EPISODE 1143

[INTRODUCTION]

[00:00:00] JM: Ray is a general purpose distributed computing framework. Ray is used for reinforcement learning and other compute-intensive tasks. It was developed at the Berkley Rise Lab, a research and development lab with an emphasis on practical applications.

Ion Stoica is a professor at Berkeley, and he joins the show to talk about the present and future of the Ray framework.

[INTERVIEW]

[00:00:28] JM: Ion, welcome back to the show.

[00:00:30] IS: Glad to be here. Thanks for having me.

[00:00:33] JM: So we're talking about the Ray ecosystem today, and I think it's worth starting off by just giving people an overview for what Ray is or a reminder.

[00:00:41] IS: Yeah. So the way we are looking at Ray is a universal framework for distributed computing. What this means is that Ray, generally enough, so you can build in principle, any distributed application on top of it. And the reason for its generality, is that it takes this two constructs you have in almost any programming language, particularly in any imperative programming language, functions and classes or objects and allows the programmer to transparently run them remotely and asynchronously.

And the functions give you the ability to support stateless operations, like side effect free. It's more – Think about more like about RPC, transfer and remote procedure calls. And an object when you instantiate an object from a class, in Ray, you create basically an actor, a stateful operator. And you can use to implement everything from model serving, training microservices. So you put these together, you are then going to end up with a very powerful framework and a very general one.

[00:02:01] JM: Great. And tell me about some of the applications that have been built on top of Ray or some of the frameworks that are built on top of Ray.

[00:02:07] IS: Yeah. This is something we are extremely excited about. And that is about the libraries and application, which are being built on top. Now and from our previous conversation, we have started almost from day one also building a set of libraries on top of Ray. Ray, again, is very flexible, very general, but its API is quite low-level. And even with high-level languages like Python and Java, a huge part of their success is a strength of their ecosystem, because they have a library for almost everything you want to do. So instead of writing hundreds of lines of code, we just make an API call and everything is great.

So we started to focus on a few machine learning libraries. We started reinforcement learning RLlib. We had tune hyperparameter tuning. And more recently, we added RaySGD, which is a simple library for doing some distributed training. And we have also Ray Serve for serving, for model serving library. So we have these kinds of libraries. And RLlib, it's extremely popular. Arguably is the most popular scalable reinforcement learning library. Tune is going very fast in popularity as a high-parameter search library. And Ray Serve, we also have a few companies already deploying it in production.

But what we are really, really excited about, it's about the third-party ecosystem of libraries. And in the past several months, we've seen a lot of traction. We have studied. We have a lot of integration with some of the most popular libraries out there. And in some cases, they use Ray in order to scale. The creator of these libraries, they use Ray to scale these libraries to go on a single node to multiple nodes to a cluster.

And here are few examples. On the natural language processing, we are having ongoing, hugging faces, and spacing, which are arguably is the most popular natural language processing libraries. So they are running on Ray, and now you can run them on a cluster. Other examples, we have integration with other high-parameter search libraries, like Hyperop, optune. Then another very exciting thing that's happening as we speak, an integration with Horobod. With Horobod, we can run Horobod in Ray. And now in addition to, obviously, a great library to do distributed training, you have access to many of the features of Ray. Like they ability to

launch clusters and manage a cluster much easier. We have auto scanner in Ray. You can use that right away. And then you can integrate with other libraries in the ecosystem. You can do high-parameter search. You can use Tune for instance, or you can use Serve for doing model serving. So I think that's very exciting.

Another thing which is happening, which is very surprising. It's, again, all of these third-party efforts come from the creators of these libraries. It's just growing the community. We have Dask. As you know, it's a popular general Python frameworks for distributed applications. Again, this comes from the community. There is an integration with Ray, so you can now run Dask on top of Ray and get better scalability and hopefully also performance.

