Designing Backend like Google Analytics

This is an agnostic solution. I talk about the logic of the system and how it will be constructed. I think that all concepts in here are true whatever tools we will take.

The concrete proposition is at the end of this document.

System volumetry

- Billions of requests from clients by day
- Millions of merchants will ask analytics results

Solution requirements

- We need to provide results before 1 hour for each merchant
- We are looking for the minimal downtime
- We need to be able to track a request and reproduce it easily

Client API

I'm assuming we providing a backend and every merchants got his own frontend on his side.

Since we could have a huge variety of websites and apps from Android to iOS and so on, we need to expose a simple REST API since every language and framework will always provide a REST HTTP API. We'll need to provide a good documentation and prepare to assume support if clients don't understand the API.

To limit this problem, we can provides libraries in a wide range of languages

Pros	Cons
No risk of bad analytics since client can less messed up the integration	Will cost a lot to our company to create and maintain each library

We could provided those libraries under open-source licence to help support more languages without too much costs.

Database capacity

	Billions	Result
Low	1	`1024 * 1 000 000 000 = 1 024 000 000 000 bytes`
Medium	3	`4096 * 3 000 000 000 = 12 288 000 000 000 bytes`
High	6	`8192 * 6 000 000 000 = 49 152 000 000 000 bytes`

- Max request size was arbitrarily determined
- Max billions were determined by Google searchs by day.

  It's probably one of the most used services in the World, so it's a good reference.

1.024 terabyte to more than 49.15 terabytes.

Database

- A raw database with all data inside.
- A middleware to analyze data and put them into specials analytic database.
  - there is no need to query multiples merchants at the same time
  - better query performance
  - better security, no data can mixed up together
- We need to remove data periodically
  - If we don't, amount of data will be unbearable
  - It will limit analytics results possibility

Back API to get query from clients

We need to use Microservices technology for scalability of traffic. API Gateway to route traffic.

Concept

- Control coherence of data
- Put every data into database for a specific merchant.

Structure

To get a best precision of our storage cost, we need to:

- API will provided a final list of what we can track or not, no free area.
- Let merchants send sublist of what can be track.
- We need to put quota on each IP Address and merchants to avoid DDOS and jaming.
    - Limits from Google Analytics

Tracability

Every call of the API will get a special trackId. This id will be the same in every step of his life. It's like a tracking number. It will be useful to help reproduce a call.

This track id will be generate from `IP + merchant id + timestamp` and send in HTTP request in a special header. We need to use that ID from API Gateway to the end. **It means, client should generates it well**.

Backend to get query from merchants

Microservices technology for scalability of traffic. API Gateway to route traffic.

Considered solution

This backend provides a graphic interface where merchants can display results and filter them.

