



The Netflix Tech Blog

Tuesday, February 10, 2015

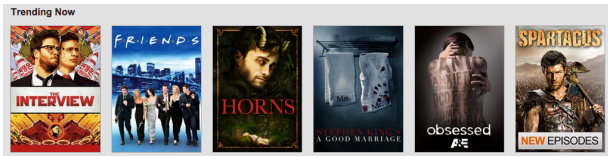
What's trending on Netflix?

By [Prasanna Padmanabhan](#), [Kedar Sadekar](#), [Gopal Krishnan](#)

Every day, millions of members across the globe, from thousands of devices, visit Netflix and generate millions of viewing hours. The majority of these viewing hours are generated through the videos that are recommended by our recommender systems. We continue to invest in improving our recommender systems that aid our members to discover and watch the specific content they love. We are constantly trying to improve the quality of the recommendations using the sound foundation of [AB testing](#).

On that front, we recently AB tested introducing a new row of videos on the home screen called “Trending Now”, which shows the videos that are trending in Netflix infused with some personalization for our members. This post explains how we built the backend infrastructure that powers the Trending Now row.

Traditionally, we pre-compute many of the recommendations for our members based on a combination of explicit signals (viewing history, ratings, My List, etc.) and other implicit signals (scroll activity, navigation, etc.) within Netflix, in [near-line](#) fashion. However, the Trending Now row is computed as events happen in real time. This allows us to not only personalize this row based on the context like time of day and day of week, but also react to sudden changes in collective interests of members, due to a real-world events such as Oscars or Halloween.



Data Collection

There are primarily two data streams that are used to determine the trending videos:

- Play events: Videos that are played by our member
- Impression events: Videos seen by our members in their view port

Netflix embraces Service Oriented Architecture (SOA) composed of many small fine grained services that do one thing and one thing well. In that vein, [Viewing History Service](#) captures all the videos that are played by our members. [Beacon](#) is another service that captures all impression events and user activities within Netflix. The requirement of computing recommendations in real time, presents us with an exciting challenge to make our data collection/processing pipeline a low latency, highly scalable and resilient system. We chose [Kafka](#), a distributed messaging system, for our data pipeline as it has [proven](#) to handle millions of events per second. All the data collected by the Viewing History and Beacon services are sent to Kafka.

Links

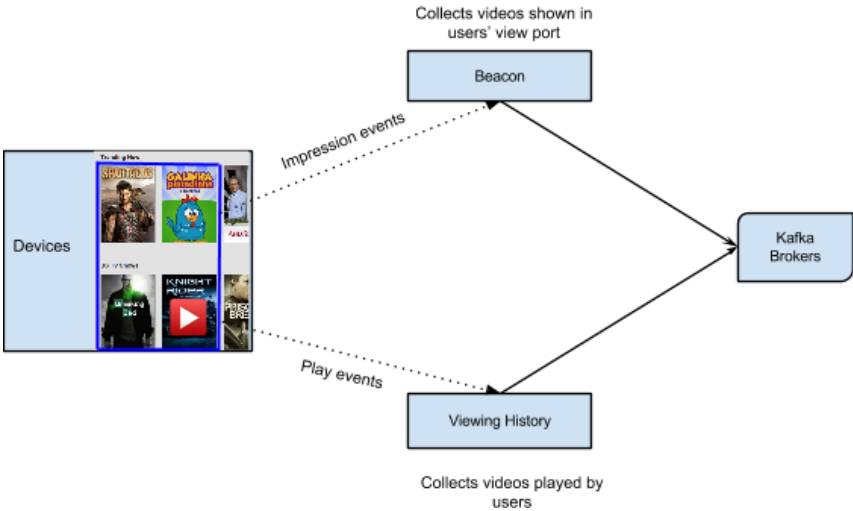
- [Netflix US & Canada Blog](#)
- [Netflix America Latina Blog](#)
- [Netflix Brasil Blog](#)
- [Netflix Benelux Blog](#)
- [Netflix DACH Blog](#)
- [Netflix France Blog](#)
- [Netflix Nordics Blog](#)
- [Netflix UK & Ireland Blog](#)
- [Netflix ISP Speed Index](#)
- [Open positions at Netflix](#)
- [Netflix Website](#)
- [Facebook Netflix Page](#)
- [Netflix UI Engineering](#)
- [RSS Feed](#)

About the Netflix Tech Blog

This is a Netflix blog focused on technology and technology issues. We'll share our perspectives, decisions and challenges regarding the software we build and use to create the Netflix service.

