Distributed Algorithms Course

Exercise 2 – CassandraA –

Due date 8 July 2025, may submit in pairs

Dr. Miriam Allalouf

The global wildlife research organization **TrackBirds** is tracking the movement of thousands of birds across different continents. Each bird is tagged with a GPS transmitter that sends location data every minute. The data includes the following: bird_id, species, timestamp, latitude, and longitude.

This data is ingested into a **Cassandra database cluster** with **four** nodes and a **Replication Factor (RF) = 3**, ensuring high availability and fault tolerance.

Researchers query the system regularly to obtain the **most recent location** of a bird and **the series of locations over a period.** The system tracks **tens of thousands of birds globally**, and your task is to design a **scalable, efficient Cassandra schema** and simulate operations under various scenarios.

Question 1 - Answer the following question in this file

Designing a Cassandra table scheme for **a query that obtains a series of locations over a period of time** for a bird. Define the requested period of time.

1.1. Define the table scheme in CQL

1.2. Explain your partitioning and clustering key choices:

- What are the partition key and clustering key?
- How does this design support data distribution and avoid hot spots?

1.3. Describe the flow of **insert/update** operations (when CL = ONE)

1.4. Describe the flow of

1.5. Describe the flow of the query to retrieve the **latest location** of a specific bird (i.e, **select** operation with CL = ONE

2. Question 2 – Write the Client Code and Trace the Workflow, and answer the questions

2.1. Write two client drivers using Cassandra Python. You may define two clients using simple execution commands, or one client with queues and threads, each thread running a different set of workload (birds' and tracker's commands)

- Bird Client:
    - Creates the birds tracking table (if not exist)
    - Runs "Insert" operations for 10 different birds and their initial location.
    - Runs 20 "Update" commands for each bird with a new timestamp and location. Each "update" command is sent every minute.
    - Set a timer between any two consecutive operations

- Tracker Client:
  - Queries to obtain the list of locations of a certain bird to derive the last location
  - The queries are sent periodically for each bird
  - Writes retrieved data to a log file

2.2. Cassandra allows execution trace by adding the “trace” flag to the execution command and by retrieving the results using the “get_query_trace” (https://docs.datastax.com/en/drivers/python/3.2/api/cassandra/cluster.html).
The trace output in Cassandra contains detailed low-level information about the steps a query takes inside the database engine. This is especially useful for diagnosing performance issues or understanding query execution in distributed environments.
Add the trace command to the clients and parse the results for the update and select operations of one bird. Provide the flow os the operation in terms of the coordinator and replicas' timestamps.

3. **Question 3 – Node Failure Simulation – run the clients and write the results in this file**

3.1. Use the nodetool ring command to inspect token distribution.

3.2. Choose a specific bird, find its token (using the token function), and locate its replica.

3.3. Simulate failure by stopping a node that holds the token of this specified bird.

3.4. Re-running “nodetool ring”.

3.5. Re-run the update and select operations and using the trace results, describe the current flow.

4. **Question 4 – System Registration Table - Answer the following question in this file**

There are two roles within the system, each requiring a unique login and initialization process:

- Admin
  - Sets up the Cassandra cluster and defines keyspaces and tables.
  - Creates metadata tables for birds and researchers.
- Bird Account
  - Periodically sends location updates (every minute) to the database.
- Tracker
  - Periodically queries the database for the list of locations of specific birds.

4.1. Design a Cassandra table to handle **registration and authentication** of all system roles (admin, Bird, Tracker).

4.2. What consistency levels should be used for authentication operations (read/write) with RF = 3?

4.3. What happens if one node fails during these operations?

5. **Question 5 – Can a Key-Value Store Be Used for the query asked in question 1?**

5.1. Describe how you would **model the data** in a key-value database.

5.2. What are the **limitations** compared to Cassandra, especially regarding historical data and scalability?

5.3. For which query might a key-value database be a better fit than Cassandra?

Please submit the following file:

1) Please submit this file with answers. At the top, write the names of the students + TZ.
2) Create a code readme with students' names, environment description, running command, and code files.