Notes from QCon

Author: Ty Rocca

Claps: 74

Date: Oct 29, 2022

A whole bunch of us were at QCon in SF this week.

Thatâ€™s me at front, left, bottom.

The number one thing I got out of the conference was our team coming home with a new vocabulary. We face all these technical challenges at Klaviyo and half the time we need to make up new terminology just to describe these problems that come with big distributed systems, rapid scaling, tons of data, and going from twenty engineers when I started to three hundred today. We all picked up new â€” and more standard â€” terminology for the problems and solutions we face all the time. I suspect this new terminology will now propagate across Klaviyo. And you know what they say about writing code â€” naming things correctly is half the battle.

This post has notes from talks, advice for future presenters, and more photos!

The Talks

We were all on slack live threading our notes and one or maybe two snarky comments. A few highlights I noticed keeping up on slackâ€¦a mix of humor and legit notes:

How Did It Make Sense at the Time? Understanding Incidents As They Occurred, Not as They Are Remembered

(abstract)

We liked these quotes from the speaker:

- â€œA near miss rarely initiates the type of scrutiny that an actual impactful incident does.â€
- â€œFailure free operations require experience with failure.â€
- â€œPedestrians jaywalk and engineers press return.â€

Incident management at Klaviyo is a big deal. We started with the PagerDuty Incident Response protocol and made it our own over the years. Our main new takeaway from this talk was to spend more time during the RCA (root cause analysis) asking why whatever actions led to the incident made sense to the human who did the action at the time.

24/7 State Replication

(abstract)

â€œâ€˜If you put some consensus on it, these problems will go awayâ€™ No, now you will have consensus problems!â€ â€” Zac Bentley

Real-Time Machine Learning: Architecture and Challenges

(abstract)

â€œI thoroughly disagree with her point that Data scientists should not have to worry about where the data comes from â€” I get that itâ€™s stream vs batch, but knowing what goes in an algorithm is fundamental to knowing if the algorithm matches the stats assumptions.â€ â€” Christina Dedrick

Programming Your Policies

(abstract)

Nice watching a little demo of OPA here.

â€œI love the idea of policy as code, but staying 2 steps ahead of unintended consequences, not giving yourself a false sense of confidence, and preventing people from doing The Wrong Thing is such a big part in making it workâ€ â€” Greg Yu

Dark Side of DevOps

(abstract)

â€œMaybe the most useful part of this is acknowledging the semi-obvious. Like paved path is not as universally useful as you might thinkâ€ â€” Chris Chiodo

Engineering an API First Product

(abstract)

Lots of good stuff in this talk.

Things to think about when building an API first business:

- Versioning â€” think like a company that shipped code on CDroms etc
- Documentation
- SDKs
- Logging + Debugging
- Billing
- Measuring Success

Building Modern Backends

(abstract)

â€œParaphrasing, learn Rust to be reminded of how much work other languages do to collect garbage for youâ€ â€” Andrew Shearer

Engineering Considerations for Running Machine Learning Models at the Edge: Application in Body Scanning for eCommerce

(abstract)

How did they implement this?

- React Native
- CoreML (iOS), TFLite (android)

How do you deal with ML issues since they happen on the client?

- Latency for ML model download
- Failure to download
- API call throttling
- Crash analytics tools
- Alarm payload informative but anonymous â€” NO PII !!!!

Stress Free Change Validation at Netflix

(abstract)

â€œBe there in a bit, stressing out about validating my changes first.â€ â€” Zac Bentley

How to Build Reliable Systems Under Unpredictable Conditions

(abstract)

â€œAs opposed to How to build unreliable systems under predictable conditions.â€ â€” Travis Hansen

â€œThis was a sales pitch, tbh, but not a bad oneâ€ â€” Alex Mitelman

Eight fallacies of distributed systems (all familiar to us):

- Network is reliable
- Latency is zero
- Bandwidth is infinite
- Network is secure
- Topology is fixed
- There is only one admin
- Transport cost is zero
- Network is homogenous

Infrastructure as Code: Past, Present, Future

(abstract)

