Semester: 5th
Subject Name:- DOS & Code:- CS 30009
Branch (s): CSE

AUTUMN MAKE UP MID SEMESTER EXAMINATION-2024

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Subject : Distributed Operating Systems
[Code: CS 30009]

Time: 1 1/2 Hours **Full Mark: 20**

Answer all the questions.
The figures in the margin indicate full marks.
Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

1. Answer all the questions. [1 Mark × 5]

a) How does a distributed system run on a collection of networked machines act like a virtual uniprocessor?
Answer: The distributed system is designed to present a single, cohesive interface to users and applications, masking the complexity of the underlying hardware. The features are Single System Image, Shared Memory Abstraction, Global Process Scheduling, Distributed Synchronization, Load Balancing, and Fault Tolerance through replication. In general, a distributed system inherently consists of multiple independent machines, however, due to these mechanisms an illusion of a single, unified processing environment is created.

b) In RPC, the client process calls a procedure **usepro**(x, y, z), where x is of char type, y is of float type and z is an array of 5 integers. In this context how does the client stub marshal the message for the server, and then send it to the server stub?
Answer: The client stub marshals the parameters in the message as below

usepro	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

c) During the process synchronization, when a transaction commits, then what would be the status of the private indices?
Answer: If the transaction is committed, the private indices are moved into the parent's workspace atomically, and the blocks that are no longer reachable are put onto the free list.

d) What is the significance of graceful degradation to ensure process resilience?

Answer: During graceful degradation, the system continues to operate at reduced functionality instead of failing. For example, if a service becomes unavailable, the system might reroute requests or provide limited functionalities instead of total unavailability. So, due to process resilience, a system continues functioning correctly even during failures or unexpected conditions.

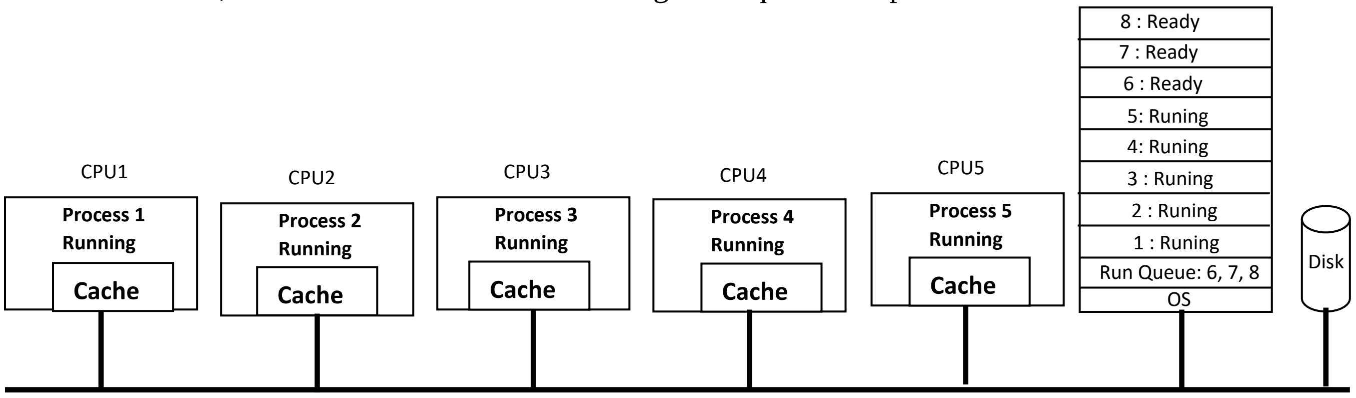
e) In a Bus-Based multiprocessor system with only one bus, which mechanism reduces the overloading of the Bus and how?
Answer: Each CPU is equipped with a snooping cache. The cache consistency protocol makes sure that different caches do not contain different values for the same memory location. One particularly simple and common protocol is called write-through.

2. a) Draw the block diagram of a multiprocessor timesharing system with 5 CPUs, 8 processes, one shared memory, and one hard disk, and all are connected via a single run queue. Explain the working principle. Also, differentiate it from a distributed system. [2.5 Marks]

Answer:

Evaluation scheme: Block diagram: 1 mark, Working principle: 1 mark; Difference: 0.5 mark

A multiprocessor timesharing system with 5 CPUs, 8 processes, one shared memory, and one hard disk, where all are connected via a single run queue is represented below:



Working Principle: All the processes are normally present in the shared memory and five processes are in the running state. Remaining three processes are in ready to run state (i.e. processes 6, 7, and 8 are in the run queue). Mutual excusion is achieved by synchronization priciples.

