<u>DISTRIBUTED COMPUTING ASSIGNMENT</u>

Name : Aniket Bote
Roll Number : 10
Class : D17C

1.)Give advantages and disadvantages of task assignment , Load Balancing and Load Sharing.

Ans: <u>Task Assignment</u> : In this each process submitted by the user for processing is viewed as a collection of related tasks and they are scheduled to suitable nodes in order to improve performance.There are two task assignment parameters , Cost of executing a task on a particular node and inter-task communication cost.

Advantages :
1. It minimizes Inter-task communication cost
2. High degree of parallelism

Disadvantages :
1. Not Dynamic in nature
2. Requires prerequisite knowledge about the process

<u>Load Balancing Assignment</u> : The aim of load balancing assignment is to keep neither a node idle or overloaded. It maintains a load equal load at each node at any moment of time during execution to obtain the maximum performance of the system. Load balancing can be either static or dynamic.

Advantages :
1. maximum utilization of resources.
2. Response time becomes shorter.
3. It gives higher throughput.
4. It gives higher reliability.
5. It has low cost but high gain.

Disadvantages :
1. Deciding policies are required for task assignment
2. No native failure detection or fault tolerance and no dynamic load re-balancing.

<u>Load Sharing Approach</u> : Aims to equalize the workload on all the nodes of a system by gathering state information . This can be implemented in two ways , sender initiated approach and Receiver initiated approach. In the former , the sender takes the decision on the transfer of process and in the latter the lightly loaded nodes search for processes to be able to share the load of the system.

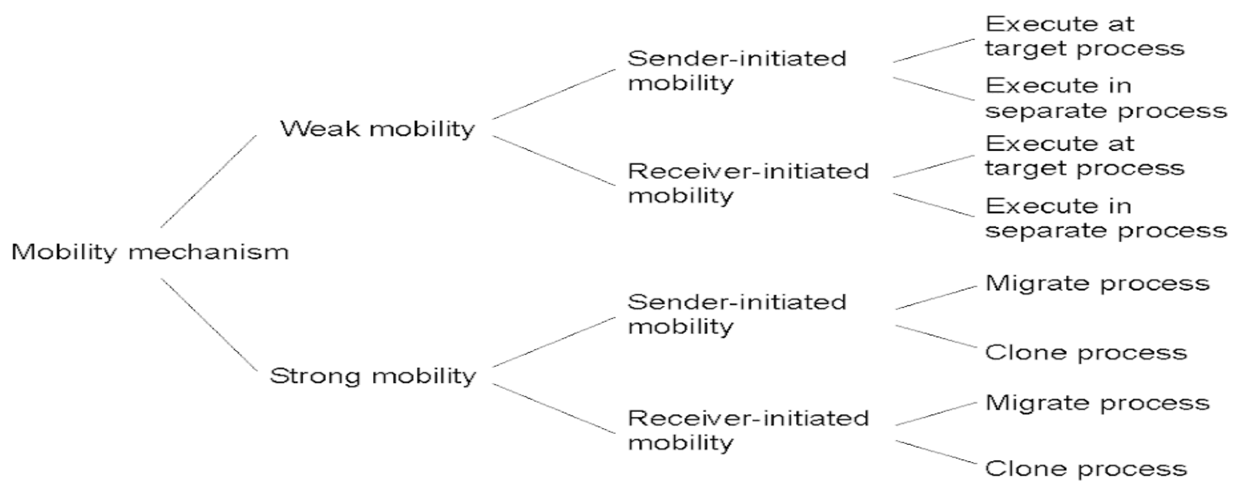
Advantages :

2.) Compare processes and threads

Ans:

Process	**Thread**
An executing instance of a program is called a process.	A thread is a subset of the process.
A process is always stored in the main memory also termed as the primary memory or random access memory.	Since the threads of a process share the same memory, synchronizing the access to the shared data within the process gains unprecedented importance.
Several processes may be associated with the same program.	A process may also be made up of multiple threads of execution that execute instructions concurrently.
On a uni-processor system, though true parallelism is not achieved, a process scheduling algorithm is applied and the processor is scheduled to execute each process one at a time yielding an illusion of concurrency.	On a uni-processor system, a thread scheduling algorithm is applied and the processor is scheduled to run each thread one at a time.

