EPISODE 84

[INTRODUCTION]

[0:00:10.8] SC: Hello and welcome to another episode of TWiML Talk, the podcast where I interview interesting people doing interesting things in machine learning and artificial intelligence. I'm your host, Sam Charrington.

This week on the podcast, we're featuring a series of conversations from the AWS re:Iinvent Conference in Las Vegas. I had a great time at this event, getting caught up on the new machine learning and AI products and services offered by AWS and its partners.

If you missed the news coming out of re:Invent and want to know more about one of the biggest AI platform providers is up to, make sure you check out Monday's show, TWiML Talk #83, which was a round table discussion I held with Dave McCrory and Lawrence Chung. We cover all of AWS's most important news, including the new SageMaker, DeepLends, recognition video, transcription service, Alexa for business, Greengrass, ML inference and more.

In this episode, I'll be speaking with Nikita Shamgunov, co-founder and CEO of MemSQL, a company offering a distributed memory optimized data warehouse of the same name. Nikita and I take a deep dive into some of the features that they recently released 6.0 version, which supports built-in vector operations to enable machine learning use cases, like real-time image recognition, visual search and predictive analytics for IoT. We also discussed how to architect enterprise machine learning solutions around the data warehouse by including components like Data Lakes and Spark.

Finally, we touched on some of the performance advantages that MemSQL has seen by implementing vector operations using Intel's latest AVX2 and AVX-512 instruction sets. Speaking of Intel, I like to thank our good friends over at Intel Nirvana for their sponsorship of this podcast and our re:Invent series.

One of the big announcements from re:Invent this year was the release of Amazon DeepLens, a fully programmable deep learning enabled wireless video camera, designed to help developers learn and experiment with AI in the cloud and at the edge.

DeepLens is powered by an Intel Atom X5 processor, which delivers up to 100 gigaflops of processing power to onboard applications. To learn more about DeepLens and other interesting things Intel has been up to in the AI space, make sure to check out intelnirvana.com.

Okay, just a couple more quick announcements. You may have heard me mention last time that over the weekend, we hit a very exciting milestone for the podcast; one million listens. What an amazing way to close out an amazing year for the show.

It occurred to us that we'd hate to miss an opportunity to show you some love, so we're launching a listener appreciation contest to celebrate the occasion. To enter, just tweet to us using the hashtag twiml1mil. Every entry gets a flytwiml1mil sticker, plus a chance to win one of 10 limited-run t-shirts commemorating the occasion. We'll be giving away some other mystery prizes as well, from the magic TWiML swag bag, so you should definitely enter.

If you're not on Twitter, or want more ways to enter, visit twimlai.com/twiml1mil for the full rundown. Last but not least, we are quickly approaching our final TWiML online meetup of the year, which will be held on Wednesday, December 13th at 3PM Pacific time.

We'll start up by discussing the top ML and AI stories of 2017, and then for our main presentation, Bruno Goncalvez will be discussing the paper Understanding Deep Learning Requires Rethinking Generalization by Chiyuan Zhang from MIT and Google Brain and others.

This will be a fun meetup and one you don't want to miss, so be sure to register at twimlai.com/meetup if you haven't already done so.

Now, on to the show.

[INTERVIEW]

[0:04:24.5] SC: All right, everyone. I am on the line with Nikita Shamgunov. Nikita is CEO and co-founder of MemSQL. Nikita, welcome to This Week in Machine Learning and AI.

[0:04:35.1] NS: Thank you. Thank you for having me.

[0:04:36.9] SC: Absolutely. Nikita, why don't we get started by having you tell us a little bit about your background. You are the CEO of MemSQL, but you've got a pretty technical background, isn't that right?

[0:04:46.4] NS: That's right. I'm on my PhD in computer science from St. Petersburg, Russia. I moved to the states to work on Microsoft SQL server, which I did for a number of years. I have a very strong database and query processing background

After that, I moved over to Facebook where I was blown away by the magnitude of data problem Facebook was solving back then. Since they only increased in magnitude. I've seen that and combining the visibility into those workflows and making that assumption that in five years, a lot more companies are going to be facing those challenges just like Facebook. I decided to start a company and combine that database background and expertise of building systems.

