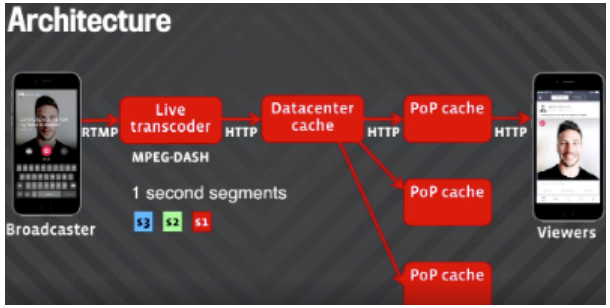
EXAMPLE

How Facebook Live Streams to 800,000 Simultaneous Viewers

High Scalability
Jun 27, 2016 — 7 min read



Fewer companies know how to build world spanning distributed services than there are countries with nuclear weapons. Facebook is one of those companies and Facebook Live, Facebook's new live video streaming product, is one one of those services.

Facebook CEO Mark Zuckerberg:

> The big decision we made was to shift a lot of our video efforts to focus on Live, because it is this emerging new format; not the kind of videos that have been online for the past five or ten years...We're entering this new golden age of video. I wouldn't be

Subscribe

> surprised if you fast-forward five years and most of the content that people see on Facebook and are sharing on a day-to-day basis is video.

If you are in the advertising business what could better than a supply of advertising ready content that is never ending, always expanding, and freely generated? It's the same economics Google exploited when it started slapping ads on an exponentially growing web.

An example of Facebook's streaming prowess is a 45 minute video of two people exploding a watermelon with rubber bands. It reached a peak of over 800,000 simultaneous viewers who also racked up over 300,000 comments. That's the kind of viral scale you can generate with a social network of 1.5 billion users.

As a comparison The 2015 Super Bowl was watched by 114 million viewers with an average 2.36 million on the live stream. On Twitch there was a peak of 840,000 viewers at E3 2015. The September 16th Republican debate peaked at 921,000 simultaneous live streams.

So Facebook is right up there with the state of the art. Keep in mind Facebook would have a large number of other streams going on at the same time as well.

A Wired article quotes Chris Cox, Facebook's chief product officer, who said Facebook:

Has more than a **hundred people** working on Live. (it started with ~12 and now there are more than 150 engineers on the project)

Needs to be able to serve up **millions of simultaneous streams** without crashing.

Need to be able to support **millions of simultaneous viewers on a stream**, as well as seamless streams across different devices and service providers around the world.

Cox said that "It turns out it's a really hard infrastructure problem."

Wouldn't it be interesting if we had some details about how that infrastructure problem was solved? Woe is we. But wait, we do!

Federico Larumbe from Facebook's Traffic Team, which works on the caching software powering Facebook's CDN and the Global Load Balancing system, gave an excellent talk: Scaling Facebook Live, where he shares some details about how Live works.

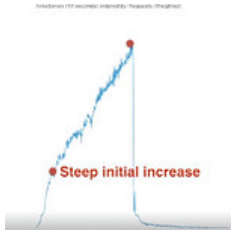
Here's my gloss on the talk. It's impressive.

Origin Story

- Facebook is a new feature that allows people to share video in real-time. (Note how this for Facebook is just another feature).
- Launched in April 2015 Live could only be used by celebrities through the Mentions app as a medium for interacting with fans.
- This began a year of product improvement and protocol iteration.
  - They started with HLS, HTTP Live Streaming. It's supported by the iPhone and allowed them to use their existing CDN architecture.
  - Simultaneously began investigating RTMP (Real-Time Messaging Protocol), a TCP based protocol. There's a stream of video and a stream of audio that is sent from the phone to the Live Stream servers.
    - Advantage: RTMP has lower end-end latency between the broadcaster and viewers. This really makes a difference an interactive broadcast where people are interacting with each other. Then lowering latency and having a few seconds less delay makes all the difference in the experience.
    - Disadvantage: requires a whole now architecture because it's not HTTP based. A new RTMP proxy need to be developed to make it scale.
  - Also investigating MPEG-DASH (Dynamic Adaptive Streaming over HTTP).
    - Advantage: compared to HLS it is 15% more space efficient.
    - Advantage: it allows adaptive bit rates. The encoding quality can be varied based on the network throughput.
  - Pied Piper Middle-Out Compression Solution: (just kidding)
- Launched in dozens of countries in December 2015.

