Date: 01st April 2020
Author: Roshni Shaik
Subject: Demystifying Netflix Architecture

1 ABSTRACT

Modern advances in technology and the ease of internet access let individuals watch their preferred shows in the solace of their own household at any point in time at a very low cost. Streaming video technology has provided users with a rich set of functionalities such as instant access to the content, rewind, pause, etc. This has led many organizations in the market to innovate and develop their products in the respective area in order to provide the best user experience. One such leading provider of on-demand internet video streaming is Netflix.

This report primarily focuses on the study of Netflix to uncover its architecture and services strategy. The study identifies various factors that drove the service provider to its leading position in the streaming video market. Different components involved and the major design goals have been summarized in this report. Understanding the Netflix architecture and its performance can shed light on how to best optimize the design as well as certain tradeoffs that were done in order to achieve the main goal. The report also entails some unique strategies that the company followed while marginally changing its business model on a continuous basis, which would seldom be possible in a brick and mortar economy. Finally, the report has been concluded with potential problems and future aspects.

2 INTRODUCTION

2.1 PURPOSE

Netflix is a global subscription provider of video streaming services. It is also one of the largest content players in the market, being responsible for a significant amount of Internet traffic. It has been consistently at the top of the rankings in the Americas, accounting for 40% of the peak downstream traffic according to the Global Internet Phenomenon report 2018. The sheer volume of the Netflix traffic in the Americas is what propels it to worldwide leadership.

Netflix, whose customer base is more than 139 million streaming members worldwide, considers thoughtfully customers' opinions and satisfaction. It gives tremendous value to the subscribers by incorporating recommendations to personalize Netflix as much as possible. It provides seamless viewer experience wherein, watching their favorite show is as simple as reaching out for their device or remote, opening Netflix app and hitting play. However, what isn't simple is what goes into running Netflix, a service that streams around 164.8 million hours of videos per day to more than 139 million subscribers in over 250 countries. Building such a highly

scalable system, while delivering high definition video, with no lag, requires significant engineering effort.

As of March 2020, there are a number of competitors providing online media streaming services; with companies like Amazon, Google, Disney, Apple, Hulu, competing aggressively for market share. In addition to the big companies listed, there are many local streaming services at the country level. Despite the competition, Netflix maintained its stance as a market leader in this space due to its highly available, scalable, and fault-tolerant system architecture.

2.2 BACKGROUND

Netflix was founded in 1997 by Marc Randolph and Reed Hastings in Scotts Valley, California. It was initially set up as an online DVD rental service, at a time when DVDs were still a new format. However, the business has been adapting during its course of development, which is evident from the way Netflix has changed the DVD delivery method. Users submit their requests for DVDs online, which are delivered by post to their homes in a day or two in a reusable return envelope. Moreover, rented content can be kept for as long as desired. All these bring convenience to its end customers, which makes them more likely to remain loyal to the service. In 1999, the company adopted a subscription model, allowing viewers to pay a single monthly rate for unlimited rentals.

In 2006 it announced a "Netflix Prize" competition whose winner got 1 million dollars. The main motive of the Competition was to design a recommendation algorithm that provided higher results than Netflix's own recommendation system, CineMatch. The algorithm was expected to predict the videos that users would prefer to watch more precisely and recommend these on the Netflix page thereby yielding better customer satisfaction.

Netflix moved into streaming in 2007. It allowed subscribers to stream videos on their personal computers or mobile devices. This was executed successfully and was proven to be a profitable move as the customer base was increased by 20 million in 3 years after this change.

Before 2008, Netflix had its own datacenters and followed a monolithic architecture, i.e., a tightly coupled architecture wherein all the functions of an application were put into a single archive package to deploy and run. In a monolithic architecture, if a tiny change is made to one part of the application, all the other components of the application must be re-written. This architecture was not suitable for Netflix as the organization was continuously growing, and thus, the monolith was not reliable and highly risky as it had a greater probability of causing failure. In August 2008, Netflix had a major outage because of database corruption that prevented them from shipping DVDs to customers for three days. Following this, they have decided to move away from a single point of failure – that could only scale vertically – and move to components that could scale

horizontally and are highly available. Netflix got rid of its private data centers and moved to a public cloud, AWS in 2008, and the complete migration took 7 years.

By employing Amazon Web Services (AWS) for its cloud computing, Netflix was able to take quantum leaps forward in terms of scalability. AWS offered Netflix the ability to change their capacity within minutes, differentiate rates between components, and scale up or down unencumbered. This has helped Netflix in expanding its services to over 130 new countries.

3 COMPONENTS

On a very high level, Netflix is majorly comprised of three units that play a key role in the overall system operation: client, Content Delivery Network (CDN) and backend. Netflix provides an uninterrupted video viewing experience by seamlessly integrating all the three components.