And the other aspect which also is very exciting we see more and more traction from a platform, machine learning platforms who are using Ray, like of course you know that from last time we met, because SageMaker is using Ray RLlib for reinforcement learning. Now, in preview also, RLlib for AzureML. Analytics Zoo is a software stack data analytics and machine learning from Intel is now integrating Ray and many more.

[00:06:35] JM: Those are ton of examples of how Ray can be used. I do want to take the conversation slightly in a different direction talking about why it not be used for. We've had a few shows on this workflow scheduler that came out of Uber. Basically, a scheduler for long-lived workflows. So like, for example, if I'm going to take an Uber ride somewhere. That might be a long-lived application with lots of distributed tasks to be done within that workflow as I'm driving around in my Uber car, it takes an hour long. It's a distributed systems problem.

And I wonder if that is something that you could conceive of being built on top of Ray. Or is Ray more for like batch data science type of things?

[00:07:22] IS: Yeah. No. Actually, with Ray Serve – And we are going to have some exciting announcement that are coming, Ray Summit, which is just at the end of the Month starting on September 30th. Actually, we can run on top of Ray and we are looking great usage going forward. We can run production workloads, interactive workloads. If you think about – This is like another one of the big users of Ray and group, it's financial, Alibaba arm, which was basically Alipay, the former Alipay. This is one of the biggest financial service. They deploy Ray

massively. And the kind of things they are deploying it is not necessarily for data science. It's for like application, like recommendation, right? It's like basically when you are going to login in their application, you get some recommendation about what services to use. Maybe what product their affiliated by and things like that. And that basically, it's an online learning application. It's basically you get the request from the user and obviously you provide the recommendations.

But then you update with a high-frequency. Every few minutes, you update the model to take into account new informations from the users. What they like and what they didn't like. So you merge with a user ground. From the new data and from that, you are going to continuously improve the recommendations. They're using also fraud detection. And like one particular example is money laundry. Before you approve a transaction, ideally, you want to figure out whether that transaction is part of the money laundry activity. And each transaction, if you think, it's an edge on the transaction graph, on a huge transaction graph, right? You add another edge between two entities who are making the transaction.

And money laundry, typically, you detect it by detecting cycles in this transaction graph. And doing that in real-time is pretty challenging, because you add a new edge and you need to figure out that's part of kind of a cycle. And there's also some machine learning involved there, because you don't have absolutely every transaction in the world. So that's another example. So it's like you have a streaming plus graph processing together and you want to do that pretty much in real-time. So they use Ray for that as well.

Ray, like I mentioned, is very general, and this is exciting part of it. And as I mentioned, Ray Serve, which is basically doing model serving, but it's not doing model serving. You can also add business logic. This is one of the things which is exciting about Ray Serve, because it used to be, if you look in fact at most of these backends, right? Best service backends. It's a huge serving applications, right? Because you get the request and you provide a reply, an answer, right? And a request can be a text query, right? And you return back a bunch of documents, or it can be an image like in Pinterest and you return back some of the similar images or something like that.

So everything – Like Uber, right? Like you mentioned. You ask for an Uber, and you are going to

get back a reply how long you may need to wait and things like that. This backends, actually they're obviously very, very complex. But the point here is that there are more and more marrying classic work set being done. [inaudible 00:11:26]. This logic about what do you need to do and how do you need to do and so forth. And when a request comes, how do you route that request in this kind of microservice deployment? And then with machine learning models, right? Because we are incorporating more and more machine learning into the decision framework into this analyzing a request and returning the possible answers.

Ray Server is in a unique position to be a platform which can unify, bring together this classic work services with model serving. So this was the long answer for your question. But the short answer is that, actually, we do believe that Ray is very well-positioned for this kind of workloads and long living workloads. And now, of course, you ask about your original question also was, well, what is Ray not good at that? It seems, "Hey, there's be something which is not good at."

And I think that, for instance, you have job orchestration frameworks, or pipeline orchestration, like Airflow, and Uzi, and things like that. You still need that, because that's a level of taking – Have multiple jobs and you want to orchestrate them. You have a more complex pipeline. You can implement the jobs in Ray and so forth. But you still need that higher level job orchestration framework.

