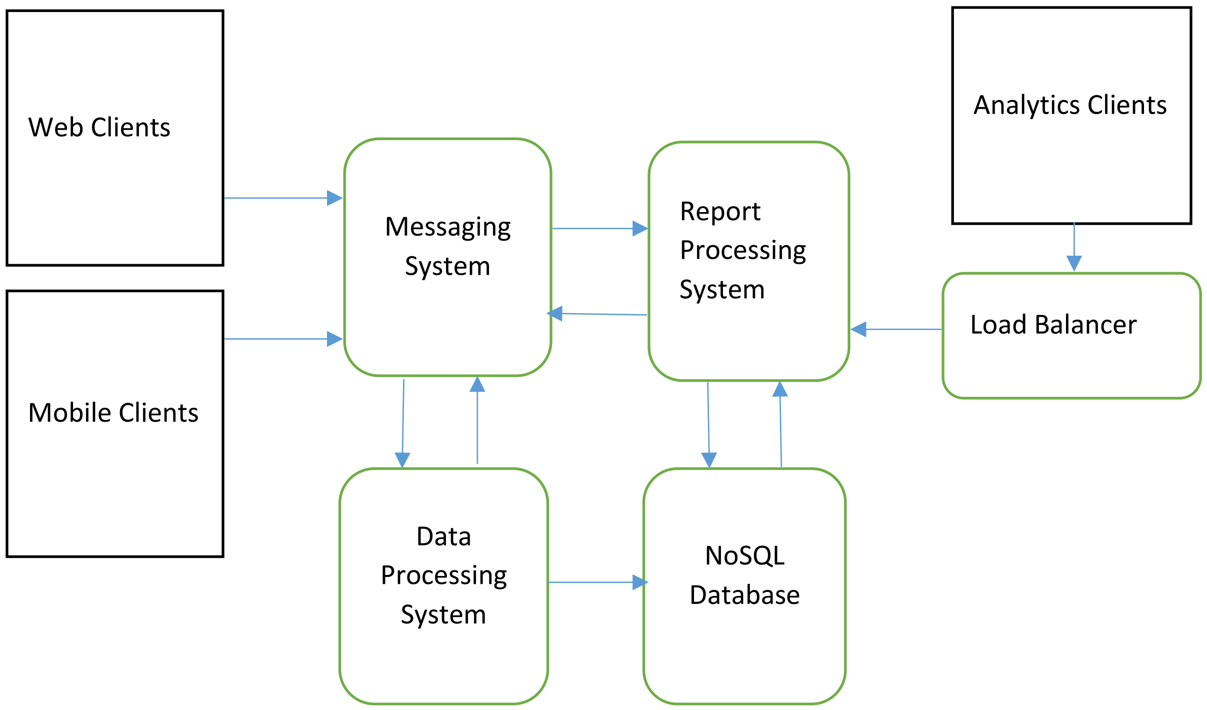

Messaging Middleware System:

Since the data volume is huge, ad hoc processing is typically not realistic. In this case, we will use client side embedded JavaScript to capture user interactions as events and publish them to the message topics. These events will be then queued up and will trigger the corresponding process in the downstream system (Data processing system). Apache ActiveMQ, Kafka are both good choices since they provide fast and scalable services.

Data Processing System:

This system will be subscribed to the message topics and process the raw data to produce report-ready data to be stored in a NoSQL database. After each successful processing, it will send a message back to the queue indicating that the data is ready to be served in the UI/report. If an error occurred during processing, an error message will also be published to the messaging system, a separate process will be trigger by the error event and start reprocessing the original event. Since we are dealing with a high volume of data, I would go with Apache Spark for its reliability and scalability.

NoSQL Database:

As we want to store a large volume of unstructured data to be used in analysis and reports, I would go with a NoSQL Database. In order to make it more reliable, we might want to have a backup database to be synchronized with the master database daily, so that we can more easily recover from data loss and system failures. MongoDB and Apache Cassandra are all good choice for NoSQL databases here.

Report Processing System:

By this time, we should have most of the report data ready in the NoSQL database. The report processing system check the database first to see if the data is available and format the data to be used in the UI/report. It also has the ability of publish message to the topics and order data processing system to reprocess the data if certain piece of data is missing or corrupted. The system will be notified again by the messaging system once the data is ready to be served. Since we are dealing a lot with formatting JSON data and file processing, I would just go with a simple NodeJS application.

Load Balancer:

As we have requests coming from millions of merchants, it is wise to have multiple service instance and load balance the incoming requests. NGINX and HAProxy are among the best of the open source load balancers.