SDS PODCAST EPISODE 886: IN CASE YOU MISSED IT IN APRIL 2025

Show Notes: http://www.superdatascience.com/886

Jon Krohn:	00:05	This is Episode #886, our “In Case You Missed It in April” episode.
	00:19	Welcome back to the SuperDataScience Podcast. I’m your host, Jon Krohn. This is an "In Case You Missed It" episode that highlights the best parts of conversations we had on the show over the past month.
	00:31	Our “ICYMI” in case you missed it this month starts with Sama Bali and Logan Lawler. Sama is from NVIDIA and Logan is from Dell, and in Episode 883, I asked them about libraries like CUDA that make up the AI software stack on NVIDIA GPUs. I loved the scenic route that Sama took me on to get us there, as it knocked so many novel concepts in AI and emerging tech into place. Here, she talks about NVIDIA’s new software and services, and how they interconnect. Here she talks about NVIDIA's new software and services and how they interconnect.
Jon Krohn:	01:03	I want to get back to the NVIDIA story from around the time and this visionary nature of what NVIDIA's done and reflected in their share price is this idea that, okay, deep learning's going to be gigantic or let's assume that deep learning's going to be gigantic. So let's build a software ecosystem. Going back to your point earlier, Sama, that supports that. So yeah, so tell us about things like CUDA, TensorRT, maybe a bit of the history and why those are so important in this GPU ecosystem and in this AI era.
Sama Bali:	01:31	Yep. I'm actually going to start first with NVIDIA AI Enterprise, just completing the story of how we are doing things, especially with Dell Pro Max AI PCs. So think of NVIDIA AI Enterprise as our version of end-to-end software development platform, which is helping you not just accelerate your data science pipelines, but also really helping you build next-gen. It can be generative AI applications, it could be computer vision applications, it

can be speech AI applications, and it has a lot of components. We've got NIM microservices.

02:04 This is how we are delivering all kinds of AI models as containerized microservices. So literally think of any other AI model in the world. We work with open source partners, proprietary partners. We have our own NVIDIA AI models as well. We are taking each of these AI models, putting them into a container and then adding our, I wouldn't say secret sauce because everybody knows about TensorRT LLM and all kinds of services, which are really helping you get the best inference possible on NVIDIA GPUs.

02:40 We are offering them as microservices. The reason being, and you'll soon start seeing this from an NVIDIA perspective that we are providing almost all of our AI software as microservices is because things are changing quickly. I'm a developer today who built an application with Llama 3, and guess what? In two months Llama 3.1 comes and then another two months, 3.2 comes up.

03:02 So we want to make it really, really easy for people to just swap in the model as quickly as possible without really disrupting that entire pipeline. So that's NIM microservices. We've gotten all kinds of models from if you want to build a digital human to actually building speech related applications to now. We also have NIM microservices for our reasoning AI models as well. So that's the first component of NVIDIA AI Enterprise.

Jon Krohn: 03:28 Really quickly before. So it's going to be obvious for sure to you, to both of you, as well as to many of our listeners, exactly what a microservice is, but could you define that for our listeners that don't know, just so that they understand what it is and why it's important, why it's helpful?

Sama Bali:	03:43	I actually don't have a definition of microservice.
Logan Lawler:	03:48	I'm going to give you not a textbook definition, but I'm going to give you a practical definition.
Jon Krohn:	03:55	Cool.
Logan Lawler:	03:55	Is let's say you're a data scientist and you have created, let's just pretend, a chatbot with Llama 3. You create that without a microservice, without an NVIDIA NIM, like Sama said, every time that model updates, if there's security, all this stuff, you're doing a ton of, I hate to say it, but background tedious work to get that to a point where you can deploy it.
	04:21	Where when things change, for example, if you don't, that's the whole point of a microservice with NIM is you basically can load that to literally one line of code and the LLM part of it is really done for you. It is containerized, it's packaged, it's ready to go. So a data scientist can focus on, well, how am I going to customize it or building whatever application wrapper around it. Versus like, I need to update the code here to get this to connect.
	04:45	That's really the point of a NIM is how quickly can I leverage the power of an LLM vision model, whatever, with one line of code. That's the power of a NIM. It runs on a workstation too. It runs on Dell Pro Max servers. It runs pretty much everywhere.
Sama Bali:	05:00	Yeah, that was going to be my point. That the key point being with these NIM microservices, you don't have to make sure that the AI model is tuned to the GPU. We've done all of that work for you. So as soon as you're downloading this locally on your Dell Pro Max PC, it already understands the kind of GPU it's running on. The only thing you have to make sure is the model you're downloading fits onto your GPU memory size now, but

with 96 of memory, you've got the entire world for you here.

