EPISODE 1466

[INTRODUCTION]

[00:00:00] JM: Real-time analytics are difficult to achieve because large amounts of data must be integrated into a data set as that data streams in. As the world moved from batch analytics powered by Hadoop, into a norm of real-time analytics, a variety of open source systems emerged. One of those was Apache Pinot. StarTree is a company based on Apache Pinot that provides fast, real-time data analytics.

Chinmay Soman joins the show to discuss Apache Pinot in relation to other real-time analytics platforms, and what StartTree has built on top of Pinot.

We are looking for a salesperson. If you're interested in working with us to help sell podcast spots, then send me an email, jeff@softwareengineeringdaily.com.

[INTERVIEW]

[00:00:43] JM: Chinmay, welcome to the show.

[00:00:45] CS: Yeah, thank you. Thanks a lot. It's great being here.

[00:00:47] JM: I'm really thrilled to talk to you because we're talking about a system that I am fairly unfamiliar with, which is Pinot. I did a show about Pinot, it must have been three years ago. I failed to look up – I can't remember who I interviewed. It was a guy at LinkedIn who had done a lot of work on it.

[00:01:07] CS: [inaudible 00:01:07]?

[00:01:09] JM: No. Not him. It was I think a guy with a European name.

[00:01:12] CS: Oh, Alexander Pucher maybe?

[00:01:15] JM: Yes. That's it. Alexander Pucher. I was very impressed with his knowledge of distributed systems and the analytics industry. But what I could not understand, and I think this was largely due to my lack of familiarity with the field at the time, was really what value Pinot brings to the world that doesn't exist or that did not exist prior.

So I was exploring Pinot in the context of Apache Spark, but more from, I guess, Apache Spark, Druid. What else is in this category? What's that? ClickHouse. Yes**.** Maybe what is commonly referred to as the open source data warehousing industry, right?

[00:01:58] CS: That's a broad term. I think where Pinot fits in would be on a slightly narrower domain, which is on the real-time analytics side of things, right? So you can think of – You have your traditional big data warehouses, which is you have the verticals of the world, or the open source technologies using Presto, Hive, and so on. And those are great at doing what we traditionally refer to as batch analytics.

So you have massive amounts of data sitting in your data lake. Most companies will just put all of the data in this data lake. And you want to do all kinds of analytics on top. You want to run your ML pipelines on it. You want to do interactive analytics on it and so on, right? So it's really good at doing that.

What these systems are not optimized for is real-time analytics. And by real-time, there are two dimensions which are relevant here. So one of the dimensions is the ingestion latency. So how quickly can the data be queried once it's produced at the source? That's what we call as the ingestion latency.

And as you may know, typical traditional data lake systems, the ingestion latency is typically in the order of multiple hours. It could even take a day. So data shows up a day later or many hours later and then you can query them. And you often want – There are many use cases where you want to quickly analyze the data as soon as it's produced, within seconds. And that's the ingestion latency.

And of course, the second dimension is the query latency. So you want to be able to query, do analytical queries on this data really quickly in the order of milliseconds. And a lot of this make

sense when I go through the use cases. And that's where the real-time analytics domain stands completely apart from the big data analytics.

So maybe we can jump into the use cases.

[00:03:57] JM: Yeah. So let's contrast what you just described with something like Snowflake, for example. With Snowflake, you have data that is produced – Let's say data is produced by analytics systems at a company like Uber or LinkedIn. You have vast quantities of data being produced, data streams being generated in rapid, real-time. They're probably being written to Kafka. And then you've got some kind of system that's like reading off of Kafka and throwing them into something else, which tends to be maybe S3. And maybe you're writing to S3, and in parallel, you're writing to some kind of data warehousing system. And then you've got like, I don't know, batch queries that are turning those raw data streams into materialized views that you're using in various systems throughout your company.

And what you're saying is that, with Pinot, you unify a lot of that into one system where you can write very aggressively such that you can have closer to "real-time" analytics.

