# **Доклад по Netflix**

## **История изменения архитектуры и проблемы, с которыми сталкивались**

На сегодняшний день Netflix — один из самых крупных в мире потоковых медиасервисов по требованию, эксплуатирующий свои онлайн-сервисы в облаке. Компания была основана в 1997 году в Скоттс-Валли, штат Калифорния, Ридом Хастингсом (Reed Hastings) и Марком Рэндольфом (Marc Randolph). Изначально Netflix предоставляла интернет-сервис по прокату DVD, позволявший клиентам вносить фиксированную ежемесячную плату за подписку на неограниченные прокатные видео без дополнительной платы. Клиенты получали DVD по почте после выбора их изображения из списка и помещения в очередь с помощью сайта Netflix.

В начале 2009 года стало ясно, что спрос резко возрастает. Известен такой факт, что на презентации в рамках конференции AWS reflnvent, состоявшейся в 2013 году, Юрий Израилевский (Yury Izrailevsky), вице-президент по проектированию облачных вычислений и платформ, сказал, что спрос с 2009 года возрос в 100 раз. «Мы не смогли бы справиться с масштабированием наших сервисов, если бы пользовались своим внутренним решением», — сказал Израилевский.

Как только Netflix начала размещать свои приложения на Amazon Web Services, сотрудники стали поддерживать переход к новой архитектуре, сконцентрированной на горизонтальной масштабируемости на всех уровнях стека ПО.

Джон Цианкутти, занимавший тогда должность вице-президента технологий персонализации компании Netflix, в конце 2010 года выложил в блог сообщение, что «облачная среда идеально подошла для горизонтально масштабируемых архитектур. Нам не нужно гадать на месяцы вперед, какими будут наши потребности в оборудовании, хранилище и сетевой аппаратуре. Мы можем практически мгновенно программным способом получить доступ к большему объему этих ресурсов из общих пулов в Amazon Web Services».

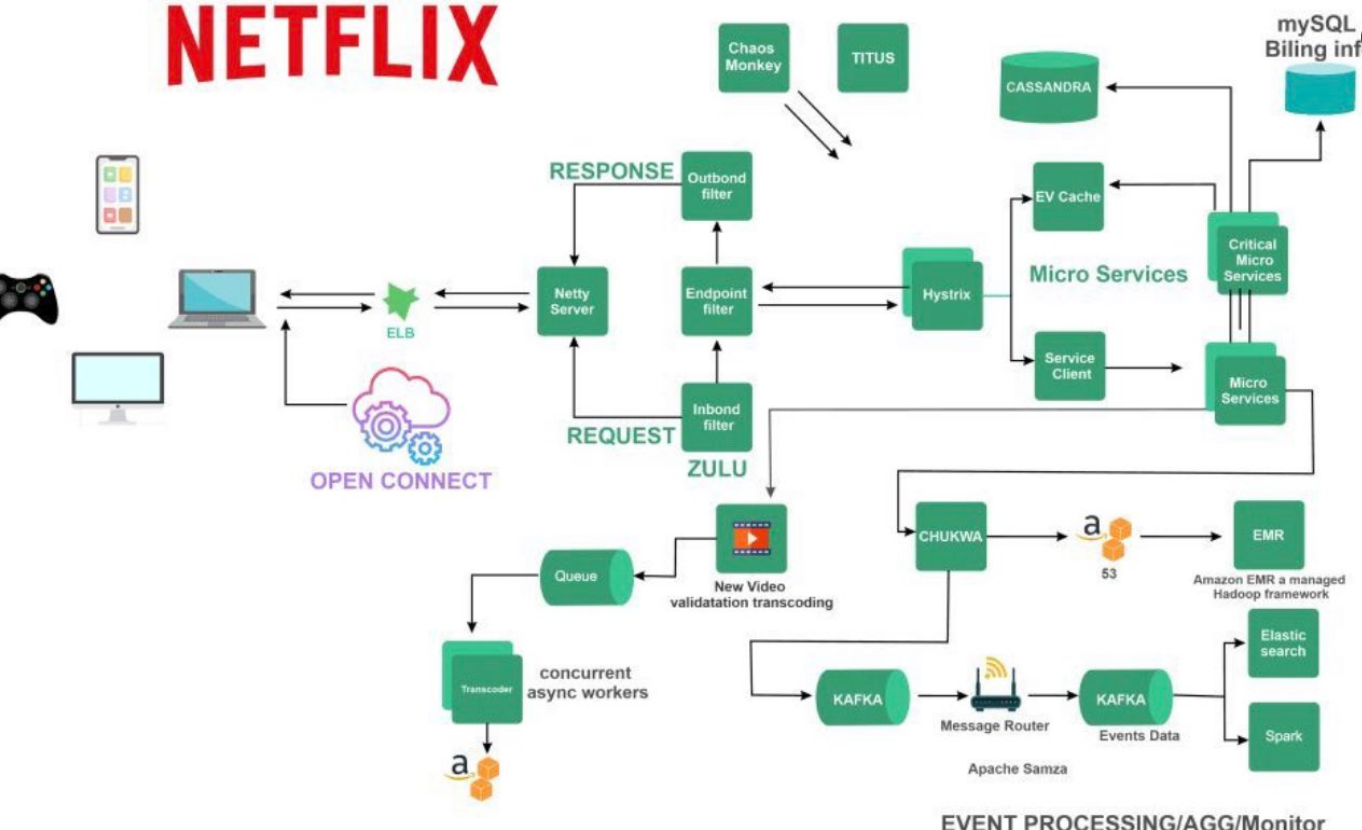
Также при переходе на облачное хранилище, Netflix перешел от монолита к микросервисной архитектуре сервиса. Специалисты компании ссылаются на два основных преимущества перехода от монолита к архитектуре распределенных систем в облаке: адаптивность и надежность.

