Q1) Suppose you - - - - - - you

In centralised load balancing algorithm the responsibility resides on a single node. The issues associated with these algorithms are :

1) Reliability - the major issue of centralized load balancing algorithm is that of reliability as if the centralised system fails the whole process would cease to function. Thus the responsibility is on only one single centralized node.

Approaches to handle this issue :

- Replicate the server on k+1 nodes to protect against 'k' crashes.
- Rather than maintaining k+1 replications a single server is maintained an 'k' entities are present to detect failures. Whenever a failure is detected a new instance of server is brought up with the current state information.

2) Load Estimation Policies - For balancing the load for the nodes it is very important to calculate the load on each node. This is a big challenge as no true solution exists for it and different algorithms are parameters can be used based on requirements.

Approaches to handle this issue :

- Memoryless method - It assumes all processes have same expected remaining service time ; independent of time used so far ; thus it reduces the load estimation to number of processes.
- Pastrepeats - It assumes remaining service time to be equal to time used so far by it.

(III) Process transfer policies - Load balancing involves transfer processes from heavily loaded to lightly loaded nodes for this it is necessary to calculate & decide at what load is the node heavily loaded. So calculating the threshold can be challenging.

Approaches to handle the issue:

a) Static Policy - decide a fixed threshold for each node which is pre-determined ; thus no state information needs to be exchanged.

b) Dynamic Policy - the value can be calculated depending on the current state of the nodes.

(IV) Location Policies - After calculated heavily loaded nodes it becomes a challenge to determine the destination node for the process. This destination node should be selected such that the load is balanced.

Approaches to handle this issue:

a) Shortlist - transferring the process to node having minimum load value unless it cannot accept due to its own threshold.

b) Bidding - In this process a node in need of location to execute is called manager who broadcasts to all nodes who can accept the request (called contractors), these contractors propose bids based on various parameters and then best bid is selected.

v) Overhead Cost - A centralized load balancing algorithm can contain many cost overhead because the data & state information is stored at only one place.

Approach :
We can use a distributed load balancing approach or use the idea that strict consistency is not important.

---

Q2) A distributed ----- answer.

Location policy in load-sharing algorithm decides the destination node for transferring the process from heavily to lightly loaded nodes.

There are two kinds of location policy :

i) Sender initiated - In these process the heavily loaded process search for lightly loaded nodes where work can be transferred ; it can broadcast the message or randomly probe the nodes. ; the condition for transfer is that the receiver node's load doesn't exceed the threshold value.

ii) Receiver-initiated - Lightly loaded nodes search for heavily loaded nodes for transfer using broadcast or random probing. A node is viable if sending of process won't reduce its load below threshold value.

Sender-initiated algorithms ; scheduling decisions are made at arrival epochs whereas in receiver-initiated ; process departure ephochs are times when decisions are made.

3/2

If a system doesn't support preemptive process migration, it is better to use sender initiated policies because

A preemptive process migration allows transfer of executing process from one node to another.

Receiver initiated policies are mostly preemptive as it is unlikely that receiver node opens negotiation with potential sender at moment when new process has arrived.

Whereas in sender initiated policies the negotiation operation according to sender. A thus can be decided when process arrives, and hence no preemption is needed.

Q.5) Which one ... Why ?

(a) Data (RPC)

(b) High reliability is main goal.

There are two methods which can be used for this goal:

I) Total freezing - In this method the execution of process is stopped while address space is transferred.

It is reliable as the process will be in frozen state & hence execution will resume without errors.

II) Transfer on reference - It is based on assumption that process use only a relatively small part of address space while execution and thus address space is left behind on source node as relocation of process is done at destination node.

It attempts to reference memory pages results in generation of requests to copy in desired block from remote location.

It is highly reliable as address space stays at remote location

So the term transfer on reference should be considered as it is term provides better response time.

4/

(a) High performance is main goal

For high performance Pretransferring can be used. Pretransferring transfers process while it executes and is running on source node. Therefore when it is decided that the process needs to be transferred it is left running on source node until the address space is transferred.

It initially transfers address space by repeated transfers of pages modified during previous transfer until the modified page become relatively less. The remaining pages can then transferred after freezing process.

