|  |  |  |  |
| --- | --- | --- | --- |
| Faculty In-charge | Dr. B. Neelima | Year and Semester | June-December-2019 |
| Course code | IT300 | Course Name | Parallel Computing |
| Core/Elective/MLC | Core | L-T-P | 3-0-2 |
| Pre-requisites | Computer Organization and Architecture; Basic C Programming | Contact Hours | Lecture: 3hours/week  Lab: 2 hours/Week |
| Type of course:  (Lecture/Tutorial/Seminar/Project) | Lecture and Lab | Course Assessment Methods: (Both continuous and semester-end assessment) | Theory: (60%)  20%: Mid-sem Theory Exam  40%: End-sem Theory Exam  Lab: (40%)  1) 10% Lab Exam:  OpenMP/MPI/CUDA  2) Mini-project (30%)   * 10% Mid-sem Evaluation * -20% End-sem Evaluation |
| **Course Highlights**: Introduction to Parallel Computer Architectures, Shared memory and distributed memory programming techniques, Parallel Programming with OpenMP, MPI, Parallel Programming techniques like Task Parallelism using TBB, TL2, Cilk++ etc. and software transactional memory techniques. Introduction to accelerator programming using CUDA/OpenCL and Xeon-phi. Concepts of Heterogeneous programming techniques. Projects to implement a few of these techniques introduced in this course through the lab.  The course covers fundamentals of Parallel Computing both in terms of hardware and software. It covers parallel computer architectures, parallel programming environments like OpenMP/MPI/CUDA  **Course Objectives**:   1. Study principles of concurrency and parallelism 2. Learn parallel algorithm design and analysis 3. Implement High Performance Parallel Programs using programming models and libraries   **Course Outcomes**: After completion of this course the students are able to:  CO1: Recapitulate concurrent parallel systems and sources of concurrency  CO2: Design and analyse parallel algorithms  CO3: Explicate parallel algorithms using shared, distributed/accelerator programming models and libraries  CO4: To implement applications using OpenMP, MPI and CUDA | | | |
| ***Course Description: Theory*: Study of Concurrent Parallel Systems**: Concurrency Vs. Parallelism; Overview of concurrent parallel systems; the evolution of multi-core processors, concept of free lunch is over; Study of Moore’s law; design principles and memory hierarchy of multiprocessor machines; classification of parallel computers; types of classification; Shared memory multiprocessors; cache coherence protocols; types of dependency; Bernstein conditions for detection of parallelism; dependence graphs and notations; studying towards vectorization as a limited granularity of parallelism; distributed memory systems; types of interconnections; study design principles of interconnections; principles of communication; analysing parallel code; speed-up and performance metrics and laws for parallelization; basic optimization techniques for serial code; principles of parallel algorithm design  **Shared-memory parallel programming**: Shared-memory programming need, motivation and evolution; overview of OpenMP development; concepts and syntax of OpenMP; parallel execution; data scoping; work-sharing constructs; synchronization; reductions; environment variables; runtime library routines and miscellaneous; parallel application development and implementation; Introduction to TBB, TL2, Cilk++ etc. and software transactional memory techniques  **Distributed-memory parallel programming**: distributed-memory programming model evolution; message and point-to-point communication; collective communication; non-blocking point-to-point communication; virtual topologies; MPI programming constructs and application development and implementation.  **Accelerator computing**: Introduction to many-core architectures; evolution of GPGPUs; compute unified device architecture (CUDA) and programming; processing elements and memory hierarchy in many-core architecture; use of occupancy calculator; study and compare two versions of many-core architectures; Introduction to CUDA programming; CUDA program structure; memory transfer; kernel launch parameter design; kernel creation; steps of compilation; CUDA application development and implementation. Introduction to Heterogeneous programming using Xeon-Phi and OpenCL. | | | |
| **Course Plan**: (Theory + Lab)  **Week 01** (15-19, Jul.):  **Theory: (1-Hour): (CO1):**Overview of the course and course evolution plan. Review of computer architecture, operating systems and compiler design based on the prerequisite of the courses  **Lab (2-Hours):** **(CO4):** Assignments based on prerequisite courses  **Week 02** (22-26, Jul.):  **Theory: (3-Hours): (CO1):** Concurrency Vs. Parallelism; Overview of concurrent parallel systems; the evolution of multi-core processors, concept of free lunch is over; Study of Moore’s law;  **Lab (2-Hours): (CO4):** Assignments based on prerequisite courses  **Week 03** (29, July-02, Aug.):  **Theory:** **(3-Hours): (CO1):** design principles and memory hierarchy of multiprocessor machines; classification of parallel computers; types of classification; Shared memory multiprocessors; Shared-memory programming need, motivation and evolution;  **(CO3):** overview of OpenMP development; concepts and syntax of OpenMP  **Lab (2-Hours): (CO4):** Introduction to OpenMP shared-memory parallel programming constructs  **Week 04** (5-9, Aug.):  **Theory: (2-Hour): (CO3):** OpenMP: parallel execution; data scoping; worksharing constructs; synchronization; reductions; environment variables; runtime library routines