До перехода на облачную среду архитектура Netflix включала одно монолитное приложение виртуальной машины Java (Java Virtual Machine, JVM). Хотя развертывание одного большого приложения приносило массу преимуществ, основной недостаток заключался в том, что командам создателей приходилось снижать темпы работы из-за необходимости координировать вносимые ими изменения.

## **Архитектурная диаграмма**

Основу приложения Netflix составляют 3 компонента:

1. Клиент: Устройство (пользовательский интерфейс) для просмотра и воспроизведения видео Netflix.
2. OC (Open connect) или CDN Netflix: CDN — это сеть серверов, распределенных по разным географическим точкам; Open Connect — это собственная глобальная CDN Netflix (сеть доставки контента). Она обрабатывает все, что связано с потоковой передачей видео в разных точках мира. Как только вы нажмете кнопку воспроизведения, видеопоток из этого компонента отобразится на вашем устройстве.
3. Серверная часть (база данных): В этой части обрабатывается все, что не связано с потоковой видеопередачей, например, ввод нового контента, его обработка и распространение на серверах, расположенных в разных частях мира, управление сетевым трафиком. Большинство этих процессов выполняются веб-службами Amazon.



## **Перечень технологий, используемых на данный момент**

1. **ELB -** упругий балансировщик нагрузки
2. **ZUUL -** сервис шлюзов
3. **Библиотека Hystrix**
4. **Микросервисная архитектура**
5. **Кэш EV**
6. **MySQL и Cassandra**
7. **Kafka и Apache Chukwa**
8. **Эластичный поиск**
9. **Apache Spark**

## **Краткое описание технологий, используемых на проекте**

1. **ELB** (Elastic Load Balancer; упругий балансировщик нагрузки) в Netflix отвечает за маршрутизацию трафика на интерфейсные сервисы. Сначала нагрузка распределяется по зонам, а затем по экземплярам (серверам).
2. **ZUUL** — это сервис шлюзов, который обеспечивает динамическую маршрутизацию, мониторинг, отказоустойчивость и безопасность системы. Он отвечает за беспрепятственное прохождение трафика с учетом параметров запросов, URL-адресов, путевых имен.
3. **Библиотека Hystrix** помогает контролировать взаимодействие между распределенными службами путем внесения в систему логики допуска задержек и отказоустойчивости. Hystrix делает это, изолируя точки доступа между службами, удаленной системой и сторонними библиотеками.
4. **Микросервисная архитектура Netflix.** Архитектурный стиль Netflix создан как коллекция служб. Такая микросервисная архитектура обеспечивает поддержку всех API, необходимых для мобильных и веб-приложений. Поступая в конечную точку, запрос вызывает другие микросервисы для получения необходимых данных. Эти микросервисы также могут запрашивать данные у различных микросервисов. После этого полный ответ на запрос API отправляется обратно в конечную точку.
5. **Кэш EV.** В большинстве приложений обычно используется определенный объем данных. Для более быстрого ответа эти данные могут быть кэшированы во многих конечных точках и могут быть извлечены из кэша без участия исходного сервера. Это снижает нагрузку с исходного сервера, но проблема в том, что если узел выходит из строя, весь кэш выходит из строя, и это может повлиять на производительность приложения. Чтобы решить эту проблему, Netflix создала свой собственный слой кэширования, называемый Кэш EV. Основанный на Memcached. Кэш EV, по сути, является оболочкой Memcached.