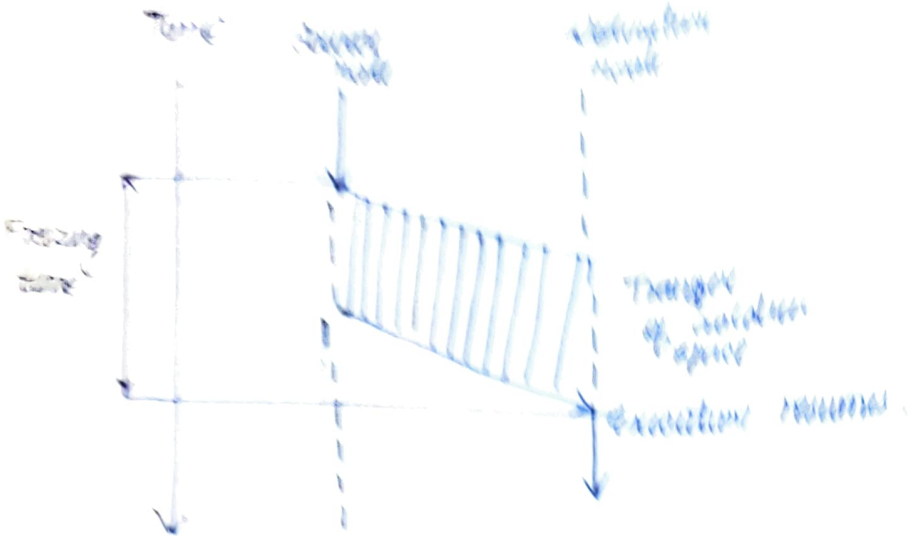
Since process is froze for a very less time, it provides high performance.

(c) Effectiveness of process is main goal

Transfer on reference proves effective policy as it transfers process while keeping its address space at source node and thus, process is transferred effectively and address space error doesn't occur.

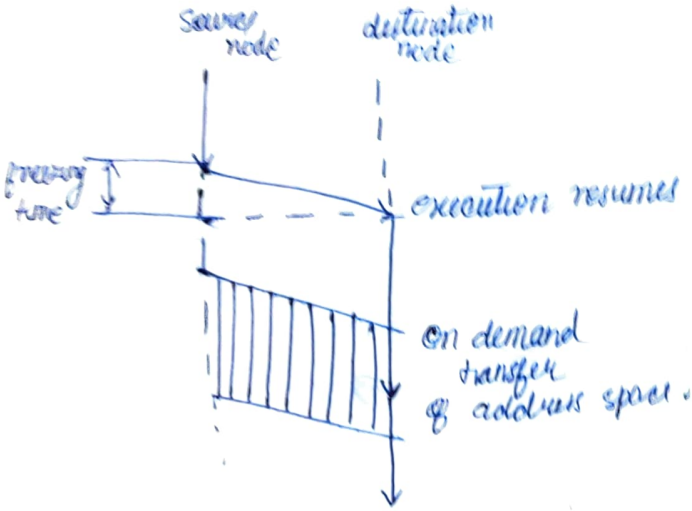
(d) Simple implementation

Total Freezing provides the simplest implementation as it stops process execution until the address space is transferred and so process is resumed only after the migration.




(c) Both reliability and effectiveness are important goals

Transfer on reference is best for effective migration and reliability of transfer as it transfers process but keeps address space at the source node and every request is handled between the two nodes. Later on by demand



Q4) Which of the ---- why?

a) Performance is main goal

Home Node or Origin Site Concept can be used if we want to focus on performance as it allows complete freedom of migrating process or subprocess independently and executing them on different nodes of system, all communication between the parent & child process take place via home node.

It allows better performance as it supports parallelism

b) Reliability is the main goal

i) Home Node or origin site concept provides reliability as the parent and child process can communicate via the home node; which is a reliable form of communication.

ii) The other method is to disallow the separation of the parent processes from their child procss as if they won't be separated the communication between the composses wouldn't be challenging and would be completely reliable.

Q the two methods Home Node is should be preferred according to me as it provides better performance.

(c) Simple Implementation

Disallowing separation of coprocesses policy should be used for this case. It can be of two types:

(I) By disallowing migration of processes that wait for one or more children to complete

(II) By making sure child processes are migrated along with their parent.

This method is very simple to implement as we don't allow coprocess to separate and thus preventing any challenges in their communication.

what is ---- shared file.

Q5)

I)

An immutable file refers to a file that cannot be modified after its creation. The file cannot be changed at all and the only possible function is to delete the file.

Immutable files use the approach of file versioning to implement updation of file. and thus makes the file management process reliable as it eliminates all the problems associated with storing the multiple copies of file.

However, immutable files suffer from two major drawbacks : Increase use of disk space and increased disk allocation activity

F)

Yes, it is possible for a file system to work correctly by using only immutable files. These files can allow all functions for shared file system and thus can be used. Infact, the semantics technically makes sure that problem of when to make changes made to file by user to other users disappears.

Create - This function can be performed as it is wherein a new file is created with given content λ is saved by given name.

This file after creation becomes an immutable file

9/20

b) **read** - Since immutable file cannot be allowed they are often referred as read-only files.

Thus read operation simply fetches the mentioned file from the memory and the user can read

c) **write** - Immutable files cannot be modified after creation and hence to perform writing operation file versioning approach is used and each file becomes a history of immutable versions.