[00:04:52] CS: That's right. Yeah. Pinot was built for working well with the streaming systems like Kafka, right? So that's a great example where companies are just pretty much all the data, all the events, are written to Kafka. And then they can be immediately ingested and queried via Pinot. That is correct. It was built specifically for the real-time aspect of it.

The Pinot also supports a component where you can bootstrap large amounts of data from something like S3 and then load them into Pinot as well. So it actually has that lambda architecture where you can get data from your real-time sources, get data from your batch sources like S3, GCS, or HDFS, and then put all of them under one logical abstraction.

For the same schema, you can have data source from S3 as well as Kafka and query that as one giant table, all right? Typically, how people use this is let's say you have a new use case and you want to analyze something like trips in case of Uber. You would put the last, let's say, three months or one year of data from HDFS into Pinot. And at the same time, you can have a real-time stream of trips coming into Pinot as well, which is ever-growing, and then query this as

one logical table, right? So this really hides the complexity of the lambda architecture from the user's point of view.

So this is where it sort of stands apart. Most data warehousing systems are able to do batch ingestion easily. So get data from S3 or HDFS, but are not able to – Are not really optimized for getting events from Kafka and be able to query them as soon as they are produced. So that's really where it stands up.

[00:06:46] JM: You know, what you've implicitly highlighted here is just how much data engineering innovation LinkedIn did. I mean, I think about the data engineering innovation at LinkedIn. You've got, obviously, Pinot, because we're discussing this. But more importantly, Kafka. At least more importantly today. Kafka, Samsa, which is probably underrated.

I don't know if you've heard these. But a few years ago, I did three shows. This was like my favorite piece of sponsored content that I've ever done. LinkedIn paid me some money to do three shows with LinkedIn engineers. And it was a blast. I did three engineers in one day. Sorry. Three interviews in one day. I think it was about Kafka infrastructure. Like, the actual engineering of Kafka. And then something else, just various data munching, and engineering, and stuff. And LinkedIn just did so much in data engineering at a very interesting time in the evolution of the world of data engineering.

Okay. So you were at LinkedIn. Let's see. I have your LinkedIn profile right here. So you're at LinkedIn from 2012 to 2014. Oh, no sorry. Earlier. You were there from 2011. Oh, Voldemort also. Voldemort was like key-value store, right?

[00:07:51] CS: Right. Yeah, I started out in that project. The same person who wrote Kafka actually wrote all the model before that, Jay Kreps. And it is a key-value store meant for tracking things like number of likes, number of shares. Where you don't necessarily care so much about consistency, but you really want to be highly available. That was really the premise of the system. So it was optimized for AP. So availability and partition tolerance within the CAP theorem.

[00:08:21] JM: I love Jay Kreps. And I think he might dislike me, because I criticized him on stage for changing the license of Kafka.

[00:08:29] CS: Oh my God. That's –

[00:08:31] JM: Wasn't that a huge mistake? That was the biggest mistake he's ever made, I'm pretty sure, as far as I can tell. This is a guy who does not make mistakes. And he made one very big mistake.

[00:08:41] CS: I mean, you have to think about – His passion right now is the company that he's building, and obviously the license plate is a big deal – The Confluent license is definitely something that they want to control as a company. But Apache Kafka is still Apache Kafka. That hasn't really changed, right? I think what has changed is the other parts, which are owned more by Confluent. And that sort of makes sense. It's meant for the likes of Amazon shouldn't just come in and start serving it out on their own without really being part of the community. I think it's more catered towards those kind of things.

And since we are a company around Apache Pinot, we sort of have similar thoughts about the Apache Pinot open source makes sense. That will never change. But some things are more proprietary, and that should be a different license, in my mind. So I'm with Jay on some of those things.

[00:09:39] JM: But this is the whole fundamental mistake, is when you change your license, like Confluent did, you're implicitly saying, "We don't have a product development workforce.' You're saying that our only product is going to be Confluent in the associated services. And somehow, AWS, with their random team that's working on Kafka, is going to be able to compete with an entire multibillion-dollar company that's devoted to building a platform on top of the queuing system. It's just not a realistic paranoia. It kind of undermines the confidence of the company.

