SUPER DATASCIENCE
MAKING THE COMPLEX SIMPLE

SDS PODCAST EPISODE 881: BEYOND GPUS: THE POWER OF CUSTOM AI ACCELERATORS, WITH EMILY WEBBER

Jon Krohn:	00:00:00	This is episode number 881 with Emily Webber, principal solutions architect at AWS. Today's episode is brought to you by ODSC, the Open Data Science Conference.
	00:00:17	Welcome to the SuperDataScience Podcast, the most listened to podcast in the data science industry. Each week, we bring you fun and inspiring people and ideas, exploring the cutting edge of machine learning, AI, and related technologies that are transforming our world for the better. I'm your host, Jon Krohn. Thanks for joining me today. And now, let's make the complex simple.
	00:00:42	Welcome back to the SuperDataScience Podcast. Today I'm delighted to have the amusing, brilliant and zen Emily Webber as our guest on the show. Emily is a principal solutions architect in the elite Annapurna Labs machine learning service team that's part of Amazon Web Services. She works directly on the Trainium and Inferentia hardware accelerators for, respectively, training and making inferences with AI models. She also works on the NKI, or NKI, Neuron Kernel Interface that acts as a bare metal language and compiler for programming AWS instances that use Trainium and Inferentia chips.
	00:01:29	She wrote a book on pre-training foundation models. She spent six years developing distributed systems for customers on Amazon's cloud-based machine learning platform, Sagemaker, and she leads the neuron data science community, as well as technical aspects for the Build on Trainium program, a $110 million credit investment program for academic researchers. Today's episode is on the technical side and will appeal most to anyone who's keen to understand the relationship between today's gigantic AI models and the hardware that they run on.
	00:02:01	In today's episode, Emily details the little known story of how Annapurna Labs revolutionized cloud computing,

what it takes to design hardware that can efficiently train and deploy models with billions of parameters, how Trainium 2 became the most powerful AI chip on AWS, why AWS is investing $110 million worth of compute credits in academic AI research, and how meditation and Buddhist practice can enhance your focus and problem solving abilities in tech. All right, you ready for this fabulous episode? Let's go.

00:02:30 Emily, welcome to the SuperDataScience Podcast. I'm so excited to have you on the show. Where are you calling in from today?

Emily Webber: 00:02:43 Hi, Jon. Excited to be here. I'm calling in from Washington, DC.

Jon Krohn: 00:02:47 Nice. It's interesting times in that part of the world. Lots of things happening, but we're not here. We've never been a political show. We won't get into it. At the time of recording, I'm excited. I'm looking forward to being at the Data and AI Summit in Richmond, Virginia, which is not crazy far from DC, or at least Virginia isn't. That part of the world, Virginia, near DC, I've always really enjoyed everything about it, except the traffic.

Emily Webber: 00:03:23 Yeah, no, the traffic is tough actually, this time of year it's lovely, because the cherry blossoms are just beginning to bloom. So peak season for cherry blossoms is coming up at the end of March. But one of the primary reasons I'm in DC, is because it's the HQ2 area for Amazon, so it's our second headquarters. You remember a number of years ago we did this sort of HQ2 search and Crystal City, Virginia was awarded HQ2, and so I moved out here a number of years ago to be a part of all of the activities and everything that's going on there.

Jon Krohn: 00:04:02 That's very cool. Now, it reminds me though, wasn't it initially supposed to be Manhattan? It was supposed to be New York City and then there was an uprising against it, and so they had to pick somewhere else. Because it was like there was this concern, I can't remember exactly, but that it would change too much too fast, this huge influx of people into an already busy place or something like that.

Emily Webber: 00:04:25 Yeah, no, there were a lot of great cities, obviously a lot of great choices. I think the original spec was spread across three cities, I think when they first announced that it was like New York, DC and then I want to say somewhere in Tennessee, if I'm not mistaken. And then, that sort of boiled down into definitely the DC area and some other places as well. But yeah, primarily DC.

Jon Krohn: 00:04:55 Nice. Well, I'm glad it's working out there. It sounds like a great environment to work in. Certainly AWS is doing a lot of exciting things. I thought that we might start, we almost never start with going with somebody's career path, but in your case, we're going to do that, because you have a unique career trajectory that I think provides some good context for the rest of the episode. So you started with a degree in international finance and now you've been a hands-on practitioner at Amazon for some time working on AI and machine learning. So, tell us about that transition to what you're doing today, your draw to AIML.

Emily Webber: 00:05:32 Yeah, totally. So I would say I got into computer science a little bit later. Definitely. I lived in Arizona actually, is where I got that degree, from a school called Prescott College. And I studied definitely finance. I was actually interested in Buddhism as well. So I lived at a retreat center for many years and studied meditation, and all sorts of things.

Jon Krohn: 00:06:03 You do seem super zen, super empathetic. Our listeners wouldn't know this, but we were talking for a while before

		starting recording and I was like, "Wow, Emily is just such an engaging, yeah, empathetic person." And I don't know, so all that time in the monastery I think paid off.
Emily Webber:	00:06:20	Yes. I find myself coming back to this grounding many times actually, because when we're in computer science, when we're trying to solve an algorithmic problem, trying to solve a compute problem, a development problem, many times what we really need is focus, actually. We need the ability to just bring our mind back to what the goal is, what the details are, what the challenge is, and not be overwhelmed by getting too fixated on something or being afraid of something. And just developing this sort of mental ability to calmly abide and calmly focus has honestly been really helpful in my computer science degree.
	00:07:10	So I studied at the University of Chicago after that and did a joint degree that was a master's of public policy with computational analysis actually. So studying public policy projects through the lens of computer science, and that was where I developed a love of data science. I interned at what's called the Data Science for Summer Good Social Fellowship, where we analyzed public policy problems and worked with organizations who were nonprofits or NGOs, analyzed their data science and then delivered projects to them. And so, that's sort of where I got very interested obviously in technology, technology development and trying to make a positive impact in the world. And that has led me to AWS.
Jon Krohn:	00:08:04	Very nice. Yeah, you've worked extensively with Sagemaker, which a lot of our data science listeners would be familiar with. Maybe you can give, because you do an even better job than I could at explaining what Sagemaker is. So you can let us know both Sagemaker and other AWS AI services that you've worked with, but now you're working on the Trainium and Inferentia team,

so it's hardware, compute hardware that you would use instead of a GPU, you'd use a Trainium or Inferentia chip to be doing a lot of the heavy lifting in training in the case of Trainium, or at inference time with the Inferentia chip. Yeah. So fill us in on Sagemaker, other AWS AI services that you've worked on in the past, and why hardware, Trainium, Inferentia took your fancy recently.

