ABHINAV JHA
230101021
B.TECH CSE CORE - 4

Page No.: 209

Assignment - 4

1. Two clerks share a single physical ledger book. Both read the current balance, each adds a deposit amount on their copy, and both write back. If one writes after the other, one deposit is lost - result depends on timing (a race). Mutual exclusion fix: Use a checkout for the ledger. A clerk must acquire the lock before reading/updating & release it after.

2.5 Part B

6. $S_1 : P_1 \to P_2, \; P_3 \to P_4$
$S_2 : P_2 \to P_5, \; P_5 \to P_6$
$S_3 : P_6 \to P_1$

a) Edges = $P_1 \to P_2$, $P_3 \to P_4$, $P_2 \to P_5$, $P_5 \to P_6$, $P_6 \to P_1$

b) Deadlock detection: Look for cycles : $P_1 \to P_2 \to P_5 \to P_6 \to P_1$ forms a cycle. Process involved : P_1, P_2, P_5, P_6 ($P_3 \to P_4$ is seperate & not in cycle)

c) Suggested distributed algorithm : Chandy - Misra - Haas (edge chasing).

7 a) Expected file access time calculation: Prob(local) = 1 - 0.3 = 0.7

Expected time = $0.7 \times 5\,ms + 0.3 \times 25\,ms$
= $3.5\,ms + 7.5\,ms = 11.0\,ms$

b) Caching strategy recommendation because it is frequently

accessed remote files become local after first access, reducing remote fetches and average latency

8.a) Proposed mix (10 second period): Full checkpoint: once every 10s (1 full checkpoint per 10s)
Cost - 200ms
Incremental checkpoints: taken every 1s b/w fulls
(at t=1, 2, . . ., 9) → 9 incrementals x 50ms = 450ms

Total overhead in 10s = 200 + 450 = 650 ms
i.e = 0.65 s of CPU time per 10s period).

b) Reasoning: RPO = 1s requires that at most 1 second of new state is lost on failure, so taking incremental checkpoints every 1s satisfies RPO.

9.a) Extremely bursty request patterns causing hotspots. Need for low latency (geo-locality) while balancing global inventory/consistency. Heterogeneous servers, network latency, and autoscaling delays. Suggested algorithm: 'Power of Two Choices' (randomized load balancing).

Edited 20/11/25

b) Active-active multi-region deployment with geo-replication critical state replicated synchronously (or semi-synchronously) across at-least two regions for services requiring low RPO; less-critical data can use asynchronous replication. Use distributed consensus (Raft / Paxos) for metadata & leader election to enable fast failover.

Part - A

1. Race condition : Two people try to write different notes on the same sticky note at the same time. If both read current text, append their own & write back without coordination, one person's update can overwrite other's - result depends on timing

2. Implementation complexity: Peterson's solution is simple algorithmically for but limited to two processes and is tricky to extend. Semaphores provides a general high level abstraction suitable for many process.
Hardware dependency: Peterson's relies on sequential consistency and atomic writes to shared variables - it may fail on modern weak memory multiprocessors without memory fences.

3. Monitors provide structured mutual exclusion plus condition variables built in. In a multi-core system, monitors can use efficient per-monitor locks & condition queues, reducing context switch churn & avoiding busy-waiting.

4. Writers starve if readers continously arrive because now readers can keep entering and block waiting writers forever. We can prevent it by using a fair policy so that once a writer arrives it eventually gets priority and waits readers are blocked until writer completes.

5. Eliminating Hold and Wait typically forces process to acquire all required resources at once (or release held resource before requesting new ones). Practical drawback: poor resource utilization and reduced concurrency process may