DC

1] Load Balancing Approach:

→ - It is used in Process management.
- Scheduling Algo^n that uses Load Balancing Approach are called as Load Balancing Algorithms.
- The main goal of load balancing algorithms is to balance the workload on all the nodes of the system.
- Load balancing aims to:
a] Optimize resource use
b] Maximize throughput
c] Minimize response time
d] Avoid overload of any single resource.
- ~~In this, the processes algo are distributed among~~

Designing Issues.

- Designing a load balancing algorithm is a critical task.

i] Load Estimation Policy:

- The first issue in a load balancing algorithm is to decide which method to use to estimate the workload of a particular node.
- Estimation of the workload of a particular node is a difficult problem for which no completely satisfactory solution exists.
- A nodes workload can be estimated based on some measurable parameters below:
a] Total number of processes on the node
b] Resource demands of these processes.
c] Instruction mixes of these processes.
d] Architecture & speed of the node's processor.

ii] Process Transfer Policy:

- The idea of using this policy is to transfer processes from heavily loaded nodes to lightly loaded nodes.

- Most of the algorithms use the threshold policy to decide 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:

a] Static Policy: Each node has a predefined threshold value.

b] Dynamic Policy: The threshold value for a node is dynamically added.

iii] Location Policy:

- When a transfer policy decides that a process is to be transferred from one node to any other lightly loaded node, the challenge is to find the destination node.
- Location policy determines this destination node.
- It includes following

a] Threshold: This policy selects a random node & checks whether the node is able to receive the process then it transfers the process.

b] Shortest method: In this 'n' distinct nodes are chosen at random & each is polled to determine its load with min. value.

c] Bidding method: In this method, nodes contain managers to send processes & contractors to receive processes.

d] Pairing: It's used to reduce the variance of load only betw^n nodes of the system.

iv] State Information Exchange Policy:

i] It's used for freq^ exchange of state information within the nodes.

ii] It includes:

a] Periodic Broadcast: Each node broadcasts its state infor^ after the elapse of T units of time.

b] Broadcast when state changes: Each node broadcasts its state information only when a process arrives or departures

c] On demand exchanges: Each node broadcasts its state infa^ req^ message when its state switches from normal to either under load or over load.

d] Exchanges by polling: The partner node is searched by polling the other nodes on by one until poll limit is reached.

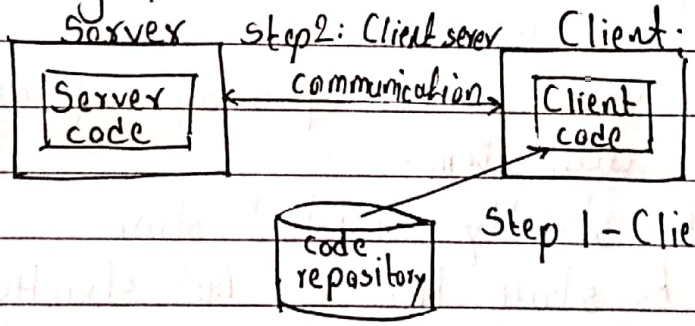
v] Priority Assignment Policy:
1] The priority of the execution of local & remote processes at a particular node should be planned
2] It includes:
a] Selfish priority assignment rule: In this local proces are given higher priority than remote process.
b] Altruistic priority assignment rule: Remote processes are given higher priority than local process.
c] Intermediate priority assignment rule: Priority in this rule is decided depending upon the number of local & remote processes on a node.

vi] Migration Limiting Policy:
1] This policy determines the total number of times a process can migrate from one node to another.
2] It includes:
a] Uncontrolled Policy: The remote process is treated as local process.
b] Controlled Policy: Migration count is used for remote processes.

3] When a process accesses a synchronization variable, the entire memory is synchronized by making visible the changes made to the memory to all other processes.

2] Code Migration:

- i] In process migration, an entire process has to moved from one machine to another.
- But this may be a crucial task, but it has good overall performance.
- In some cases, a code needs to be migrated rather than the process.
- Code Migr^n refers to transfer of program from one node to ano
- Eg- Consider a client server system, in which the server has to manage a big database.
- The client application has to perform many database operations.
- In such situation, it's better to move part of client application to the server & the result is send across net
- Thus code migration is better option.
- Code Migra^n is used to improve overall perf^n of the syster by exploiting parallelism.



- Here the server is responsible to provide the clients implementation, when the client binds to the server.
- The advantage of this approach is that, the client does not need to install all the required software.
- The software can be moved in as when necessary & discarded when it is not needed.

3] Client Centric Consistency Model:

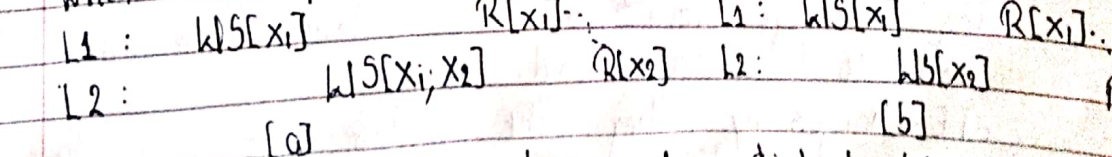
- It's one of consistency model in distributed system.
- Client centric consistency model is more prefered as system wide consistency is hard.
- C.C.C model can be used to hide inconsistencies in cheap way.
- It aims at providing a consistent view on a database

1] Eventual Consistency:

- An eventual consistency is a weak consistency model.
- It lacks in ~~simulation~~ simultaneous updates
- It defines that if updates do not occur for a long period of time all replicas will gradually become consistent.
- Eventual Consistency is a lot inexpensive to implement..
- Requirements for eventual consistency are:
  a] Few read/write conflicts
  b] No write/write conflicts
  c] Clients can accept temporary inconsistency.
- Eg - klklkl.

2] Monotonic Reads:

- A data stored is said to offer monotonic reads consi^n if following cond^n "If process P reads the value of data item x, any successive read operation on x by that process will always return the same value or more recent one."
- Consider a distributed email database.
- It has distributed & replicated user mailboxes.
- Emails can be inserted at any location
- However, updates are propageted in a lazy [i.e: on demand] fashion.
- Suppose the end user reads his mails in Mumbai. Assume that only reading mail does not affect the mailbox.
- Later when the end user moves to Pune & opens his mail box again, monotonic read consistency assurance that the messages that were in the mailbox in Mumbai will also be in mailbox when it is opened in Pune.



fig[a] demonstrates a monotonic-read consistent data store.
fig[b] —//— a data with not so monotonic reads.

Page No.:
Date: / /

4] Data Centric consistency model:
- Data centric consistency models aim to provide system wide consistent view of the data store.
- A data store may be physically distributed accross multiple machines.
- Each process that can access data from the store is assumed to have a local or nearby copy available of the entire store.

1] Strict Consistency Model:
- Any read on a data item X returns a value corresponding to the result of the most recent write on X.
- This is the strongest form of memory coherence which has the most stringent consistency requirement.

P1: W[x]a
P2: R[x]a
[a]

P1: W[x]a
P2: R[x]NIL R[x]a
[b]

- Behavior of two processes, operating on the same data item.
a] A strictly consistent store.
b] A store that is not strictly consistent.

2] Weak Consistency Model:
- The basic idea behind the weak consistency model is enforcing consistency on a group of memory reference operations rather than individual operations.
- A Distributed shared memory system that supports the weak consistency model uses a special variable called a synchronization variable which is used to synchronize memory.