Emily Webber: 00:08:49 Yeah, absolutely. So I joined Amazon actually as one of our first Sagemaker solution architect SAs. So I got to work with some of our earliest customers in the Sagemaker days and figure out-

Jon Krohn: 00:09:03 What's an SA?

Emily Webber: 00:09:05 Cool. So what is an SA, a solutions architect at AWS, fundamentally, we work with customers. So that means sort of your fingers are like on the heartbeat, they're on the pulse of the business or on the pulse of the service, because you're explaining what the service does to customers every day. You're in the weeds with developers, with data scientists, leadership on both the customer and the service team about what feature A is doing, how well it's doing, and what it needs to do in the future. So I love being a solutions architect. I've always profoundly enjoyed this as a role, because you have visibility into the whole picture, you get to be a part of the whole lifecycle. And so, I was one of our first solution architects for Sagemaker.

00:10:01 So Sagemaker is a managed ML infrastructure at AWS. Essentially you can use Sagemaker to spin up a notebook server, use Sagemaker to spin up what we call training jobs, which is where you're training your model in the context of a job. Use Sagemaker to spin up ML hosting infrastructure. We have prepackaged models that are available in Sagemaker that you can pull down for training and hosting. And we have a really cool development environment. So Sagemaker Studio and the unified studio that lets a data scientist. Actually what it does is it decouples the UI that's hosting your development environment from all of the compute that's running your notebook and running your analytics job. And we package it up really, really nicely. So Sagemaker Studio is a great data science workbench, for example, where an enterprise data science team can just get onboarded, have all the tools that they need to go analyze some data and train some models.

Jon Krohn: 00:11:08 Very nice. Yeah. And yeah, so then what was the transition like? Why did you go from software to hardware in the AI space?

Emily Webber: 00:11:18 Yeah, absolutely. So through many years on Sagemaker, like many people, I saw how important foundation models were. It was obvious that customers were increasingly going to foundation models for their ability to unlock a variety of use cases, but also the size of the models just kept getting larger and larger, and they were just consuming so many resources. And so, I set up many of our distributed training capabilities. So we were running distributed training workshops with customers, we were doing accelerator health checks, we were developing managed clusters, and that led to a service called Sagemaker HyperPod, which is a fully managed parallel environment to establish clusters essentially.

00:12:14 So when you want to train and host large language models and large foundation models on AWS, Sagemaker HyperPod is a really easy way to have a managed slurm environment that you can hop into and take advantage of optimized libraries, and have a variety of health checks and cluster management tools already available for you without needing to develop that. And through this journey, I became convinced that obviously foundation models were the future of AI, but I also saw increasingly how infrastructure was just the make or break really everything came down to, from a customer perspective, how many accelerators can I get? What is the size of those accelerators? How healthy are they and how efficiently can I train and host my models on top of that?

00:13:11 Once I realized that that was the game, that that was the primary focus for customers, I just wanted to dive in and figure out what does it take to actually develop a new accelerator? How do you develop a software stack on top of that, and then how do you expose that through the rest of the cloud? So fundamentally, I love the business opportunity. It's just really exciting to think about, obviously developing new accelerators and bringing those to customers, but also the technical problems are just so interesting. It is absolutely a joy to sit down and think about, okay, how do I write a kernel for this algorithm? How do we design communication collectives for this whole host of workloads? Like, reinventing many of the foundations of the ML technology stack as a whole on the cloud is just the absolute biggest draw in my mind.

Jon Krohn: 00:14:09 Excited to announce, my friends, that the 10th annual ODSC East, the one conference you don't want to miss in 2025, is returning to Boston from May 13-15! And I'll be there leading a four-hour, hands-on workshop on designing and deploying AI Agents in Python. ODSC East is three days packed with hands-on sessions and deep dives into cutting-edge AI topics, all taught by world-class AI experts. Plus, there are many great networking opportunities. ODSC East is seriously my favorite conference in the world. No matter your skill level, ODSC East will help you gain the AI expertise to take your career to the next level. Don't miss - Online Special Discount ends soon! Learn more at odsc.com/boston.

00:14:56 Wow. Yeah, your genuine excitement for it really shines through. Absolutely. So you've said the word accelerate a few times. I just want to disambiguate. So, earlier when I said you would use a Trainium or Inferentia chip in lieu of a GPU, that term accelerator would apply to, it's the broader umbrella that includes Trainium, Inferentia and GPUs. These are all different kinds of hardware accelerators specialized, in the case of Trainium and Inferentia, specifically for neural networks, for deep learning models like the large language models that have taken the world by storm, and that got you, the foundation models that got you excited about moving into this space, and it's hardware driven.

00:15:41 It's such an interesting phenomenon to watch from a distance, where the scientific advances, the kind of new ideas in terms of how should we model. They're not necessarily super fast moving. Like, the transformer idea many years later is still the dominant paradigm and at some point that may be replaced, and that builds upon deep learning, which seems like an even more entrenched paradigm that will be difficult to shake. Maybe a nice thing when you're designing accelerators, because it means you have some kind of like linear algebra, some kinds of matrix multiplication operations that you can be like, "We're probably still going to be doing that in five years."

00:16:21 But yeah, it sounds like a really exciting space to be working in. There's a term that you mentioned as you were describing what excites you most about your work, that I've got to admit, I don't understand very well, and so I bet a lot of our audience doesn't as well, which is this idea of a kernel. So when you talk about an algorithm kernel, what does that mean?

Emily Webber: 00:16:42 Absolutely. So fundamentally, a kernel is a function that's defined by the user. And when you're thinking about programming up at the Python level, we don't really think about that way. Everything we define is user defined. So we're like, "What gives? Everything I write is a user defined function."

