# Google Analytic Backend system design

## Main Features

1. Site owners / clients can register their websites on our analytic system.
2. For users who accessing the registered sites, accessing data will be retrieved in our analytic system.
3. Sites owners can get the insight of their websites, such as the QPS, the location distribution of users, the average time they stay, etc.

## Design goals

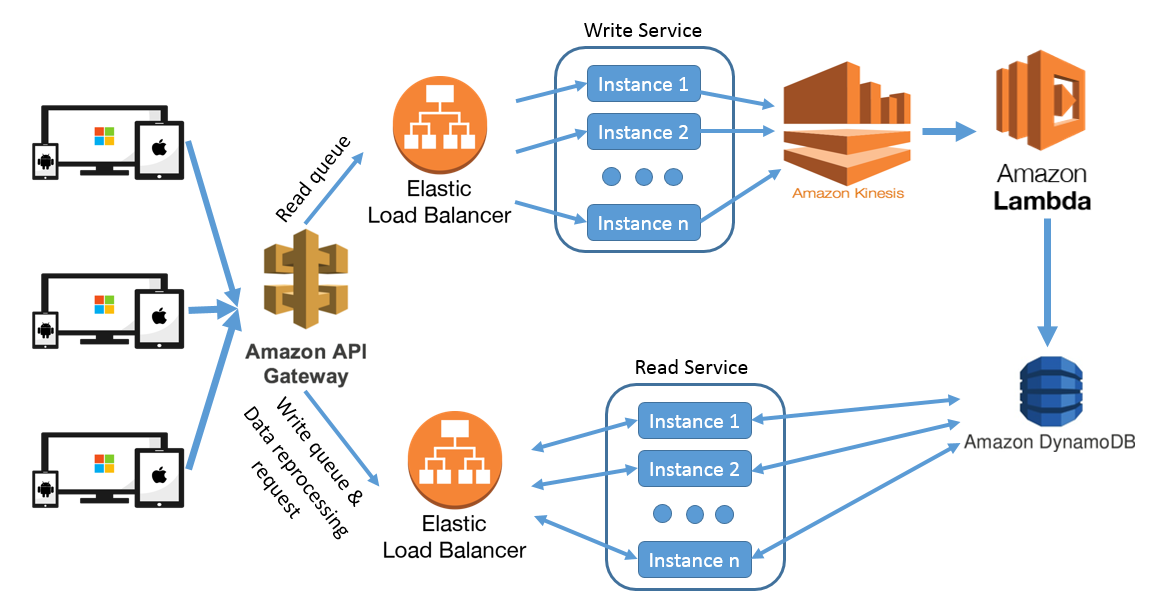
Referring to the CAP theorem, we can only choose two from Latency, Consistency and Availability. And based on the following requirements:

1. Provide metrics to customers with at most one hour delay.
2. Run with minimum downtime.

Our goals is to design a low latency and high availability system. Our system allow data to have delays.

To handle with high QPS services, the system should have scalability and preventing single point of failure, so in our design, we should take care about the load distributing. Nowadays, companies should as amazon provide auto scaling on their services and we can make our system much more stable and scalable by using their services.

## Big picture of the system design:



## Breaking the system into micro service

I will try to break the analytic system into 2 micro services

1. Write Service: Handling the incoming users’ data, i.e. the user accessing websites data. Basically we will get the HTTP request information and process it to the data we need. In the server side, we may convert / append data for the analytic uses, such as getting the physical location from IP, appending the dates or any useful information based on our business logic.
2. Report Service: To retrieve an insight or report of clients’ business. The back end is responsible to select data from the database and return the necessary data (filtering and formatting the data) back to the front end.

It also responsible to handle fixing incorrect data if there is bugs. As I expect this will not happen frequently. So instead of creating one more micro service, I choose to put it inside the report service.

## Distribute request to corresponding service

In order to receive the HTTP request and redirect to different services, I choose to use amazon API gateway. By configuring the path in the gateway, we can redirect the request to target instance based on the HTTP request. Besides, it also give the ability to add and remove micro services. For example, if we want to extend the analytic system to allow clients to add/update data, we may create one more micro services and easily add to the API gateway.

## Load balancing

As we expecting there are billions write events, which means our system will handle around 12000 QPS. We definitely need to distribute the queue in to different instances.

Here I choose amazon elastic load balancer. It provides algorithm to distribute load to different instances, such as merry go around, or based on the CPU usage.

Besides it also provide auto scaling, so we may freely to increase the instance with the load balancer functions, as well as save cost if our system have non-peak hour.

## Data processing

After our backend services prepared the data for analyzing, I will pass the data to amazon Kinesis and Amazon Lambda for data processing.

Amazon Kinesis is a data streaming services, it able to processing data near in real time and is easy to integrate with any other amazon data processing related services.

AWS lambda is a computation / calculation services, which work well for times series data computation. By defining the algorithm or the data we want to calculate, it will continually receive data from Kinesis, processing and insert the result to the database.

## Database choice

As mentioned in the CAP theory that we want to provide a low latency services, and consistency is not an important point for us. So noSQL will fits as the most, as we can skip table joining which spend a lot of computation time. Although the data may not be consistence, noSQL support eventually consistence, which still fit the “at most 1 hour delay” requirement.

Here I will also use the amazon service, the AWS DynamoDB.

## Provide business insight to client

After the data is ready in database, then this will be only simple database selecting.

Our backend server will be responsible to queue the database based on client request. For example, user want to know the webpage accessing detail in this one hour, then the server will queue the data for that 1 hour, and wrapping to the format for the front end rendering.

## Reprocess historical data

In order to reprocessing the data, firstly we need to have to original data in our database, which we can configure Amazon Kinesis and Lambda to insert the original data to database.

Secondly before reprocessing the data, I expect the business logic bugs in the Amazon Lambda was being fixed.

Lastly we should have an API in the report services to allow developer to request data processing. If the report service receive a data reprocessing request, it will collect the data within the request time interval, resend to the Kinesis and reprocessing the data.