Name : Sanchi Shukla
Roll No : 2301410028
Course : B.Tech CSE (Cyber Security)

Date			
Page No.			

Assignment-4

PART-A

Ans 1) A race condition happens when two people access the shared resource at the same time.
Eg: Two people withdrawing money from the same bank account simultaneously → incorrect balance.
Mutual exclusion allows only one person to access the shared resource at a time.

Ans 2) Peterson's solⁿ is complicated, works only for two processes & depends on strict hardware ordering.
Semaphores are simple, scalable & use hardware supported atomic instructions like test & set making them practical for real system.

Ans 3) Advantage of monitors in multi-core system monitors handle locking & unlocking automatically this reduces synchronization errors & makes multi-core programming safer & easier.

Ans 4) Starvation occurs when continuous readers keep entering & the writer never gets a chance.
Prevention → ~~the~~ Writers never gets a chance, use writers-priority or a fair queue (FIFO). So each writer eventually get access.

Ans 5) To eliminate hold & wait, a process must request all resource at once before starting.
Practical drawback: Many resource stay idle & wasted because a process holds them even when it doesn't need them yet. This greatly reduces system resource utilization.

Date | Page No

Ans6) a) $P_1, P_2, P_3, P_4, P_5, P_6$

Global wait-for graph:

$$P_1 \longrightarrow P_2 \longrightarrow P_5 \longrightarrow P_6 \longrightarrow P_1 \quad \text{(Cycle)}$$

$$P_3 \longrightarrow P_4 \quad \text{(No cycle)}$$

b) A deadlock exist if there is a cycle in the wait-for graph.
We have the cycle $P1 \rightarrow P2 \rightarrow P5 \rightarrow P6 \rightarrow P1$.
Process involved in the deadlock $\rightarrow$ P1, P2, P5, P6.

c) Suggest one distributed deadlock detection: ~~Cloud~~ Chandy – Misra – Haas (CMH) probe algorithm is stored choice for distraction deadlock detection.

How it works:
A process that suspect it is blocked (or a site periodically) initiates probe messages.

Ans7) Given, Local access time, $T_L = 5ms$
Remote access time, $T_R = 25ms$
Probability file is remote, $P = 0.3$

a) Expected time E(T) is:

$$E(T) = P \cdot T_R + (1-P) \cdot T_L$$

Avg. numbers:

$$E(T) = 0.3 \times 25 + (1-0.3) \times 5 \Rightarrow 7.5 + 3.5 \Rightarrow 11.0 \text{ ms}$$

b) Suggested strategy: Client-side LRU caching with write-back for read-heavy file \$ validation TTL.

Justification :

→ LRU (Least Recently Used) : Works well in-practice because file access pattern often have temporal locality — recently used file are likely to be reused.

→ Write-back with validations TTL : For performance, write can be batched (write-back) but use short TTL or invalidation message to ensure consistency : this balance lower latency & acceptance.

Ans8) Given, Full check point cost = 200 ms
Incremental checkpoint cost = 50 ms
RPO requirement = Atmost 1 second
Period plan = 10 sec

a) Proposed optimal min :
Steps : Take one full checkpoint every 10s.
Take incremental checkpoint every is b/w full checkpoints.

# Total checkpoint overhead per 10s :
Fulls = 1 × 200 ms = 200 ms
Incremental = 9 × 50 ms = 450 ms
Total = 650 ms for 10s
Average overhead = 65 m/s

b) Reasoning :

→ RPO constraint : The system must be prepared to ~~restore~~ restore a state no older than 1s. Therefore, an incremental checkpoint must be taken atleast once per second.

→ Full checkpoint are expensive (200 ms), doing them less frequently reduces heavy overhead full checkpoint reduce recovery time because only one full.

→ Tradeoff: Frequent incremental add modest overhead but keep RPO tight. Periodic fulls keep recovery efficient.

Ans 9) a) Key challenges:
1) Sudden bursty traffic: Requests spike order of magnitude within seconds.
2) Geographic latency & data locality: User should be served from the nearest region when possible.
3) Hot-spots / skewed load: Certain product / regions get disproportionate attention.
4) Fairness & SLA guarantees: Mission-critical request (payments) must get prioritized.

Hybrid global dispatcher using weighted least connection.
→ Weighted least-connection: Take server capacity into account (weights) & sends new request to the least busy servers — effective under non-uniform, capacities.

→ Adaptive local queueing: Each region maintain a local queue & admission control to avoid overload request exceeding local capacity can be forwarded to other region.

→ Work stealing: Underutilized regions can pull tasks from overloaded regions to smooth spikes.

b) Fault tolerance strategy (RTO & RPO focused):
1) Active-Active Multi region deployment: Deploy application stacks active in atleast two geographically seperated region.

Each region handles local traffic, If one region fails, traffic automatically shift to healthy regions.

2) Durable logs & checkpointing : Maintain an append - only transactions log with multi - region replication. On failover, replay log to reach consistent state.
Combine with frequent checkpoint to bound recovery work.

3) Automatic failover & health checks : Global load balancer & control plane perform health probes & route traffic away from unhealthy region automatically.
Use traffic & gradual failover to avoid overloading remaining regions