3.1 THE CLIENT

The client can be considered as the user interface that is used to browse and play Netflix videos. Some of the examples of the client could be the application on a phone, tablet, Smart TV, streaming devices connected to TV screens, or web browser on a personal computer. Netflix has designed its system in such a way that it effectively handles the requests from every customer operating on any device.

Netflix initially used Microsoft Silverlight, which is offered as a web browser plug-in, for media streaming on the desktop web browser clients. Silverlight provides high-quality interactive video experiences across multiple formats while protecting the content. Netflix used Silverlight to download, decode and play video content on desktop web browsers. However, it has transitioned to HTML5 Technology in early 2013 to ensure playback on Chromebooks because there is no Silverlight implementation for Chrome OS. This has allowed users to watch Netflix TV shows and movies on PCs with HTML5 player, eliminating the need to install new software(Silverlight). However, users can still stream videos by explicitly installing Silverlight or using the HTML5 player.

3.2 CONTENT DELIVERY NETWORK

Netflix's content delivery network handles all the work that happens after the user hits the play button. A content delivery network (CDN) is a network of geographically distributed servers that work together with a goal of delivering the web content to the requested client as quickly, cheaply, reliably, and securely as possible. A CDN places its servers at the Internet exchange points (IXPs) to minimize latency and improve connectivity. These IXPs are the primary locations where different Internet providers connect in order to provide each other access to the traffic originating on their different networks, thereby strategically distributing traffic to the lighter

servers, resulting in reduced cost and transit time. Netflix employs a CDN to place the video content as close as possible to the users by spreading the CDN nodes across the various geographical locations.

Initially, to support on-demand streaming, Netflix built its own CDN ranging in five different locations within the United States. However, this CDN was too small and ineffective in keeping up with the increasing demand and customer base of the organization.

In 2009, Netflix employed three third-party CDNs to deliver the video content to end-users. The three CDNs were: Akamai, LimeLight, and Level-3. For the same video with the same quality level, the same encoded content is delivered from all three CDNs. With the third-party CDN services, Netflix was able to focus on other priority aspects like AWS services that are mentioned in section 3.3 that would enhance the overall user experience.

Netflix launched its own CDN, named Open Connect in 2012 which was faster and more reliable than the above mentioned third-party CDNs.

3.2.1 OPEN CONNECT

Open Connect stores Netflix videos in different locations throughout the world. When the user hits play, the video streams from Open Connect to the client's device and is then displayed on the client. The low power, high storage density servers that store the Netflix videos are called Open Connect Appliances (OCAs). They are mainly designed for high availability. Since Netflix does not operate its own data centers, OCAs are placed within the Internet Service Providers (ISPs) datacenters at common Internet exchange points. Netflix provides free OCAs for larger ISPs that have over 100,000 subscribers.

OCAs cache Netflix content within the ISP's data centers to reduce the transit costs and decrease the latency. They run the FreeBSD operating system (open-source Unix-like operating system from Berkeley Software Distribution based on Research Unix), Nginx (a web server that can be used as a load balancer and HTTP cache) and the Bird Internet routing daemon (Internet Routing Protocol for Unix-like operating systems).

3.3 THE BACKEND

The backend of Netflix comprises various Amazon Cloud Services like EC2, S3, Amazon Elastic Load Balancer, Amazon EMR, Elasticsearch, etc.; a number of open-source tools such as Cassandra, Apache Kafka, Apache Spark, etc; and Netflix's own open-sourced software Genie and Spinnaker that primarily handle all the requests and services that run before hitting the play button of the video.

3.3.1 NETFLIX ON AWS

Netflix uses AWS for a plethora of services for different aspects of its business, such as video transcoding, personalized recommendations, computing and storage needs, including databases to store videos, and many other functions that nearly use 100,000 server instances on AWS. Netflix can quickly deploy thousands of servers and terabytes of storage within minutes by leveraging AWS.

Netflix uses reliable Amazon EC2 servers for cloud computing and Amazon Simple Storage Service (S3) for storing all the video content. It also stores the frequently accessed data in EVCache for faster response times. EVCache is a distributed in-memory caching solution that runs on EC2. There are also tools like AWS elastic Transcoder used by Netflix to transform the videos into a format best suitable for the client.

Netflix uses Amazon EMR, a big data platform, for processing massive data quickly and cost-effectively at scale. EMR gives the compute engines and elasticity to run the analytics by coupling Amazon EC2 and S3 with various open-source tools such as Hadoop, Spark, Kafka, etc. Netflix has also developed its open-source software Genie on top of the Hadoop infrastructure, so as to get Hadoop to perform consistently on a large scale. These platforms influence its decisions on what content to create and promote to viewers.