[00:10:13] CS: Yes. It might appear that way. But keep in mind, most of the innovation is still happening in the open source Apache Kafka, right? As far as I see – And same thing with a lot of the innovation is being contributed back to open source. And every day we take a really hard look at what feature goes well. And the principle is community comes first.

If something is good for the community – For example, the Zookeeper-less Kafka would be one example. On the Pinot side, we added a feature called Upserts, which allows you to upsert data

for an analytical database, which seems trivial if you're coming from the OLTP database land. But it's really hard to do in case of analytical databases.

And so this is a really fundamental feature that we supported, we added in the open source land, right? But there are some things which are just proprietary which might be needed for big companies or some of the specialized use cases. There might be more product focus on some of these components, which are not in the open source, right?

And I don't want to speak about Confluent. I'm definitely not in the position to do that. But my feeling is it is a complex problem, right? We need to, as a company, be able to control the roadmap and things like that. But at the same time, it should also be source available so that other people can at least – If they want to, they can still run it on their own.

It's a tricky position of how you want to influence the roadmap for these products. How do you want to grow the community around these specific proprietary components? And at the same time, make sure you have an edge in the market. So it's not an easy answer.

[00:12:03] JM: It is an easy answer. There's no reason to change your license. You're a technology company. You can pivot. You can do new products. It doesn't make any sense.

[00:12:11] CS: Okay. All right. Again –

[00:12:15] JM: It's a topic for another show. I'd love to have that conversation with Jay or somebody else. But I'll just say publicly on air, Jay, you're an inspiration. I love you. But you made a bad decision.

[00:12:26] CS: Fair enough.

[00:12:28] JM: And no hard feelings, I hope. Okay. Let's talk more about like real-time analytics and stuff. This is a pretty critical idea. Like, the fact that you can generate data on the fly and write it very quickly and read it very quickly.

I've done enough shows with various distributed systems to know that this is not easy to do. Actually, you know what I'll say, is it's not even that hard to do, I think. Conceptually, it's

obviously possible. What I find interesting about LinkedIn and the stuff you guys built is, really, what you had to do is just lay out the spec, right? You just had to say, "This is what we need." And it's very interesting you were able to define the spec, which is effectively we need a system where you can write quickly and read quickly, basically. And once you define that system, there's a whole lot you can do there. You can build what would commonly be called a data analytics or data engineering data warehouse thing. But it's actually like kind of novel to even think about as like a transactional database. I don't know if you can actually do that as a transactional database. But what you're defining there is a very interesting set of distributed systems properties.

[00:13:34] CS: Right. Yeah. In hindsight, it looks like we sat down and we wrote down all the spec. But I think the reality is things organically happen. So we started out with massive amounts of data coming from Kafka. Kafka really took off in LinkedIn bringing in events of all kinds; application logs, system logs, business events that were being produced by all kinds of microservices. And then there was a real need to start analyzing this data as soon as it's produced, right?

A good example was people you may now, which was as soon as some things happen if someone likes a profile, or visits a profile, or even the analytics around your posts, those are real-time events. So you post something on LinkedIn. You want to see how it's trending within your network. And specifically, what categories of people are watching the content?

So as soon as the events are being produced, you want to be able to group a breakdown of these views over a different industry, or geographical locations and so on. Same thing with ad analytics. The ads being shown on LinkedIn, you want to be able to see in real-time how effective it is. And who's clicking? How many people are viewing? What is the ratio? And so on. So all this data was already in Kafka when these questions were being asked. So it was natural to say, "Hey, we need something that can ingest from Kafka at a very high rate and be able to query very quickly."

And at the time, there was no need for a transactional semantics. When Pinot was originally built, it was not – In fact, it was designed to be not a transactional system. And a lot of the assumptions in the system were made to make it very simple. Events come in and they are

append only. And most of the segments that are created in Pinot are immutable, right? So the data never changes within Pinot. That makes it a simple system to operate and at scale.