[00:13:07] JM: Very interesting notes. So what other applications do you anticipate being built on top of Ray that have not been built yet?

[00:13:15] IS: I think that one exciting thing is, in general, about a platform which is general. Is that you can run on that platform workloads. It was hard run before, because before you may need different systems to run the same workloads. So it is putting them together. And I think that we have this like – There are analogies like iPhone, right? Before the iPhone and smartphones, you have different use cases or different capabilities, which are embedded in different devices, like GPS for getting directions, or maps, right? And getting directions, driving directions.

You can remember like cameras, digital cameras, or you can have Flip. It was this device for recording, or the MP3 players and things like that. And then something like iPhone like bring all of these together. So now you can – It's an application on your device. And now you can build

new application, which can use like virtual reality. You can put that into – Put together in a game. And you can move in the virtual world. Things like that, which you are about the location, because you also have a GPS.

I think that's one direction which is exciting in that bigger picture. So the examples I have in mind and which I talk a little bit about them was that embedding, now, you can have one system to run almost your entire background, right? And business logic, plus machine learning, you can have – Say, many companies don't run only one model. Run a huge number of models in their backends. And then maybe the business logic to figure out which models are they going to use. How they're going to route the request between this model.

For instance, if you look at the image and you may want to identify, look at the license plates, like when we have – You can have a model which look at your image, and then if the image has a car, you identify a car, only then you can look after to identify something about that car, like a license plate, to read a license plate, right? If it's not a car, you don't need to do that. That's an example, a very simple example of business logic.

There are other examples of business logic in which are very useful, like for instance some industries. You need to provide – There are some regulations in terms of the decision you are going to make. And you want to verify before you give an answer back that these regulations are met. So that's another example. For instance, say, one example is about you make a decision on something like a mortgage, right? You apply for a mortgage, and I'm a bank, and I'm going to decide on that mortgage. And it turns out that if I deny you, you can ask me why I denied you. I need to explain you. And the reason for that, I need to make sure to tell you and to assure you that I didn't deny you because using some, say, racial profiling or things like that, right? Which are illegal.

So then if I was a machine learning model, a deep learning model, it's pretty hard to explain how the decision has been made. So that's why. On the other hand, before I give you the answer, I kind of try to double-check the answer to make sure that it's in – So to speak, it cannot be interpreted like being based on the race or something like that, which somehow the model may have picked it up. Then that's one way to use a machine, deep learning model, which I cannot explain an application. Requires me by regulation to met some rules and even be able to

explain that I didn't use in my decision some of the inputs, which are not against regulations, right?

So I think that's one exciting thing. Marrying classic set of application with deep learning application, building it on an end-to-end platform. The other thing is that, one example of this, is like I mentioned, like money laundering. The examples I gave to you earlier. And in that example, again, I do need to graph processing and streaming. This is basically what you need to do. [inaudible 00:18:25], right? Because you have streaming application, you have graph application, large scale graph application. But if you put them together, it's hard, because you need to manage this to different systems. You need to move the data between them. For instance, many of the graph engines, graph processing engines, are not, so to speak, interactive. They don't work in a streaming regime, right? Every new – You get a new request that you change in a graph. And you need to evaluate something on the graph. And you need to give an answer in basically real-time.

I think these are some of the things I think that are very exciting. Like for instance, the other example like I mentioned, and this is again talk about the richness. Once you have on the ecosystem, you can put together. Like Horobod today, you do training. Then you need something else for serving, right? Maybe [inaudible 00:19:25] or something like that. You need something else for high-parameter tuning, like Hyperop or something like that.