â€œWeirdest intro ever: suggested we come find a tattoo on the speakerâ€™s body.â€ â€” Chris Chiodo

â€œBecame enamored with YAMLâ€¦is this true? did anyone ever actually *love* yaml?â€ â€” Chris Chiodo

The Future of Work: How Flexibility Unlocks Potential for People and Organizations

(abstract)

â€œSAY HYBRID AGAINâ€ â€” Travis Hansen

â€œMy entire career has been built outside those hours lol. The pandemic may have made the situation worse for some, but it did not introduce working outside of 9â€“5 as a norm for many.â€ â€” Sean Kelly

Modeling Patterns for Digital Transformation

(abstract)

Some things we agree with (and already do!):

- Code deployments should be often, waiting for big releases caused big needs for downtime. Avoid like plague
- Invest in telemetry, this is core and build kpi off of this

- You build it, you own it. DevOps is not the place to huck your problems

Dark Energy, Dark Matter and the Microservices Patterns?!

(abstract)

This talk was packed.

â€œThis is a good, unbiased trade off between monoliths and microservice architectures. Also glad the physics reference is an applicable metaphor and not an attempt to coin buzzwords.â€ â€” Tom McDevitt

Quote from speaker we liked: â€œYou donâ€™t get the benefits from doing microservices. You get the benefits from doing microservices wellâ€

Panel: â€œJustâ€ Engineering Culture

(abstract)

Lindsey Curran liked this point: â€œWhatâ€™s wrong with root cause: doesnâ€™t exist, people use it to construct a narrative that makes them feel better.â€

Suf Hamzah liked this point: â€œItâ€™s all about collaboration, inviting others to share their POV during the meeting instead of filling up the templatesâ€

Someone suggested reading how.complex systems.fail.

Panel: Building Performant Microservice Architectures

(abstract)

How do you handle slow microservices?

- Redraw boundaries to avoid latencies of interservice communication
- Look at how everything is composed together
- Need to clarify: what is slow? Throughput or latency? Follow the data
- Align services with structure of organization
- Look at long tail latencies â†’ too many retries

Performance

- Your entire system does not need fast performance
- More services involved with a single function increases risk of high latency and low availability
- Some people want to get latency as low as possible all the time. â€œThatâ€™s not how it worksâ€
- Latency budget
- Need to be realistic, often serialization/deserialization is biggest overhead

â€œThat last point is a huge one. JSON is expensive as hell, relatively speaking.â€ â€” Sean Kelly

Honeycomb: How We Used Serverless to Speed Up Our Servers

(abstract)

- When considering serverless, always study limits.
- Change SDK retry limits â€” default wait is way too high
- And talk to cloud provider. Donâ€™t surprise them.
- Consider tasks that are a fit (Need a lot of work that rapidly scales. Needs to be urgent. If no one is waiting for this work, spin up k8s, use EC2, etc.)

Architecture:

- Move state to object storage
- Shard into units
- Process concurrently
- Reduce outside lambda

Before scaling:

- Tune properly. Cost, performance, CPU/RAM ratio
- Optimize properly (for each architecture)
- Observability (Lambda layer)
- Measure carefully (especially cost)

Amazon DynamoDB: Evolution of a Hyper-Scale Cloud Database Service

(abstract)

Really liked this talk and this speaker.

2004 Amazon outage analysis â€” do we really need a relational DB â€” led to 2007 Dynamo paper, 2012 DynamoDB launch.

Maintained single-digit (?) milliseconds response time while serving 105.2 million requests per second during Amazon prime day 2022! (Assume this is peak loadâ€¦)

Interesting twitter thread from the speaker: â€œBimodality in distributed system is a ðŸŒ‹ waiting to eruptâ€¦.â€

Unsolicited advice for presenters

In our slack channel, Dmitry Mamyrin asked what people **liked** about the talks they really enjoyed, and what people **disliked** about the talks they thought were bad.

