

Previous Year Qns:

Question1.)

Linda is a homogeneous distributed system. It is found that 55% of nodes remain busy at day hours and rest 45% found idle at that moment. Node X of Linda has 90% of workload and has to migrate few processes to some idle node. Give your arguments in favour / against following statements on the basis of given scenario:

- a. Pairing will be chosen as location policy.
- b. State information exchange policy should be "Broadcast when State Changes".
- c. Controlled migration policy must be used with migration count (k) 1.
- d. An idle node executes migration request in a selfish manner

Solution:

(a.) Pairing will be chosen as location policy:

** Arguments in favour:*

Pairing can help to improve load balancing by distributing the workload of node X across two nodes.

Pairing can also help to improve fault tolerance, as if one node in the pair fails, the other node can continue to operate.

** Arguments against:*

Pairing can increase the complexity of the system, as it requires additional mechanisms to maintain the pair relationship.

Pairing can also reduce the flexibility of the system, as it makes it more difficult to migrate processes between different nodes.

Overall, whether or not to use pairing as a location policy will depend on the specific requirements of the system. In the case of Linda, pairing could be a good option if the goal is to improve load balancing and fault tolerance.

(b.) State information exchange policy should be "Broadcast when State Changes":

*** Arguments in favour:**

Broadcasting state changes ensures that all nodes have the most up-to-date information about the system.

This can be important for maintaining consistency and coordination between nodes.

*** Arguments against:**

Broadcasting state changes can generate a lot of network traffic, which can impact performance.

Broadcasting can also be vulnerable to denial-of-service attacks.

Overall, whether or not to use a "Broadcast when State Changes" policy will depend on the specific requirements of the system. In the case of Linda, broadcasting state changes may be a good option if the goal is to ensure consistency and coordination between nodes. However, it is important to consider the impact on performance and security before making a decision.

(C.) Controlled migration policy must be used with migration count (k) =1:

*** Arguments in favour:**

Using a controlled migration policy can help to minimize the impact of migration on the system.

By limiting the number of processes that can be migrated at a time, the migration policy can help to avoid overloading the network and reducing performance.

*** Arguments against:**

Using a controlled migration policy can slow down the migration process.

It is also important to choose a migration count (k) that is appropriate for the system. If k is too low, it may take a long time to migrate all of the processes from node X. If k is too high, it could overload the network and reduce performance.

Overall, whether or not to use a controlled migration policy will depend on the

specific requirements of the system. In the case of Linda, using a controlled migration policy with a migration count $(k) = 1$ could be a good option if the goal is to minimize the impact of migration on the system.

(d.) An idle node executes migration request in a selfish manner:

****Arguments in favour:***

An idle node may want to execute migration requests in a selfish manner to maximize its own performance.

For example, an idle node may want to prioritize migration requests from nodes that are similar to it in terms of workload or resource requirements.

****Arguments against:***

Executing migration requests in a selfish manner can lead to poor load balancing and reduced performance for the overall system.

It is important to consider the needs of the entire system when making decisions about where to migrate processes.

Overall, whether or not to allow idle nodes to execute migration requests in a selfish manner will depend on the specific requirements of the system. In the case of Linda, it is important to consider the impact on load balancing and performance before making a decision.

Question:2)

An application has been built where the resources are replicated and can also migrate. The number of replica for any resource could be more than 100. The protocols used for ensuring consistent copy of all replicas is sequential consistency, in which the write operation is carried out by updating all copies of the data on which write is performed. Explain the mechanism by which the sequential consistency can be achieved for the above mentioned scenario. It is to be ensured that system maintains high quality of service.

Answer:

To achieve sequential consistency in a replicated system with resource migration, the following mechanism can be used:

**** Maintain a global total order of all write operations.***

This can be done using a variety of techniques, such as a centralized sequencer or a distributed consensus algorithm.

**** When a write operation is received by a replica, the replica updates its copy of the data and then forwards the write operation to all other replicas in the order specified by the global total order.***

**** Replicas do not respond to read requests until they have received and applied all write operations up to and including the last write operation in the global total order.***

This mechanism ensures that all replicas see the same sequence of write operations, even if resources migrate between replicas.

