PaytmLabs SDE Challenge System design

Pranay Patadiya

Design Question

Design A Google Analytic like Backend System. We need to provide Google Analytic like services to our customers. Please provide a high level solution design for the backend system. Feel free to choose any open source tools as you want.

Requirements

1. Handle large write volume: Billions of write events per day.
2. Handle large read/query volume: Millions of merchants wish to gain insight into their business. Read/Query patterns are time-series related metrics.
3. Provide metrics to customers with at most one hour delay.
4. Run with minimum downtime.
5. Have the ability to reprocess historical data in case of bugs in the processing logic.

Introduction

This document provides high level system design for near real time data analytics systems(like GA). This document has been made for a high level system design perspective and does not include any low level design.

Overview

Analytical system, defined below, using Kafka heavily as this system is designed 100% event based and with open source tools and technology. It supports millions of read, process and write transactions.

In GA like analytical tools, all the data from the browser has been sent via XHR (ajax) request asynchronously and from mobile it is been sent to process via asynchronous mechanism via rest API periodically.

When ever any data has been generated to the system from data source (browser, mobile, API or any) having a **TrackingID** been generated from server side and given to the front end so in each request this trackingId is must so it will be tagged with merchant data and also used in multi tenancy.

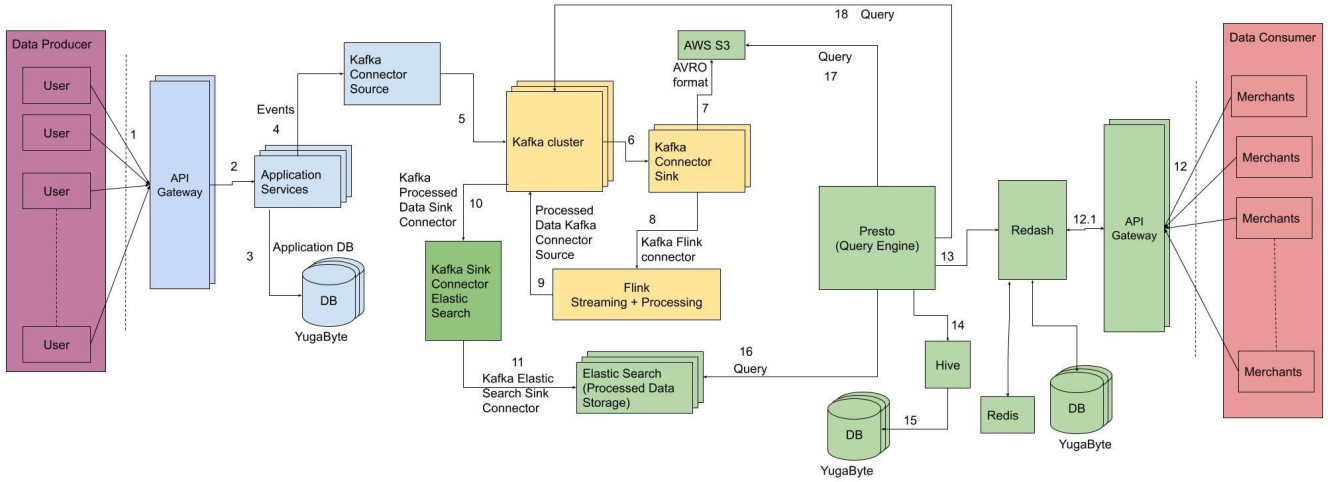
Points to be considered:

- High availability
- Faulty Tolerance
- Cloud agnostic
- Multi cloud load balancing and high availability
- Open source tools and technology
- Using as much distributed component in nature
- Analytical Dashboard real time
- Historical data storage
- Data Lake, Data classification, Data partitioning, Multitenancy
- Generic data formats
- Plug N Play components
- Future scope

System Design

This section provides the idea how the whole system will work at a high level.

Diagram



This design has designed keeping in mind 3 major portion of the system as below:

1. Data ingestion
2. Data Processing and Storage
3. Data Query and Visualization

Data Ingestion

Data ingestion is one of the pillars of the analytical system. Data has to be seamlessly ingested in the system and stored asynchronously. There are multiple data source

Data Source

There will be different kinds of data sources in the system. For this design we have consider below type of data sources:

1. Web Browser
2. Mobile Application
3. APIs

All above sources emit the data in the system and then the streaming and processing part of the system will take further care of data.

Tools

#	Module	Technology	Description
1	API Gateway	Nginx	Used Nginx as an API gateway. As Nginx works as API gateway, Load balancer and reverse proxy too.
2	Application Services	Spring Boot	Application services are built in Spring Boot.
3	Distributed database	Yugabyte	Yugabyte wil be used as a distributed database for application services for ingestion of data.
4	Kafka connector source	Kafka-connector-source	Kafka connector source is used to stream the data for further process. There will be multiple consumers for this data.
5	Kafka cluster	Kafka	Kafka cluster is the main broker of events streaming in the system. It will forward data for further processing.

Data Processing

Data processing is the next phase of data in the stream received from data ingestion. Here we are sending data at two places for processing and storing.

1. AWS S3 (AVRO data format) for historical purpose

   When Avro data is stored in a file, its schema is stored with it, so that files may be processed later by any program. If the program reading the data expects a different schema this can be easily resolved, since both schemas are present..

2. Apache flink for processing real time data

   Apache flink is an open source distributed data processing engine for data streams for bounded and unbounded data streams. We have a bounded stream here, as we are getting all data before processing it and do not concern order of the data.
   Data classification also will be taken care when processing data, so processed data will be stored in different schema to achieve multi-tenancy.

Once the data is processed, it has been sent further for storage as structured data to storage for further query part of analytics.

Here, data storage, **Elastic search** is being used. Elastic search is enabled with search engine and supports a wide range of queries in a distributed manner.

So, Flink sent the processed data to Kafka broker. From kafka broker, with kafka-elastic-sink connector, stored in elastic search.

Data Analytics & Querying

Data analytics and query is happening from Elastic search.
As an analytical dashboard we have used Redash - open source analytical dashboard.

From Redash, Presto is used as a query engine. Presto has a metadata store in Hive backed with yugabyte as distributed storage.

Presto has been configured with metadata. Beauty of having Presto/Trino is it can query any data source where there is no need to change in Redash.

So in Redash we can see data from:

- Elastic search,
- AWS S3
- Kafka
- Redis
- Or any other data source

Redash has different logins for different merchants and all data is maintained multi-tenant wise. Redis will be used as caching mechanism.

With the above setup, merchants can see live processed data as well can query historical data if in requirement.

All the merchant requests been through the API gateway.

Deployments

This analytical system deployed in HA in nature.

- All the services are dockerised/containerised
- All component are orchestrated with Kubernetese
- RDBMS Database we used as Yugabyte which is itself distributed in nature
- Load balancing, Reverse Proxy functionality achieved by Nginx.

High availability will be used across multi cloud orchestration. We will use GSLB (Global load balancing) domain level and do the load balance across different clouds.