Reporting Architectural Design Challenge

Last update: 2022-10-12

Scenario

OpenTable generates reservations and guests data from various touch points within our ecosystem, via Kafka topics. The data includes reservations details (eg: reservation dates, party size, tags, status etc), information about the guests (eg: name, address, contact details etc) and POS data (eg: spent amount, tax, currency etc).

Restaurant owners want to view their reservations and guests information so they are able to analyze their business performance and trends. We also have internal and external partners who need to ingest this information into their systems.

Raw Data Samples

Raw data is available for us to consume via Kafka topics. The retention period for these messages is 7 days. The following are samples of raw data provided by other engineering teams around the world for us to consume.

Reservations data (cropped sample)

```
{
    "ReservationID": 888999,
    "ReservationState": 2,
    "ReservationDate": "2022-09-30",
    "PartySize": 2,
    "Status": "SEATED",
    "RestaurantID": 12345,
    "GuestID": 1122334455,
    "TransactionID": "charge111",
    "VisitNotes": "Likes outdoor seating",
    "Tags": ["smoking_area", "friend_owner"],
}
```

Guests data (cropped sample)

```
{
  "RestaurantID": 12345,
  "GuestID": 1122334455,
  "FirstName": "John",
  "LastName": "Smith",
  "Address": {
    "AddressLine1": null,
    "AddressLine2": null,
    "City": null,
    "State": null,
    "ZipCode": null,
    "Country": null
  },
  "PhoneNumbers": [
    {
      "Label": "Mobile",
      "Number": "+61412345678",
      "CountryCode": null,
    }
  ],
  "Email": "john.smith@web.mail",
  "Notes": "Likes red wine",
  "Tags": ["vip"],
}
```

POS transactions (cropped sample)

```
{
  "RestaurantID": 12345,
  "PosID": "charge111",
  "Currency": "usd",
  "Denominator": 100,
  "TotalAmount": 2930,
  "Tip": 500,
  "Fees": 130,
  "Amount": 2200,
  "Tax": 100
}
```

API Layer

With the help of the product manager, the Melbourne team has envisaged this API will be the final result for the restaurant owners (or stakeholders) to utilize.

The API will fetch a list of reservations with guest information based on the range of the reservation date and a restaurant ID. Each reservation will also provide an aggregation of the total amount paid and total visits by the guest to this restaurant (e.g the guest is a returning customer).

API /api/v1/report/

Parameters accepted

- RestaurantID
- Reservation Date - startDate (yyyy-mm-dd)
- Reservation Date - endDate (yyyy-mm-dd)
- Tags (reservation or guest) - “vip,friend_owner”
- Status - SEATED, CANCELLED, BOOKED, DONE
- Limit

Example

```
?restaurantID=12345&startDate=2022-09-02&endDate=2022-09-03&tags=vip,friend_owne
r&limit=100
```

API Payload (Response) Sample

```
{
    "ReservationID": 888999,
    "ReservationState": 2,
    "ReservationDate": "2022-09-30",
    "PartySize": 2,
    "Status": "SEATED",
    "VisitNotes": "Likes outdoor seating",
    "ReservationTags": ["smoking_area", "friend_owner"],
    "FirstName": "John",
    "LastName": "Smith",
    "Address": {
      "AddressLine1": null,
      "AddressLine2": null,
      "City": null,
      "State": null,
      "ZipCode": null,
      "Country": null
    },
    "PhoneNumbers": [
    {
      "Label": "Mobile",
      "Number": "+61412345678",
      "CountryCode": null,
    }
    ],
    "Email": "john.smith@web.mail",
    "GuestNotes": "Likes red wine",
    "GuestTags": ["vip"],
    "Currency": "usd",
    "Denominator": 100,
    "TotalAmount": 2930,
    "Tip": 500,
    "Fees": 130,
    "Amount": 2200,
    "Tax": 100,
    "TotalVisits": 5,
    "TotalSpend": 153900
}
```

What we need from you

Your job, as a Principal Engineer, is to design the architecture of the above system. The architecture diagram should showcase the flow of data, storage and how we will expose this information via an API.

What we need from you as an outcome (preferably in PDF format) are:

- Clear solution diagram of the architecture using a tool of your choice.
- Clear outline of tech choices used and where.
- Clear outline of tradeoffs and assumptions.

We're keen to understand what your thought process is, how you approach this challenge and how you make your decisions on what to compromise etc.

We would like to hear your proposal on these aspects:

- How will you ingest the data.
- How will you architecture the data storage and accessing the data.
- How will you architect the application so it addresses a number of concerns - high availability, scalability, performance, real-time data exposure and addressing any single point of failures.
- How do you deal with data availability e.g. data retention in Kafka is only for 7 days.
- How do you work with the front end engineers to agree on a contract to serve the data for the front end apps?

Architectural design criteria and guidelines

A few things to consider when thinking of your recommended approach/solution:

- The API layer is able to serve a large volume of traffic and is scalable, especially with spikes.
- Data will be stored in retention for 10 years or more.
- Data will be accessible regardless the age of data.
- Raw data will be ingested from a number of sources in real time.
- Messages are consumed at a rate of 200 messages/sec for reservations and approx. 100 messages/sec for guests. We estimate 150 hits/sec to our API endpoints.
- We have over 2.5 billion records currently in our database (restaurants, reservations, guests and POS data).
- The raw data can grow by approximately 1.5TB to 2TB per year.
- We have over 60,000 restaurants globally, which means restaurant owners are hitting the API endpoint 24/7.
- We have a need to aggregate/analyze the data for some reports.
- Some reports need the raw, "real-time" data.
- We have both internal and external partners who access the APIs. How would you design this so it mitigates the possibility of DDoS?
- The solution for collecting, aggregating, and ingesting the data should be done with low latency (real time), high accuracy and high throughput. We have to guarantee data accuracy.
- API responses should be within an acceptable time. We are keen to hear your thoughts on what's considered acceptable under what conditions.
- We are restricted to a number of technology stacks - Kafka, Java (SpringBoot). We are keen to get your thoughts and recommendations on what you will use for the different layers - data store, cache etc and any other tech stack you would recommend.
- Clearly define tradeoffs between the desired and realistic outcomes.