So, again, it's like you can do it today, but the barrier is a little bit high. Because, again, there are different systems, you need to put them together. But with, for instance, Horobod running on top of Ray, it's the same system. Now you have access to ingest another library you use to do high-parameter tuning and another library just to use to serving. And this tremendously lowers the barrier of, I believe, developing new exciting applications. It's like in Python, right? It's in Python. I can use Pandas to do data processing and get some insights. Then I get [inaudible 00:20:15] to do numerical computation. I can do it very easily. Use them in the same application.

So that's how we hope that at the end of day will be as easy to mix these kind of different workloads and to build this end-to-end application in Ray as it is today to mix and match different libraries in traditional languages like Python and Java.

[00:20:44] JM: And I think that one thing that underlies a lot of these is just that machine learning is way too hard to use today. And that's why there's kind of a narrow set of use cases for machine learning relative to where machine learning could be used. Like we're talking over Zoom right now, and if machine learning was easier to use, I wouldn't even have to click the end meeting button. There would be enough inputs that Zoom would just know when the meeting is over and it would just end the meeting instantaneously. But today, that's just not realistic to implement that, because it's too hard to use. But I can imagine if the abstractions were right, it would be very possible.

[00:21:29] IS: Yes. Absolutely. These are exactly – That's a terrific idea. And look, let me take a step back here. While we've been doing Ray in the first place and we started Anyscale. It's because things are hard. In particular, what is hard is building different application. All these applications that we are talking about, they are kind of distributed. They ran on a backend and they need to scale more and more and so forth.

So that's a premise. The premise is that more and more of this application are going to be distributed. And [inaudible 00:22:11] application is very – And one of the reason building this application is hard is because there are many pieces you need to put in place. You need to assemble them. And many systems you have to use. And when you have to use multiple systems, it's hard on many levels. It's hard to write the application, because you need to write across different systems. Well, people, ideally we want to write – It's almost like ideally some monolithic application. And I say monolithic application, it's not all that code is on one file. You have in one repository, right? Let's think about in one repository.

But you can write it, you can test it. It's like what'd you do on an application on your laptop. You could develop and you run it on application on your laptop. So you want this kind of experience. That's what people want. But then, again, when you have different systems in place, it's very hard to achieve that. And then you need to deploy it and you need to run it in production and it will manage it. Again, when you have different system, different moving parts, it's incredibly difficult. So that's what have helped to do with Ray, to simplify that process. Because, again, many of these pieces are going to be like libraries on top of the same platform on Ray.

So I think that's where we believe that is going to simplify. And, again, right now we are really

focusing on targeting developers, people who build this application, people who build these libraries. But I do hope that we are going to – Because you simplify and you make it much easier for developers to build this kind of distributed large scale applications, make it much easier to put together in the same applications do machine learning, do deep learning. Like I mentioned, business logic with this together. I think that we do have to see a tremendous and very exciting new applications.

Another thing I wanted to say about this application, one of the things which leads to our focus on distributed applications is that – And I think we discussed last time is the fact that the demands or the application we are talking about, which do machine learning, which might data processing, are increasing much faster than capabilities of a single node, of a single processor even if you are talking about specialized processor like GPU and TPUs, right?

I was looking over this data and, of course, you have the open AI plot, this open AI 2019. They have this famous plot which shows that the computation requirements to train the state of the art models, machine learning models, are doubling every 3.5 months, which is equivalent growing 35 times every 18 months. And I was looking also recently, if you look at the size of the models, the state of the art models, for the 4 years into is like 20 times every 18 months, right? And now with GPT, if you look for the past 2 years with the GPT 1, GPT 2, and now GPT 3, they increased like 200 times every 18 months. So these things are huge, right? These increases are huge.

For comparison, Moore's Law used to increase – What? Is two times improved performance every 18 months. So two times. So times when you compare, two times with like 35 times or 20 times or things like that. It's like order of magnitude difference, right? And we are talking about Moore's Law, and we also know that Moore's Law has ended. So this is a thing, right? It's like you have this kind of huge disparity between the demands of these applications and the capabilities of a single node.

