Vidyalankar Institute of Technology — Accredited A+ by NAAC	**DEPARTMENT OF COMPUTER ENGINEERING**

PBLE 1

Semester	B.E. Semester VIII – Computer Engineering
Subject	Distributed Computing Lab
Subject Professor In-charge	Dr. Umesh Kulkarni
Assisting Professor	Prof. Prakash Parmar
Academic Year	2024-25

Student Name	Deep Salunkhe
Roll Number	21102A0014

Title: Efficient Scheduling in a Distributed Cloud Computing Platform.

Introduction

A distributed cloud computing platform consists of multiple geographically distributed nodes that handle user requests for computational tasks, such as data analysis, machine learning training, and video rendering. To ensure optimal performance, the system must effectively manage resource allocation while balancing workloads, minimizing costs, and maintaining fairness.

However, challenges arise when:

1. Some nodes become overloaded while others remain underutilized.
2. High-priority tasks experience delays due to resource allocation to lower-priority tasks.
3. Task dependencies across nodes lead to inefficiencies and execution delays.

To address these challenges, an efficient scheduling algorithm is necessary.

Problem Statement

The objective is to design a scheduling algorithm that:

1. **Balances workloads** across distributed nodes to maximize resource utilization.
2. **Ensures fairness** while prioritizing critical tasks.
3. **Handles task dependencies** and reduces communication overhead.
4. **Adapts dynamically** to workload and resource availability changes.

Challenges in Distributed Scheduling

1. Load Balancing

- Some nodes may be overloaded while others are underutilized.
- Resource heterogeneity leads to uneven performance across nodes.
- Dynamic workloads require real-time adaptation.

2. Task Prioritization

- High-priority tasks should not be delayed due to lower-priority ones.
- Preemptive scheduling is needed to reassign tasks dynamically.
- Fair allocation is required to prevent resource starvation.

3. Task Dependency and Communication Overhead

- Some tasks rely on the output of other tasks, creating dependencies.
- Cross-node communication introduces latency.
- Inefficient task placement increases dependency resolution time.

4. Dynamic Adaptation

- Resources may become unavailable or new nodes may join the system.
- Varying workloads require a flexible scheduling approach.
- Failure recovery and fault tolerance must be considered.

Proposed Scheduling Algorithm

A **Hybrid Adaptive Scheduling Algorithm (HASA)** is proposed to efficiently schedule tasks across distributed nodes. It consists of four key components:

1. Load-Aware Task Distribution

- Uses a **Load Balancer** that continuously monitors CPU, memory, and network usage across all nodes.
- Employs a **Weighted Least-Loaded First (WLLF)** strategy:
    - Assigns tasks to the node with the lowest load, weighted by available resources.
    - Reduces congestion by spreading tasks across nodes.
- Implements a **Task Migration Mechanism** to move tasks from overloaded to underutilized nodes.

2. Fairness and Priority-Based Scheduling

- Utilizes a **Priority Queue (PQ) Scheduling Model**:
    - High-priority tasks (e.g., real-time requests) get immediate access.
    - Lower-priority tasks are queued with aging to prevent starvation.
- Preemptive scheduling:
    - If a high-priority task arrives, lower-priority tasks can be temporarily suspended or migrated.

3. Dependency-Aware Execution

- Implements a **Directed Acyclic Graph (DAG)-Based Task Scheduling** approach:
    - Breaks tasks into a dependency graph where nodes represent tasks, and edges represent dependencies.
    - Uses **Topological Sorting** to determine execution order.
    - Prefers scheduling dependent tasks on the same node when possible to minimize communication overhead.

4. Dynamic Adaptation Mechanism

- **Auto-Scaling:**
    - If workload increases, new nodes are automatically provisioned.
    - If workload decreases, unused nodes are put into standby mode.
- **Fault Tolerance:**
    - Failed tasks are re-executed on alternate nodes using checkpointing.
- **Reinforcement Learning (RL) Optimization:**
    - The system continuously learns optimal task distribution based on past performance metrics.

Implementation Considerations

1. **Algorithm Complexity:** Ensuring efficient scheduling decisions in real-time requires optimization techniques like heuristic-based search and reinforcement learning.
2. **Communication Overhead:** Techniques like data locality-aware scheduling can minimize inter-node dependencies.
3. **Security and Fault Tolerance:** Implementing redundant task execution and encrypted inter-node communication enhances reliability and security.

Conclusion

By integrating **load-aware distribution, priority scheduling, dependency management, and dynamic adaptation**, the proposed **Hybrid Adaptive Scheduling Algorithm (HASA)** optimizes resource utilization, ensures fairness, and efficiently handles dependencies in a distributed cloud computing platform. This approach improves overall system performance while reducing costs and execution delays.