00:17:05 This thinking breaks down the further down the compute stack you go. So if you want to run a program on Trainium and Inferentia, for example, the way that happens is you write your program in Python, you write your program in PyTorch, and then you're going to compile that through something that's called PyTorch XLA. So accelerated linear algebra. What PyTorch XLA is going to do, it's going to take the model that you defined and it's going to represent that as a graph essentially. So the structure of your model is represented as a graph. We call that graph an HLO, high level operations. So you get this HLO graph and then essentially we do a handshake between that HLO graph that's generated from XLA, and we feed that into the top of our compiler. And so, we maintain a compiler that takes the graph that you produced from PyTorch and from PyTorch XLA, and we convert that through a variety of algorithms and processes to ultimately generate the instruction set that actually gets executed on the hardware directly.

00:18:29 So what's a kernel? A kernel is where you override the compiler and you get to define the operations on the chip yourself using our kernel library. And our kernel library is called NKI, the neuron kernel interface. So fundamentally, a kernel is a function that's defined by the user and not as generated through the compiler. Now, there's a huge variety of sizes of kernels. So you can have a kernel that's really just a hello world function, where it's like, "Hey, I did a matmul," or, "I did like tensor add," and that's to get the software working and make sure you have the environment ready.

00:19:17 But then, what most people will do is build on top of that to define a full operation. So you'll define a full forward pass for your model or a full backward pass, or even just a part of it, like maybe just the MLP up projection or down projection. And then, what you're doing is you're studying the compute optimization of that kernel. You want to look at the data movement, you want to look at the utilization, you want to look at your memory utilization, your compute utilization. And so, I know you've had Ron on the session in the past, and so everything Ron teaches us and teaches the world about compute optimization. We try to apply that when we're developing our kernels. So we study the compute requirements of a workload and we try to improve it. That is the heart of writing a kernel and implementing this algorithm that you have that's trying to improve something for large language models. We implement that as a kernel in order to improve the performance for it.

Jon Krohn:	00:20:24	An excellent explanation. You are an outstanding teacher and we will actually get to later in this episode some of the educational stuff that you've been doing. You're kind of an inspiration there, but you're naturally such an amazing explainer. That was like, 99th percentile of explanations of technical concepts that I've ever heard. So thank you for that introduction to kernels. And if people are interested in that Ron Diamant episode, it's number 691 of this podcast, also an amazing explainer of technical concepts if you want to understand a lot. In that episode, we talked a ton about designing accelerators and I learned so much in that episode. It was amazing. In fact, Ron is such a luminary in this space that at NeurIPS, neural information processing systems, arguably the most prestigious academic AI conference in the world. I was there in Vancouver in December and I met somebody new at like a lunch or a dinner, or something. I can't remember exactly the context, but they worked on Trainium and Inferentia chips, and I'd said, "Oh, we had someone from the show."
	00:21:28	He was like, "Was it Ron?" So yes.

Emily Webber:	00:21:34	Absolutely.
Jon Krohn:	00:21:35	He's an iconic person in this space. Yeah, so fantastic. There was a lot in there that some people might want to go back over to learn again about kernels. There was one term in there that I might define quickly for the audience. So you said MLP kind of casually in there is one of the things that you could be implementing in a kernel. And so, multilayer perceptron I'm guessing is what that is there. And so, kind of like one of the fundamental building blocks, it's like when you're thinking about building a deep learning network, a multilayer perceptron is kind of like it was an early deep learning network, but then you can also think about it now as something that you can scale up into a bigger architecture.
Emily Webber:	00:22:23	Yes, and it's really interesting to think about how we represent data for kernels actually. So the MLP itself, when you're designing say like a baby MLP or a tiny MLP, in PyTorch, it's crazy easy to do that. It's so easy to just define your tensor, define the operations you want to do and call it. From the PyTorch perspective, that's it. Your job is done, you've created an MLP. But it becomes so interesting when you think about the size of that, when you want to scale it up, when you want to shrink it, but also when you want to actually process it, when you want to run the computations on that to execute the operations you've defined. And it quickly becomes very challenging to do, actually.
	00:23:20	So when we're defining our kernels and when we're defining our programs in Trainium, part of what we want to do is think about how we're representing the data, how we're structuring the data from the PyTorch perspective. And then, the trick, the game is to try to optimize the data representation and optimize the program for the hardware. Actually, what we want to try and do is pick designs within the data structure and within the

algorithm that leverage some of the lower-level capabilities of the hardware to ultimately get the best utilization and the best performance that we can. Once you have sort of hardware and software programs that are well-synced and running together, and using the same assumptions, that's when you can really scale and get excellent utilization, and then excellent price performance. And that's really where we want to help customers go.

Jon Krohn: 00:24:23 Speaking of the connection between very popular deep learning libraries like PyTorch and the interaction of those libraries with your hardware, with Trainium, Inferentia accelerators, there's something called the AWS Neuron SDK, software development kit, which is the SDK, the software development kit for these AI chips, for Trainium and Inferentia. Can you tell us how AWS Neuron enables builders to use the frameworks of their choice, like PyTorch or JAX without having to worry about the underlying chip architecture?

Emily Webber: 00:24:57 Yeah, absolutely. So the Neuron SDK is a term that we use to cover a very, very large variety of tools. And the tools essentially are capabilities that we offer to developers to easily take advantage of Trainium and Inferentia. Some of the tools are really low-level things, like the runtime, the driver, the compiler that pulls everything together. Some of them are much higher-level, so something like Torch Neuron X, TNX, or essentially Neuron X Distributed, NXD.

