NETFLIX SOFTWARE ARCHITECTURE

Addressing Quality Attributes

Sandeep Kumar Sutharapu
School of science and engineering, Saint louis university.

Abstract:

"NETFLIX" which is one of the prominent entertainment streaming providers. It is launched in 1997 as a DVD rental startup, in 2008, due to the heavy demand for DVDs, Netflix began an infrastructure transformation allowing them to embrace scalable, robust data infrastructure management. We all are familiar with Netflix services. It handles large categories of movies and television content, and users pay the monthly rent to access these contents. Netflix has about more than 180M subscribers in 100+ countries. Handling such as complex functionalities within a system is not just a simple task. Ensuring uninterrupted, 24/7 access to such a complex functionality is no small feat. This paper will investigate the architectural aspects that have made Netflix's success possible in the digital data driven world. Netflix's architectural evolution has played a pivotal role in its ability to deliver seamless streaming experiences to viewers worldwide.

This architectural structure of Netflix modified from a monolithic system and divides the application into a network of smaller and interconnected services, each with a specific function. This implementation divided the system into independent services and formed a network of services called as Microservices Architecture, these services individually handle various functions such as content delivery, user authentication, recommendation engines, and payment processing. Microservice architecture focuses on both how individual systems are developed and how they communicate with each other. It's an approach to designing and building software systems. In this paper will manifest how the Netflix architecture addresses the identified quality attributes. These Quality Attributes influence the prioritization and are implemented by making necessary trade-offs.

Introduction:

We often speak about layered architecture, which allows developers to build specific components within a software, such as the Presentation layer, business layer, Persistence layer, and Database layer or about a client-server architecture. However, in today's world, where software systems are rapidly developed and integrated, there is much more importance of software entities how they are connected, how communicating is achieved with each other and sharing information, providing a level of abstraction. Sounds weird right, imagine a scenario where a customer tries to make a subscription purchase on the Netflix platform. He may choose to pay using a credit/debit card, bank transfer, or an app like PayPal. Does the platform contain all the banking logic required to process these diverse payment methods? No, that would be impractical, incorporating the banking logic into the Netflix code is like building an entire restaurant just to prepare one dish. While you can simply order your food from a restaurant.

Netflix doesn't need to be a full-fledged bank to handle payments. Instead, it relies on services provided by banks, payment gateways to process transactions securely and efficiently. Components of Software typically organized using module structure, connected using Component & Connector structures. How can a software or a service be considered a module itself? However, software to communicate to different systems and other services is considered a portion of enterprise architecture. In this paper we hold up to a perspective where the scope of the architecture is not a specific task-oriented software zooming out it's like a larger software Architecture that is build up on integrating different software, services, data sources etc. Let's move further into the detail manifestation of Netflix architecture and its implementation by microservice architecture.

The System:

Netflix adopted Amazon Web Services for managing their entire application and IT infrastructure, and they replaced their existing monolithic architecture hosted on their data servers loosely coupled with microservices architecture hosted on the public cloud. Communication between these microservices is facilitated through Application Programming Interfaces (APIs), allowing them to work together seamlessly. "The Microservices Architecture within the Netflix system primarily comes into play in the Backend module of their architecture." Various microservices are responsible for handling different tasks related to content management, video processing, content distribution, and network traffic management. Each microservice is designed to take care of a specific task/role, and they communicate through APIs to work together. This architecture allows Netflix to effectively manage and scale its operations, ensuring the high-quality streaming experience that millions of users enjoy worldwide.

Microservices:

Netflix's microservices architecture is a complex and distributed system with numerous services. While the exact number and specific services may have evolved over time, here are some of the key types of services:

Edge Services: These services handle client requests and often serve as the entry point for external requests. They include services responsible for user authentication, API gateways, and request routing.

Core Business Services, Billing and Payment Services, Data Services, Playback Services, Operational Services, Recommendation Services, Content Delivery Services, Device Management Services, Content Licensing Services, Testing and Quality Assurance Services. These are general categories of services, and within each category, there can be numerous specific microservices that perform various tasks.

The Architecture:

Netflix High Level System Architecture has 3 components,

Client: Device which is used to browse and play Netflix videos. TV, XBOX, laptop, or mobile phone, etc

Open Connect or Netflix CDN: Is the network of distributed servers in different geographical locations, and Open Connect is Netflix's own custom global Content delivery network. It handles everything which involves video streaming. It is distributed in different locations and once you hit the play button the video stream from this component is displayed on your device. So, if you're trying to play the video sitting in North America, the video will be served from the nearest open connect or server instead of the original server which gives faster response from the nearest server.

Backend: This part handles everything that doesn't involve video streaming such as onboarding new content, processing videos, distributing them to servers located in different parts of the world, and managing the network traffic. Most of the processes are taken care of by Amazon Web Services.