Netflix uses Amazons Elastic Load Balancer (ELB) service to route traffic from the client to the services. ELB's are set up such that load is balanced across zones first, then instances.

3.3.2 DISTRIBUTED DATABASES

When migrating to the cloud, Netflix used three NoSQL distributed databases, Amazon SimpleDB, Cassandra, and HBase as key components for the storage of user-related information like profile information, billing information, etc. SimpleDB is highly durable, with writes automatically replicated across availability zones within a region.

Netflix uses HBase, a column-oriented database, for the systems based on Hadoop. HBase and Cassandra have a dynamic partitioning architecture, which makes it really easy for the system to scale horizontally by adding more servers. HBase can re-distribute load across nodes at runtime, which is particularly useful for Netflix to manage its burgeoning data volume needs with minimal response times. Further, Netflix mainly uses Cassandra because of its highly scalable, multi-regional and fault-tolerant architecture. Its high availability with tunable consistency makes it the best fit for Netflix.

3.3.4 APACHE SPARK

As a data-driven company, Netflix uses Machine learning algorithms and A/B tests to drive all of the content recommendations for its users, wherein the viewers get personalized canvases based

on their past activities. A majority of the machine learning pipelines for member personalization run on top of large managed Spark clusters.

Apache Spark enables Netflix to use a single, unified framework for ETL, feature engineering, model training, and validation. With the pipeline framework in Spark ML, each step within the Netflix recommendation pipeline (e.g. label generation, feature encoding, model training, model evaluation) is encapsulated into different units, thereby enabling modularity, composability, and testability.

3.3.4 APACHE KAFKA

Apache Kafka is a stream-processing software that provides a unified, high throughput, and low latency platform for handling real-time data. Kafka publishes the data records to a category or feed name called Topic. Producer, that generates the real-time data, writes to the topic and consumer reads from the topic. Netflix uses Kafka to route all the real-time Netflix events such as video viewing activities, UI activities, error logs, performance and diagnostic events to various sinks like S3, Cassandra, or to the stream consumers (Spark).

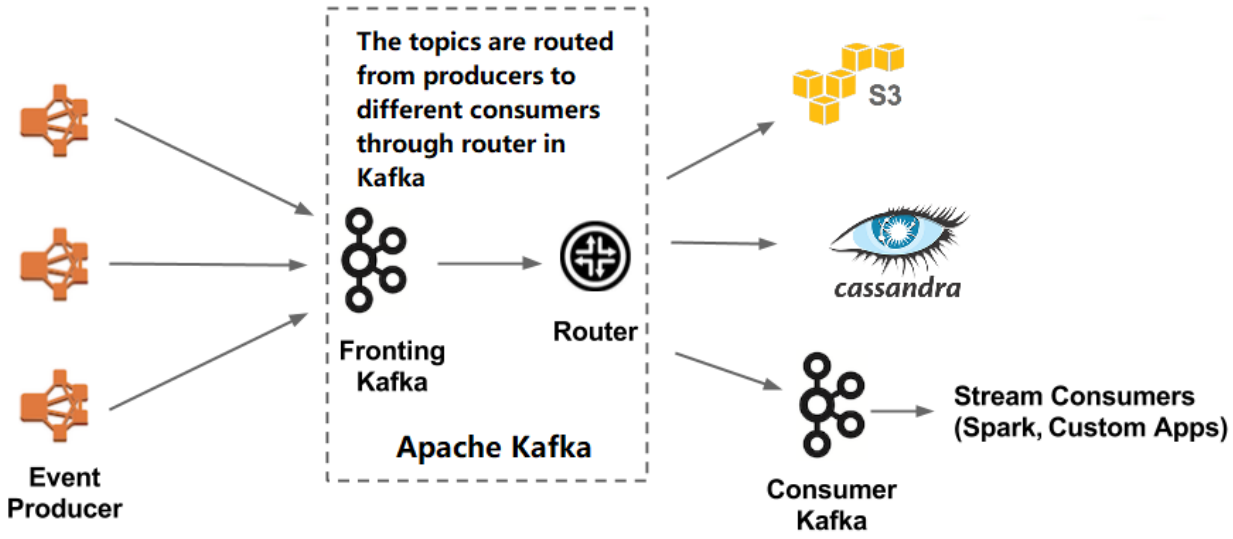

FIGURE 1 OVERALL FLOW OF EVENTS FROM KAFKA PRODUCERS TO CONSUMERS [SOURCE]

3.3.5 APACHE DRUID

Apache Druid is a high-performance real-time analytics database. Druid excels at instant data visibility, ad-hoc queries, operational analytics, and handling high concurrency. Netflix processes the real-time logs from playback devices through Apache Kafka and derives metrics (measurements) in the cloud to understand and quantify how seamlessly users' devices are

handling browsing and playback. The metrics are then processed through Kafka topics and stored in Apache Druid for real-time analytics.

