Date :

ASSIGNMENT-04

Name : Khushi sharma
Roll no : 2301410007
course : B.Tech CSE (Cyber Security)

(1) Imagine two people trying to withdraw money from the same bank account at the same ATM at the same time. Both see the balance as ₹500. Each withdraw ₹500, & the system allows both, causing inconsistently. The final outcome depends on "who finishes first" not on correct logic → This is a race condition

→ HOW MUTUAL EXCLUSION SOLVES IT :-
A mutex would allow only one user to access/update the account at a time. When one withdrawal is in progress, the other must wait. This prevents conflicting updates & ensures consistent balan

(2) Peterson's Solution's :-
- Pure software-level algorithm; simple but error-prone; limited to 2 processes.
- Requires strict assumptions about atomic read/write ordering → not reliable on modern multi-core CPUs.

29/11/2025 17:29

→ Semaphores :-

- More General :- easier for programmers ; widely supported.
- Depends on hardware atomic operations like test-and-set ; explicity supported by processors.

(3) Monitors automatically combine mutual exclusion with condition variables , so synchronization is handled implicity by the language / runtime.

Advantage :- They Prevent low-level race-conditions automatically , allowing threads on multiple cores to safely access shared data without manually managing semaphores , reducing bugs & complexity.

(4) Starvation :- If the system gives priority to readers , a continuous stream of readers can keep arriving , causing a waiting writer to never get access.

→ Prevention Method :- use writer preference once a writer requests the lock, no new readers are allowed to enter. Existing readers finish, then the writer is served. This ensures no indefinite waiting.

(5) To prevent deadlock, Processes must request all resources at once before starting.

→ Practical Drawback :-

This leads to low resource utilization. A process may hold several resources if it is not currently using, while others wait unnecessarily. This increases waiting time & reduces time system throughput!

(6) a) Combining all edges from three edge fragments Graphically :-

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6$

& Seperately :- $P_3 \rightarrow P_4$

b) Detect deadlock & list processes involved

→ A deadlock exist if there is a cycle in the wait-for-graph.

→ we have the cycle $P_1 \rightarrow P_2 \rightarrow P_5 \rightarrow P_6 \rightarrow P_1$, Process involved in the deadlock → P_1, P_2, P_5, P_6.

c) Suggest one distributed deadlock detection.

→ Chand-Misru-Mass (Probe) algorithm is started choice for distruction deadlock detection.

How it works :-

- A process that suspect it is blocked initates Probe messages of from ::

④ Given :-

- local access time T_L = 5ms
- Remote access time T_R = 25 ms
- Probability of file is remote $= p \rightarrow 0.3$

a) Expected time $E[T]$ is :

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

Plug Numbers :-

$$E[T] = 0.3 \times 25 + 0.7 \times 5 = 7.5 + 3.5$$
$$= 11.0 \text{ ms}$$

b) Caching strategy Suggestion :-

→ Suggested strategy : client site LRU caching with write-back for read-heavy file & validation on TTL.

Justification :-

- LRU (Least Recently used) :- works well in practice because file access pattern often show temporal locality or recently used file are likely to be reused.

(8) a) Proposed optimal mix

Steps :-

→ Take one full checkpoint every 10s.

→ Take incremental checkpoint every 2s b/w full checkpoints.

Total checkpoint overhead per 10s :-

- Fulls : 1 X 200 ms = 20 ms
- incrementals : 9 X 50 ms = 450 ms

Total = 650 ms per 10s

→ average overhead :- 65 ms/s.

b.) Reasoning :-

- RPO constraint : The system must be prepared to restore a state no older than it ; therefore an incremental checkpoint must be taken at least once per second.
- Full checkpoint are expensive I/O ops. Doing them less frequently reduces heavy overhead full checkpoint reduce recovery time bcoz only one full.
- Trade-off :- Frequent incremental add modest overhead but keep RPO tight Periodic fulls keeps recovery efficient.

(9) a) KEY CHALLENGES :-

i) Sudden bursty traffic : Requests spike order of magnitude within seconds.

ii) Geographic latency & data locality :- user should be served from the nearest region when possible.

iii) Hot-spots / skewed load :- certain

Page :
Date :

products / regions get disproportion-ate attention.

iv) Fairness & SLA guarantees :- Mission-critical request (payment) must get prioritized.

b.) Fault-tolerance Strategy :-

1. use multi-region Deployment.
2. Replicate data based on RPO Require-ments.
3. Enable Automated failover to meet RTO.
4. Maintain Cross-Region backups & Regular DR Drills.