UNIT – III

PROCESS AND RESOURCE MANAGEMENT: Process migration: Features - Mechanism. **Resource Management**: Load balancing approach - Load sharing approach

Process Management in a Distributed Environment

Main goal of process management in DS is to make best possible use of existing resources by providing mechanism and policies for sharing them among processors

Concept of Process

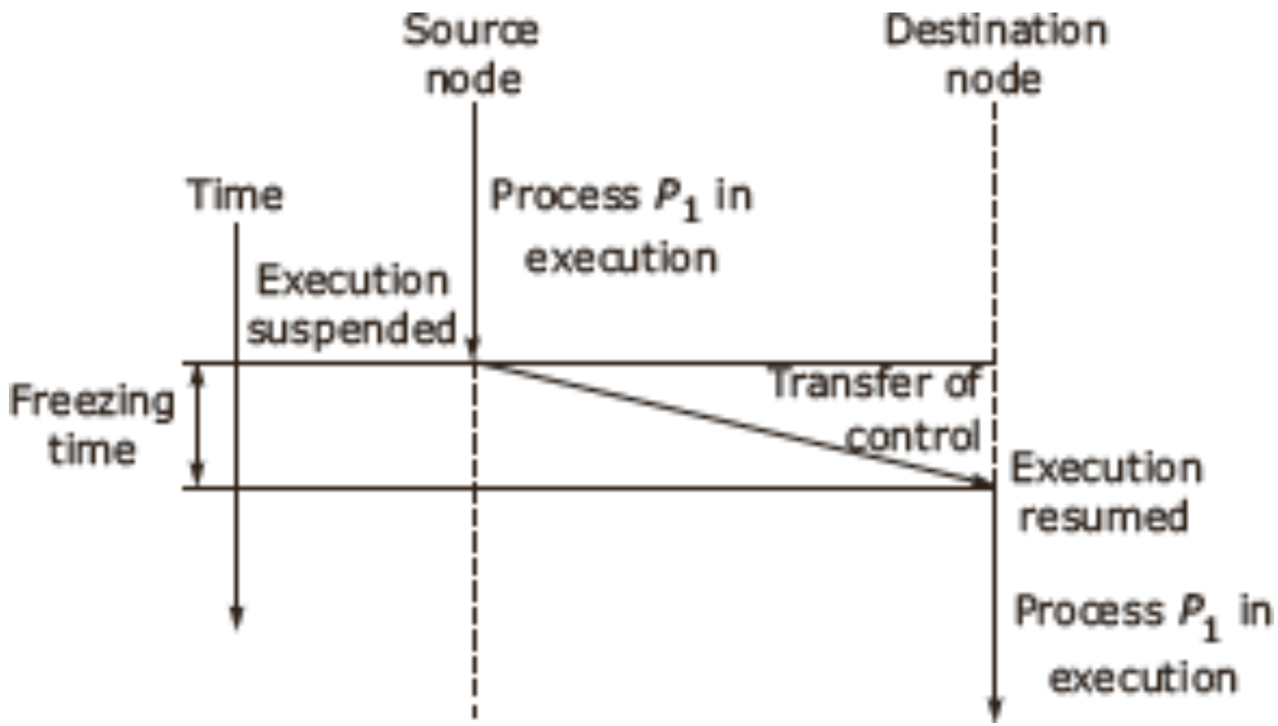
- Process: An operating system abstraction representing an instance of a running computer program."
  - Consists of data, stack, register contents, and the state specific to the underlying OS
  - Can have one or more threads of control
    - Consists of their own stack and register contents, but share a process's address space and signals.

Process management

- Conventional OS: deals with the mechanisms and policies for sharing the **processor of the system** among all processes
- Distributed operating system: To make best possible use of the **processing resources of the entire system** by sharing them among all processes
- **Three concepts to achieve this goal:**
  - **Processor allocation:** Deals with the process of deciding which process should be assigned to which processor
  - **Process migration**: Deals with the movement of a process from its current location to the processor to which it has been assigned
  - **Threads:** Deals with fine-grained parallelism for better utilization of the processing capability of the system

Process Migration

- Process Migration
  - The act of transferring a process between two machines during its execution
    - Relocation of a process from its current location (the source node) to another node (the destination node)
- Goals of Process Migration
  - Dynamic load distribution
  - Fault resilience
  - Improved system administration
  - Data access locality