So the initial adoption of Pinot was around analytics where consistency is not very important. But then over the years, especially at Uber, people really started caring about consistency. Because within LinkedIn, if your number of likes on a post are 10 versus 12, who cares? But in Uber, they really started to care about the demand supply ratio of drivers and riders within a specific geographical region, or the sales, or the gross bookings that we're having within specific areas, right?

And they really started to care about consistency of data in that case. And uber is really interesting, because a lot of the data keeps changing, keeps getting updated. So if you do a trip, the cost of a trip might actually change over the next few days.

So then we started to – The new requirement came in, where not only should you be able to write data fast enough and query data fast enough, but the data itself must be consistent. So if you're updating the data, it should be reflected in your queries. Otherwise you're just going to double count and get inaccurate results, right?

The transactional semantics really came in from companies like Uber. I guess the meta point there is the spec or the requirements actually happened organically as the problems became more and more complex for us to solve.

[00:17:05] JM: Let's kind of fast forward a bit. You've got a super interesting background. And, actually, I was looking for LinkedIn for the first time right before the show. And we can definitely do additional shows. You've got so much experience. You have three-and-a-half years at LinkedIn during some of the most critical times of the company. And then almost six years at Uber, which is incredible. And then you leave Uber and you started StarTree almost immediately. So you are relentless, which I love.

Let's get to StarTree pretty rapidly. Just talk briefly about like your experiences at LinkedIn and Uber and how that took you to – I'm sure you had a bajillion business ideas. Or at least more than one business idea. Why did you think StarTree was the best way to go?

[00:17:50] CS: Right. So let me start with LinkedIn, which really opened my eyes to distributed systems. You're right, that was a fascinating time. I think, at the time, we saw that the big data problems were all showing up across multiple areas, real-time analytics, batch analytics, stream processing, messaging. And at the same time there's not many open source solutions out there. So it was a fantastic time and place to be in LinkedIn to get to work on all these things.

As you already mentioned, it was the birth of Kafka, Samsa, which is a stream processing engine. And I was intricately part of that team, and also things like Apache Pinot. It gave me a very good foundation of, I would say, real-time analytics, which is not just Pinot, but also includes Kafka and Samsa. And these are all the moving parts of the very complex real-time analytic problems that we need to solve.

So with that good foundation, I was, again, fortunate to be in a good place at Uber, which really pushed the need for real-time analytics, much more so than LinkedIn, to a point where if some of our analytical pipelines were not performing well, we would actually lose a significant amount of money because of that, right? So the analytics done at Uber was really business critical. And it was not good to have in that case.

At Uber, I started with just building a system to compute metrics, business metrics, at scale, right? You can do slice and dice your different metrics, like number of trips, gross bookings and so on, and slice and dice by geographical location, or obviously time, the type of device that you're using, and then so on, right?

And pretty soon – Initially, the system was used by execs and PMs to do future planning and think about financial incentives. And which areas do we need to invest in more? But pretty quickly, we got into fraud detection. We got into real-time incentives for drivers. And then, of course, doing things like search calculation and so on.

What we saw at the time is analytics was going from an offline good to have use case, to a much more business critical user-facing use case. We saw a good example being restaurant managers, where the analytics on the Uber Eats deliveries were being exposed directly to the restaurant owner. That's very transformational for such restaurant owners, right?

Initially, if you can think about how analytics was done a couple years back, someone internal would run complex queries, generate reports, and then share reports to the restaurant owner. And the restaurant owner only got to know what happened a day later or almost a week later.

With real-time analytics, the restaurant owner knows exactly what is happening at this point in time. How many inaccurate orders, or missed orders, or menu popularity is true for that point of time? And that's being powered by Apache PInot today.

And another example being Orders Near Me feature. If you use Uber Eats app, you will see an Orders Near Me of like what is the most popular items being ordered around me in that particular geographical region. And this is something also being powered by Pinot.

