EPISODE 1458

[INTRODUCTION]

[00:00:00] JM: The Presto and Trino project makes distributed querying easier across a variety of data sources. As the need for machine learning and other high-volume data applications has increased, the need for support, tooling and cloud infrastructure for Presto and Trino has increased with it. Justin Borgman is the CEO of Starburst and joins the show to talk about the impact of presto, and how his company has architected its product.

[INTERVIEW]

[00:00:22] JM: Justin, welcome to the show.

[00:00:24] JB: Thanks, Jeff. Super excited to be here.

[00:00:28] JM: Yeah, it's great to have you back. To start off, we covered this in the last time you came on. But I want to get a sense for how Presto, which is, obviously the focus of Starburst, or well, Trino now, how a data infrastructure built around Trino compares to a data infrastructure that is perhaps more segmented and disparate, perhaps a data infrastructure that doesn't have some kind of a unifying system for data aggregation.

[00:01:09] JB: Sure, yeah. I mean, I think that's one of the special characteristics of Trino and Starburst, is the fact that you can query data anywhere. That really allows new possibilities. I think, pretty much every organization, large or small, has data silos of some kind. It's obviously magnified for companies at significant scale, and particularly those that have been around a long time, a big bank is certainly going to have this in spades. But even small digital native startups have this challenge as well, where they've got some data in MongoDB. And they've got some data in Redshift, and RDS, and Snowflake and S3, and so forth. I think the reality is, the traditional model of data warehousing has always been predicated upon this idea of data movement, extracting it from these data sources, maybe doing some transformation, and then ultimately landing it somewhere to be analyzed.

What makes Trino and Starburst so unique, is the fact that you don't have to do that anymore. ETL becomes an optional phenomenon, and you can actually query the data where it lives, you can choose

what data you want to keep in a relational database, and what data maybe goes into a data lake. Of course, a data lake is always going to be great from a TCO perspective, from an economics perspective. You're going to save a lot of money by saving data in a data lake. But you have the flexibility. I like to say, we give our customers optionality to decide how they want to design their data architecture at large, and it can evolve, and Trino and Starburst can evolve with it, because really, any new data source you add is just another node in the overall design, if you will.

[00:02:56] JM: Describe what happens in a typical Trino query, and maybe talk about some of the typical use cases for a Trino query.

[00:03:08] JB: Well, I'll speak about a data lake use case first, just because that's maybe the simplest and easiest to comprehend. It's probably also one of the most popular use cases out there, and that is to say, you've got data in s3, stored in some type of open data format, maybe it's Parquet, maybe it's ORC, maybe it's Avro. Maybe it's one of these newer formats, like Iceberg or Delta or Hoody. Regardless of how you store the data, one of the nice things of storing it that way in S3, or whatever data lake you have, is it's an open data format. So you can use multiple engines to query that data. You kind of have freed yourself from vendor lock-in to a certain extent by doing that. You could use Spark on the same Parquet file that you use Trino or Starburst, which is a nice benefit of storing data that way.

Now, the way a query gets executed is the SQL query comes into Trino. Effectively, it goes through a SQL parser. There's a query optimizer that decides the most efficient way to execute that query. And then it goes and effectively executes the query on the data in the data lake. It's executing that query in parallel, in memory, in this sort of scale out architecture. So Trino and Starburst is an MPP design, Massively Parallel Processing design, very similar to, let's say, a Snowflake. Its overall design, the key difference being that Trino doesn't have storage. Snowflake does have its own storage. It's separated, but it's its own proprietary storage.

In the case of Trino, there is no proprietary storage. The storage is whatever you point it to. So essentially, you've got a scale out cluster that is going to execute that query in memory, deliver the results, and then it's done. Nothing's persisted, nothing is really moved other than into memory just for the purposes of executing that query. It operates incredibly quickly and at Internet scale. It's run today by many of the leading hyper scalars out there.

[00:05:04] JM: How has the query processing the query architecture of Trino changed over time? Can you describe some of the – I know you're not probably as deep in the weeds in engineering, because you have a lot of different hats to wear. But maybe you can talk about how the platform has evolved over time.