Process migration mechanism

Process Migration

- Two types:
  - Preemptive process migration
    - Process may be migrated during the course of its execution
  - Non preemptive process migration
    - Process may be migrated before it starts executing on its source node(A process is moved **only when it is not executing**, typically **before it starts or after it completes execution** on a processor)
- Involves three steps:
  - Selection of a process that should be migrated
  - Selection of the destination node to which the selected process should be migrated
  - Actual transfer of the selected process to the destination node

Desirable Features

- **Transparency**
  - Object access level -minimum requirement to support non preemptive process migration- access to the object should be in location independent manner. System must provide mechanism for transparent object naming and locating
  - System call and interprocess communication level- migrating process should not depend on origination node- system calls and IPC should be done in location independent manner.
- **Minimal interference**
  - Migration of a process should cause minimal interference to the progress of the process
  - Can be done by minimizing freezing time
  - **Freezing time:** a time for which the execution of the process is stopped for transferring its information to the destination node
- **Minimal residual dependencies**
  - Migrated process should not continue to depend on its previous node once it has started executing on new node
- **Efficiency**
  - Time required of migrating a process
  - The cost of locating an object
  - The cost of supporting remote execution once the process is migrated

- **Robustness**

  The failure of a node other than the one on which a process is currently running should not affect the execution of that process

- **Communication between coprocesses of a job-** Single job distributed over multiple nodes – to minimize the communication cost co-processes should be able to directly communicate with each other irrespective of location

Process Migration Mechanisms

- Four major sub activities
  - Freezing and restarting the process
  - Transfer of process's address space
  - Forwarding messages meant for the migrant process
  - Handling communication between cooperating processes

Process Migration Mechanisms

- Mechanisms for freezing and restarting a process

General issues involved in freezing and restarting

- **Immediate and Delayed blocking of the process**

  Process may be blocked immediately or delayed

  - if the process is not executing a system call blocked immediately
  - if the process is executing a system call but is sleeping at an interruptible priority waiting for a kernel event to occur, it can be immediately blocked from further execution
  - if the process is executing a system call and sleeping at non interruptible prioritywaiting for a kernel event to occur, it can not be blocked immediately
- Fast and Slow I/O operation
- frozen after the completion of all fast I/O operations
- What about slow I/O operations???

Process Migration Mechanisms

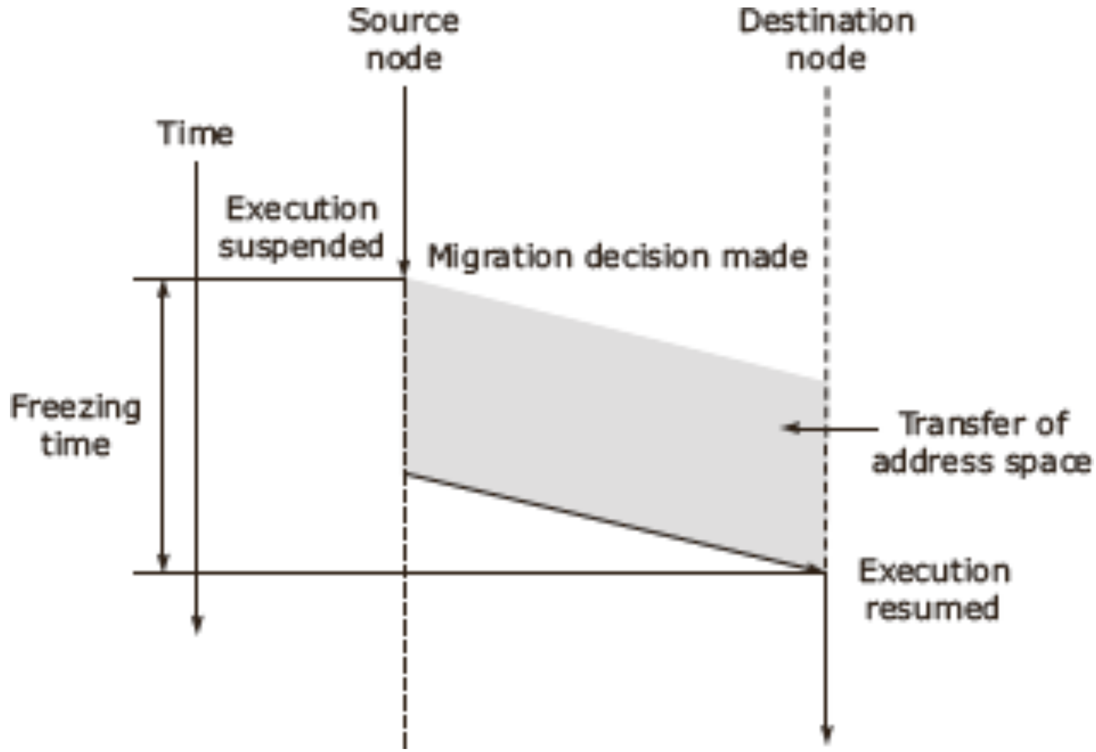
- Mechanisms for freezing and restarting a process
- Information about the open files
  - No problem for network transparent execution environment
  - What about UNIX like systems???
    - creation of link
    - Reconstruction of file's path when required
  - What about frequently used files like commands???
  - What about temporary files?
- Reinstating the process on its destination node
  - Creation of a new process
  - Process identifier
  - What about the process which was blocked while executing a slow system call????