Liked

- Brief but detailed intros into problem statements
- Technical details, depth

- Examples (code snippets, tech used, diagrams)
- Pros/cons, insights, advice/recommendations
- References (books, articles, research)
- Strong opinions, even when disagreed with
- Where they talked about specific problems they encountered, all the *alternatives* they considered, and how they solved them.
- Details and specifics especially when discussing implementation
- Thoughtful slides or briefing aids which helped clarify / summarize / visualize the points being made in a way that was clearly readable from 100 feet away
- Speakers who were prepared and rehearsed to deliver their talks clearly and enthusiastically
- The unconferences were the most reliable way to get 1-on-1 and small-group interaction among peers

Disliked

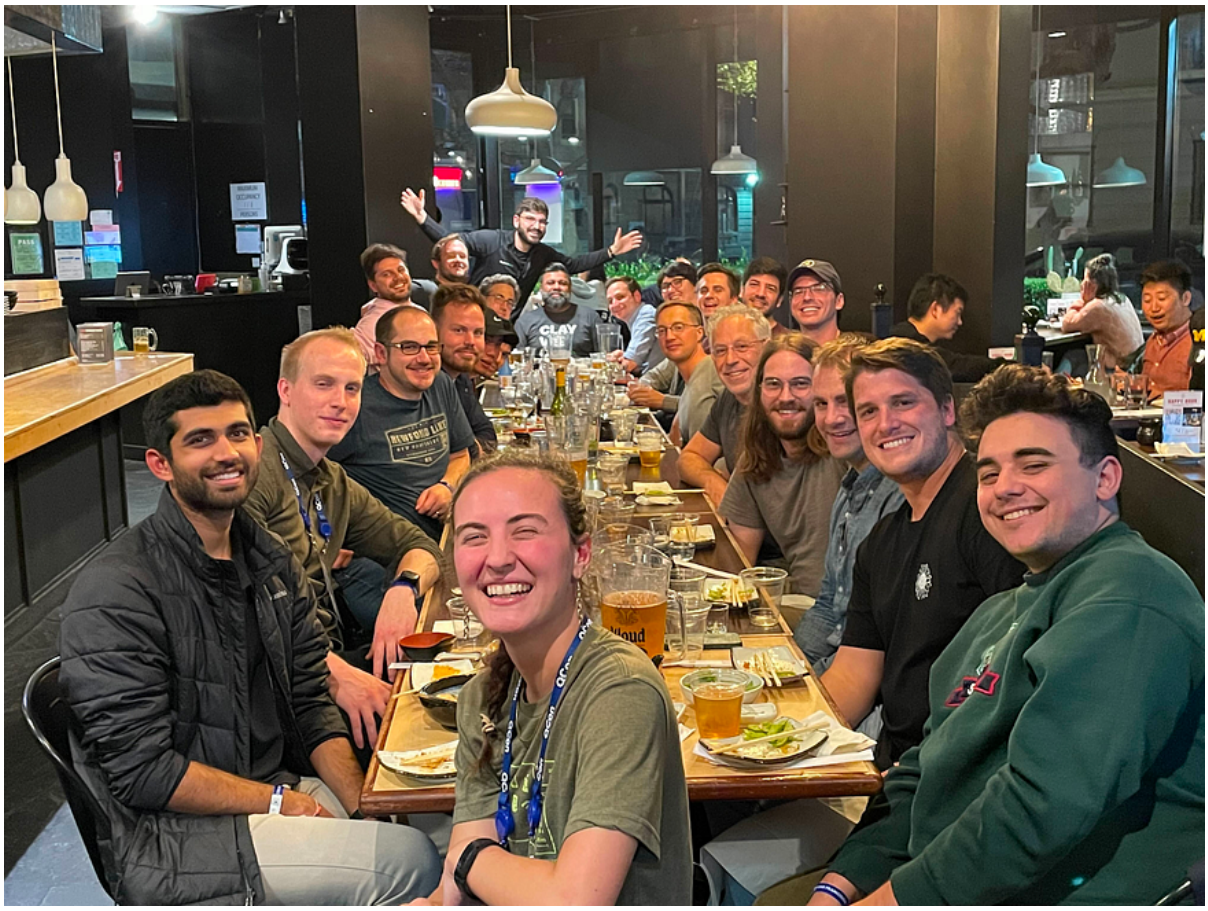
- Talks without clear goal/subject
- Slide decks full of gifs and no details
- Long intros, too much focus on definitions
- Long sales pitches
- Talks without details and without examples
- Slides that clearly were not UX tested for contrast, lighting, â€œback of roomâ€ legibility
- Talks that were too high level. They were problem -> solution or just high level architectures. I had trouble filling in how they got there
- Talks that were not actually about the stated topic
- Unmoderated panel discussions