[00:05:24] JB: Yeah, sure. Happy to the extent that I can. You're absolutely right. My co-founders who are the creators of Presto and Torino, certainly could go into much greater detail, but I'll do my best.

One of the biggest advances was probably four years ago, when we first started Starburst. In fact, one of the first things we did is we built a cost-based optimizer. And the cost-based optimizer or CBO, as it's known, is sort of like the brains of a query engine. It's the brains of a database system. It effectively decides the joint ordering, exactly how the sequence of steps of how a query plan is going to be executed. For the most optimal way to do that, from a speed perspective. So it basically improves performance dramatically. And the moment that we introduce that first generation of the cost-based optimizer four years ago, performance jumped by 10x, basically, if you were doing joins. The more complicated joins, the more performance improvement you saw, as a result.

So that was one major milestone four years ago. And I think that milestone actually drove even greater Presto and Trino adoption, when we made that contribution. More recently, some of the things that we've been working on, are adding caching capabilities, the notion of materialized views, the ability to also run long running in a sort of batch oriented style queries, as well, because one of the traditional drawbacks, if you will, of the Trino architecture is that it didn't have query fault tolerance. Meaning that if you ran a query that was going to take an hour, and during that hour, one of the nodes in the cluster failed, you'd have to restart that query. And that was kind of by design initially, because really, Trino was intended to be for fast running interactive queries that might take a second or three seconds to return.

The likelihood of a node failure was de minimis. Even if it failed, restarting a three-second query is no big deal. But what we started to see was actually more and more users, and more and more of our customers started to actually use Trino and Starburst as an ETL tool, surprisingly. So even though we make ETL optional, when people were doing ETL, they were deciding to actually use Starburst to do that, particularly in the T of ETL, the transformations. S what we found was the workload mix was expanding from faster interactive queries, which it's always been known for, to now include some of these longer running batch jobs. As a result, there's been a lot of work more recently, to introduce

query fault tolerance for Trino, as well, which means that going forward in the next few years, I think you'll see more and more workloads for transformations for these longer running queries being done on the platform as well.

[00:08:16] JM: So, as the platform has evolved, you not only have to make engineering advances, we have to make advances on the UI side, the design side. Can you talk about the interface with customers, and how that's advanced the kind of the platform, the interface platform?

[00:08:40] JB: Absolutely. And there have been a couple of phases of that development. One of the first things that we did in the last few years is create something that we call our Insights Platform, and Insights is really data about your Trino cluster and what's going on. So trying to understand where resources are being consumed, how long queries are taking, what users are maybe drawing the most resources from the system. So you can go slap some people on the wrist, if that's what it takes.

All of that instrumentation, that ability to really monitor what was going on in the cluster was done at the command line previously, or captured in log files, and we were pretty quick to try to turn that into a more visual presentation that a data engineer or data administrator could really view and understand what was going on in the cluster.

We've taken at another step forward with our new SaaS product. We now have a cloud-hosted product called Galaxy. And Galaxy is intended to basically make Trino easy. Trino's incredibly powerful. It's always been a very powerful platform, but not always the easiest to deploy, and Galaxy makes it easy. In fact, you can get up and running in five minutes and that's no joke. You're welcome to try it, for those of you, may be listening in, there's a free tier, so you don't have to pay anything you can. You can play around and try it out.

Through that, the user experience is a critical part of the entire product experience. Because again, we're really aiming to make this easy to use, easy to manage, easy to just start running queries. So we've actually added an IDE within the platform, so you can literally start running queries directly within the platform. You can, of course, still connect it to your favorite tools, whatever that might be, whether it's Tableau or Power BI, or Superset, or Looker, or whatever. But for the power user that just wants to log in and start firing off some queries and joining tables, you can do that directly within the interface now.

Galaxy has given us kind of more opportunity to really enhance the overall experience and just make it easy, simple and fast to get value out of.