Process Migration Mechanisms

- Address Space Transfer Mechanisms
- Information to be transferred from source node to destination node:
  - Process's state information – Execution status, scheduling info, info about main mem, IO states,list of objects to which process has rights to access, process identifier, processes user and grp identifier, info abt file opened by the process
  - Process's address space – Code ,data and stack of the program
- Difference between the size of process's state information(few kilobytes)and address space (MB)
- Possible to transfer the address space without stopping its execution
- Not possible to resume execution until the state information is fully transferred
- Three methods for address space transfer
  - Total Freezing
  - Pretransferring
  - Transfer on reference

Process Migration Mechanisms

- Total Freezing
  - Execution is stopped while its address space is being transferred
  - Process is suspended for a long time during migration
  - Simple and easy to implement
  - Not suitable for interactive process



Process Migration Mechanisms

Pretransferring (precopying)

- Address space is transferred while the process is still running on the source node
- Initial transfer of the complete address space followed by repeated transfers of the pages modified during previous transfer
- Remaining modified pages are retransferred after the process is frozen for transferring its state information
- freezing time is reduced
- Pretransfer operation is executed at a higher priority than all other programs on the source node
- Total time of migration is increased due to the possibility of redundant page transfers

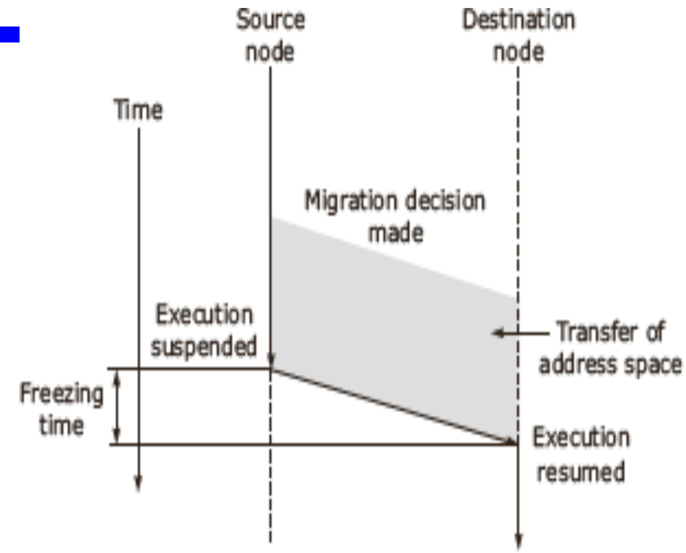
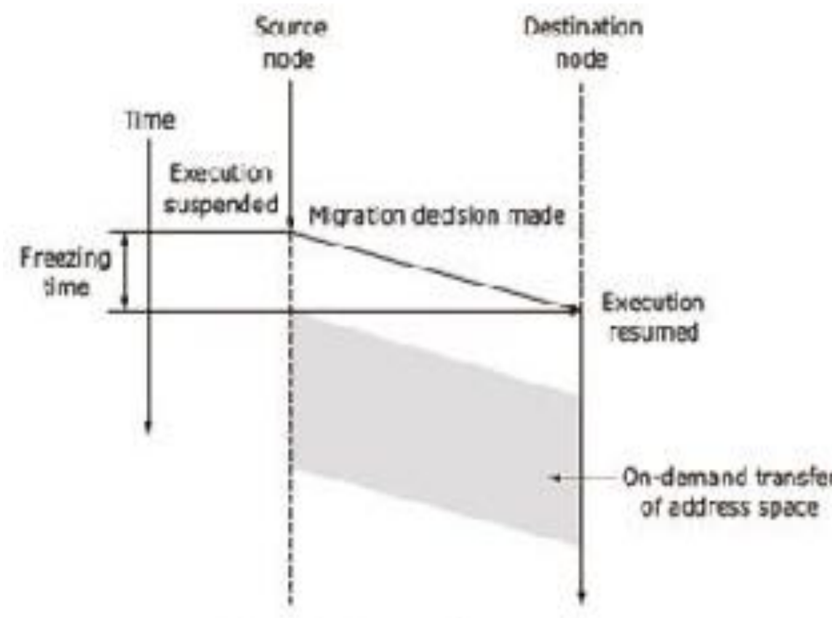


