EPISODE 704

[INTRODUCTION]

[00:00:00] JM: Demand for live streaming video over the internet is increasing. After the emergence of early live streaming platforms, like Twitch and Facebook Live, more forms of video have become accessible over live stream, such as sports. Live streaming is a harder engineering problem than delivering a static video file, because the information distributed on a livestream is constantly changing.

DZN, spelled D-Z-N, is a live streaming service for watching fight events, such as boxing. The workloads for live streaming can be highly bursty. When a fight is scheduled to happen, the vast majority of traffic will hop on to watch the fight 20 seconds before the fight starts. A huge number of users logs into DZN and starts watching all at the same time. This quick spike in traffic means that DZN has to have servers spun up and be ready in advance.

Luca Mezzarlira and Yan Cui are engineers at DZN. Yan was previously on the show in a few amazing episodes to talk about serverless infrastructure and the complexities of real-time video game software development. Those episode links are in the show notes. I highly recommend checking them out. Today's show is a discussion of architecting a system to handle a high-bandwidth bursty customer use case. I hope you like it.

[SPONSOR MESSAGE]

[00:01:38] JM: Data holds an incredible amount of value, but extracting value from data is difficult, especially for non-technical non-analyst users. As software builders, you have a unique opportunity to unlock the value of data to users through your product or service. Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love.

Give users intuitive access to data in the ideal place for them to take action within your application. To check out Jaspersoft, go to softwareengineering daily.com/jaspersoft and find out how easy it is to embed reporting and analytics into your application.

Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it is not just your company that wants analytics. It's also your customers that want analytics.

Jaspersoft is made by TIBCO, the software company with two decades of experience in analytics and event processing. In a recent episode of Software Engineering Daily, we discussed the past, present and future of TIBCO as well as the development of Jaspersoft. In the meantime, check out Jaspersoft for yourself at softwareengineeringdaily.com/jaspersoft.

Thanks to Jaspersoft from being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:11] JM: Luca Mezzarlira, you are the chief architect at the DZN, and YAN, you are a principal engineer at DZN. You've also been on the show several times to talk about various serverless efforts and companies you've worked at. Guys, welcome to Software Engineering Daily.

[00:03:25] LM: Thank you, Jeff.

[00:03:26] YC: Good to see you again, Jeff.

[00:03:27] JM: Yes, good to talk to you as well. Those shows we've done in the past have been really popular and I'm looking forward to exploring technical issues with you once again. So DZN, where you guys both work, is a live streaming platform for fight fans. So boxing, for example, is something that is live streamed on DZN. Why is live streaming a hard technical problem?

[00:03:52] LM: That's a good question, Jeff. First of all, we need to explain a little better what is our aim in DZN. That's to understand deeply why is that difficult. DZN was created three years ago and currently we are available in seven markets. We are not just streaming, say, fighting matches, but it is a sport platform. It depends from which country we're streaming, but for instance in Italy, we have Serie Rights. We have Serie B rights that are the main football rights that you can have in Italy.

In Germany, we have Premier League, we have Champions League. In Canada, we have NFL. So let's say our platforms is very, very rich on that side, but for each single country, we have different rights available.

In the case of U.S., we have a lot of exclusive rights around fighting sports, and the main difficult when you work on these kind of things is accountability issues. Definitely, there are a quite a few challenges in order to have your content from stadium or from the pitch or soccer pitch, to the device that the user is watching the content. Because as you can imagine, there are a lot of transformation, a lot of changes in the video delivery pipeline, and therefore the hiccups could happen at any given point.

[00:05:06] YC: Also, from a resileince point of view as well, we can't just say, "Hey, sorry guys. We're down. Please come back in half an hour when the fight is already finished." So we do have to put a lot of effort into making sure that our platform is up especially during those really crucial, hour and a half, maybe two hours when the sporting event is actually happening.

[00:05:25] JM: Live streaming has become a popular way of consuming content. You could imagine live streaming platforms for lots of different verticals, not just fights, not even just sports, but also for – Well, you see this on Twitch. Of course, people are live streaming all kinds of stuff. Why hasn't this been commoditized? Where is there not some off the shelf live streaming technology?