00:25:31 So, NXD is really the primary modeling library that's really useful for customers, where when you want to go train a model and you want to go host a model on Trainium and Inferentia, NXD packages up many of the lower-level complexities and it makes it easily available for customers to access. Compiling your model, for example, is handled by NXD. Sharding your model

		actually, so taking a model checkpoint, say like a Llama or a Pixar model, and then sharding that across the accelerators that are available on your instance.
	00:26:11	NXD actually handles the model sharding for you both from a data perspective, taking the checkpoint itself and splitting the checkpoint into number of shards, but then also the communication and the optimizer updates, and the forward pass. NXD is a very, very comprehensive modeling library. And so, NXD is useful for of course implementing your own model, but also just pulling down a model and running it. So when you want to just get something that's prepackaged and test it for say something like alignment or supervised fine-tuning, or hosting, you can pull down the model packages that are pre-built and pre-set with NXD, and just run them with your experiments and with your changes. And there should be very little complexity that's exposed to the customer in those cases.
Jon Krohn:	00:27:10	Very cool. We'll have a link to the Neuron SDK in the show notes so people can check that out more. But as usual, another great example of your ability to explain technical things really well.
Emily Webber:	00:27:23	Thank you.
Jon Krohn:	00:27:24	So with your experience previously on the Sagemaker side, which we talked about earlier, does training in Inferentia work with Sagemaker as well? Are there some kinds of, just as the SDK allows you to take your framework of choice, is it easy to have Sagemaker blend on the hardware side with Trainium and Inferentia?
Emily Webber:	00:27:45	Yes, absolutely. I mean, you can run Sagemaker Notebook instances, you can run Sagemaker Studio on Trainium. If you want to say develop a new kernel or test NXD, you can do that very easily on Sagemaker as a development

environment. We also have many models that we've already supported on NXD that we'll make available through what's called Sagemaker Jumpstart, where Sagemaker Jumpstart is sort of a marketplace for machine learning models and LLMs that are prepackaged, and available. And when Sagemaker customers are, say, browsing in Sagemaker Studio, they can click a button to download the model, but they're not actually downloading the model. What's happening is they're accessing the model through the marketplace, the training and hosting infrastructure. A lot of the software is fully managed by Sagemaker, and then customers can bring their own data sets, they can fine-tune the model, they can host the model all through Sagemaker Jumpstart, and that absolutely is well integrated with Trainium and Inferentia.

Jon Krohn: 00:28:56 Hey hey! This is your host, Jon Krohn. I'm excited to announce that this northern-hemisphere spring, I'm launching my own data science consultancy, a firm called "Y Carrot". (If you're an ML practitioner who's familiar with "y hat", you may get our name!). Regardless of who you are, if you're looking for a team that combines decades of commercial experience in software development and machine learning with internationally-recognized expertise in all the cutting-edge approaches, including GenAI, multi-agent systems, and RAG, you've found us! We have rich experience across the entire project lifecycle, from problem scoping and proof-of-concept through to high-volume production deployments. If you'd like to be one of our first clients, head to ycarrot.com and click on "Partner with us" to tell us how we can help! Again, that's Y Carrot, y-c-a-r-r-o-t-dot-com.

00:29:46 Very nice. And yeah, you've been working closely with customers on adopting Trainium, so it's something that's picking up a lot of speed. Probably because Ron Diamand was on this show a couple of years ago.

Emily Webber:	00:29:58	No doubt, no doubt.
Jon Krohn:	00:30:01	And in fact, huge companies like Apple. Apple joined your Reinvent CEO keynote last year to talk about their use of Inferentia and another chip called Graviton, which you'll need to explain to us in a minute, because we haven't talked about that on air ever. But they talked about their use of Inferentia and Graviton, and why they're excited about Trainium 2, another thing that we haven't talked about yet in this episode. So what are some of the most interesting technical challenges that customers like Apple are trying to solve? 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:	00:31:04	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.

00:32:00 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.

00:32:57 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: 00:34:02 So that's Nitro, like nitroglycerin, N-I-T-R-O?

Emily Webber: 00:34:06 N-I-T-R-O, yeah.

Jon Krohn: 00:34:08 Explosive.

Emily Webber: 00:34:09 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:	00:34:44	Ooh.
Emily Webber:	00:34:44	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?
	00:35:49	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:	00:37:05	Trainium 2 is the most powerful on AWS?
Emily Webber:	00:37:08	Correct.
Jon Krohn:	00:37:08	Wow, that's super cool. And so, what are the key differences between the first generation Trainium chip and Trainium 2? This is all stuff that's new since Ron's episode on the show, since episode 691 a couple of years ago. And so, is it like one or more kind of big conceptual changes that lead to this leap from Trainium one to two, or is it kind of a bunch of incremental changes that together combine to have all of this power in Trainium 2 and such cost-effectiveness?
Emily Webber:	00:37:44	Yeah, sure. So we try to keep it easy for you. And so, the way we keep it easy for you is that the core compute engine design isn't that different, actually. The neuron core itself, particularly between Trn1 and Trn2 is pretty much the same. So what's nice about that is it means the kernels that you write for Trn1 to Trn2, and the development modeling code would say like NXD is really, really easy to just move up from Trn1 to Trn2. The big difference-
Jon Krohn:	00:38:20	Can I interrupt you for one quick second? It sounds like, so are you saying Trn1 and Trn2, is that like an abbreviation of Trainium?
Emily Webber:	00:38:27	Yes.
Jon Krohn:	00:38:28	Okay, okay. Okay, gotcha.
Emily Webber:	00:38:29	That's also the name of, yeah, abbreviation of the name, and it's the instance name directly.

Jon Krohn:	00:38:36	Right, right, right, right. So yeah. Gotcha, gotcha, gotcha.
Emily Webber:	00:38:39	Yeah, Trn1 and Trn2.
Jon Krohn:	00:38:41	My apologies for interrupting. Carry on.
Emily Webber:	00:38:42	No worries. Yeah, so the key differences between Trn1 and Trn2, is that on Trn2 you have four X the compute.
Jon Krohn:	00:38:52	Yeah, it's a big number.
Emily Webber:	00:38:53	That is a big number. Now, the reason why that happens is because you have four times more neuron cores per card. So in Trn1, you have two neuron cores that are packaged up together in a single card, and then you have two HVM banks, and that is the accelerator, the combination of those. Trn2, you have eight neuron cores. So just multiplied by four, you have eight neuron cores, you have four HVM banks, each card itself is 96 gigs of HVM capacity. On the instance as a whole, you have 16 of those cards. So at the instance as a whole, you have 1.5 terabytes of HVM capacity. And then, we gave you an ultra server. So, the ultra server is where you take four Trn2 instances, and then these are all combined in one giant server, actually.
	00:40:05	The reason why we say that, so it's two racks, four servers, and then 64 cards that are all connected by Neuron link, which is our chip to chip interconnect, such that there is a minimum of two hops from one card to any other card. When you do like a neuron top or a neuron LS on a single Trn1 instance, it's going to show you 128 trainable accelerators, because you have 128 neuron cores on your single instance. We actually have a way of grouping those. You can group them by what we call like a logical neuron core, which is kind of cool, because then you can change the size of the accelerator that you want based on the workload, which I think is very fun.