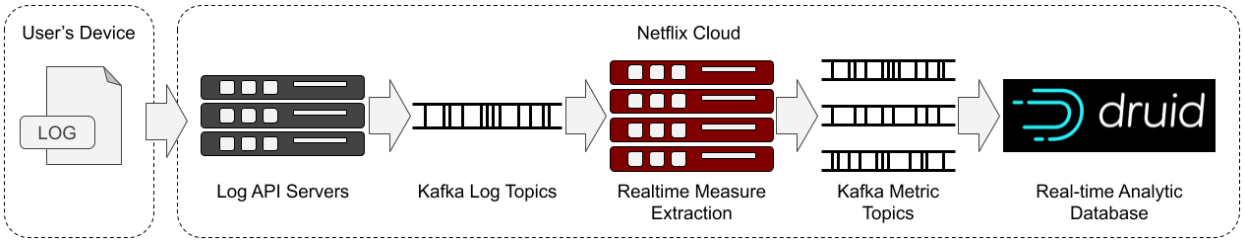

FIGURE 2 OVERALL FLOW OF EVENTS THROUGH KAFKA TO DRUID. [SOURCE]

3.3.6 SPINNAKER

The other open-source tool that Netflix built for effective Continuous Integration was Spinnaker which provides developers a room for experimentation to make small changes, test them with automated tests, and then get them live to a small subset of users. After obliterating the bugs found during the tests, the tool automatically rolls out the changes globally, once an acceptable confidence level has been attained.

4 BUILD VERSUS BUY

The components of the system that were taken off the shelf are AWS components like S3, EC2, Elastic Load Balancer, Amazon EMR, and Apache Kafka, Cassandra, Apache Druid, Apache Hadoop, and Apache Spark in the backend; and Akamai, LimeLight, and Level-3 third-party CDNs in the initial design of Netflix. The components that were specifically built for this organization are OpenConnect CDN, Genie, and Spinnaker.

AWS does the undifferentiated heavy lifting for Netflix. Since managing the scalable infrastructure doesn't add any direct value to Netflix's core business of providing quality video watching experience, the organization has employed AWS for the same.

As discussed in section 3.2 Netflix built its own CDN for higher availability and faster delivery of the video content to the users. It realized that at the scale it operates, it needed a dedicated CDN solution to maximize network efficiency. Open Connect has proven to be successful than the third party CDNs that Netflix was initially using, as the number of subscribers and the demand exponentially increased after the Open Connect implementation. Open Connect proved to be beneficial to Netflix because the organization exactly knew where the subscribers were geographically located and what movies or shows they are likely to watch. Netflix has taken these aspects into consideration and designed Open Connect specifically for its users.

5 ARCHITECTURE

In this section, we shall focus on the overall flow of how the three major components – client, backend, Open Connect are connected and seamlessly integrate with each other to provide end to end functionality.

Netflix acquires the video content from Video Source Media (Production Houses and studios) which is then processed by the Content Operations Team within Netflix. The video must be processed into a format that is most suitable for the client's device in order to provide a high-quality viewing experience. This process of transforming the video file from one format to another, such that it is tailor-made for every platform or device is called **Transcoding**.

The high-definition video obtained from the source is first validated for any missing frames, color changes, etc. that might have happened during data transmission. It is then fed into a media pipeline wherein the video is transformed at multiple stages by various applications. Since the video content is multi-terabyte sized, it is broken down into chunks of data and processed parallelly on multiple servers in EC2. The processed chunks are then assembled back and re-validated.

Since every device has a specific video format that works best for it, the entire transcoding process generates multiple encoded files suitable for every client. Additionally, it also generates files that are optimized for different network speeds, different audio formats based on the audio quality and language, and also different text files for subtitles in a number of different languages. All these files are stored in Amazon S3.