[00:05:49] LM: I think it's a very, very complex area that one, and everyone is trying to find the silver bullet. Obviously, it's very difficult, because I think it's very different also that the source of Twitch, for instance, or when you try to have live streaming from another computer that is showing a game or whatever.

In our case, you have to be reminded that our company started with a broadcaster background, and therefore we have a lot of broadcaster hardware that in a certain way is facilitating us, but at the same time there were some decisions that didn't allow us to scale as quickly as possible on all the brand new technologies. There is also on our side a lot of push to move a lot of things on the cloud, from on-prem solutions, and that will speed up the possibility for us to make our video delivery pipeline smoother and with more monitoring in place and having the de facto a better experience for all of our users.

[00:06:45] JM: There are some AWS services that help with video. There's also services like Brightcove, there's Mux. Is solving for video delivery the same as solving for live streaming video?

[00:07:01] LM: I think there is a big difference on delivering VOD content, so video on demand, and the live content. As Yan suggested, we need to, during a live event, we need to, let's say, always available and provide the best experience possible. We cannot afford any hiccup. If you can imagine when you're recording something on the pitch of – I don't know, an NFL pitch and you're recording with a physical person there, that the content has to stream across the world in no time, on top of it, commentators, voice and also the graphics, then you have to package and then call the content [inaudible 00:07:39], package for all the different devices that currently are supported on the zone. It's literally a lot of stuff that are happening.

On top of that, you need to control the quality. Be sure that these are watched, and then it's not finished yet, because obviously you have all the CDN capacity that you have to take care off. You have to be reminded that you have different video players, because each single device has a different video player. In our case, we are targeting around 30 different devices. Let's say the implementation of the video player implantation that is available on a Samsung Tizen device is completely different from a Roku one, and you need to find the minimum common thing in order to make everything smooth and provide a good experience for our users.

There are a lot of things that we need to tweak in advance and learn from how you're streaming. We have a lot of monitoring around video quality and we are tracking it in order to check what we can improve, how we can improve. Maybe during the week we do some small twitches, small tweaks on our system and our configuration and then automatically we see some improvements. Sometimes we'd make things slightly worse, but thanks God we have a huge – The most important fun part for us is the observability of our video players, because we understand a lot and we are gathering a lot of information from our video players, and that allows us to understand where we have to make improvement in our video pipeline.

[00:08:59] YC: Also, I think from the business point of view as well, there's also a lot of challenges in that particular space given that the rights for sporting events is such a scatter place, where depending on a country, depending on the league, on a sporting event itself, you may not be the one that's controlling the source as well. Often times, for other events that we are streaming, we are getting secondhand streams from other providers that we work with, and vice versa, we sometimes for some of the events, we are the primary source for the streaming content and other providers working with, versus sourcing our content into their own platform as well. So all of these things is all going to add up to really, I guess, a complicated pipeline that involves our own content as well as other third-party content that we can't control.

[00:09:48] JM: Let's say I am going to use DZN to watch a fight. So I download the app. I open it up and I find a fight that I want to watch. I press play. What happens? What's happening on the backend? How is the live stream making its way to my device?

[00:10:07] LM: That's a good question. The first thing that is happening, there is an API call to our API layer that these first checking that you are entitled to, to watch that content, we are checking which country you are based, because obviously those rights are specific for a specific country and we cannot afford to have leaks on that side.

After that, we do an orchestration on our BI layer where we are trying to understand which is the status of our CDNs and the status of the network in order to deliver the best user experience for our users. Then based on the type of content that the user is requesting based on the country where he's in, based on the status of the network, we are providing a list of 4, 5, 6 manifests that basically are the description of the live content or video on demand for the video player.

At that stage, the video player is basically doing – Is retrieving the license acquisition for DRM, and when it do the handshake with the last acquisition server, license acquisition server, we start to parse the different ABR, so the bitrate that are available inside the manifest. Start to understand which is the bitrate of the user, and based on that, picks the best fragmental audio and video that it starts to – The vector could play property in that specific device and in that specific configuration.

Obviously, this is just, say, the technical part on the API layer. Behind is the scene, is that if we – DZN is a very lucky company, because we can jump in any part of our video distribution delivery. We can have physical people that are on the pitch recording the sport event, or we can – Like Yan was suggesting, we can take some other content that is produced by a third-party and just re-stream it and package it for our users. It really depends what kind of rights and deal we are cutting with for the leak.