Figure 6-18 Pretransfer

Process Migration Mechanisms

- Transfer on reference
- Based on the assumption that the process tends to use only a relatively small part of their address space while executing.
- A page of the address space is transferred from its source node to destination node only when referenced
- Demand driven copy on reference approach



- Switching time is very short and independent of the size of the address space
- Not efficient in terms of cost
- Imposes continued load on process's source node
- Results in failure if source node fails or reboots

Process Migration Mechanisms

Message forwarding mechanisms

- Ensures that all pending, en-route and future messages arrive at the process's new location
- Classification of the messages to be forwarded
  - Type 1: Messages received at the source node after the process's execution has been stopped on its source node and process's execution has not yet been started on its destination node
  - Type 2: Message received at the source node after the process's execution has started on its destination node
  - Type 3: Messages that are to be sent to the migrant process from any other node after it has started executing on the destination node
- Mechanism of Resending the Message
  - Messages of type 1 and 2 are returned to the sender as not deliverable or are simply dropped
  - Locating a process is required upon the receipt of the nonnegative reply (messages of type 3)
  - Drawback: nontransparent to the processes interacting with the migrant process

Procss Migration Mechanisms

- Message forwarding mechanisms: Origin Site Mechanism
  - Process identifier has the process's origin site(orhome node) embedded in it
  - Each site is responsible for keeping information about the current location of all the processes created on it
  - Messages are sent to the origin site first and from there

  they are forwarded to the current location
  - Drawbacks:1. not good from reliability point of view

  2. continuous load on migrant process's original site
  - Link Traversal mechanism
  - Uses message queue for storing messages of type 1
  - Use of link (a forwarding address) for messages of type 2 and 3
  - Link has two components: process identifier and last known location of the process
  - Migrated process is located by traversing a series of links
  - Drawbacks: 1. poor efficiency 2. poor reliability
  - Link Update mechanism
  - Processes communicate via location independent links
  - During the transfer phase, the source node sends link update message to all relevant kernels

Process Migration Mechanisms

- Mechanisms for
- handling coprocesses
- Communication between a process and its subprocesses
- Two different mechanisms
  - Disallowing separation of Coprocesses
    - By disallowing the migration of processes that wait for one or more of their children to complete.
    - By ensuring that when a parent process migrates, its children processes will be migrated along with it
      - Concept of logical host
      - Process id is structured as {logical host-id, local-index} pair
    - Drawback :1. Does not allow parallelism within jobs 2. Overhead is large when logical host contains several processes
  - home node or origin site concept
    - Complete freedom of migrating a process or itssubprocesses independently and executing them on different nodes
    - Drawback:Message traffic and communication cost is significant

Advantages of process migration

- Reducing average response time of processes
- Speeding up individual jobs
  - Execute tasks of a job concurrently
  - To migrate a job to a node having faster CPU
- Gaining higher throughput
  - Using suitable load balancing policy
- Utilizing resources effectively
  - Depending on the nature of the process, it can be migrated to the most suitable node
- Reducing network traffic
  - Migrate the process closer to the resources it is using most heavily
  - To migrate and cluster two or more processes which frequently communicate with each other, on the same node
- improving system reliability
  - Migrating critical process to more reliable node
- Improving system security
  - A sensitive process may be migrated and run on the secure node

Resource Management in Distributed Systems

Introduction

- Distributed systems contain a set of resources interconnected by a network
- Processes are migrated to fulfill their resource requirements
- Resource manager are to control the assignment of resources to processes
- Resources can be logical (shared file) or physical (CPU)
- We consider a resource to be a processor

Types of process scheduling techniques

- Task assignment approach
  - User processes are collections of related tasks
  - Tasks are scheduled to improve performance
- Load-balancing approach
  - Tasks are distributed among nodes so as to equalize the workload of nodes of the system
- Load-sharing approach
  - Simply attempts to avoid idle nodes while processes wait for being processed