One last thought

I talked to so many people at the conference who never heard of us. When I told them about our scale, that we ingest billions of events per day, a lot of conversations got very interesting very fastâ€¦

And more photos

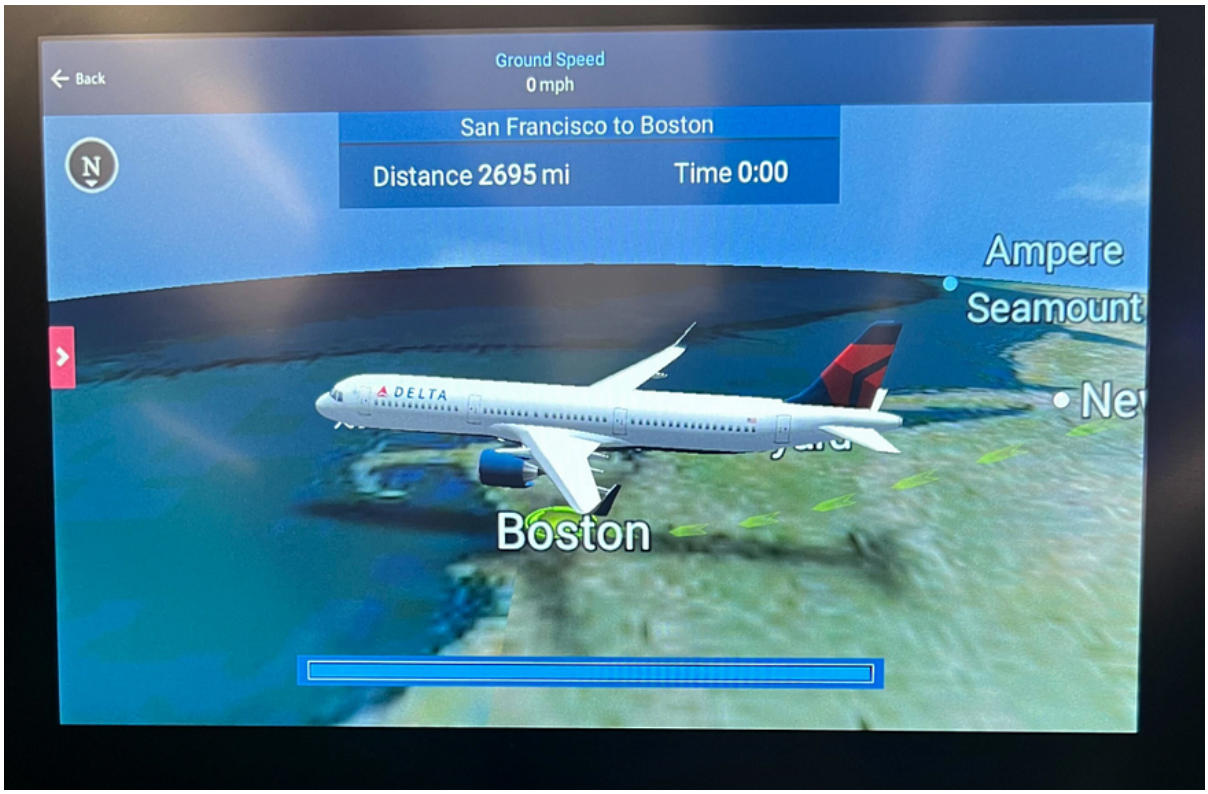
Nikita, Oliver, Chris, Tian, Jon, Kez, Greg, Andrew (taking selfie)



Harsh, Maya, Andrewâ€¦and many others!

The full Klaviyo contingent

Taken by Andrew on his runâ€¦



To scale picture of my plane