Live Video is Different and that Causes Problems

- The traffic pattern of the Watermelon video mentioned earlier:
  - A very steep initial rise, in a few minutes it reached more than 100 requests per second and continued increasing until the end of the video.
  - Then traffic dropped like a rock.
  - In other words: traffic is spiky.



- Live video is different than normal videos: it causes **spiky traffic patterns**.
  - Live videos are more engaging so tend to get watched **3x more** than normal videos.
  - Live videos appear at the top of the news feed so have a higher probability of being watched.
  - Notifications are sent to all the fans of each page so that's another group of people who might watch the video.
- Spiky traffic cause problems in the caching system and the load balancing system.
- **Caching Problems**

  - A lot of people may want to watch a live video at the same time. This is your classic Thundering Herd problem.
  - The spiky traffic pattern puts pressure on the caching system.
  - Video is segmented into one second files. Servers that cache these segments may overload when traffic spikes.
- **Global Load Balancing Problem**
  - Facebook has points of presence (PoPs) distributed around the world. Facebook traffic is globally distributed.
  - The challenge is preventing a spike from overloading a PoP.

Big Picture Architecture

This is how a live stream goes from one broadcaster to millions of viewers.

A broadcaster starts a live video on their phone.

The phone sends a RTMP stream to a Live Stream server.

The Live Stream server decodes the video and transcodes to multiple bit rates.

For each bit rate a set of one-second MPEG-DASH segments is continuously produced.

Segments are stored in a datacenter cache.

From the datacenter cache segments are sent to caches located in the points of presence (a PoP cache).

On the view side the viewer receives a Live Story.

The player on their device starts fetching segments from a PoP cache at a rate of one per second.

How does it scale?

- There is one **point of multiplication** between the datacenter cache and the **many PoP caches**. Users access PoP caches, not the datacenter, and there are many PoP caches distributed around the world.
- Another multiplication factor is **within each PoP**.
  - Within the PoP there are **two layers**: a layer of HTTP proxies and a layer of cache.
  - Viewers request the segment from a HTTP proxy. The proxy checks if the segment is in cache. If it's in cache the segment is returned. If it's not in cache a request for the segment is sent to the datacenter.
  - Different **segments are stored in different caches** so that helps with load balancing across different caching hosts.

Protecting the Datacenter from the Thundering Herd

- What happens when all the viewers are requesting the same segment at the same time?
- If the segment is not in cache one request will be sent to the datacenter for each viewer.
- **Request Coalescing**. The number of requests is reduced by adding request coalescing to the PoP cache. Only the first request is sent to the datacenter. The other requests are held until the first response arrives and the data is sent to all the viewers.
- New caching layer is added to the proxy to avoid the **Hot Server problem**.
  - All the viewers are sent to one cache host to wait for the segment, which could overload the host.
  - The proxy **adds a caching layer**. Only the first request to the proxy actually makes a request to the cache. All the following requests are served directly from the proxy.