Jon Krohn: 05:28 Nice. So as you've been speaking, I've tried to look up quickly online what NIM stands for. It doesn't seem to stand for anything that I can find easily.

Sama Bali: 05:39 Okay. I'm going to let the secrets out. It actually stands for NVIDIA Inference Microservice, but then we also use NIM Microservice. So it's like chai tea kind of a thing. They mean the same thing.

Logan Lawler: 05:51 Potato, potato, yep.

Jon Krohn: 05:52 Potato, potato.

Logan Lawler: 05:53 Potato, potato.

Jon Krohn: 05:55 A potato brand potato.

Logan Lawler: 05:56 Exactly. Cheese queso. That's what I would say. I go to a restaurant, I'll say I want cheese queso, and then my wife always gives me a hard time. Yeah, cheese queso.

Jon Krohn: 06:06 Nice. Yeah, now I understand perfectly. Thank you for giving us that insight. It is interesting. It isn't something that's very public, so people really are getting the inside scoop on NIM. Yeah, it's just spelled N-I-M for our listeners who are wondering what word we're saying. It's exactly like it sounds. N-I-M, in all caps. I'll have a link to that in the show notes, of course. Anyway, so I interrupted you. Oh, go ahead, Sama.

Sama Bali: 06:28 Oh, I was just on the same topic of NIM microservices, I was going to say, we've got a website called build.nvidia.com. That's where we host all of these NIM microservices. It's a good website to not just go try out these different kinds of AI models. You have the ability to prototype on the website itself. There are no charges for it

at all. You can see models, again, by all kinds of partners that you work with, including NVIDIA models as well.

06:56 They're segregated by the industry you work with or the use case you're trying to build. So it's easy to maneuver around, find the exact model you want to work with. Then once you want to download this, we've made it easier. So if you really sign up for our NVIDIA developer program, we actually let you download these models and then continue to do your testing, experimentation, free of cost. There are no charges at all. So you can continue as a developer. I would want to go try out different kinds of models, see what's working with my application. So we like to do that as well.

Jon Krohn: 07:27 What I was going to say, and I'm glad that you had more to say on NIM microservices because my transition was going to be that the last time I interrupted you, you were about to I think start talking about other aspects of the AI enterprise. So now I'll let you go on that.

Sama Bali: 07:43 So outside of NIM microservices, we've got NeMo. Which really helps you build, train, fine tune your own models, but also gets you the ability to add guardrails to your model so that whenever you're deploying your application, you are making sure that the application gets used exactly the way that you want to do it itself.

08:02 We've got AI blueprints, think of these as reference AI workflows. So we give you the ability to build different kinds of AI applications. So think of this as a recipe. You've got the step-by-step process to actually build an application. There's a reference architecture, but we also get you the ability to add your own data to it. That's what gets every company their own edge. You want to add your data, which is your differentiation at this point in time. So you have the ability to build different kinds of applications.

	08:33	What else do we have? Oh, we've got different kinds of frameworks and tools. So we actually do support different kinds of AI frameworks like PyTorch, TensorFlow, we also have our CUDA library. So I think this is a good time to talk about CUDA as well, which really stands for Compute Unified Device Architecture.
Jon Krohn:	08:53	I didn't know that. I didn't know that. I've been using that word for a decade now. Thank you.
Sama Bali:	08:59	So this really has been playing a crucial role in AI development by enabling efficient parallel computing on NVIDIA GPUs. So the idea was its entire architecture really helps you train different kinds of models significantly faster, which means that you can in some scenarios actually reduce your training times from weeks to days. It is also helping you get better and better inference.
	09:28	So you see higher inference performance on NVIDIA GPUs because of this architecture of parallel processing if you're comparing it to just CPU-only platforms. We now have, and I'll have to look up the right number of how many CUDA libraries we have, but we've got tons and tons of these CUDA libraries. These are GPU accelerated libraries.
	09:52	So a good example I'll give you is of RAPIDS cuDF. So the idea, and Logan touched on this earlier as well, is the way RAPIDS cuDF works is that it tends to mimic the APIs of a lot of data frames like Pandas, Polars. So if you are in that process of pre-processing your data in your data science workflow, it can actually accelerate that entire process by a hundred X on our 6,000 GPUs without any kind of code change.
	10:28	That's the beauty of it. That as a data scientist, all I'm doing is adding that one API line of code and then it

actually accelerates entire process by a hundred X. So that's massive time saving from a data scientist perspective.

