Build a Big Data Analytics Pipeline

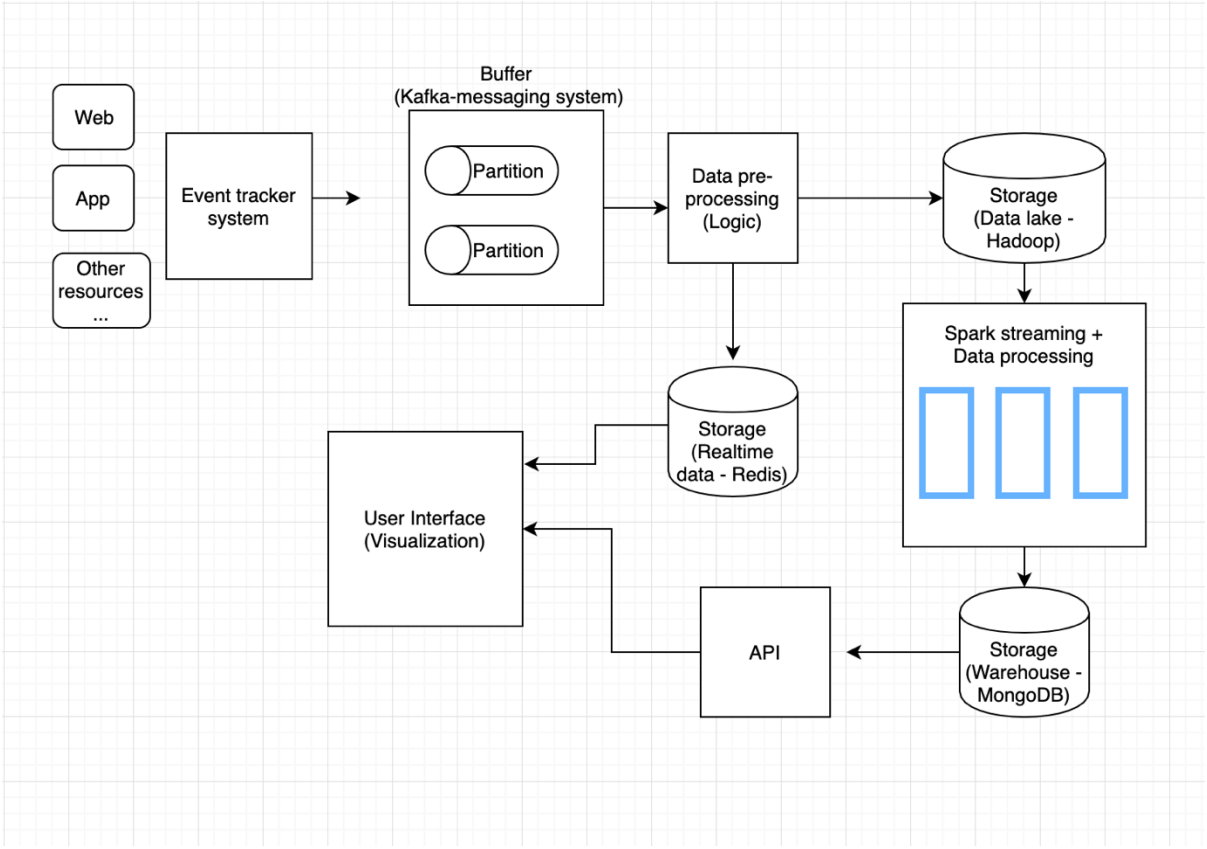
Context

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

Critical Parameters of the System

- **Handle large write volume:** Billions write events per day
- **Handle large read/query volume:** Millions merchants want to get insight about their business. Read/Query patterns are time-series related metrics.
- **Show Low Latency:** Provide realtime metrics and provide metrics to customers with at most one hour delay.
- **SLA:** run with minimum downtime.
- **Be economical:** have the ability to reprocess historical data in case of bugs in the processing logic.
- **Be scalable**
- **Be flexible**