[00:11:58] JM: How does the delivery of a live stream vary depending on the device, depending on if I'm streaming to my mobile or my tablet or my desktop or my Xbox and depending on the bandwidth that I'm on? Whether I'm on Wi-Fi or whether I'm on a cellular connection.

[00:12:22] LM: Basically, each single manufacturer can – There are some industry standards, but unfortunately, the video delivery pipeline in general is very fermented. That reminds me a lot like in early 2,000, how was the mobile ecosystem with Nokia, Ericson, and then Sony Ericson and all the other manufacturers that were trying to emerge and create their own standard. It's a very, very similar for smart TVs and for console and so on.

Obviously, currently we have three different packagers. There is HMS. There is the packaging for HMS. There is the packaging for DASH and one for smooth streaming. Those three are the three that we can use in order to delivery tiny given device. Obviously, each single device has some capabilities that are mainly managed by the hardware and the connection they have. So some of them maybe they can stream up to a certain level of quality and they cannot go 4K for instance, or full HD. Therefore, we need to add some a capping. Then we have other device that are allowing us to provide everything, but they struggle with low bandwidth. Therefore, we need to provide the right set of ABR and quality inside the manifest.

Overall, let's say the video players are very fragmented well. There are different implementation for different platforms. Vast majority of our platforms are HTML based, but we have some interesting one, like obviously native iOS, native Android, or Roku that are with native code. Basically, they have a completely different approach from web technologies.

[00:13:54] JM: Now, in some shows that we've done about video delivery, we've spent a lot of time talking about codecs. A codec is a compression and decompression algorithm that is made specifically for a media file, and at least in the typical case that I have encountered them. A codec, it might vary depending on what kind of compression rate you need and how fast you need the decompression to happen. How do codecs fit into the process of live streaming delivery?

[00:14:31] LM: Yeah, in our case we are not developing video players end-to-end. We are working with an open stream platforms or native video players that are provided directly from manufacturers. I would say that at this level here, let's say, I cannot answer in an appropriate way or a meaningful way to this question.

[00:14:50] JM: You don't need to make your own codec ladders, for example. That's just a problem that's out of your scope.

[00:14:56] LM: Yes.

[00:14:56] JM: Okay. Cool. The video streams that the fights are actually coming from. Do you have to do engineering around guaranteeing that those video streams stay up, that they're coming to you reliably, or is that kind of taken away from you by other technology providers? How do you have guarantees around the cameras that are at the fights and actually streaming the videos to you? Is that a dependency that you worry about at all?

[00:15:29] LM: Let's say, DZN is a large organization. So we have over a thousand people and a lot of those people are dedicated with a video delivery pipeline. What's happening at the moment is that we have different support teams that are iron glass checking all the devices in all the countries 24/7 to see if there are glitches inside our stream. On top of that, we have obviously a lot of dashboards and monitors that are helping us to understand when something is going wrong or is going to be wrong. We are investing a lot of our time in order to make more visible our, let's say, video delivery pipeline and in general also API pipeline, because it's very important that you know exactly what's going on inside your platform and you can react or potentially also be proactive and prevent that something bad could happen on the quality of your video.

[00:16:18] JM: What role do content delivery networks, CDNs, what dole to they play in your architecture?

[00:16:24] LM: CDNs are fundamental, because a video delivery at the end is, let's say, one of the most important thing for providing a reliable way to serve a content across the globe to millions of users, right? Currently, we have multiple CDNs that we are using and we basically define some priorities for our videos to be delivered from one CDN to another, one based on the capabilities of the CDN on a specific country and also the type of the stream that we are going to serve. Live, for instance, have different configuration than VOD, because certain CDNs are performing better with live. Certain others are performing better with VOD. Therefore, we need to be aware on which one is the best based on test that we are running every time we want to approach a new country.

Overall, let's say, we are trying – We have a set of standard CDNs that we are using, like Akamai, or Limelight, or – I don't remember the third one. Akamai, Limelight, Ladder Free. That's which one it was. Basically, what we're doing is testing also different ones. We recently tried CloudFront. We tried Horizon CDN. We are doing our best in order to provide the best experience to all of our users.