What I saw was the role analytics was playing is getting more and more critical for the success of the business. And the requirements for these analytical systems were also getting more and more complex. We went from internal analytics. Internal analytics is where the data scientists or engineers, which are like hundreds of them, will be querying your system. Versus external analytics, where all the restaurant owners, which is half a million of restaurant owners, or 700 million LinkedIn users, were directly querying your analytical database. And those are external analytics.

I think that's where we saw there's a real change in how real-time analytics systems are being positioned in a company's technological stack, right? So it was clear that you need a solution, which can sustain tens of thousands of QPS for an analytical database, and query latency within, let's say, millisecond P99. That’s the scale we started to see within LinkedIn and Uber.

And, again, this might seem trivial if you are dealing with MySQL and Postgres. They already do that. But for analytical databases, that was never the case. They were always mostly positioned as an internal tool rather than an external-facing tool. So that's where we saw there's a real opportunity in Apache Pinot to be the analytical database that can support such external or user-facing analytics.

It was an obvious choice for me at the time to see – LinkedIn and uber saw this before other companies. But every single company is going to need this at some point of time. So if you talk about personalization, every single company will need to personalize the user experience. And

if you really drill into personalization, it will turn pretty quickly into, "We need to do analytical queries per user or per segment of users at scale," right? So it boils down to, again, external-facing analytics.

So that was the motivation behind, "Let's build a company around Apache Pinot and then make it available for those who cannot run this on their own, really." Right? So that's how StarTree began. The mission of StarTree being to make Apache Pinot an easy to use experience for all kinds of companies.

[00:24:07] JM: What's the best way to productize something like Pinot? I've already pointed out my critique of the contemporary theory around building an open core company. What's the best way to productize this stuff from your point of view?

[00:24:22] CS: Yeah, that's a great question. And then again, this is something –

[00:24:25] JM: By the way, I was looking at your Crunchbase. This is totally unrelated. I was looking at Crunchbase. Bane is doing a great job. It's kind of random. But Bain does really good investments in data infrastructure and related highly technical engineering evaluations.

[00:24:40] CS: Yeah. I think it's part of the new line of thinking. It was not traditionally investing in such companies. But recently, I think they are looking at the core technological companies that can, as I mentioned, be transformative for a lot of these businesses and customers. But, yes, it's great to have them back us up. It's a good company for sure.

Regarding the product angle, that's a great question. I think we have – This question, debate over it every day. It is a hard question. With open source technologies, as I mentioned, every day we have to think about, "Hey, we are building this new feature. Where should it go? Should it go in the open source land? Or should it be sold as a paid feature?" And this is one angle. The other angle is everything is in the open source. And then customers will pay you for support, which is also fine, right?

And then we are still early in this journey. The way we are thinking about things is community first, right? So if a feature is going to benefit a large section of the community, that's an easy answer. Upsert was an example where it's just contributed back to the open source.

And then where we focus on the product side is how do we make it really easy for people to use Pinot, right? So that includes deployment, and installation, ingestion of data. If you have played around with Pinot, there's a lot of things that you need to deal with for getting data into Pinot, right?

For example, the data from Kafka may not be in the right format. And it might be highly nested. It could be some of the columns are missing. The time column may not be in the right format that you want and so on. And you have to use a bunch of things to get it easy to query within Pinot, right? So this includes Pinot has all these inbuilt tools to simplify ingestion. But you need to be able to configure it accordingly.

So there is a little bit of a ramp up on new engineers or people like data scientists to start using Pinot. As a company, we focus on making that experience really easy. How can our data scientist who has no knowledge of ingestion transforms within Pinot still be able to use that and be able to do his or her job, right?

In some sense, we want to get Pinot out of the picture and focus on the problem. The problem being your data is in Kafka and you want to get this metric out. What is the easiest way to get that? Or you want to serve a personalized feature to the user from your complex data sitting across S3 and Kafka, what's the best way to get there, right? The product angle is to tie up those loose ends and then give a higher-level abstraction to the users, if that makes sense.

[00:27:38] JM: Yeah, definitely. How would you compare your approach to – What's the Druid? Imply.