∴ Instead of writing something on the same file a new version of file is created each time old version is returned as it is.

d) **Delete** - delete operations works the same way & the file is deleted from memory along with its all versions.

create (ABC, "some text")


read ("ABC.txt")


=> some text

write (ABC.txt, "change text)



delete (ABC.txt) ×

26) A distributed - - - - - replication

Caching refers to the idea of retaining recently accessed file data in main memory so that if the same file is repeatedly accessed it can be handled faster without much overhead. and increase the performance of file system

Replication's primary motive is to provide high availability it involves replicating the file and making its copies with each copy being stored in a different file server.

Caching and replication can be often used synonymously and they provide more or less the same goal of high availability however to think that either of them is sufficient is wrong because :-

a) Replica is associated with server and cache is associated with client.

b) Existence of a cached copy is primarily dependent on locality in file access patterns whereas replica depends on availability & performance requirements.

c) Replica is more persistent than cached copy.

d) A cached copy is contingent upon replica. only by periodic revalidation w.r.t replica can a cached copy become useful.

Caching makes sure that the frequently accessed files are readily available.

Replication makes sure all the files are always available even after server crashes.

19115081

(a) System with object caching and no replication

Advantages :

(I) Improved I/O performance because of caching
(II) Reduced disk transfer substantially
(III) Increased overall performance

Disadvantages

(I) System crashes will result in clearing of process as there is no replication
(II) Availability factor is low.
(III) Traffic congestion is possible

(b) System with replication but no caching

Advantages

(I) Increased availability of files because of their replicas in different servers.
(II) Increased reliability
(III) Response time is improved as data can be accessed from nearest server.
(IV) Network traffic is reduced
(V) Throughput of the complete system is increased.

Disadvantages

(I) There is no mechanism for accessing frequently accessed files easily & without delays. and thus the overall average response time is reduced.
(II) Consistency in replication becomes a challenge.

12

Both caching and Replication are implemented

Advantages

i) System is highly reliable with increased availability
ii) Response time for all files is gradually decreased
iii) Network congestion can be avoided easily
iv) System is more scalable and good in performance.

Disadvantage:

Since both caching and replication are implemented it becomes a challenge to revalidate the ~~replicas~~ cached data with respect to replica.

---

7) Why are ---- consistent matter?

Transactions are needed in file service because:

i) For improving the recoverability of files in event of failures:

Since atomic transactions have the proper of atomicity & consistency; the transactions revert back to original state if interrupted before completion.

And so in a file system if server or client crashes unexpectedly before transaction is completed the server will restore the files that were undergoing modification to their previous state thus preventing inconsistencies.

Without transaction; these inconsistent state would be impossible to recover from.

Example :

Considering a banking transaction with some operations ($O_1, O_2, O_3, O_4$), Two accounts (A, B) and transfer of ₹05 from A → B.

This transaction is carried out in order :

$O_1$ : read balance (a) of A
$O_2$ : read balance (b) of B
$O_3$ : write (a − 5) in A
$O_4$ : write (b + 5) in B.

Let initial amount be : A = ₹100, B = ₹100

Successful Execution =

$O_1$ : a = 100
$O_2$ : b = 100
$O_3$ : a = 95
$O_4$ : b = 105.

final ⇒ a = 95, b = 105

Unsuccessful execution (system crashes after $O_3$)

$O_1$ : a = 100
$O_2$ : b = 100
$O_3$ : a = 95

final : a = 95, b = 100

Thus there is an inconsistency as updation wasn't complete and so need transaction as if $O_1, O_2, O_3, O_4$ are treated as single txn, after crash the changes revert back and final values are : a = 100, b = 100 which is consistent.

ver allowing concurrent sharing of multiple files
nultable files by multiple clients in consistent manner

the request of access by multiple clients for a single
le is not handled synchronously the read &
rite requests of clients will overlap leaving the file
n inconsistent state and can have unpredictable effects
ansactions are needed to serialise the access requests
m multiple clients.

Example :

nsidering two banking transactions $T_1$ and $T_2$
olving 4 operations :

$y_1$ : read balance (a) of A
$y_2$ : read balance (b) of B
$y_3$ : write (a-10) on A
$y_4$ : write (b+10) on B

} transfer ₹10 A → B

$z_1$ : read balance (c) of C
$z_2$ : read balance (b) of B
$z_3$ : write c-5 from C
$z_4$ : write b+5 on B

} transfer ₹5 C → B

f no serial scheduling is followed then :
et's assume initial amount of all accounts to be
00 each no scheduling the two txn will overlap.