Obviously, CDNs are not the final level that we should use in order to improve the quality. So there is another additional level that can became the mid-tier CDN. The current CDN could became the mid-tier level of caching, having caching on the ISP level, but that is something that we are working right now and we trying to understand how that technique could fit inside our pipeline.

[SPONSOR MESSAGE]

[00:18:13] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you can check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[00:20:22] JM: Now, when you’re trying to deliver a live stream, that’s, I guess by definition, impossible to do, because the stream is going to have some sense of latency. That’s just physics. You can’t have a direct line into what’s going on in the fight. What amount of latency is tolerable and is there a tradeoff between latency and cost that you’re conscious of and that you make decisions around when you’re deciding how aggressively to push out the stream and reduce the latency?

[00:21:01] LM: Yes. As I said, it’s impossible to remove at this stage the latencies. There are a lot of interesting studies in order to minimize the impact that the latency has on OTT platforms.

Currently, I think the problem is not the cost. It's more the quality that you want to provide with your service, because obviously we have some solution that could speed – Let's say, reduce the latency dramatically, but reduce also the quality of the video that we are going to delivery. In our opinion, the video is the king. So the content is the king and we need to provide the best video possible to our user. Therefore, we think that also there is a small amount of latency, but with high quality content is a better experience for our users.

[00:21:42] JM: And a CDN – I'm trying to understand better how the CDN plays a role, because it's very easy for me to understand how a CDN would work for a static video file. For a movie, for example, you can push out the movie to CDNs at the edge so that I can access the move more quickly. But with a live stream, it seems like you would really be needing to update the file that's in the CDN quite aggressively, because you need to update it with the most recent blocks of time since it's a live stream. So it's not like pushing an episode of House of Cards out to the edge of the CDN. You have to push out the most recent parts of a live stream to this CDN. Is it kind of a different problem there in the sense that you're having to update these files that are in the CDN pretty aggressively?

[00:22:37] LM: Yeah. It's the manifest is going to be updated very aggressively. We remind one thing. So all video channels can be cached at the level of the CDN like you do with video to VOD. So when you have streaming video, what usually happen, you have the description of all the different chunk of videos available inside the manifest and all the rail. So when the first one hits the CDN, then it's served directly from the CDN.

What we may see is that basically if you are – 2e have one of the functionalities available on all our devices that allow us to rewind a live event since the beginning of the event and it's mainly because you would see in our monitors that it's just drawing, drawing, drawing and adding more video and audio fragments. So you can, at any given point, decide to go back and watch a specific part of the video also if there is a live without having any hardware of set up box specific with this feature. That's mainly because are caching a lot of video fragments, audio fragments directly on the CDN.

Yes, I agree with you. We need to, let's say, have strong performance in order to write constantly the manifest and update that. After that, the video are divided in chunks that are smaller chunks and can be cached for being served on multiple devices and countries.

[00:23:58] JM: Yan, you and I have had a great conversation around the networking difficulties of real-time gaming. When you look at the challenges of delivering a live stream of video fights, are there similar challenges you deal with at DZN, or is it simpler than the networking difficulties of real-time gaming?

[00:24:21] YC: Certainly, in terms of latency, as Luca mentioned, a lot of the networking stuff is all done by vendors that we're working with already. Unfortunately, we don't have to do a lot. There's a low-level networking work ourselves. Also, in terms of comparing to real-time gaming, your video streams are always going to be some seconds behind. You're not looking real-time within several milliseconds. So I think in terms of the challenge there, it's quite different to making a real-time multiplayer game.

[00:24:52] JM: One place it seems to me to be simpler is the event stream is more predictable. You don't have these different people who are playing a game and changing the nature of the event stream by making decisions. A game decision-based. The event stream for a live stream video fight, it's higher bandwidth, I guess, but it's more predictable. It's just a stream of video content. So you don't have as much variability. Is that an accurate picture?

[00:25:22] YC: Yeah, I guess so. Also, for this this type of live streaming video content is actually quite mature. I imagine, from what I've seen so far, a lot of technologies that you can use out of the box from working with vendors, there's a lot of stuff that we customize on top of that. Ultimately, the infrastructure, the technology underneath, they are quite mature and quite stable already.

[00:25:45] JM: I'd like to get a more detailed picture of the overall architecture of the DZN app. Can you tell me more about your backend infrastructure?