3.) Draw models of code migration
Ans:


Mobility mechanism
Weak mobility
Sender-initiated mobility
Execute at target process
Execute in separate process
Receiver-initiated mobility
Execute at target process
Execute in separate process
Strong mobility
Sender-initiated mobility
Migrate process
Clone process
Receiver-initiated mobility
Migrate process
Clone process

Code migration not only involves moving code between machines, the term actually covers more.Code migration deals with moving programs between machines, with the intention to have those programs be executed at the target.

Weak Mobility: Code is moved in it’s original state , it is simple but it’s applicability is meagre.

Strong Mobility : code and the state is moved. The execution restarts from the next statement.It is harder to implement but is very powerful.

Sender Initiated : Code migration is initiated by the sender , where the code is currently residing.

Receiver Initiated : Code migration is taken by the receiver machine or the target machine.

Cloned Process : It is executed parallely with the original process

4.) Draw Matrix of Process to Resource and Resource to machine binding.

Ans : Resources are of the following types:

1. Unattached: a resource that can be moved easily from machine to machine.
2. Fastened: migration is possible, but at a high cost.
3. Fixed: a resource is bound to a specific machine or environment, and cannot be migrated.

Process to Resource Binding :

1. Binding-by-identifier (BI) : the referenced resource, and nothing else, has to be migrated.
2. Binding-by-value (BV) :only the value of the resource need be migrated.
3. Binding-by-type (BT): nothing is migrated, but a resource of a specific type needs to be available after migration (eg, a printer).

	Unattached	Fastened	Fixed
By Identifier	MV(or GR)	GR(or MV)	GR
By Value	CP (or MV,GR)	GR(or CP)	GR
By type	RB (or MV,CP)	RB(or GR,CP)	RB(or GR)

GR : Establish a global system wide reference
MV : Move the resource
CP : Copy the value of Resource
RB : Rebind process to locally-available resource

5.) Explain desirable features of scheduling algorithm

Ans:

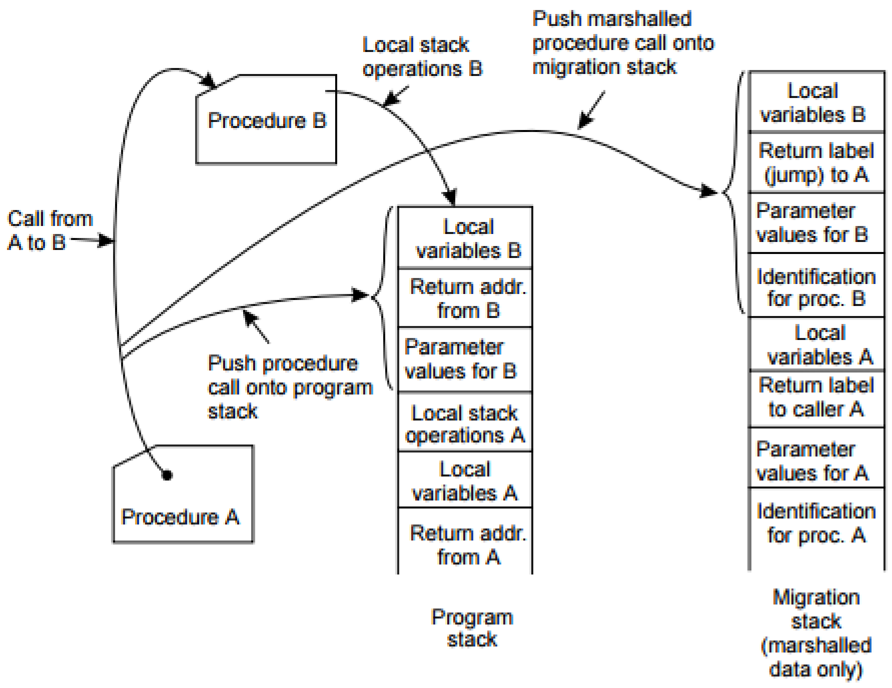
1) No apriori knowledge about the processes:Scheduling algorithm is based on the information about the characteristics and resource requirements of the processes. These pose extra burden on the users who must provide this information while submitting their processes for execution.