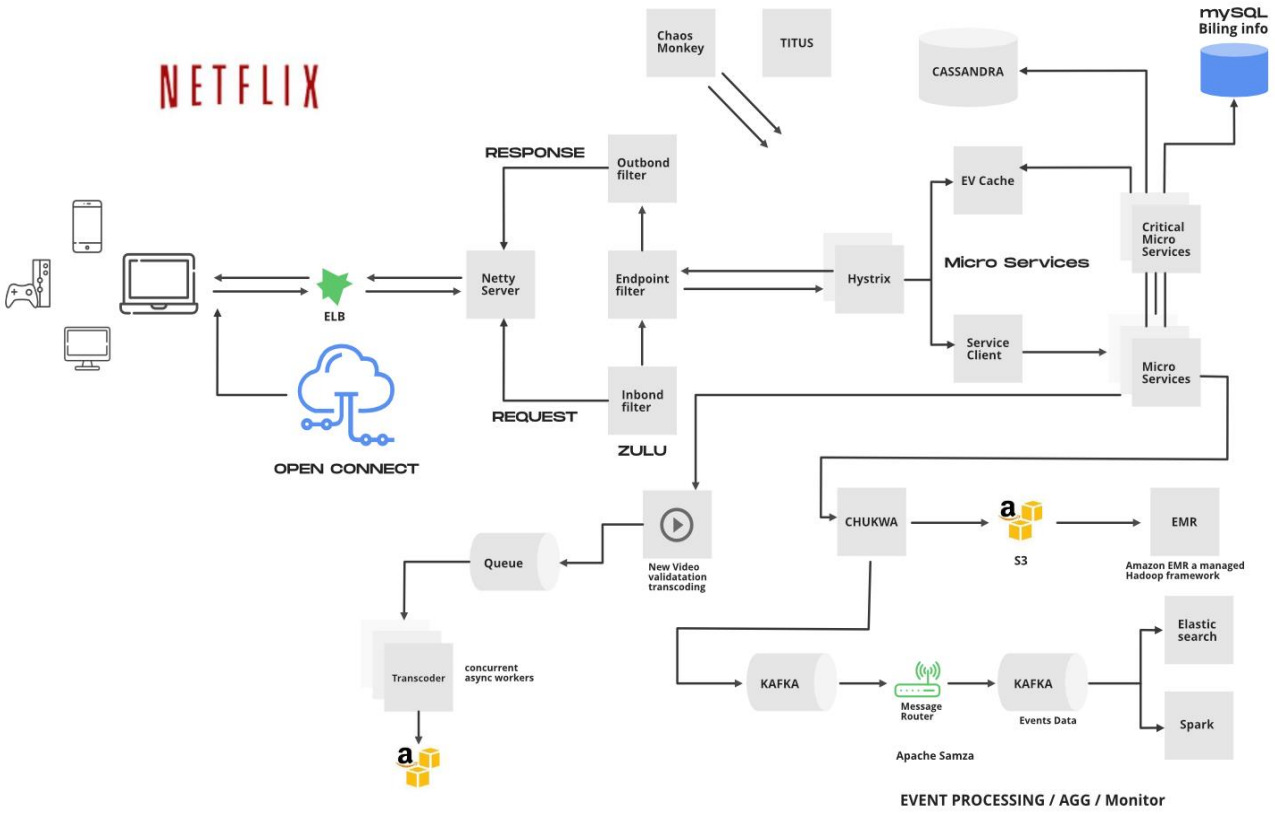
So, while the "Client" module deals with user devices and the "Open Connect or Netflix CDN" module handles video streaming and content delivery, it's the microservices within the "Backend" module that play a pivotal role in ensuring the efficient operation of Netflix's extensive streaming platform. This modular approach allows Netflix to continuously innovate and meet the demands of its vast subscriber base.

Netflix's microservices architecture is known for its granularity, where small, focused services work together to provide a seamless streaming experience while allowing for easy maintenance, updates, and scalability.

The specific services may change over time as Netflix continues to evolve its platform and services to meet user demands.

"An architecture can either inhibit or enable a system's driving quality attributes"

Architecture Diagram:



KEY Quality Attributes:

"A quality attribute is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders beyond the basic function of the system. Before going through the quality attributes, I would like to make a clear point that QA does not exist isolated"

Availability:

High availability and reliability are important for a streaming service like Netflix. Availability refers to a property of software, that it is there and ready to carry out its task when you need it to be. what is normally called reliability. Below are the methods adopted by Netflix architecture to address availability.

Active Redundancy:

Netflix primarily employs active redundancy as a key element of its high availability strategy (hot redundancy). In an active redundancy setup, multiple instances of critical components, such as microservices are actively serving traffic simultaneously. This approach ensures that there is no single point of failure. If one instance experiences issues, traffic can be instantly routed to the redundant instances.

Service Isolation:

Performance:

Performance is defined as "It's about time and the software system's ability to meet timing requirements". Streaming high-quality video content in real-time demands exceptional performance. Netflix architecture adopted below method to address performance ensuring that the streaming platform.

Scalability:

Scalability is achieved through a combination of microservices, load balancing, and auto-scaling mechanisms that allocate resources as needed. It can handle a massive number of concurrent users and content requests. Netflix can scale its different services independently and rapidly via horizontal scaling and workload partitioning. More extensive software programs are broken down into smaller programs or components based on modularity, and every such part has its data encapsulation.