[00:25:55] LM: Yeah. Currently, we are on AWS for the API layer. Partially, also with the video delivery pipeline, we are on the cloud, partly on prems. We have a hybrid cloud solution there.

Overall, let's say we are working with micro services. Currently, we spent last year, or this year basically, revamping completely end-to-end our solution. Working with micro frontends and micro services in order to create, let's say, independent and scalable teams that will allow us to deal with a subdomain of our platform, let's say, owning the entire end-to-end solution that could over infrastructure frontend and backend. Obviously, we are a partner of AWS, so Amazon. On that end, we are working hard in order to, let's say, port what is the existing infrastructure to the new one based on micro services.

[00:26:47] YC: Yeah. On a backend side of things, we're also using an offer of different AWS services that's available to us. In terms of the compute side of things, we use both containers with ECS as well as with Lambda for a lot of the background processing and streaming as well for most of the APIs, because we are quite latency sensitive and we do have quite a lot of scale as well. At this point, I don't know if I can share the official numbers, but we have many millions of subscribers worldwide. At peak, we have a million concurrent viewers on our platform as of today. But that number is going to grow pretty significantly. Given that is all live events, what you find instead, everybody logs in about 20 seconds before the event starts. So we have very spiky traffic, which is not particularly what was suited for serverless.

For most of our APIs along that critical path – Well, in fact, for all our APIs along that critical path is all running containers right now. Lambda is most being used for doing, I guess, more asynchronous event processing, which we have a lot of that.

[00:27:50] JM: You pointed out there that this use case where people log on 20 seconds before the fight happens, and in that 20 second window perhaps, you have massive bursty workload changes and you said that this was not a good use case for Lambda. Now, I've talked to people in the past where they say, "Oh! Lambda is great for bursty workloads." Why is this particular bursty workload not a good fit for Lambda?

[00:28:20] YC: Because it's so bursty, you hit the sailing up limits on how fast you're able to scale up.

[00:28:26] JM: So you're saying you need to pre-provision some containers that are ready for serving traffic so that you're ready for those bursty workloads. So you don't have to endure the cold start time of the Lambda function.

[00:28:41] YC: Part of that is the coaster, but also Lambda has to limit on how many new containers able to create per minute, which right now the default limit is 500. We are talking to AWS about potentially we can raise that for our count, but then again, while our traffic just keep on increasing, every single time we open up in a new market. So every time I sit down and try to work out what our number is going to be, that number changes the next week. So I haven't been able to come up with a number that say, "Okay! We cannot build it. We have to raise the 500 per minute rate and make sure we can create new containers and come up with a number that we can work with AWS and ask them to give us."

Right now, even though you can increase the total number of concurrent executions in your region in your account, that's just a request via support. But that 500 per minute rate, that is kind of a hard limit. Therefore, most people are going to hit, going anywhere near that. But for us, given the scale that we already have and given the way we are increasing that scale, we are hitting that constantly when we run our load test that assimilate what happens when production workload happens against this particular API for instance.

[00:29:53] JM: You said there's a 500 lambda spin up per – What is it? Per minute?

[00:29:59] YC: Per minute per region.

[00:30:00] JM: Per minute per region. Okay. How many containers do you need to spin up when you're getting ready for peak load?

[00:30:07] YC: That I can't share.

[00:30:10] JM: Okay. It's sounds like more than 500.

[00:30:15] LM: The thing is simple. We need to have a certain control what we're doing, because obviously, as you can imagine – Because it's very spiky, our traffic, we need to be ready at the worst case scenario. Usually the rule of thumb that also as architects we are using is trying to have a full control on the critical path and have all the APIs that are highly cachable and chrome jobs that are happening inside the system, or even driven architecture that can be done with Lambdas, we are more than happy to use Lambdas. Obviously, as the business is growing, we need to be cautious that we need to be, let's say, very responsible of our user needs. For instance, if you think about – I remember last year at the AWS Reinvent, where a couple of folks from Netflix that we're saying that we're even optimizing the EMI image that was released on EC2 in order to have in-memory buffer compared to what is provided by Amazon by default. That is just one way to improve that. Then at the moment you couldn't do that with Lambdas. So we need to be very pragmatic on the decision we are making. We think that our future would be servless, but obviously we need to be also cautious that are some limitation there and we want to do our best in order to make that thing better.