And then you have also a lot of applications are fundamentally distributed in nature, like backends, like we are talking about. Our premise is that because of this trend, the distributed applications are going to be the norm. No longer the exception, right? So almost in 5 years from now or 10 years from now, almost everyone will build distributed application from the beginning

to start this, right? Is still not going to be afterthought.

And if that will happen, obviously, for that happen, we need to clearly lower the barrier of building these applications, right? Because today, it's super hard to do it, right? How many people build distributed applications compared with the total number of developers you have out there? A minority. Very small number, right? I think that's what we are so excited about Ray and why we build Ray, because we do believe that it will make a big step forward in building this distributed application generally through the applications much easier.

[00:27:29] JM: So you mentioned a lot of different aspects there. One element was the hardware trends. And we are speaking in anticipation of Ray Summit, which you're going to be speaking at. It's a virtual conference about Ray. I encourage people to check it out if they have a chance. And somebody who's speaking is Dave Patterson. He's been very interested in computer architecture for a long time. So when you talk to Dave Patterson about these trends in hardware, what kind of insights has that brought up and how has that helped to lead to how you're constructing Ray?

[00:28:04] IS: Obviously, Dave, has been – He's a very insightful person. And his [inaudible 00:28:10] doc. It's about specialized hardware. We are living in the golden age of specialized hardware. Look now, Nvidia, very valuable company, right? At least they exist. And Google build the TPUs. Almost every other cloud providers build specialized hardware. And all of these – Again, let's go back to what is our original problem. So the original problem is that the requirements of this application, in particular machine learning application, are growing much faster, the hardware capabilities, right?

So you need to solve this problem, right? And again, the numbers I talk about, 35 times growing movement. It's compounding, right? It was exponential curve, right? So in order to bridge this gap, the performance gap, what you can do? One answer, great answer, specialized hardware. You specialize hardware. You trade generality for performance, right?

If I do a single operation, I can make a very fast hardware for it. If I need to 1000, different type of operations is much harder, because I need to do all these tradeoffs. In some sense, this is our specialized hardware you're doing. And don't get me wrong, specialized hardware and

GPUs and TPUs are much, much faster than a single core processor, right? They are on a different level. There can be order of magnitude faster than 1 core, right?

Now, the point is that these specialized hardware, still, they cannot – Once you have the architecture and you can slightly improve the architecture. Still, the performance is going to increase as a Moore's Law, right? Because it cannot cram more transistors, right? You get the main advantage because of this tradeoff and you have the architecture, which is specialized for you for a particular workload. And you're going to see more and more on this.

But it's not going to be enough, right? Another example why it's not going to be enough for the foreseeable future. Everyone talks about processing, but the memory, it's a big deal. And it's a memory bandwidth. And then it's a memory capacity. I'm going to focus on the memory capacity. If you look about – I mentioned earlier that the number of parameters, the size of the networks, the deep learning networks, are increasing faster than ever. I think like around 20 times every 18 months over the past 4 years.

And for the last two years was insane, because GPT3 is what? 170 billion parameters, right? So if you look about this and if you look now a GPU memory, right? Because if a GPU memory increased, now you have 32 gigabytes. You have on V100, I think it's 40 gigabytes on A100, the latest Nvidia GPU. I think TPUs are the same of magnitude in terms of the RAM of the memory on the GPU.

Also, if you look over here, it doesn't increase much. I think it increased – I looked, I think, for the past 8 years, I looked at the Nvidia GPUs. It was like 1.5X every 18 months. Nowhere close about how much the size of the deep learning models has increased over the same time period.

So here you are, right? Memory, you cannot do much about memory to specialize it, right? Specialization doesn't help us much, because with memory, for every bit of information, you are going to still need a few transistors to store bandwidth, right? And that's why right now, when you trade this huge models, they train it on many, many GPUs, hundreds or thousands. Some of they say is prohibitive for a few companies to train these models. And that's why now we are talking about refinement. You train it. You spend all these money to train it – I don't know, hundreds of thousand dollars. Maybe even millions of dollars. And then you have the train

network and now you fight for your dataset, right? We just improve it, incrementally improve it.

