UNIT- 4

Date ___/___/___
Page ______

DISTRIBUTED SCHEDULING AND DEADLOCK

DISTRIBUTED SCHEDULING -

(1) Distributed scheduling refers to the ~~execution of non interactive~~ chaining of different jobs into a coordinated workflow that spans several computers.

(2) Issues in Load Distributing -

(1) Load -

Load estimation is ~~called~~ calculated by using resource queue lengths and CPU utilization.

Queue length of waiting task proportional to task response time, hence a good indicator of system loads.

| Distributed load = transfer tasks / process among nodes |

If a task transfer (from another node) takes a long time, the node may accept more tasks during the transfer time causes the node to be highly loaded and affect performance.

Solution - Artificially increment the queue length when a task is accepted for transfer from remote node.

(2) Types of algorithm -

Basic function of a load distributing algorithm is to transfer load (task) from heavily loaded computers to idle or lightly loaded computers. It can be characterized as -

(i) Static load distribution algorithm - Decisions are hard coded into an algorithm with a priori knowledge of system.

(ii) Dynamic load distribution algorithm - Use system state information such as task queue length, processor utilization

(iii) Adaptive load distribution algorithm - adapt the approach based on system state.

Adaptive stop collecting information at high load but dynamic still collects information at high load.

(3) Load Balancing Vs Load Sharing -

Load Balancing algorithm is also ~~co~~ classified as load balancing and load sharing.

Add in Selection policy -
Another approach is that a task is selected for transfer only if its response time will be improved upon transfer

Date ___/___/___
Page ______

Load balancing - More no. of task transfers → degrade performance
Load balancing algorithm go a step further by attempting to equalize loads at all computers (participating nodes)
Transfer tasks even if a node is not heavily loaded so that queue lengths on all nodes are approximately equal.
Load Sharing - Less no. of task transfer
Reduce burden of an overloaded node by task transfers to idle or lightly loaded nodes. This task transfer is known as anticipatory task transfer
Transfers tasks only when the queue length exceeds a certain threshold.
(4) Preemptive Vs Nonpreemptive transfers -
Preemptive task transfers involve the transfer of a task that is partially executed which is expensive as it involves collection of task states such as virtual memory image, process control block, IO buffers etc.
Nonpreemptive task transfer involve the transfer of a task that has not begun execution (that means no task states transfer required). It can be considered as task placements. Suitable for load sharing not for load balancing.

(3) Components of load distributing algorithms -
(1) Transfer policy -
Determines when a node is ready to participate in a task transfer.
When a load on a node exceeds a threshold T, the node becomes a sender. When it falls below a threshold, it becomes a receiver.
(2) Selection policy -
Determines which task should be transferred.
Approach - Select newly originated tasks because transfer cost is lower as no state information is to be transferred. Non-preemptive transfers are allowed.
Factors of selection - (1) smaller task have less overhead, small response time
(2) Location - dependent system calls should be minimal.
(3) Location Policy -
Determines the receiving node for a task.
Polling is generally used which can be done serially or in parallel (using multicast

Date ___/___/___

Page ________

Alternative - broadcasting a query, sort of invitation to share load.

(4) Information Policy -

Responsible for triggering the collection of system state information.

Demand-driven collection - Only when a node is highly or lightly loaded.

- Sender initiated policies → sender looks for receivers to transfer their load.
- Receiver initiated policies → receiver solicit load from senders
- Symmetric initiated policies → combination of sender & receiver initiated policies

Periodic - Do not adapt to system state, are slow to respond, and can make the situation worse by increasing system load.

State-change driven - Only when state changes by certain degree.

④ Different types of load distributing algorithms -

Four types of load distributing algorithms are -

(1) Sender-initiated algorithms -

Load distributing activity is initiated by an overloaded node (sender) that attempt to send a task to an underloaded node (receiver).

→ Transfer Policy - CPU queue threshold T for all nodes. Initiated when a new task arrives.

→ Selection Policy - Once the transfer policy decides that a host is a sender, a ~~section~~ selection policy selects a task ~~for~~ for transfer.

The simplest and popular approach is to select the newly arrived task for transfer that just transforms the host into a sender.

→ Location Policy - One of the major tasks of location policy is to check the availability of the service(s) required for proper execution of the migrated and/or re-scheduled task(s) within the selected transfer partner.