[00:10:55] JM: There are a number of other modern data tools that have different ways of unifying the data stack. I think a lot of those tools are sort of modern ETL tools, or data conveyance tools, I think of things like Fivetran, or the open source, various open source Fivetrans, and there's also things like RudderStack or other customer data platform systems. I wonder if you see opportunity since you have in Starburst, basically a connection point between all of these different areas of the stack, basically all the areas of the data, the data layer. If you see an opportunity for not just querying, but movement, and ETL.

[00:11:56] JB: Yeah, absolutely. I mean, ETL just broadly, if I can make a comment, I think on like the role it's played in the industry for decades now. Going all the way back to like Informatica and Teradata, in the on-prem world, a couple decades ago, has always been a pain point, because of the time involved. And not just the processing time, that's actually probably less of a concern. But more so, the human time involved with creating pipelines, maintaining pipelines. You add a field and a source database, now you get to add that field in your data warehouse. There's just a tremendous amount of work and friction that basically slows down the overall, what we call time to insight. The amount of time it takes from data being created to when data can be understood. So ETL, has introduced friction in that process for decades.

With the advent of cloud data warehouses, many of those ETL tools have just been basically reimagined and recreated in the cloud context. But the overall model hasn't necessarily changed all that much. Our view of the world is one where the customer has the opportunity, the user has the opportunity to decide, is this something that I want to move? And again, like, what would drive a movement decision? I would say, probably, primarily, economics. I want to move this into a data lake, because that's going to be the lowest cost place for me to store the data, just inherently. S3 is going to be the cheapest place for me. Or perhaps a data quality pipeline, like you want to do some transformations and landed in a particular place where that becomes more of a gold standard. Okay, those might be reasons to do that.

But we're giving customers the opportunity to say actually, we're going to bypass it. We're just going to go directly to the data source and query it where it lives. So, that's kind of the paradigm shift, is really inverting the data warehousing model, which has always been about centralization and saying, actually,

you can operate in a decentralized world. That paradigm now actually has a name. I wish I could say we created it, but we didn't. There's a woman named Zhamak Dehghani, who coined this term of the data mesh. And that data mesh concept is now starting to gain momentum as a as a new way of just thinking about your data in a decentralized fashion, and being able to skip ETL.

The other thing I would say is, it's probably important to distinguish between the E and L from the T. So ETL, of course, is extract, transform and load. The extract and load are probably not the highest value parts of ETL. It's really the T, the transform. So, some of these technologies are really focused on the transformation, like DVT, I would say, plays in the T part of things. Whereas some of these other tools are more focused on the extract and load and that's the piece that we think is probably the most optional in a world going forward.

I think what's core to our belief anyways, is this notion that we just think centralization has never been possible in data history and why would it be possible now. I spent a few years at Teradata. Teradata bought my first startup back in 2014. And Teradata was the pioneer of the data warehouse model. They invented the single source of truth concept. While I was working there, none of their customers, and they were very big company at the time, when I was there, $2.7 billion in revenue. None of their customers had actually consolidated all of the data in the enterprise into one enterprise data warehouse.

So if the leader in the industry couldn't do that over decades, with its customers, what changes now when we have arguably even more fragmentation, even more heterogeneity in our data ecosystems. There are more tools, more databases, each one purpose built for a particular job. I think that's kind of at the core of our thesis is that that's not going to change, that you're not going to be able to get your whole world wrapped into one particular data source to analyze it. If we acknowledge that reality, how can we manage that? How can we manage a decentralized world? And that's really what a data mesh is about, and that's really, I would say, what we believe the future will be. So even though we're really good at data lake analytics, and that's a core use case, this notion of being able to extend beyond a data lake and access different data sources, we think is going to be more and more important over time.

[00:16:24] JM: You mentioned building a cloud-hosted product, that's a notoriously difficult phase of any open source enterprise product. Can you talk about the most difficult portions of building that cloud-hosted platform?

[00:16:42] JB: Yeah. First of all, I'm laughing only because you're right, it is difficult and difficult is maybe an understatement, at least for us, it was. It never sort of goes exactly the way you planned, I think the first time you conceive of it. So in the moment that we raised venture capital. And again, just as a quick refresher for maybe some of your listeners, we started as a bootstrap company four years ago, we didn't raise any venture capital. We built a profitable business, just serving members of the open source community, around Presto, and what would later become Trino.

