J. Varun Iyer
Distributed Systems
7th Sem CSE
19115037

Ans 4. Advantages of Process migration are:

(i) Better response time & execution speed-up due to dynamic load balancing among multiple nodes.

(ii) Higher throughput and effective resource utilization by migrating I/O & CPU-bound processes to file & cycle servers.

(iii) Reducing network traffic by migrating processes closer to the resources they are using most heavily.

(iv) Improving system reliability by migrating processes from failed sites to more reliable sites as well as replicating & migrating critical processes to remote sites.

Disadvantages / Issues in progress migration are:

(i) In uncontrolled process migration, a process may keep on hopping between nodes without ever being executed.

(ii) Since process must be freezed at source node & moved to destination node, this transportation may be time taking.

(iii) Transfering the process's address space & environment variables to destination to ensure smooth execution give rise to problems due to separation of address space between 2 communicating processes because addresses may change at the new destination node.

(iv) Residual dependencies between the source and destination nodes will continue to impose load on the previous node & thus diminish the benefits of migrating the process.

Ans 3. Differences between pre-emptive & non pre-emptive process migration are:

- A pre-emptive process migration facility allows the transfer of an executing process from one node to another. On the other hand, in systems supporting only non-pre-emptive migration facility, a process can only be transferred prior to beginning its execution.

- Preemptive process migration is costlier than non-pre-emptive process migration since the process state, which must accompany the process to its new node, becomes much more complex after execution begins.

These differences in their nature leads to relative advantages & disadvantages.

- Preemptive migration is capable of immense flexibility as processes can be interrupted, freezed & transferred as per our needs. But this flexibility comes with increased cost of transfering all the elements associated with the process's state to the new node.
- Non preemptive migration is strong & pretty straightforward to implement with minimal overhead as compared to pre-emptive process migration. But once the process starts executing, it will finish its useful execution at the same node.

Factors which influence the designing of pre-emptive / non - pre-emptive process migration techniques are :

- Resource capabilities of different nodes: If all the nodes offer similar performance then non-pre-emptive process migration facility can be designed. However if the nodes in the distributed system are varied in performance levels, then pre-emptive process migration can enable much better performance enhancement.

• Cost of overhead: If the costs & performance involved in pre-empting a process & then migrating it to a different node balances out well & the costs are less, pre-emptive process migration can be designed. However, since if transportation of process state includes substantial overhead, then non preemptive process migration is opted. (4)

Ans 2. Policies used for load estimation are:

- Total number of processes on the node at a time.
- Total remaining service times of processes present at a node.
- CPU utilization at the nodes.

Relative advantages & disadvantages for policies:

(i) Total no. of processes: Straightforward to calculate but do not present an actual idea of load because of variation in processes.

(ii) Total remaining service time: Better accuracy of load obtained but estimation of remaining service time is tricky.

(iii) CPU Utilization: Very relevant for modern systems with extensive apps running all the time.

for load-sharing algorithms, both (5)
"Total number of processes" & "CPU Utilization"
can be used for load estimation.

Ans 5. Processor A: exponent = 8
mantissa = 16

Processor B: exponent = 12
mantissa = 32

Processor C: exponent = 16
mantissa = 64

In this distributed system, process migration may be allowed

(i) From processor A to processor B
(ii) from processor A to processor C
(iii) from processor B to processor C.

This is because data translation is simple when no. of bits get increased for both mantissa & exponent. However, on migrating from processor using more bits to a lesser one, data translation is problematic & throws overflow/underflow errors & thus, must not be allowed.