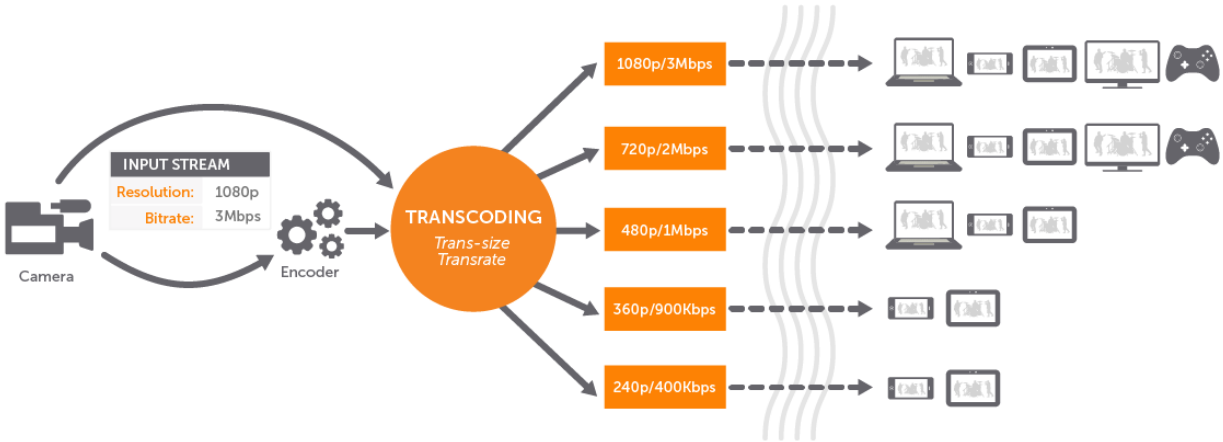

FIGURE 3 TRANSCODING PROCESS. [SOURCE]

Netflix copies these encoded files from S3 to the Open Connect Appliances (OCAs) every day during off-peak hours. This process is called **proactive caching**. Netflix predicts which videos the users are more likely to watch tomorrow at every region, and copies those predicted videos to the OCAs during off-peak hours, thereby reducing the bandwidth usage for ISPs. Higher the popularity of the video, the higher the number of OCAs the video will be copied to. AWS service is responsible to list all the videos a particular OCA should have. OCA verifies the list and the content is copied accordingly from the nearest local OCA or S3.

When the client hits play, a play request will be sent to the Playback Apps service in AWS. After the request is validated, the service picks up to ten different best possible OCA servers for the client and generates URLs for those servers. The client then tests the network connection quality to each OCA and chooses the fastest and most reliable OCA to stream the video from. If the network quality is too poor, it switches to another OCA.

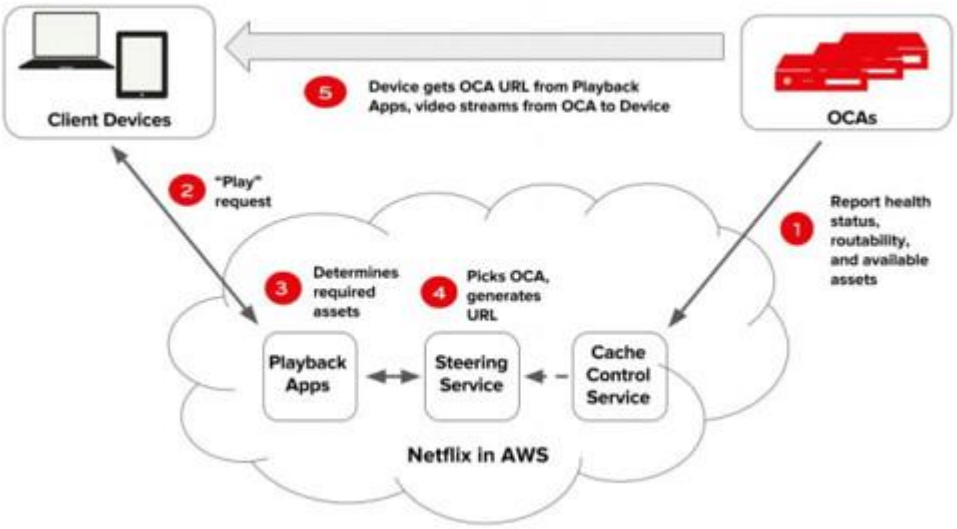

FIGURE 4 OVERALL FLOW OF THE PLAY PROCESS. [SOURCE]

In addition to the transcoding process mentioned above, there are a number of functionalities that happen on the backend on the AWS infrastructure. Netflix collects a massive amount of users data like billing information, profile information, movies or the TV shows that the users have watched, when and where they have watched, the number of times a particular video has been watched and many more. It processes all this data and stores in the distributed databases. This data is replicated across the nodes to make the system highly available and fault-tolerant.

The data is then processed by the Hadoop framework and Amazon EMR to get useful insights from the data. Netflix's recommendation system uses this data and runs various Machine Learning algorithms on Apache Spark to predict the content that the user is likely to watch. The company personalizes the content based on these insights. For example, every user views a

different header image for the same video based on the past viewing history. Much of the computation that Netflix does when running personalization machine learning algorithms is offline. In the case of online computation, real-time events are processed through Kafka and stored in Apache Druid to make use of the most recent data and are responded quickly. For instance, based on the member's current context, the system assembles a gallery of thriller movies sorted.

6 DESIGN GOALS

The key design goals for the Netflix system is to have high availability without a single point of failure, low latency, and elastic scalability.

