SMVS
Page No.
Date :

Load Balancing and Termination Detection

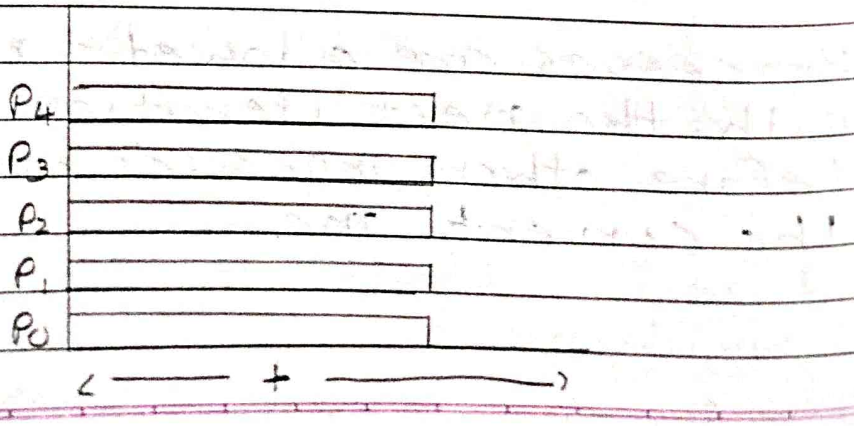
1 What is Load Balancing and how load balancing will produce minimum exection time?

=> Load Balancing:

Load Balancing is the distribution of tasks or computational workload evenly across multiple processors in parallel processing system.

The goal is to ensure that no processor remains idle, while others are still working.

It ensure that all processors contribute to the computation equally.



It helps to improve the performance of parallel systems by minimizing the execution time.

=> How Load Balancing Produces Minimum Execution Time:

(i) Even Distribution of Work: Tasks are assigned to processors in such a way that the workload is evenly distributed.

(ii) Avoiding Processor Idle Time: Without Load Balancing processor may finish their assigned tasks early and remain idle, Leading to inefficient use of resources.

(iii) Redistribution of Tasks: Load Balancing redistributes work from busier processors to idle ones, ensuring equal processing time.

(iv) Minimizing Total Execution Time.

(v) Dynamic Load Balancing

(vi) Reducing Communication Delays.

SMVS
Page No.
Date : / /

2 Explain Static and Dynamic Load Balancing.

=> Static Load Balancing:

Static Load Balancing involves distributing tasks or work across processors before the execution begins, without considering the runtime conditions.

The task assignment is predetermined and does not change during execution.

Tasks are assigned to processors at the beginning based on estimates of execution time.

Once tasks are distributed, the workload remains fixed throughout the execution.

- Optimization Techniques: Round Robin, Randomized Algorithm, Recursive Bisection or Genetic Algorithm

SMVS
Page No.
Date : / /

-> Limitations:

1 Difficulty in predicting exact execution time.

2 Real time communication delays.

3 Unevenly distributed task by predicting wrong time.

=> Dynamic Load Balancing:

Dynamic Load Balancing adjusts the distribution of tasks during execution.

Distribute task based on the ~~at~~ actual progress and workload for each processor.

This approach continuously monitors and reallocates work to ensure that processors remain balanced throughout the execution.

If some processors finish their work early, tasks can be reassigned to them from processors that are still busy.

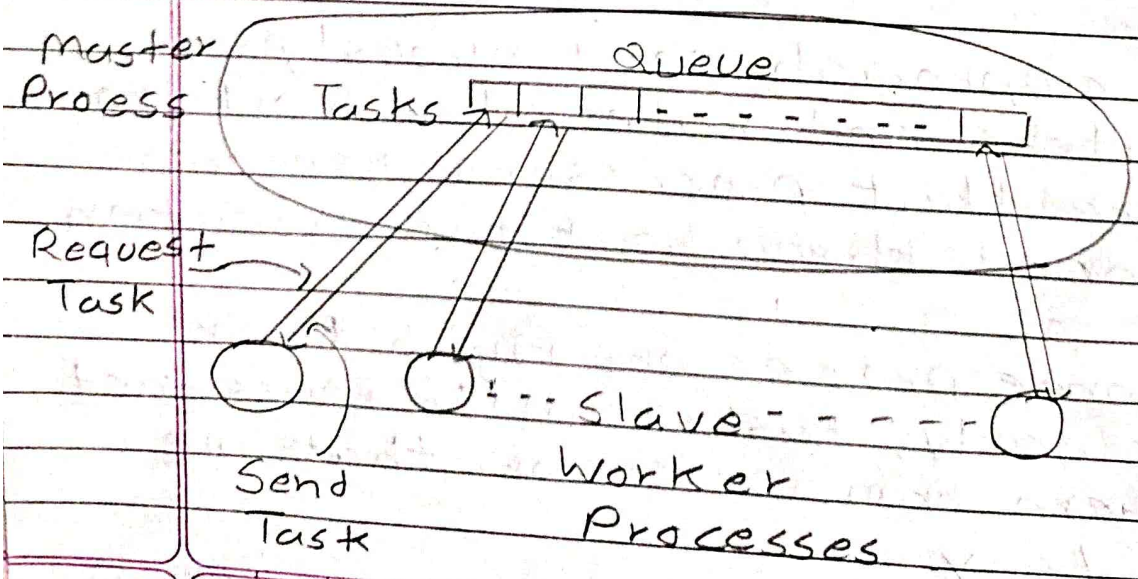
There are two Primary method for Implementing Dynamic Load Balancing.