That was a really fun period of the business, not having any external expectations and running, again, a profitable business during that period of time. But two years in, we felt like there was an opportunity to build a really big business, and that necessitated taking capital. So the moment that we did, we said, “Okay, we're going to build a cloud product.” That was two years ago. It took us almost a full two years, we just launched our cloud product in November of last year. It's now generally available on all three clouds. But that took almost two years, from the day that we raise capital. I mean, almost exactly two years.

Our plan was, we thought we could do it in one year, it ended up taking twice as long. So maybe for anybody thinking about building a SaaS platform, a general good rule of thumb is always double your estimate on how long some of these things take. I think, for us, one of the biggest challenges was getting the overall architecture right relative to what would be hosted and managed by us versus what would exist in the customer’s tenant. We actually ran through two designs during that period. We literally disrupted ourselves along the way and throughout version one of our design and went with version two before we even released it publicly. The main difference there between the two designs was our first design was really just a control plane, where we were going to spin up everything in the customer’s environment. And we thought that would be great, because it would all be compute, consumed on the customer's account, the customer’s bill. They would have complete control. We thought it’d be easier with security departments potentially.

But there ended up being a lot of usability challenges around that. Given that our primary design goal was to make things easy, we actually scrap that in favor of a model where we actually manage the control plane and the compute plane. We still have storage within the customer's account. So we're accessing their S3 buckets or accessing their MySQL database or their Redshift instance. But compute is now managed by us. Just by bringing compute into our domain, if you will, allowed us to do a lot of clever things to just make the overall experience very fast, very fluid. We can actually keep compute

sort of like ready for people who want to just jump on the system. We can do some really clever things around shutting down compute when customers aren't using it to save them money. There's just a lot of benefits to kind of the second architecture, and it took us time to discover that. We had to actually get the first architecture into beta and start using it with customers to realize that wasn't the optimal way to build it before kind of inventing it. Ultimately, we're super happy with what we came up with. But it was multiple iterations, behind the scenes to get us there.

[00:20:07] JM: How does it compare across different cloud providers? Is it significantly different?

[00:20:14] JB: No. So one of the beauties of kind of how we designed it was, we wanted to be able to deploy on all three clouds and have it be the exact same experience. Part of the secret sauce under the covers is Kubernetes, and that allows us to sort of unify the way that we deploy and have effectively the same software running, regardless of where we go. It all looks and feels the same. You're basically just connecting two data sources that may be on one cloud or another. But the overall experience is effectively the same.

[00:20:45] JM: When you work with people who are trying to set up Trino on their own, is there anything in particular that causes difficulties? Just perhaps different connection points, or learning the mechanics of querying? Or what's the most difficult onboarding?

[00:21:07] JB: Yeah, the biggest challenges around onboarding with the self-manage, downloading Trino and deploying it yourself, it's really dependent upon your own environment. I would say, it's the various integration points that create the potential challenges. So, if you're deploying in an on-prem world, maybe you have Kerberos. I hope you're lucky enough to not have Kerberos. But if you're a large enterprise, you might, and that is probably the number one pain point. If I was going to put something at the top of the list. There is Kerberos integration, we have a ton of experience with it. But it can be a thorny thing, and it re-enterprise deploys Kerberos a little bit differently. That's a good example.

Also, connections to different data sources, and depending on how those data sources are set up from a security perspective, can also be a challenge. It really depends on what you're trying to do with it. If you're just trying to connect to a data lake, that's a lot easier. That is a much slimmed down experience. And again, regardless of what you're trying to do, with our cloud-hosted offer and we try to make it just super simple regardless. But again, if you're trying to roll this your own, those are the types of things

that get people stuck, is sort of integrating with other tools, other technologies, other forms of security, SSO, et cetera, et cetera.

[00:22:27] JM: Have you learned anything about data infrastructure, modern data infrastructure, that has been surprising to you, maybe over the last couple of years since we last spoke?