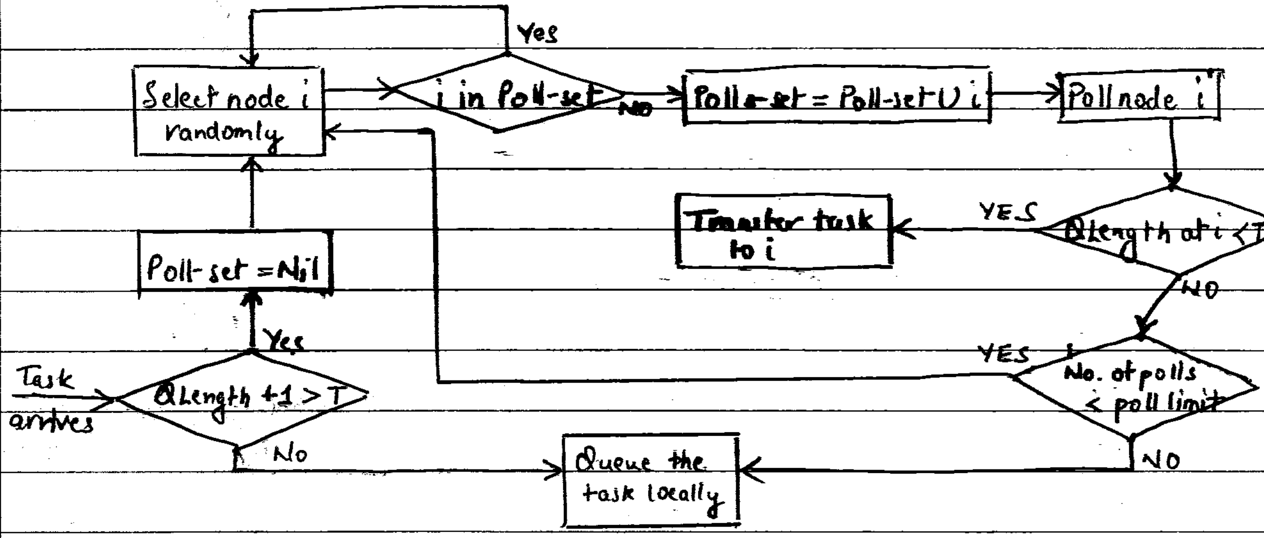
(1) Random - Select any node to transfer the task at random. The selected node X may be overloaded. If transferred task is treated as new ~~task~~ arrival, then X may transfer the task again. Limit the no. of transfers for a task.

It is effective under light-load conditions.

(2) Threshold - Poll nodes until a receiver is found. Up to polllimit nodes are polled. If none is a receiver, then the sender commits to the task.

myCOMPANION

Date / /
Page



SENDER INITIATED LOAD SHARING WITH THRESHOLD LOCATION POLICY

(3) <u>Shortest</u> - Among the polled nodes that where found to the receiver, select the one with the shortest queue. Marginal Improvement.

→ <u>Information Policy</u> -

Demand driven policies → Uses decentralized approach

Periodic policies → Either centralized or decentralized approach

State-change driven policies → Either centralized or decentralized approach.

→ <u>Stability</u> - Unstable at high loads (Drawback)

(2) <u>Receiver Initiated Algorithms</u> -

Load distributing activity is initiated from an underloaded node (receiver) that is trying to obtain a task from an overloaded node (sender).

