EPISODE 728

[INTRODUCTION]

[00:00:00] JM: The Berkeley AMPLab was a research lab where Apache Spark and Apache Mesos were both created. In the last five years, these projects; Mesos and Spark, they've changed the way that infrastructure is managed and they've improved the tools available data science. Because of its proximity to Silicon Valley, Berkeley has become a university where fundamental research is blended with a sense of industry applications, and Apache Spark and Apache Mesos were perfect examples of this. They came out of the union of university research talent and highly applied corporate problems in distributed systems.

At Berkeley, students and professors move between business and academia, finding problems in industry and bringing them into the lab where these problems can be studied without the day-to-day pressures of a corporation. This makes Berkeley the perfect place for research around serverless. Serverless computing abstracts away the notion of a server allowing developers to work at a higher level and be less concerned about the problems inherent in servers, such as failing instances and unpredictable network connections. Of course, there are servers in the mix the cloud provider is operating servers to power this functionality, but the user is not exposed to those servers.

With serverless functions as a service, the cloud provider makes guarantees around the execution of serverless code. This is the case with AWS Lambda, Azure Cloud Functions, Google Cloud Functions. With serverless backend services, the cloud provider makes guarantees around the reliability of a database or queuing system.

Today's show centers around the serverless functions as a service. This is a new paradigm of computing and there are many open questions. We've done many previous shows about serverless computing. Some of the open questions; How can the servers for our functions be quickly provisioned, because these functions are going to run on servers and we want them to execute quickly? There's what's known as the cold start problem that we've covered in previous episodes.

How can we parallelize batch jobs into functions as a service? If you have a large MapReduce job, it would be great to parallelize it into functions as a service so they can run efficiently and cheaply. How can large numbers of serverless functions communicate with each other reliably so that they can coordinate actions despite being these ephemeral distributed functions as a service running in containers?

In today‘s production applications, functions as a service are mostly used for event-driven applications. They serve the purpose of being glue code, but the potential for functions as a service is much larger. Ion Stoica is a professor of computer science at Berkeley where he leads the RISELab. Ion is the cofounder of Conviva Networks and Databricks. Databricks is the company that was born as a result of the research on Apache Spark. Ion now serves as executive chairman of Databricks and Ion joins the show to describe why serverless computing is exciting, the open research problems and the solutions that researchers at the RISELab are exploring.

Before we get to today’s show, I want to mention that we are looking for several open positions that are available at softwareengineeringdaily.com/jobs. We’re looking for journalists, podcasters, a couple engineering interns and also an entrepreneur in residence. If you’re interested in researching a subject and potentially building a business around it, then check out that role. You can go to softwareengineeringdaily.com/jobs. We’d love to hear from you.

[SPONSOR MESSAGE]

[00:03:52] JM: How do you know what it’s like to use your product? You’re the creator of your product. So it’s very hard to put yourself in the shoes of the average user. You can talk to your users. You can also mine and analyze data, but really understanding that experience is hard. Trying to put yourself in the shoes of your user is hard.

FullStory allows you to record and reproduce real user experiences on your site. You can finally know your user’s experience by seeing what they see on their side of the screen. FullStory is instant replay for your website. It’s the power to support customers without the back and forth, to troubleshoot bugs in your software without guessing. It allows you drive product engagement by seeing literally what works and what doesn’t for actual users on your site.

FullStory is offering a free one month trial at fullstory.com/sedaily for Software Engineering Daily listeners. This free trial doubles the regular 14-day trial available from fullstory.com. Go to fullstory.com/sedaily to get this one month trial and it allows you to test the search and session replay rom FullStory. You can also try out FullStory's mini integrations with Gira, Bugsnag, Trello, Intercom. It's a fully integrated system. FullStory's value will become clear the second that you find a user who failed to convert because of some obscure bug. You'll be able to see precisely what errors occurred as well as the stack traces, the browser configurations, the geo, the IP, other useful details that are necessary not only to fix the bug, but to scope how many other people were impacted by that bug.

Get to know your users with FullStory. Go to fullstory.com/sedaily to activate your free one month trial. Thank you to FullStory.

[INTERVIEW]

[00:06:13] JM: Ion Stoica, you are a professor of computer science at Berkeley. Welcome to Software Engineering Daily.

[00:06:18] IS: Thanks for having me.

[00:06:19] JM: I reached out because I heard from Dave Patterson, I was talking to him in a podcast episode not too long ago. He said that you are interested in serverless computing or you're working on some projects associated with serverless.

[00:06:33] IS: Yeah, like many others.

[00:06:34] JM: Like many others, yes. Serverless computing, this can mean a number of different things. What's your definition of serverless?

[00:06:43] IS: Serverless computing is just a further context why we are talking about serverless computing, right? Cloud was, so to speak, invented by around 2006 with the emergence of Amazon of services and what they provided is they provided the ability of people

to instead of building their own datacenters, to spin-off a bunch of VMs on the cloud so it was much faster to get resources than ordering and receiving servers to that process takes months.

Also, effectively, elasticity and free-scaling. You are only paying for how much you are going to use these virtual machines. So that was a huge improvement comparing to what was in the past. So you need to build all these – The clusters in the datacenters.

