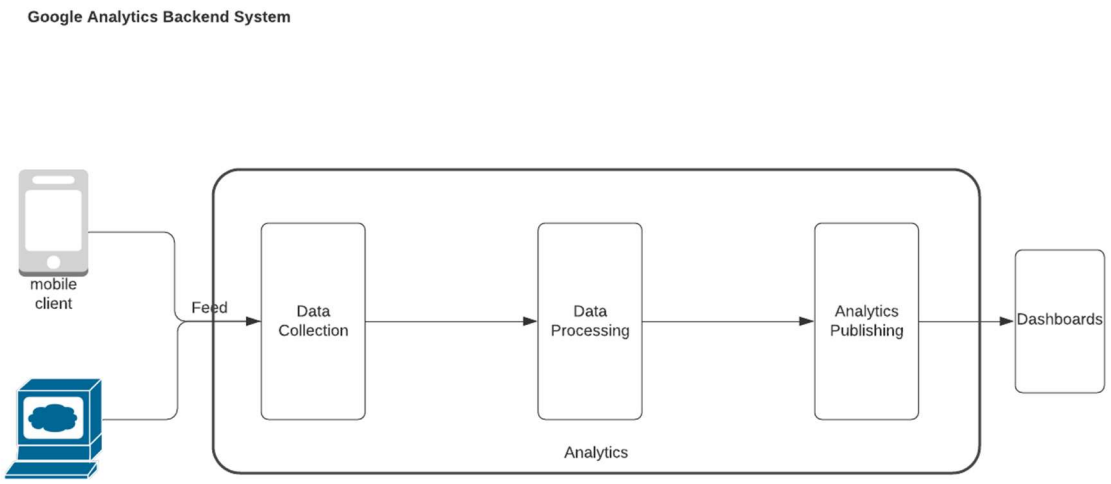
A Google Analytic like Backend System

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.



There are 3 main parts:

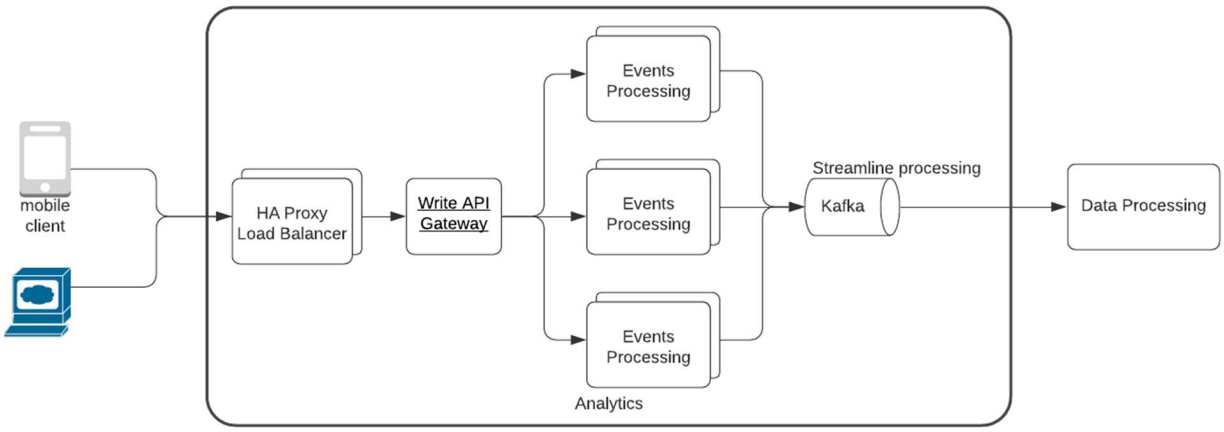
Data Collection part is used to receive the data from users, applications. The system needs to be able to handle the billions of events, writing requests from the web, mobile clients. The system should be stable enough to be able to provide reliable services to the clients.

Data Processing part is needed to process data like filtering, reformatting, analysis and deliver the essential data to other parts. This system should be capable of processing the analytics with minimum delay and should provide the ability to reprocess the historical data.

Analytics publishing part actually publishes the analytics time metrics in different forms, responding to huge number of queries at the same time.