The early stages and the glimpse of the workloads that I saw at Facebook, so I knew where the world was going into. That triggered my desire and gave me some insights into starting MemSQL, the company. We've been at it for six plus years, and certainly validated some of the assumptions we had starting that journey.

I've come across MemSQL and I think of the company as an in-memory database. Tell us a little bit about what the focus is there, and in particular what's the intersection between what you're doing and machine learning and AI. Are you seeing a lot of those types of workloads nowadays?

[0:06:27.9] NS: Definitely. That's certainly where our customers are moving towards. But let me step back for a second and talk about MemSQL and in-memory, database and the technology. We started as a purely in-memory database. Then since we've evolved to support a large class of applications that I built on top of MemSQL. In-memory became an enabling technology, but it's not the technology at MemSQL.

As a matter of fact, the key advantage that MemSQL brings to the world is the fact that it supports SQL, which is Structured Query Language, and the fact that it runs the database in a distributed environment. You can run MemSQL on your laptop, or you can run MemSQL on a thousand hosts that gives you an immense compute power to enable the new class of applications.

Let me talk about what kinds of applications we support. Certainly some of them have to do very little with AI and ML and MemSQL enables scale, latency. You can build very low latency applications on top of MemSQL and that's where in-memory technology has come handy.

You also can build applications that require very high levels of concurrency. That concurrency is enabled on top of the system of record, so MemSQL supports state and it supports full durable persistence. SQL also allows you to build what we call real-time applications. The idea of real-time applications is the opposite of batch. Every time you need to do analytics and you do some sort of pre-calculations upfront using Hadoop, or data warehouses, or any other offline and batch-oriented technology, that's basically our enemy.

We're bringing the world to be completely real-time and perform all the computations that you need to live. We deliver it on extremely low latencies by leveraging an immense amount of compute. That we can do very, very well because MemSQL is a scalable technology that can run on clusters of commodity hardware.

Now where AI and ML comes in, well because we support this new class of applications, the fact that our customers are incredibly forward-looking, they want to do more with the technology, and they want to blend classical database workloads with the new types of computations that are mostly stemmed from the AI and ML needs.

One of the big ones that we see all the time is image recognition. We can talk a little more about it. So let's say you have an application and you use that to power and you use MemSQL to power that application. The application is large-scale. There is a ton of data that's stored in MemSQL. Like I said earlier, in-memory is an enabling technology, but it's not the technology. You can actually put a lot more data into MemSQL than there is memory on the cluster.

What you want to do is you want to enable smart applications, the ones that make decisions either on behalf of a user, or they provide recommendations, or they provide some sort of search capabilities on top of unstructured and semi-structured data.

Imagine an app that has a camera that runs on your cellphone, you snap a picture with this app and just in a few tens of milliseconds, this app finds similar images to the one you just took a picture of. Why is it useful? Well, it's useful because you just enabled visual search. The way we do it is we built some of the building blocks that allow you to run and operationalize a machine learning models, and we built them straight into the database. Now the database allows you to scale them and deliver very low latencies for these type of operations. That would be one example of an application. I can give you another one, which we see a lot in the IoT space.

[0:10:56.9] SC: Before we go into the next example, can you drill down a little bit into what specifically MemSQL is doing to enable the first example? For example, are there pre-trained models for image recognition, or image similarity in this case built into the database? You might think of a stored procedure. Or is there something else? Is it a different type of functionality?

[0:11:25.2] NS: Yeah. It's even lower level than that. MemSQL certainly supports storage procedures, but in this particular case we implemented a few building blocks, particularly dot product and Euclidean distance between vectors.

[0:11:40.1] SC: That is pretty low level.

[0:11:41.5] NS: Yes. If you take a deep learning model and you look at the layers, matrix multiplication, tensor multiplication, vector multiplication is the fundamental building block. What we do is as we train that model, we take all the layers except for the very last one and apply it on the database of images that we have. That allows us to extract what we call, or anybody else calls a feature vector, which is just a vector.

Once we have that and we have a model, applying that model to an incoming image which you just took a picture with your cellphone will produce another feature vector. It just so happens that the multiplication of those two feature vectors normalized gives you the similarity score, how close those images are together.

