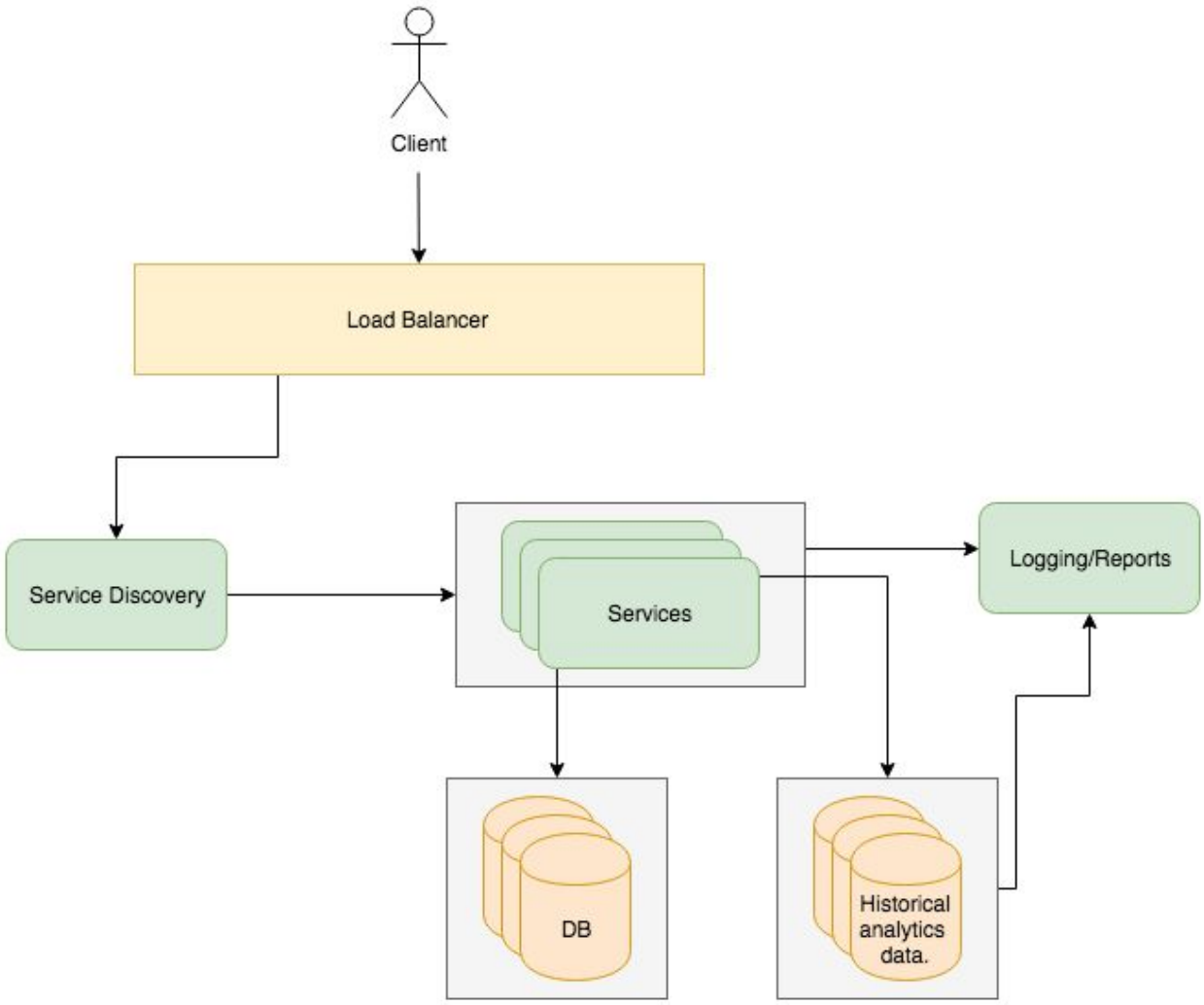
Google Analytics Backend Design



Assumptions and Overview :

- My proposal of the design assumes that money is not a problem.
- This proposal only highlights the high level design of the system. Details about programming language to be used, implementation of DB clusters, implementation of load balancer is not discussed here.

Details:

- The basic architecture is that of a micro service. Details of the micro services need thorough analysis of the implementation of the system and is not discussed here.
- Why should we use microservices ?
    - Allows scaling up of particular services when needed. For example, the service that runs the analysis algorithms can be scaled up when needed. At the same

time, the service handling the user profiles can scaled down. This allows selective scaling since the services as decoupled unlike a monolith.
    - Easier deployment. Got a bug in production user profiles feature ? Just fix the user profile service and redeploy instead of redeploying the whole application.
    - Another reason which may not be so important is that it is easier to take in new developers as they won't need to know the entire context of the application and can start working on a "box" of service.

- The Api Gateway provides security to internal services and takes in requests for the analysis reports. The requests are then directed to the appropriate services using a service discovery.
- In case of bugs in the processing logic, historical reports can be used to provide analysis reports on old data. Log reports can be used to analyse the requests and the response generated for the same.

Database and tech stack:

- Since the system needs to handle large queries and billions of read/writes every day. I would suggest using a NoSql, particularly a wide column store like Cassandra/Hbase/BigTable. Wide column stores provide much faster query processing and stores data that is somewhat related.
- Google's Big Table uses the google file system on the cloud and maybe more performant in terms of writes but again, this recommendation assumes that money is not a problem and no one minds paying Google. Further discussion can be done on the type of DB to be used.
- For the microservice architecture, I would recommend using Netflix's Spring framework. The framework includes a tons of services like API gateway (Zuul) which provides security as well as monitoring and a bunch of other features. It also include a service registry which provides efficient load balancing.