The Distributed Data Show Podcast Episode #139:

The Decade of Scale out Data with Sam Ramji - TRANSCRIPT

Release Date: March 10, 2020

Sam Ramji: Looking at the next decade, nobody's really nailed scale-out data. So scale-out network, awesome. 10 years. Scale-out compute, awesome. 10 years. The next decade really looks like, "Hey, what would it look like if you had cloud native data? What if you had this ability to run your data wherever you wanted? What if you thought about it as a workload that it could be well managed, governed anywhere on the planet, doing the things that you needed?" By 2030 I think we'll look back and say, "Wow, that was another super hard problem. Boy did we nailed it."

Speaker 2: From DataStax, this is the Distributed Data Show podcast.

Jeff Carpenter: Hi, this is Jeff Carpenter. Welcome to part two of my conversation with DataStax chief strategy officer Sam Ramji. In the first half of our conversation we talked about Sam's background, how he got into data as well as his thoughts on open source culture and the reasons behind some of DataStax's recent changes and opening up more of our tools. This week we'll hear more of Sam's thoughts on how to speed innovation in open source projects as well as the moral imperative to use artificial intelligence to automate database operations. And finally, why the hard problem we solve as an industry in the 2020s will be scale-out data.

Jeff Carpenter: Before we move off of the idea of open source and kind of philosophy of open source, anything else that you want to add that you didn't get to yet?

Sam Ramji: On the topic of open source? I think that there's a need for open source communities to have an architecture of participation and that's a term from Tim O'Reilly, which I've thought about ever since he said it, which I think was in the late two thousands. Not everything in an open source project needs to move at the same speed, right? So when you think about the Linux environment, you've got something that moves cautiously forward with a heck of a lot of testability, right? And that's the low levels, the kernel. You've got another speed lane for drivers, the Linux kernel driver project, which needs to move as fast as the hardware market moves.

Sam Ramji: So being able to decouple those two things and you've got Linus Torvalds looking over the core kernel. Then you've got Greg Kroah-Hartman looking at Linux kernel driver project. You've got two speed lights. Some projects like CloudFoundry had four different speed lanes because that was appropriate. As we look at how we evolve the community, we need to have a group

conversation on what are the pieces of Cassandra that need to move fast? What are the things that need to be super slow and stable and how do we make sure that the whole thing feels like it's moving at the right velocity so that there's some harmony in the project rather than having tugs of war between some groups trying to go really fast and some trying to have no change at all. I think the CEP that you mentioned fits into that, but I think the fully mature nuanced conversation is, "Hey, what are the speed lanes that we need and how do they take care of everybody? In order to build those properly, do we need to shift the architecture so that we can increase participation?"

Jeff Carpenter: Nice. So yeah, I could see from an off the cuff, very simplistic level, having that higher level of rigor for the core engine versus having a super fast lane for tools and then maybe some interesting conversations about which lane the Kubernetes operator belongs in.

Sam Ramji: Exactly. And you know what lane should a pluggable storage engine be in? How many pinions are there really about storage engines? Are there going to be 80 or are there going to be eight, or are there going to be four? So those kinds of things, when we add the nuance of what's the Delta T that can take a lot of the frustration out of a conversation, which previously seemed like it was all or nothing. So just adding in the dimension of time makes things a lot more tractable.

Jeff Carpenter: Right. Okay, cool. So the gear switching really has to do with the future of databases in particular. We had Jonathan Ellis on recently on the show talking about a talk that he gave on five areas of prediction of the future databases. And I'm not going to ask you to recite those from memory and we're not going to dig into all of them here, but one that I wanted to follow up on with you in particular, just because of your background in AI and machine learning, is that very aspect to that database operations and the role of AI in improving the database operations... His assertion is that's going to be a huge deal for the next five years of databases. So, what's your thoughts about that?

Sam Ramji: Yeah, I think it's a given, right? So there's a technical component and there's a moral component, right? And I'll talk about that technical component first. How many different configuration options are there for any given database? Let's just pick...

Jeff Carpenter: Hundreds!

Sam Ramji: In fact, combinatorics quickly take you to trillions and if you get into a distributed database like Cassandra, you end up actually well over a duodecillion. So let's just pretend it's only trillions. What's the likelihood that any of the databases in the world that have a trillion possible combinations are properly configured and tuned? It's pretty close to zero. Now let's also add into

the fact that any database that's functioning has got variable load and in fact, most functioning databases have variably increasing load, right? Whether it by spike or whether by a swell, they ended up having more load and that ends up putting back pressure on how the database ought to be configured. So it's a complicated multivariate equation that changes over time.

Jeff Carpenter: So you're saying the problem is not just how should the database be configured or what is the optimal configuration, but what is the optimal configuration now?