[00:31:36] JM: So you mentioned that you would like to eventually be entirely "serverless", but today there are just limitations in the ecosystem, perhaps limitations on the rate limiting of the lambda service. But what are some of the other ways that you use serverless functions at DZN?

[00:31:58] LM: Another one that pops in my mind is for the automation build pipeline. Currently, for the frontend for instance, we created an even driven automation pipeline that is building our artifacts step-by-step through Lambdas. We basically drop – I don't know, stuff like web block inside the Lambda. We allow each single team to build their micro frontend through the Lambda, and each single step is a different part. So we have a step that is building, and that is one Lambda. Then we have another stuff that it is optimizing the code and then other one that is uploading the aftifactory repository and so on. That is another way to do that. As well for the frontend, we are using Lambdas for the deployment pipeline that basically is a Lambda that is deploying content and retrieving it from artifactory and deploying that on S3.

[00:32:51] YC: Yeah. From the backend things perspective, we're using Lambda for a lot of the background processing, asynchronous processing with queues, Kinesis Streams, SNS, SQS, but also I think the SI team, they're also using Lambda for quite a number of different automations, things at how we capture and process all the logs from both our container space workload as well as Lambda-based workload. We also, for some our more critical payment processing of workflows, we're also using Lambda with step functions, which gives you a lot more control over retries or around error handling and so on, things that we – actually makes money for us. Whereby it makes sense to spend a bit more money on services like step function, which is quite a bit more expensive compared to Lambda itself. Those services had been really good for us in terms of giving us both the visibility we need for a critical flow, but also being able to – again, to sit down with our product guys and show them, “Hey, this is our flow. You can do X, Y and Z first and then talk to PayPal and then do X, Y and Z and send email to so and so if things go wrong.”

Those are just some appraises that we use in Lambda pretty heavily. Some internal APIs that doesn’t have a lot of traffic that are far away from the critical part, we’re also using Lambda for those as well and API gateway.

[00:34:12] LM: Yeah. In particular, if they are highly cacheable.

[00:34:15] YC: Yeah. Especially the highly cacheable.

[00:34:17] JM: What do you mean by that? What’s highly cacheable?

[00:34:19] LM: For instance, one of the series that we are using that is what we call the frontend dictionary, where we have all the labels that should be showing inside the frontend. That one is going to change only when, I don’t know, there is someone from product or someone from customer support that wants to change a label or change – Adding more information on a specific error or whatever it is. They change once per week or something like that, or even less than that.

It means that every time that you hit that specific API, we can run Lambda, then we can cache at the cloud form level. We have a high TTL on that. Therefore, we can always serve and protect the Lambda through CloudFront, because it’s going to serve all the responses that are coming from a specific region.

[00:35:06] JM: There’s another application of serverless that you wrote about Luca called Lambda at Edge. What is Lambda at the edge?

[00:35:15] LM: Basically, the concept of this, using Lambdas, but not inside the data center. Let's say using that on the edge. Therefore, it's a big doubt every time that there is a request that is in CloudFront. You can decide which, let's say, life cycle you want to use. It could be when the request is coming in CloudFront, the request is coming out from CloudFront and going to the data center, or when it's coming from the data center to CloudFront, or when it's coming from the CloudFront to the request, therefore the client in that case. You can apply different logic on that. Obviously, it has to be very quickly, because otherwise there would be an impact on the response time.

We started to do some interesting thing also with Lambda at the Edge. For instance, we apply the strangler pattern on the frontend task to the Lambda at the Edge, because every time that the user is requesting a specific URL, we can pick it up, understand where the user is coming from, which is the device and then say, "Okay. If you are –" I don't know, "a user is based in Canada and you want to access or sign in URL, we can decide to show you the sign in URL or the old one," and it's basically all the rules that we are applying our map inside the Lambda at the Edge that are describing basically how we want to tweak or how we want to serve basically our content.

[00:36:36] JM: So this allows you to do A-B testing.

[00:36:39] LM: It does at certain extent obviously. We can issue a cookie at the Lambda at the Edge and create a sticky session and say, "Okay, ever time that this browser is trying to – Browser or device obviously, is trying to access a request, that specific URL, we serve always the same content, either that is the new one of the old one."