The heavy-lifting of building model belongs to somewhere else. The data might as well be still stored at MemSQL and we enabling very fast data transfer in and out of MemSQL in a parallel, so we can send it into Spark or Tensorflow, or any other training framework. Once it gets to operationalizing that model and performing the last mile computation by really powering your app, then MemSQL gives you that scale and it allows you to perform those computations pretty much at the memory bandwidth speed, that enables really low latencies for that last mile computation. Does it make sense?

[0:13:21.6] SC: Okay. It does. You’re computing these feature vectors. Is that happening on write of new images, or is it happening – I’m assuming that’s the way you would do it since you were anti-batch. You’re not doing some big batch job that’s like updating some column in your database with your feature vectors for all the images that are in there.

[0:13:42.6] NS: Correct. You can either, but the typical workload is once you have that model, you’re applying that model on write and we have technology, it’s called Pipelines that allows you to perform arbitrary computations, either in a storage procedure or an external piece of code. That’s where you can invoke all third party libraries to apply that computation, then the feature vector in the database.

You can do it and then adjust. Let’s say you built something, like you crawled the web or you crawl your own product catalogs. Once you identify those images, you immediately stick them into the database. As you do that, we trigger that computation. The feature vector arrives into the database at the same time instantly as the actual data.

Now it’s immediately participates in all sorts of computations. That allows you to never have stop and go computations. You never do, “Okay, step one load all the data. Step two, perform the batch computation on top of all the data into the database. Step three, do something else with it.” Rather it’s all streamlined and it just flows in and out.

[0:15:00.7] SC: How does this play out in the IoT case?

[0:15:05.0] NS: In the IoT case, data isn’t coming constantly. One of the use cases we have with a large energy company is to ingest IoT data from drill bits. Apparently in the world of fracking, you tend to drill a lot more. Then there is a non-trivial cost for a broken drill bit. Those things are extremely expensive. You have to stop your operation if a drill bit breaks. You’re losing not only on the fact that you’re fishing this thing out from the ground from hundreds, or maybe even thousands of miles deep, but you’re also not producing oil, which is an operational cost.

What we do, we ingest that real-time data, IoT and we score in that data applying the machine learning model in real-time. Then there is a application built that arrest the drill bit before it hits a problem, just by measuring temperature and all the very – temperature, throughput, all sorts of vital signs of the drill bit as it goes through the ground. That was step one.

Step two is obviously you feed back all these – you feed all these information back to direct the drilling. In the world of fracking, drilling is directional. It's not just vertical down and into the ground, but it's more like you're changing the direction as you go and drilling. Using all that input, you can direct the drill bit to make the whole operation a lot more efficient.

[0:16:46.2] SC: What are some of the algorithms that come into play in that use case?

[0:16:51.4] NS: They started with – like every typical data sense, they started with some sort of linear regressions, that they moved it to decision trees very quickly. Now they're experimenting with deep learning for that as well. The beauty of that – of the solution that we presented is it integrates natively with all sorts of third party libraries. We made the experimentation for them very, very straightforward. They started with SAS, where they would produce models in SAS. As you know SAS is a proprietary technology. Since they've played with Spark and now experimenting with Tensorflow as well.

[0:17:31.1] SC: Okay. I was actually going to ask you about Spark and how that fits in with your model. I imagine you see it out working with customers. Is it a competitive technology? Spark plus the rest of the Hadoop ecosystem, or is it complementary? How do you see that?

[0:17:51.5] NS: All the big data technologies overlap a little bit. In the case of Spark, I would say it's 90-10. It's 10% competitive and 90% complementary. Here is why; Spark doesn't have state. The state is usually stored somewhere else. It's either HDFS, a relational database, or some sort of object store in the cloud.

What we do in this case is we provide an extremely performance state. The combination of MemSQL and Spark, that's the 90% case. They work really, really well together because we give you that transactional scalable state, and nothing else on the market can give you that state. Not only you can store, you can retrieve, but you can also compute. We have a world-class SQL query processing engine that allows you to produce reports, but also allows you to modify that state in a transactional fashion.

You can say it became transaction, insert, update, delete, you can run it at high concurrency and you have full durability and all sorts of guarantees for that data. That's where we're extremely complementary.