and miscellaneous; parallel application development and implementation  Note: One class is not there due to Inst. Foundation Day  **Lab (2-Hours): (CO4):** Understanding OpenMP constructs through exercise programs  **Week 05** (12-16, Aug.):  **Theory: (3-Hours): (CO2):** cache coherence protocols; MSI and MESI protocols; types of dependency; Bernstein conditions for detection of parallelism; dependence graphs and notations; studying towards vectorization as a limited granularity of parallelism;  **Lab (2-Hours):** Note: No laboratory due to Monday’s time table on Wednesday  **Week 06** (19-23, Aug.):  **Theory: (3-Hours): (CO2):** Bernstein conditions for detection of parallelism; dependence graphs and notations; studying towards vectorization as a limited granularity of parallelism; distributed memory systems; types of interconnections;  **Lab (2-Hours): (CO2 & CO4):** Application development-1using OpenMP  **Week 07** (26-30, Aug.):  **Theory: (3-Hours): (CO2):** study design principles of interconnections; principles of communication; analyzing parallel code;  **Lab (2-Hours): (CO2 & CO4):** Application development-2 using OpenMP  **Week 08** (2-6, Sep.):  **Theory: (3-Hours): (CO2):** speed-up and performance metrics and laws for parallelization; basic optimization techniques for serial code; principles of parallel algorithm design; Introduction to many-core architectures; evolution of GPGPUs;  **Lab (2-Hours):**  Note: No laboratory due to Monday’s time table on Wednesday  **Week 09** (9-13, Sep.):  **Theory:** (**2-Hours): (CO3):** compute unified device architecture (CUDA) and programming; processing elements and memory hierarchy in many-core architecture; use of occupancy calculator;  **Lab (2-Hours): (CO2 & CO4):** Application development-3 using OpenMP  **Week 10** (16-20, Sep.):  **Theory: (0-Hours):**  Note: No classes due to Mid-sem exam  **Lab (2-Hours):** Note: No lab due to Mid-sem exam  **Week 11** (23-27, Sep.):  **Theory: (3-Hours): (CO3):** study and compare two versions of many-core architectures; Introduction to CUDA programming; CUDA program structure; memory transfer;  **Lab (2-Hours): (CO4):** Mini-project themes and team details submission  **Week 12** (30, Sep.- 4, Oct.):  **Theory: (3-Hours): (CO3):** kernel launch parameter design; kernel creation; steps of compilation; CUDA application development and implementation  **Lab (2-Hours):** Note: No Laboratory due to Holiday  **Week 13** (7-11, October):  **Theory: (0-Hours):** Note: No classes due to holidays and Engineer.  **Lab (2-Hours): (CO4):** Understanding CUDA programming constructs through exercises  **Week 14** (14-18, Oct.):  **Theory: (2-Hours): (CO2):** distributed-memory programming model evolution; message and point-to-point communication; non-blocking point-to-point communication; syntax and routines in MPI  **Lab (2-Hours): (CO2 & CO4):** Application Development using CUDA  **Week 15** (21-25, Oct.):  **Theory:** (**3-Hours): (CO2):** collective communication; types and usage  **Lab (2-Hours): (CO4):** Understanding MPI constructs through MPI programming  **Week 16** (28, Oct.-1, Nov.):  **Theory: (4-Hours): (CO3):** virtual topologies; MPI programming constructs and application development and implementation  **Lab (2-Hours): (CO4):** MSE evaluation of PC-Lab  **Week 17** (4-8, Nov.):  **Theory: (3-Hours): (CO3):** Introduction to TBB, TL2, Cilk++ etc. and software transactional memory techniques; Introduction to heterogenous programming using OpenCL and Xeon-phi.  **Lab (2-Hours): (CO4):** Mini-Project evaluation  **Week 18** (11-13, Nov.):  **Theory: (1-Hour): (CO1):** Review of important concepts in parallel computers and programming models  **Lab (2-Hours): (CO4):** Mini-Project report submission | | | |
| **Prerequisites:** Knowledge of computer programming using C/C++; Knowledge of general-purpose processor architecture; Knowledge of operating system and compiler design; Basics mathematics | | | |
| **Text Books**:   1. Parallel Computer Architecture: A Hardware/Software Approach, By David Culler, Jaswinder Pal Singh, Anoop Gupta, 1997 2. Introduction to Parallel Computing, Second Edition By Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, 2003 3. Programming Massively Parallel Processors: A Hands-on Approach, 1st, David B. Kirk and Wen-Mei W. Hwu, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2010 4. Using OpenMP-Portable Shared Memory Parallel Programming-by Chapman, Jost and Van Der Pas, 2007 5. Using OpenMP-The Next Step by Ruud Van Der Pas, Eric Stotzer and Christian Terboven, 2017   **References**:   1. Introduction to High Performance Computing for Scientists and Engineers, Georg Hager and Gerhard Wellein, CRC Press, 2010. 2. Using Advanced MPI: Modern Features of the Message-Passing Interface by William Gropp, Torsten Hoefler, Rajeev Thakur and Ewing Lusk, MIT Press, 2014 3. B. Wilkinson and M. Allen, "Parallel Programming: Techniques and Applications”, 2nd ed., Pearson, 2004. 4. Benedict R. Gaster et al., Heterogeneous Computing with OpenCL, 2nd Edition, Morgan Kaufmann. 2012. 5. Rezaur Rahman, Intel Xeon-Phi Coprocessor Architecture/Tools - The Guide for App. Developers, Apress, 2013. 6. CUDA for Engineers by Duane Storti and Mete Yurgotlu, Addison-Wesley, 2016 7. J. Dongara, I. Foster, G. Fox, W. Cropp et al, "Sourcebook of Parallel Programming", Morgan Kaufmann, 2002. 8. <https://www.openmp.org/> 9. <https://developer.nvidia.com/cuda-zone> 10. <https://www.open-mpi.org/> | | | |