NAME: STD.: DIV.:

DATE :
PAGE :

OPERATING SYSTEM ASSIGNMENT - 4

RUCCHIKA KAPOOR

23010I0240

PART-B

Q6 Distributed Deadlock Detection:

given fragments:

→ $S_1 : P_1 \to P_2 , P_3 \to P_4$

→ $S_2 : P_2 \to P_5 , P_5 \to P_6$

→ $S_3 : P_6 \to P_1$

a) Global wait - for graph

combine edges (all):

$P_1 \to P_2$

$P_2 \to P_5$

$P_5 \to P_6$

$P_6 \to P_1$ (cycle)

$P_3 \to P_4$

b) Deadlock detection

The cycle:

$P_1 \to P_2 \to P_5 \to P_6 \to P_1$

Hence, deadlock exists. ~~Proceed~~

Processes in deadlock:

P_1, P_2, P_5, P_6

c) Distributed algorithm to detect it

A suitable algorithm:

Chandy - Misra - Haas Probe - Based Deadlock

Detection ~~sends~~ Algorithm

This algorithm sends PROBE messages across sites to detect cycles in distributed wait for graph.

NAME:__________ STD.:______ DIV.:______ PAGE :

Q7 Distributed file system performance:

given:

Local access = 5 ms

Remote access = 25 ms

Probability = 0.3 → local = 0.7

a) Expected access time:

$E = (0.7 \times 5) + (0.3 \times 25)$

$= 3.5 + 7.5 = 11$ ms

b) Caching Strategy:

Client-side caching with LRU (Least Recently Used)

JUSTIFYING:

Users usually access the same files repeatedly. Caching them locally reduces remote accesses → significantly lowering avg. access time and network load.

Q8 Check Pointing Strategy:

given:

full checkpoint = 200ms

incremental = 50ms

RPO = 1 sec

period = 10 sec

a) Optimal mix:

→ Take 1 full checkpoint at the start.

→ then take incremental checkpoints every 1 second → total 9 incremental checkpoints.

NAME: STD.: DIV.: DATE : PAGE :

Total Overhead :
full : 200ms
incremental : 9X50 = 450 ms
total : 650 ms over 10 seconds

b) Reasoning :
→ RPO = 1s → system must checkpoints at least once every 1 second

→ full checkpoint is expensive; incremental is cheaper
→ taking only incremental checkpoints avoids high overhead while staying without RPO.
→ one full checkpoint ensures a clean baseline for recovery and consistency.

Q9 Case Study: Global E-Commerce Platform
a) Distributed scheduling flash sales:
→ sudden traffic spikes across continents
→ hotspot servers overloaded while others idle.
→ maintaing fairness across regions.
→ handling network latency & replication delays.
→ Ensuring fast task migration b/w servers

Suitable load-balancing algorithm :
Consistent hashing with dynamic load redistribution.

BEST CHOICE:
Work stealing algorithm Idle serves "steal" tasks from overloaded serves → perfect for unpredictable flash-sale workloads

NAME:__________ STD.:______ DIV.:________ DATE : PAGE :

b) Fault tolerance strategy (RTO and RPO focused)

use Geo-replicated active-active clusters with:

→ synchronous replication inside a region (low RPO)

→ ~~sy~~ Asynchronous cross-region replication (low latency)

→ Automatic failover using health checks + DNS-based global load balancing.

ENSURES:

→ low RPO:

Data loss minimised since updated are replicated frequently.

→ low ~~RP~~ RTO:

Traffic can switch to a healthy region within seconds.

EXAMPLES:

→ multi-region distributed file system.

→ checkpoint + log-based recovery in each region.

→ stateless microservices + replicated databases.

Satind 21/11/25