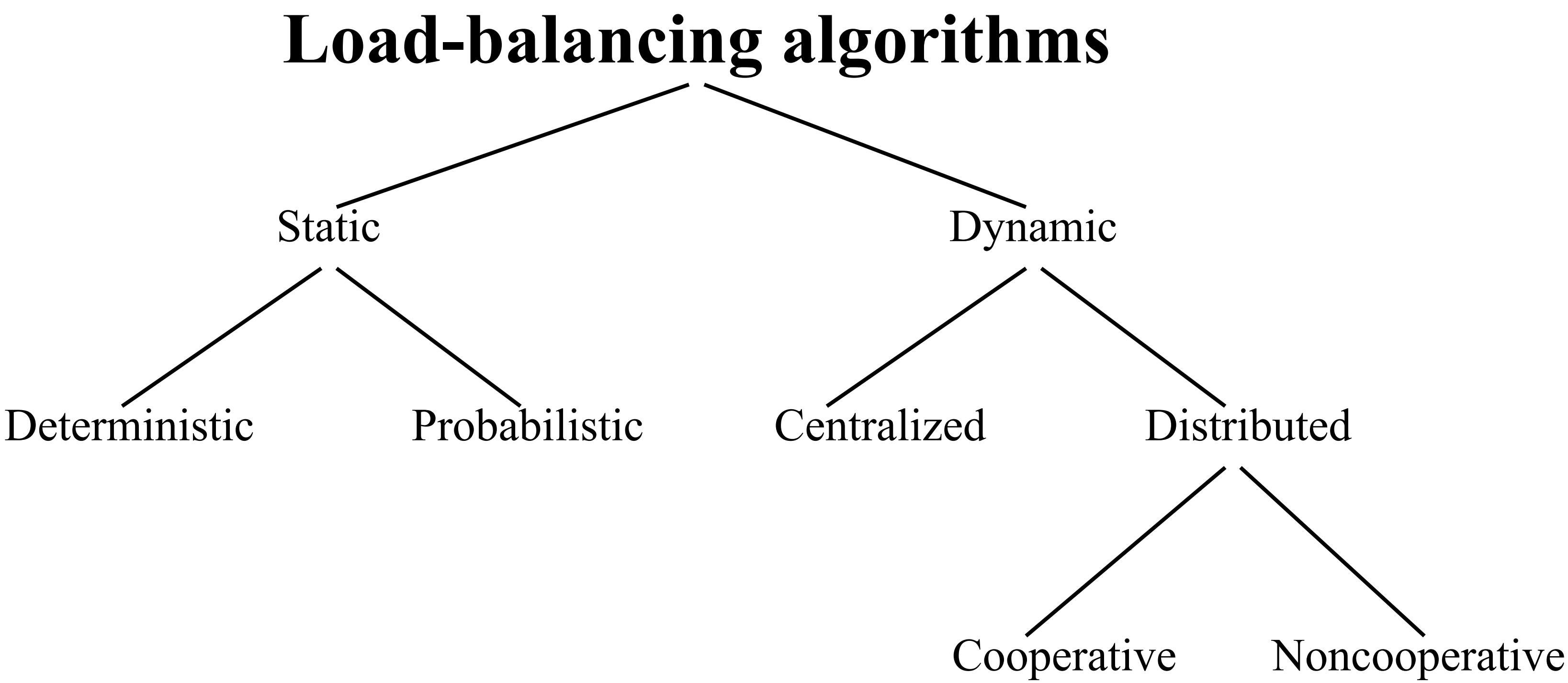
Desirable features of a scheduling algorithm

- No A Priori Knowledge about Processes
  - User does not want to specify information about characteristic and requirements
- Dynamic in nature
  - Decision should be based on the changing load of nodes and not on fixed static policy
- Quick decision-making capability
  - Algorithm must make quick decision about the assignment of task to nodes of system
- Balanced system performance and scheduling overhead
  - Great amount of information gives more intelligent decision, but increases overhead

Desirable features of a scheduling algorithm

- Stability
  - Unstable when all processes are migrating without accomplishing any useful work
  - It occurs when the nodes turn from lightly-loaded to heavily-loaded state and vice versa
- Scalability
  - A scheduling algorithm should be capable of handling small as well as large networks
- Fault tolerance
  - Should be capable of working after the crash of one or more nodes of the system
- Fairness of Service
  - More users initiating equivalent processes expect to receive the same quality of service

