

DEC 18, 2015

Twitch Engineering: An Introduction and Overview

If you're a Twitch user, you probably know that Twitch is a pretty big site that streams live video, but you may not have an idea of the sheer scale and complexity of all of the different moving parts needed to hold everything together.

Among other things, we have:

- One of the largest live video distribution systems in the world
- A real-time chat system

IPR2025-00268
SportsCastr Inc. EX-2033

- Web services that provide access to functionality and data
- Data storage systems
- Client applications on the web, and on a multitude of platforms — mobile and console in particular
- Data science infrastructure
- Internal tooling and systems — configuration management, deployment systems, hardware and software provisioning, testing and QA
- Network infrastructure that keeps the bits between all of these systems and the end users flowing

Each of these different components of Twitch has different challenges that need to be tackled — among them low-level optimization of video encoding, large scale system scaling, implementing products across multiple platforms and form factors, and improving our engineering efficiency via improved tooling.

Read More.

Engineering at Scale

Before we dive into what technology we currently use, it's important to note that many of the decisions that we make surrounding technology are driven by the size of Twitch and how rapidly it is growing:

- We have over 30,000 people streaming video on Twitch simultaneously.
- We have peaked at over 2 million concurrent video streams on our site.
- Our chat service delivers over 10 billion messages a day.
- Our web APIs handle over 50 thousand requests a second on average.
- Over the last 3 years, the number of engineers at Twitch has grown by over 800%.

This means that we have to scale at both an architectural and organizational level. As each individual systems grows, we expose different kinds of failure modes that aren't problems in smaller systems, but become problematic as systems grow larger. This includes problems like hardware and network

failures that are rare in small systems but commonplace in larger ones, and doing things like capacity planning to make sure that you can build and architect new infrastructure and systems before they are needed.

Beyond individual systems growing larger, we are also adding more systems to Twitch to make our product better. Each of these new systems adds more complexity to our overall system, which makes engineering more complex for everyone. And adding the extra engineers to build those systems also adds organizational complexity — communication and coordination is an O(n2) scaling problem if left alone. There have been many great articles written (and ideas) about how to try to solve these problems, but there isn't enough room here to do them justice.

Our Technology Stack (A 50 thousand foot view)

There is way too much to talk about in depth here, but I'll attempt a high-level overview of the the key pieces of technology that keep Twitch up and running.

Video System

The video system is responsible for getting video from the broadcaster to our viewers. This includes the following core components:

- Video ingest — We take RTMP video in, and then transport it to the transcode system.
- Transcode system — We take the incoming RTMP stream from the broadcaster, and transcode it into multiple HLS streams. This is implemented via a combination of C/C++, and Go.
- Distribution and Edge — Distribute the HLS streams to our geographically disparate POPs, such that you have the highest quality video streaming experience. Again, mostly written in Go.
- VOD (Video on Demand) — we take all of our incoming video systems and archive them for our VOD system.

Chat

Chat is a highly scalable real-time distributed system written in Go. It delivers hundreds of billions of messages per day to users who are watching video via our own protocols, as well as supporting IRC as a communication protocol, making it easy for developers to create IRC bots to add their own custom chat features. Core chat components include:

- Edge — Receives and distribute messages between clients and backend services. Edge speaks the IRC protocol over both raw TCP and WebSockets.

- Pubsub — Distributes messages internally across Edge nodes. Pubsub and Edge combine to form a hierarchical message distribution system which executes massive fanout.

- Clue — Perform analysis on user actions and send them through our business logic pipeline. Is the user banned in the channel they're speaking in? Are they a subscriber? Are they banned or exhibiting abusive behavior? To inform these decisions, Clue aggregates data across many sources, including databases, internal APIs and caches.

- Room — Responsible for the viewer list. Room aggregates, stores, and queries membership data across all our edges to retrieve viewer lists for each channel's chat room.

Web APIs and Data

Besides our real-time services (video and chat) we also have a substantial number of services including, but not limited to:

- Web APIs that allow users to manage and customize their profiles and subscriptions

- Search and Discovery services that help you to find the streams that you want.

- Revenue systems — these are systems that allow us to manage advertising and subscriptions, and make sure that our partners get their income.

- Administrative tools that help our support staff resolve your problems.

This is all built on a mix of Rails, Go, and various different open-source applications that are used for routing, caching, and data storage. We have also been doing a lot of vertical and horizontal partitioning of our data and data APIs in order to make our systems scale better — both at a technical and organizational level.

Web and Client Applications

All of these services are lovely, but useless without easy to use client applications that create a good user experience. We have a multitude of different client applications that make it easy to access Twitch.

- Our desktop web application, which started off as a vanilla Rails application, but has evolved into an Ember app.

- Native apps for the major mobile and tablet platforms (iOS and Android).

- Various console applications, including XBox One, XBox 360 and Playstation 4.

Data Science Infrastructure

Science Engineering builds data collection systems and analysis tools to study how our users interact with our products. These systems include:

- The data pipeline, which collects, cleans, and loads over a billion events a day into our data warehouse
- Streaming aggregators to summarize metrics in near real-time for broadcasters.

Tools and Operational Infrastructure

The above services directly support the application and product. However, the operational infrastructure that we need to manage these services is also extremely important. We have and maintain tools for:

- Testing all of these services by manual QA and automated testing — we currently use Jenkins for automated build and test.
- Tools for rapid deployment and rollback of changes — we have a couple of home-grown tools here.
- Monitoring and alerting systems that let us know when things are going wrong, and help us to diagnose and fix them. Currently a mix of tried and tested tools like ganglia, graphite, and Nagios.
- Distributed trace. This is something new that we've been working on, to help us understand and debug our systems as we move further and further into the realm of distributed systems and microservices.

At the physical level, we currently run an actually fairly large number of "bare metal" POP (points of presence) all over the world — this allows us to deliver higher quality video due to the unusual requirements of video delivery (lots of bandwidth!).

We also have been moving an increasing amount of our services to Amazon Web Services — this helps to reduce the amount of operational overhead, as well as to take advantage of the convenience and scalability of many of their services.

In Closing

We hope this has been an interesting overview of what makes Twitch tick under the hood. If you are interesting in learning more about these (or other) systems, please look at the Twitch Engineering site!

In other news



DEC 21, 2015

Twitch Gives Back: 2015 Charity Recap



DEC 18, 2015

Pay for Channel Subscriptions with Major Brand Gift Cards

Security Instagram

Terms of Service

Privacy Policy

Ad Choices

Cookie Policy

Partners

Affiliates

© 2024 Twitch Interactive, Inc.