Troy Nguyen

009938461

CMPE172

Midterm Case Study

# Case Study on AppLovin

1. **What do you propose AppLovin’s Computation model is?**

AppLovin’s Computation Model is Lambda architecture. Lambda architecture is designed to process massive amount of data with batch and stream processing methods. Lambda architecture is very helpful with processing big data. AppLovin claims to process 30 billion requests and 60 terabytes of data per day. Therefore, AppLovin infrustratucture should be designed to handle big data to be latent, throughput-balanced, and fault-tolerant. Similar to lambda architecture, AppLovin consists of three layers: batch, speed, and serving layer.

Batch layer provides pre computing query functions called batch views, and it provides immutable set of raw data. Batch can be in a distributed file system, so I believe AppLovin uses mmap to serve these batch views. Speed layer is for processing data streams in real time. I believe AppLovin uses Nginx as part of their speed layer due to its efficient performance to serve an ad in less than a millisecond. Serving layer indexes, the batch layer so they can be queried with low latency. Serving layers are best implemented with NoSQL technologies such as AppLovin use for AeroSpike for its heavy number of user profile.

1. **How do you justify the rationale behind replacing Node specifically in AppLovin**

Node is an Javascript runtime environment. It consists of event-driven architecture that allows asynchronous I/O, which makes it lightweight and efficient. Node was designed to improve performance in web applications with many I/O operations such as messaging system. However, there are performance limitation due to Node’s design. Node events are run asynchronously without multithreading, therefore, is the event loop is single-threaded by design. Due to single threading, it does not provide scalability and it is not effective for CPU-intensive tasks. Node.js asynchronous event-driven I/O runs on one CPU, which is not enough for CPU intensive tasks. Especially for AppLovin model, which is based for optimizing the best computing performance, it is not very ideal to use Node.js when there are other more powerful languages that are threaded-driven and are suited for CPU intensive tasks.

1. **Aerospike is used for storing user profiles. (MongoDB was replaced by Aerospike). Explain the difference between Aerospike database and MongoDB for storing user profiles?**

Aerospike is a key/value-store database and MongoDB is document-store database. Document databases store entire documents in JSON objects, and these objects can be nested key-value pairs. Key/Value databases use associative arrays like dictionary or hash.

Unlike many other database solutions, Aerospike is flash-optimized with RAM/SSD storage architecture. They claim that the performance can be 10x better with 10x fewer servers at the cost of 10x less than main memory. For high scaled design like AppLovin, it’s hard to turn down Aerospike. According to Aerospike benchmark, the maximum throughput of SDD-back data is 5-10 times faster than MongoDB. Therefore, in practice, Read/Write throughput by Aerospike is virtually unbeatable. With vast amount of user profiles that is continuously updated, MongoDB will take much longer in worst case scenario since it does not provide a set structure or allows nested pairs. AppLovin claims to use Aerospike for ad serving to user profiles, which resultsless than 1ms to fetch a profile. AppLovin seems to very pleased to this data, something MongoDB isn’t able to match. Furthermore, there is a notion that Aerospike is more reliable than MongoDB in terms of fault tolerance because of its self-managing architecture to detect failures.

Overall, with large number of user profile and transactions that need to be consistent 24/7. Aerospike is proven to be more reliable, significantly more efficient, and effective than MongoDB.

1. **Refer to the technology stack and architecture overview and describe at-least 4 technologies (that is of interest to you) used in the AppLovin**

**Asana** is a powerful task management software tool. It allows teams to share, plan, and track the progress of the team’s tasks and projects. It also allows teams to communicate as a group or one on one. It is one of the best existing task-management tool. One of the reason why Asana is so successful is because of its ease-of-use for a fairly powerful management tool. Asana also has many integrations with Dropbox, WordPress, Usersnap, etc

**Redis** is an open-source, in-memory database solution. What makes it different from other systems is that it supports and specifies many data structures such as strings, sets, hashes, list, etc. Redis supports mast-slave replication as the data can replicate to a great number of slaves. Redis is good for high scalability data store systems than is shared by multiple resources. Newer versions of Redis also support clustering, which makes it more scalable.

**VividCortex** is a monitoring system to see what MySQL is doing. It is used to measure and analyze the queries. One key feature is that VividCortex shows chart and alert on threshold. VividCortex measure more than just status, it also measures the queries behavior. VividCortex seek to reduce companies’ time for employees to manually measure and analyze the queries in production by automating the process.

**HipChat** is a group chat software that is built mainly for businesses. It has many other features besides chatting such as file sharing and integrations with other services. The software is cross-platform for any available devices. Private and public chats are available on Hipchat, as well as group video chat. These team-chat applications are great for constant updates and communications that unwanted on email or phone messages.

# References

<http://highscalability.com/blog/2015/3/9/AppLovin-marketing-to-mobile-consumers-worldwide-by-processi.html>

<http://marketingland.com/hyperlocal-mobile-advertising-changes-everything-92979>

<http://adage.com/article/digital/mobile-targeting-accurate-cookies-desktop/240464/>

<http://www.aerospike.com/user-profile-store/>

<http://s3-us-west-1.amazonaws.com/aerospike-fd/wp-content/uploads/2016/09/CS03-AdTech%E2%80%93Real-Time-Bidding.pdf>

<https://www.mapr.com/developercentral/lambda-architecture>