COLEMAN
VISASQ

AI Driving Demand for GPUs

Our Expert :
Alex Ermolaev

Managing Director – TeamCamp

- Director of AI at Change Healthcare (07/2018 – 06/2020)
- Head of AI Software at NVIDIA (05/2016 - 02/2018)

Alex holds deep AI and GPU experience including: Bell Labs (Expert Systems, NLP), startups (object detection, ads/brand detection, drone navigation, self-driving), Nvidia (deep learning, AI platforms/tools, cyber, fraud, enterprise AI) and Change Healthcare (NLP, Healthcare AI, Medical Imaging).
At NVIDIA, he was responsible for GPU Deep Learning adoption by the AI community, creating $1B in revenue in 2 years. He developed brand new deep learning use cases and scenarios for data science platforms, business intelligence, malware detection, fraud, NLP, video detectors and enterprise AI.
His current company, Team Camp, functions as an AI incubation and advisory company. TeamCamp helps startups and large corporations to bring new AI technologies to market: develop new ideas, develop prototypes, refine market opportunities, achieve precise product-market fit, develop superior go-to-market, scaling, recruiting, financing, and exit.

Moderator:
Max Le Sieur

Founder & Managing Partner at Rosemont Legacy

- MBA, Harvard Business School
- Investment Banking Associate at BMO Capital Markets (07/2016 – 08/2020)

Expert Insights On:

- Overview of the GPU hardware landscape: CPUs vs GPUs, Types of GPUs, Overview of the key players, the importance of GPUs to the advent of AI.
- What are TPUs? Link between the software and a semiconductor and why the software is important
- Software landscape. CUDA, OpenCL, PyTorch, ASICs, and the barrier to entry caused by CUDA.
- Application specific GPUs. Startups and large player relationships
- End markets being impacted and the different types of AI application
- Considerations regarding the future of purpose-built GPUs
- Creation of compilers that allow a non-CUDA code to be used with CUDA
- Advantage of native compilers and outlook

Introduction -

Max: Okay, perfect. Yeah, so listen, pleasure to meet you. My name is Max and I'll be leading this call on behalf of Coleman today. Just a brief compliance reminder before we get started. The purpose of this discussion is really to learn about semiconductors and how AI is driving demand for GPUs in many different end markets. We are in no way soliciting any non-public information or any information that is confidential and related to any company or organization that you are currently or have ever been affiliated with. If any answer to a question feels like it may involve non-public information, please tell me right away and I'll take us in a different direction. Does that make sense?

Alex: Sounds great.

Overview of the GPU Hardware Landscape -

Max: Okay, awesome.
And so we're ready for the first topic. Can you start by explaining the difference between CPUs and GPUs and just maybe a very brief overview of the history of each please?

Alex: Yes. So CPU is a general purpose computing device. It's designed to process every single task imaginable. So if you're doing your email, that will be CPU, your computer communicating with the server, that's CPU. Database applications of any sort, and that's all CPU. What GPU is accelerator or co-processor. So this is a supporting device which is useful to offload certain amount of work from the CPU. So it does not replace CPU, but what does it helps with particular type of workloads.

Alex:

CPU been around for a very long time, the digital version of course, since the early 70s. GPUs emerged in early 90s when we have a significant demand for graphics becoming available or becoming popular. NVIDIA in particular was founded in 1993 and at that point in early 90s, we had more than a hundred different GPU companies. And the tasks that was uploaded to GPUs at that time was specifically graphics. Every pixel you see on the screen needs to be calculated. When pixels get calculated, they have overlapping windows, there is colors, there is shading. Anyway, there are quite a few different things that needs to be calculated, and GPU is a device was designed for that particular purpose. So that's early 90s. 10 years later in 2003 or around 2003, people both at NVIDIA as well as AMD looked at it and says, "Okay, we have this device which good for graphics." But what the device actually does is multiplication in parallel. So you have a special purpose device which does multiplication in parallel. Is it good for anything other than calculating pixels we see on the screen?