Difference:

Parameter	Distributed System	Multiprocessor System
How many copies of the OS are there?	N	1
How is communication achieved?	Messages	Shared Memory
Are agreed upon network protocols required ?	Yes	No
Is there a single run queue?	No	Yes
Does it look like a virtual uniprocessor	Yes	Yes

b) In the Client-Server communication mechanism, mention the five types of addressing schemes, their working principles, and limitations. [2.5 Marks]

Answer: Evaluation scheme: Each method and its limitations: 0.5 * 5 = 2.5

Method 1: Send messages to processes. *Limitation*: Congestion in a large pool of processes.

Method 2: Each process is assigned a local-id and informs the kernel that it listens to local-id. The addressing scheme is machine. local-id. *Limitation*: No transparency

Method 3: Each process has a unique address that does not contain the machine number. A centralized process address allocator maintains a counter. Upon receiving a request, it returns the current value of the counter. *Limitation:* They do not scale well.

Method 4: Each process picks its identifier from a large address space. *Limitation*: Broadcasting overhead

Method 5: For the first time client sends a query to the Name Server, asking for the machine number where the server is currently located. Then the request can be sent directly to the machine address. *Limitation:* If the name server is replicated, then there would be a consistency problem.

3. a) Consider a Distributed System with a set of sites, S={S_1, S_2, S_3}. Concerning real time, t, the global state G={LS_1, LS_2, LS_3} (where LS_i is the local state of site S_i) is recorded.

Let LS_1={recv(e_{21}, e_{11}), send(e_{12},e_{31}), recv(e_{22}, e_{13})}

Let LS_2={send(e_{21}, e_{11}), send(e_{22}, e_{13}), recv(e_{14}, e_{23})}

Let LS_3={recv(e_{12}, e_{31}), send(e_{32}, e_{15}), recv(e_{16}, e_{33})}

Where e_{ij} is the j^{th} event at the i^{th} site; send/recv(x,y)-send/receive a message from x to y. Put your comment on the recording of the global state G with justification. [2.5 Marks]

Answer: Evaluation scheme: Explanation of each local state: 0.5 * 3 = 1.5 marks; Justification : 1.0 marks

In a distributed system, a global state G represents the combination of the local states of all the sites in the system at a given global time t. It helps to track the state of each site and the events that occurred, such as sending or receiving messages between the sites. The global state G = {LS1, LS2, LS3} in this case is composed of the local states LS1, LS2, and LS3, representing the states of sites S1, S2, and S3, respectively.

Each local state consists of the messages sent and received by that site up to time t. We have to break down the events and check whether the global state has been correctly recorded or if there are any occurrences of inconsistency.

Site S1: LS1 = {recv(e21, e11), send(e12, e31), recv(e22, e13) }

recv(e21, e11): Site S1 receives a message e21 from S2and records event e11.

send(e12, e31): Site S1 sends a message e12 to S3 and records event e12.

recv(e22, e13): Site S1 receives another message e22 from S2 and records event e13.

Site S2: LS2 = {send(e21, e11), send(e22, e13), recv(e14, e23)}

send(e21, e11): Site S2 sends a message e21 to S1 and records event e11.

send(e22, e13): Site S2 sends a message e22 to S1 and records event e13.

recv(e14, e23): Site S2 receives a message e14 from S3 and records event e23.

Site S3: LS3 = {recv(e12, e31), send(e32, e15), recv(e16, e33)}

recv(e12, e31): Site S3 receives a message e12 from S1 and records event e31.

send(e32, e15): Site S3 sends a message e32 to S2 and records event e15.

recv(e16, e33): Site S3 receives a message e16 from S2 and records event e33 .

Checking for the Consistency:

The key principle when recording the global state in a distributed system is that the order of events and the causality between them must be respected. That is, if event e1 causally affects event e2, then e1 should be recorded before e2 in the global state. This requires consistency between the send/receive events at different sites.