However, what this early cloud services didn't provide you or don't provide you or I should say what they still ask you to do is to manage the resources. You still need to manage the VMs to make sure that you know how many you need. When you are going to be done with the computation, you need to power them down. You need to also scale explicitly the number of VMs both up and down. Really, you need to operationalize and manage these VMs.

Now, what basically serverless is providing you, it's getting rid of all these operational and management overhead. Now to get to your question, what I think there are the key characteristic of a serverless platform and now every major cloud providers provide you serverless services starting with AWS, which provides AWS Lambda, Cloud Functions, Azure Functions from Microsoft and Google. It provides you following serverless platform has a following key characteristics.

First, it's basically the decouples of computation from storage. Let me give you an example here. For instance it used to be that – Or every major cloud provider also provides, say, database like services. In the past that was running database instances. There are many such examples. Like for instance, one of the most successful [inaudible 00:09:15] parallel database provided by Amazon.

Now, with these services, you have to pay for these instances which have both computation and storage. Now, if you do not use the database, you still have to pay for it. You still need to pay also for the compute capabilities of that node. What is decoupling computation from storage means? Now, take for instance BigQuery, which is a first serverless query engine. In that particular case, basically what you don't use, don't own a query, then you pay only for storage. You pay for storage, which actually can be in a blob store. Then you can use – You're going to pay only for computation. You are only going to pay when you are going to perform a query,

right? This decoupling storage from computation, it's one of the most defining features of serverless.

The second one, it used to be that you have to pay for the resources. I'm using a VM for one hour. I'm going to pay for that. With serverless, you are going to, instead of instantiate the VM and installing your software, running your software on it, you are paying for the execution of both of the function, say, by the function. You pay only for how much your function or your code is executing instead of paying for the server on which you are going to install the code and then run your application.

The last one is that when you are going to – The last one is this capability to transfer and rescale up and down. You have more functions you want to execute. You are going to get more resources. You have no longer functions to execute. You are not going to pay for anything. This transparent scaling up and down, which with respect to your computation demands, it's also defining feature of serverless.

Again, in the past world, you have to do it explicitly. You need to manage these VMs. You need to explicitly scale them up and down. If you forget to tear down or to shut off to a VM, you are going to pay for it even if you don't run anything on it. Actually to facilitate that, typically today, all these serverless platforms, they have a timeout associated with each execution of every function. For instance, AWS Lambda, they have 10 minutes now. They used to have 5 minutes. Now, 10 minutes. This means that after 10 minutes, even if your computation didn't finish, hasn't finished, they are going to shut off the Lambda. Again, this is also protects you as a user, because you don't need to explicitly keep in mind that, "Oh! My computation now is done. So I need to shut down the servers."

[00:12:22] JM: You've described a couple different flavors of serverless, a couple of different examples of serverless in terms of – You gave the example of BigQuery, which is a rich query service that has both compute and storage associated with it. You also mentioned AWS Lambda, which is more of a primitive and it's an efficient scalable way of executing simple functions. These are two different categories of serverless, these primitives as well as these rich computational services. Are you more excited about either one of these particular areas?

[00:13:03] IS: Yeah. I think it's a very good point. Yeah, there are two types of serverless, so to speak, platforms. One is called function as a service, or FaaS. This is AWS Lambda or cloud functions, Azure Functions. The other one is called BaaS, backend as a service. This is more about – Like I give example, like BigQuery, or Athena in AWS, in which you are going to get a service which is, so to speak, serverless in the sense that you don't need to be worried about explicitly allocating and managing the resources.

I think that just before answering your question, is that about which one I'm more excited about. It's a continuum there, right? In both cases, notice that you decouple – In both cases, both of these BaaS and FaaS, they decouple the storage from the computation. Which means, again, when you are going to need some compute, you're going to pay for it. If you don't use a compute, you are not going to pay for it.

In both cases, if you storage each, you are going to pay for it. Obviously, if you run a query engine, like BigQuery or Athena, you are going to have some data and you have to store it somewhere. Typically in S3 in a blob store as a storage like that. The function as a service, again, if your application requires to process some data, then you are going to pay for that separately.

Now what is different, it's how opinionated is the framework. What you can execute? In one case, you have already opinionated framework, like providing with SQL like interface. You just can't write SQL queries. In the other case – So this is what is a BaaS, backend as a service. In the other case, you have a lot of more flexibility in the function as a service. You can execute any arbitrary function using some of the in general high-level programming languages, like Python or even JavaScript or a thing like that. That's the key difference actually.

Now, what I'm more excited obviously as a researcher I'm excited more about things that have more flexibility we are going to do more things. Clearly, the function as a service platforms are more interesting for us.

[00:15:34] JM: That makes sense. Now, I find this somewhat in contrast to the work of the AMPLab, because the AMPLab was all about these resource management tools, like Mesos and Spark, which really improved the programmability of infrastructure. Serverless computing

takes advantage of some of these resource managers and builds simpler interfaces that are in some ways – I mean, more restrictive, I guess. You're just launching a function, but you can do anything by composing functions together with some serverless storage system as well.