To maintain high quality of service, the following techniques can be used:

**** Use a distributed consensus algorithm to maintain the global total order of write operations. This will help to ensure that the system is scalable and fault-tolerant.***

**** Use caching to reduce the number of times that replicas need to access the shared data.***

**** Use load balancing to distribute the workload evenly across replicas.***

Question 3)

In a distributed shared memory system, if w11 and w12 are two write operation performed by a process P1 in that order and w21 and w22 are two write operation performed by a process P2 in that order. What will be the possible order of operation that process P3 and P4 may see if the system follows:

- a) PRAM (pipelined random access memory) consistency model?**
- b) Processor consistency model?**

a) PRAM (pipelined random access memory) consistency model:

Under the PRAM consistency model, all processes see the same order of write operations. This means that the possible order of operations that processes P3 and P4 may see is:

w11 -> w12 -> w21 -> w22

This is because the PRAM consistency model guarantees that all write operations are seen in the order in which they were issued, regardless of which process issued them.

b) Processor consistency model:

Under the processor consistency model, each process sees its own write operations in the order in which they were issued, but it may see other processes' write operations in a different order. This means that the following possible orders of operations are allowed:

w11 -> w21 -> w12 -> w22

w11 -> w12 -> w22 -> w21

This is because the processor consistency model does not guarantee that all processes see write operations in the same order.

It is important to note that the PRAM consistency model is a stricter consistency model than the processor consistency model. This means that any system that implements the PRAM consistency model will also implement the processor consistency model.

Question 4)

In distributed systems, the choice of call semantics depends on the specific requirements and characteristics of the application. Here are suggestions for which call semantics may be suitable for each of the given applications, along with reasons:

(a) For making a request to a time server to get the current time:

Suggested Call Semantics: Exactly-Once

Reason: Getting the current time is a read-only operation, and it is essential to ensure that the response is accurate and not duplicated. Using exactly-once semantics guarantees that the time request is executed precisely once.

(b) For making a request to a node's resource manager to get the current status of resource availability:

Suggested Call Semantics: Maybe

Reason: Resource availability information may change frequently but is not necessarily critical if an occasional request is missed or duplicated. Maybe semantics provide a balance between performance and reliability.

(c) For periodically broadcasting the total number of current jobs by a process manager in a cooperative system:

Suggested Call Semantics: Last-One

Reason: In this scenario, the goal is to inform other nodes about the total number of jobs, and it's essential to ensure that only the latest information is propagated.

(d) For making a request to a computation server to compute the value of an equation:

Suggested Call Semantics: At-Least-Once

Reason: It is crucial to ensure that the computation request is not lost, and it's acceptable to tolerate duplicates. At-least-once semantics ensures that the computation is performed at least once, which can be compensated for if needed.

(e) For making a request to a booking server to get the current status of seat availability:

Suggested Call Semantics: Maybe

Reason: Seat availability can change frequently, but occasional missed or duplicated requests are usually acceptable. Maybe semantics provides a good balance of reliability and efficiency.

(f) For making a request to a booking server to reserve a seat:

Suggested Call Semantics: Exactly-Once

Reason: Reserving a seat is a critical operation, and duplicating the reservation can lead to overbooking. Exactly-once semantics ensures that the reservation is processed only once.

(g) For making a request to a file server to position the read-write pointer of a file:

Suggested Call Semantics: Exactly-Once

Reason: File pointer positioning is a critical operation, and duplications or missed requests could lead to incorrect file access. Exactly-once semantics ensures precise positioning.

(h) For making a request to a file server to append a record to an existing file:

Suggested Call Semantics: At-Least-Once

Reason: Appending records is important, but duplicates can be handled or identified and removed if necessary. At-least-once semantics ensures that the record is appended at least once.

(i) For making a request to a name server to get the location of a named object in a system that does not support object mobility:

Suggested Call Semantics: Maybe

Reason: In a system without object mobility, the location of named objects is relatively stable, and occasional missed or duplicated requests are typically acceptable.

(j) For making a request to a name server to get the location of a named object in a system that supports object mobility:

Suggested Call Semantics: Last-Of-Many

Reason: In a system with object mobility, object locations may change frequently. Last-of-many semantics ensures that the request receives the most up-to-date information about the object's location.