Causality and Global State:

recv(e21, e11) at S1 and send(e21, e11) at S2 are part of a causal pair, meaning S1's receiving of message e21 must occur after S2 sent it. Since event e21 is sent at S2(event e11) and received at S1 (event e21), this ordering is consistent.

recv(e22, e13) at S1 and send(e22, e13) at S2 are another causal pair. Event e22 is sent by S2 (event e13) and received by S1 (event e22), so this ordering is also correct.

recv(e12, e31) at S3 and send(e12, e31) at S1 represent a causal pair where S1 sends message e12 to S3, and S3 receives it. This is consistent.

recv(e32, e15) at S3 and send(e32, e15) at S2 are also part of a causal pair, and the ordering is consistent as event e32 is sent by S3 and received at S2.

The global state G keeps track of every sent and received event consistently and maintains consistent tracking of causality

b) Consider a bank database that is fully replicated. Give an algorithm/protocol for ordering transactions in this situation. [2.5 Marks]

Answer: In a fully replicated bank database, ensuring the correct ordering of transactions is crucial for maintaining consistency and preventing issues like double spending. One common approach to achieving this is through the Total Order Broadcast protocol. Here’s a high-level outline of an algorithm that can be used to order transactions in a fully replicated system:

Total Order Broadcast Algorithm

i. Leader Election: A leader (or coordinator) is elected among the replicas to manage transaction ordering.

ii. Transaction Proposal: Transactions are proposed to the leader for ordering.

iii. Consensus on Order: All replicas agree on the order of transactions.

iv. Broadcast and Execution: Once the order is decided, transactions are broadcasted to all replicas for execution in the agreed order.

Steps of the Protocol

(i) Leader Election: Use an algorithm like Bully or Paxos to elect a leader among the replicas. The leader will be responsible for ordering the transactions.

(ii) Transaction Proposal: When a client submits a transaction, it sends the transaction to the leader.

The leader collects incoming transactions from clients and assigns them a unique identifier (timestamp, sequence number, etc.) for ordering.

(iii) Ordering Phase: The leader proposes the order of transactions based on their arrival or assigned timestamps. The leader sends the ordered list of transactions to all replicas.

(iv) Consensus Phase: Each replica receives the proposed order and must confirm it. Replicas respond with an acknowledgment. If a majority of replicas acknowledge the order, it is considered committed. If a replica does not acknowledge, the leader may need to retry or initiate a new consensus round.

(v) Execution Phase: Once consensus is reached, the leader broadcasts the committed transaction order to all replicas. Each replica applies the transactions in the agreed order.

(vi) Handling Failures: In case of a leader failure, the remaining replicas will initiate a new leader election. Replicas should also implement a mechanism to handle stale transactions and rollbacks if necessary.

4. a) Mention the five different types of Data-centric consistency models with their features. Explain the Linearizability with a suitable example. [2.5 Marks]

Answer: Evaluation scheme: Each Model with their features = 2.0, Example: 0.5

The five different types of Data-centric models are :

(i) Strict

(ii) Linearizability

(iii) Sequential

(iv) Causal

(v) FIFO

Example: Linearizability: Any proper example with valid sequences for the processes with respect to the time sequence.

b) Why there is a requirement for “home memory” in Memnet but not in Dash? Briefly explain the reasons for thereof. [2.5 Marks]

Answer: Memnet is one of the distributed shared memory systems with ring-based multiprocessors. In Memnet, a single address space is divided into a private part and a shared part. The private part is divided up into regions so that each machine has a piece for its stacks and other unshared data and code. The shared part is common to all machines(and distributed over them) and is kept consistent by a hardware protocol roughly similar to those used on bus-based multiprocessors. Share memory is divided into 32 blocks, which is the unit in which transfers between machines take place. All the machines in Memnet are connected in a modified token-passing ring. The ring consists of 20 parallel wires, which together allow 16 data bits and 4 control bits to be sent every 100 nsec. The ring interface, MMU, cache, and part of the memory are integrated into the Memnet device. In Memnet there is no centralized global memory. Instead, each 32-byte block in the shared address space has a home machine on which physical memory is always reserved for it, the Home memory field.

In the Dash machine, there are 16 clusters, each cluster containing a bus, four CPUs, 16M of global memory, and some I/O equipment (disk, etc.). The total address space available in the prototype is 256M, divided into 16 regions of 16M each. The global memory of cluster 0 holds addresses 0 to 16M. The global memory of cluster 1 holds addresses 16 M to 32M, and so on. Memory is cached and transferred in units of 16-byte blocks, so each cluster has 1M memory blocks within its address space.