Design A Google Analytic like Backend System

There are numerous way of designing a backend. We will take Microservices route because the web scalability is required for Google Analytics (GA) like backend. Micro services enable us to elastically scale horizontally in response to incoming network traffic into the system. And a distributed stream processing pipeline scales in proportion to the load.

Here is the High Level architecture of the Google Analytics (GA) like Backend System.



# Components Breakdown

# Analytics events data source:

Every web page or mobile site tracked by GA embed tracking code that collects data about the visitor. It loads an async script that assigns a tracking cookie to the user if it is not set. It also sends an XHR request for every user interaction.

# HAProxy Load Balancer

to improve the performance and reliability of a server environment by distributing the workload across multiple servers HAProxy performs load balancing (layer 4 + proxy).

# Istio Service Mesh and Kubernetes microservices cluster

Istio makes it easy to create a network of deployed services with load balancing, service-to-service authentication, monitoring, and more, with few or no code changes in service code. You add Istio support to services by deploying a special sidecar proxy throughout your environment that intercepts all network communication between microservices, then configure and manage Istio using its control plane functionality, which includes:

* Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
* Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
* A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.
* Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
* Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Istio is designed for extensibility and meets diverse deployment needs

We can run polyglot microservices, with such a setup to power UI and business logic implementations.

# Apache Kafka Streams

Apache Kafka is used for building real-time streaming data pipelines.

The ingested data is read directly from Kafka by Apache Spark for stream processing and creates Timeseries RDD (Resilient Distributed Datasets).

# Apache Spark + Ignite Processing

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.

It provides a high-level abstraction called a discretized stream, or DStream, which represents a continuous stream of data.

Apache Spark is a perfect choice in our case. This is because Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

Apache Ignite is a distributed memory-centric database and caching platform that is used by to share RDD between spark jobs and later persistence.

This will power any high computations to power collective data set creation.

InfluxDB

InfluxDB, is a time series database, to support efficient data ingestion and expensive time series queries. This will store the processed data either from Apache Spark processing or from microservices(primarily spark processing).   
Later, microservices can consume data directly from influx, with inbuild aggregation support.

Redshift

Redshift, being an AWS managed data warehouse, can be used to store historical datasets for later retrieval of data and processing.   
It also supports pre-planned queries across millions of records within milliseconds, so Redshift can be effectively used for supporting basic crytal reports.