2) Dynamic in Nature:The decision regarding the assignment of process should be dynamic, which means that it should be based on the current load of the system.Algorithm must have the flexibility to migrate the process more than once should it be the need. The process should be placed on a particular node which can be changed afterwards, adapting to the change in system load.
3) Decision - making capabilities: Scheduling algorithms require less computational efforts that result in less time requirement for the output. This will provide a near optimal result and has decision - making capability.
4) Balancing System performance and Scheduling overhead:It is better to collect minimum information about the state information, such as CPU load. This is important because as the amount of global state information collected increases, the overhead also increases. This affects the result of the cost of gathering and processing the extra information so there is a need to improve the system performance by minimizing scheduling overhead.
5) Stability:Unnescessary transfer must be prevented.Ex: if nodes n_1 and n_2 are loaded with processes and it is observed that node n_3 is idle, then we can offload a portion of their work to n_3 without being aware of the offloading decision made by some of the other nodes. In case n_3 becomes overloaded due to this, it may again start transferring its processes to other nodes. The main reason for this is that scheduling decisions are being made at each node independently of decisions made by other nodes.
6) Scalability:The scheduling algorithm should accommodate if the number of nodes increase. An algorithm can be termed as having poor scalability if it makes a decisions by first inquiring the workload from all the nodes and then selecting the most lightly loaded node. This concept will work fine only when we have few nodes in the system. It won’t work when the number of nodes are more as a flood of replies will increase the time required to process the reply messages for making a node selection. As the number of nodes (N) increases, the network traffic consumes network bandwidth quickly.
7) Fault Tolerance:In case one or more nodes of the system crash, good scheduling algorithm should not be disabled by this and coping mechanism to handle this should be available.In order to have better fault tolerance capability, algorithms should decentralize the decision - making capability and must consider only available nodes in their decision - making and have better fault tolerance capability.
8) Fairness of Service: Scheduling policy blindly attempts to balance the load on all the nodes of the system, then they are not fair . This is because in any load - balancing scheme, heavy nodes will obtain all benefits while lightly loaded nodes will suffer poor response time than in a stand alone configuration.

6.) Explain process migration in heterogeneous environment

Ans:One of the features of Distributed systems is that it is heterogeneous , i.e each node has a different OS and runs on a different platform and uses different data structures . Migration in such systems requires that the code segment can be executed on each platform despite of their differences . Issues with migration are eased during weak mobility where there is basically no runtime information that needs to be transferred between machines, the code compiles to produce different code segment for each target machine. Transfer of the execution segment poses as a problem during strong mobility. Execution Segment is highly dependent on the platform on which the process is being executed. In fact, only when the target machine is identical , is it possible to migrate the execution segment.

Code migration is restricted to specific points in the execution of a program. In particular, migration can take place only when a next subroutine is called. The runtime system maintains its own copy of the program stack, but in a machine-independent way. This copy is called the migration stack. The migration stack is updated when a subroutine is called, or when execution returns from a subroutine.When a subroutine is called, the runtime system marshals the data that have been pushed onto the stack since the last call. These data represent values of local variables, along with parameter values for the newly called subroutine. The marshaled data is then pushed onto the migration stack, along with an identifier of the called subroutine. In addition, the address where execution should continue when the caller returns from the subroutine is pushed in the form of a jump label onto the migration stack as well.



If code migration takes place where a subroutine is called, the runtime system marshals all global program-specific data to form part of the execution segment. Data specific to the machine is ignored along with the current stack. This marshaled data is transferred to the destination, along with the migration stack. Besides, the destination loads the appropriate code segment containing the binaries fit for its machine architecture and operating system. The marshaled data belonging to the execution segment are unmarshaled, and a new runtime stack is constructed by unmarshaling the migration stack. Execution can then be resumed by simply entering the subroutine that was called at the original site.