10:43 At GTC, we announced cuML, which is again one of our CUDA libraries as well. This is helping you accelerate your machine learning tasks as well. So if you're using scikit-learn, you have the ability to go up to 50 X acceleration for your ML tasks as well. So each one of these libraries, and as I said, we've got tons of these right now, but depending on the data science tasks that you're doing, these are all designed to then offload that work to the GPU so that you can see that massive acceleration.

Jon Krohn: 11:14 From NVIDIA's AI Enterprise, we turn now to AWS's Graviton and Trainium 2 chips. I'm taking this clip from my conversation in Episode 881 with Emily Webber, a Principal Machine Learning Specialist from AWS's elite Annapurna division. In the clip, Emily explained why one might elect to use a specialized AI accelerator over a GPU.

11:39 So let's start there and you can tell us about this Graviton chip, the Trainium 2 chip, and maybe this kind of relates to a general question that I've been meaning to ask you this whole episode, and I have it just continue to forget with each wonderful explanation that you give after another, which is that why should somebody, why should a listener, for example, consider using an accelerator like Trainium and Inferentia instead of a GPU? Maybe that's a great question to start with, and then I'll remind you of the other, the series of questions that led me to that question.

Emily Webber: 12:09 Sounds good. Thank you. Thank you. Yeah, so I mean, fundamentally at AWS we really believe in customer choice. We believe in a cloud. We believe in a cloud service provider that enables customers to have choice about data sets, have choice about models, and have choice

about accelerated hardware. We think it's good for customers to have that ability and to have real options that is ultimately best for consumers and that's best for customers. So fundamentally, that's the direction. Annapurna Labs is an awesome company. Annapurna Labs has been building infrastructure for AWS for many years. So Annapurna Labs is a startup that Amazon acquired in 2015 primarily to develop the hypervisor, actually.

13:03 So they developed what's called the Nitro System. Yeah, we'll talk it through. They developed it. Yeah, it's like the coolest story in tech that is the least told. So here's the scoop. So in 2015, the way people were doing cloud 10 years ago is you had this thing called the hypervisor. And the hypervisor essentially was this giant monolithic software system that managed the entire host of all servers. And the challenge with the hypervisor systems is that it made it really hard to innovate for the cloud, because all of the control, the communication, the data at the server level was implemented in this giant monolithic thing called the hypervisor.

14:00 So Annapurna had this crazy idea of decoupling the parts of the hypervisor that you need to scale up the cloud at the physical level. So they developed what's called the Nitro system today, which provides physical separation for things like the data that's running on the instance from the communication that's controlling the instance. And so, this is both how AWS scales and how AWS provides such strong security guarantees, is because physically there are two different controls. There's one physical chip or there's one physical component of the hardware system that is managing the data, the customer's data, and there's a different physical control that's managing the governance of the instance. And so, every modern EC2 instance today is built on the Nitro

system. So that was the first major development for Annapurna Labs was Nitro.

Jon Krohn: 15:06 So that's Nitro, like nitroglycerin, N-I-T-R-O?

Emily Webber: 15:09 N-I-T-R-O, yeah.

Jon Krohn: 15:12 Explosive.

Emily Webber: 15:14 Yes, yes. So after the Nitro system, Annapurna started developing their second sort of main product line, which is Graviton. So Graviton are custom CPUs, custom ARM-based CPUs developed by Annapurna Labs. And if you watched Reinvent, one of the highlights that you saw is that today more than half of new compute that comes onto AWS is actually Graviton CPU.

Jon Krohn: 15:47 Ooh.

