.
2. Scientific Parallel Computing by Scott, Clark, and Bagheri.

3. Using OpenMP: Portable Shared Memory Parallel Programming by Chapman, Jost, and van der Pas.

**Course Outcomes:**

| Sr. No. | CO statement | Marks % weightage |
|---|---|---|
| CO-1 | Develop parallel algorithms for high performance systems. | 25% |
| CO-2 | Perform problem decomposition and load balancing using MPI and OpenMP. | 25% |
| CO-3 | Identify hot spots in the codes using performance analysis tools. | 25% |
| CO-4 | Evaluate the performance of the code using the tools | 25% |

Distribution of marks weightage for cognitive level

| Bloom's Taxonomy for Cognitive Domain | Marks % weightage |
|---|---|
| Recall | 5 |
| Comprehension | 5 |
| Application | 20 |
| Analysis | 25 |
| Evaluate | 25 |
| Create | 20 |

**Practical List:**

Use Valgrind, Vtune Amplifier, Nvidia Visual Profiler and Nvidia Nsight to identify hotspots and other parameters for detailed analysis of following the practicals.

1) Calculate standard deviation using Pthread, OpenMP and MPI.
2) Write parallel code for Matrix Matrix Multiplication using MPI cluster of 4 nodes, OpenMP, PVM cluster of 4 nodes, OpenACC and CUDA and compare and plot the performance in terms of execution time for Matrix size of 1000 x 1000, 5000 x 5000, 10,000 x 10,000, 20,000 x 20,000.
3) Write the programs in MPD or in C with the Pthreads library for the following:
   a) Sequential Jacobi iteration program
   b) Parallel Jacobi iteration program
   c) Sequential multigrid program
   d) Parallel multigrid program
4) Perform Monte Carlo simulation using NVIDIA's CURAND library for random number generation.
   Write your own small program to compute the average value of
   $az^2 + bz + c$

where z is a standard Normal random variable (i.e. zero mean and unit variance, which is what the random number generator produces) and a, b, c are constants which you should store in constant memory. It is suggested to use each thread to average over 100 values, and then write this to a device array which gets copied back to the host for the averaging over the contributions from each of the threads.

(Note: the average value should be close to a + c.)

5) Implement 3D Laplace Finite Solver using CUDA and OpenACC.