[00:22:41] JB: Yeah, I will say one thing that surprised me, and it's turned into a use case for us actually, was learning that one of the most popular use cases for Snowflake in the market is actually just taking data out of Salesforce, and loading it into Snowflake so that you can run analytics. I guess what surprised me about that is like, the data is already in Salesforce. So we're literally taking it out of Salesforce, we're putting it into another database system, just so we can run analytics. And when we discovered this, we were like, “Why don't we just build a connector for Salesforce? It's just another data source. Now, you can run queries directly on it, you can join data in Salesforce with data sitting in S3, product data, plus sales data, plus billing data.” It became a really interesting use case for this notion of having data decentralized and like why. I guess I've been consistently surprised just how popular of a use case that is, like, literally extract from Salesforce dumped in snowflake, and not really necessary, in our view. Maybe it was necessary, but certainly not necessary anymore.

[00:23:52] JM: Tell me more about the process of managing the engineers. I know you're not directly responsible for it, but as CEO of an engineering heavy company, can you talk about what kinds of engineering questions bubble up to you?

[00:24:12] JB: Yeah, sure. I will start by saying, I am incredibly lucky to have probably some of the best engineers on the planet, certainly the most incredible team I've ever assembled or had the opportunity to work with. And not only the creators and leading contributors to Presto and Trino here, but just exceptionally talented folks. So fortunately, the hard problems they figured out themselves, but the ones that come to me, tend to be really more oriented around product priorities or architectural priorities, I will say. One of the unusual things about our company is we have four CTO, so a lot of people ask me about that during an interview process. Why do you have four CTOs?

The reality is, there are multiple creators of the open source project, we couldn't decide what role they would play, and we gave them the CTO title. The beauty of it is, they actually work really well together. So they operate as sort of like a hive mind, in a lot of ways, on making important decisions. But because of that, occasionally there are topics where I have to weigh in on a particular strategic

direction, or we've got a tradeoff to make between two big moves, two big directions. I mean, I referenced earlier in the podcast, this pivot that we made internally, within engineering, moving from one architecture for our SaaS product to a different architecture, that was a big decision. That was the type of decision where I certainly had to make that call, because we already had most of our engineering team marching down one path with one version, and we had a very small team prototyping this other approach. And ultimately, we decided that was a better approach.

So, those types of calls are generally the ones that bubble up to me. It's not that I'm bringing any great insight by any stretch of imagination, but really more so trying to weigh the priorities of the business against the information that I'm given, and trying to make clear decisions, and then driving forward with them.

[00:26:25] JM: You've talked a little bit about the data mesh concept, and we've done a show on that. I'd like to get a little bit more of a sense for what an intelligently managed data mesh looks like, and what the different components of that mesh are.

[00:26:44] JB: Yeah, sure. It's definitely a hot topic. We actually just ran a digital virtual event. It's recorded, if anybody's interested in it, called Data Nova. That's our annual customer event. The theme for this particular event was all around data mesh. So we had Zhamak Dehghani, who coined that term, give some of the philosophical underpinnings around it. And then we had a number of different users and customers and practitioners, and even systems integrators and partners and everyone who had a perspective on data mesh, we tried to pull them together in one room to talk about it. So you can get a lot of perspectives, if you just look for our most recent Data Nova event.

But I would say at its core, it's really built on four principles. One of them is this idea of really giving ownership of the data to the domain owners, the people who know the data. So, one example I like to give is maybe there's a web team that owns the Clickstream data. They're collecting the web logs, and allowing them to basically curate that data and create data as a product, which is really the second pillar. This notion that data itself should be a first class product that is shared and shareable within the organization, potentially external to the organization too, but really kind of starting within internal to the company perspective that others can consume.

At its core, data mesh is really about decentralization and democratization. So, you see that in each of these pillars. Again, kind of pillar one is domain centric ownership. Pillar two is thinking about data as a

product. The next pillar is this notion of a self-service sort of infrastructure. And I'd say that's what we really hope to be in that overall data mesh design, is really allowing people to execute their queries, in a self-service fashion, access the data where it lives, create data as a product. That's actually a new feature we just added to our offering that allows you to pull together tables from different data sources and kind of stitch them together as a product and make it available to the organization.