Load-balancing approach



A Taxonomy of Load-Balancing Algorithms

Load-balancing approach

Type of load-balancing algorithms

- Static versus Dynamic
  - Static algorithms use only information about the average behavior of the system
  - Static algorithms ignorethe current state or load of the nodes in the system
  - Dynamic algorithms collect state information and react to system state if it changed
  - Static algorithms are much more simpler
  - Dynamic algorithms are able to give significantly better performance

Load-balancing approach

Type of static load-balancing algorithms

- Deterministic versus Probabilistic
  - Deterministic algorithms use the information about the properties of the nodes and the characteristic of processes to be scheduled
  - Probabilistic algorithms use information of static attributes of the system (e.g. number of nodes, processing capability, topology) to formulate simple process placement rules
  - Deterministic approach is difficult to optimize
  - Probabilistic approach has poor performance

Load-balancing approach

Type of dynamic load-balancing algorithms

- Centralized versus Distributed
  - Centralized approach collects information to server node and makes assignment decision
  - Distributed approach contains entities to make decisions on a predefined set of nodes
  - Centralized algorithms can make efficient decisions, have lower fault-tolerance
  - Distributed algorithms avoid the bottleneck of collecting state information and react faster

Load-balancing approach

Type of distributed load-balancing algorithms

- Cooperative versus Noncooperative
  - In Noncooperative algorithms entities act as autonomous ones and make scheduling decisions independently from other entities
  - In Cooperative algorithms distributed entities cooperatewith each other
  - Cooperative algorithms are more complex and involve larger overhead
  - Stability of Cooperative algorithms are better

Issues in designing Load-balancing algorithms

- Load estimation policy
  - determines how to estimate the workload of a node
- Process transfer policy
  - determines whether to execute a process locally or remote
- State information exchange policy
  - determines how to exchange load information among nodes
- Location policy
  - determines to which node the transferable process should be sent
- Priority assignment policy
  - determines the priority of execution of local and remote processes
- Migration limiting policy
  - determines the total number of times a process can migrate

Load estimation policy I.

for Load-balancing algorithms

- To balance the workload on all the nodes of the system, it is necessary to decide how to measure the workload of a particular node
- Some measurable parameters (with and
  time dependent factor) can be the following: node
  - Total number of processes on the node
  - Resource demands of these processes
  - Instruction mixes of these processes
  - Architecture and speed of the node's processor
  - Several load-balancing algorithms use the <u>total number of processes</u> to achieve big efficiency

Load estimation policy II.

for Load-balancing algorithms

- In some cases the true load could vary widely depending on the remaining service time, which can be measured in several way:
  - *Memoryless method* assumes that all processes have the same expected remaining service time, independent of the time used so far
  - *Past repeats* assumes that the remaining service time is equal to the time used so far
  - *Distribution method* states that if the distribution service times is known, the associated process's remaining service time is the expected remaining time conditioned by the time already used

Load estimation policy III.

for Load-balancing algorithms

- None of the previous methods can be used in modern systems because of periodically running processes and daemons
- An acceptable method for use as the load estimation policy in these systems would be to measure the CPU utilization of the nodes
- Central Processing Unit utilization is defined as the number of CPU cycles actually executed per unit of real time
- It can be measured by setting up a timer to periodically check the CPU state (idle/busy)

Process transfer policy I.