Elastic Load Balancer:

The microservices architecture isolates services from each other. A failure in one service does not necessarily impact others. This isolation keeps localized issues from affecting the broader system.

Automated Recovery:

Automated processes can replace failed instances, reroute traffic, and apply patches as needed.

Hystrix:

In a complex distributed system, a server may rely on the response of another server. Dependencies among these servers can create latency and the entire system may stop working if one of the servers will inevitably fail at some point. To solve this problem, they isolated the host application from these external failures. Hystrix library is designed to do this job. It helps you to control the interactions between these distributed services by adding latency tolerance and fault tolerance logic.

Security:

Protecting user data and content rights is highly essential. Netflix's microservices include robust security measures, with specific services dedicated to user authentication and access control. Netflix has developed a component named Zuulx for handling these backend requests by AWS ELB, and it allows advanced features such as Security, and data safety, dynamic routing, traffic monitoring, besides ensuring zero points of failure.

The Netty server takes responsibility to handle the network protocol, web server, connection management, and proxying work. When the request hits the Netty server, it will proxy the request to the inbound filter. The inbound filter takes responsibility for authentication, routing, or decorating the request. Then it forwards the request to the endpoint filter. The endpoint filter is used to return a static response or to forward the request to the backend service (or origin as we call it). Once it receives the response from the backend service, it sends the request to the outbound filter. An outbound filter is used for zipping the content, calculating the metrics, or adding/removing custom headers.

ELB in Netflix is responsible for routing the traffic to front-end services. ELB performs a two-tier load-balancing scheme where the load is balanced over zones first and then instances. The First tier consists of basic DNS-based Round Robin Balancing. When the request lands on the first load balancing, it is balanced across one of the zones that your ELB is configured to use. The second tier is an array of load balancer instances, and it performs the Round Robin Balancing technique to distribute the request across the instances that are behind it in the same zone.

"Netflix rely AWS Load balancer (ELB) handles the request for Playback sent by the Client to the backend, running on AWS."

Modifiability:

It is evident that most of the cost of the typical software system occurs after it has been initially released. Modifiability is about change and our interest in it is to lower the cost and risk of making changes. Modifiability in Netflix's architecture is addressed through its microservices approach. This decoupling of services allows for independent development, testing, and deployment of changes.

Granularity with Microservice:

Netflix follows a granular approach to modifiability, a crucial quality attribute. This granularity ensures that modifications can be made with a fine level of control. Developers can focus on a particular microservice, making it easier to understand, adapt, and enhance that aspect of the system without the need to overhaul the entire architecture.

Increase Cohesion and Reduce Coupling:

In Netflix's architecture, "Increase Cohesion" and "Reduce Coupling" are essential tactics employed to enhance modifiability. Cohesion ensures that each microservice has a well-defined and focused purpose, allowing for independent development and modification. By reducing coupling interdependencies between microservices are minimized, reducing the risk and cost associated with system changes. This approach ensures that modifications to one microservice have minimal impact on others, making Netflix's system more flexible and cost-effective for addressing changing requirements.

Trade-offs:

It's important to recognize that the pursuit of quality attributes often involves trade-offs. Netflix's architectural choices are not without their compromises. For instance, while the microservices approach enhances modifiability and scalability, it also introduces complexities in terms of system management and communication between services. Balancing the need for high availability with the cost of redundancy and automated recovery is another trade-off that Netflix must navigate. Moreover, the pursuit of exceptional performance through scalability and elastic load balancing comes with the challenge of efficiently managing a massive and distributed system. These trade-offs are intrinsic to the architectural decisions made by Netflix to deliver high-quality streaming services while maintaining flexibility and adaptability in a dynamic digital landscape.

Conclusion:

This paper we had look up into how Netflix addresses key quality attributes, highlighting the significance of these attributes in ensuring the platform's success in the competitive digital landscape. In conclusion, the success of Netflix's architecture lies in its ability to strike a balance between the pursuit of quality attributes and the necessary trade-offs to maintain a competitive edge in the world of digital entertainment. The continued evolution of its architecture will be shaped by the ongoing quest to provide seamless streaming experiences to its global user base.

References:

Author(s): Varshneya, K. (2021, December 15).
Understanding design of microservices architecture at Netflix. TechAhead Corp.
https://www.techaheadcorp.com/blog/design-of-microservices-architecture-at-netflix/

GeeksforGeeks. (2023, August 04).
System Design of Netflix - A Complete Architecture.
https://www.geeksforgeeks.org/system-design-netflix-a-complete-architecture/

Oat, E. (2015).
Analysis of Netflix architecture and business model. Aalto University School of Science and Technology.
http://www.cse.tkk.fi/fi/opinnot

Wolff, Elberhard. (2016). Microservices: Flexible Software Architecture.

Bass, Len., Clements, Paul., & Kazman, Rick. (2021, August 03). Software Architecture in Practice (4th ed.).

Wikipedia contributors. (2018, October). Microservices. Wikipedia.
https://en.wikipedia.org/wiki/Microservices

Wikipedia contributors. (2007, January 16). Netflix. Wikipedia.
https://en.wikipedia.org/wiki/Netflix