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

Assignment No. 06

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: Global Scheduling Algorithm in Distributed Computing

1. Introduction

In a distributed computing environment, efficient scheduling is crucial to optimize system performance while managing trade-offs between task assignment, load balancing, and load sharing. A global scheduling algorithm must ensure equitable resource utilization, minimize task execution time, and accommodate dynamic process migration and code migration. This paper explores how global scheduling algorithms balance these factors to enhance overall system efficiency.

2. Key Trade-offs in Global Scheduling Algorithms

2.1 Task Assignment

Task assignment refers to allocating computational tasks to appropriate processors. An efficient algorithm should consider factors such as processing power, network latency, and data locality. The challenge lies in ensuring optimal placement without overloading specific nodes.

2.2 Load Balancing

Load balancing ensures that all processors in a distributed system handle an approximately equal share of the workload. This prevents bottlenecks and idle resources. A scheduling algorithm must dynamically adapt to workload variations while minimizing migration overhead.

2.3 Load Sharing

Unlike load balancing, which aims for equal workload distribution, load sharing focuses on improving resource utilization by offloading tasks from overloaded nodes to underutilized ones. Effective load-sharing mechanisms reduce overall response time while preventing unnecessary migrations.

3. Accommodating Dynamic Process Migration and Code Migration

3.1 Process Migration

Process migration involves transferring a running process from one node to another to achieve better resource utilization. A global scheduling algorithm must decide when and where to migrate processes based on real-time system conditions. Factors such as process dependencies, execution state, and communication overhead must be considered.

3.2 Code Migration

Code migration allows segments of a program (e.g., functions or objects) to move across nodes for execution. This is beneficial in environments where lightweight tasks need to be executed closer to data sources or where nodes have specialized capabilities. Scheduling algorithms should determine when code migration is beneficial, minimizing transfer costs while optimizing execution efficiency.

4. Optimization Strategies for System Performance

4.1 Adaptive Scheduling

Adaptive scheduling techniques dynamically adjust task allocation based on real-time system conditions. Machine learning models and heuristic approaches, such as genetic algorithms and reinforcement learning, can improve scheduling decisions.

4.2 Work Stealing Mechanisms

Work stealing enables underloaded nodes to pull tasks from overloaded ones, improving system responsiveness. This strategy is particularly useful in decentralized scheduling models.

4.3 Hierarchical Scheduling

Hierarchical scheduling employs multiple levels of decision-making, with local schedulers managing individual nodes and a global scheduler coordinating tasks across the entire system. This approach balances scalability and control.

5. Conclusion

A global scheduling algorithm must carefully navigate the trade-offs between task assignment, load balancing, and load sharing while accommodating dynamic process and code migration. By incorporating adaptive scheduling, work-stealing mechanisms, and hierarchical approaches, distributed computing systems can achieve optimal performance with minimal overhead. Future advancements in AI-driven scheduling strategies hold the potential to further enhance efficiency and resource utilization in complex distributed environments.