Overview

This document is going to cover the event booking system architecture like book my show.

Problem Requirements

- Many available events for which tickets are up for purchase.
- Search should be allowed in (in context of book of my show)
  city -> movie-> theatre->time->seats
- In case of the available seats, users can claim it for x minutes. This claim can either expires or user can purchase the ticket.

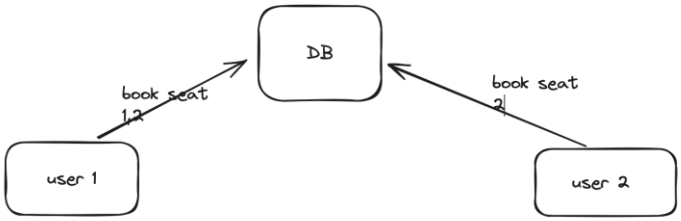
Capacity Estimate

- There are 500 cities, and each city has 20 theatres.
- There are 2000 seats per show and each theatre can have 10 shows per day.
- Considering we need to store 50 bytes of information in the database (
  seat booking information, theatre metadata, city metadata,
  movie metadata and user metadata).
- Total storage required is 500*20*2000*10*50 bytes = 5 GB per day.

API

- GET getCity() To get the city that app is active in.
- GET getMovies(String cityId)
- GET getTheatres(String cityId, String movieId)
- GET showTimes(String movieId, String theatreId)
- POST bookSeats
    - user_id
    - List of seat_id
    - show_id

Database Choice



- We want the booking to be both atomic and serlizable
    - Atomic -> If user is booking seat 1 and 2, we either want to book both 1 and 2 or none.
    - Serlizable -> Is user is booking 1, 2 and user 2 is booking 2 simultaneously, chances are seat 1 belongs to user 1 and 2 belongs to user 2 (split booking for user 1). We don't want that.
- We will need transactional database.

Database Schema

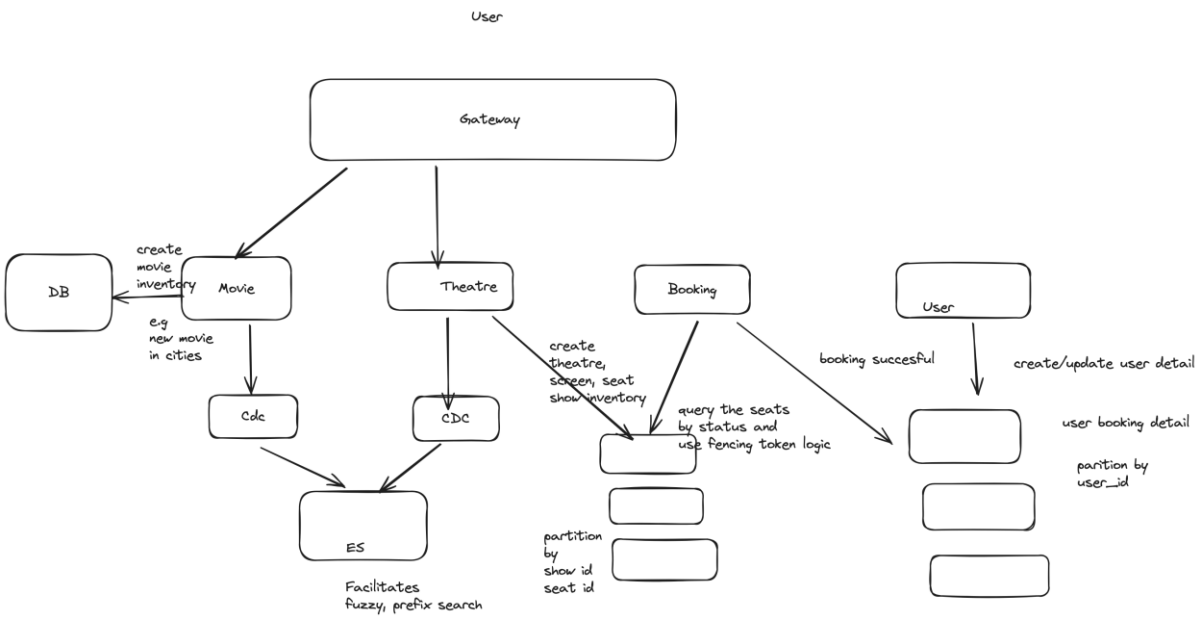
- Cities {Id,Name}
- Movies {Id, Name, Release date, Genre, Running time, Language}
- MoviesToCities {Movie Id, City Id}
- Theatre {Id, Name, Address, Theatre Type, Contact Number}
- ~~MoviesToTheatre {Movie Id, Theatre Id}~~
- screen {Id, Type (IMAX or HD), name, size, theatre_id}
- show {Id, screen_id, start time, end time,movie_id)
- seat (id, show_id,row_id,number,seat_type)
- show_seat (seat_id, status, price)
- booking (id, user_id, seat_id,booking_time,payment_id)
- user {id, email, contact number, name}

Database Partitioning

- In case we want to increase our write throughput in cases where certain events are seeing very high traffic (booking for Taylor swift concert).
- We might need to partition the data.
- However distributed transactions are super slow.

- Usually, user books the seat in the same row. What if we can somehow guarantee that, a particular row will always be on single partition.
- Partition key -> ~~event_id(or movie_id) +~~ show_id + row_id (hash).

Architecture



- Movie service
    - Exposes CRUD APIs to create and update the inventory of the movies. This includes movies in a city etc.
    - Also, they will produce the events async to CDC pipeline to update in ES. This is to facilitate the fuzzy or prefix-based movies searches.

- Theatre Service
    - Exposes CRUD APIs to create and update the inventory of the theatres.
    - This will include creation of theatre, screen, show, seat and show_seat table.
    - Also, they will produce the events async to CDC pipeline to update in ES. This is to facilitate the fuzzy or prefix-based theatre names.

- Booking service
    - Expose the APIs to claim your ticket or book the ticket.
    - How does the flow will look like ?
        - User 1 is requesting to book the seats 1,2,3 for a particular show.

- Client will call the API with the user id, show id and seat id to make the claim.
- In case all the seats are available.
- Each user has some claim expiration of x minutes.
- How does a particular row in order claim database will looks like in this case seat id, user_id , claim expiration, row,id show id, status.
- Here the claim expiration will be populated by the database where the query will look like now-xmin.
- We will use combination of user, now-xmin as fencing token and return to client to send it in subsequent request.
- User is going to do the payment. Client will send the fencing token. Server will validate if user is eligible for booking the seats using user_id and now-xmin. If the fencing token matches with the database, we will accept the payment, change the status of the seats to occupied and create the entry in the booking database.
- How will we clear if the user claim expires, and he has not made the payment
  Some offline batch job can do it for us.
- What will happen for the 2 concurrent users trying to book the seat?
  Because the transaction level is serlizable, only one of them will be successful.
- What are the implications of using serlizable isolation ?
  serlizable isolation doesn't allow the read on the rows for which the write is going on. This will cause the issue as either write must wait in case this is read heavy system or read must wait (to get the number of empty seats).
- **What are the other solutions for this problem**
  Instead of fencing token what we do ?
  Create the keys in redis for each seat that user is trying to book. TTL can be x min which is your claim expiration. This creation of the keys can be done by LUA script to take care of concurrent user trying to book same seat.
  In case the key already exists, the booking of one the user will fail.
  In case LUA script execute successfully, update the show_seat status update to BLOCKED.
- Also, we can listen to keyspace notification once the keys expire. In case user has not made the payment, we will change the status of the seat to available.