Analytics System

By Daksh Kapur

High-Level design for Google Analytics like Backend System.

Requirements

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

System Attributes

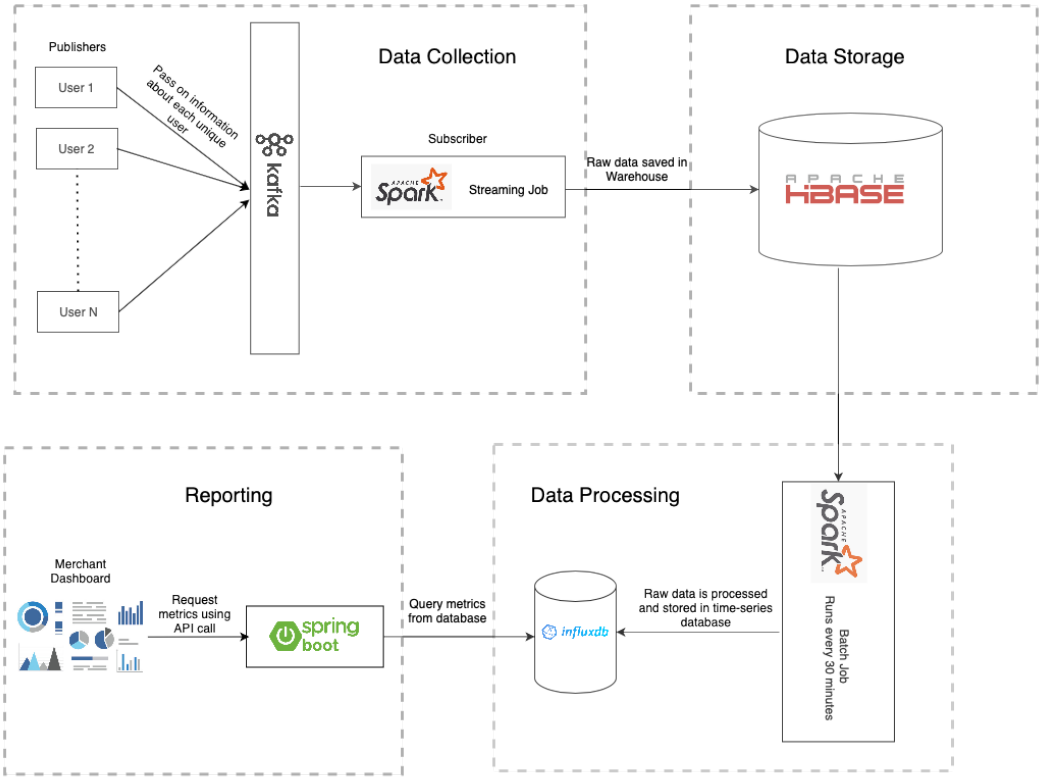
The system should be able to track, process and report time-series related metrics. Some of the metrics that the system will be tracking are as follows:

- Total number of users
- Active users
- Pageviews
- Average time on page
- Number of sessions
- Session duration
- Pages per session
- Bounce rate
- Incoming traffic sources

System Design

Upon analysis of the requirements it becomes evident that Big Data Pipeline based Architecture will be best suited for this problem. Most of the components used in the system are open-source.The system will consist of 4 major components:

1. Data Collection
2. Data Storage
3. Data Processing
4. Reporting



1. Data Collection

It is proposed to collect data based on streaming architecture because of heavy volume of incoming requests. Here's an estimation of number of requests.

```
Number of requests per day: ~1 Billion
Number of requests per hour: ~41.5 Million
Number of requests per second: ~11.5K
```

Collection of such a large amount of data can be done effectively using Streams.

It is proposed that each user is assigned a universally unique identifier (UUID).

Each website will be using browser side JS and SDK for mobile applications. Which would communicate with Kafka REST proxy for data ingestion using an HTTP request. The raw data would be transmitted to Kafka. Kafka is chosen because it supports high-throughput and low latency. It is also Fault-tolerant and easily scalable in nature.

The unique ID would be stored in cookies for each user to track sessions. If the user has disabled cookies then the user would be considered as a new user every-time the website is accessed by the user.

A Spark streaming job would be used to store raw data into Hbase.

Alternatives: Google cloud Pub/Sub, Apache Beam

2. Data Storage

All of the raw data is stored into Hbase, which is a Hadoop based non-relational database. Hbase is horizontally scalable and can handle large amounts of data. Below are some of the reasons why Hbase was selected:

- Supports random and consistent Reads/Writes in large volume
- Fault-tolerant
- Open source
- Low latency access to data
- Horizontally scalable

Alternatives: Amazon Redshift & Google BigQuery

3. Data Processing

A Spark batch process runs every 30 mins which extracts and processes the data from Hbase. Raw data is cleaned and stored in InfluxDB as time-series data. InfluxDB will be a subordinate storage for faster read queries to the reporting system.

Data storage has a lot of raw data and if reporting talks to Hbase directly it will cause high latency to access the user data and reporting will not be performant enough.

In case there is a bug within our business logic Spark batch process can be altered to reprocess the raw data in Hbase using the new business logic.

Alternatives: Apache Beam**,** KairosDB

4. Reporting

A Spring Boot based REST API would be used to access InfluxDB and support a front-end dashboard for our Merchants.