There are some unsolved problems. These serverless systems are often built on top of these resource managers, like a Mesos or a Kubernetes. There's a bunch of different Kubernetes-based serverless platforms that you can basically layer a serverless function platform that you run yourself on top of a resource manager like Kubernetes. There seemed to be challenges in the scheduling in the cold start problems on top of – In those serverless scheduling systems. What are the challenges that you see in building these simpler serverless interfaces on top of the resource schedulers?

[00:16:54] IS: Yeah, that's a great question. Serverless, initially though – From now on unless I otherwise specify, I'm just going to – By serverless, I'm going to mean as a function as a service. One of the motivating examples or introduced for was to process this kind of even-driven – For even-driven applications. For instance, you have some data, some logs landing on S3 in Amazon. When you get this data you want to process it to do some ETL extract transform loud that logs, right? Maybe push it to a database from which you can query.

The problem was that if these logs are landing infrequently, the only choice you had in the past is to have just a VM running, waiting for the data to come, which is quite expensive. Obviously, maybe you can monitor in other way whether the data is coming and when you know that is coming you instantiate a VM, but instantiating a VM takes a long time.

For this particular use case, a Lambda was a grateful solution because now you have something which can be to some computation, which can be triggered when the new data arrives in S3 and process it immediately at least when you compare this VM. The start time for Lambda for a function, cloud function, is measured in seconds rather than minutes or tenths of minutes of a VM like it was in the past.

Now, once you have this kind of, obviously, it's very nice abstractions. I'm talking about functions. People obviously, they try to do more than what they're initially may be designed for and they try to do analytics to do implement query engines. Here we have systems like

[inaudible 00:18:54], which tries to make it even more convenient to use the Lambdas as a cloud function or Azure functions. We try to do even linear algebra on top of Lambdas.

Really trying to use it for general purpose parallel and distributed AWS. Now, when you try to push the existing platforms, FaaS platforms to this application, to this very general applications, obviously you are going to see their limitations. Each limitation, it's an opportunity for reset, a challenge, you could address.

Let me say a few of them. So right now when you use a function as a service, typically if you want to process the data, the data is on S3. If you want to run more complex application, like say MapReduce analytics, you need to exchange that data between functions, between Mapper functions and reduce functions.

When you have to do that, the only way you can do today is through something like S3. But S3 is pretty slow. Not only can be pretty slow, but also it has a limit of how many IOPS per second, the number of input/outputs operation per second. So the number of files you read write per second is limited. Also you pay for each read and write. That can be both quite limiting in terms of performance and also quite expensive especially since, for instance, if you want to do a shuffle between the Map and Reduce, you have to do a shuffle between the mappers and the reducers. You are ending up with an inquire problem and because you need to send something from each Mappers to each reducer. Each of this send, it's going to be – You need to implement it as a writing to the store and reading form the store.

There are many other examples. One challenge is basically to provide much faster and in some sense cheaper store for keeping the application state. We are not talking here about the permanent state, but keeping the application state with only the duration of the computation. For instance, the state with respect to shuffle, which expect maybe if we do distributed training, [inaudible 00:21:28] the model during the duration of the computation or the duration of the application. That's one.

The second one which is also it's like in with these cloud functions. Like you said, you have no control where is that located. It turns out that for some application, this is a problem because a communication becomes much more inefficient. Let me give you an example. Say you want to

implement a broadcast operation to send a piece of data from one function to all the other functions. There are many examples like that, like you do distributed training and other – Or linear algebra and things like that.

Now, lambda is pretty small. It's maybe one core in general. Now, say, I want to send a piece of data to 1,000 Lambdas, 1,000 functions, I have to send one piece of data, again, storing and reading it to every of these Lambdas. So 1,000 times. Now, if you think about how would I do it in the old world or with the VMs, then the way I would do it, I would buy some big VMs which have 32 cores or 64 cores. Then, say, if it has 64 cores, if I have 16 such VMs, I go out the equivalent of 1,000 cores. Then I need to send a piece of data only to each VM. I need only to send 16 times. Because each VM can distribute or all the cores on that VM can share the data.

Now compare sending 16 pieces of data with sending 1,000 pieces of data. That's one issue, and that fundamentally because you have no control on where these Lambdas are executed. That's another challenge. Another one you mentioned about, the startup time. I was saying Lambda, you can start it in, say, seconds, one seconds, two seconds. The problem though is that you start a Lambda, but say you want to run some Python code. You need [inaudible 00:23:49] entire environment libraries. This alone it may take – The environment, it will be, say – Can be hundreds of megabytes, or even one gigabyte. Just downloading this environment and libraries may take tens of seconds. That's a problem.

Here, again, it's a challenge. There are ways to address it. For instance, maybe caching it, maybe dynamically download only what you need. But, again, it's an interesting research problem to work on. Finally, I would say it's about isolation and security. In many of these cases, the lambdas, they run as containers and the container they provide less isolation and less security than, say, a VM.

On the other hand, the fact that they can run anywhere, it's positive things for some – It makes them more robust against some security attacks, because you cannot target them. You cannot target where the code is running. So you can just [inaudible 00:25:05] there some malicious code to snoop on the activity on the communication of an application.