Question 5)

Q2. Consider a broadcasting method of data locating in RMB. What shall be the status of owner table of node 2 and node 3, when node 2 generates a write fault for block 6000 for following scenario: [5]

- When the requested block is replicated
- When the requested block is migrated

The present owner table is as follows:

Node 1	
Block Address	Copyset
1000	1
2000	1

Node 3	
Block Address	Copyset
5000	3
6000	3

Node 5	
Block Address	Copyset
9000	5
10000	5

Node 2	
Block Address	Copyset
3000	2
4000	2


Node 4	
Block Address	Copyset
7000	4
8000	4

Answer :

When the requested block is replicated:

When Node 2 generates a write fault for block 6000, it will broadcast a write request to all nodes in the system. All nodes that have a copy of block 6000 will respond to the write request by sending their copy of the block to Node 2. Node 2 will then update its owner table to reflect that it is now the owner of block 6000.


The owner table of Node 2 and Node 3 will be as follows:

Node	Block Address	Copyset
Node 2	6000	{2, 3}
Node 3	6000	{2, 3}
 Export to Sheets		

When the requested block is migrated:

When Node 2 generates a write fault for block 6000, it will broadcast a write request to all nodes in the system. The current owner of block 6000, Node 3, will respond to the write request by sending its copy of the block to Node 2. Node 2 will then update its owner table to reflect that it is now the owner of block 6000. Node 3 will also update its owner table to reflect that it no longer owns block 6000.

The owner table of Node 2 and Node 3 will be as follows:

Node	Block Address	Copyset
Node 2	6000	{2}
Node 3	-	-
 Export to Sheets		

Note: The copyset field in the owner table contains a list of all nodes that have a valid copy of the block.

Question 6)

Q1. Consider a scenario consisting of two Sites (systems) S1 and S2. S2 consists of P3 & R3. Rest of the processes and resources reside at S1. The request and allocation of the resources by various processes are R1→P1, R2→P2, R3→P3, P2→R1, P3→R2, P1 releases R1 & R1→P2, P1→R3. If we apply Centralized Approach for Deadlock Detection, is the distributed system deadlocked? What happens when the message P1 releases R1 is delayed? [5]

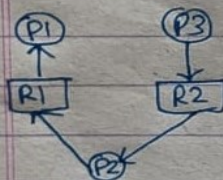
Answer)

Q1) Sites : S1 and S2
Processes : P1, P2 and P3
Resources : R1, R2 and R3

Request and allocation process is as follows:

- Step 1 P1 requests for R1 and R1 is allocated to P1
- Step 2 P2 requests for R2 and R2 is allocated to P2
- Step 3 P3 requests for R3 and R3 is allocated to P3
- Step 4 P2 requests for R1 and waits for it
- Step 5 P3 requests for R2 and waits for it
- Step 6 P1 releases R1 and R1 is allocated to P2.
- Step 7 P1 requests for R3 and waits for it

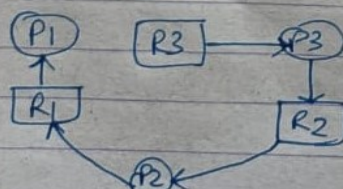
Resource allocation graph of S1



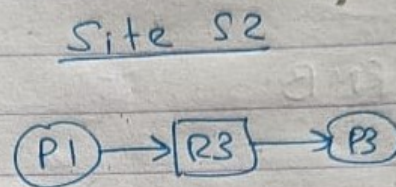
Resource allocation graph of S2



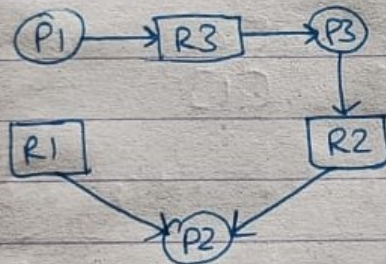
Resource Allocation graph of central coordinator



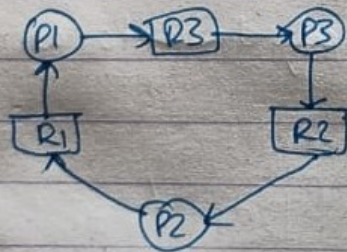
Resource allocation graph after
step 7 :



Central Coordinator



false deadlock if step 7 ^{will} ~~if~~ occurs
before step 6



Soln: use Lamport's algorithm to append
unique global time stamp.