System Designing

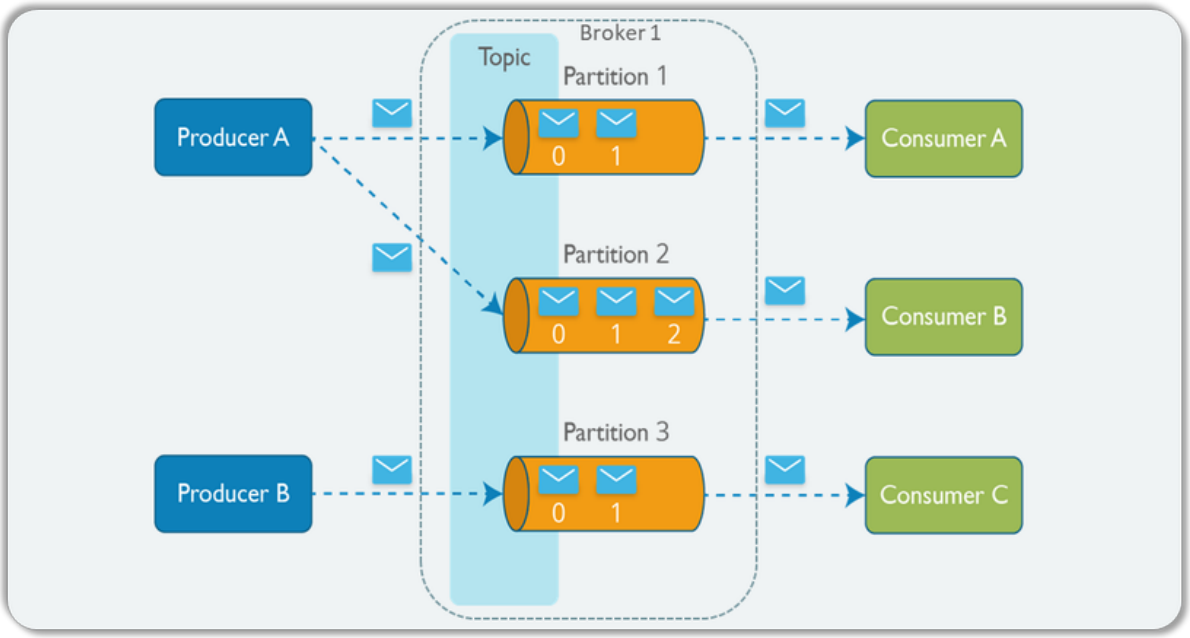


System in detailed:

1. **Event tracking system**

- On all resources (web application, mobile application), implement logic to track all events like click, open, page view…
- Backend logic will sent all these events with params to the buffer.

2. **Buffer – The messaging system (Kafka)**



Why kafka?

- Kafka can handle large volumes of data & is a highly reliable system, fault tolerant, scalable.
- Kafka is a distributed publish-subscribe messaging system(The publish-subscribe messaging system in Kafka is called brokers).
- Kafka can handle high-velocity real-time data.
- Kafka is a highly durable system as the data is persistent and cannot be replicated
- Kafka can handle messages with very low latency

3. **Data pre-processing**

Logic implementation is required. Depend on the business, the logic will receive data from Kafka and handle.

The logic will separate data to:

- Realtime data:
  - active viewers, users looking at an item
  - most popular content such as the top 10 active pages
- Processing data:
  - Conversion rate data, Bounce, Page Visiting time
  - And data for other metric calculations, depends on business

All realtime data will be pushed to Redis, and frontend API will fetch and display it on UI for end users. All processed data will be stored to storage: Hadoop

4. The First Storage (Data lake - Hadoop)

Using Hadoop to store data lake, and it's able to reprocessing historical data
Why Hadoop?

- **Cost-effective solution:** Hadoop provides a cost effective storage solution for business.
- **Scalability:** It is a highly scalable storage platform.
- Unique storage method of Hadoop is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing.
- **Hadoop is fault tolerance.** When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.
- Hadoop is more than just a faster, cheaper database and analytics tool. It is designed as a scale-out architecture that can affordably store all of a company's data for later use.

5. Spark streaming + Data processing

This is the stage where the processing of data is actually done. Logic implementation is required.
Why Spark?

- **In-Memory Computation in Spark**
  With in-memory processing, we can increase the processing speed. Here the data is being cached so we need not fetch data from the disk every time thus the time is saved. Spark has DAG execution engine which facilitates in-memory computation and acyclic data flow resulting in high speed.
- **Swift Processing**
  Using Apache Spark, we achieve a high data processing speed of about 100x faster in memory and 10x faster on the disk. This is made possible by reducing the number of read-write to disk.
- **Dynamic in Nature**
  We can easily develop a parallel application, as Spark provides 80 high-level operators.
- **Fault Tolerance in Spark**
  Apache Spark provides fault tolerance through Spark abstraction-RDD. Spark RDDs are designed to handle the failure of any worker node in the cluster. Thus, it ensures that the loss of data is reduced to zero.
- **Real-Time Stream Processing**
  Spark has a provision for real-time stream processing. Earlier the problem with Hadoop MapReduce was that it can handle and process

data which is already present, but not the real-time data. but with Spark Streaming we can solve this problem.

6. The Second Storage (Data warehouse - MongoDB)

After processing data with spark streaming, our business data will be sotred to MongoDB.

Why MongoDB?

- Large volumes of structured, semi-structured, and unstructured data
- Agile sprints, quick iteration, and frequent code pushes
- Object-oriented programming that is easy to use and flexible
- Efficient, scale-out architecture instead of expensive, monolithic architecture

7. API and Data visualization

API to fetch data from MongoDB and display for a user interface