[SPONSOR MESSAGE]

[00:25:22] JM: Today's sponsor is Datadog, a cloud monitoring platform for dynamic infrastructure, distributed tracing and logging. The latest AWS X-ray integration from Datadog allows you to visualize requests as they travel across your serverless architecture, and you can use Datadog's Lambda layer to collect business critical metrics, like customer logins or purchases. To see how Datadog can help you get a handle on your applications, your infrastructure and your serverless functions, signup for a free trial today and get a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started and get that free t-shirt. That's softwareengineeringdaily.com/datadog. Thanks to Datadog for being a continued sponsor.

[INTERVIEW CONTINUED]

[00:26:19] JM: Now, as a researcher, in the AMPLab you are able to build everything from scratch. Mesos, or Spark can run on your own machines or in whichever cloud provider you want. But as you start to study the AWS Lambda environment or the function as a service environment within a cloud provider more broadly, you're essentially building a research, a set of research projects on top of proprietary material.

To some degree, the AWS resource is going to be opaque and you're not going to really know what's going on underneath it. Is there a degree to which you feel constrained in your research because of that opacity?

[00:27:10] IS: On one hand, yes, there are constraints. On the other hand, sometimes constraints are good because it focuses your work. Even before, we had constraints. The fact that we could only use and run on top of VMs was 7, 8 years ago constraint, because you cannot run on the bare hardware. You cannot manage a hardware, the VMS. Also, it is obviously interposition layer, which it takes a toll on performance overhead and so forth. It's just next a level.

I would say two things. One, again, it's like we are – And we are doing, one is you can still address some of the challenges. For instance, I was talking about providing an ephemeral storage to fast, high-throughput to maintain the application state. So that you can still something to work on, and actually you can deploy it on the existing public cloud infrastructure. There are

some things, like you notice, you have less control about the location and the scheduling of these cloud functions.

However, this will make you focus more on the algorithms you can design around this kind of constraints and they are going to take advantage about other features of the Lambda, like seamless scalability. Anyway, in many cases when you have constraints, it does make the problem harder and some researchers like that way. But that's one thing. That's what we are now. Again, we are building PyWren, which is a layer on top of Lambda so it will make it easier to use and write sophisticated application [inaudible 00:29:00] applications we are writing on top of PyWren. It's non-PyWren. Basically, it's a linear algebra on top of Lambdas.

Again, we are looking at building this kind of fast ephemeral storage to keep the application state. That's one line of research. The other line of research, we are also providing full-fledged – Building full-fledged stacks to provide Lambda's like functionality. You can run that. I can run that using the existing VMs infrastructures in the public clouds or I can run it obviously on other datacenters, these private datacenters.

We have a system called Ray, which provides you a very – It's bindings for Pythons, in which you can parallelize your Python code in a simple way, and intuitive way. The interface resembles. So I would say micro-functions or micro-Lambdas. There obviously we have all the flexibility to design very scalable schedulers and resource allocations and also in-memory storage and things like that.

You hope that that is, again, on one hand we can still deploy it today and you hope that if we have some good idea there, in terms of what you said, resource management scheduling and so forth, they are going to be borrowed and used by the existing cloud providers. We do have – We always have a very high-throughput dialogue with them as part of the new RISELab, which follows AMPLab.

[00:30:48] JM: Right. Definitely. With PyWren, I think PyWren is a good example of Lambda used to make scalable cloud resources easier to use. AWS Lambda, it's this multi-pronged efficiency tool and that it can lower the costs of the operations that we're already doing that might be kind of complex to do. But it's also – Since it's such a low level primitive, we can use it

to make our pre-existing, or the code that we're writing, for example, just Python code, more scalable. Like you said, you can basically build a layer on top of Python and create bindings that make that code naively parallelizable. Can you just explain what PyWren and what PyWren does to improve the scalability of Python code?

[00:31:45] IS: PyWren, it's about – The Lambdas, you execute individual functions. PyWren provides you the ability to – Provides higher level primitives, which allow you to specify executing some function in parallel. For instance, it provides a Map primitive. You can apply the same function to a bunch of files. Then on top of that, we implement something like MapReduce. On top of that, we implement something like, I mentioned to you, linear algebra libraries. You implement a bunch of libraries, which can be used to build applications, spiral application much easier. It's a higher level of abstraction, which provides you the ability to parallelize, transfer and almost transparently parallelize your application.

[00:32:43] JM: Right. It also uses S3 for the stateful coordination.

[00:32:48] IS: Yes, S3 for stateful. Like I mentioned, many of these application they are using S3 for storing the data if they need to store the data. PyWren is also using S3, but it's trying to make it as transparent as possible.

[00:33:03] JM: Can you give an example that would illustrate when PyWren is useful?

[00:33:08] IS: Yes. If you want to provide – Again, if I want to process data in parallel, if I want to process data in parallel, I'm going to write to that and I'm going to do that in PyWren to one with one line of code by passing the function which needs to be applied on a bunch of inputs.

Another why I gave you an example, it can be used to build on top of MapReduce. MapReduce is nontrivial to express using the current low-level Lambdas or cloud function interface.