Emily Webber: 15:49 Yes. So when you're looking at instances on AWS, when you see that little G at the end of a family, so like a C6G or even a G5G, that second G means it's a Graviton CPU. So that means you're going to get much better performance at a very competitive price. And so, the Graviton CPU is our second main product line. And then, Trainium and Inferentia is the third main product category from Annapurna Labs, which is now let's take this awesome ability that we've created in developing infrastructure and scaling infrastructure across AWS, and let's focus that on AIML. So Inferentia of course was developed and came out a number of years ago. Trainium 3 is our third generation chip. So it's the third generation accelerator for AIML. And that is why it's such an exciting moment, right?

15:53 Because you see the breadth and the scope, and the incredible results that Annapurna has delivered over the years. And now this is totally focused, and now a large

focus is AIML. So when customers are taking advantage of this, fundamentally they're interested, because they get the benefits of price performance. More than anything, it's this benefit of highly optimized compute that is scarily energy efficient. Annapurna is so good at identifying improvement areas to just take cost out of the equation and reduce complexity, and pass performance, and pass cost savings back to customers while meeting performance and in many cases, exceeding performance. So Trn2 is actually the most powerful EC2 instance on AWS for AIML, like full stop. When you look at the performance metrics that we're seeing, it's a very exciting moment. It's an exciting moment for customers, exciting moment for the whole group.

Jon Krohn: 18:09 Trainium 2 is the most powerful on AWS?

Emily Webber: 18:12 Correct.

Jon Krohn: 18:14 Performance and power are certainly crucial for measuring chip efficacy. But what good is AWS' chip if it's not being used in deploying AI models? Deployment can become a real problem for teams where data scientists outnumber software engineers. In Episode 879, I talk about the issues of model deployment with Greg Michaelson Dr. Greg Michelson, Co-Founder and Chief Product Officer of Zerve.

Jon Krohn: 18:40 So another kind of tricky thing that data scientists maybe even myself have difficulty with is deploying AI models. So something that's been intuitive for me for literally decades is opening up some kind of IDE, Jupyter Notebook, something like that, and getting going on inputting some data, doing some EA and building a model.

19:06 But the thing that hasn't been intuitive to me, and it's probably just because I haven't been doing it as much, I've had the luxury kind of working at companies where

machine learning engineers or software developers, backend engineers take and then the model weights that I've created and they put them into a production system. So on a smaller team or on a team where there's huge demand for software engineers, which is often the case, you can end up having more data scientists creating models than there are software engineers to deploy in a lot of companies, that creates a bottleneck. So how Zerve's built-in API builder and GPU manager remove those kinds of barriers?

Greg Michaelson: 19:46 Yeah, it's not just a bottleneck, it's also kind of a problematic dependency because at the end of the day, the software developers that are deploying these things probably aren't data scientists. So it's not obvious that they are going to understand what is supposed to be done. And there's a lot of subtlety to this sort of thing. So you can get mistakes introduced in really easily here as well. So yeah, so if you think about the deployment process, there's a lot of hurdles to overcome. If you've ever been slacked or emailed a Jupyter Notebook and tried to run it, you know what some of them are, right? You have the wrong version of this package installed. Oh, you got to PIP install a whole bunch of other stuff to make that work. And so you might spend an hour trying to even get the code to run, assuming that you have the data and that all the folders and file paths are the same and all that sort of stuff.

20:47 So at the end of the day, what data scientists spend most of their time doing today is building prototypes. And then those prototypes get handed off to another team to recode in another environment with [inaudible] and deployed and managing servers and stuff like that. But it's not obvious to me that data scientists know how to do that and it's really not obvious that they have the privileges to do those kinds of things in terms of just the infrastructure and all that kind of stuff. So Zerve kind of handles all of

those problems. So every canvas inside Zerve has a Docker container that's supporting it. So anybody that logs into that canvas doesn't have to worry about dependencies because it's all saved in that project. And so those environments are reusable and shareable and so on.

21:36 So if I wanted to start a new project using the same docker container that another project was in, it's really easy to do that. And so when you have a new data scientist join your team, they don't have to spend their first week getting Python installed and making sure everything, oh, we use NumPy 0.19 and you've got 0.23 installed and none of those conversations have to really happen anymore as we manage all of that. And then let's say that I did train like a random forest. I mean, you mentioned using your weights, like if I train a linear model or a logistic regression or something, then maybe it's just a vector of weights that need to be handed off.