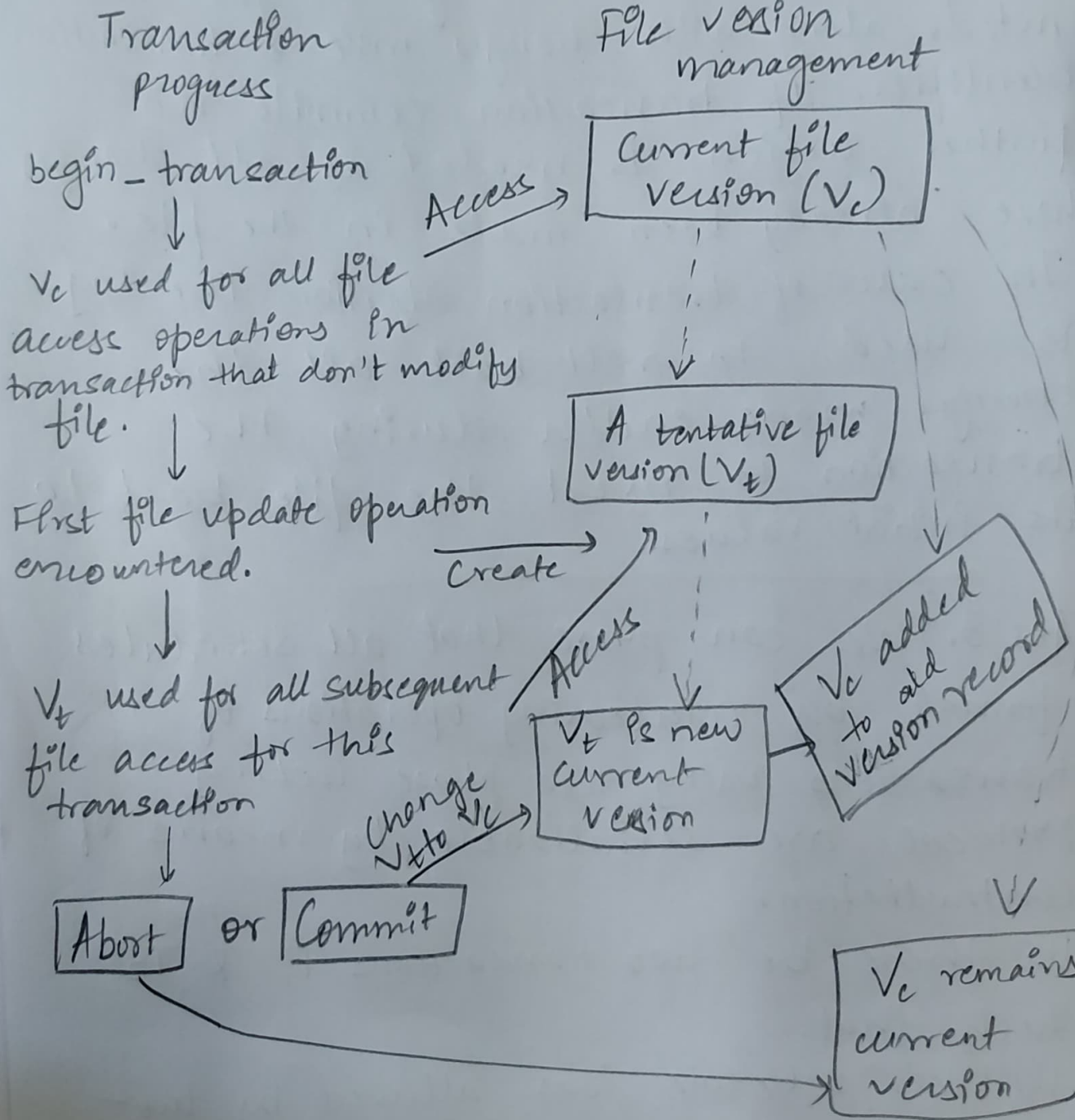
Ans 6. Yes it is necessary to always keep cached data up to date and consistent with the master copy in a distributed file system. This is especially important when it comes to time-sensitive information such as in a stock broking company where values change frequently & its necessary to stay updated at every decision.

Moreover, regular updation becomes even more necessary in a distributed environment as multiple copies of the same file may exist at different nodes & must be kept updated throughout the system.

Ans 7. The two commonly used methods that are used to design a file system which can store updates in a reversible manner are:

(i) File versions approach:- We avoid the overwriting of actual data in physical storage by creating a tentative version same as the current file & performing updates on this tentative copy. Once the

transaction commits, this new tentative file is deemed as the new current version & the old copy is added to the sequence of old versions maintained as a record. Also, if the transaction gets aborted then the tentative file is simply discarded without any effect on the current version.