Netflix развернул множество кластеров в ряде экземпляров AWS EC2. В этих кластерах много узлов Memcached, а также клиентов кэша. Данные распределяются по кластеру в пределах одной зоны; несколько копий кэша хранятся в разделенных узлах. Каждый раз, когда производится запись из клиента, все узлы во всех кластерах обновляются, но чтение, которое выполняется из кэша, отправляется только в ближайший кластер (не во все кластеры и узлы) и его узлы.

1. **База данных.** Netflix использует для разных целей две базы данных — MySQL(RDBMS) и Cassandra(NoSQL)

Netflix сохраняет в MySQL данные о биллинге, пользователях и транзакциях, поскольку эта библиотека соответствует требованиям ACID. Netflix создала для MySQL настройку “ведущий-ведущий” и развернула ее на крупных экземплярах Amazon EC2 с использованием InnoDB.

Настройка выполняется по “Протоколу синхронной репликации”, в соответствии с которым запись, произведенная на основном ведущем узле, реплицируется на другой ведущий узел. Подтверждающее уведомление отправляется только в случае подтверждения записи, произведенной как на основном, так и на удаленном ведущих узлах. Это обеспечивает высокую доступность данных.

Cassandra — это база данных NoSQL, которая может обрабатывать большие объемы информации, а также интенсивно выполнять запись и чтение. Когда Netflix начала приобретать больше пользователей, данные истории просмотров каждого участника также начали увеличиваться. В результате возросло общее количество данных историй просмотров, и Netflix стало сложно обрабатывать такой огромный объем данных. Netflix изменила масштаб хранилища данных историй просмотров, преследуя две основные цели:

- уменьшение объема памяти;

- достижение стабильной производительности чтения/записи по мере увеличения количества просмотров на одного участника (соотношение записи и чтения данных историй просмотров в Cassandra составляет примерно 9:1).

1. **Обработка данных в Netflix с использованием Kafka и Apache Chukwa.** Netflix использует Kafka и Apache Chukwa для приема данных, которые создаются в другой части системы. Apache Chukwa — это система с открытым исходным кодом для сбора данных в виде журналов или событий из распределенной системы. Система поставляется платформой Hadoop с функциями масштабируемости и устойчивости. Кроме того, она включает в себя множество мощных и гибких инструментов для отображения, мониторинга и анализа результатов.
2. **Эластичный поиск.** В последние годы мы наблюдаем значительный рост использования Elasticsearch в Netflix. Netflix запускает примерно 150 кластеров эластичного поиска и 3500 хостов с экземплярами. Netflix использует эластичный поиск для визуализации данных, поддержки клиентов и обнаружения некоторых ошибок в системе. Netflix использует эластичный поиск для визуализации данных, поддержки клиентов и обнаружения некоторых ошибок в системе. Например, если клиент не может воспроизвести видео, работник техподдержки решит эту проблему. Команда Playback перейдет к эластичному поиску и найдет пользовательское устройство. Получив полную информацию и проанализировав события, относящиеся к этому конкретному пользователю, специалисты определят, что вызвало ошибку в видеопотоке.
3. **Apache Spark для рекомендации фильмов.** Netflix использует Apache Spark и машинное обучение для рекомендаций фильмов. Разберемся, как это работает, на примере. Когда вы загружаете первую страницу, вы видите несколько строк, состоящих из названий различных фильмов. Netflix персонализирует эти данные и решает, какие строки с какими фильмами должны отображаться конкретному пользователю. Эта информация основана на данных пользовательских историй и предпочтений.

## **Заключение**

Именно так Netflix, YouTube и аналогичные приложения для потоковой передачи видео отслеживают и загружают свой контент, чтобы показать его миллионам пользователей. Некоторые из этих приложений обладают какими-то специфическими функциями, которые делают их уникальными, но основной функционал подчиняется схожим алгоритмам.

## **Список литературы**

- <https://oracle-patches.com/net-cloud/istoriya-uspekha-netflix-i-oblachnaya-java-platforma>

- <https://twitter.com/AmigosCode/status/1675830836489338882/photo/1>

- <https://bool.dev/blog/detail/netflix-architecture>

- <https://medium.com/nuances-of-programming/%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-netflix-2dfc62417223>