7) List types of virtualization

Ans: Virtualization can take place in two ways :

A. Process Virtual Machine : Builds a runtime system which provides an abstract instruction set which is used for executing applications. The instruction can be interpreted, emulated as it is done for running windows on UNIX platforms. This virtualization leads to call a process virtual machine stressing that virtualization is done essentially only for a single process.
B. Native Virtual machine monitor : Provides a system that is essentially implemented as a layer

completely shielding the original hardware , but now offering the complete instruction set of the same as in interface .It is possible to have multiple and different OS run independently and concurrently on the same platform.

8) Explain data centric consistency model

Ans:Data Centric Consistency model specifies a contract between (distributed) data store and processes, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency.It is the strongest data centric consistency model as it requires that a write on a data be immediately available to all replicas, ensuring consistency.In a distributed system, it is very difficult to maintain a global time ordering, along with the availability of data which is processed concurrently at different locations, which causes delays for transferring of data from one location to another.Thus Strict consistency model is impossible to achieve.
Eg: P1 & P2 are two processes. P1 performs a write operation on its data item x and modifies its value to a.This update is propagated to all other replicas of data item x.Suppose if P2 now reads and finds its value to be NIL due to the propagation delay. So the result of this value read by P2 is not strictly consistent. But as the law of Strictly consistent model says that results should be immediately propagated to all the replicas as shown in figure below:

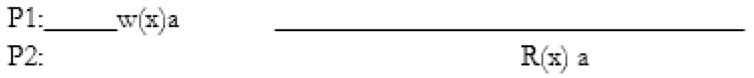
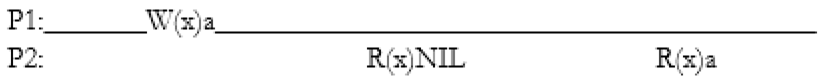
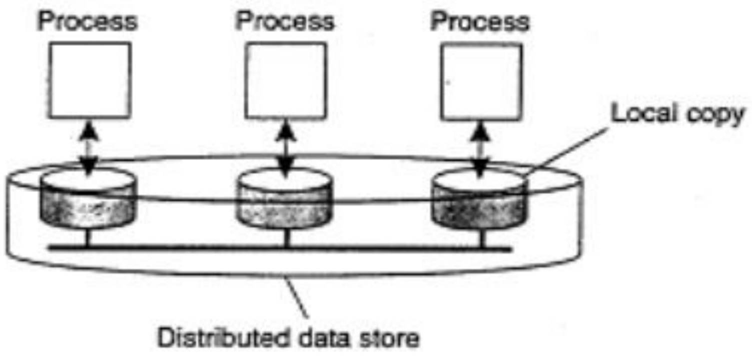


Fig (a) Strict consistent model





9)Explain Client centric consistency model

Ans:This consistency model does not handle simultaneous updates. The emphasis is more on maintaining a consistent view of things for the individual client process that is currently operating on the data-store.It states that “If a process P reads the value of a data item t, any successive read operation on x by that process at later time will always return that same or a recent value.

For eg Each time one connects to the email server (may be different replicas),the email server guarantees that all the time will always return that same or recent value.

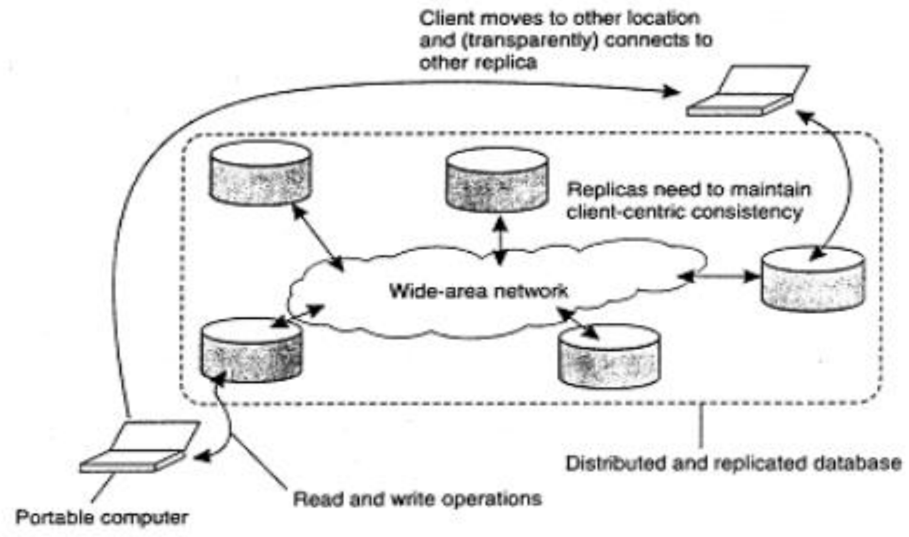
```
L1:____WS(x1) _____________________R(x1)________
 L2:     WS(x1,x2)                         R(x1)
```