But it's what I'm saying. So now, yes, specialized hardware is amazing. And we are going to continue to improve it to bridge the gap. But it's not enough. Then what do you have to do? Is also to go distributed using multiple chips, right? This is what every company is doing, right? And that will give you another two, even three order of magnitude to scale, right? Because you can use 100, or 1000 chips, or things like that. It's a good improvement.

Of course, there'll be also a lot of efforts to make [inaudible 00:33:15] now to reduce the requirements of some of these security networks. Desecuritization is like quantization, right? It's like you want to use instead of 64 bits flopping point, maybe you can get away with 80 bits, right? And there are a lot of efforts to compress these networks once you learn this and you sell them.

Yeah, all of the research on that side. But just to summarize, let's make no mistakes. The demands today are going through the roof. And they are much, much bigger than what one [inaudible 00:33:51] can provide you and one machine can provide you, right? And the gap is just increasing, right? Because exponential curve.

So then that's the problem. How do you solve the problem? There is no one solution. Specialized hardware, going distributed. Distributed, distributing the computation. Obviously, algorithmic improvements are all part of the solution to bridge this performance gap.

[00:34:20] JM: Are there any benefits to be gained on the cloud computing side of things? Because we're executing a lot of these workloads on the cloud. Does the cloud offer the right primitives that we need to run these kinds of applications?

[00:34:36] IS: Yeah. So it's a great question. Let's step back, because I just want to make sure that I provide all the context. So let's you believe me that we need to write distributed applications, and more applications become distributed, right? Clearly, in order to have that, you need to have the resources and you need to be able to make it much easier, because like we discussed, writing distributed applications today is incredibly hard. And if most of the application are going to become distributed, you need to make possible for everyone to do it.

And obviously, one of the questions her is about how do I get access to lots of resources, distributed resources? How am I going to manage them? How am I going to start the cluster and things like that, right? Now, cloud, it makes easier. Make it easier for me to get access to these distributed resources, right? It's terrific. But when it comes to, obviously, running distributed application, managing them is hard, right? It's still hard.

Now, what is the answer? One of the answer today to make that easy, right? To make using a large number of CPUs very easy for the developers, and I say it's for the developers, are these serverless? Lambdas, and cloud function and things like that? Which allows you to run many functions in parallel, thousands, or whatever, or even more, without – By abstracting away the hardware, you don't need to set up a cluster, right? You don't need to set up to allocate 100 instances and run on them.

However, the serverless offerings right now, they do have a lot of limitations. They are very good for things like to trigger the computation when some new data comes in and to process it and then you are done, right? You just pay only for how much you process it. You don't need to provision anything to process that piece of data who just landed on your storage. So that's great.

However, you can only have some functions. You cannot, for instance, use – It doesn't work well because you cannot support stateful computations like actors or microservices. You can do – You cannot build. It doesn't work well for deep learning training, right? It doesn't work well for serving. It doesn't work well for all these implementing all these business logic, which you deploy it on – We call it today on microservices. As a functions themselves, this offering themselves. Right now, although I don't think there's any fundamental limitation, but still, they don't support GPUs. They don't support TPUs. For now it's only CPUs.

It's also a little bit fragmentation, right? The most popular offerings are coming from cloud providers, but each cloud has its own version. They also have limits about for function, for instance, can run, or cloud function can only run for – I don't know, 15 minutes or something like that. It depends on the cloud. So you have all these limitations.

So I think that on one hand, this serverless idea, it's absolutely the right one, because it's good for the developers. It lowers the barrier for the developers to write these applications. However, there are many limitations. And I do think that – And with Ray and Anyscale, we are looking to remove many of these limitations.