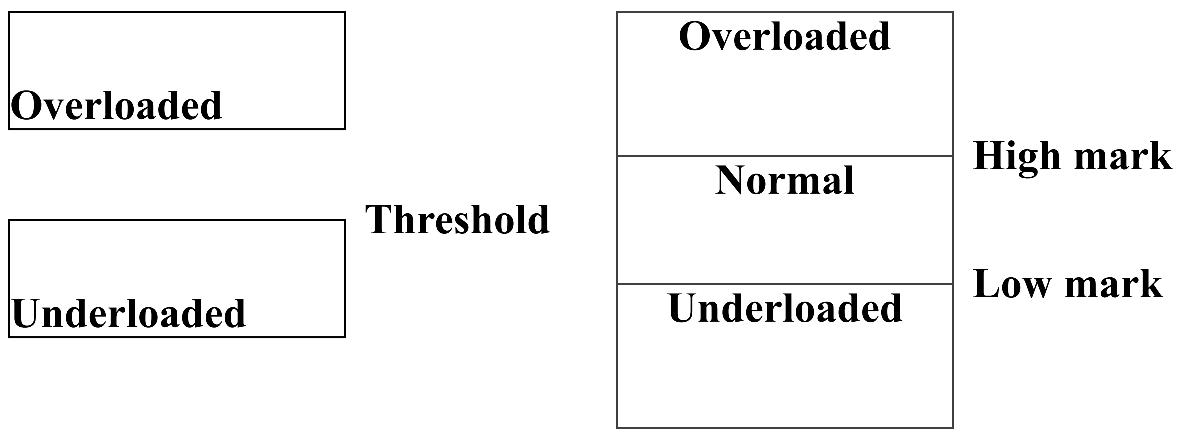
for Load-balancing algorithms

- Most of the algorithms use the *threshold policy* to decide on whether the node is lightly-loaded or heavily-loaded
- Threshold value is a limiting value of the workload of node which can be determined by
  - Static policy: predefined threshold value for each node depending on processing capability
  - Dynamic policy: threshold value is calculated from average workload and a predefined constant
- Below threshold value node accepts processes to execute, above threshold value node tries to transfer processes to a lightly-loaded node

Process transfer policy II.

for Load-balancing algorithms

- Single-threshold policy may lead to unstable algorithm because underloaded node could turn to be overloaded right after a process migration
- To reduce instability double-threshold policy has been proposed which is also known as high-low policy



Single-threshold policy

Double-threshold policy

Process transfer policy III.

for Load-balancing algorithms

- Double threshold policy
  - When node is in overloaded region new local processes are sent to run remotely, requests to accept remote processes are rejected
  - When node is in normal region new local processes run locally, requests to accept remote processes are rejected
  - When node is in underloaded region new local processes run locally, requests to accept remote processes are accepted

Location policy I.

for Load-balancing algorithms

❑Threshold method

- ❑ Policy selects a random node, checks whether the node is able to receive the process, then transfers the process. If node rejects, another node is selected randomly. This continues until probe limit is reached.

❑Shortest method

- ❑ L distinct nodes are chosen at random, each is polled to determine its load. The process is transferred to the node having the minimum value unless its workload value prohibits to accept the process.
- ❑ Simple improvement is to discontinue probing whenever a node with zero load is encountered.

Location policy II.

for Load-balancing algorithms

❑Bidding method

- ❑ Nodes contain managers (to send processes) and contractors (to receive processes)
- ❑ Managers broadcast a request for bid, contractors respond with bids (prices based on capacity of the contractor node) and manager selects the best offer
- ❑ Winning contractor is notified and asked whether it accepts the process for execution or not
- ❑ Full autonomy for the nodes regarding scheduling
- ❑ Big communication overhead
- ❑ Difficult to decide a good pricing policy

Location policy III.

for Load-balancing algorithms

- Pairing
  - Contrary to the former methods the pairing policy is to reduce the variance of load only between pairs
  - Each node asks some randomly chosen node to form a pair with it
  - If it receives a rejection it randomly selects another node and tries to pair again
  - Two nodes that differ greatly in load are temporarily paired with each other and migration starts
  - The pair is broken as soon as the migration is over
  - A node only tries to find a partner if it has at least two processes

State information exchange policy I

for Load-balancing algorithms

- Dynamic policies require frequent exchange of state information, but these extra messages arise two opposite impacts:
  - Increasing the number of messages gives more accurate scheduling decision
  - Increasing the number of messages raises the queuing time of messages
- State information policies can be the following:
  - Periodic broadcast
  - Broadcast when state changes
  - On-demand exchange
  - Exchange by polling

State information exchange policy II.

for Load-balancing algorithms

- Periodic broadcast
  - Each node broadcasts its state information after the elapse of every *T* units of time
  - Problem: heavy traffic, fruitless messages, poor scalability since information exchange is too large for networks having many nodes
- Broadcast when state changes
  - Avoids fruitless messages by broadcasting the state only when a process arrives or departures
  - Further improvement is to broadcast only when state switches to another region (double-threshold policy)

State information exchange policy III.

for Load-balancing algorithms

❑On-demand exchange

- ❑ In this method a node broadcast a State-Information-Request message when its state switches from normal to either underloaded or overloaded region.
- ❑ On receiving this message other nodes reply with their own state information to the requesting node
- ❑ Further improvement can be that only those nodes reply which are useful to the requesting node

❑Exchange by polling

- ❑ To avoid poor scalability (coming from broadcast messages) the partner node is searched by polling the other nodes on by one, until poll limit is reached

Priority assignment policy

for Load-balancing algorithms

- Selfish
  - Local processes are given higher priority than remote processes. Worst response time performance of the three policies.
- Altruistic
  - Remote processes are given higher priority than local processes. Best response time performance of the three policies.
- Intermediate
  - When the number of local processes is greater or equal to the number of remote processes, local processes are given higher priority than remote processes. Otherwise, remote processes are given higher priority than local processes.