The typical deployment model is that there is a data lake and that data lake stores hundreds of terabytes of petabytes of data. MemSQL is deployed alongside of the data lake to provide to power applications, because you cannot write applications on top of Hadoop, because Hadoop is batch. Then Spark is the glue between those two and it allows you to have rapid data transfer, all the data that is born in MemSQL based on the interactions with applications.

That data is captured, you can pick it up in Spark very, very easily. We have a world-class Spark connector. Drop it in the data lake for historical, archival compliance type storage and then provide some – then perform some overnight batch computations, take the results of those computations, stick it into MemSQL. Spark oftentimes give you that unified API. Because through that API you can interact with MemSQL, you can interact with a data lake, and it becomes the go-to API for application developers.

MemSQL in this case just gives you SQL, then you can attach a BI tool directly into MemSQL, and they can scale the concurrency of data scientist that attached their BI tools to MemSQL and look at the reports and visualizations.

[0:20:39.8] SC: Is it primarily data scientist and folks that are – and end user use of MemSQL? Are you also in this scenario attaching your traditional applications to MemSQL, or are you using some other state technology for building applications that refer to this data?

[0:21:02.3] NS: Well mostly, it's actually application developers, because MemSQL powers applications. Because the nature of applications is changing all the time and we have higher scale requirements for the applications, MemSQL is a perfect technology to power applications like this. I'll give you a few more examples of such applications.

Now because modern applications have AI and ML requirements, that's where that intersection comes in, where you need to have those models that you produced somewhere in your data science lab and you want to operationalize those models. That's where MemSQL plays very, very strongly.

[0:21:44.5] SC: Do you envision along the lines of what Spark has done, building higher level abstraction beyond dot product and Euclidean distance, to enable folks to do machine learning and AI more easily?

[0:21:58.3] NS: Absolutely. Absolutely. We have a number of ideas that are circulating inside the engineering team and certainly influenced by our customers in what they want to do as a technology. What the customers see in MemSQL is that very, very fast, scalable state that they can deploy inside their data centers or in the cloud on the keep, right? Because MemSQL provides world-class compression, it has column-store technology. That state is very fundable, because we support transactions, you can change that state and reach that data with attributes that you compute on the fly, etc., etc., etc.

Now what people want to do is they want to perform computations that are beyond SQL. We are world-class in SQL query processing and that's great. But our customers want to do that and more. When they want to do more right now, we resort to Spark. We say, "Okay, well deploy Spark alongside of MemSQL. Pull the data out and we will give you that data very quickly." Perform the computation, put that data back and then leverage some of the building blocks than we have, you know basic arithmetic operations, vectorized operations, vector operations to do the last mile computation to power your applications.

Now once you do that, you just want to take that loop and you want to tighten the knot. You start bringing all the computations that you – today you've taken data out of MemSQL and put it somewhere else in a temporary store like Spark data frames and you want to perform those computations in place.

There are two interesting problems in that space. One is what the API should be. Today for machine learning AI, the APIs tend to be either Spark-driven, or the Python library is consistent-driven. The likes of NumPy, SciPy, Pandas, Tensorflow. It seems like the Python world lives in one universe. The Spark world lives in another universe.

Then there is the SQL universe that is pretty much ubiquitous, because every database exposes the SQL as an API, and also data warehouse expose SQL as an API. Our view that the Spark universe and MemSQL universe should be enabled by rapid data transfer between the two.

Then the Python universe should be enabled by pushing some of the computations that you express through Python API into the database and putting them on steroids.

[0:24:53.0] SC: Now what exactly does that mean, and can you give an example?

[0:24:55.3] NS: Yeah, totally. Let's say you perform a non-trivial computation on your data. Let's say you have a 100 terabytes of data. Certainly, more data that it can fit on your laptop and you're a data scientist.

As a data scientist, you stack, you live in the Python world and you're a big expert in the libraries like Pandas, NumPy and SciPy and let's say you play with Tensorflow as well. It's very natural for you to express computations on that data to perform – you perform training or perform a scoring on that data in that Python world.

The problem is all the computation is single-threated and all the computations are in-memory. You're stuck at this point in time, because you cannot access a 100 terabytes of data because there is no way you can put a 100 terabytes in-memory. That's where we see the world is going to, where you want those computations to become instant and paralyzable and scalable.

You can just instead of bringing that data into your high-memory machine, you can perform those computations in place, in something that's very, very scalable like MemSQL. You're going to see more and more that happening over time.