Then the fourth pillar is this notion of federated governance. If you're going to provide a data mesh architecture that allows access to all of your data in this sort of single point of access, then you need to have some form of governance to make sure that the right people are seeing the right data. That's really where access control has come into play, and the ability to enforce those access controls across all these different data sources.

That's kind of the pillars behind the concept. We see different customers deploying data mesh in different ways, starting with different points. I think very few people kind of say, “Okay, I'm going to deploy data mash from zero to one overnight.” I think that's probably setting yourself up for failure. It's kind of like, how can you start to take some of these principles and put them to practice in a bite-sized way. One way to start might be simply just connecting your different data sources and making them queryable. Simply query federation might be an early step.

Another might be creating these data products that can be consumed and shared and discovered by others in the organization. That might be another step in the journey. And then there's a people and process side to all of this that's not to be underestimated. Technology can only solve so much of this data mesh philosophy. A lot of it is also people in process and getting people comfortable with delegating ownership, delegating responsibility, delegating accountability, to those who know the data best, rather than purely a central data warehousing team. I think one of the things that's been very interesting for me, in my data journey over the last 12 years, is how often the data warehousing team itself doesn't actually know that much about the data, or the queries being run. That's no disrespect to any data warehousing team. It's just like, they're focused on the infrastructure, they don't know necessarily the business problems that are being solved very often. I've had countless conversations where I'm like, “Hey, what is the business use this for? What are these queries that are running on the platform? you're running thousands of queries a day, you know, millions of queries a month, but what's going on here?”

They have very rudimentary understandings, and that's because that knowledge lives somewhere else, understandably. If we can put more of the power in the hands of the people that know the data and know what they want from the data, that could be a powerful thing to allow enterprises to make faster, better decisions. That's really at the core of, I think, a lot of this data mesh thinking. But I'll stress that it's early days. A lot of this theory does have to be put into practice and I think that's the phase that we're entering into now.

[00:31:45] JM: How does that guide your product development?

[00:31:50] JB: Yeah, I mean, I think we see our roles as enablers in this. We're very cautious to make sure we don't say that we deliver a data mesh. Data mesh is much bigger than what we provide. But we certainly believe that we're enablers of a data mesh design and to that extent, we're always thinking about how can we help facilitate that more easily. So even going back to the governance side of things, we've built a lot of governance capabilities into our own product. You can do fine-grained, role-based access control, that you set up in one place within Starburst, and then enforce those access controls across all the datasets that you have, and all the queries that are being run and, even the data products.

One user might see different things in a data product than another user. That's an example of where we're trying to take part of that theory and put it into practice. We also do that through our partnership strategy. We work with other governance providers. We have a partner called Immuta, and we have another one called Privacera, that also provide access controls in different ways. Immuta does something called attribute-based access control. And some customers might need that. So, we have a strong integration and partnership there, where customers can leverage that in their data mesh approach.

I guess, to answer your question, I would just say like, we're trying to pay very close attention to the needs of the pioneers in this movement, and what tools and capabilities they need to help accelerate adoption of a data mesh model. When we hear something that we think falls into our wheelhouse, into our domain, into our expertise, we go off and try to build it. Whether that's our new data product feature, or the governance features or other things we're working on today.

[00:33:40] JM: There are different kinds of users of Trino. Depending on the role of the user, you might be a consumer of data, or you might be a producer of data. Can you talk about how you've built out the

experience of Starburst to look at that producer and consumer relationship? And what kinds of tools you've engineered to facilitate that?

[00:34:16] JB: Yeah, that's an awesome question, because we definitely think about it in a similar fashion. On the producer side, that's where we're trying to give them more control over how the data that they produce is consumed, and make them an active stakeholder in that, again, because they have the domain knowledge. We think that they play a unique role in being able to craft the data that they produce, as a product that can be consumed.