22:15 But if it's a more complicated model like a random forest or an XG boost or a neural network or something like that, it's not as simple as just like here's some weights to put into a formula. It's a more complex thing. And so then you've got to figure out, okay, I'm going to serialize this model, pickle it and then dump all the dependencies out and dockerize it and then hand that thing off, that's also beyond the skillset of a lot of data scientists too. So Zerve handles all of that. So every block inside of Zerve, when you execute it, it creates serialized versions of all of the variables that you've worked through. So if I train a random forest in a model or in a block, then it's there and it's accessible. So I can access it from external to Zerve using an API, I can reference it in other layers.

23:04 So when it comes time to say make an API, maybe I want to make a post route where I send in a payload of predictor columns and then I want a prediction back from

that random forest, well, then I just say, "Hey, remember that random forest?" And I just point at it instead of having to figure out how to package that thing up so that it could be deployed as an API. So we handle all of that stuff and then when you deploy in Zerve, you also don't have to worry about the infrastructure stuff because all of our APIs utilize Lambdas. Like serverless technology, again, so you don't have long-running services that are out there, it's just there.

23:44 So a lot of the infrastructure stuff and the DevOps stuff and the kind of picky engineering stuff that can trip you up is stuff that we've just sort of handled so that it's easy for the user. And that means that data scientists can start to deploy their own stuff, but in some organizations they may still might not be allowed. So then we have like a handoff system where it's really easy to take something that a data scientist has done who by the way aren't building prototypes anymore. Now they're building software that can actually be deployed in Zerve and we can hand that off to other teams to actually do the deployments.

Jon Krohn: 24:22 My next clip is from Episode 875 with Kai Beckmann. Kai is an expert on something I hadn't previously heard of called "heterogenous integration", and I wanted to know: How does this capability power up AI chips?

24:37 So when we were doing research about you, we uncovered something called heterogeneous integration in AI chips. So what is heterogeneous integration and how does it impact performance and the packaging density of AI chips, this density thing being critical to building more and more powerful chips? Because obviously the more transistors you can get in a smaller space, the more powerful a chip can be.

Kai Beckmann: 24:58 Yeah, that's an important area and I call it earlier in our conversation. So this is like what is more than Moore, so what dimension drives a performance or allows to scale performance beyond just making smaller transistors on a chip? This is the additional dimension driven by heterogeneous integration. Maybe let me just quickly with a sentence come back to the AI for AI. We have branded it the way that we call that materials intelligence. This is the use of artificial intelligence to drive the development of novel materials for applications in electronics.

25:46 We call that materials intelligence and this is with how our team works as a global R&D team, not just in a traditional way, sequentially improving properties of materials by using AI to replace experiments in order to avoid unnecessary experiments and going straight into where it really matters. Where can you really make a difference for the customer technology? How can you anticipate how material works in a customer setup and how does it drive the solution of their problems and not just chemicals properties in the first glance? So this is how we drive the development of novel materials. We talk about millions of different options that need to be optimized in order to drive the performance of material. So this just to give you the idea on how that blends into AI for AI.

26:45 Second is then driving the different aspects of how our customers improve the performance of their devices. Besides shrinking the transistor, building more integrated systems and heterogeneous integration is the important area here. It started traditionally with what is called a front end process making a transistor and back end was the new wire. It somehow that at the end, the signal gets to the outside, which is then called in a product scheme packaging. Now there's something between these two extremes that it's called heterogeneous integration, when at the end the chip is not just one die, one single chip

anymore, when you combine different chips to a system. I refer to it in this specific example as CoWos, these structures being built in the examples I've used. I can use different customer examples here as well. Just wanted to use one nomenclature, which is pretty common in a current conversation.

27:53 This is when you glue dies on top of one another in order to build memory stacks, for example, or you build a memory stack and you almost glue it next to a GPU in order to shorten the transfer of data and to make it more efficient in getting the data to the GPU. That is called heterogeneous integration to make that possible. It requires, of course, technologies well advanced from what was used in packaging historically, so much smaller structure sizes, much more complicated efforts to get your heat out of the system as one example or to optimize power consumption. The precision required then needs different technologies more front-end like technologies, which makes it an area, of course, for materials innovations and for metrology innovation as per what our company is focused on.