And they looked, they thought about it, say, "Okay, yes, we can identify several use cases besides graphics where GPU style device, which is multiplication and parallel, can be helpful." And I saw some papers, different people identified different opportunities obviously, but deep learning or at that time it was called neural networks. Neural networks was one of the use cases which was identified since the very beginning back in 2003. But in order to make neural networks work on GPU, you have to make a bunch of additional changes. So the libraries which are used to run in graphics still works, there is no change. But in order to make GPU available for something more complicated like neural networks or weather forecasting or molecular dynamics, what you need is you need to general purpose computing environment which allow you to program GPU to do whatever you want to do ago. And that's in 2003, NVIDIA launched general purpose GPUs or GPU used for general purpose compute with CUDA and stuff. And 10 years later, 2013, as AI become popular, NVIDIA got very excited about it and made additional bunch of work to make GPUs suitable specifically for AI compute. But the history of GPUs is every 10 years you have a major innovation which enables it to do something that it hasn't done before.

Alex: Now in all those cases, GPU is still supporting device. So the way it works is you have a normal program, you have normal computer program, you have your follow-up and exam statements and all those kind of things. And what you do, you look through your computer program and says, "Okay, look, there is a particular section in my computer program, this three lines of code which take 90% of my compute power. Can I take those three lines of code and instead of running those multiplication on CPU, can I run them on GPU instead?" In some cases as possible, if the things are easy to parallelize. In other cases it's not possible, things are not suitable for parallel applications. But if the things are suitable, what you do, you add several lines of additional code where you said move the data, move the memory, allocate memory, move the data, do the compilation on GPU instead of CPU and bring the results back.
So most of the code remains the same, but for particular cases where parallel multiplication, very helpful, you're going to move that processing to GPU and bring the results back. Why GPU is better at parallel processing? Because CPU does parallel processing as well, right? But if you think about CPU, let's say you have a powerful modern server. Let's say you have 48 cores. Let's say you can do four parallel multiplication per core, which will give you about 200 computations in parallel. 200 multiplication or 200 computations in parallel sounds like a good number. But if you look at the modern GPU, what it gives you is 20,000. It's actually 18,000 multiplication in parallel. So the GPU can give you a hundred times more multiplication in parallel compared to what CPU can do. And therefore if you can prioritize particular compute intensive workload, it makes sense to use GPU for that type of computation and leave CPU to do what it is designed to do, which is basically everything else.

Max: Are there different types of GPUs?

Alex: The GPU evolved over time and it's actually kind of interesting topic, but GPUs evolved over time. There are some GPUs for example for consumer, optimized for consumer as they run on air cooling boards. They're cheaper, if they break it's easier to replace. And you have a data center GPU which run on liquid cooling board with a lot of other hardware and software loaded with it. So it's the same chip, almost identical actual chip, but depends on the configuration and the workload, you end up with having two different use cases and two very different usage scenarios for them.

Alex: There are also evolutions within GPU itself. So what I described previously is kind of a general view of GPU, it's a device which does large number of multiplication and parallel. But as GPU evolved, there is also a lot more complexity gets added over time. Modern GPUs have additional circuitry for pixels, operations or for graphics separations like pixel shading or race tracing in the recent versions. GPU also have additional circuitry allocated specifically for deep learning compute, which is tensor cores. So over time GPU become more complicated, and they start to evolve a little bit in the different directions based on what the use case. However, up to now in most cases, people prefer to keep GPU as one device suitable for multiple purposes due to economies of scale. So if you have the same GPU chip, which that was graphics and AI, you basically split your cost, your development cost of larger volumes and you can keep the cost down this way.

Max: Yes, yeah. I see. So there are some differences in configurations, but generally speaking we're still at a point where there's generalized GPUs and very few different types in the market today.

Alex: Yeah.

Max: Okay. Can you please comment on the key players in the GPU hardware market today?

Alex: So the two major players is AMD and NVIDIA, they've been in this market from the very beginning. NVIDIA has been GPU company from the very beginning and that's kind of survived as a GPU company up to today, which is out of 130 companies. This is the one which survived as an independent GPU focused company. AMD made an acquisition, so it's acquired A GPU company 10 years ago, something like that. So AMD also has a very much mature GPU business and it's kind of structured in a very similar way in the sense that it's the same chip which runs both data center... Or almost the same chip, it's not exactly the same, but almost the same chip which runs both the consumer graphics as well as the data center.