Blog Archive

- 2016 (24)
- ▼ 2015 (50)
 - December (7)
 - November (5)
 - October (5)
 - September (6)
 - August (6)
 - July (3)
 - June (2)
 - May (2)
 - April (3)
 - March (3)
 - ▼ February (5)
 - [RAD - Outlier Detection on Big Data](#)
 - [A Microscope on Microservices](#)
 - [What's trending on Netflix?](#)
 - [Nicobar: Dynamic Scripting Library for Java](#)
 - [SPS : the Pulse of Netflix Streaming](#)
- January (3)
- 2014 (37)

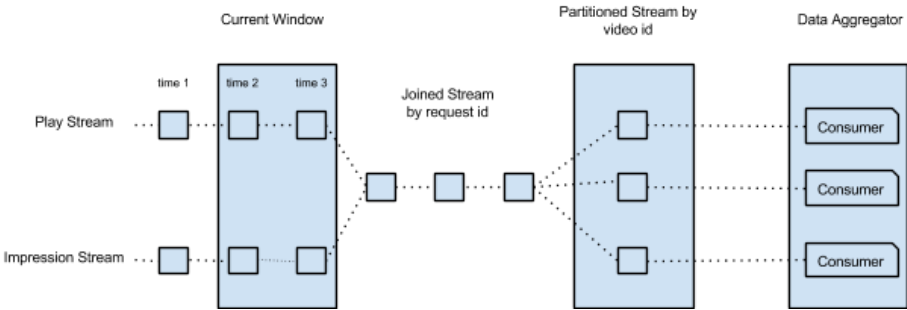


Data Processing

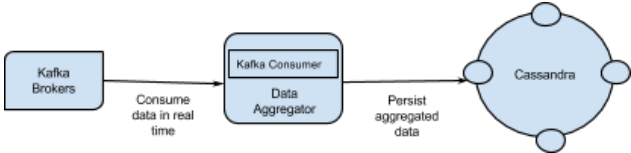
We built a custom stream processor that consumes the play and impressions events from Kafka and computes the following aggregated data:

- Play popularity: How many times is a video played
- Take rate: Fraction of play events over impression events for a given video

The first step in the data processing layer is to join the play and impression streams. We join them by request id, which is a unique identifier used to tie the front end calls to the backend service calls. With this join, all the plays and impressions events are grouped together for a given request id as illustrated in the figure below.



This joined stream is then partitioned by video id, for all the plays and impression events of a given video to be processed at the same consumer instance. This way, each consumer will be able to atomically calculate the total number of plays and impressions data for every video. The aggregated play popularity and take rate data are persisted into Cassandra, as shown in the figure below.



Real Time Data Monitoring

Given the importance of the data quality to the recommendation system and the user experience, we continuously do canary analysis for the event streams. This involves simple validations such as the presence of mandatory attributes within an event to more complex validations such as finding the absence of an event within a time window. With appropriate alerting in place, within minutes of every UI push, we are able to catch any data regressions with this real time stream

- 2013 (52)
- 2012 (37)
- 2011 (17)
- 2010 (8)

Labels

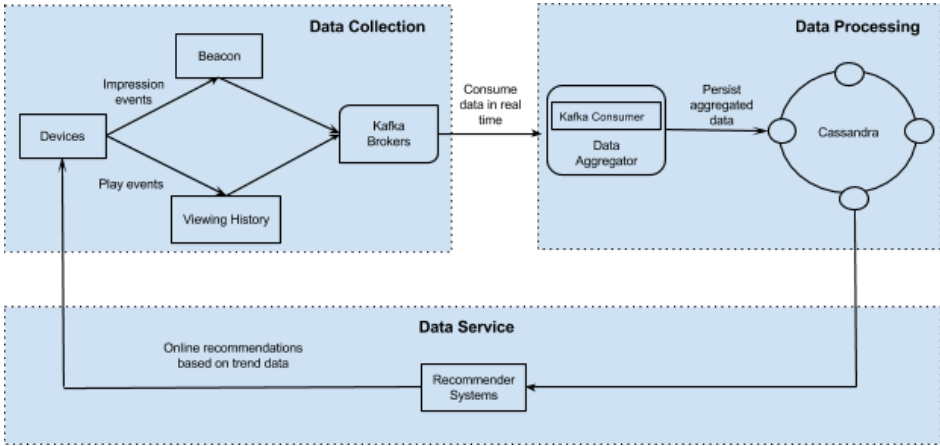
- "cloud architecture" (3)
- A/B Testing (2)
- accelerated compositing (2)
- adwords (1)
- Aegisthus (1)
- algorithms (4)
- aminator (2)
- analytics (5)
- Android (3)
- angular (1)
- animations (1)
- api (16)
- appender (1)
- AppsAndDevices (3)
- Archaius (2)
- architectural design (1)
- architecture (1)
- artwork (1)
- Asgard (1)
- Astyanax (4)
- authentication (1)
- automation (2)
- autoscaling (3)
- availability (4)
- AWS (31)
- bake (1)
- benchmark (2)
- big data (11)
- billing (1)
- Blitz4j (1)
- build (4)
- Cable (1)
- caching (5)
- Cassandra (14)
- chaos engineering (1)
- chaos monkey (5)
- chukwa (1)
- ci (1)
- classloaders (1)
- Clojure (1)
- cloud (26)
- cloud architecture (17)
- cloud prize (3)

monitoring.

It is imperative that the Kafka consumers are able to keep up with the incoming load into Kafka. Processing an event that was minutes old will neither provide a real trending effect nor help find data regression issues soon.

Bringing it all together

On a live user request, the aggregated play popularity and take rate data along with other explicit signals such as members' viewing history and past ratings are used to compute a personalized Trending now row. The following figure shows the end to end infrastructure for building Trending Now row.



