



Process Resilience and Distributed Commit

Process Resilience

- Definition:
- The ability of a system to keep functioning properly even if some processes fail.
- Importance:
- Ensures smooth functioning of distributed systems despite process failures.

Process Resilience: How It Works

- Redundancy: Extra copies of processes handle failures.
- Process Migration: Failed processes move to working servers.
- Heartbeat Mechanism: Regular signals to monitor process health.

Process Resilience: Example & Key Point

- Example:
- In web apps, servers handle requests seamlessly even if one crashes.
- Key Point:
- Ensures system resilience with backups and quick recovery.

Distributed Commit

- Definition:
- Ensures a transaction fully completes on all systems or is canceled.
- Importance:
- Prevents partial transactions and data inconsistencies.

Distributed Commit: How It Works

- Two-Phase Commit Protocol (2PC):
 - 1. Prepare Phase: Coordinator checks if all nodes are ready.
 - 2. Commit/Abort Phase: Based on node responses, transaction is either committed or aborted.

Distributed Commit: Example & Key Point

- Example:
- In online shopping, systems agree to confirm or cancel orders collectively.
- Key Point:
- Ensures consistency by fully completing or canceling transactions.

- Process Resilience:
- Maintains smooth operation despite failures using redundancy and monitoring.
- Distributed Commit:
- Guarantees transaction consistency by ensuring full completion or cancellation.