Alex: There are a bunch of other interesting evaluations over time. For example, for Sony PlayStation, Sony has its own GPU, which was developed by Hitachi, which offers that. And that was done for the cost reduction because Sony wanted to have a complete control over that the GPU production in order to keep the cost to a minimal because it does everything at very high volume. So a few bucks is a big deal for the consumer gaming business where you have a relatively small margins. So that's kind of unusual to have a GPU done specifically for device.
And Intel, Intel has a lot of different initiatives. It has its own GPU now, Ponte Vecchio, which is not very well known. It also has some of the CPU that Intel has allocated certain amount of silicone to do GPU style tasks. So when you calculate your normal laptop screen, which is not very intensive in terms of computation, Intel chip will do it itself rather than using GPU from other vendors. So you have NVIDIA, you have AMD, you have Intel kind of in and out, you have Sony, which is kind of special purpose GPU. And you also have a lot of innovation today in Chinese market because we have restrictions on the export of certain chips to China. It created a whole industry over there where there's like a dozen different companies do GPU hardware. Most of them will not succeed, but it's possible that some will.

Max: Got it. That's super helpful. Just briefly, one more is can you explain why GPUs are so important to artificial intelligence?

Alex: Yes. So when artificial intelligence became to emerge, people were trying to figure out how to do the compute efficiently because artificial intelligence is much more compute intensive than the normal programming. In case of normal programming, you solve the problems like two plus two equal what? And then normal programming says, "Two plus two equal four." And the problem solved. When you have artificial intelligence tasks, you have inputs two and two and you have an output four. And the question for artificial intelligence, "What is the function which converts your inputs into your output?" And the function can be plus, the function can be multiply, the function can be the degree of. So there are multiple ways to get from inputs to outputs. Therefore, you need not just one set of inputs and outputs, but thousands of sets of inputs and outputs. So you can figure out what is the function, which is the best for converting inputs to outputs. So the typical AI program will require a thousand times greater compute or 10,000 greater computes than a normal traditional programming.

Alex: The question is how do you do it? How do you do such a huge amount of computations faster? And then obviously the logical solution is what kind of computation it is? It's multiplication. Who does a lot of multiplication parallel? What the GPU do? And therefore from the very beginning people were trying to say, "Okay, in order to do my extremely compute intensive application, develop artificial intelligence calculations, I need a device which does lots of multiplications parallel." And at the time GPU was the most suitable device available for the purpose. GPU also had a developed set of linear algebra libraries, which is very important because in order to figure out which function transfers your inputs into your output, you need to do a lot of linear algebra calculations. And GPU at the time already had a lot of linear algebra work done. It wasn't done specifically for AI, it was for general purpose and for many other use cases. But you had GPU hardware with a preexisting linear algebra libraries, which is a good start to do AI.

Max: Got it. Super helpful. Very briefly, what's a TPU?

Alex: TPU?

Max: Yeah, TPU.

Alex: TPU is tensor processing unit. So that would be an ASIC application specific integration circuit, which is kind of like one of the way kind evolution. GPU is still a general purpose, more or less graphic processing units which also do AI. They're semi general purpose device. Once you go further into special purpose computations, the device used for special purpose computations are called ASIC, application specific integrated circuits. TPU is one of those things. CPUs do AI but they don't do graphics.

Overview of the Software Landscape -

Max: Got it. Got it. Super helpful. Okay. Can you please explain the link between the software and a semiconductor and why is the software important?

Alex: Yeah, so it's important to remember that semiconductors themselves, the chips, the chips don't do any AI. Chips don't know anything about AI and don't care. What chips do is multiplication in parallel. So the question is how do you get from AI to multiplication which the chip is able to perform? And what you have in the middle is a whole stack of software. When we do AI today, we use a bunch of methods, approaches or you can call them tricks to extract patterns from data. Those approaches, methods and tricks are developed mostly from academia and those things called things like convolution, max pooling, dropouts. There is a whole language of those technique which help you to find a pattern in the data you're looking for.

And those particular artifacts, those artificial intelligence artifacts.... So a typical AI developer operates within those AI artifacts and doesn't know anything about anything else. He doesn't know anything about linear algebra, he doesn't know anything about the chips, he doesn't know anything about parallel processing. And that's good because all of those things are highly complex and highly specialized things and we don't want AI developer to know anything about this stuff. What we need is a software stack which first converts AI artifacts into linear algebra artifacts. So we convert convolution and max pooling and dropouts into metrics, multiplication and metrics, inversions and tensors and all those kinds of things. Second, we have libraries, linear algebra libraries, which convert linear algebra artifacts efficiently into C code, CC++ code. So both conversions have to be done very efficiently. So AI artifacts into linear algebra has to be done highly optimized and highly efficiently. Linear algebra into C code has to be done highly optimized and highly efficient.

