UCS607: PARALLEL AND DISTRIBUTED COMPUTING

L	T	P	Cr
3	0	2	4.0

Course objective: To learn the concepts of Parallel and Distributed Computing and its implementation for assessment of understanding the course by the students.

Detail contents:

Parallelism Fundamentals : Scope and issues of parallel and distributed computing, Parallelism, Goals of parallelism, Parallelism and concurrency, Multiple simultaneous computations, Programming Constructs for creating Parallelism, communication, and coordination. Programming errors not found in sequential programming like data races, higher level races, lack of liveness

Parallel Architecture : Architecture of Parallel Computer, Communication Costs, parallel computer structure, architectural classification schemes, Multicore processors, Memory Issues : Shared vs. distributed, Symmetric multiprocessing (SMP), SIMD, vector processing, GPU, co-processing, Flynn's Taxonomy, Instruction Level support for parallel programming, Multiprocessor caches and Cache Coherence, Non-Uniform Memory Access (NUMA)

Parallel Decomposition and Parallel Performance: Need for communication and coordination/synchronization, Scheduling and contention, Independence and partitioning, Task-Based Decomposition, Data Parallel Decomposition, Actors and Reactive Processes, Load balancing, Data Management, Impact of composing multiple concurrent components, Power usage and management. Sources of Overhead in Parallel Programs, Performance metrics for parallel algorithm implementations, Performance measurement, The Effect of Granularity on Performance Power Use and Management, Cost-Performance trade-off;

Communication and Coordination: Shared Memory, Consistency, Atomicity, Message-Passing, Consensus, Conditional Actions, Critical Paths, Scalability, cache coherence in multiprocessor systems, synchronization mechanism.

CUDA programming model: Overview of CUDA, Isolating data to be used by parallelized code, API function to allocate memory on the parallel computing device, API function to transfer data to parallel computing device, Concepts of Threads, Blocks, Grids, Developing kernel function that will be executed by threads in the parallelized part, Launching the execution of kernel function by parallel threads, transferring data back to host processor with API function call.

Parallel Algorithms design, Analysis, and Programming : Parallel Algorithms, Parallel Graph Algorithms, Parallel Matrix Computations, Critical paths, work and span and relation to Amdahl's law, Speed-up and scalability, Naturally parallel algorithms, Parallel algorithmic patterns like divide and conquer, map and reduce, Specific algorithms like parallel Merge Sort, Parallel graph algorithms, parallel shortest path, parallel spanning tree, Producer-consumer and pipelined algorithms.

Laboratory work : To implementparallel programming using CUDA with emphasis on developing applications for processors with many computation cores, mapping computations to parallel hardware, efficient data structures, paradigms for efficient parallel algorithms.

Text Books:

1. *C Lin, L Snyder. Principles of Parallel Programming. USA: Addison-Wesley (2008).*
2. *A Grama, A Gupra, G Karypis, V Kumar. Introduction to Parallel Computing, Addison Wesley (2003), 2nd edition .*

Reference Books:

1. *B Gaster, L Howes, D Kaeli, P Mistry, and D Schaa. Heterogeneous Computing With Opencl. Morgan Kaufmann and Elsevier (2011).*
2. *T Mattson, B Sanders, B Massingill. Patterns for Parallel Programming. Addison-Wesley (2004).*
3. *Quinn, M. J.,Parallel Programming in C with MPI and OpenMP. McGraw-Hill (2004).*

Distributed Computing: In distributed computing we have multiple autonomous computers which seems to the user as single system. In distributed systems there is no shared memory and computers communicate with each other through message passing. In distributed computing a single task is divided among different computers. Difference between Parallel Computing and Distributed Computing: S.NO. Parallel Computing. Distributed Computing. 1. Parallel and Distributed Computing (PDC) is a specialized topic, commonly encountered in the general context of High Performance/Throughput Computing. We mainly see three kind of material that could be considered when it comes to teaching PDC. â€¢ First, the literature.Â Teaching parallel and distributed programming at any level is a genuine requirement nowadays. In order to fulll this crucial need, PDC course should be incorporated into stan-dard scientic and engineer curriculum. There is certainly a pedagogical effort to bring this topic, previously reserved for specialists, into the standard. Parallel computing is a methodology where we distribute one single process on multiple processors. Every single processor executes a portion of the program simultaneously and once execution complete the result merged and return to the main computer. Nowadays computers support the concept of Multiple Instruction Multiple Data stream (MIMD) which is another form of the parallel computing. It is also known as multi processor computing system. You may found another type of parallel computing where multiple computers are used to implement same task and return result to main computer or master compu... The Cisco UCS solution for Cloudera is based on Cisco UCS Integrated Infrastructure for Big Data, a highly scalable architecture designed to meet a variety of scale-out application demands with seamless data integration and management integration capabilities built using the following componentsÂ Deployed in redundant pairs, Cisco fabric interconnects offer the full active-active redundancy, performance, and exceptional scalability needed to support the large number of nodes that are typical in clusters serving big data applications. Cisco UCS Manager enables rapid and consistent server configuration using service profiles, automating ongoing system maintenance activities such as firmware updates across the entire cluster as a single operation. Distributed and parallel computing in Machine Learning Server. 12/19/2017. 3 minutes to read. In this article. Machine Learning Server's computational engine is built for distributed and parallel processing, automatically partitioning a workload across multiple nodes in a cluster, or on the available threads on multi-core machine. Access to the engine is through functions in our proprietary packages: RevoScaleR (R), revoscalepy (Python), and machine learning algorithms in MicrosoftML (R) and microsoftml (Python), respectively. RevoScaleR and revoscalepy provide data structures and data op