[00:33:45] JM: Right. I think this relates to a paper that you coauthored last year, which was occupy the cloud. This is distributed computing for the 99%. The emphasis on that paper was the usability of the cloud.

[00:34:01] IS: Yeah. Basically, this is – The way to think about that is the following, it's like we have these frameworks to do parallel computation and parallel data processing, like Spark which we've done here in the past. However, in order to use those, you need to deal with allocating VMs and managing them. You need to deal with operationalizing and managing that cluster on which you're going to build Spark.

That turns out to be a high barrier especially for non-computer science scientist, biology, physics and so forth. They didn't want to deal with that. Now you have Lambdas, which are cloud functions which provide you don't need to manage them, because they are just going to run the code you provide. But they have a very low interface. You just execute individual function.

PyWren is try to get the best of the both worlds to provide you a higher level interface so that you can't run parallel application easily, which comes together with – The other way to say it, while getting rid of the need to manage the clusters. That's what is someway PyWren is.

[00:35:23] JM: Right. I mean, I think there's one point in that paper that said even machine learning graduate students were – Most of them had not run a distributed computing job in a cloud provider by the time they became a graduate student, which might seem kind of strange to some people who are working in an industry or who have worked with data engineering tools and industry for a while. But this extends far beyond computer science graduate students.

My older is a biology researcher and he has a GPU cluster that he uses on his machine. I told him one day, I was like, "You might be able to take advantage of the cloud," and that was kind of foreign to him. Here he is managing GPUs and he thinks the cloud is too complicated to use. He doesn't know how to use the cloud.

[00:36:13] IS: Yes. That's exactly what it is. The main reason the cloud is too complicated, because when you are going to go to the cloud basically, this is also [inaudible 00:36:20] the paper, it's not you are going to be confronted. It's not only that you need to manage. You are confronted with a myriad of choices. You have around probably, today, I think on any major cloud provider, you have – Together, you have hundreds of choices. What type of instance you are going to use? How big is it going to be? What kind of resources you need to have on it? What kind of storage you are going to use? It's a huge management nightmare.

Then, again, you need to instantiate the VMs. You need to make sure that when they finish, they are going to – You are going to shut them down. That's hard. Imagine that many of these machine learning task when they do training, they do overnight. Before they go to sleep, they start them. Then you need to have to programmatically – To programmatically figure out when you finish a computation to shut down the resources. That's not easy. At least it's not trivial. While something like cloud function or Lambdas, when the computation is done, you are no longer going to pay for it because you pay only for how long you are going to use the Lambdas. I think there are these aspects, which fundamentally it's about – these are all cloud. You just – Or traditional one, you just need to manage all these resources.

[00:37:49] JM: Any other areas of cloud computing that you think are too intimidating that could be simplified?

[00:37:56] IS: I think just to summarize. One, it's about allocating resources and reallocating them. The second related one, if you need to scale them up and down. There are many application over there, lifetime, that have different requirements. This is a hard problem, scaling up and down resources during an application while the application is running is hard.

The third thing is just the choice. You are confronted, it's like a huge array of choices. You don't know which one is better. Just a tyranny of choice. It's that. It's very intimidating. Imagine that you have to order a dinner and you are going to be confronted or you are going to have a menu with hundreds of items. It will take you some time. It will be intimidating. It's not that experience you are going to enjoy as much or many people don't enjoy that.

The other things will be also related a little bit with a cost. People are afraid they are going to forget clusters on. Also, there is another dimension of the choice, it's, "Am I going to use, for instance, spot instances?" Spot instances much cheaper than on-demand instances or reserve instances. But on the other hand, you are not guaranteed. You can lose spot instances and then you may lose all the computation. So all of these I think it makes the ecosystem quite complicated, especially for someone who has never done that. I think this is basically what is intimidating.

[00:39:39] JM: Yeah. Now, this is from the perspective of the individual programmer, the individual researcher.

[00:39:44] IS: Of the individual programmer, yes.

[00:39:46] JM: Enterprises also have issues adopting the cloud or just adopting other modern technologies in general. This is something you've seen firsthand from Databricks, which you cofounded. You were the CTO. I interviewed Matei a while ago and we talked about Databricks Delta. We talked about some of the other projects at Databricks as well. What are the difficulties of enterprise data science? How does the enterprise usage of cloud computing and data science? Is it the same thing as that graduate student struggling to figure out spot instances versus reserved instances or are they a unique set of problems that the enterprise is encountering?

[00:40:32] IS: Yeah. Good question. There are two answers to that question. One is that, say, a data scientist today which if he does not use a cloud, what he's going to use, he's going to use internal resources and is going to this internal resources, are going to be managed by the IT organization of that company. One of the main reason this data scientist want to go to the cloud is because they want to, in some sense, bypass the organization, because it's much – Again, if you need a cluster in the enterprise datacenter or need machine allocated, takes a long time.

They go there with a help that they can bypass a little bit that IT organizations and that they can get a much – Resources much faster. They can scale much faster. So they can run their experiments much faster. Now, they have to manage it. That's a challenge. Definitely it is, right? So that's one answer. Yes, for them, because they didn't use to manage, but they not they have to manage at least these VMs. It's an additional thing to do.