In our most recent release that just came out, in fact, what you can do is effectively create a product that might be the result of a join between two tables. It might be the result of particular aggregation. It's a product that the producer themselves could actually create, and then effectively publish to the organization in a way that it can be consumed. That publishing step is effectively making it discoverable by theoretically, anybody in the organization who finds it, can read about that data product. Okay, this is the customers that visited our website last month, and read metadata about the product itself, what kinds of fields, what's in this data product, and even read reviews about that product.

This is a four-star product, that's a five-star product. People are getting a lot of value out of this one. This one's been queried 10,000 times. That one's only been queried three times. So really trying to create almost like an E-commerce like experience within your enterprise around this notion of data products, but putting the control of the creation of those products in the hands of the producers. And I think that's what's a bit different from the way things have sort of traditionally worked. Traditionally, the data producer just produces the data and then maybe hands it off to the data warehousing team who loads it into some data structure to be queried, at some point. This is putting more of that power and control into the hands of the producer to curate what they view is a high quality product. We think the result of that is higher quality data, higher quality insights, and a feedback loop, by the way, between the consumer and the producer. Because the consumer can now say, “Hey, you know what, I really wish you added this field to that data product.” And maybe the producer says, “Oh, yeah, we've got that field, that's a good idea. We can enhance the product that way and publish a new product that's a new version of that product.”

On the consumer side, and the consumer has that opportunity to discover all the different products available, query it directly if they'd like to, with the built in query tool. If they're a power user, they know SQL, they can just start querying it right away. Or what we think is probably going to be more common,

they can connect to their favorite tool. That tool could be anything from a popular BI tool like Tableau. We have strong partnerships with really all the BI tools, Tableau, Looker, Power BI, et cetera, and visualize it that way, or their favorite query editor or Python notebook. They have a lot of flexibility on the sort of tool side of how they integrate. We support JDBC, ODBC, and REST API. So you can even build your own applications against this data on the consumption side, as well.

[00:37:42] JM: As you look to additional tooling, I'd like to get a sense for where your platform is going. So, you've got the core product, pretty dialed in, you've got the cloud product built. When I think about other mature data organizations like Databricks, for example, they developed so much functionality on top of Spark, such that they're not even, like entirely focused on Spark anymore. Now, they've gotten into data lake technology, and they have all this work around Jupyter Notebooks. Tell me what the higher level products of Starburst will look like?

[00:38:28] JB: Yeah, well, I mean, I think at the end of the day, what we're trying to do is connect everyone in the enterprise to all the data that they have. It's sort of like facilitating that connection is what we're driving. If we do that, we think we are freeing up every member, every constituent in the organization. We're freeing the consumer, by giving them the freedom to effectively be curious and just run queries the moment that they have a thought. We're freeing up the data engineers, the data architects to basically decide where that data is going to be stored and not be bound by a storage centric mentality. Maybe it's Hadoop today, and S3 tomorrow, and Azure Data Lake Storage the next day, and they have that flexibility.

So trying to inject freedom to each constituent in that chain, and provide, again, like immediate access to everything. Anything that furthers those missions will be of interest to us, whether that's continuing to improve performance. One of the most popular questions we get, when we talk about the ability to query data anywhere is like, “Is that fast? I would expect that to be slow because you're federating across different data sources.” The reality is, it's much faster than people expect, largely because of the parallel nature of how queries get executed, but also because of the parallel nature of the connectors that we build.

There's a whole host of additional optimizations that we can add there. So we'll always be focused on performance, both in terms of federating queries, but also in the data lake. The data lake is like our bread and butter and where we started. So we're never going to stray far from being able to execute queries very efficiently there. But also investing in ways to further democratize access. How can we put

the power of data in the hands of every single potential consumer with an enterprise, from the intern to the CEO? I think that's ultimately the vision that a lot of enterprises have themselves of how can I create self-service analytics across broad demographics within my company. We want to help facilitate that.

So, I know that sounds a little bit vague in terms of exactly what we're focused on. But you'll see more and more stuff come out later this year that sort of amplifies those goals of access, and I'll call it optionality on the data architecture side.

[00:41:04] JM: Cool. Well, Justin, thank you so much for coming back on the show. It's been a pleasure talking to you once again.

[00:41:11] JB: Thank you, Jeff. Really appreciate it.

[END]