[00:27:43] CS: Imply? Yeah. Imply – And first of all, Druid and Pinot are quite similar in terms of architecture. Also, the focus on low query latency, obviously, some of the things are different. Pinot was more optimized for user-facing external analytics. Whereas Imply focused – I guess the focus was more internal analytics.

Between imply and StarTree, I think the approach is, again, very similar. They're community first. So most of the things – Make sure the open source project is successful. And as soon as that happens, obviously, the large companies are going to run there on their own. But a lot of

the small companies and even non-technical companies will need paid support to reuse a managed service. The approach is very similar across both these companies.

I think where the differences lie are in the use cases, as I mentioned before. Pinot will focus more on personalization, feature store for ML, large scale user analytics. That's really the focus, right? And our selling point would be one click experience for any user, whether it's an engineer, whether it's a data scientist, whether it's an exec, or a PM to be able to do this quickly. I think that's really the selling point. So we're not catering specifically for engineers. That's really the philosophy behind StarTree.

[00:29:12] JM: How does the go-to-market compare then? The customer base, are they similar? Are they overlapping? Do they want both StarTree and Imply?

[00:29:27] CS: Well, no. You will either have Imply or StarTree, right? The way I think about it is companies will think about their hardest problems, right? If the hardest problems are personalization, that's a hard problem.

Let's take an example of personalization, right? So in case of, let's say, LinkedIn, LinkedIn uses Pinot to personalize the news feed that you see on your LinkedIn homepage. And by that, I mean, there's a lot of relevance and ranking that goes on behind the scenes. And it's using Pinot to do things like how many times has a given user seen this news item in the last 14 days or so? And that becomes an input for the real-time ranking. As things are changing, as user are viewing stories, the news feed keeps evolving. And this is a way of personalization.

Another way could be financial incentives for Uber drivers, right? You have all the features from drivers being collected in real time, as well as historical features. And you use those features to make decisions around how much incentive should we give to a driver for them to continue driving for Uber? Or how much discount should we be giving for eaters to be able to continue using Uber Eats, for example?

And for these features, you're really looking at massive amounts of data, all the Uber drivers, and riders, and trips, and everything, and you want to do them at scale. Every time an eater even opens the app, you want to be able to run these queries, which translates to multiple tens of thousands of QPS.

In the end it boils down to how much QPS you're doing. What is the complexity of queries? And what is your low query latency requirements? And I think that's where companies will make a decision on using Pinot versus other solutions.

For these large-scale use cases, they are going to opt for Pinot, because the other solutions don't compare in this spectrum. Also, the good thing about Pino is if you can do the complex use case, you can also do the simple use cases, which is the internal analytics. Our pitch is really Pinot is the solution you need for real-time analytics.

[00:31:51] JM: I have this theory. You know the Michael Stonebreaker idea, the age of one size fits all is over? Have you read that paper or at least seen that paper? You get the idea?

[00:31:59] CS: Yeah, yeah, of course. Yeah.

[00:32:00] JM: I feel like we're going through like the age of one six fits all is over 2.0. Basically, the same thing is happening for data warehouses. You want domain-specific data warehouses these days, right?

[00:32:12] CS: Yeah. True. And I didn't mean to say we position Pinot as the only analytic system. Just to clarify, you do need your Snowflakes. And you do need your Hive, and Presto, and Tableau, and Looker, and all these things are needed. By all means, I mean to say, Pinot will replace them all.

But where we are focusing on is there is a big gap in the real-time data analytics component, right? That's the gap we are trying to fill. So we are not going after the batch analytics workloads. The stuff where all your ML pipelines run, or you're generating reports for your interactive analytics, or ad hoc analytics even, where your data scientists are trying to do big data exploration on massive amounts of data. All those things are definitely needed.

What happens typically is once you do all these data explorations, you start thinking about, "Okay, I have these features. Or I have these metrics that I want to compute in real time." And that's where, suddenly, you see a lack of a cohesive platform or a system to be able to do that.