→ <u>Transfer Policy</u> - Same as in components of load distributing algorithm

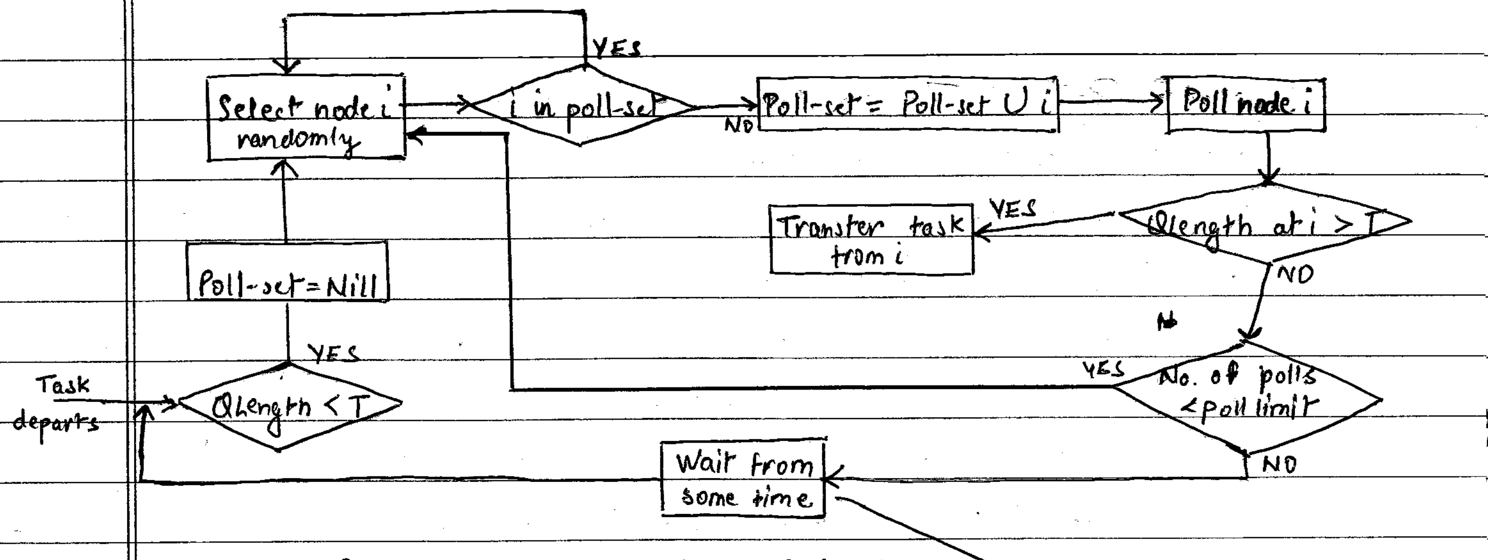
→ <u>~~Section~~ Selection Policy</u> - Any of the approaches can be used that is described in components for load distributing algorithm.

→ <u>Location Policy</u> - A node selected at random is polled to determine if transferring a task from it would place its queue length below the threshold level. If not, the polled node transfer the task. Remaining process are explained in the given diagram of receiver initiated load sharing.

→ <u>Information Policy</u> - Demand driven because polling activity starts only after a node becomes a receiver.

→ <u>Stability</u> - Stable at high loads as well as low loads.
(because of CPU cycles available)
(receiver will find sender with high probability with a small number of polls)

mycompanion



RECEIVER - INITIATED LOAD SHARING

Note that if the search does not start after a predefined period, the extra processing power available at a receiver is completely lost to the system until another task completes, which may not occur soon.

→ Drawbacks - Most ~~task~~ transfers are preemptive and therefore expensive.

(3) Symmetrically initiated algorithms -

Combination of sender-initiated and receiver-initiated algorithms in which senders search for receivers and receivers search for senders.

Low loads - Senders can find receivers easily.

High loads - Receivers can find sender easily.

Drawbacks - Unstable at high load and transfers are preemptive (expensive)

A simple symmetrically initiated algorithm can be constructed by using both the transfer and location policies of sender & receiver initiated algorithms.

Another symmetrically initiated algorithm is given as -

→ Above Average algorithm -

It tries to maintain the load at each node within an acceptable range of the system average.

→ Transfer Policy - Two adaptive thresholds instead of one. If a node's estimated average load is A, a higher threshold [TooHigh > A] and a lower threshold [TooLow < A] is used.

Load < TooLow ⇒ Receiver , Load > TooHigh ⇒ Sender

→ Selection Policy - Any of the approaches can be used described in components for load distributing algorithms.

Date ___/___/___
Page ______

→ Location Policy - Two components -

(i) Sender initiated component -

1> Node with TooHigh load, broadcasts a TooHigh message, sets TooHigh timer, and listen for an accept message

2> A receiver that gets the TooHigh message sends an accept message, increases its load, and sets AwaitingTask Timer

3> If AwaitingTask timer expires, load is decremented.

4> On receiving the Accept message :- If the node is still a sender, it chooses the best task to transfer and transfers it to the node.