	00:40:58	And then, yeah, those are all packaged up into this giant ultra server. If you watched Reinvent, actually Peter Desantis wheeled out an ultra server on stage and spent much of his keynote just talking about it. It's such an awesome moment. But so, ultra servers are unambiguously the best way to train and host the largest language models on AWS. And you have the most powerful instances combined in a really compelling, innovative way that connects all of the cores and makes them very easy to train, and then to host while minimizing the number of hops that you need to do between hosts, because they're all logically one server. So ultra server is pretty cool.
Jon Krohn:	00:41:48	Ultra server does sound pretty cool. I might be putting you on the spot with this question, but how many model parameters of a large language model say can you fit on an ultra server?
Emily Webber:	00:41:59	Yeah, so it's kind of a weird question to answer to be honest, because you can fit a lot. But no, but what I mean is that realistically, you don't actually want to max out the memory, like realistically, you want to give yourself space, like for your batch size, for the optimizer, for the adapters. If you're training it, you're going to want to have multiple copies of it for something that's really large. If you're hosting it, you're also going to want multiple copies of it, because you're responding to many different users at a point in time. So it's actually a pretty complex question to answer, and it's highly use case dependent.
	00:42:43	The rule of thumb we use though is like, and it's not what will fit, but again, it's what is actually good for a normal use case. So for a normal use case, language models that are in the 70 billion parameter range, we recommend those for Trn1. Trn1 is a good candidate for language models that aren't gigantic, but that are still sizable, and

Trn1 gives you competitive and powerful compute for training and hosting those models. Language models that are significantly larger than that go to Trn2. By all means, go play with an ultra server and get all those neuron cores, and see what you can do with it.

00:43:29 And again, what's nice about it, what I love about the stack is that NXD gives you both the connection into the compiler. So when you implement your modeling code in NXD, by default, you get a nice sync with the neuron compiler and all of the lower level XLA benefits, but we also shard the model for you. So when you want to play with different TP degrees, like say you want to try a TP degree of eight on Trn1, but then on Trn2, you want to try TP 32 and TP 64, and TP 128, because why not? And sort of see what happens. NXD makes it super easy to do that, because you're just changing a parameter right at the top of the program to then shard your checkpoint itself and redefine your distribution method. And so, yeah, NXD handles all of that for you, which I just absolutely love.

Jon Krohn: 00:44:31 Do you ever feel isolated, surrounded by people who don't share your enthusiasm for data science and technology? Do you wish to connect with more like-minded individuals? Well, look no further, Super Data Science community is the perfect place to connect, interact, and exchange ideas with over 600 professionals in data science, machine learning, and AI. In addition to networking, you can get direct support for your career through the mentoring program, where experienced members help beginners navigate. Whether you're looking to learn, collaborate, or advance your career, our community is here to help you succeed. Join Kirill, Hadelin and myself, and hundreds of other members who connect daily. Start your free 14-day trial today at superdatascience.com and become a part of the community.

	00:45:16	Nice. And so, to define for our listeners that kind of that idea of TP 8, TP 32, TP 64, it's the precision of the digits at these model parameters, right?
Emily Webber:	00:45:26	It is not.
Jon Krohn:	00:45:27	Oh, it's not?
Emily Webber:	00:45:28	Yeah. No, it's not. What I meant by TP was tensor parallel degree. So how many neuron cores you'll use to host one copy of your tensor, for example. So if you are doing a TP of eight, that means you're going to consume eight neuron cores to do operation X with your tensor. If you're doing a TP of 32, that means you're going to shard your model over those 32 neuron cores. In a data type world, you would be thinking like FP 32 and BF 16, and INT 8. I know they're similar, but very different meaning.
Jon Krohn:	00:46:17	I said define for our audience, but I ended up meaning define for me. And so, tell me about this. So this TP 8, TP 32, that now you've just explained, why would I make those changes and what impact does making those changes have?
Emily Webber:	00:46:33	Yeah, sure. So it is pretty impactful. It impacts the collectives a lot actually. It impacts how much time your workload will spend in an all reduce, for example, or in a reduce scatter, or in a gather scatter. So those are these collective operations that support the program that you define and support your model, and they communicate across all of the cores and they collect information. And so, you use collectives when you're running distributed training and hosting very regularly, and it's important to understand the impact those collectives can have on your compute when you're profiling your workload and trying to improve it.

	00:47:23	And so, when you experiment with different TP degrees, it can improve performance and it can also degrade performance, because of the impact of the collectives, the impact on memory, it'll impact how large of a batch size you can hold, it'll impact your overall step time, et cetera. And so, that's why it's helpful to have this ability to easily test different TP degrees. Also on Trn2, because you have this LNC feature, logical neuron core feature that lets you actually change the size of the accelerator logically based on grouping it in sets of one, which is LNC one or grouping it in sets of two, which is LNC two.
	00:48:13	And so, what that does is it actually shrinks the total number of available accelerators to your program. So on Trn2, an LNC one shows you like 128 trainable devices or available devices to your program. But when you set LNC to two, that shrinks the number. So instead of 128, then you see 64 and it makes it slightly more available in the HPN bandwidth. The banks obviously don't stay the same. I mean, the banks stay the same physically, the hardware doesn't change at all between those two settings, but it changes how much is available per core to your program. So it's changes and modifications like this that let you find the optimal balance in your program and in your workload, while easily experimenting with them through NXD.
Jon Krohn:	00:49:13	I got you. So these configuration parameters like TP degrees, when we are dealing with a large language model, that's huge. So it's distributed across many different accelerators, many different compute nodes. These kinds of configuration parameters like TP degrees need to be configured to figure out for exactly your model in the situation you're using it. What is the optimal config?
Emily Webber:	00:49:40	Totally, how many tokens per second can you get? What's your time to first token? How can you reduce your overall

cost by having fewer resources, but still being able to respond to N number of responses at a time. So, all of these questions we need to consider when we're trying to find the right instance and trying to find the right instance settings.