[SPONSOR MESSAGE]

[00:37:07] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[00:39:15] JM: You have talked a little bit about what you do to scale up in the event of a bursty workload, for example, a super popular fight, and 20 seconds before the fight happens, a bunch of people hop on and they start streaming it. What are some other things that you have to do to scale up in the event of one of these super popular fights and one of these bursty workloads? Also, how do you prepare for that? What kinds of load testing do you do?

[00:39:41] YC: We are also now working with the SIE team as well as the performance team on trying to make that as part of our pipelines so that every time you build, we always run that regardless what happens. Right now, a lot of that is not automated as much as we like. We are using tools like – I think with Locust, which we're using which allows to run distributed load test from [inaudible 00:40:02] to generate the amount of traffic that is going to representative of what we are likely to see. But in terms of actually running the load test itself, that is not as automated as we like. We are working towards making that as part of our pipelines so that every time we run the load test automatically.

In terms of predicting how much traffic we're likely you to see, we have a lot of heuristics and analytics data that we can use based on – or I guess the last few years and we also have control of the schedules so we know up to about three weeks ahead of time what's going to be on and when based on the analytics data we have. We roughly know how many people are likely to log in for a particular event. Some matches are not as popular as others. If you follow football, or I guess soccer for you, some matches in Spanish League for example or in the Italian League, they're likely to have a massive spike in traffic. Whereas others are less likely.

Based on historical data as well as how many Burst subscribers we have in each region, we can figure out roughly how many people are likely to log in at the same time. We can use that to drive our provisioning decisions and also use that help us to create profiles for our load tests as well.

[00:41:19] JM: Is it a challenge to build this kind of infrastructure for live streaming while keeping the cost of that infrastructure cheap enough to have good margins? Is cost an important feature here or do you feel like you're safely in the realm of not spending too much on the infrastructure?

[00:41:40] LM: I would say that at the moment we are focusing more on quality than cost. Obviously, we keep an eye on that and we make sure that are on the tradition that we have every year. Overall, let's say the quality is the most important thing, the quality of the service and what we want to achieve obviously. As you can understand, we are relatively a new company in this world and it's very, very important considering that it's a B-C business provide to provide the best video quality and best experience for our users.

We are often a compare to linear channels and stuff like that. Therefore, there is also an education that we are bringing with us every time that we approach a new country. Because a lot of people that are used to maybe sign a contract with a cable provider more than using an OTT in certain countries. Therefore, for them, they try to compare our service with the cable service that it's not really comparable, because it's a completely infrastructure and way to deliver the content. Therefore, at the moment, we need to, in a certain way, fight against the idea that OTT is not ready yet for live content.

[00:42:51] YC: Yeah, also to add to that as well, compare – Actually, infrastructure cost, our cost for retaining the image rights as well as people, the infrastructure cost is peanuts.

[00:43:03] JM: What's your continuous delivery process like?

[00:43:06] LM: From a backend's point of view, we are using drones to automate the different process for all of our APIs and backend services. Depending on whether or not is it containerized solution or is it running on Lambda, we're using slightly different tools. For a lot of the infrastructure stuff we use Terraform, and that's kind of across both Lambda as well as containerized workloads for, I guess, all the common infrastructure pieces. We also use a server framework as well to provision and all through the Lambda function themselves and also for API gateway, because it just makes like a bit easier compared to Terraform.

But for everything else, we use Drone to automate the process of building the image, publishing it to the ECR, Elastic Container Registry Service that Amazon has and then deploying to various environments from there.

[00:43:56] JM: You say you use the serverless frame?

[00:43:58] LM: Yes, we do. I think for the front end stuff as well.

[00:44:02] YC: Yeah, on the frontend, currently – before Drone, we created our own solution on Docker Swarm, basically is parallelizing our build system for part of the system. Another part is made with Lambdas with an even driven architecture where basically each single step is receiving a TAR file as input and export a TAR file as output on an S3, and we are using basically S3 events in order to trigger different Lambdas in order to decouple completely the step from the architecture. Therefore, if we want to add a new step, we don't have to, say, have any issue on that.

Because at the end of the day, we are capable to create around 300 artifacts in a matter of minutes, because we are parallelizing the build of the same artifact for multiple countries.

Therefore, the code base that is available for Japan will never be released in Germany and vice versa.