Netflix has a data-driven culture that is key to our success. With billions of member viewing events and tens of millions of categorical preferences, we have endless opportunities to improve our recommendations even further.

We are in the midst of replacing our custom stream processor with [Spark Streaming](#). Stay tuned for an upcoming tech blog on our resiliency testing on Spark Streaming.

If you would like to join us in tackling these kinds of challenges, we are [hiring](#)!

Posted by [Prasanna Padmanabhan](#) at 12:38 PM

+1 Recommend this on Google

Labels: [big data](#), [recommendations](#), [scalability](#), [stream processing](#)

[Newer Post](#)

[Home](#)

[Older Post](#)

CO2 (1)
collection (1)
complex event processing (1)
computer vision (2)
concurrency (1)
configuration (2)
configuration management (2)
conformity monkey (1)
content delivery (1)
content metadata (1)
content platform engineering (2)
content quality (1)
continuous delivery (4)
coordination (2)
cost management (1)
crypto (1)
Cryptography (2)
CSS (2)
CUDA (1)
dart (1)
data (1)
data migration (1)
data pipeline (5)
data science (8)
data visualization (1)
database (5)
DataStax (2)
deadlock (1)
deep learning (1)
Denominator (2)
dependency injection (1)
device (3)
device proliferation (1)
devops (2)
distributed (11)
DNS (1)
Docker (1)
Dockerhub (1)
DSL (1)
Dyn (1)
DynECT (1)
efficiency (2)
Elastic Load Balancer (1)
elasticsearch (3)
ELB (1)
EMR (2)
encoding (4)
energy (1)
eucalyptus (1)

eureka (2)
evcache (2)
Experimentation (2)
failover (2)
falcor (2)
fault-tolerance (12)
flamegraphs (2)
Flow (1)
footprint (1)
FRP (1)
functional reactive (1)
garbage (1)
garbage collection (1)
gc (1)
Genie (4)
git (1)
Governator (1)
GPU (2)
gradle (1)
green (1)
Groovy (1)
Hack Day (3)
Hadoop (12)
HBase (1)
high volume (4)
high volume distributed systems (11)
Hive (2)
HTML5 (8)
https (1)
Hystrix (5)
IBM (1)
ice (1)
images (1)
IMF (3)
IMSC (2)
infrastructure (1)
initialization (1)
innovation (3)
insights (1)
inter process communication (1)
Interoperable Master Format (3)
iOS (1)
Ipv6 (2)
isolation (1)
ISP (1)
java (5)
JavaScript (19)
jclouds (1)
jenkins (2)

kafka (4)
Karyon (2)
keystone (2)
lifecycle (1)
linux (2)
lipstick (2)
load balancing (3)
localization (1)
localization platform engineering (1)
locking (1)
locks (1)
log4j (1)
logging (2)
machine learning (6)
Map-Reduce (1)
media pipeline (1)
meetup (3)
memcache (2)
memcached (1)
message security layer (1)
Mobile (3)
modules (1)
monitoring (1)
msl (1)
nebula (1)
negative keywords (1)
Netflix (17)
Netflix API (8)
netflix graph (1)
Netflix OSS (13)
NetflixOSS (12)
neural networks (1)
node.js (4)
NoSQL (5)
observability (1)
Open source (10)
operational excellence (1)
operational insight (2)
operational visibility (1)
optimization (2)
OSS (3)
outage (1)
page generation (1)
payments (1)
Paypal (1)
performance (24)
personalization (6)
phone (1)
Pig (4)

pipeline (1)
pki (1)
Playback (2)
prediction (2)
predictive modeling (3)
Presto (2)
prize (1)
prs (1)
pubsub (1)
pytheas (1)
python (3)
Quality (1)
quality control (1)
rca (2)
React (3)
Reactive Programming (2)
real-time insights (2)
real-time streaming (3)
Recipe (1)
recommendations (9)
Redis (3)
reinvent (2)
reliability (7)
remote procedure calls (1)
renewable (1)
research (2)
resiliency (7)
REST (2)
Ribbon (2)
Riot Games (1)
root-cause analysis (2)
Route53 (1)
rule engine (1)
Rx (2)
Samza (1)
scalability (12)
scale (1)
scripting library (1)
search (4)
security (8)
Servo (1)
shared libraries (1)
simian army (5)
SimpleDB (3)
site reliability (1)
spark (2)
spinnaker (1)
sqoop (1)
ssd (2)

ssl (2)
STAASH (1)
Stamos (1)
stream processing (4)
streaming (2)
suro (1)
SWF (1)
synchronization (1)
tablet (2)
testability (1)
Timed Text (1)
tls (2)
traffic optimization (1)
TTML (2)
TV (5)
UI (15)
UltraDNS (1)
unit test (2)
uptime (2)
user interface (5)
Velocity (1)
video quality (2)
visualization (1)
WebKit (3)
websockets (1)
Wii U (1)
windows (1)
winner (1)
winners (1)
workflow (1)
workshop (1)
ZeroToDocker (1)
ZooKeeper (1)
zuul (1)