Traditionally, yes, people have been using – I'll give an example, right? Before we adopted Pinot in Uber, the way they we're doing metrics was to pre-compute everything. They know what metrics they want to serve. Then they hired a bunch of engineers to go write these stream processing pipelines that will crunch all the data and then pre-compute metrics across different segments, right? And then these metrics were basically simple aggregations over different time buckets. Like, 5 minutes, 15 minutes, 30 minutes and so on. And then they build a query system on top to be able to combine all these buckets based on the given time range. So they really build all these complex moving parts to solve a real-time analytics problem.

And what we're saying is, with Pinot, all these complexities will go away, right? You just need to point to your input Kafka database. And you don't need any precomputation. You don't need an additional investment in data engineering for real-time problems. You can just query the data on the fly. I think that's really – It's actually not a generic problem. It's a very specific real-time analytics problem, which we are trying to focus on.

[00:34:42] JM: Okay. Yeah, that makes complete sense. But, really, the thing is, like, you've said, that Druid is effectively a competing system with Pinot, right?

[00:34:51] CS: Yes. Right.

[00:34:52] JM: Are you sure about that? Does Druid really have the same spec as Pinot?

[00:34:52] CS: Yeah. I mean, at a high level, Druid can consume from Kafka. They both support analytical queries. It is meant for – There is some overlap, of course. It's heavily optimized for internal analytics and be able to serve such queries over fresh data, right? The freshness, query latency, those things are similar.

I think the parts where they don't overlap are the external-facing analytics. That's where, I think, Pinot was built from the ground-up for supporting these massive QPS and low-latency queries. There is a performance difference. There is also a difference in some of the use cases.

For example, we recently added geospatial indexes to Pinot. Now you can do geospatial queries directly in Pinot. Or the fact that upserts are now available in Pinot, which is not the case in Druid. So you can actually have data, which either have duplicates or have updates in

them and be able to get consistent query results out of Pinot. That's actually not true in Druid or Clickhouse. There are definitely a lot of the feature differences as well as performance differences.

But if squint your eye, and at a very high level, they're all trying to do t he same thing. They're all trying to give you ability to run analytical queries on fresh data. That fact doesn't change. They are all competing in that domain.

If you want to build a dashboard, a simple dashboard that plots metrics for your real-time data, you could actually build with any of these systems. But now the question is you know you want to now expose that dashboard to your end user. Can that be supported on Druid and Clickhouse? Probably not. Because now you suddenly grow from hundreds of QPS to 100,000 QPS, because you have directly enabled it for your end user. And that's where I think you need to make the right choice of what is your end audience for your real-time analytics.

[00:37:03] JM: All right. Well, we only have like 10 minutes left. But I'd love to do another show as soon as possible. Let's close out with some like macro level stuff. You saw the whole streaming versus data warehousing thing, which surprised us all, right?

I mean, I don't know about you. I thought streaming systems were going to be the new hotness. It appears they're not. Nobody's using streaming systems. Most people today probably have not heard of Apache Beam.

[00:37:29] CS: Yeah. I wouldn't say nobody's using it because – I mean, for example, Lyft and Uber both are using Beam to a certain extent. You're right, though. It hasn't uh replaced the old systems, if you may. I think that hasn't happened. In that sense, you're right. And I can talk about my thoughts on why that is the case. But did you have any other question there? Sorry to cut you off.

[00:37:58] JM: Well. No. No. I mean, I just find it – It's really interesting. I feel like streaming systems will have their day in the sun. Maybe it's that company, Materialize, right? Isn't that an interesting – Wait.

[00:38:10] CS: Yeah. Materialize is making –

[00:38:13] JM: Wait. But Materialize, they're not open source, right?

[00:38:16] CS: No. No. They're not.

[00:38:18] JM: That's crazy. Wait. They have a GitHub. Wait. Materialize cloud is now available. They have a GitHub. But is it open source?

[00:38:25] CS: Last I checked, probably not. But I could be wrong. I think they are trying to make the stream processing a one-click magical experience for the users, right? So you don't need to write code. You can basically write a query to do –

[00:38:41] JM: Yeah. I don't know. Whatever. It's not good marketing. It's not good marketing.