possible outcome

$y_1$ : $a = 100$
$z_1$ : $c = 100$
$y_2$ : $b = 100$
$z_2$ : $b = 100$
$y_3$ : $a = 90$
$z_3$ : $c = 95$
$y_4$ : $b = 110$
$z_4$ : $b = 105$

final outcome : $a = 90$, $c = 95$, $b = 105$

which is wrong

Thus, if synchronization of txn is used then correct result will be produced

legal :

$y_1$ : $a = 100$
$z_1$ : $c = 100$
$y_2$ : $b = 100$
$y_3$ : $a = 90$
$y_4$ : $b = 110$
$z_2$ : $b = 110$
$z_3$ : $c = 95$
$z_4$ : $b = 115$

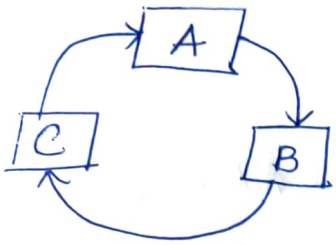
final : $a = 90$, $c = 95$, $b = 115$

which is correct

19115031

2) What is ---- cases (a) and (b).

Transaction deadlock is an illegal state in which transaction waits for a data item locked by another transaction that in turn waits, via a chain of other waiting transactions. For the first transaction to release its locks. Since transaction cannot release any lock until it finishes none of transactions can move unless one is aborted




A is waiting for B
B is waiting for C
C is waiting for A
A, B, C are in a deadlock.

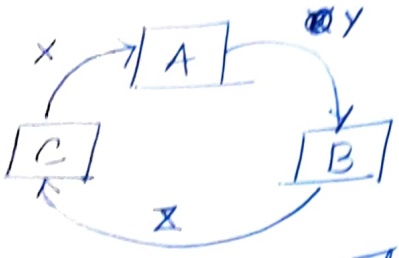
(a) Same types of lock are used

A is a transaction that needs to transfer money from X → Y. ; after reading the value from X it needs to acquire a lock on Y to read its value & update it. It has a lock on X.

B is a transaction that needs to transfer money from Y → Z and it needs a lock on Z. ; it has a lock on Y already.

C is a transaction that needs to transfer money from Z → X it has a lock on Z but needs a lock on X.

17/20



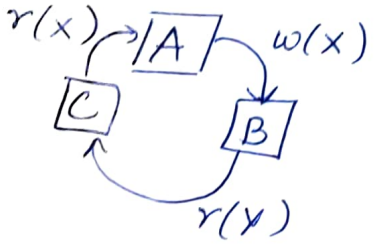

∴ A, B, C are waiting on each other this is a deadlock situation.

(b) Different type - specifics locks are used

Transaction A has a read lock on X but after the execution it also requires write lock on X.

Transaction B has a write lock on X, but it required a read lock on Y to complete

Transaction C has a read & write lock on Y but to function it needs a read lock on X as well




∴ A, B, C are in circular wait this is a condition of deadlock.

method by which the deadlock can be prevented is:

wait scheme

this method if a transaction requests for a resource held by another transaction then the timestamp of both transactions are compared.

the request is made by an older txn then the younger transaction is killed to release resource and it is restarted with same timestamp

if the younger txn is requesting for a resource it is only asked to wait.

method can be applied for both the cases.

Suppose timestamps: $A < B < C$.

Thus A is older than B; so, B will be killed and restarted by that time A will complete its execution after which C can complete its execution as resources will be released and then B would be free to complete the transaction

19115081

Q9) The password ------ workstations.

(a) The login program and password file can be stored in each workstation for faster access. These files are however encrypted to prevent any attack.

(b) The password file can be stored on a centralized server which can communicate via all the workstations.

(c) In a processor-pool model the password file can be stored on a centralized server or decentralized servers in communication with all nodes.

The password file stored anywhere is vulnerable to attacks & threats and therefore it is important that these files are stored using suitable encryption which cannot be deciphered without using any key or authentication service.

---

[1911508]

... it possible ------ systems?

... (Common Object Request Broker Architecture)
... standard defined by the Object Management
... (OMG) that enables software components written
...ltiple computer languages and running on multiple
...puters to work together.

...RBA is a a standard for distributing objects across
...tworks so that operations on those objects can be
...ked remotely.

...RBA is not associated with any programming language
...ny language can be used with a CORBA binding
...o call and implement CORBA objects.

...t is possible to have system specific implementations &
...RBA object references while still being able to
...change references with other CORBA-based system
...ecause:

...) CORBA enables a collaboration between systems with
different operating systems. Even though it might be
specific for a system with particular operating system
it can still collaborate with others as long as they
are based on CORBA.

21 /

(b) CORBA has no specific programming language i.e. it doesn't define a fixed programming language for calling and accessing objects. Any programming language with CORBA binding can function.

(c) CORBA is not dependent on the computing hardware of the system and can exchange reference & communicate with different hardware, of different systems based on the CORBA.

— END —

22/22