Distributed Systems Group Assignment

Title: Designing a Fault-Tolerant Distributed Payment processing System

Scenario:

You are required to design and implement a prototype for a simplified Distributed Payment Processing System used by an e-commerce platform. The system should support multiple clients performing payment transactions concurrently, ensuring that all payments are recorded correctly and consistently even in the presence of node failures or network delays. The historical payment details should be accessible from any server to any client.

Tasks:

Divide the tasks among the 4 team members, ensuring collaboration and integration of all components. Each member will focus on one core aspect of the system, but the team must work together to ensure the system functions as a whole.

1. Fault Tolerance (Member 1):

Objective: Ensure the system continues functioning even in case of failures.

Tasks:

- Implement redundancy mechanisms across multiple servers.

- Design a failure detection system to identify when a payment server node becomes unavailable.

- Develop an automatic failover mechanism to redirect payments in case of server failures.

- Propose a recovery mechanism for nodes that rejoin the system.

- Evaluate the impact of redundancy on system performance and storage overhead.

2. Data Replication and Consistency (Member 2):

Objective: Ensure payment details (ledger) are replicated across multiple nodes while maintaining consistency and availability.

Tasks:

- Design a replication strategy (e.g., quorum-based, primary-backup, or sharding).

- Choose a consistency model (e.g., strong consistency, eventual consistency) and justify the trade-offs.

- Implement a deduplication mechanism to handle duplicates caused by retries or failovers.

- Optimize payment performance while ensuring consistency across servers.

- Analyze how replication impacts latency and storage efficiency.

3. Time Synchronization (Member 3):

Objective: Ensure payment details have accurate timestamps for event correlation and debugging across distributed servers.

Tasks:

- Implement a time synchronization protocol (e.g., NTP, PTP) across all logging nodes.

- Analyze the impact of clock skew on log ordering and consistency.

- Develop a mechanism to reorder logs that arrive out of sequence due to network delays.

- Implement log timestamp correction techniques to ensure event accuracy.

- Evaluate the trade-offs between synchronization accuracy and system overhead.

4. Consensus and Agreement Algorithms (Member 4):

Objective: Ensure distributed payment servers agree on log storage, indexing, and retrieval policies.

Tasks:

- Research and implement a consensus algorithm (e.g., Raft, Paxos) for distributed payment consistency.

- Design a leader election mechanism to manage payment coordination.

- Evaluate the performance of the consensus algorithm under high payment processing rates.

- Propose optimizations to reduce consensus overhead while ensuring consistency.

- Test the system under different failure scenarios (network partitions, node crashes).

Deliverables:

1. Report:
   - A detailed report (10-12 pages) explaining the design choices, log replication strategies, and system optimizations.

2. Implementation:
   - A working prototype of the distributed logging system using Python, Java, or Go.

3. Presentation:
   - A 15-minute presentation explaining the architecture, implementation, and evaluation.

4. Code:

 - Github link to the code in a text file.

5. Presentation video:

 - Link to the Youtube video with the presentation.

Evaluation Criteria:

Criteria	Weight	Description
Fault Tolerance	20%	Effectiveness of failure detection, log redundancy, and recovery mechanisms.
Data Replication	20%	Efficiency of log replication, consistency guarantees, and retrieval speed.
Time Synchronization	20%	Accuracy of timestamps and resolution of out-of-order log events.
Consensus & Agreement	20%	Reliability and efficiency of the consensus algorithm for log coordination.

Overall Integration	20%	Seamless operation of all components and scalability of the system.

Submission Guidelines:

- Submit the report, source code, and presentation slides and text files with the links in a single zip file.

- Include a README file with the names, registration numbers, and emails of the team members, along with instructions for running the prototype.