Alex: And the C code, the core of the CUDA is CC++ compiler, which is a standard CC++ extended with a bunch of features and needed to perform GPU compute. So GPU will process the C code and issue commands to the actual hardware using drivers as a mechanism. So as a result, in order to do AI compute, you don't just need chips, the chip itself will not do anything for you. What you need is a full stack of software and hardware. And the very top, you have AI developer who describes his neural network architecture using the tools he has like TensorFlow, PyTorch. Then in the background, those artifacts get converted into linear algebra artifacts which get converted into C code, which get converted into the instructions for the hardware. All of that stuff on the background has to be done and unless it's done, the AI is not going to work at all. So therefore what companies are buying, what customers are buying are not just the cheap and not just software, what they're buying is a complete software stack, which enabled them to do AI development and takes all the complexity out of the equation.

Max: Got it. And that's what they're getting in CUDA, right?

Alex: Yes. So what they get is a complete stack, which I would say includes chips, includes CUDA, which is the middleware, and on top of that, NVIDIA also developed domain specific libraries which make it even easier for people to develop instead of low level AI code, something a little bit easier. There are libraries for computer visions, there are libraries for speech, for large language model. So there is always a question how much the hardware vendor should offer, but in general, the hardware vendor should offer at least the hardware itself, which is the chips, the middleware, which is CUDA, and at least some support to its customer to help them getting started with development and in some cases more than a little bit, and that's the main specific libraries for specific AI use cases.

Max: What is OpenCL?

Alex: So OpenCL is a language that was developed quite a while ago. It was originally promoted by AMD more than NVIDIA. And what it does, it allows you to manage variety of devices. So it can equally well manage CPU, GPU, edge devices. So it's designed more flexibly to manage variety of devices and still CC++ extension, which allows you to kind of hide a lot of complexity in managing variety of accelerators or core processes and makes it easy for you to write C program which runs on anything. It's very helpful when you have highly heterogeneous environment. So if you have a program which needs variety of devices and they all need to be used to the same times and OpenCL is very helpful. However, when you do training for AI algorithms and you just need billions of multiplication done most efficiently, it's not necessarily the best approach. Most people just go with CUDA and the CUDA tools.

Max: Got it. Super helpful. Does CUDA represent a barrier to entry given they kind of define a lot of the artifacts?

Alex: Partially, yes. So in order to do general purpose compute on GPU, you have to have CUDA type environment. However, a lot of competitors already have it. So AMD has ROCm, Intel has a bunch of things there as well, which are kind of very similar. And that's kind of very funny because sometimes they even call libraries the same way. CUDA has blast libraries for linear algebra and ROCm has blast as well and Intel has blast as well. So they even call things the same way sometimes. But basically CUDA is a general purpose, CUDA and everything similar to it like ROCm from AMD and Intel stack, all of those tools provide a set of middleware which has several parts which are kind of similar, very similar.
One is the compiler, so it's an extension of CC++ compiler which use CC++ and extended for the purposes of managing particular hardware in the most efficient way possible. Then it has profiler to see how well the chip is utilized. It has communication libraries, it has sparse libraries, it has linear algebra libraries. So there is basically like a set of middleware tools, middleware software, which is necessary to make the whole stack work. And for a while because NVIDIA was the first to jump on board, the advantage was kind of significant. The CUDA was much more mature than other companies were offering.

Alex: But over time others are catching up, and by the way it's not only AMD and Intel, but also a lot of companies offering ASIC, many of them have hardware but working on building up their software stack and the software stack to a certain degree looks very similar. So it's CC++ plus with extensions to run particular type of hardware with a lot of other tools which are very similar. So over time the whole industry, all of the competitors are kind of building that middleware layer and in some cases they're pretty close to what NVIDIA has by now, even though they were behind, they were catching up. So that's why NVIDIA is kind of much more invested into the main specific extensions. So they know that on the hardware there is only so much they can do before others can catch up. On the middleware, there is only so much they can do before other catch up. So NVIDIA latest strategy is to go after the main specific libraries while competitors are catching up on the other two areas.

Max: Got it. So just to be clear, interoperability is possible.

Alex: Yes, interoperability is possible, but obviously different vendors have different preferences. Many of the hardware vendors will prefer to have a lock-in. So once you start working with particular type of hardware, it becomes more difficult for you to switch. At the same time many software vendors, so if you look at what Facebook doing or what Google are doing, all of them would like to see the hardware as interchangeable commodity. So from a software vendor point of view, people kind of up the stack from the hardware companies. From software vendor point of view, it's much more beneficial to have interchangeability of the hardware so they can get hardware cheaper so they can get more innovation and low price from hardware vendors.