1) Centralized
2) Decentralized

1 Centralized Dynamic Load Balancing:

In Centralized Dynamic Load Balancing, a master process is responsible for managing all tasks.

The task are stored in a central work pool and slave processes request tasks from the master when they complete their current task.



SMVS
Page No.
Date : / /

Work Pool : A Central Queue holds tasks.

Task Assignment : Tasks are handed out by the master to the slaves upon request.

Termination : The master detects Termination when,
(1) The task queue is empty.
(2) All slaves are idle.

-> Advantages :

1 Simple to implement.
2 Effective for a smaller number of slaves.

-> Disadvantages :

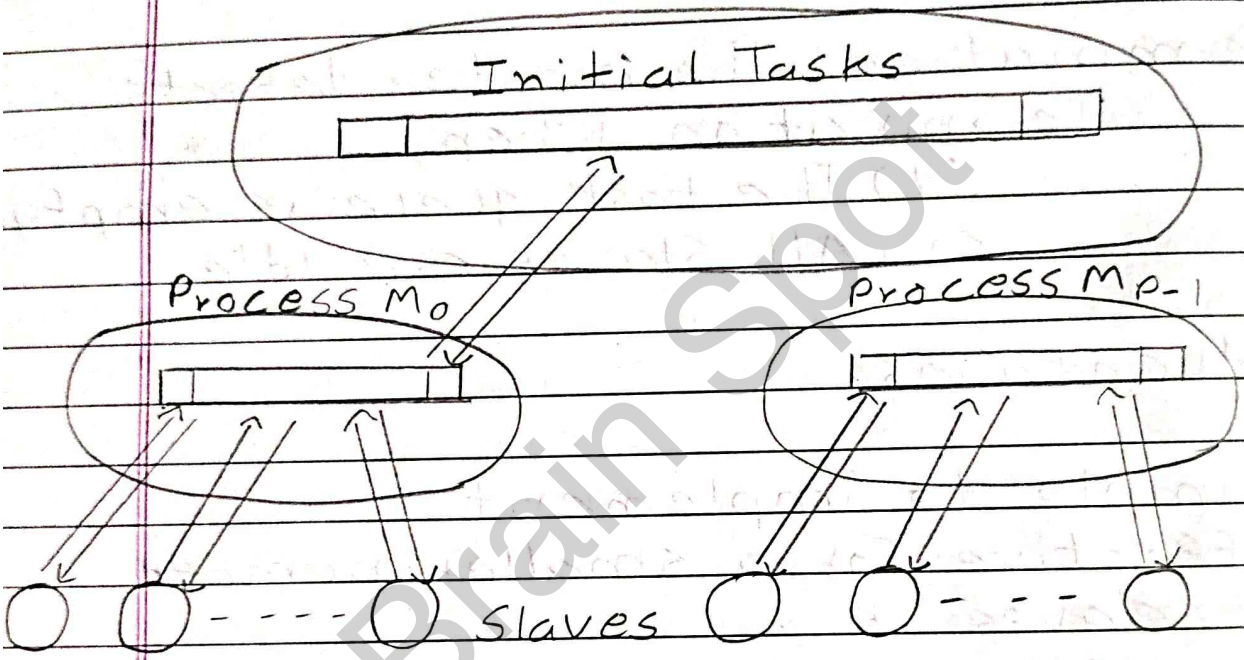
1 Master can handle only one tasks at a time.
2 Limited Scalability

2 Decentralized Dynamic Load Balancing :

In Decentralized Dynamic Load

balancing, the responsibility of managing tasks is distributed among multiple processes.

This eliminates the single master and enables better scalability.



Distributed Work Pool : Tasks are distributed across Processes.

Task Sharing : Two Ways :

(1) Receiver - Initiated : A process requests tasks from other processes when it becomes idle.

(2) Sender - Initiated : A heavily Loaded process distributes some of its task to others.

Page No.
Date : / /

Termination : Can involve processes detecting local or global condition for termination.