There is another aspect, which obviously makes the transition to the cloud maybe a little bit more challenging for enterprises. Obviously it's about security, policies about where to store data and things like that. I think that's another dimension, which – Although if anything, the transition to the cloud from what we are seeing is tremendously accelerating in the industry. I think maybe a few years back, the question was if. Now the question is what?

[00:42:25] JM: In terms of an enterprise adapting cloud.

[00:42:27] IS: Adaption of the cloud, and the question is obviously what will be the first service we are going to migrate with the cloud and so forth.

[00:42:33] JM: Has that made life easier at Databricks?

[00:42:36] IS: Oh, definitely. The more people and the more enterprises are moving to the cloud, means some more potential customers for Databricks.

[00:42:44] JM: Yeah.

[00:42:45] IS: Right. Again, if you think about it, one of the big promises Databricks is making, we are taking about as data scientist. If I go now to just Amazon and so forth, they have to manage all these cluster and so forth. One of the big features of Databricks is that it allows you to start, run a cluster at a click of a button. It scales the clusters up and down. So it takes away all these management and operations overhead from data scientists and let them focus for what they are good at to just get insights from the data, develop new models to improve the business of their company. Come up with new products. Improve the existing products and so forth.

[00:43:36] JM: Tell me more about it. You cofounded Databricks in 2013. Tell me more about how the company has evolved in the last five years.

[00:43:45] IS: Yeah. I cofounded the company in 2013. I've been the COO for the first few years. Now I'm executing chairman. I think that we had pretty strong region from the beginning. We focused on the cloud. We focused on Spark. I think that we always – When we look at Databricks in the past, we have this kind of plan so to speak. That initially we are really going to focus on Spark to making Spark successful.

The second we really make, we are going to focus on building a product, the best product for Spark in the cloud. Then to evolve Databricks in being a platform for everything being data and obviously machine learning. That's kind of where the stage is.

I think in the first one year and a half, two years, we executed just focusing on Spark, on the open source Spark. We partnered with many and we encourage everyone to distribute Spark. We partnered with Cloudera, Hortonworks, MapR, to only name the few, IBM later. We had close relation, interaction with them.

Then in 2015, we released our product which provides cluster management and management for Spark security and a set of tools, like notebooks and job execution, job launchers, to make it very easy for data scientists and data engineers to do their work on Databricks. Then we grew that business and now it's a logical continuation. We are focusing more and more on making – Not making very easy to do not only data processing and getting insights, but also machine learning. Just remember that one of the first applications and one of the early motivation for Spark or machine learning. That was one of the reasons Spark was built for.

[SPONSOR MESSAGE]

[00:46:12] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15 day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW]

[00:48:01] JM: I've done some interviews recently with different companies that are working on a data platform, and this has been – Those two words that bigram is used, I've also talked to companies where it's like Uber, or Airbnb, and they're building their internal data platform. Whether it's a company like Airbnb or Uber who has massive engineering experience in-house, or it's a company like Dremio that's building a data platform that's for enterprises that may not have computer science expertise. What's really clear is that this thing is really hard to build, having this data platform that unifies all the data sources in the company and makes it really easy for the right people to have access to the right datasets and to build machine learning models productively. It seems like such a hard problem. What kind of lessons have you learned in trying to build that unified data platform?

[00:48:56] IS: Yeah. I'm very happy you asked that. First of all, yeah, what we call our platform is unified data analytics platforms, unified analytics platforms, which is what Databricks is. Yes, going from the main motivation is exactly like you said. You have all these companies starting Google, Facebook, Uber, Airbnb getting a lot of value of their data and they do so by building very sophisticated platforms.

You have all these enterprises have also very valuable data and everyone looks at Google, Facebook and Uber, which are fundamentally data companies. They obviously want to take advantage of their data to improve their business, to improve their competitive advantage. But they don't have the same expertise and they don't have even the time or to build these teams and to build the internal infrastructure.

This is the primary target for Databricks and companies like Databricks, how to help the 99% get advantage, get value out of their data? This is what it is. [inaudible 00:50:17]. Now you are asking about lessons. I think that, here, what one of the lesson is – And which why Databricks is so well-positioned to provide an interim solution to these enterprise companies.

We, even early on, have a lot of customers buying Databricks' platform because they wanted – At the end of the day, they say you want to do machine learning because they want to do recommendation. They want to do root cause analysis on the logs and things like that. But it turns out that before you get to that point, it's a lot of things you need to do in trying to process a data. Trying to figure out, "Do you have the right data? Do you have the right logs? Is the data accurate?"

So in many of these cases, they spend many months or years just on these data preparation, so to speak, phase. This is what we see. Basically, you shouldn't underestimate that. Databricks is doing that very well. It's a bread and butter for Spark. That's how it's so well-positioned, right? Because many people want to try to do that, they use Spark anyway. Databricks now provides the fastest Spark implementation, the most accurate Spark implementation and so forth. We have the expertise, because we are behind – Databricks is a company behind Spark.

I would say this is one thing. Just never underestimate how difficult it to repel the data and to just understand and figure out that you have the data – You have information, enough information in your data to achieve your goal to improve whatever key performance indicators you want to do.