PoPs are Still at Risk - Global Load Balancing to the Rescue

- So the datacenter is protected from the Thundering Herd problem, but the PoPs are still at risk. The problem with Live is the spikes are so huge that a PoP could be overloaded before the load measure for a PoP reaches the load balancer.
- Each PoP has a limited number of servers and connectivity. How can a spike be prevented from overloading a PoP?
- A system called Cartographer maps Internet subnetworks to PoPs. It measure the delay between each subnet and each PoP. This is the latency measurement.
- The load for each PoP is measured and each user is sent to the closest PoP that has enough capacity. There are counters in the proxies that measure how much load they are receiving. Those counters are aggregated so we know the load for each PoP.
- Now there's an optimization problem that respects capacity constraints and minimizes latency.
- With control systems there's a delay to measure and a delay to react.
- They changed the load measurement window from 1.5 minutes to 3 seconds, but there's still that 3 second window.
- The solution is to **predict the load before it actually happens**.
- A **capacity estimator** was implemented that extrapolates the previous load and the current load of each PoP to the future load.
  - How can a predictor predict the load will decrease if the load is currently increasing?
  - **Cubic splines** are used for the interpolation function.
  - The first and second derivative are taken. If the speed is positive the load is increasing. If the acceleration is negative that means the speed is decreasing and it will eventually be zero and start decreasing.
  - Cubic splines predict more complex traffic patterns than linear interpolation.
  - **Avoiding oscillations**. This interpolation function also solves the oscillation problem.

- The delay to measure and react means decisions are made on stale data. The interpolation reduces error, predicting more accurately, and reduces oscillations. So the load can be closer to the capacity target
- Currently prediction is based on the last three intervals where each interval is 30 seconds. Almost instantaneous load.

Testing

You need to be able to overload a PoP.

A load testing service was built that is globally distributed across the PoPs that simulates live traffic.

Able to simulate 10x production load.

Can simulate a viewer that is requesting one segment at a time.

This system helped reveal and fix problems in the capacity estimator, to tune parameters, and to verify the caching layer solves the Thundering Herd problem.

Upload Reliability

Uploading a video in real-time is challenging.

Take, for an example, an upload that has between 100 and 300 Kbps of available bandwidth.

Audio requires 64 Kbps of throughput.

Standard definition video require 500 Kbps of throughput.

Adaptive encoding on the phone is used to adjust for the throughput deficit of video + audio. The encoding bit-rate of the video is adjusted based on the available network bandwidth.

The decision for the upload bitrate is done in the phone by measuring uploaded bytes on the RTMP connection and it does a weighted average of the last intervals.

Future Direction

Investigating a push mechanism rather than the request-pull mechanism, leveraging HTTP/2 to push to the PoPs before segments have been requested.

Related Articles

On HackerNews

Scaling Facebook Live

Why Facebook And Mark Zuckerberg Went All In On Live Video

Connecting the World: A look inside Facebook's Networking Infrastructure

Gamoloco tracks live video stats for 1476093 channels.

Member discussion

0 comments

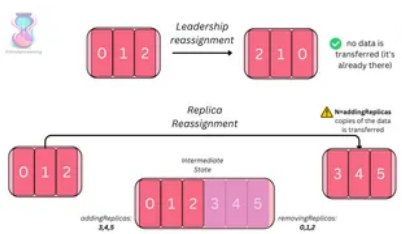
Start the conversation

Become a member of **High Scalability** to start commenting.

Sign up now

Already a member? Sign in

READ MORE



Kafka 101

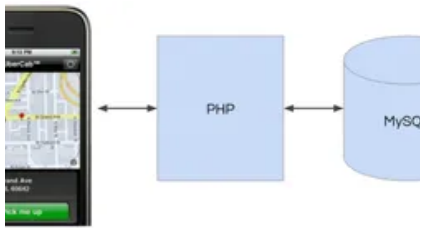
This is a guest article by Stanislav Kozlovski, an Apache Kafka Committer ...

By HS Editor — May 9, 2024

Capturing A Billion Emo(j)i-ons

This blog post was written by Dedeepya Bonthu. This is a repost from her ...

By HS Editor — Mar 26, 2024



Brief History of Scaling Uber

This blog post was written by Josh Clemm, Senior Director of Engineering ...

By HS Editor — Mar 14, 2024

Behind AWS S3's Massive Scale

This is a guest article by Stanislav Kozlovski, an Apache Kafka Committer ...

By HS Editor — Mar 6, 2024

High Scalability

Sign up

High Scalability

Building bigger, faster, more reliable websites.

jamie@example.com Subscribe