Lab 2, Microservices at Netflix

HOU MAN [Raymond] Xie

[This lab was proofread by ChatGPT]

Netflix originally was structed as a monolithic architecture, meaning that all the services that they provided were tightly coupled together within a single code database. As Netflix's business grew bigger, they started to see issues with higher demands for streaming services and reached a dead point where scalability, development bottle neck issues had occurred, not to mention the possibility of a single breaking point in their services, so to resolve this problem, Netflix decided to adapt to the microservice architecture, which decomposed their monolithic systems into smaller services, which are responsible for a specific function.

Their microservices design composes of user-profile, content recommendations, streaming delivery, and billing/payment processing.

Microservices architecture offers advantages like scalability, as services can be scaled independently to handle demand, fault tolerance through service isolation and fault testing, and faster development enabled by independent teams working in parallel. It also provides flexibility with polyglot persistence and ensures performance through CDNs and regional deployments. However, it introduces challenges like increased complexity in managing distributed systems, higher operational costs, additional latency from inter-service communication, and the need for robust CI/CD pipelines to handle frequent updates.

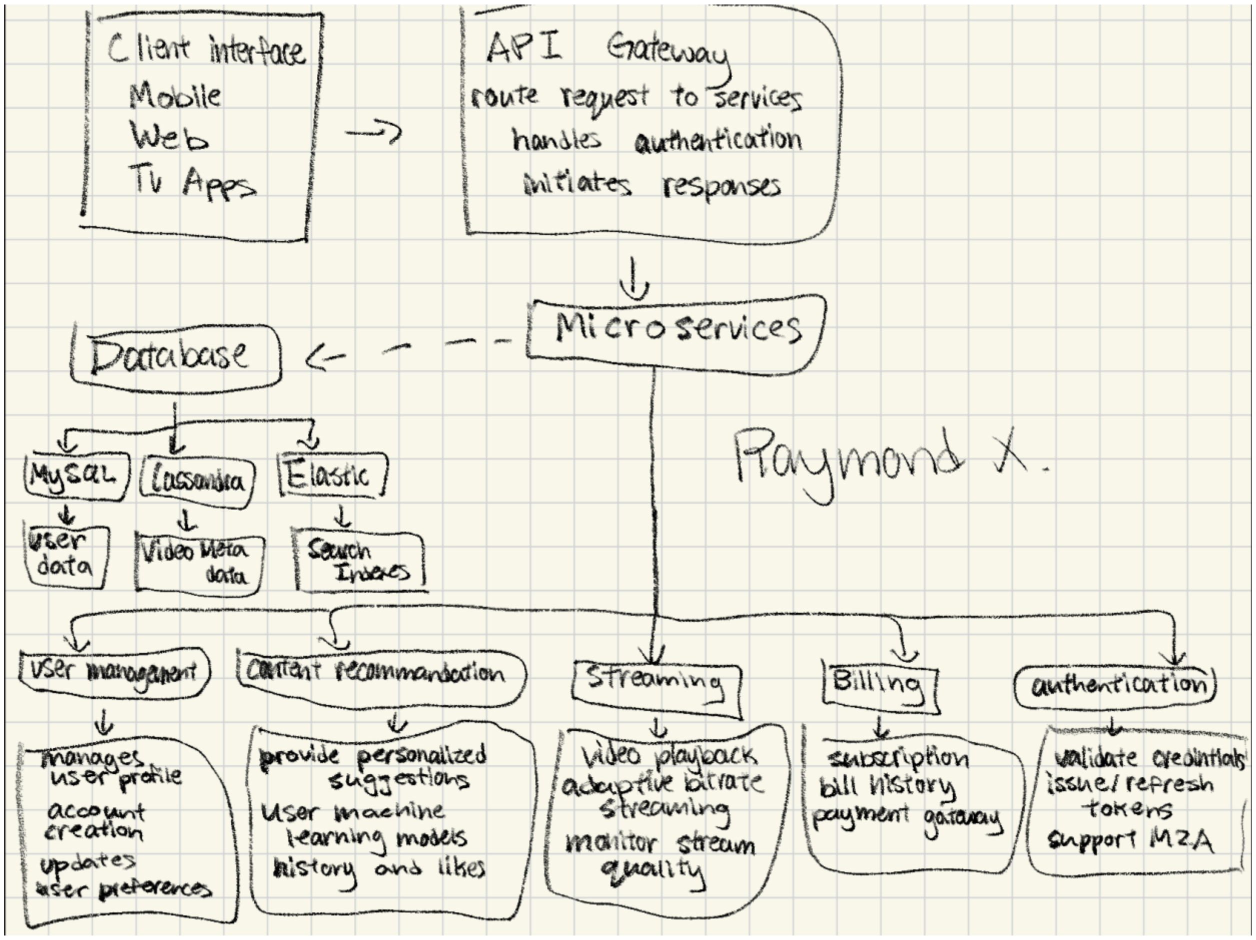
Each microservice is designed to scale independently based on demand. For example, "streaming service" scales up during peak hours, while less critical services like "account settings" can remain unused which is stable.

To improve system security, Netflix introduced chaos engineering and tools like Chaos Monkey to simulate failures and test the system's ability to recover. Microservices are designed with features like, circuit breakers, which prevent continuous failures by isolating and managing failed or failing services, retry logic, which handles transient failures, and the most important thing being the fallback system, which provides degraded functionality if a service failed.

Each microservice manages its own database, ensuring data independence and reducing the risk of bottlenecks. Netflix utilizes polyglot persistence, where different services use databases best suited to their needs like "MySQL", and "Cassandra".

Netflix employs an API Gateway as a single-entry point for customer requests. The gateway routes requests to the responding microservices, handles authentication, and response back to the customer via the corresponding microservice.

Through Microservices, Netflix achieved better reliability, which increased the fault tolerance of the site/app itself, a more rapid deployment, which means a faster time to create/market new features, and most importantly, scalability.



Sources:

https://about.netflix.com/en/news/completing-the-netflix-cloud-migration

https://www.geeksforgeeks.org/the-story-of-netflix-and-microservices/

https://dev.to/yashrai01/architectural-battle-monolith-vs-microservices-a-netflix-story-2ddk

https://expertcloudconsulting.com/blog/netflixs-journey-to-microservices-revolutionizing-streaming-with-scalability-and-innovation-1