Sam Ramji: Yes. How many seconds or your current thoughts and configuration valid for? That's a smaller and smaller number of seconds therefore there's a technical need to be able to solve this. The good news is that we are long, long way from AI as people imagine it in science fiction, which most of us as practitioners call it AGI, Artificial Generalized Intelligence or Artificial General Intelligence. That's not a thing. But when we talk about AI and operations, we're really talking about being able to play Chess or Go or play Jeopardy, right? A closed end, yes, complex, but closed end problem that a computer can play again and again in simulation and learn from and get right. So that's a really nice technical environment to learn given this demand pattern, which the database configuration be and then to be able to change that over time.

Sam Ramji: The moral challenge here is, is a job that is fundamentally about dealing in a high stress environment where you're getting yelled at to stay up to date on the new configurations that somebody just came up with for the latest version of the database and then modifying that as part of your day job and being required to go at machine speed. Is that a moral use of people's time? Think about this, why do you struggle with email? I would argue that you struggle with email because it is an implicit attempt to turn a person into a machine at the speed at which people can send you requests far outstrips your ability to respond to them.

Jeff Carpenter: And we've got the automation to prove it.

Sam Ramji: Right, that's super unfair. Why would you do that to a human being? That is torture. The moral angle of that is as we do more and more AI, the beautiful thing to me about AI is it's teaching us more and more about what's amazing about being human. We can create the future together, we can dream, we can ideate, we have emotions. Our ability to pattern match and push buttons actually isn't special. That's the stuff that we can automate. So let's automate the automateable. Let's humanize the things that make human jobs meaningful and interesting.

Jeff Carpenter: Now there are steps along the way. I am totally on board with this vision of the self configuring database that is optimally tuning itself at every particular moment but I imagine like you said, that we're some ways away from getting

there, are there some intermediate points maybe where I'm getting more recommendations from a system about things that I should be doing or what does that look like?

Sam Ramji: Absolutely, if you're going to eat a meal, it makes sense to eat it a bite at a time. You look at somebody trying to eat the whole meal in one shot and you'd be like, "That's a weird person." And it's probably not going to go well.

Jeff Carpenter: That's called a batch job or a hot dog eating contest.

Sam Ramji: Well, even then you still have to eat a hot dog at a time, but imagine if you had to eat 50 in the same moment, that would be trouble.

Jeff Carpenter: That's too much.

Sam Ramji: What we used to teach people at Google was, customers, who are asking the exact same question, "How do we take advantage of TensorFlow and of the TPUs and all these other training systems? How do we consume all the data?" We gave really practical advice and I'll provide the same here because it's been tested. Draw out a workflow. Draw out all the steps that you go through, include the time sequences, not just the time taken at each point, but how long does it take between and then go back and circle in red, the ones that really bite you. The things that are very error prone or things that are just really time consuming, super tedious and frustrating.

Sam Ramji: One at a time, prioritize those and tackle one of those at a time and say, "Okay, how do we ring fence this operation? What is the data input that we get? What is the action that we take on that data? And then how do we teach an ML? How do we get a training set that we can run an algorithm against so that that network tunes itself, learns how to match the inputs we're providing with the outputs?" Get that one thing right. Then you can get a second thing right. But that way you're looking at it in a stepwise workflow process. When you draw out that workflow, you might also find some things that you can just collapse and not do twice, but the most important thing is; solve for the highest point of pain, determine whether that's repeatable and automateable if so automated and use your ML techniques to do pattern matching style automation.

Jeff Carpenter: Perfect. Yes.

Sam Ramji: One step at a time. You can think about it, probably a dozen ways that you could do with Cassandra today, right? What are all the things that you do from having your first idea? "Oh my gosh, I need to install Cassandra on a node." All the way to, "Oh my gosh, I've been running this for two years and parts of this distributed database are getting super hot and falling over and my uptime's not

what I wanted." And "How do I scale it up to these particular environments?" Or whatever those things are...

Jeff Carpenter: "I want to upgrade to the new version because the new version has the bug fix that I want, but I'm also scared about this other new feature over here that I don't know how to tune." Like you mentioned.

Sam Ramji: Think about red/green deployment or some people call it a red/black deployment, if you want to bring a new version into being, what's the right way to do that? That's something that could be automateable and that automation doesn't have to be an expert system. That automation could be something that has learned how these things work well with this particular version. Being able to manage that complexity and mask it behind an ML, those things will make our life a lot better.

Jeff Carpenter: And there's actually ways... Cassandra is fairly well poised for this anyway because you can add additional nodes to the cluster that are taking on load but then are not necessarily servicing queries. There's definitely ways in which the architecture is definitely amenable to that.