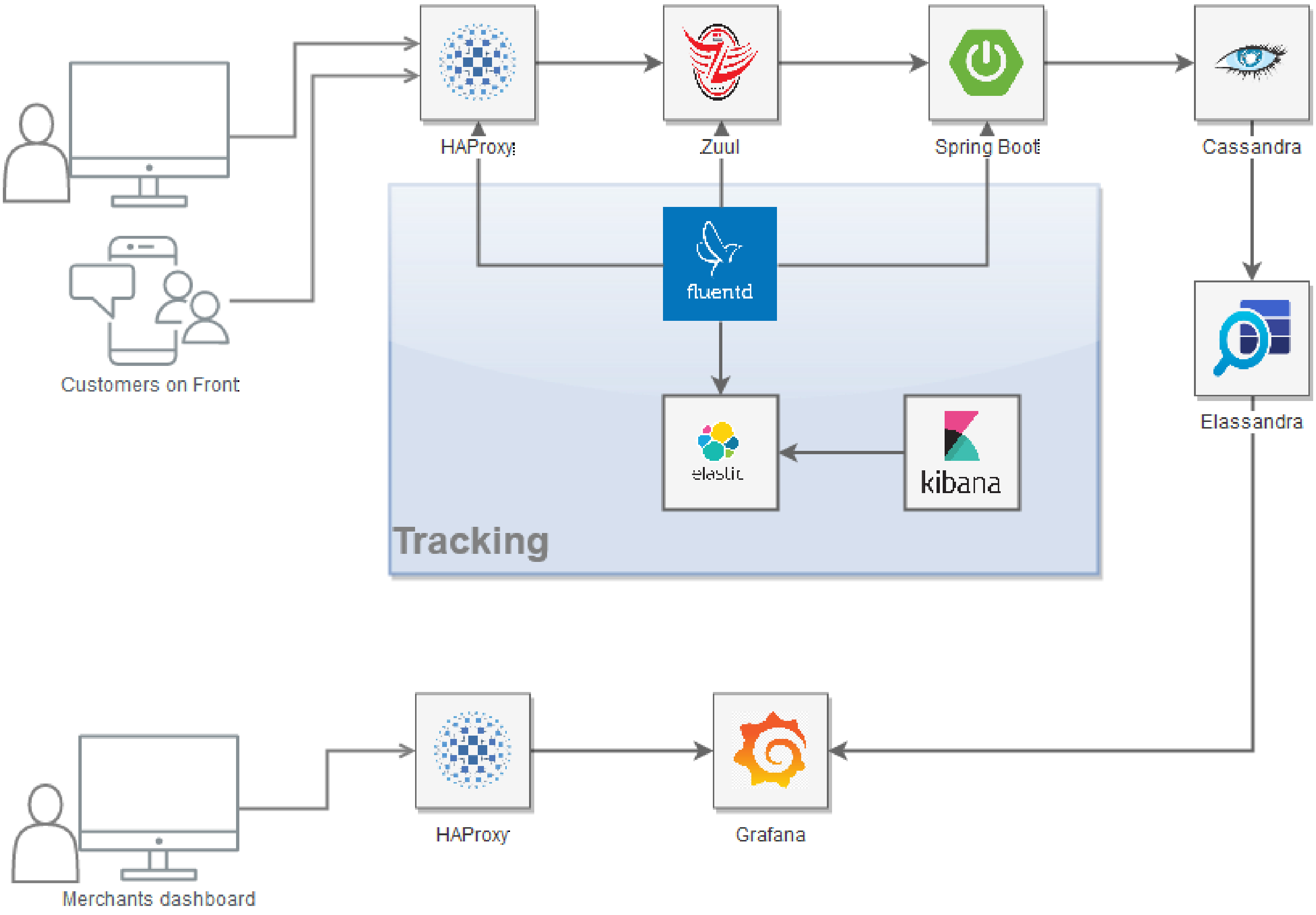
Needs

This service needs to provide to each merchants data before 1 hour.

And now, what are we using?

This proposition is based on feedbacks read online. That kind of task should be a team work to exchange ideas and to confront thoughts.

Concrete proposition



Type	Solution	Why
Load balancer	HAProxy	Open-source, popular and often used in many high-profile environments like GitHub, Imgur, Instagram, and Twitter. **Can generate a unique tracking id.**
Gateway to API	Zuul	Developped by Netflix, last release from few weeks ago. Largely used.
Pairing between Gateway and API	Netflix OSS Eureka	Eureka server acts as a registry and allows all clients to register themselves and used for Service Discovery to be able to find IP address and port of other services if they want to talk to. Pairing very well with Zuul and Sprint Boot.
API for clients	Spring Boot	Spring Boot is very famous and regularly updated. Works well with Eureka. Using JVM ecosystem is a mature environment and provides a huge range of libraries.
Tracking System	Elasticsearch and Fluentd and Kibana	HAProxy, Zuul and Spring Boot will write logs with the unique track id. Fluentd will parse them and expose it to elasticsearch. We will be able to reproduce requests with this. We can use Kibana to read easily results.
Backend for merchants	Grafana	Wide ranges of graphics, Good pairing with Elasticsearch but can be used with others inputs too. Designed for Graphics and analysis.

Type	Solution	Why
Database for raw data	Apache Cassandra	Fault tolerant, Free, Scalable, Distributed Processing. Can manage petabytes of data.
Analytics provider	Elassandra (Elasticsearch fork)	We can easily extract data we need for analyze from Cassandra with it. Grafana can read it easily.

Sources (some are in French) :

- Google Analytics (GA) like Backend System Architecture
- (fr) Microservices for big e-commerce website
- An Example of Load Balancing with Zuul and Eureka
- Migrating Messenger storage to optimize performance
- Cassandra Vs MongoDB In 2018
- Cassandra vs. HBase
- Apple has the biggest Cassandra instance
- Elasticsearch for Hadoop
- Complementing Cassandra with Elasticsearch
- Elassandra : Elasticsearch+Cassandra framework
- HAProxy can add headers
- Monitoring HAProxy Real-time with Elasticsearch and Fluentd
- Fluentd vs. LogStash: A Feature Comparison
- Zuul Filters, log customization
- Elasticsearch cluster
- Scaling Elasticsearch
- Grafana vs Kibana
- Grafana vs. Kibana: The Key Differences to Know