-> Advantages :

1 No Center Master Process.
2 Scalability to a larger number of processes.

-> Disadvantages:

1 More complex implementation
2 Communication Overhead.

3 Termination :

Termination in distributed system involves ensuring that a computation or set of tasks distributed across processes has concluded.

Recognizing termination can be challenging due to delays and communication complexities.

There are two main Termination Condition :

1) Local Termination Conditions:
Every process must complete its tasks.

2) No Message in Transit:
There should be no pending message between processes.

=> Ring Termination Detection Algorithm:

The Ring Termination Algorithm Organizes processes in a ring structure to detect global termination.

There are two method in Ring Termination Detection Algorithm:

(1) Single Pass Termination Algorithm:

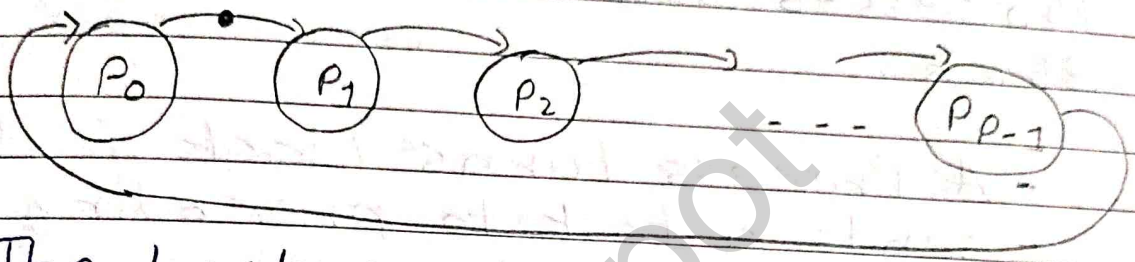
-> Steps:

1 The First Process P_0 generates a token when it has terminated and passes it to P1.

2 Each Process Pi passes the token

to the next process if it has already terminated.

Otherwise it waits for its local termination condition before passing the token.



3 The Last process P_{p-1} passes the token back to P_0.

4 When P_0 receives the token, it confirms that all processes have terminated.

A Global termination message can then be broadcast to all processes.

(2) Dual-Pass Ring Termination Algorithm:

It uses a two-pass approach with colored tokens (White and Black).

→ Steps:

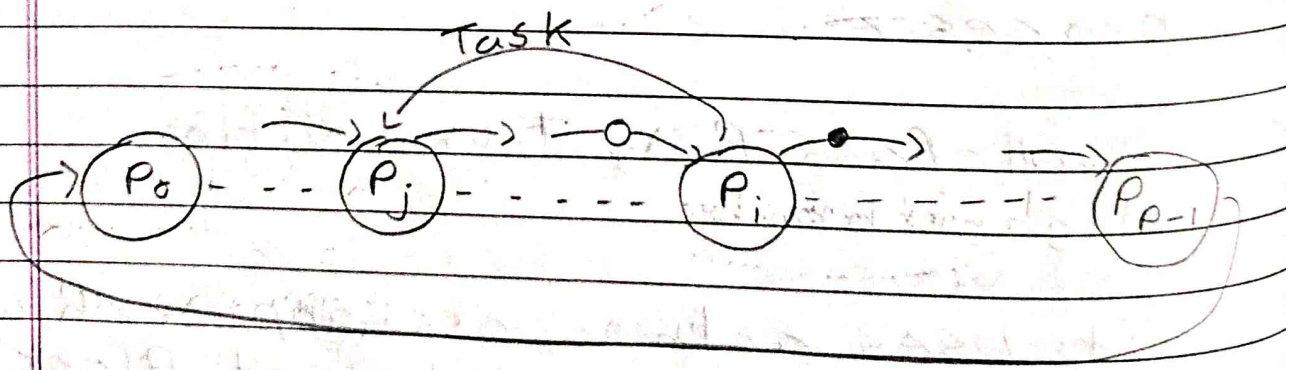
1 P_0 generates a white token and passes it to P_1 when it terminates.

2 Processes are colored white or Black:

- A Process turns black if it sends a task to preceding process in the ring.

- A Black Process passes a black token and a white process passes the token in its current color.

After passing the token, the process resets to white.



3 If P_0 receives a black token, it knows termination has not occured and generates another

Date : / /

white token.

