## 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.

Google Analytic system can be designed based on Microservices architecture. The reason behind using microservices is flexibility in using technologies and scalability in response to incoming traffic to the system.

Main components that can be used while designing system are:

1. Firewalls and Web/Mobile Tracking Code System
2. Load Balancer like NGINX, HAProxy, and F5 etc.
3. Web Servers like Apache HTTP server, Nginx Web Server etc.
4. API Gateway and service discovery
5. SpringBoot microservices
6. Apache Kafka for distributing message stream
7. Apache Spark for parallel processing
8. Cassandra

Below is the brief description for each of the components.

1. Firewalls carefully analyze incoming traffic based on ***pre-established rules*** and filter traffic coming from unsecured or suspicious sources to prevent attacks. We would need to track codes of visitors from web/ mobile for further analysis.
2. Load Balancer will distribute the traffic into ***multiple webservers*** to increase capacity and reliability of applications. The traffic can be distributed in round robin, least connections or least response time.
3. Web Servers accepts request and serve them to API gateway.
4. API gateway takes call from clients, then routes them to appropriate microservice with request routing, composition, and protocol translation and update whenever new service has been added. So we can use ***Zuul registered with Eureka server*** to discover the services.
5. Springboot microservices makes application easy to build, test, deploy, maintain and even reduce time to bring in market. The most important feature is ***scalability and modularity.*** We can easily scale one application without impacting any other application.
6. Apache Kafka is a distributed streaming platform which can be used to build ***real time streaming data pipelines*** to get reliable data between systems. It has capability to store records in a fault-tolerant durable way and process stream record as they occur.
7. Apache spark is a lightweight-fast cluster designed for fast computation. It is based on ***Hadoop MapReduce*** and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries, stream and batch processing.
8. Cassandra is highly scalable, high performance, fault-tolerant database designed to hold large number of data without fail. It is NoSQL database type.

### Requirements

1. Handle large write volume: Billions of write events per day.

* We can use **Apache Kafka** that follows publish-subscribe model on distributed logs of data in the self-replicating clusters. When data is captured by microservice, it is stored it in original binary format into a Kafka topic assigned to a particular customer.
* Having dedicated topics per customer will give the flexibility to build separate consumers for particular topics, throttle processing of some topics or drop some topics entirely in the case of a data flood.
* Additionally, we can make use of **Cassandra** database provides the ability to write billions of data from Apache spark inbuilt database.

1. Handle large read/query volume: Millions of merchants wish to gain insight into their business. Read/Query patterns are time-series related metrics.

Apache Spark integrated with Kafka, cache memory will help us in implementing this requirement.

* Spark has cluster manager which allocates resources to each application. Once Spark Context connects to cluster manager it acquires executors on a cluster node, these executors are worker nodes on cluster which work independently on each tasks and interact with each other which helps in faster retrieval of data.
* Cache memory can be used for repeated requests.
* Kafka maintains a time index for each topic. So, data will be stored till retention time does not expire. Apache Spark and Casandra database can be used to retrieve data for older time period.

1. Provide metrics to customers with at most one hour delay.

* Kafka topics are clustered, partitioned and offset so they can be multiple reads based on time.
* Cassandra’s main feature is to store data on multiple nodes with no single point of failure. The reason for this kind of Cassandra’s architecture was that the hardware failure can occur at any time. Any node can be down. In case of failure data stored in another node can be used. Hence, Cassandra is designed with its distributed architecture.
* Apache Spark provides in-memory data processing engine that can do ETL, analytics, machine learning and graph processing on large volumes of data continuously.

4. Run with minimum downtime.

* System can be designed in such a way that there is minimum downtime in the application with fault tolerance and scalability.
* Load balancer passes the request from client to web servers depending upon availability of web servers. In case on webserver is down, request can be passed to other servers. Even alert can be set up for down server.
* Microservice architecture will help to run the application independently and in case one service goes down, it won’t affect other services.
* Kafka follows replication policy in which streams are copied in more than broker so that if broker fails other data can be used to provide information to consumer.
* Spark runs on multiple nodes and  if a failure over the active master occurs, the newly elected Master will contact all the previously registered Applications and Workers to inform them that a new master has been elected and the registered applications and Workers gets registered with the newly elected Master.

5. Have the ability to reprocess historical data in case of bugs in the processing logic.

* Kafka Topic has ability to replay message. Messages published to the cluster will stay in the cluster until a configurable retention period has passed by. Kafka retains all messages for a set amount of time, and therefore, can be used to reprocess data in case of bugs.
* Cassandra is used to store huge amount of data from cache of inbuilt database of spark. We can use it in case of bugs which are very old.