Groub members:

Rawad abdalla: 315281964

Mohamad Abed Aljwad: 206954950

Maher Bathish : 209417112

Date : 30/12/2024

Queue Server Design Document

1. Architecture Overview

The RabbitMQ Queue Server is designed as a clustered environment with multiple nodes to provide high availability and durability. The architecture includes the following components:

1. **Producer Services:**
    - **ParkingService**: Publishes parking transaction events.
    - **PEOService**: Publishes parking citation events.
2. **Queues:**
    - **transactionsQueue**: Handles parking transaction events.
    - **citationsQueue**: Handles parking citation events.
3. **Consumer Services:**
    - **MunicipalityOfficerService (MOService):** Consumes messages from transactionsQueue and citationsQueue for reporting purposes.
4. **RabbitMQ Cluster Configuration:**
    - Three-node cluster with quorum queues for resiliency.
    - Quorum queues ensure message replication and durability.

2. Queues and Message Flow

2.1 Queues

- transactionsQueue:
    - Purpose: Stores parking transaction events (e.g., start and stop parking).
    - Message Structure:
        - VehicleID: Unique identifier of the vehicle.
        - ZoneName: Name of the parking zone.
        - SpaceID: ID of the parking space.
        - StartTime: Parking start time.
        - EndTime: Parking end time.
        - TotalCost: Total parking cost.
- citationsQueue:
    - Purpose: Stores parking citation events issued by Parking Enforcement Officers (PEOs).
    - Message Structure:
        - VehicleID: Unique identifier of the vehicle.
        - SpaceID: ID of the parking space.
        - ParkingZone: Name of the parking zone.
        - InspectionTime: Time of the citation.
        - TotalCost: Cost of the citation.

2.2 Message Flow

1. Producers:

- ParkingService publishes parking transaction events to transactionsQueue.
- PEOService publishes parking citation events to citationsQueue.

2. Consumers:
   - MOService subscribes to both transactionsQueue and citationsQueue to process and generate reports.

3. RabbitMQ Cluster Configuration

3.1 Cluster Overview

- Nodes: Three RabbitMQ nodes forming a cluster.
- Queues:
  - transactionsQueue and citationsQueue configured as quorum queues.
  - Replication factor: 3 (each message is stored on all nodes).
- Durability: All queues are durable to ensure message persistence.

3.2 RabbitMQ Settings

- High Availability: Quorum queues ensure resilience against node failures.
- Connection Failover: Producer and consumer clients will use failover mechanisms to connect to available nodes.
- Acknowledgments: Consumers will acknowledge messages after successful processing to ensure reliable delivery.

4. Routing Decisions

Cluster Resilience and Failover

- Connection Failover:
  - Producers and consumers are configured to connect to any available RabbitMQ node in the cluster.
  - Routing logic remains unaffected by individual node failures.
- Quorum Queues:
  - Messages are replicated across all nodes in the cluster, ensuring delivery reliability even in the event of a node failure.

5. Interaction with Clients and Other Servers

The queue server facilitates communication between the backend services and the user interfaces (UIs). Here's how it interacts with each client:

1. Customer UI:
    - Start Parking: No interaction with the queue server. This is handled directly by the database via ParkingService.
    - Stop Parking:
        - The Customer stops parking through the UI.
        - ParkingService calculates the total cost and generates a transaction event.
        - The event is published to transactionsQueue.
2. PEO UI:
    - Check Parking: No interaction with the queue server. Parking validation is handled directly by the database via PEOService.
    - Issue Citation:
        - The PEO identifies an illegally parked vehicle.
        - Citation details are sent to PEOService, which generates a citation event.
        - The event is published to citationsQueue.
3. MO UI:

- Generate Reports:
    - The Municipality Officer requests transaction or citation reports through the UI.
    - MunicipalityOfficerService consumes messages from the respective queues, processes them into objects, and displays the report.

Nodes

Name	File descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Cores	Info	Reset stats	+/-
rabbit@DESKTOP-1I6V4EL	0 65536 available	426 1048576 available	81 MiB 19 GiB high watermark	24 GiB 48 MiB low watermark	1d 1h	8	basic 1 rss	This node All nodes	
rabbit@Mohamad98	0 65536 available	469 1048576 available	94 MiB 9.4 GiB high watermark	134 GiB 48 MiB low watermark	7h 41m	8	basic 2 rss	This node All nodes	
rabbit@stu11	0 65536 available	324 1048576 available	86 MiB 9.4 GiB high watermark	193 GiB 48 MiB low watermark	22h 53m	8	basic 0 rss	This node All nodes	

Overview						Messages			Message rates		+/-
Virtual host	**Name**	**Node**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
/	**citationsQueue**	rabbit@DESKTOP-1I6V4EL +2	quorum	D Args	running	0	0	0			
/	**transactionsQueue**	rabbit@DESKTOP-1I6V4EL +2	quorum	D Args	running	10	0	10		0.00/s	0.00/s

Details

Features	arguments: x-queue-type: quorum durable: true
Policy	
Operator policy	
Effective policy definition	
Leader	rabbit@DESKTOP-1I6V4EL
Online	rabbit@DESKTOP-1I6V4EL rabbit@Mohamad98 rabbit@stu11
Members	rabbit@DESKTOP-1I6V4EL rabbit@Mohamad98 rabbit@stu11