5> When sender is waiting for Accept, it may receive a TooLow message (receiver initiated). Sender sends TooHigh to that receiver.

6> On expiration of TooHigh timer, if no Accept message is received, system is highly loaded. Sender broadcasts a ChangeAverage message.

(ii) Receiver initiated component -

1> Node with TooLow load, broadcasts a TooLow message, sets a TooLow timer, and listens for TooHigh message.

2> If TooHigh message is received, do step 2 and 3 in sender initiated component.

3> If TooLow timer expires before receiving any TooHigh message, receiver broadcast a ChangeAverage message to decrease the load estimate at other nodes.

→ Information Policy - Demand driven. Average load is modified based on system load. High loads may have less number of senders progressively.

(4) Adaptive Algorithms -

→ Stable symmetrically initiated algorithms -

It utilizes the information gathered during polling (instead of discarding it as was done by previous algorithms) to classify the nodes in the system as either Sender/overloaded, Receiver/underloaded, or OK.

The knowledge concerning the state of node is maintained by a data structure at each node - a sender list, a receiver list, and an OK list.

Initially, each node assumes that every other node is a receiver.

~~→ Transfer policy - Same as symmetrically initiated algorithm - to OK if LT ≤ ...~~

myCOMPANION

Date ___/___/___
Page ______

→ Transfer Policy - LT → lower threshold UT → upper threshold.

~~Additional is~~ OK if LT ≤ queues length ≤ UT

Sender if its queue length > UT and receiver if its queue length < LT

It is triggered when a new task originates or when a task departs.

→ Selection Policy -

Sender-initiated component considers only newly arrived tasks for transfer

Receiver-initiated component can make use of any of the approaches of this policy

→ Location Policy - Two components -

(i) Sender-initiated component -

① → Sender polls the node at the head of its receivers list to find out whether it is still a receiver

② → The polled node removes the sender node ID from the list it is presently in & puts it in the sender's list

③ → The polled node returns its status (receiver, sender, ok) to the sender.

④ → The sender transfers a task to the node if it is a receiver.

⑤ → The sender puts the polled node in appropriate list based on its reply.

⑥ This process may continue.

⑦ → This ~~poll~~ polling process stops, if suitable receiver is found or if no. of polls reaches a polllimit or if receivers list becomes empty

⑧ If receiver is not found, then the task is processed locally

(ii) Receiver initiated component -

① → The receiver polls the nodes from the first to the last in the sender's list, then it polls the OK list from last to first and then it polls the receiver's list from last to first

② → If the polled node is a sender then it transfers a task and informs the receiver about its status after the task transfer.

③ → If the polled node is not sender then it removes the receiver node ID from the list it is presently in and puts it in the receiver's list and informs the receiver about its status.

④ → The receiver puts the polled node in appropriate list based on the reply.

⑤ This process may continue.

⑥ → Polling process stops, ~~then~~ if sender is found, if receiver is no longer a receiver ~~and~~ or if no. of polls reaches polllimit

mycompanion

Date ___/___/___
Page ______

→ Information Policy - Demand driven, as the polling activity starts when a node becomes a sender or a receiver.

→ Stable sender-initiated algorithm -

Two desirable properties -

(1) Does not cause instability

(2) Load sharing is due to non-preemptive transfers only.

Similar to stable symmetrically initiated algorithm with the ~~modificat~~ modification of receiver initiated component

In this algorithm, Statevector array is used by each node to keep track of which list (senders, receivers or OK) it belongs to at all the other nodes in the system.

Receiver initiated component modified protocol - When a node becomes a receiver it informs all the nodes ~~whose~~ that are misinformed about its current state with the help of statevector at the receiver ~~to~~ to find the misinformed nodes

(5) Task Migration -

Task Migration refers to the transfer of a task that has already begun to a new location and continuing its execution there.

Task placement refers to the transfer of a task that is yet to begin execution to a new location and ~~con~~ start its execution there.

→ Benefits of task migration - load balancing, reduction in communication overhead, resource access and fault tolerance.

→ Steps involved in task migration -

(1) Suspending (freezing) the task on the source.