Max: Got it. It would technically be possible to use a chip with a different language, but as of today they're sold, NVIDIA sells the chip, you have to use CUDA on that chip or can you just buy it the chip and then configure it to use another language?

Alex: So there are both Meta and Google have a project to develop CUDA compatible compiler. So basically makes it not necessary for you to use CUDA and they figured out the reverse engineer NVIDIA instruction set and did all those kind of things. So CUDA PyTorch is able to compile CUDA compatible code making CUDA is not necessary. Google GCC compiler does something very similar in a slightly different way. OpenAI has a project Triton, which is also kind of in that category even though it's more for research purposes.
So if you ask NVIDIA they'll say, "Oh, just buy chip with CUDA and everything works perfectly well." Which is true. But if you look at the software vendors, they say, "Well, we'll develop a tool which makes it not necessary for you to be stuck with particular package for particular hardware vendors and makes it much easier for you to have an interchangeable software and software components so you can change things as you need to." So I think historically up to this moment a lot of the usage of NVIDIA hardware included CUDA as a more or less requirement. I think going forward there are a lot of companies like Meta and OpenAI and Google trying to change the approach and trying to make it possible to use NVIDIA hardware with any kind of software and any type of software with a different type of hardware. So making it easier for people to kind of mix and match those components.

Max: Got it. Got it, that's super helpful. And just to put a finer point on the ASICs, are these startups that are building application specific GPUs or is it someone else?

Alex: Yeah, it's both startups and large companies. So Google has TPU and Amazon has Trainium. Those are technically ASICs, but we don't call Google startup, but it's kind of startup project within Google. But those are ASICs from Amazon and Google. There are also a lot of startup companies. The list of those startups is very long. I know several kind of little bit more mature, they already have thousands of people. So I don't know if I call them startups, but I still call them startups. Those are companies like Graphcore, Cerebras, Krock, those are companies that have been around for a while. They started like eight years ago, something like that, eight, 10 years ago. And what they're building is application specific integration circuit. So they're not trying to address all of the GPU use cases. They're saying, "We're going to be specific, we're going to be specific for AI and if we don't fit other requirements that so be it. We're just going after the biggest market and forget about the rest."

Alex: And so they're focused on that particular use case and therefore they can cut a lot of things. They can cut a lot of the complexity out of the chip. They don't have anything graphics related, they don't need pixel trading or ray tracing. They don't need to support any graphics specific libraries. They don't care about backward capabilities with all the GPUs. They just have the circuitries they need to do deep learning, basically compute. So basically, Google GPU looks very similar to NVIDIA Tensor Cores. So individual GPU have a section of the GPU dedicated to the learning compute and kind of Google GPU is similar to that section of GPU, but don't try to replicate the rest of the GPU. It also makes it easier on the software layer because since all of those ASIC companies are not so much general purpose computing environment, but they're specific for AI, they just do the linear algebra pieces that they need and kind of forget about everything else. So they might not give you way to calculate necessarily molecular dynamics. Some of them might, but some of them might not. But they're not obligated to give you any other, they're not obligated to support any other use cases. They'll just make it easy for you to calculate neural networks and basically the rest is no guarantees.

GPUs and Specific End Markets -

Max: Got it. Super helpful. Okay. And so those two sections kind of tee this one up, Alex, and so I kind of want to ask about specific end markets and maybe we can start with if we could just provide an outline of the different types of AI applications that will require GPU hardware. So I have a few examples, but I don't want you to be confined to this list, but it's maybe just to get the conversation started. So things like data centers and autonomous devices, then there's things like edge devices like sensors and cameras, and then there's maybe off-grid applications like military systems. I'm kind of curious how you think about that outline of the different types of AI applications that are likely going to need this hardware.

Alex: Yeah, I think there are many different ways to slice and dice this market because it's kind of big and very diverse. One way to slice it is between data center training and inference. All of the training or almost all of the training is done in a data center because it's very dense compute, GPUs is a good device, et cetera. For inference, you have a variety of different situations. In some cases, GPUs are good for inference. In other cases, they're not good. For example, when you do AI inference for large language model or the internet pictures, GPU is still good. But in other cases when you do decision making on device like a street camera or even car, the GPU might or might not be the right device for you. So one way to split the market is between training and inference because the dynamics of those markets kind of different.