I think the second lesson which I see now more and more is about now you have the data and now you start to develop models. You need that process; experimentation, developing models, deploying and then learning about how they perform in production. Then going back and improving the models and deploying them again. That's extremely complicated product lifecycle.

That's why now at Databricks, we are developing this new open source tool and platform called MLFlow, which is trying to address some of these hard problems about tracking experimentation. You develop hundreds of models, or thousands of models. Now, after a while, I want to go back and to understand what we know, what parameters are good for this particular model? What data are used to train that model and things like that? Then deploy it. Monitoring the model performance in production and then using that to improve the model, iteratively improve the model. That emerges as one of the next level of problems you need to solve in order to really get to this promise land of turning the data into value.

[00:53:46] JM: When you start working with a company – None of the big enterprises come to mind. I'm not sure what some of the customers are. But like Coca-Cola, or a giant insurance company, or a bank, one of these companies that needs help first of all with the data cleaning process and the data organization process and sorting out how to do these things. Eventually they're going to get to maturity and they're going to be able to have problems like how do we rollback our machine learning models. That's a totally separate problem. How long does that take? Are there any case studies that you've done that have been particularly interesting or that might be useful to people who are –

[00:54:25] IS: There are many and it depends. There are many talks at these Spark Summits, and we have one Europe and one U.S. this year, was in [inaudible 00:54:37] Moscone in San Francisco.

[00:54:39] JM: Yeah. I'm going to go to the next one.

[00:54:41] IS: 4,500 people. It's pretty large. The short answer obviously is it depends, but we always try to focus on a particular use case. We start with a use case. We help the customers to be successful with that use case. Because it's much easier once you see the success of one use case, it's much easier than to try to replicate it to other use cases. You asked about an example. I think, for instance, there are two talks maybe that – Two examples. One is Apple, which was a talk on these Spark Summits.

Their problem, it's a security application. Fundamentally, they want to get logs from all their notebooks and so forth and perform intrusion detections almost in the aerial time. It's a very large scale problem.

[00:55:45] JM: [inaudible 00:55:45] all the laptops in the world?

[00:55:47] IS: Not necessary. [inaudible 00:55:48]. Their own internal laptops.

[00:55:49] JM: Okay. Their internal laptops. Got it.

[00:55:52] IS: There's still a lot of them.

[00:55:53] JM: Right.

[00:55:53] IS: There is no solution out there, out of the box solution which will scale to the amount of data they need to process. They use Databricks for that and just huge amount of data, petabytes of data. They process this data in kind of real-time using Delta and using sparse trimming to figure out the – Detect the intrusions and attacks pretty much in real-time. Yeah, you can imagine that they need also – When performing the detection, they need also to look at the previous attacks to compare, "This is a pattern from the past." That's one.

I think the other one it's Regeneron, which is in healthcare, life sciences. What these companies do at the high-level, it's a very exciting area as I'm sure you've heard many times. Because now they have the data to do many of these syncs they were talking for – Everyone was talking for a long time, like personalized diagnosis, personalized treatment, drug development and so forth. The high-level the way they do that, now they have genomics data and you have genomics data and you combine with clinical data and the claim data and then you want to query all these data in order to try to figure out the answers and to give diagnosis to people, or like for instance I get your DNA. I sequence it and so forth, and then I am going to check. I have libraries or mutation or variants and I'm going to check whether you have any of those.

This will signal, maybe some of them will signal that you have a risk for certain disease. Then you can do personalized treatment. You can compare other people which are similar to you and for which there are successful treatments, say, for your condition and you can apply the same kind of treatments. Or you can try to figure out, like for instance when they do clinical trials. Clinical trials is very expensive. Before you can get approved – FDA approves the drugs, they need to do this expensive clinical trials. Then you want to look at all these people. They are getting you the clinical trial and you are starting to study what people in terms of DNA the people for which they respond to the treatment they have in common and what are the differences for people who do not respond to the treatment.

It's a huge amount of information they want, again, to process. For that, they need platforms to process each amount of data to find all the similarities or do machine learning. Yeah, we have quite a few customers doing exactly that.

[00:59:16] JM: In the AMPLab, companies got started based off of Mesos and Spark and Tachion, and these projects seem to overlap and they fed off each other in a productive way. It's clear that there was something magical that happened in that lab environment. Were there particular characteristics that you take away from that AMPLab environment that you're now applying in the RISELab or that you're now seeing emerge in the RISELab?

[00:59:47] IS: Yeah. I think that there are other few things there. So first, I was lucky to have some great students, like Matei and others, Ben Hindman, cofounder of Mesosphere, and [inaudible 01:00:05] who is the founder of Tachion, now [inaudible 01:00:09]. That's number one.

The other thing I think that was our close contact with industry allowed us to understand some of the problems very early on. With Mesos, we understand, we've seen that problem early on in how you're going to visualize a cluster. Before Mesos, you have to have a cluster for your production, for your experimentations. You need to have a cluster for the real-time stack or streaming, cluster for data historical analysis and so forth.