Jon Krohn: 00:50:06 Very cool. All right, so now we have a great understanding of why a Trainium chip or a Trainium 2 chip might be the obvious choice for a listener when they're thinking about training a large language model or Inferentia might be when deploying a large language model. So give us some real world examples of customers that you've had that have been able to take advantage of these chips to great effect.

Emily Webber: 00:50:31 Yeah, sure. So our flagship customer example of course is Anthropic. So Anthropic has been a very active developer and customer with Trainium and Inferentia for quite some time. And so, the partnership has been phenomenal. Anthropic is a great team, it's an absolute privilege to support them as a customer. And we are developing some big projects together. So I don't know if you heard about Project Rainier, but Rainier is a absolutely gigantic cluster that we are developing in collaboration with Anthropic, with obviously state-of-the-art Trainium cards and instances. So it's just fascinating and it's just a pleasure to innovate with them. So Anthropic is a great example.

Jon Krohn: 00:51:27 Fantastic. Yeah, they certainly are one of the leaders at the forefront of AI. For me personally, you wouldn't know this, Emily, regular listeners probably would that. My go-to large language model for most everyday use cases is Claude, and it has been for some time. So, love Anthropic and I'm not surprised to hear that there's amazing, intelligent people to work with there on big mountains of a problem like Project Rainier. For our listeners around the world, Rainier Mountain is a big mountain in Washington State, not Washington, DC.

Emily Webber: 00:52:05 It is. That is correct. But yeah, no, and then so obviously we have customers across the spectrum, so Anthropic is such an important customer. We also work with startups, so we work with startups like RC or Ninja Tech, who are training and hosting small language models. And in the small language model space, it's exciting for customers, because our price performance and our overall availability is just really compelling. They love the benefits that they get. They love the price, they love the performance, they love the models, they love the software stack. So we definitely see some great movement there. We also see customers like Databricks. We are doing some big projects with Databricks.

Jon Krohn: 00:52:57 Not a small startup.

Emily Webber: 00:53:00 Not a small startup. Yeah, no, we're doing some great work with Databricks, and then now we're expanding into the academic sector with Build on Trainium.

Jon Krohn: 00:53:11 Cool. What's the Build on Trainium program?

Emily Webber: 00:53:13 Yeah, so Build on Trainium is a credit program that we are running, which is $110 million in credits that we are offering to academics who are working on the future of AI. So fundamentally, this is a way for universities, academics, PIs, principal investigators to submit their research ideas to us about their big ideas. We want to know sort of what they've already tested on Trainium, what their early modeling and early kernel results are. And then, we are working to scale those results with them on a cluster that is up to 40,000 Trn1 cards. So, we have a very significant cluster that is available for researchers for the best AI projects in the world. And so, yeah, this is a big project, of course. We've been working for it on quite some time.

Jon Krohn: 00:54:12 Sounds really cool. Indeed, we'll have a link to the Build on Trainium program in the show notes for those academic listeners out there who would like to take advantage of this $110 million program from AWS. I'd also like to highlight another client of Trainium and Inferentia chips that I'm aware of, is Poolside. And I'm aware of that because back in episode 754, we had Jason Warner, the CEO of Poolside on the show, and it's a really cool startup. It isn't Databricks size yet, but Poolside, they're trying to tackle artificial general intelligence from the perspective of software, of code generation. And there's compelling arguments that Jason makes in episode 754 about how that might be feasible. So, cool episode to highlight there and another Trainium, Inferentia customer.

Emily Webber: 00:55:06 Absolutely. Yeah, we're very excited about the Poolside partnership.

Jon Krohn: 00:55:09 Nice. All right, so when you're trying to figure out what the right instance is, so we've talked about Trn1, Trn2, Inferentia chips as well, the other kinds of instances that are available on AWS. How do you pick the right kind of instance type for a particular machine learning task?

Emily Webber: 00:55:29 Yeah, sure. So of course when you're, let's assume we're in the Trainium and Inferentia space for the moment. So when you're in that space, I mean really you have a couple of questions. Obviously, we have two product lines, Trainium and Inferentia, the neuron core itself though, the fundamental acceleration unit is the same, actually the neuron core is the same, the software stack is also the same. So you can mix and match, go back and forth, good compatibility. What's different between the two is that the instance topology is just configured differently.

00:56:06 So with Trn1, we assume that you're going to be training. So we connect the cards in what's called a Taurus topology, or a 4D Taurus topology, which means that the cards are connected to each other in a way that you can easily do a backward pass, you can easily gather the results from all of the cards, and then update the optimizer state. So the connectivity between the cards is much more suited for a complex backward pass, whereas in the Inferentia line, again the same neuron core, but the topology is more aligned for just a forward pass.

00:56:51 So when you study the architecture, you'll see that you have just one row of the cards, for example. It's not this 4D topology, it's sort of more aligned for just taking a large tensor, sharding a large tensor on the fleet and then doing a forward pass. So that's some of the difference. Another difference is that in Inferentia you have more choices, you have many different options. For instance, size between how many accelerators you want, your HBM capacity as a result of that. Whereas in Trainium, it's sort of really small and really large. So that's why we see a good benefit on training, where you're doing your small development with a single Trn1 and then you're scaling it up for one large instance, and then as many instances as you can get, and you don't really need that flexibility. Whereas in Inferentia, you might want to host your seven billion or your 11 billion parameter model that isn't going to have the same compute requirements.

Jon Krohn: 00:58:11 Nice. That was a great explanation, as they have been throughout this episode. And actually, speaking of your great explanations, you do have a history of education. I mentioned that earlier in the episode. We would talk about some of the educational stuff you did. So for example, you wrote a big 15 chapter book called

Pre-Trained Vision and Large Language Models in Python. It's got a good subtitle too, End to End Techniques for Building and Deploying Foundation Models on AWS. So short form title, Pre-trained Vision and LLMs in Python. And so, that's a big 15 chapter book. And you have also been an adjunct professor and a startup mentor. You created a course called Generative AI Foundations on AWS.