(2) Extracting and transmitting the state of the task to destination

(3) Reconstructing the state on the destination

(4) ~~Removing~~ Resuming the task's execution on the destination

(6) Issues in Task Migration -

Three issues in task migration are -

(1) State Transfer, (2) Location transparency, (3) Structure of a migration mechanism

Date ___/___/___

Page ______

→ State Transfer - Two important issues are -

as little time as possible

(1) The cost to support remote execution, which includes delays due to freezing the ~~do~~ task, obtaining and transferring ~~a value~~ the state & unfreezing the task.

(2) Residual dependencies - refers to the amount of resources a host of a migrated task continues to dedicate to service request from the migrated task. They are undesirable for three reasons - reliability, performance and complexity.

→ State transfer mechanisms -

(1) Precopying the State → bulk of the task state is copied to the new host before freezing the task

(2) Location-transparent file access mechanism

(3) Copy-on-reference - Just copy what is migrated task need for its execution.

→ Location transparency -

Task migration should high the locations of tasks. Location transparency in principle requires that names (process name, file names) be independent of their locations (host names)

Uniform name space throughout the system.

→ Structure of ~~the~~ a migration mechanism -

Typically, there will be interaction between the task migration mechanism, the memory management system, the inter-process communication mechanism and the file system. The ~~of~~ mechanism can be designed to be independent of one another so that if one mechanism's protocol changes, the other's need not the migration mechanism can be turned off without interfering with other mechanisms.

Date ___/___/___

Page ______

DEADLOCK -

① Issues in deadlock detection and resolution -

→ Detection - Two issues -

maintenance of the WFG (wait-for-graph) and search of the WFG for the presence of cycles (or knots).

Depending upon the manner in which WFG information is maintained and the search for cycles is carried out, there are centralized, distributed and hierarchical algorithms for deadlock detection in distributed systems.

A correct deadlock detection algorithm must satisfy two conditions -

(1) Progress - No undetected deadlocks - it detect all existing deadlock in finite time and progress continuously to find more deadlocks

(2) Safety - No false deadlocks - Should not report deadlocks which are non-existent (phantom deadlocks). Due to no global memory or communication, sites may obtain out of date & inconsistent WFG's of the system.

→ Resolution -

Deadlock resolution involves breaking existing wait for dependencies in the system WFG to resolve the deadlock.

It involves rolling back one or more process that are deadlocked and assigning their resources to blocked process in the deadlock so that they can resume execution.

When a wait for dependency is broken, the corresponding information should be immediately cleaned from the system so that it may not result in detection of phantom deadlocks.

② Deadlock Handling Strategies - Three strategies -

(1) Deadlock Prevention -

It is achieved by either having a process acquire all the needed resources simultaneously before it begins execution or by preempting a process that holds the needed resources.

Drawbacks - ① Inefficient, decreases the system concurrency.
② A set of processes can become deadlocked in the resource acquiring phase.
③ Future resource requirements are unpredictable.

mycompanion

Date ___/___/___

Page ______

(2) Deadlock Avoidance -

A resource is granted to a process if the resulting global system state is safe (a global state includes all the processes and resources of the distributed system).

Drawbacks - ① Every site has to maintain information on the global state of the system → huge storage requirements and intensive communication cost.

② Process of checking for a safe global state must be mutually exclusive → limit the concurrency and throughout of the system.

③ Due to the large no. of processes & resources → expensive to check for a safe state.

(3) Deadlock Detection -

It requires an examination of the states of process - resource interactions for the presence of cyclical wait. Two favorable conditions -

(i) Once a cycle is formed in the WFG, it persists until it is detected and broken

(ii) Cycle detection can proceed concurrently with the normal activities of a system

③ Distributed Deadlock Algorithms -

(1) Centralized deadlock detection algorithm -

(i) Completely centralized algorithm

(ii) The Ho-Ramamoorthy Algorithms

- Two Phase Algorithm
- The One Phase Algorithm.

(2) Distributed deadlock detection algorithm -

(i) Path pushing algorithm

(ii) Edge-chasing algorithm

- Other edge-chasing algorithms - The Mitchell - Merritt Algorithm

(iii) Diffusion Computation based algorithm

(iv) Global State Detection based algorithm

(3) Hierarchical deadlock detection algorithm -

(i). Menasce - Muntz Algorithm

(ii) The Ho- Ramamoorthy Algorithm.

myCOMPANION