Jon Krohn: 28:51 Widening the capabilities of real-world applications is a must for new AI-product developers, and AI Product Manager Shirish Gupta has come up with the easy-to-remember mnemonic ‘AIPC’ to help you determine whether your particular application might be ideally suited to local inference with an AI PC as opposed to relying on cloud compute. Here’s my final “In Case You Missed It” clip, coming from Episode 877:

29:23 So we're talking about taking capabilities that today might require you to have an internet connection and depend upon some cloud service in order to get some like say large language model or other foundation model capability. But instead with an NPU, you could potentially have the operations, the inference time calls. Instead of

going out over the internet and using cloud compute. You can have it running locally on device, so you're also probably going to get lower latency, you have fewer dependencies. Yeah. Talk us through some of the other advantages of being able to now do things on edge instead of having cloud reliance.

Shirish Gupta: 30:06 Yeah. I think this is a perfect segue. In fact, this is a mnemonic that I came up with myself. The term that is being thrown around for these devices with NPUs is an AI PC. Sure you've heard of it, So to think about the benefits of an AI PC, I've created a mnemonic with those four letters. So A is accelerated. It's basically you have now a local hardware accelerator that gives you that low latency real-time performance for things like translation, transcription, captioning, and other use cases where latency is super important for persistent workloads. So that's A. I is individualized. Again, this is great because if you have an AI that is on your box, it has the ability to learn your styles. Let's say if you're creating emails, if you're using it to generate emails. It's learning your style. It starts writing in your style.

31:04 It's great for ... We had a healthcare customer that we've been working with on a use case. There's two parts to it. I'll talk about the second part. The first part is even more interesting, but I think it's related to a different example that we'll come back to later. But the second part of the AI solution is that they were taking information from a physician's diagnosis of a patient in the ER and they were using that information to auto-generate the physician's report. Mundane stuff. Physicians don't like spending time on that. They'd rather go to the next patient, have that interaction, increase their ability to spend time with patients. But the feedback they gave was with this solution, now that it started seeing the way that I'm changing and editing its initial draft, it's starting to take on my style, and now it just sounds like me. And I love it

because I don't have to do this report generation, it does it for me and I've got more time for my patients. That's the individualized value.

32:17 The third is P, it's private. Like you said, the data doesn't have to leave your device and its immediate ecosystem. You don't have to send it back and forth to a public tenant or even a private tenant for that matter. You may have confidential information with PII that you have access to, but you don't want to merge with even a private tenant. There is sensitive information like that or unclassified information depending on your vantage point. So that inherent privacy of data and the inherent security of running the model locally on your device gives you that assurance that this is more private than it would be. That's P.

33:02 And C ... This is really important because I hear this from customers. It is an important cost paradigm shift. And I'm starting to hear this from some of our maybe earliest adopters of on-device AI. Which by the way is not ubiquitous today. In terms of enterprises building out their own AI capabilities and using on-device accelerators for offloading them, we're at the tip of the spear with Dell Pro AI studio, and we'll come back to that later. But the early adopters, what they say ... And I had a finserv or financial services customer tell me, Shirish, my developers are using CodeGen and they're using our data center compute. 15% of my data center compute is going to these developers that are using it for CodeGen or code completion or writing test cases, unit tests, what have you. They all have PCs. I want to get them to an AI PC with a performant NPU so I can offload that compute from my data center because they don't need H100s hundreds to do that code completion. I think I can do that with the NPUs on your Dell devices.

	34:24	So that's a real opportunity as well, is just because you have the compute doesn't mean you should use it. It's like the right compute or the right engine for the right workload at the right time. So there's plenty of use cases where offloading from even your private data center to a on-device capability makes a ton of sense. And then if you're actually using the cloud, you're paying for every inference. It's tokens and an API access. So now that you've got an AI PC, it's no cost to you. You built your solution, you're using it on the device. That's it. So cost is a big factor. Now, you'll argue that the cost of inferencing in the cloud is coming down. It's scaling very fast. But again, I get back to the point that it's the right engine for the right use case at the right time. Just because you have it doesn't mean you should.
Jon Krohn:	35:24	All right, that's it for today's ICYMI episode. To be sure not to miss any of our exciting upcoming episodes, subscribe to this podcast if you haven't already but most importantly, I hope you'll just keep on listening! Until next time, keep on rockin' it out there and I'm looking forward to enjoying another round of th SuperDataScience podcast with you very soon.