2. Component Details:

2.1 Data Collection



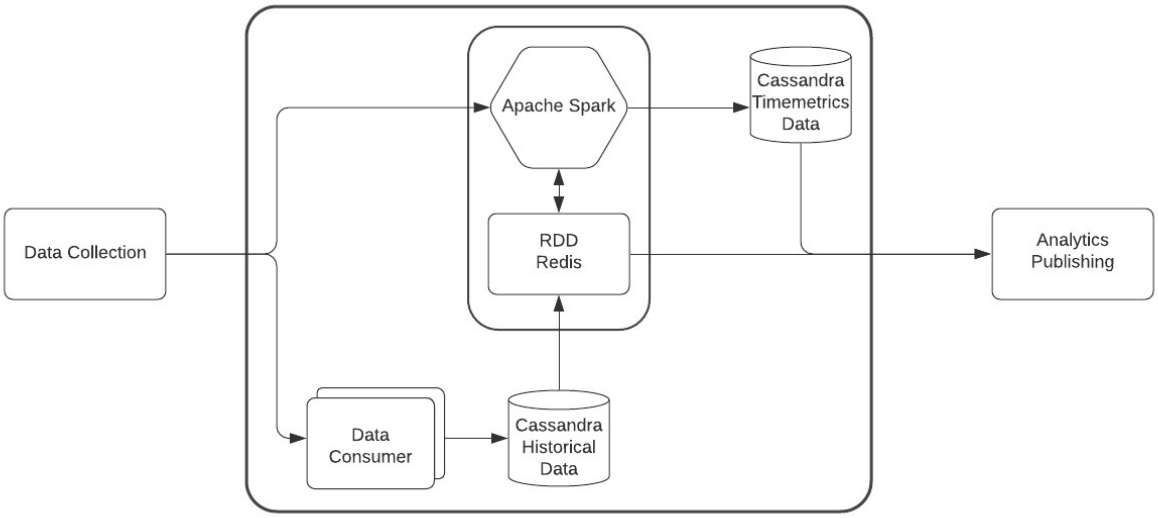
Data Collection service provides API for client applications for receiving data. It is mainly responsible for authentication checking for clients and pushing the data to data streaming middleware. The operations performed are very simple. It ensures that the operations are done fast in order to process a huge volume of incoming data.

Considering above, that there will be billions of requests. System needs to maintain high availability with services spinning on multiple servers to share the incoming load. Load Balancer is used to distribute the load to these services. Different strategies can be used to improve the performance. Load Balancing factor can also be configured accordingly.

For Data Streaming in Asynchronous fashion, Kafka is used. It should satisfy following:

- Pub-sub model for fault tolerance and scalability to process large volume of streaming data.
- high throughput,
- low latency,
- data persistence,
- easy scale. Kafka is pretty fast.

2.2 Data Processing



Data Processing is put in between collection and analytics publishing to the dashboards. It processes the incoming data. It needs to process the data and provide the analytics data for monitoring purpose. It should cover the following:

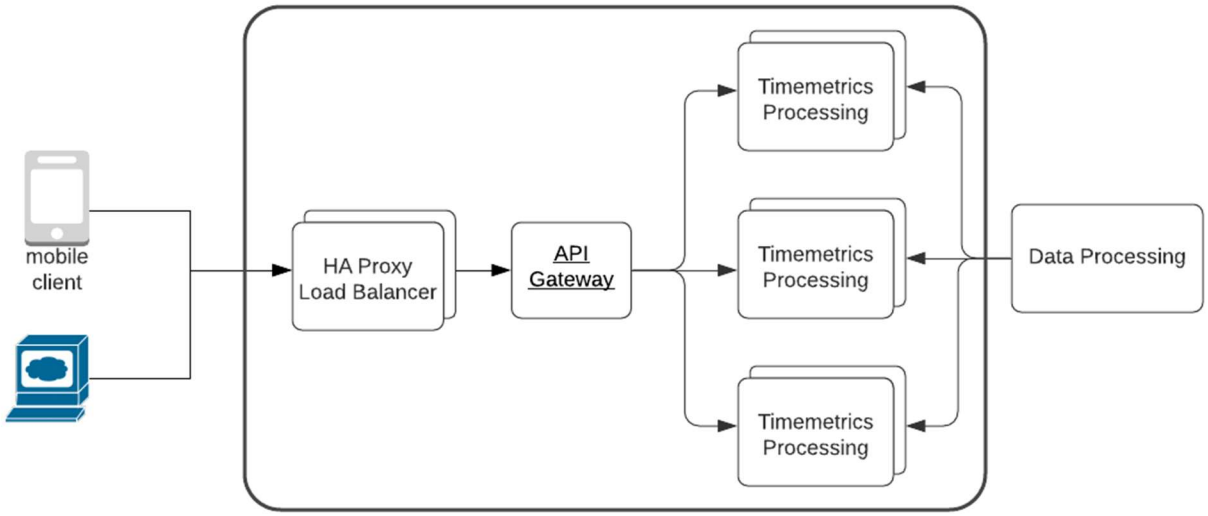
- Processing the data using Machine Learning, AI.
- Persisting the data for reprocessing of historical data.
- Deliver data to Metrics services for visualization on different dashboards

Apache Spark is used for the real-time processing of the input stream of data. It provides high performance with distributed machine-learning framework supported. It uses RDD(Redis) for big dataset collection and analysis of data and state that is shared by multiple spark jobs.

For analytics data persistence, Apache Cassandra should be a good fit. It is a wide column database which is suitable for dynamic data. It doesn't need the schema to be defined and offers good read write performance. Data can be sharded by Timestamps.

It also supports reprocessing of historical data.

2.c Analytics Publishing



This basically works with different business requirements, different type of visualizations. To make it scalable, it is designed with microservices based design, domain driven design.

To make it simple with number of services increasing and running, API Gateway is used to:

- Single Entry point for all the microservices
- Avoid redundancy in terms of same logic in all the services
- Routing to the appropriate service
- User authorization

Every service has its own database based on the queries that service needs to provide.

To process the high volume of write/read requests, multiple services should be spinning out to process and serve the time metrics. If the read load is high, separate read and write model is another approach to be taken. A system needs to available, scalable, reliable, durable.