(ii) Write-ahead log approach: In this method, for each operation of a transaction that modifies a file, a record is first created & written to a log file known as the write-ahead log file. After this, the file contents may be modified. The log is maintained in stable storage & all operations which change the file are noted, along with changes made & transaction identifier. If transaction commits, no further action is needed as all changes have already been made in the file. In case of transaction abortion, the log is used to roll-back all the changes made earlier during the transaction to revert the file back to its initial values.

Ans 8. We can prove that all schedules formed by interleaving operations of transactions using two-phase locking protocol are serializable by means of contradiction.

Let there be two transactions T_1 & T_2 such that

(i) their schedule was allowed by two-

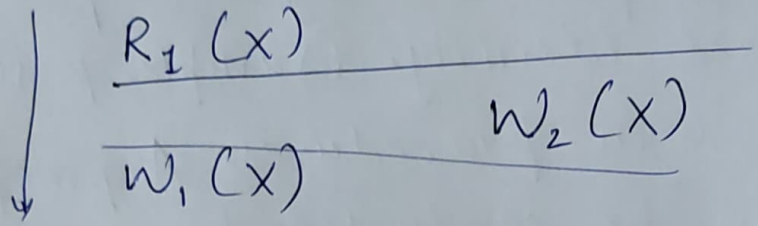
ase locking

(ii) It is not serializable.

We shall show that schedule cannot exist.

Since, we assume that the schedule of T_1 & T_2 is not serializable, a conflict exists.

Let this be example



Here, the transactions 1 & 2 are in conflict.

Now, in two-phase locking protocol,

(i) for $W_2(X)$ to follow $R_1(X)$, transaction 1 must release its lock on resource 'X' & thus, go into the shrinking phase.

(ii) Now, for $W_1(X)$ to follow $W_2(X)$, T_2 must release 'X' & T_1 must lock X. However, from statement (i), we already know that T_1 is in its shrinking phase & thus cannot lock a new resource.

Hence, this schedule is impossible in two phase locking. So, all schedules following two phase locking with interleaved operations are serializable. Hence proved

Ans 9. To ensure that server should serve (10)
the client only after verifying the client's identity, we can use the following techniques

(i) Proof of knowledge: The claimant must demonstrate knowledge of some information regarding the claimed identity that can only be known or produced by the principal with the claimed identity. Example, password logins in most login procedures. A proof of knowledge can be conducted by a direct demonstration like typing out a password or by indirect ways such as computing replies to challenges by a verifier.

Solely, direct demonstration suffers from a lack of security as if the password gets leaked, anyone can access the server.

(ii) Proof by Possession: The claimant produces an item that can only be possessed by a principal with claimed identity, ex. an ID badge or telephone number. The item must be unforgeable & safely guarded.

Usually a combination of both of these techniques is employed such as a password followed by OTP on mobile number.

to ensure two-fold security.

Ans 10. Static invocations require that the interfaces of an object are known when the client application is being developed. It also implies that if interfaces change, then the client application must be recompiled before it can make use of the new interfaces.

Dyamic invocation is different as it selects at runtime which method to invoke at the remote object. This gives the system great flexibility & reliability. Because it does away with the fixed Interface requirements.

It also helps uncouple the client & server in the fact that a change on either end does not gravely affect the architecture on the other side.

Compatability is greatly improved as well. Moreover, subtle variations in execution can be effected due to method overriding behaviours.

Ans 1. (a) A LAN-based distributed (12)
System: High speed & relatively low cost are the defining characteristics of LANs. Load balancing can be implemented to maximise the overall throughput of the system so as to ensure that resources are used in the most efficient manner.

(b) A WAN-based distributed system: Load balancing can be tricky due to the volume of processors & resources at hand. Moreover, in wireless WANs speed is not assured for all devices. Therefore, load sharing is a much better option than load balancing as optimal distribution of load is a very expensive calculation.

(c) A Distributed System based on processor-pool model: In this model, all processing is done by processing bank. Due to a large number of processors, load balancing will take some effort in terms of load estimation. However, process migration will be easy due to shared memory implementation.

d) A distributed system based on workstation-server model: The system consists of network of workstations where each workstation provides local processing & an interface to the network. So, process migration is an issue due to separate local disks for every work station. & Especially in unix systems, users must choose their workstation & so, the identification of idle systems become the responsibility of user. State information exchange between workstations can also pose problems.