Max: Got it.

Alex: Another way to split the market is by data type and the third one is by industry. So everybody thinks by industry, but let me get to it first. So the second way to split the market is by data type and by data type I mean vision, speech, text and structured data. So there are four different data types and from technical point of view, from development point of view, there are specific difference between those four markets. But this kind of definition of markets into data types also helps you think where AI is applicable because it's not applicable everywhere, right? It's only applicable in specific cases where you have specific particular type of data accumulated in large volumes. So you can use that large volume of data to train particular models to solve particular problems with vision, speech, text or structured data.

And the third approach to the markets is by industry. So if you look at any kind of industry. For example, if you want to start with, I don't know, manufacturing for example. There are specific use cases for AI that are kind of well established and there are a lot of use cases which are kind of emerging or less stable. The same is true for healthcare or for supply chain or for military applications because AI in general is general purpose technology. But that's not very helpful because what you want is to find specific use cases where AI is most helpful and in many cases what you're going to find is within each industry there are some use cases where AI is able to do things right away. There are some use cases where AI is closed but not quite good enough given algorithms we have or the way the data is collected. So the use cases might not be quite ready for AI and there will be other use cases where which are long way out. So within each industry you also have that kind of variety of immediate use cases, medium term and long way out.

Max: What do you think are the most immediate use cases?

Alex: It depends on the industry. So for manufacturing, for example, the more mature use cases are predictive maintenance. So you have every time airplane makes a flight, they collect huge amount of data on every vibration, every noise, every everything. And they don't always do much with this data. So if people are now developing algorithms to get more information out of this data. So you can predict for example, when the engine fails are likely to occur. So which means, which gives you information to repair the engine right away or do opposite, try to keep it running longer because every flight on an engine without repair on a commercial flight, that's a lot of money. Repairing engine is very expensive. So the predictive maintenance is one use case in manufacturing. It has been available for a while and it's becoming better and better and people are able to use more and more of the data they collect to make more and more accurate predictions on what you need to do in order to keep the equipment running properly.

Alex: In other standard example for manufacturing is the quality assurance. For example, you can do visual inspection of a fabric and determine if the textile machine is not weaving the fabric correctly just based on visual. And right now in the past we were using people for that just to look at the fabric, make sure that it looks okay, but you can usually use AI just as well for that. So for manufacturing, predictive maintenance and quality assurance. For healthcare, a lot of it is financial administrative. You can do a lot of it right now if you have the data arranged the right way. But clinical decision support a little bit longer term. For supply chain, a lot of asset tracking is doable today. Some optimization, some of the supply chain optimization works, others is not quite there yet.
For military technology, obviously there are some more simple applications which work pretty well, like identify the object. But then when people say robots doing the fight autonomously, I think it's not quite there yet. So we're kind of in a process of developing kind of more. It takes a while to have those type of algorithms accurate before you can do things like that. In the consumer space, the answer of application is consumer space. But every industry you look at, what you're going to see is evolution, where people start doing something and it might not look necessarily that impressive, but over time as the algorithms become better and the processing power becomes better and the data collection improves, you can do more in more advanced use cases.

Max: Yeah. What is your prediction for the end markets that are going to change the most in the near term?

Alex: I think in the near term, of course you have to define the end term, but I think in near term, the easiest things to do are using AI in support role. So for healthcare, it's financial administrative. In fraud detection, kind of run the algorithms, make sure there is no fraud going on. So those type of use cases, kind of most traditional use cases, which been around for a while, they're becoming strong and powerful enough to do this. But even for large language models, all the rage recently for the last year. If you look at it where the large language models are useful at the moment, they're primarily useful for low cost, high volume communication. So you don't want to use large language model to write PhD dissertation, right? That's not what it is for. But get a quick answer, what's the good sleep medication for me? Those type of questions, the systems are very good already. So what we're going to see AI adoption in many kind of a small supportive simple scenarios which not necessarily change the world dramatically, but make their life easier or kind of more simple for us.

Max: Got it.

Alex: Yeah, and I think there are many scenarios where this can be true.