00:58:56 So we'll have that in the show notes as well. And when I put all of that into context for our listeners, it's probably totally unsurprising that you're involved in the Build on Trainium academic program that we were talking about earlier because that involves amazing research universities like UC Berkeley, Carnegie Mellon, the University of Texas at Austin and Oxford University. So very, very cool. I think this comes from our researcher, Serg Masis, who's always pulling in really interesting pieces from our guest backgrounds. I have this quote that at a Swiss machine learning conference called ALMD, you told the story of Francesco Petrarch, a poet from the Italian Renaissance and how this story relates to the AI development project. So could you elaborate on this story and how it influences your approach not only to AI development, but also your efforts in AI education?

Emily Webber: 00:59:57 Sure. Yeah. So there's a lot to unpack there. Let's try and take it step by step. So again, because I don't have an undergrad in computer science and I don't have a PhD in computer science, I didn't have that opportunity. I feel like I've had to teach myself a lot and obviously I've had phenomenal mentors and have worked on phenomenal teams that pushed me, worked with phenomenal customers who pushed me. What I love about technology, the reason why I love software so much, is because in software, if you build it, you can understand it. At least that's how I feel. It doesn't matter how complicated something is, it doesn't matter if I didn't take that class. It doesn't matter if I didn't have a PhD in whatever it is, if I can code it, I can convince myself that I can probably understand what's going on.

01:00:53 And so, from that perspective, that is the perspective by which I teach, because I understand that we live in a world where not everyone has every opportunity that maybe they wish they had, but nonetheless, here we are and we're doing our best. And so, I love teaching, because I love taking things that were hard for me to understand, that were hard for me to explain to myself. But because it was challenging, somehow I was able to find a way to simplify it to myself. Then I love sharing that with other people, because I know it simplifies their journey and it simplifies their path. Certainly simplifies their experience on AWS with my own technology stack. And so, that ethos, I guess I just love, I've always loved. So education is a part of why I enjoy that and it's a way to scale that, and help others grow. So that I just really enjoy.

01:01:53 I want to address the Petrarch point. I love that that came up. It's sometimes surprising how things you post on the internet show up later. So that's beautiful. I'm also just a humanist, like I love so many things in this world. I love art, I love art history, I love philosophy. I love thinking about things in ways that didn't previously consider to me. So the reason why I did that, I was prepping to do an invited talk, like an invited keynote at this conference in Switzerland, and this was at the time when LLMs were like just becoming to be popular and foundation models were just becoming to be popular. So I wanted like a nice quote about intelligence that would feel like culturally relevant.

01:02:46 So I thought Petrarch would be a good quote. So I got some nice phrase about human intelligence and the impact of human intelligence. And it's funny that we live in a world where we need to talk about human intelligence as an important thing that matters. Like, I don't know, I see so much going on in the LLM space and the AGI space. I'm like, don't get me wrong. Obviously I'm all about scaling out computers and like developing AI, but I also care a lot about human intelligence. I find it super valuable in my own life to maintain my own intelligence as a goal. I find that valuable in the life of my team and people that I work with. It's like we need to continue to grow our own intelligence while we're obviously growing the intelligence of the machines. But that balance between those two I really enjoy and I just find so fun to consider.

Jon Krohn: 01:03:49 Well, that idea of AI being for humans and supporting our intelligence and our actualization as individuals, and as a society. That's actually a theme of two recent episodes of this show. So, we have two episodes largely dedicated to that kind of idea, which is episodes 869 and 873, with Varun Godbole and Natalie Monbiot respectively. So yeah, there seems to be, it's something that kind of has just started to come onto my mind as well, and at the time of recording and preparing a keynote, kind of along those themes as well. So yeah, I think there's something special there. My final technical question for you before we get into our kind of wrap up questions, Emily, is just kind of your insights into what's going to happen next. Obviously it's a fast moving field, but you're right at the heartbeat of it there, working on hardware like Trainium and Inferentia. So field of AI is moving incredibly fast. What emerging technical challenges most excite you? What do you think is coming next?

Emily Webber: 01:04:57 Yeah, sure. So I think where five years ago this was a question, unambiguously large language models are here to stay. This is just clear. How these continue to be integrated into applications, the nature of them, the fine-tuning of them, the agentic systems that are built on top of them, the pre-training of them, the dataset selection for them, the evaluation of them, all of those will change. All of those are in flux. All of those will evolve. For a while, I've seen, particularly in my Sagemaker days, how over time it just makes so much sense to push knowledge into the model as much as possible. It simplifies the lift for development teams, simplifies the lift on data management, simplifies the lift on the application management.

01:05:50 So I think you'll continue to see a variety of ways that people try to push knowledge into LLMs, like push knowledge into an LLM in the pre-training stage, when you're creating the foundation model from scratch, you do it when you're doing supervised fine-tuning to teach it how to follow commands. You do it when aligning the model to perform complex reasoning. You do it when you're designing your rag system, you do it when you're designing your agentic system, but really all of those are just fluff compared to what's actually in the neural network itself. And so, what I think you'll continue to see is this synergy between people solving problems at the agentic system, at the agentic level that then are absorbed by the rag, that are absorbed at pre-training, that are absorbed by the dataset itself.

01:06:47 And then, obviously hardware is going to keep cruising. Like, we have a lot in store. Trn3 was pre-announced at Reinvent. Trn3 is on the way. We are very much just getting started in what you're going to see from Trainium and Inferentia with Nikki, with Bill on Trainium. So stay tuned. But in terms of the LLMs themselves, yeah, there's a lot that's going to continue to be the case. But it's also, you know, it's kind of encouraging that like the core problem is the same. Everyone's still trying to train their model the best they can and figure out how to host it, and figure out how to do inferencing on it the best. That has not changed. I don't expect that to change ever. But now the focus obviously is language models and doing that the most efficiently with the best results, with the best mixture of results. And so, I think there's a lot that you'll continue to see in that space.