[00:44:56] JM: Let's talk about observability. So this is a high-throughput, highly interactive application stack. What kinds of observability tooling do you have to make sure that things are staying up and running and the system is healthy?

[00:45:13] YC: Sure. On a backend side of things, we're using Datadog. We are collecting metrics from both containerized services running ECS, but also Lambda functions as well and we're using Correlation IDs. So whenever services talk to one another, we're making sure that those Correlation IDs are passed along including all the different logs. Also, we're using – In some places, they're using Xray.

We are looking at few others tracing solutions as well. Right now, Xray is just easier to get started with. So we're using X-Ray for a lot of the tracing locations, request as it flows through various components. Also, we have built some, not open source, built some tooling to make it easy for the Lambda functions so that when we're processing events through asynchronous events such as Kinesis and SNS and SQS, we can also funnel those Correlation IDs across and make it easy for developers to not have to think about, "Okay. Now I need to extract the Correlation IDs and so on and so forth and just make it happen automatically by a middleware that you just inject into your function easily.

[00:46:14] JM: X-Ray is the Amazon distributed tracing system?

[00:46:17] YC: Yes, that's right.

[00:46:18] JM: Are there any areas of the stack that you have less observability into than you would like?

[00:46:23] YC: To be honest, I'd like to have a lot more observability in all the places than we currently have. But still it's a work in progress. We are moving towards where we want to be. You can imagine, between moving from the version one of the platform, which is big monoliths to version two, which is mirco frontend and micro services on a backend and building new features at the same time. There's a lot of things happening all at once. So a lot of things that's just ben sitting in a backlog, but hopefully we'll pick it up soon.

Right now, we're aiming to have something that's good enough, and then as we get more and more – What's the right word for it? As the architecture gets more complicated and more complex and we're going to need more help, then we are investing time in the background to look at tools that are more fit for purpose.

[00:47:09] JM: Do you have a frontend crash monitoring tool to figure out when people's devices are crashing when they're consuming a lot of stream?

[00:47:18] LM: Yes. We're using Sentry for that. We are looking also to LogRocket at the moment. That is interesting, because with LogRocket you can basically plug directly to the statement admin system that we're using on the frontend. Also, we are using Google Analytics to understand – We are sending some custom events directly there.

That is for, let's say, the user interaction, is that for monitoring the data, for video players that we have SDK installed in each single platform that is from Conviva and retrieving all the data directly from the video players. Basically, the SDK is gathering all the information from the video player from buffering, video start time and other stuff, and we have full visibility on that on any platform.

[00:48:02] YC: I guess I forgot to talk about logging on the backend side of things. We're using logz.io. So all the containerized services, so Lambda writes their logs to Cloud Watch logs and then we have using Kinesis to aggregate orders, all the logs from all the different regions and then putting to one centralized stream in the ops account. From there, we pump everything into logz.io.

[00:48:25] JM: As we begin to wrap up, what's in the future for DZN? What are you guys focused on working on right now?

[00:48:30] LM: Currently, there are quite a lot of new features that we want to put DZN, because obviously our teams, product teams are always keying to interact with our user base.

So we are running a lot of experiments with them and we are gathering information, doing some panels directly in different cities where we can, let's say, gather some of our users and try different things and see how they want to evolve the product and stuff like that.

At the same time, let's say, we are currently only, if you want, certain countries, but we want to extend and basically become like Netflix and be a global organization that we'll be able to have this presence all over the world. Therefore, we'll have, for sure, new countries to deploy next year. Let's say, on top of that, obviously you have to think also a massive organizational growth happening since day one basically, because we moved from 30 people in day one for years ago to over thousand. So you can imagine how difficult it is scaling at this piece in such shorter amount of time and putting layer on top of layer and creating also some smooth discussion and constructive discussion between different departments. There is a lot of work on any part of DZN to be honest, and that's probably where the challenge is at the moment and also the exciting part.

[00:49:49] JM: Okay, guys. Well, it's been really great talking to you about DZN and live streaming and I look forward to following DZN and what you guys are doing in the future.

[00:49:58] YC: Thank you, Jeff. It was a pleasure for us too.

[00:50:00] LM: Thank you, Jeff.

[END OF INTERVIEW]

[00:50:04] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]