It was even harder, even more than that to deploy a new version, say, the time of Hadoop. It was very serious task, because there are quite a few – Not always error, backer compatible, and you have a cluster which is working how you're going to deploy. We saw that at the beginning of the cloud. All of these things was in the beta center, private datacenters. So you build another cluster to test a new version and things like that. You have to do that. Very complicated. Virtualizing the cluster was a very interesting problem and a very useful abstraction. That's what we setup to solve with Mesos.

Then with Spark, again, we have seen firsthand how – Challenges which were people had when trying to do machine learning, say, on top of Hadoop, and this was happening here in this lab, or doing query processing. Because at that time, I'm not sure if you remember, people started to build on top of Hadoop this hive and pig Latin, so to provide more SQL like interface. Now you build that and you have people who know SQL. Now I can query you on the big data, but they're coming from this databases which are pretty damn quick. Now you go there and, well, you need to wait for maybe an hour or more to get the result. It's not exactly what you're expecting. It's not exactly the interactivity you're used to and clearly that hurt their productivity.

We've seen, again, this firsthand. So that was how we developed Spark. Also, we also said that now we have Mesos. So it would be much easier to develop Spark, because we run on top of Mesos. We don't need to build all these resource management from ground-up, because this was what happened in the previous frameworks, like MapReduce and so forth. You have to build all the resource management from ground-up. That's one.

I think that you see that – What we've done is like – So beside great students, we are early on seeing some of the problems, which are industry confronted ways and we are able to use the system we already build to accelerate the development of the new systems. Tachion is the same. Tachion, we are basically one of the key – Taking a step back, one of the key characteristic of Spark was in-memory processing. This is how we are able to do machine learning much faster, because we didn't have to write data and read the data from the disk between every iterations, between every two iterations. This is how we are able to do interactive query processing.

But now you have to – You have this kind of cases in which you have multiple cluster, maybe a production cluster and maybe experimental clusters operating with the same data. Okay. Then if you have it as being two separate Spark cluster, even if they run on top of Mesos, each of these instances, cluster instances, they need to cache the data. It's like in-memory is still expensive, right?

Tachion was, one, allows them to share the data. It's also not only that, but now the data was allocated off Heap. Has certain advantages from being necessary allocated on Heap. In particular, allows us to a little bit get rid of the garbage collection overhead, which is nontrivial in Java. It also allow us to decouple the storage from the compute in the sense that now that if your computation fails, you don't lose a cache data. You just spin down your [inaudible 01:05:03] task and the data hopefully is still there and you can read it, say, in Tachion. I think these are what we are lucky to – Obviously, the last thing is luck. At a right area at the right time.

[01:05:20] JM: Yeah. I know we're nearing the end of our time. I just have a few other questions that are kind of random relative to what we've explored so far. The distributed systems people that I talked to, there seems to be – When it comes to cryptocurrencies, there seems to be a

bimodal distribution where either they're very interested or they're not interested at all. With you, I could imagine a level of interest, but maybe it's just not an industry yet. It doesn't really fit the thesis of, "Well, let's attack problems that are being encountered by industry today." What level of interest do you have in cryptocurrencies?

[01:05:58] IS: Less interest than I should. When I say I should, is that I've been very active in the peer-to-peer research early on. If you've seen some of the technologies and some people say it's the second coming of peer-to-peer. I think that probably it's also a matter of time and priorities. I think it's very interesting area, but I haven't had a chance to do too much or to look into much detail in the area.

[01:06:35] JM: Yeah, fair enough. What did you learn from working with Dave Patterson?

[01:06:38] IS: What did I learn? A few things. Number one is like you can only learn from his leadership skills and thought leadership skills. I think he's a very good organizer. He's a very good leader. You're talking about – We are talking about the AMPLab. We are talking about RISELab. This are labs which follows in a long list of labs which are started by Dave Patterson and Randy Katz more than 30 years ago. It's a very unique lab model.

In many schools or even in industries, labs are focused on a particular area. Their lifetime, it's unbounded at least. There is not a clear sunset for the lab. This is not true for the labs we have at Berkeley, for each of the lab, each lab is around 5 years and it has a clear vision, clear goals, clear deliverables. Actually, at the end of the lab, you can evaluate yourself whether you're successful or not. This is all thanks to Dave. All of these I learned from him. We just try to do a poor imitation.

The other thing I learned is that he's very astute in terms of realizing what are the [inaudible 01:08:03] industry. When we are talking about the research, is that it's the solution as a result of the problem you are working on and the trends. For instance, in the case of Spark, we did bet on memory, because we've done some studies and we realized that a large part of even MapReduce jobs, they can fit all their inputs in the memory even as that 10 years ago, 8, 10 years ago.

But Dave was very good in trying to – He's been always good in figuring out what are the dominate trends and bet on these trends. Now, he's betting and he bet early on specialized hardware. I think these are the two things – The other thing, despite his huge accomplishment, he's still very humble in the sense like, for instance, when he has important talks, he gives practice talks and gets feedback from graduate students. So that's what I would say.

[01:09:17] JM: Okay. Well, let's stop there. Ion Stoica, thank you for coming on Software Engineering Daily and being so generous with your time. It's been really fun talking to you.

[01:09:23] IS: Thank you. I really enjoyed it.

[END OF INTERVIEW]

[01:09:28] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]