6.1 HIGH AVAILABILITY

Since most of the Netflix services run on AWS, the system has employed multi-region architecture in order to be highly available. It operates AWS in three different regions, and within each region, it operates in three availability zones. Netflix has designed its system in such a way that if one zone or region fails, the other available zones or regions will be used to provide uninterrupted services, thus making the system very reliable. In order to make the system available during the deployments, Netflix uses the Spinnaker tool. It aims to limit the blast radius of any change to the systems, validate the change, and then roll it out to a broader set of customers. More specifically, it rolls out deployments to one AWS region at a time. This gives the system an extra layer of safety in the production deployments. It also gives the ability to quickly shift traffic away from affected customers. A detailed explanation of how the system is reliable amidst the occurrence of failures is covered in detail in section 8.

6.2 LOW LATENCY

Streaming high-quality video can take up an extensive amount of bandwidth. This ends up being expensive and can overwhelm the servers which can result in higher response times. Netflix's content delivery network, Open Connect aims at minimizing this latency by installing OCAs at IXPs and proactive caching. Open connect places the video as close as possible to users by spreading OCAs throughout the world. This results in reduced transit time and the client will be served the video from the closest OCA. It has specifically designed the Internet Exchange points where the OCAs have to be placed based on its subscribers' geographical data.

The system also uses Amazon Elastic Load balancer for effective load balancing techniques to distribute the load from a heavily loaded node to a lighter one. This enhances parallel processing and results in lower latency. By distributing all the functionalities into independent microservices, the latency will be minimized as the servers wouldn't be heavily loaded with a single huge process.

6.3 ELASTIC SCALABILITY

The system's auto-scaling clusters monitor the demand and automatically scale up or shrink the servers to maintain steady, predictable performance at the lowest possible cost. The detailed description of the system's scalability is provided in section 9.

7 TRADEOFFS

As discussed in section 6, Netflix aims for a highly available, fault-tolerant system ensuring maximum customer satisfaction. Thus, Netflix has employed a NoSQL distributed database, Cassandra for data persistence and faster querying. However, there exists a trade-off between high availability and consistency in such distributed systems. This has been theoretically claimed by CAP(Consistency, Availability, and Partition) theorem that states that in the case of a network partition, the system must choose between consistency and availability.

Cassandra trades off consistency in order to achieve high availability and low latency. A Cassandra consistency level is defined as the minimum number of Cassandra nodes that must acknowledge before a read/write operation. There are multiple Cassandra levels that can be set ranging from 'ANY' to 'ALL' with 'ANY' being the least consistent, and 'ALL' being highly consistent as it confirms with all the nodes before a read/write operation. When a lower consistency level is set, the system offers very high availability as it will allow the user to read and write data even if one node is up. However, in this case, there is a high probability that data is inconsistent.

Consistency and latency have an inverse relationship with each other. If the system is required to be highly consistent, the system must wait for the response from all the replicas which in turn increases the request latency. For a consistent system, the replication is done synchronously which means that the master node waits until all updates have been made to the replicas before the update is committed. However, Netflix employs asynchronous replication to aim low latency, so that the system can read from any of the available replicas. This results in inconsistency, since different replicas stored at different data centers may have different versions of the data. However, Cassandra guarantees eventual consistency.

8 RESILIENCE TO FAILURE

Netflix operates over 1000 microservices on the cloud. Since no single component can guarantee 100% uptime, there is a high probability that it can fail. To provide high service availability amidst the failures, Netflix induces controlled failures on systems regularly to discover vulnerabilities and resolves them. Netflix has created a suite of tools, called the Simian Army, which aims at improving the resiliency of the cloud environment.

In order to make the system resilient to failure, Netflix employed exception handling, auto-scaling clusters, redundancy, fault tolerance and isolation, and fall-backs and degraded experiences. The system can easily accommodate a stateless node (Node without database or cache) failure by auto-replacement with Auto-scaling. Amazon auto-scaling clusters, spin up a replicated node from Amazon S3 to scale up and shrink with traffic, which will be discussed in detail in section 9. The important factor is not just the scaling but the replacement dimension; if an instance is removed on purpose, it gets replaced with an active instance with redundant data and the traffic gets redirected to the new node without affecting the overall functionality.

Netflix handles the failure of stateful service (that holds data, like databases, cache, etc.) by redundancy; it uses EVCache architecture, wherein every time data write happens, multiple copies are written to multiple nodes on different availability zones so that it spreads the data across the network partitions to accommodate availability zone outages as well. When there is an availability zone outage, Netflix uses Elastic Load Balancer to quickly move all the traffic to the active availability zones and the auto-scaling cluster scales up the instances based on the traffic. To handle the excessive load, Netflix partitioned the workload into the batch and real-time and implemented request-level caching to make the successive request calls to be inexpensive.

