**NAME: KOLAWOLE ENIOLA EUNICE**

**MATRIC NO.: DU0441**

**PROGRAM: COMPUTER SCIENCE**

**COURSE: CSC 421 (NET-CENTRIC COMPUTING)**

**CLASS EXERCISE**

Write on the topic **PARALLEL PROGRAMMING MODELS** covering definition, types, and Concepts like Message Passing Interface (MPI), OpenMP (Open Multi-Processing), MapReduce, OpenCL (Open Computing Language), and CUDA (Compute Unified Device Architecture) programming model.

**WHAT IS PARALLEL PROGRAMMING MODEL?**

**Parallel Programming Model** is an abstraction that defines how computational tasks are structured and executed concurrently across multiple processing units. It provides a framework for parallel programs, specifying how tasks communicate, share data, and synchronize with each other.

**Parallel Programming Model** can be implemented on different hardware architectures, such as multi-core CPUs, GPUs, and distributed computing clusters.

**WHAT IS PARALLEL PROGRAM?**

A **parallel program** is a computer program designed to execute multiple tasks simultaneously by dividing computations among multiple processors, cores, or computing units. The goal is to improve performance, reduce execution time, and efficiently utilize hardware resources.

**KEY CHARACTERISTICS OF PARALLEL PROGRAMMING MODELS**

* **Concurrency;** Multiple computations happen simultaneously.
* **Synchronization;** Mechanisms (locks, barriers) ensure proper coordination.
* **Communication;** Defines how tasks share data (shared memory or message passing).
* **Scalability;** Supports increasing workloads efficiently.

**TYPES OF PARALLEL PROGRAMMING MODELS**

1. **Shared Memory Model:** In this model, multiple processors share a single memory address space and communicate by reading and writing to shared variables. Synchronization mechanisms such as mutexes, semaphores, and locks are used to prevent conflicts when multiple threads access the same memory.

**ADVANTAGES**

* **Fast Communication;** No need for explicit message passing.
* Easier to program than distributed memory systems.
* Efficient for multi-core architectures.

**DISADVANTAGES**

* **Race Conditions;** If multiple threads modify the same data without proper synchronization, results may be unpredictable.
* **Limited Scalability;** Performance decreases as more processors compete for shared memory access.
* **Synchronization Overhead;** Managing memory access requires additional processing.

**Examples are;** Open MP (Open Multi-Processing), Pthreads (POSIX Threads), and Java threads

It is used for; Multi-core processors, Operating systems, High-Performance Computing (HPC).

1. **Distributed Memory Model:** In this model, each processor has its own private memory, and communication happens via message passing. Processors must send and receive messages to share information.

**ADVANTAGES**

* **Highly Scalable;** Suitable for large-scale distributed computing.
* **Avoids Memory Contention;** Since processor has its own memory, there is no competition for shared memory.

**DISADVANTAGES**

* **High Communication Overhead;** Sending messages between processors adds latency.
* **More Complex Programming;** Explicit message-passing requires additional code.

**Example are;** MPI (Message-Passing Interface), Apache Spark (for big data processing), Hadoop MapReduce.

**It is used in;** Supercomputers, Cloud Computing, and Large-Scale Simulations.

1. **Hybrid Model (Shared + Distributed Memory):** This model combines shared memory (within a node) and distributed memory (between nodes) to maximize efficiency and scalability.

**ADVANTAGES**

* **Best of both worlds;** combines efficient shared memory communication with scalable distributed memory.
* Optimized for **High-Performance Computing** (HPC).

**DISADVANTAGES**

* **More Complex Programming:** Requires handling both threading and message passing.

**Examples are;** Hybrid MPI + OpenMP code (c).

**It is used in;** Supercomputers, Large-Scale Simulations (weather forecasting, AI training).

1. **Data Parallel Model:** In this model, a single instruction is executed across multiple data elements.

**ADVANTAGES**

* Highly efficient for data-intensive tasks.
* Maximizes hardware utilization (ideal for GPUs)

**DISADVANTAGES**

* Not suitable for irregular tasks; works best when operations are uniform across data.

**Examples are;** CUDA (for NVIDIA GPUs), OpenCL (for heterogeneous computing).

**It is used in;** Machine learning, Image processing, and Financial Modelling.

1. **Task Parallel Model:** Different tasks are executed in parallel rather than operating on the same data set. Each processor may perform a different task simultaneously.

**ADVANTAGES**

* Best for dividing workloads into independent subtasks.

**DISADVANTAGES**

* Hard to manage task dependencies.

**Examples are;** Intel Threading Building Blocks (TBB), JavaFork, Join Framework.

**It is used in;** Multi-Threaded applications, and Webservers.

1. **Pipeline (Stream) Parallel Model:** Tasks are broken into stages, and different stages execute in parallel. **Examples are;** Tensorflow (Machine Learning), Apache Kafka (Real-time streaming).
2. **Speculative Parallelism:** Predicts future computations and executes them in advance. **Examples are;** CPU Branch Prediction, Transactional Memory.