Dor Skoler, Yon Garber

Failure Modes and Mitigation Strategies

1. Security Breaches

Failure Mode: The system's security suite is designed for ease of demonstration, allowing all traffic on default ports.

Mitigation:

- In a production setup, configure the security groups to allow access only to the EndpointServer instances and WorkerNodes.

- Implement proper authentication and access control mechanisms for the Server, such as usernames, passwords, and protected mode.

- Regularly update and patch the system components to address any known security vulnerabilities.

2. High System Load

Failure Mode: The system may experience high traffic and CPU/RAM usage, potentially leading to performance degradation and violating latency service level agreements (SLAs).

Mitigation:

- Implement a load balancer node that evenly distributes requests among the EndpointServer instances, ensuring load balancing and adherence to SLAs.

- Dynamically scale the number of EndpointServer instances based on the system load, using auto-scaling techniques.

- Consider partitioning the workload across multiple Server instances to distribute the load efficiently.

3. App Failures

Failure Mode: Application failures may occur, and there is a need to detect, handle, and recover from these failures.

Mitigation:

- Implement comprehensive monitoring and alerting systems to detect application failures promptly.

- Employ automated error handling mechanisms to gracefully recover from failures and prevent cascading failures.

- Implement proper error logging and error reporting mechanisms for easier debugging and issue resolution.

4. Unreachable Servers / Network Issues

Failure Mode: Servers may become unreachable due to network issues or other connectivity problems.

Mitigation:

- Each system component should have a primary instance responsible for performing health checks on other instances.

- Implement a fault detection and recovery mechanism that can detect unreachable instances and initiate the instantiation of replacement instances.

- Use load balancers or service discovery mechanisms to route traffic only to healthy and available instances.

5. Poor Performance

Failure Mode: System performance is may be suboptimal, affecting the overall user experience.

Mitigation:

- Analyze and tune the system configuration, including resource allocation, caching strategies, and indexing, to improve performance.

6. Machine / Hardware Failure

Failure Mode: Individual machines or hardware components may fail, resulting in service disruptions.

Mitigation:

- Ensure redundancy and fault tolerance in critical system components, such as using multiple instances of RedisServer and EndpointServer.

- Employ strategies such as replication and clustering to distribute the system across multiple availability zones.

- Implement automated failover mechanisms to detect and handle machine or hardware failures promptly.

7. Race Conditions

Failure Mode: Race conditions may occur during concurrent access to shared resources, potentially leading to data inconsistencies or incorrect results (since we use threading this is extra true).

Mitigation:

- Identify critical sections of the code where race conditions can occur, such as accessing shared data structures, threeading or resources.

- Implement proper locking mechanisms, such as mutexes or semaphores, to ensure mutually exclusive access to shared resources.

- Utilize atomic operations or transactional mechanisms provided by the underlying database or storage system to maintain data integrity.

By considering these failure modes and implementing the suggested mitigation strategies, the system can become more robust, secure, and resilient in real-world scenarios. Regular testing, monitoring, and continuous improvement are essential to address any evolving failure modes and ensure the system's reliability and availability.