What's Next for the Market? -

Max: Got it. That's helpful. What are the considerations when we think about the future of purpose-built GPUs, so ASICs, and the technical capabilities and the capital investment required for those? I guess what I'm trying to get at is where is it more likely that we see ASICs and where is it more likely that it's just standard GPUs because the economics make more sense? When did the economics of trying to do something specific, even if it means lower ability to amortize over large number of volume, when does that make sense? Hopefully, that's clear.

Alex: Yeah, that's exactly the dilemma because as you define more of a specific circuit, the cost per unit becomes higher because you're not spreading it over large enough volumes. We do have quite a few ASICs, a lot of companies already calculated that they can make this work. But in general the market has to be big enough. So GPU, for example, if we go five years ago or seven years ago, GPU were used a lot for bitcoin compute, but over time specific bitcoin chips were developed and nobody used GPU for that. So AI theoretically will have its own specific chips which are very designed for this particular type of compute and kind of reduce the need for GPUs.

But it's not necessarily going to happen as fast as it was for Bitcoin or any kind of other cyber crypto markets because AI is actually not a thing, it's more of a collection of things. And because AI is a collection of things, it's actually very hard to design a circuit, which is very simple for the particular use case. You still have to have a device which is kind of does variety of things and therefore ASICs, even though they're called ASICs, application specific, they still have a pretty broad set of algorithms that they have to run and they have compilers and profiles and all other complexity that you have when you have general purpose device. A lot of the ASICs exist today because NVIDIA GPU prices have been kind of high and this is kind of offers opportunity to people to come in and all the ASIC, which works somewhat better for AI compute at the comparable prices. But that's partially the factor of reach pricing environment.

Max: Got it. You mentioned earlier that this was the direction NVIDIA is going to take in your opinion. Like they realized that there's only so long they'll have an advantage in the hardware and the software if it remains general purpose. So they're testing kind of application specific, they're basically testing ASICs. Is that the right characterization?

Alex: They're not testing ASICs, they're developing application, the main specific libraries. What they're trying to do is accomplish the same goal using libraries. So they still committed to general purpose hardware in order to keep the prices low, but they're trying to solve the variety of needs by developing more and more domain specific libraries. So they have libraries for speech called Rivers, they have libraries for checks called NeMo for large language model called TenseEngine something. So they're build in more and more libraries for domain specific functions to make it easier for developer just to take something off the shelf without thinking and just start using it to see how it works. The way they're doing it is going after use cases with software while trying to keep the hardware the same even though NVIDIA does have an option to offer ASICs and they actually experimented with that. They had deploying ASICs in some car. They offer to car manufacturers, for example. They experimented with it but they don't necessarily promote it. For now they are very rock solid to stay with GPU model.

Max: And where do you think that ends up? So you have the general purpose GPU, general purpose CUDA with some PyTorch about adaptability. Then you have NVIDIA's viewer like, "Hey look, we can just develop software that is application specific, but we can still use our general purpose chips." And then there's the ASICs, which sounds like is the chip configuration and the software specific to an application.

Alex: I think the future is actually even more unknown than that because all of the ASICs we have today are still digital technology, ones and zeros. If you think about how to make AI compute more efficient, you might say, why are we doing digital? So you take analog data, you convert it into ones and zeroes and multiply billions of times and then you give an answer, cat or dog. Is there other technologies that better suited? So this is all in the lab at the moment, but there are a lot of people thinking about what are the other technologies which would be better suited for AI compute where the precision is not important or not as important. And analog compute would work, but analog compute is not very reliable, then people look for memory resistors and they're looking for optical aging amplitude of optical waves. People are looking at quantum, people are looking at spiking technology. So there are a lot of different ideas, how to do AI compute different because AI compute is different from the traditional compute, right? So why are we doing AI compute on the same device which was developed for the classic compute?

Alex: Can we have a better device for that? And I think we're going to have a more proliferation of devices longer term, not immediately, but it's coming. It's also possible that we're going to have more interesting algorithms because if you're doing neural networks, GPU is good, right? GPU is good, ASIC is good. Is it better? In some cases it is, some cases it's not. But as long as you're doing GPU learning, GPU is good alternative because GPU specific architecture, which is called SIMG, single instruction multiple data, is actually a good match for neural networks specifically. However, if you do any different type of AI compute, let's say if you do trees or random forest or anything of those type of nature. For those type of AI computes, GPU are actually not good. And imagine next year another type of algorithm becomes popular, it's quite possible that people say, "Well, GPU is not suitable for that particular algorithm, but they ASIC are good." And then you kind of going to see migration towards different device based on the algorithms we're trying to accelerate. So I think we have a lot of opportunity for change and it's not necessarily happening very fast, but we might see changes in algorithms as well as in the basic compute fabric going forward, which might offer significant change to the way we do business today.