Sam Ramji: Exactly, you could actually be tweaking your consistency guarantees as you make this change in operation. You're like, "Okay, what matters to me most is uptime and I'm not going to worry about inconsistent multi-region right now because I know a minute later I'm going to go back to multi-region consistency that's going to collapse the complexity of the queries that I'm running in order to do my rights and then we're all going to be fine." All those strategies are strategies that, just like a chess playing computer, you can come up with and you can even have those computers learn via simulation and you can end up with a few different strategies and say, "Hey, I want to take a real conservative strategy on this one." Or "I want to take a real aggressive strategy and have the implementation of what that means." Be a known repeatable thing as opposed to, "Well, Joe thinks it should be this and Susan thinks it should be that." And then you've got like a political context which is no good for anybody either.

Jeff Carpenter: Exactly. No, and I love what you're saying about the simplicity of breaking down the problems into smaller steps, not to be flip or anything, you were literally taking me back to engineering 101 right? This is solid engineering principles. This is how we apply the scientific method in practical standpoints to build real world systems that are doing things that are helping people, I love it.

Sam Ramji: It's all the same thing, it all comes out of control theory, right? There's no magic, it's just cybernetics. We understand systems to be composed of people and processes and machines and we go a step at a time. It's not magical. What's really cool is that you can now have a pattern recognition engine that occupies

stance in the workflow where it can look at multi-dimensional, multi-variate problems that are changing very rapidly and come up with the right answer.

Jeff Carpenter: Nice. Okay, so I've got one more for you before I let you go. I know that you are very excited about this area of data in particular as an industry, as a part of the open source software world, et cetera. But why data, why now?

Sam Ramji: Yeah, that's a great question. These layers, network, compute, data, we know that they all work together and that we needed to rely on them, but they can't all change at the same time. Networking really dominated the 2000s, that was the decade of network innovation and by innovation, not just invention, this thing is possible, but innovation. This thing's actually used. Look at Andy Bechtolsheim and Arista Networks and stuff that many of us never worry about. You get to 2010 and networking and the internet just works.

Sam Ramji: So that's the decade of network. You look at 2010 to 2020 that's the decade of compute. We figured out, how to do scale-out compute, internet and cloud scale. And that's mostly what we think of as the cloud today. You can start up a job no matter whether you want it in a container or a virtual machine, better if it's a container, but know nothing wrong with VMs. Looking at the next decade, nobody's really nailed scale out data. So scale at network, awesome. 10 years scale out compute, awesome. 10 years. The next decade really looks like, "Hey, what would it look like if you had cloud native data? What if you had this ability to run your data wherever you wanted? What if you thought about it as a workload that could be well managed, governed anywhere on the planet doing the things that you needed?" By 2030, I think we'll look back and say, "Wow, that was another super hard problem but boy did we nail data?." So I want to be part of that.

Jeff Carpenter: Is that a different distinction between scale-out data versus scale-out database?

Sam Ramji: Yeah, I think a database is an instance of data, but when we think about where data is going, we think about events. We think about streaming. We think about different elements of a data workload. Data turns out to be pretty valuable, you need to have it backed up. What does streaming backup look like? What does it look like to manage the data? Because most of the policies that governments come up with, like if you are dealing with a citizen of Spain, you want to make sure that the data about that person doesn't leave Spanish borders. They don't give you a specification of the database. The database is an implementation detail of what you're doing with that user's data. So I think networking, compute data, it's broad. When we say compute, we're not being super specific in saying you have to use a 2007 edition AMD CPU. Who knows what it's actually going to be resolved to. But you want your data to resolve to a smart thing and you want to feel like it supports your applications and your AIs, wherever they might go.

Jeff Carpenter: Got it. Got it. Okay. So good. Let me read this back to you, we may have instances right now of particular technologies that are scale-out, streaming, engines and scale-out databases but there's a larger problem of scale-out data that we need to handle. And it's not about any one particular technology, it's about handling it. It's about international laws and regulations. It's about data movement. It's about data preservation and lineage and all of these things.

Sam Ramji: Exactly. And anybody else who's thinking about these problems is closer associated to us than people who are thinking about other problems. So I look at this as a great invitation to help figure out, what does the next decade of data look like and how are we all going to manage it together?

Jeff Carpenter: Awesome. Well, I am looking forward to more conversations about that because there's N jumping off points that we teased in this conversation, where we could take things going forward. So I just want to thank you, Sam, for joining me today.

Sam Ramji: Awesome, thank you Jeff.

Jeff Carpenter: Great, and we look forward to talking to you next time on the Distributed Data Show.

Speaker 2: Thanks for listening to the distributed data show. Please subscribe with your favorite podcast app. Give us a rating and don't forget to share with your colleagues.