Migration limiting policy

for Load-balancing algorithms

- This policy determines the total number of times a process can migrate
  - Uncontrolled
    - A remote process arriving at a node is treated just as a process originating at a node, so a process may be migrated any number of times
  - Controlled
    - Avoids the instability of the uncontrolled policy
    - Use a *migration count* parameter to fix a limit on the number of time a process can migrate
    - Irrevocable migration policy: *migration count* is fixed to 1
    - For long execution processes *migration count* must be greater than 1 to adapt for dynamically changing states

Load-sharing approach

- Drawbacks of Load-balancing approach
  - Load balancing technique with attempting equalizing the workload on all the nodes is not an appropriate object since big overhead is generated by gathering exact state information
  - Load balancing is not achievable since number of processes in a node is always fluctuating and temporal unbalance among the nodes exists every moment
- Basic ideas for Load-sharing approach
  - It is necessary and sufficient to prevent nodes from being idle while some other nodes have more than two processes
  - Load-sharing is much simpler than load-balancing since it only attempts to ensure that no node is idle when heavily node exists
  - Priority assignment policy and migration limiting policy are the same as that

for the load-balancing algorithms

Load estimation policies

for Load-sharing algorithms

- Since load-sharing algorithms simply attempt to avoid idle nodes, it is sufficient to know whether a node is busy or idle
- Thus these algorithms normally employ the simplest load estimation policy of counting the total number of processes
- In modern systems where permanent existence of several processes on an idle node is possible, algorithms measure CPU utilization to estimate the load of a node

Process transfer policies

for Load-sharing algorithms

- Algorithms normally use all-or-nothing strategy
- This strategy uses the threshold value of all the nodes fixed to 1
- Nodes become receiver node when it has no process, and become sender node when it has more than 1 process
- To avoid processing power on nodes having zero process load-sharing algorithms use a threshold value of 2 instead of 1
- When CPU utilization is used as the load estimation policy, the double-threshold policy should be used as the process transfer policy

Location policies I.

for Load-sharing algorithms

- Location policy decides whether the sender node or the receiver node of the process takes the initiative to search for suitable node in the system, and this policy can be the following:
  - Sender-initiated location policy
    - Sender node decides where to send the process
    - Heavily loaded nodes search for lightly loaded nodes
  - Receiver-initiated location policy
    - Receiver node decides from where to get the process
    - Lightly loaded nodes search for heavily loaded nodes

Location policies II.

for Load-sharing algorithms

- Sender-initiated location policy
  - Node becomes overloaded, it either broadcasts or randomly probes the other nodes one by one to find a node that is able to receive remote processes
  - When broadcasting, suitable node is known as soon as reply arrives
- Receiver-initiated location policy
  - Nodes becomes underloaded, it either broadcast or randomly probes the other nodes one by one to indicate its willingness to receive remote processes
- Receiver-initiated policy require preemptive process migration facility since scheduling decisions are usually made at process departure epochs

Location policies III.

for Load-sharing algorithms

- Experiences with location policies
  - Both policies gives substantial performance advantages over the situation in which no load-sharing is attempted
  - Sender-initiated policy is preferable at light to moderate system loads
  - Receiver-initiated policy is preferable at high system loads
  - Sender-initiated policy provide better performance for the case when process transfer cost significantly more at receiver-initiated than at sender-initiated policy due to the preemptive transfer of processes

State information exchange policies

for Load-sharing algorithms

- In load-sharing algorithms it is not necessary for the nodes to periodically exchange state information, but needs to know the state of other nodes when it is either underloaded or overloaded
- Broadcast when state changes
  - In sender-initiated/receiver-initiated location policy a node broadcasts State Information Request when it becomes overloaded/underloaded
  - It is called broadcast-when-idle policy when receiver-initiated policy is used with fixed threshold value value of 1
- Poll when state changes
  - In large networks polling mechanism is used
  - Polling mechanism randomly asks different nodes for state information until find an appropriate one or probe limit is reached
  - It is called poll-when-idle policy when receiver-initiated policy is used with fixed threshold value value of 1