BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES

COURSE HANDOUT

Part A: Content Design

Course Title	Introduction to Parallel and Distributed Programming
Course No(s)	CC ZG501
Credit Units	5
Course Author	K HARI BABU
Instructor in charge	DN Sachin
Version No	1
Date	14-12-2022

Course Description

As the Cloud is a distributed system, the prime focus of this course is to teach how to program a basic distributed system. The course is driven by a set of lab exercises that are connected through a single use case to build a distributed system. Programming concepts & system design details are discussed in class, expecting that the students will implement the relevant parts of the system incrementally through labs as the course progresses.

Course Objectives

No	Course Objective
CO1	To motivate the need of programming the parallel and distributed systems and introduce cloud use cases
CO2	To introduce low level communication through secure socket programming for the establishment of a master-slave architecture and centralized control based on dynamic membership of nodes
CO3	To introduce high level communication with openmpi programming and develop prototype of a cluster computer that can distribute the work and demonstrate parallel data processing across the cluster nodes and also to deal with fault tolerance, synchronization and efficiency in communication
CO4	To emphasize the possibilities of performance enhancement by converting sequential programs to a multithreaded parallel programs
CO5	To explore the challenges that arise in the large scale data storage/processing systems aiming high availability, scalability and durability of data by introducing checkpointing and failure handling
CO6	To introduce CUDA library and Demonstrate use of GPUs for massive multi-core programming
CO7	To introduce queues, pub-sub concepts in communication for the development of an event-driven distributed system prototype

Course Objectives

No	Course Objective
LO1	Learners can recall the need of parallel and distributed systems, design or choose an appropriate programming model to solve a the given problem, also recall the evolution of these programming paradigms, their applicability; benefits, as well as current and future challenges;
LO2	Learners will be able to design and develop programs that can communicate processes running across the machines in a Distributed memory model using MPI libraries
LO3	Learners will be able to employ Shared-Memory Parallelism through OpenMP and Pthreads' libraries

LO4	Learners will be able to develop GPU programs using CUDA library
LO5	Learners will be able to design an event driven system and program the prototype

Modular Structure

Module 1: Nodes, Processes and communication (Sessions 1-2)

- Understand concept of processes running on different machines
- Communication of processes using sockets / SSL
- Introduce concept of a master-slave architecture for centralized control
- Introduce client-server communication using RPC
- Encrypted communication - use of SSL
- Discover / maintain membership of nodes
- Checking for health and status of nodes
- Understand basic notions of fault tolerance where the fault model includes node crashes including a critical master node or slave nodes, communication failures

Lab 1: Discover nodes and status to create a group / cluster

- A set of nodes when added to a cluster need to discover other nodes and form a group that can communicate, track membership status and health.
- Assume a master-slave architecture and designate a master node that can drive this status collection and state maintenance. Slaves send heartbeats to master. Why shouldn't the master initiate this? Helps with scale / fault tolerance.
- Master collects resource status on slaves, current utilization etc. Master detects a slave crash and let’s others know.
- Demonstrate use of multi-node communication to implement a simple cluster management protocol for membership and health / resource status.
- Use secure transport mechanisms (SSL) so that nodes cannot impersonate. Use basic secure socket programming, RPC etc.
- Slaves can crash and rejoin.

Module 2: Distributed processing (Sessions 3-5)

- Introduce concepts of data partitioning and parallel computation across multiple nodes
- Higher level communication models built on top of sockets - blocking and non-blocking, fault tolerance in communication (dropped packets, connection failures)
- Using OpenMPI to implement the communication layer
- Communication cost - bandwidth, latency, payload size
- What is throttling / flow control and where is it useful
- Payload compression to reduce size at increased processing cost

- Synchronization problems, Deadlocks, Mutex/locks between communication processes
- Basic fault tolerance - handling node crashes, communication failures, restart computations fully or partially
- Basic resource usage and performance data collection
- Work allocation by master - Finding the nodes to distribute work based on resource availability and performance etc. (Basic capacity and resource management)

Lab 2: Use cluster to distribute work and do some parallel data processing

- Now that you have a group of nodes managed as a cluster, create a distributed data processing capability. Master node takes a dataset and distributes / partitions the data on multiple active slaves that have enough resources. Slaves process the data and send results back to Master. Master merges / reduces the data and reports back to the user.
- Use blocking vs non-blocking messages in OpenMPI.
- Status of the work at slaves is reported back to the master as work progresses and finishes. Master collects the performance data (CPU, mem, IO) and stores it for analysis. Any errors are also reported to the master.
- Use compression to transfer data. Quantify communication cost - bytes sent over the network, latency of transfer.
- Use OpenMPI for communication between nodes, zlib etc. compression libs
- Handle a node crash - Master should get work done on remaining nodes. What are the options - should it restart the job or only shift part of the work to another node ? How is a slave crash detected ? Use the heartbeat mechanism.

Module 3: Multi-threading for single node performance (Sessions 6-7)

- Multi-core/SMP architectures
- Multi-threaded programs
- Thread synchronization - locks
- Thread pools / management, scheduling
- Latency hiding using threads
- Reduce I/O - buffering, pre-fetching, caching
- Performance measurements
- Converting a sequential program into a multi-threaded program (using OpenMP)
- Introduce thread programming using Java / OpenMP directives

Lab 3: Enhance performance using multithreading within one node

- Use multithreading to improve Lab 2 code on slave nodes. Idea is to improve performance using multiple cores on the same node and show synchronization problems/solutions.