Fig (a) Monotonic read consistent

Here, in fig (a) operations WS(x1) at L1 before the second read operation. Thus WS(x1,x2) at L2 would see the earlier update.

```
L1:__WS(x1) ___________________R(x1)__________
L2:   WS(x2)                         R(x2)
```

Fig (b) A data store which is monotonic read inconsistent


Client moves to other location and (transparently) connects to other replica
Replicas need to maintain client-centric consistency
Wide-area network
Distributed and replicated database
Read and write operations
Portable computer

10) Explain DFS,NFS and AFS with respect to file models , file acessing models , file caching schemes etc

Ans:
<u>Distributed File System :</u>
DFS provides a common view of the File system with the implementation being a distributed process.It enables the user to open any file on any machine present in the network.
Model:
In the unstructured model, a file is an unstructured sequence of bytes. The interpretation of the meaning and structure of the data stored in the files is up to the application.In structured , a file appears to the file server as an ordered sequence of records. Records of different files of the same file system can be of different sizes.

Most existing operating systems use the mutable file model. An update performed on a file overwrites its old contents to produce the new contents.In the immutable model, rather than updating the same file, a new version of the file is created each time a change is made to the original file and the old version is retained unchanged. The problems in this model are increased use of disk space and increased disk activity.
File access:
Remote file processing of a client's request is performed at the server’s node. Thus, the client's request for file access is delivered across the network as a message to the server, the server machine performs the access request, and the result is sent to the client. Need to minimize the number of messages sent and the overhead per message.
File Caching :
This model attempts to reduce the network traffic of the previous model by caching the data obtained from the server node. This takes advantage of the locality feature found in file accesses. A replacement policy such as LRU is used to keep the cache size bounded.While this model reduces network traffic it has to deal with the cache coherence problem during writes, because the local cached copy of the data needs to be updated, the original file at the server node needs to be updated and copies in any other caches need to be updated.

Network File System :
It is a model to integrate different file systems.Based on the idea that each file server provides a standardized view of its local file system.NFS runs on heterogeneous groups of computers.
File Model :
Files are hierarchically organized into a naming graph in which nodes represent directories and files.A directory node contains the mappings between file names and file handles .
File access:
NFS uses a remote access model: clients are unaware of file locations.
File Caching :
A client opens file,it caches the data it obtains from the server as the result of various read operations.Write operations are carried out in the cache.Modified Data in the cache are flushed back to
the server when the file is closed.When a client opens a previously closed file that has been (partly) cached, the client must contact the server to determine if the file has been changed.The cache is purged if the file has been changed.

Andrew File System :
AFS facilitates stored server file access between AFS client machines located in different areas. AFS supports reliable servers for all network clients accessing transparent and homogeneous namespace file locations.
File Access :
An AFS may be accessed from a distributed environment or location independent platform. A user accesses an AFS from a computer running any type of OS with Kerberos authentication and single namespace features. Users share files and applications after logging into machines that interact within the Distributed Computing Infrastructure (DCI).AFS also equips users with multiple access control permissions

File Caching:
In distributed networks, an AFS relies on local cache to enhance performance and reduce workload. For example, a server responds to a workstation request and stores the data in the workstation's local cache. When the workstation requests the same data, the local cache fulfills the request.