There can be many failures associated with intra-service requests within microservices, like network latency, congestion, hardware failures, etc. which would result in a time-out. When the requests are dependent, this could lead to cascading the failure from one service to rest other services causing the entire system to fail. Netflix created Hystrix to deal with such scenarios. Hystrix's isolated thread pools and circuits help in isolating the blast radius for any given failure by providing exception handling and giving some fall-back options to the user while waiting for the failed service to recover instead of making repeated calls to the failed service. The fall-back options would at least allow the user to use other services.

Netflix ensures that the data is persistent in the case of the network partition by employing distributed Cassandra database, where the writes would still be successful to the active nodes, as it chooses high availability over consistency as discussed in detail in section 7.

To obliterate any software bugs induced by the developers' change requests, Netflix uses the Spinnaker tool as described in section 3.3.6.

9 SCALABILITY

The number of paid subscribers for Netflix has seen very good growth over the years. So, from the graph below, it is clear that overall demand has gone up over the years. Thus, scalability is of utmost importance to Netflix. It is not just adequate for the system to satisfy the current requirements, but also be scalable enough to meet the increasing future demands.

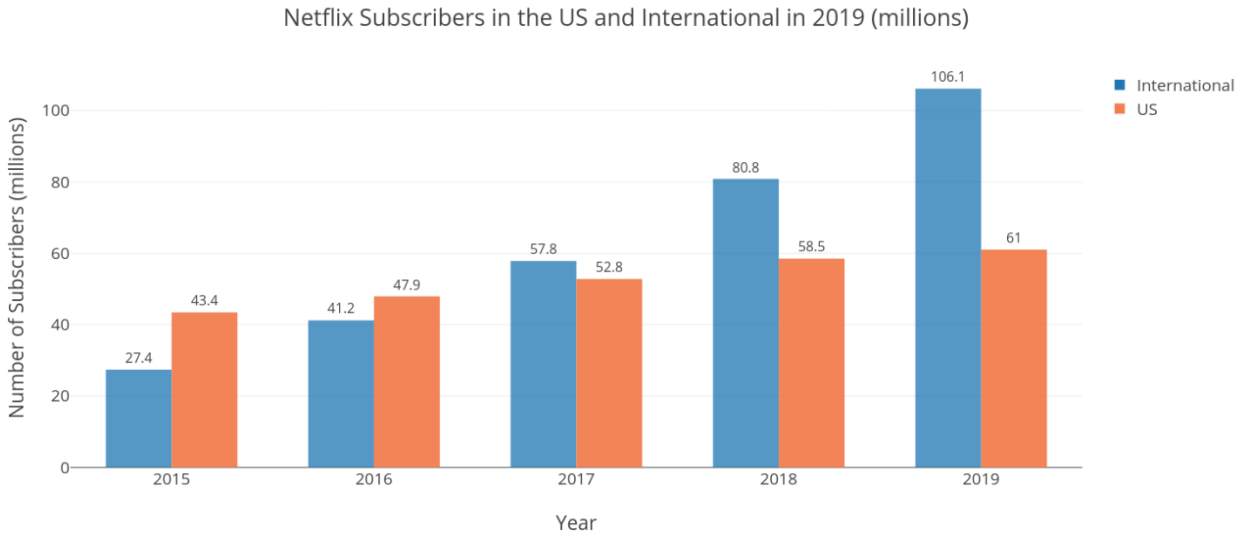

FIGURE 5 NETFLIX SUBSCRIBERS IN THE US AND OTHER COUNTRIES IN 2019

Some of the key solutions that Netflix has executed to achieve scalability are as follows:

- **Amazon Autoscaling Group:** This was the most innovative feature offered by AWS, to automatically scale depending upon the load metrics (metrics can be customized depending upon the need). New instances could be quickly added to the group and removed from the group based on the average CPU usage, the average number of requests coming in, etc. It has also proactively employed predictive mechanisms based on the trends to predict the times when the system might have to scale up. There are many critical tasks that require very low latency. Load balancers in combination with the autoscaling group can quickly assess the turnaround time for requests and add more instances to speed up tasks to reduce latency.

  Load Balancers are primarily used to distribute traffic uniformly across multiple servers or multiple endpoints. Netflix at a high-level implements load balancer in 2 levels, 1st level works at a DNS (Domain Name System) level distributing the traffic across multiple zones, whereas the 2nd level works at the application level. Each load balancer at the 1st level is connected to the load balancer at the 2nd level.

- **Microservices**: Netflix has over 1000 microservices running to perform various business logic. These are service-oriented mini-architectures responsible for carrying out very specific tasks like fetching the popular videos, computing recommendations, enable searching of titles, etc. The most important characteristic of microservices is loose

coupling, which enables to design these services with very little dependence between each other. Each service can, therefore, be scaled based on how critical it is. It is very common for the application to have some features being used more often than the others.