Jon Krohn:	01:07:48	Fantastic answer. Thank you. Lots to look forward to. And of course driven by hardware, that's what's happening right now. Fantastic. Emily, this has been a sensational episode. I've learned a ton. I marked down for our show notes, maybe a record number of links that I'm going to have to, like terms, interesting terms that people will be able to dig into after the episode. So clearly a huge amount of concrete, useful content conveyed in this episode. Thank you so much. Before I let you go, I always ask my guests for a book recommendation.
Emily Webber:	01:08:25	Yes. So I got this book today actually delivered by Amazon. I don't know if this is coming through, it's probably not coming through.
Jon Krohn:	01:08:38	Most of our listeners are listeners only. We do have, there are some YouTube viewers. So Emily was holding up the book delivered by Amazon onto the video camera. What's the title of the book?
Emily Webber:	01:08:57	Yeah, so the book is called Voice for the Voiceless and it's a book by his holiness the Dalai Lama. So I mentioned to you that I love to meditate and I'm a Buddhist practitioner, so of course I love to read, just personally, I love to read the words of his holiness the Dalai Lama. So I'm very much looking forward to reviewing this book, to reading it and sympathizing with his struggles, but also with his wisdom. I find his holiness to just really do a remarkable job of combining wisdom with compassion in the modern time, while also holding onto the strength of his lineage. And so, I'm very much looking forward to reading this book and I would tentatively offer my recommendation for it on that basis.

Jon Krohn:	01:09:50	Nice recommendation. I'm sure it's an exceptional book. I have read books by his holiness the Dalai Lama in the past. I read An Open Heart some years ago, which I thought was great. It includes some kind of introductory tips on meditation. I had already been meditating for a few years at that point, but there were some great pointers in there and just some great life advice. He seems to be quite sage. He's a sage maker, if you read his books.
Emily Webber:	01:10:19	Indeed, indeed.
Jon Krohn:	01:10:23	Yeah. He makes his readers sage. So there you go. A nice AWS joke. All right, Emily, so how can we follow you after this episode?
Emily Webber:	01:10:34	Yeah, sure. So you're welcome to follow me on LinkedIn. I will warn you that I am just not active on social media.
Jon Krohn:	01:10:43	How could you be Buddhist, centered and active on social media? Those are incompatible.
Emily Webber:	01:10:50	I'm not saying they're incompatible, I'm just not that active on social media.
Jon Krohn:	01:10:52	I bet it makes it harder.
Emily Webber:	01:10:56	But yeah, you're actually the first podcast that I've ever done.
Jon Krohn:	01:10:59	No.
Emily Webber:	01:11:00	It's true. It's true. So I'm excited to burst my podcast bubble and yeah, follow me on LinkedIn, but hit me up on GitHub. I'm actually super active on GitHub.
Jon Krohn:	01:11:13	There you go.

Emily Webber:	01:11:14	I forgot to mention this on. So for Build on Trainium, we just wrapped a competition actually, so we are offering $25,000 in cash to the top team who can develop the fastest NKI Llama, the fastest Llama implementation using NKI, actually. So the competition is over, but I am totally expecting projects like this to pop up again, so definitely stay tuned for more. But yeah, I'm really active on GitHub. When you're cutting issues in the Neuron SDK or the NKI SDK, feel free to tag me. I will respond. Shoot me your kernels. I'd love to see the work that people have. And yeah, let's go build, in the words of Werner Vogels.
Jon Krohn:	01:12:08	And a reminder there, we talked about NKI earlier in the episode, but it's NKI, neuron kernel interface, and I'll have a link to that in the show notes too.
Emily Webber:	01:12:17	Beautiful. Thank you, Jon.
Jon Krohn:	01:12:19	Thank you, Emily, for taking the time. I'm so delighted to have had you on your first podcast appearance. You are a natural, and every podcast should have you on. I don't even care if they're in data science or not, to explain something wonderful about the world. Thank you, Emily.
Emily Webber:	01:12:35	Thank you, Jon.
Jon Krohn:	01:12:42	What a sensationally interesting episode with Emily Webber. In it, she covered how her background in meditation and Buddhist practice provided mental tools that helped her excel in computer science, by developing focus and calm problem-solving abilities. She talked about the Nitro system developed by Annapurna Labs that was acquired by Amazon in 2015 that physically separates data and instance control in cloud infrastructure, creating better security and scalability. She talked about how the Build on Trainium program is AWS's $110 million investment program, providing cloud

credits to academic researchers working on cutting-edge AI at institutions like Berkeley, Carnegie Mellon, UT Austin, and Oxford.

01:13:21 She talked about how Trainium 2 offers four times the compute power of Trainium one, with eight neuron cores per card instead of two and 1.5 whopping terabytes of high bandwidth memory capacity per instance. She talked about the AWS Neuron SDK that helps developers easily optimize and deploy models on Trainium and Inferentia chips through tools like NXD, which handles model sharding across accelerators. And she talked about hardware design decisions like TP, tensor parallelism degrees that significantly impact model training and inference efficiency, requiring careful optimization for specific workloads.

01:13:56 As always, you can get all the show notes, including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Emily's social media profiles, as well as my own at superdatascience.com/881. And if you'd like to engage with me in person as opposed to just through social media, I'd love to meet you in real life at the open data science conference, ODSC East, running from May 13th to 15th in Boston. I'll be hosting the keynote sessions, and along with my longtime friend and colleague, the extraordinary Ed Donner, I'll be delivering a four-hour hands-on training in Python to demonstrate how you can design, train, and deploy cutting-edge multi-agent AI systems for real-life applications.

01:14:38 Yeah, and we could also just meet for a beer or whatever there. Thanks of course, to everyone on the SuperDataScience Podcast team. Our podcast manager, Sonja Brajovic, media editor Mario Pombo, partnerships manager Natalie Ziajski, researcher Serg Masis, writer Dr. Zara Karschay, and our founder Kirill Eremenko. Thanks to all of them for producing another fabulous episode for us today. For enabling that super team to create this free podcast for you, we are deeply grateful to our sponsors. You can support the show by checking out our sponsors links, which are in the show notes. And if you yourself are interested in sponsoring an episode, you can get the details on how by heading to Jonkrohn.com/podcast.

01:15:18 All right, share this episode with people who might like to have it shared with them. Review the episode on your favorite podcasting app. I think that helps us get the word out about our show, subscribe if you're not a subscriber, feel free to edit our videos into shorts or whatever you like, just refer to us. But most importantly, just keep on tuning in. I'm so grateful to have you listening and hope I can continue to make episodes you love for years and years to come. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the SuperDataScience Podcast with you very soon.