Max: Got it. That's super helpful. I want to go back to NVIDIA for a second. Obviously, or not obviously, it sounds like, would love your opinion on it, but it sounds like Google and Facebook making compilers that allow a non-CUDA code to be used with CUDA kind of opens up the software piece and potentially commoditizes the actual chip a little bit more or call it a little bit faster, commoditizes the general purpose GPU chip configuration a little bit faster. How likely do you think it is that they'll be successful in doing that? Do you think the future is a disassociation between the software and the hardware or do you think that stays?

Alex: Well, if you look at the way laptops server was done today, every layer is done by different vendor, right?

Max: Yeah.

Alex: So this vertical disintegration already complete. In case of the GPU, general programming on GPUs, we still have that kind of semi-vertically integrated offering today. I think the industry is pretty determined to succeed with having at least a second source provider. For GPU hardware, I think the most logical second source provider would be AMD and AMD has very interesting chip, MI300 coming out. So AMD is trying to position itself as a second source provider for the AI suitable GPUs today. But by themself AMD is not strong enough because it's missing whole bunch of, not a whole bunch, but it's missing significant part of the software stack, especially domain specific stack. So AMD would kind of depend on cooperation with from Amazon and Meta and Google, et cetera in order to make it work.
So I think there is a chance that all those efforts from Facebook and Google and OpenAI, together with AMD becoming more competitive on the hardware side and the middleware side might give us an option where we have a second source supplier for data center GPUs. And the idea would be similar to what we have in CPU market today because in CPU market today, we have Intel as a primary supply and we have AMD as a secondary supplier. And having that competition by itself already offering us CPU at pretty good prices and driving consistent innovation on the CPU side. So the industry is kind of hoping we can achieve the same for data center GPUs as well. So I think the industry is pretty determined to make it work.
The question is how long will it take? How when will they succeed? Because the market has keep changing all the time and NVIDIA is still making the best effort in terms of keeping up with the market evolution, providing the libraries necessary to make the work for developers easy. It's not that easy to replace existing leader. However, when the market becomes more mature and slightly less fast-growing or the algorithms become kind of known and not changing too much, I think it would be easy for AMD to become a second source provider for this particular market. Now second source supply doesn't necessarily replace NVIDIA. What it does basically takes a big chunk of the market reduce price and increase innovation.

Max: Yeah. Just last question on this point. Is there anything that can't be done using an AMD chip and a Google compiler plus PyTorch plus whatever, like CUDA or an equivalent language makeup, tech stack makeup that you can do with an NVIDIA chip and CUDA? Has the modular alternative fully caught up?

Alex: Probably not fully. So if you have a compiler like Google Compiler or PyTorch compiler, which is trying to work with multiple hardware devices, it's probably going to be never as optimal as a native compiler. But in some cases, you don't necessarily care about getting the last 10% if you get 10% reduction price or you get the 10% otherwise, right? So if it's multi hardware compiler might not be as optimized as a hardware specific compiler. But as long as you get your money back in other ways, like by having cheaper hardware, by having more variety, by having a bunch of completely new vendors which are willing to undercut price dramatically, I think maybe you don't care about total optimizations.
I think the biggest area where they need to catch up is just basically having all of the software libraries, everything is integrated, everything is packaged. Because today, even if it's possible to run algorithms in a new way using the new approaches. Many people say, "Well, yes, it's possible to do it, but I have to do extra work. Do I want to do extra work? I have a shortage of developers, I have unreasonable shipment timeline." What I'm going to do is I'm going to go with most established vendor, which is today's NVIDIA. And because AI is kind of a little bit of a stressful today, the timeline and reason might be short. The probability of success is not as high as it would be in other areas. People might feel it like it's not worth the risk trying new approaches, that's kind of benefits NVIDIA. But if the new approaches become more established, there are more partnerships there, there is greater ecosystem, more people writing additional tools or software or application to that new stack. I think that over time, it will certainly become a different world.

Max: Got it. Got it, got it. Well, Alex, thank you so much for your time today. I think that sums up the questions we had. This has been super, super exciting, super, super informative. I can't think of a better person with whom to have this conversation, so we really appreciate you taking the time.

Alex: Thank you very much for having me.