[00:38:45] CS: We have been down that road. In fact, a system that we build, AthenaX, in Uber. You should definitely check that out. Bad name. It's a terrible name. But that was one of the first systems to actually do –

[00:39:01] JM: It sounds like a pharmaceutical drug. AthenaX.

[00:39:05] CS: It's also confusing, because there's Amazon Athena, which is pretty different than what we are trying to do. But we started out with the goal of making SQL as a first-class citizen. And our end vision was this is going to replace everything. If people can write SQL – And here's a Kafka topic, write the SQL, and I'm done. We really thought this is going to dramatically change things.

And then Uber, a lot of things did dramatically change because of that. We saw engineers and operators. Uber operators are people who don't necessarily have engineering background. They were able to write complex pipelines with using SQL, because everybody knows SQL. So we did see a lot of that improve significantly.

But it's an 80-20 rule. A lot of the common problems are in the 80% of the problems. But the remaining percent are just SQL may not be enough. It's just too complex to author in SQL. And then you have to bring in all these other technologies to solve the problem. That's one.

We also saw push versus pull analytics, right? Streaming is more push analytics, where you have data continuously coming in. You do some processing. You make a decision. And then you take an action based on that decision. This is what I refer to as push analytics.

The problem with push analytics, it works great when there's not much state to reason about. If you are maintaining a state for like 30 minutes – You're getting data in. The state stays there for minutes, and then you flush it out. That works great. But if your state grows to let's say one week, these systems suddenly don't appear as useful. Either they are too costly to operate, or they just don't work for such a massive state when you want to make a decision. And that's where like you need things like Pinot, where Pinot can handle a lot of state. And a lot of these same queries can be executed in Pinot to more like a pull analytics. You put your data in Pinot and you pull your insights out of Pinot, right? I think you need both.

For things like fraud detection or alerting system, you need push analytics, because you need to continuously process the incoming events, look for anomalies. And then as soon as you see something, you trigger it. So you take an action. That's why you need the push analytics.

And then on the pull analytics side, this is where your dashboards come in. Massive state. It's not continuous. It's based on your applications or user interactions. And that's why you need Pinot.

But none of these will replace, as I mentioned, the batch analytics, which is much more complex, right? You need running regression analysis on something like this is just unthinkable right now. And it has to do with how easily can you store petabytes of data in your streaming systems. How easily can you run complex computation in your streaming system? I think it is still growing. And one day, it might come to a point that everything is streaming. But not today. I think we have ways to go for that reality to happen.

[00:42:22] JM: Okay. Well, we're very close to the end of our time. Any last reflections on building a data engineering product in 2021 in two minutes or less?

[00:42:36] CS: Yeah. Obviously, lots of options out there. There's not one thing out there. We need to focus on the right value for the customers, right? A, for any of these products, open

source is key. Making it extremely easy for a large community of users is very important in my mind. If you make your open source project successful, the company will naturally follow through. That's one.

Focusing on data quality, consistency, these are the problems which have traditionally been ignored in data engineering. I think those are the new things that are not exactly new. But on the real-time data engineering pipelines, I think that's the new focus. We do see that for renewed interest and accurate real-time analytics, which was never the case before.

And then third, it is going to take time for – All these systems are immature, in my mind. They're all starting out new. If you think about MySQL and Postgres, it took them decades to get to where they are. And my thinking is the real-time analytics system is still young. We still need to make significant investments no matter what open source technology you're talking about to be able to get to the majority of something like MySQL or Postgres. I'll end with that thought.

[00:43:57] JM: All right, man. Well, until next time. I had a blast. I look forward to learning more. We should do the next one in person.

[00:44:01] CS: Yeah, definitely. I would love that. I've been a fan. I mean, obviously –

[00:44:05] JM: Oh, no way. No way.

[00:44:07] CS: I should have started out with that.

[00:44:09] JM: No, no, no.

[00:44:09] CS: I definitely enjoy hearing about all your stories. It's fantastic. Very, very happy to be part of this.

[END]