- **Distributed Databases:** By employing NoSQL distributed databases for storage, Netflix ensures that the system is scalable. These distributed databases have a highly scalable architecture that can be easily scaled by the simple addition of extra nodes in a linear fashion resulting in the throughput increase without any significant downtime.

9.1 BOTTLENECKS

One of the bottlenecks that the system faces is that although autoscaling instances acquire scalability, it can introduce varied response times. When more servers are added to the system than a particular threshold, response times might oscillate between the demand going up and systems scaling to catch up.

10 NEXT STEPS

Although Netflix has employed a fault-tolerant architecture, complex distributed systems can experience failure. They fail more and in different ways as they scale and evolve over time. Moreover, testing distributed systems is challenging due to massive changing datasets, web-scale traffic, complex workflows, asynchronous requests, 3rd party services, etc. There are a huge number of scenarios that could result in failure of microservices and the designers have to constantly test the system's ability to actually survive these occasional failures.

Furthermore, as mentioned in section 9, when the user demand is doubled at such a scale, the system will have a shortcoming of taking greater response times as the system will not be pre-scaled. It would take additional time to scale up the system in order to meet the sudden increase in demand. The designers could use time series analysis to identify the time necessary to ramp up servers before an expected overload. They could then use this information to pre-scale the servers when there is an expected overload.

Netflix’s proactive caching mechanism stores popular videos on OCAs for faster access to the video content. It also uses machine learning algorithms to predict the videos to be stored on OCAs. However, it gives greater priority to the popularity of the video and saves multiple copies of the video based on how popular the video is. On the other hand, OCA being a physical setup the disadvantage is having fixed cost, unlike services on the cloud and, additionally, the popularity of videos fades away quickly. More than one-tenth of Hulu top videos are replaced by

others every hour, according to studies by H. Li et al [8]. Thus, an optimal update algorithm has to be considered, in accordance with which video content needs to be updated.

Another drawback that Netflix faces is that, by allowing shared logins, Netflix runs into an issue with personalization. The recommendations can only be specific to the customers when they choose to create separate profiles for each user. Since personalization is one of the major components in Netflix's success, the designers can address this issue by keeping track of frequently used devices, location, user's interests. If the current user's context varies widely from the previously tracked data, Netflix could give a pop-up with the suggestion to create a different profile for the new user.

11 CONCLUSION

Netflix's proactive planning and investment in highly-scalable solutions have enabled the organization to handle the benchmark beating growths in its customer base. The company strives to enhance customer satisfaction, by providing favorable recommendations to the users and also enriching its video content database in parallel. By seamlessly integrating the client, Open Connect CDN, and the backend while ensuring high availability, low latency and elastic scalability of the system, Netflix has managed to be the dominant player in the market. The foresight of the organization to build its own Content Delivery Network to facilitate faster streaming is remarkable. Despite the consistency trade-off that the system makes in exchange for high availability, the overall service quality is not affected. The areas that the designers can focus on have been summarized.

12 REFERENCES

[1] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In INFOCOM, 2012 Proceedings IEEE, pages 1620 – 1628, March 2012.

[2] J. Ciancutti. Four Reasons We Choose Amazon's Cloud as Our Computing Platform. Technical report, December 2010. http://techblog.netflix.com/2010/12/ four-reasons-we-choose-amazons-cloud-as.html. Resource last accessed 10.03.2013.

[3] A. Cockroft, C. Hicks, and G. Orzell, "Lessons Netflix Learned from the AWS Outage," Netflix Tech blog, 2011.

[4] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network characteristics of video streaming traffic. In Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies, CoNEXT '11 pages 25:1–25:12, New York, NY, USA, 2011. ACM.

[5] A. Cockroft, “Netflix Cloud Architecture,” Velocity conference, 2011. [10] “Amazon Web Services,” http://aws.amazon.com.

[6] Osur, Laura, "Netflix and the Development of the Internet Television Network" (2016). Dissertations - ALL. 448.

[7] A. Vinay, P. Saxena, and T. Anitha. An efficient video streaming architecture for Video-on-Demand systems. In Signal and Image Processing (ICSIP), 2010 International Conference, pages 102 –107, December 2010.

[8] Mandal, G., Diroma, F., Jain, R. (2017). Netflix: An In-Depth Study of their Proactive & Adaptive Strategies to Drive Growth and Deal with Issues of Net-Neutrality & Digital Equity. IRA-International Journal of Management & Social Sciences (ISSN 2455-2267), 8(2), 152-161. doi:http://dx.doi.org/10.21013/jmss.v8.n2.p3

[9] Yury Izrailevsky, “NoSQL at Netflix,” Netflix Tech blog, 2011.

[10] Yoni Heisler, https://bgr.com/2019/04/02/netflix-binge-worthy-hours-spent-watching-every-day/, April 2019.