Lab 4: Converting a sequential program to a multi-threaded version

- Use openMP to improve Lab 2 code on slave nodes, compare pthreads and openMP code performance. implement mutex and deadlock prevention

Session 8 is reserved for revision

Module 4: Scale-out and redundancy (Sessions 9-11)

- Introduce P2P architecture to remove dependency on master nodes, especially in large scale systems
- Basic hashing techniques for mapping data and nodes
- Concepts of scale-out clusters, adding/removing nodes on the fly
- Redundancy - node and data, failure handling and checkpointing
- Advanced communication models - broadcast, reduce / scatter-gather
- Bulk data transfer for replication
- Towards development of MapReduce system

Lab 5: Explore multi-node programs but for larger scale

- As more nodes are added to the cluster, the master node becomes a bottleneck to run all operations, esp. on the data path.
- The initial data partitioning and result gathering is done by the master but data replication for redundancy needs to avoid master involvement.
- A P2P method is adopted where slaves replicate data to each other without consulting the master. Use a hash of slave IP address and data partition ID to map replicas. The hash is used to locate replicas during read. Master does not store any meta-data about replicas.
- Check-point your program (on each node individually): user-level check-pointing with libraries. Simulate process/node failures and recover based on checkpoints.

Module 5: Massive multi-core programming (Sessions 12-13)

- Introduce basic concepts of massive multi-core GPU architectures
- Use cases in parallel programming
- Basic concepts of programming libraries like CUDA

Lab 6: GPU based programs

- Use CUDA library to Demonstrate use of GPUs for massive multi-core programming, performance analysis / profiling.

Module 6: Event driven programming (Session 14-15)

- Event-driven architecture
- Queues, pub-sub concepts in communication
- Calling an API vs registering an API as a webhook / callback on an event
    - The basis of Function-as-a-service called on an event condition
- Timing / ordering of events in distributed systems
- Handle cases when events have timeouts
- Cases when events need to be stored, recovered or replayed from history

Lab 7: Event-driven distributed programs

- Assume a web application is built to take data sets from users. Submitted data sets are put in a queue as an event with details, such as location of new data set in a certain file path.
- The master node process checks for any event in the queue whenever it is idle and in presence of an event, invokes the data processing function entry-point developed in Lab 2.
- When a master finishes processing a data set it puts another event in a queue. A finisher process polls for this event and performs the cleanup activity to delete the data files and send the results to the user by email.

Session 16 is reserved for revision

Text Book(s)

T1	Peter S. Pacheco, Matthew Malensek, An Introduction to Parallel Programming (Second Edition), Morgan Kaufmann, 2022
T2	Michael J. Quinn. Parallel Programming in C with MPI and OpenMP. McGraw-Hill Education Group, 2003.
T3	Jason Sanders and Edward Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming (1st. ed.). Addison-Wesley Professional, 2010.
T4	Adam Bellemare, Building Event Driven Systems, Oreilly, 2022

Reference Book(s) & other resources

R1	Michael J. Quinn. Parallel Computing: Theory and Practice. McGraw-Hill Education Group, 1994.
R2	Ajay D. Kshemkalyani and Mukesh Singhal, Distributed Computing, Principles, Algorithms and Systems, Cambridge, 2021
R3	- PI-1: Online tutorial "MPI Complete Reference" - PI-2: Online tutorial: MPI-2: Extensions to the Message-Passing Interface

Evaluation Scheme:

Evaluation Component	Name (Quiz, Lab, Project, Midterm exam, End semester exam, etc)	Type (Open book, Closed book, Online, etc.)	Weight	Duration	Day, Date, Session, Time
EC – 1	Quizzes / Assignment	Online	30%	NA	To be announced
EC – 2	Mid-term Exam	Closed Book	30%	2 hrs	22/03/2025 (FN)
EC – 3	End Semester Exam	Open book	40%	2 ½ hrs	24/05/2025 (FN)

Note *- Evaluation components can be tailored depending on the proposed model.*

Notes:

Syllabus for Mid-Semester Test (Closed Book): Topics in Session Nos. 1 to 8 (contact hours 1 to 16). Any reference materials such as books, notes, or slides will NOT be permitted during the examination.

Syllabus for Comprehensive Exam (Open Book): All topics

Important links and information:

Elearn portal: https://elearn.bits-pilani.ac.in

Students are expected to visit the Elearn portal on a regular basis and stay up to date with the latest announcements and deadlines.

Contact sessions: Students should attend the online lectures as per the schedule provided on the Elearn portal.

Evaluation Guidelines:

1. EC-1 consists of either two Assignments or three Quizzes. Students will attempt them through the course pages on the Elearn portal. Announcements will be made on the portal, in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted.
3. For Open Book exams: Use of books and any printed / written reference material (filed or bound) is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam which will be made available on the Elearn portal. The Make-Up Test/Exam will be conducted only at selected exam centres on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self study schedule as given in the course handout, attend the online lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.