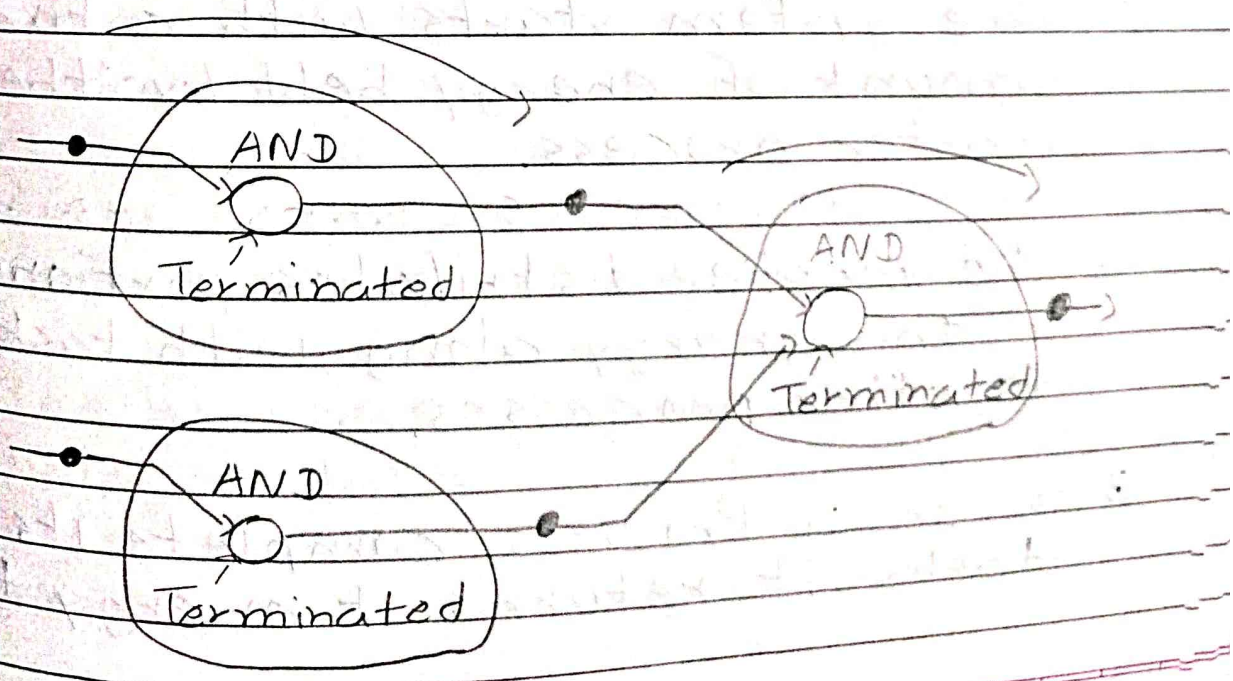
If P_0 receives a white token, global termination is confirmed.

=> Tree Termination Detection Algorithm.

The tree termination detection algorithm uses a hierachical tree structure.

-> Steps :

1 Each process waits for local termination conditions and token from its children.



2 When all tasks are completed and token from all branches are received, the process pass its token to its parent.

3 The root process confirms global termination when it receives all token and completes its tasks.

=> Fixed Energy Distributed Termination Algorithm:

It is used concept called "Energy".

-> Steps:

1 The system starts with a fixed amount of energy held by the master process.

2 The master distributes portions of this energy along with tasks to other processes.

3 When a Process complete its tasks, it returns it energy to

its master process.

4 The System achieves global termination when all energy is returned to the master process and the master process is idle.

4 Explain Shortest Path Problem in respect to Load Balancing and Termination Detection.

=> The shortest path problem involves finding the path with minimum total weight between two nodes in a graph.

-> Load Balancing:

In distributed computing, Load Balancing ensures that computational tasks are evenly distributed across nodes to optimize resource utilization.

The shortest path problem can help to determine the most efficient way to transfer workloads between nodes.

- Graph Representation:

Nodes: Represent Computational Units or server.

Edges: Represent communication channels between nodes.

Weights: Represent the cost of transfering tasks.

By Finding the shortest path From an overload node to an underutilized one.

This ensures that tasks are transfered along the most efficient route.

→ Termination Detection:

In distributed systems, termination detection involves determining when all nodes have completed their tasks.

- Graph Representation:

SMVS
Page No.
Date : / /

Nodes : Represent Processes

Edges : Represent Communication between processes

Weights : Represent Time taken for messages to be processed.

If a process detects its task is complete, the shortest path algorithm can determine the most efficient way to notify dependent processes.

5 Explain Moore's Algorithm with example

=> Moore's Algorithm is a graph traversal algorithm designed to find the shortest path.

It uses a First-In-First-Out queue for its operation.

-> Algorithm Steps :

1 Initialization : Set all distance to infinity except the source vertex, whish is set to 0.

Date : / /

Initialize a queue with source vertex.

2 Processing the Queue:

While the Queue is not empty, dequeue a vertex i.

For each neighboring vertex j, calculate $dj = \min(dj, di + W_{ij})$

If dj is updated, enqueue vertex "j".

3 Termination:

The algorithm ends when no vertices remain in the queue.