But I do think that that's the general direction, and I think a lot of people in the industry and research and we hope we are going to give a big help here. Looking to this, basically you want ideally to be able to use a cluster like your laptop. That's one of the holy grail, because if I ask you or I ask the developers what is their favorite environment to develop? I think many of them will tell my laptop, right? Because I want to use my same workflow and so forth. But on the other hand, they also want to have a lot of more resources than you have on the laptop. A lot of different types of resources, GPUs, TPUs. You don't have TPUs on your laptop. They want to be available. You cannot deploy a service on your laptop, because when you shut it down and you close it, that service is not available.

I think that what they want is like they want, on one hand, to have the experience of development and running applications like they have on their laptop, but then they want to also this laptop to have unbounded, like infinite resources to be always on. We are referring to this like – They want really an infinite laptop. So that's basically in some sense – It's again, a long answer to your question. But if you are looking about the cloud, so in short, absolutely. The public clouds will do lower the barrier for developers to get access and companies to get access to large number of nodes, which otherwise would be extremely expensive.

But we are just at the beginning. There are many more steps to really make this, the cloud infrastructure, transparent to being like magic, right? You just run it and your applications will just use how many resources it needs. Not more, because you don't want to spend more. And everything is available, and also it's easier to develop the application on your laptop. The same experience.

[00:40:25] JM: We've talked barely any about reinforcement learning in this episode. And reinforcement learning is one of the hallmarks of Ray. So the last conversation we had was really centered around reinforcement learning. I wonder if you've had any insights in the last couple months about reinforcement learning and how that pertains to Ray.

[00:40:45] IS: [inaudible 00:40:46] so RLlib was the first libraries we developed on top of Ray. Actually, it did have a significant impact on the Ray architecture. And also, Ray has to sync to RLlib for generality, because RLlib, reinforcement learning like we discussed last time, it's a very general workload, right? Like we discussed. We need to do simulation, large scale simulations. We need to do training. We need to do serving, serving the model to take decisions based on the observation of the environment. So you need to do all of those, right? That's why you need a general substrate, and that was we build Ray for.

We do see more and more reinforcement learning applications. We are seeing that in warehouse fulfillment. We are seeing everything, designing engines, designing like sailing boats, designing drugs. We have someone using Ray for COVID research. It's like to design drug vaccines for COVID, and many more. So we see an increasing number of reinforcement learning applications. I think that what we discussed during this podcast is from – You've seen some of the reflection of the fact that we've seen a lot of more growing in a lot people started to know and learn more Ray, they started to use to doing other things. First, more and more things on the machine learning. And we haven't even discussed about things which are not machine learning related, like backend testing and things like that.

I think you see a reflection in broadening the scope, broadening the range of application that Ray is supporting. But I think that one thing to keep in mind, and when I reflect, I'm thinking about that why this happened. It did happen because reinforcement learning is such a general workload, much more general than just doing training. Much more general than just doing serving. And in order to support reinforcement learning, you need to put everything together. So I think that's one thing.

So, the summary is like reinforcement learning and RLlib is growing. Now, we do believe we have the most complete set of algorithms. We do have [inaudible 00:43:21]. We have parity between TensorFlow, and PyTorch, and – Yeah, a lot of more use cases. I think that's very exciting. That's one thing. So it's growing nicely. It's very exciting.

And the other thing is that the one things which is different from the last year, we are seeing a lot of more interest in other applications from the community. And in particular, and a lot of this

interest, and I believe this will continue, the scope of Ray will continue to broaden with this ecosystem of libraries, because hopefully more and more people who are using these libraries, they're going also to use on top of Ray.

[00:44:06] JM: Well, that seems like a great place to close off. Ion, is there anything else you want to add about Ray or the Ray Summit?

[00:44:12] IS: Everyone should take a few hours, attend the Ray Summit. It's a free event. And it's our first one. So you are going to hear about [inaudible 00:44:24] announcement and a lot of excitement talks from the community. And it's on September 30th and October 1st. So I hope to have you all there tune in. Thank you.

[END]