[0:26:21.9] SC: You mentioned the three different kind of modes of interacting with this for machine learning, or in general; Spark, Python and SQL. Are there any efforts to extend SQL to give it some kind of machine learning expressiveness? Like SQL's got all kinds of aggregation operations, like select average of X where, whatever. I can imagine something like select predict, why, where, whatever, where you're telling, expressing in SQL that you want a prediction based on something. Does that exist? Are folks playing with that today?

[0:27:12.1] NS: Yeah. If you look at the history and you look at products like – good old products like Teradata, or Netezza, or if you look at SQL survey integration with R, that's certainly something that is happening, where training is not happening in the database, but scoring is. That's a natural way to do things.

We achieve the same functionality through pipelines, where we score data and ingest. I think this is valuable and you'll see databases exposing more and more primitives, the likes of dot product and Euclidean distance. But you'll have instead of two, you will have a 100 of those built into the database. Then you will see packages where it's just a package of storage procedures, which allows you to run those predictions.

Those will work really well for simple use cases. Now if you look at the world of a data scientist today, they actually don't like that. They like to live in the world of data frames, where there is a lot more control over the types of computations that people are expressing. People understand that world of data frames very, very well. I think the future is going to be actually in paralyzing and speeding up computations inside data frames. That's just my opinion.

[0:28:39.7] SC: Do you envision doing training directly in the database?

[0:28:44.3]NS: In the same way as I described the interaction with the data frames. If you want to enable inference in the database, then you don't want to have the database to be a crutch where you're constantly fighting and trying to shoe horn that computation into SQL.

The natural way to expressing those computations is operations on top of data frames. However, if you make the database naturally support those data frame computations, then you have tremendous value from the fact that the database actually owns the state and you never need to transfer that state from your storage to something else. So you're bringing computations closer to data. That's where I see the industry is moving in the next five years.

[0:29:37.6] SC: You just mentioned bringing computations closer to data and that's obviously been one of the things that the Hadoop ecosystem has made more readily accessible to folks, this idea of data, locality. Do you get to take advantage of that with machine learning types of models in general, and this pipelining approach that you have in particular? Or are the pipelines run outside of – separate from any concept of locality of data.

[0:30:08.6] NS: Well, locality is a loaded concept. You can start with, "Okay, well I want to perform computation exactly on the same machine where the data is stored." That's the extreme case of locality. Another way to think about is as you look at the computation plan, let's say in the database terminology, that would be a query plan.

Let's extend the definition of a SQL query plan to a broader concept of you performing some sort of arbitrary scalable computation and you know all the steps in the computation upfront, so you can optimize those computations and produce a query plan for those computations and you will run that query in the distributed environment.

Now then, you look at this and certain primitive computations you really, really want to perform closer to the data, because you will save tremendously on the amount of IO that is happening for in the data transfer. Now from there if you do that, you also deliver on the concurrency of such computations. Now you can perform those computations at a highly concurrently – in a highly concurrent environment where thousands of data scientists are attaching to the same data that is collected and centralized and stored in something like MemSQL.

Then they are performing their computations concurrently, so there are no – you opening up all the data to the organization and not having any data silos. Data locality for computations is useful where it makes sense, and not as useful in a way – in a broader general purpose way.

The perfect system would be engineered around pushing the computations closer to data where it makes a ton of sense. For example, you do a lot of filtering, or you're performing a lot of what is called group by operations. Those operations make sense to be done locally. Those operations make sense to be done in a vectorized fashion by leveraging the latest and greatest CPU instructions in the Intel hardware, and so on. It makes sense to leverage indexes, so you prune massive amounts of data so you don't even touch them for your computations.

Then from there, it makes sense to bring all that data into a stateless distributed environment and perform the rest of the computations. That approach allows you to build greater scalability for your system compared to a traditional database, compared to Hadoop, or compared to Spark.

[0:33:04.4] SC: Okay. Now you just mentioned the taking advantage of the instruction sets of the underlying hardware. You guys are making an announcement with Intel at Amazon re:Invent next week. Is it related to that area?

[0:33:19.8] NS: Definitely. Intel has been supporting vectorized instructions for some time for over a decade and every new generation of CPUs increases the size of the vector that you can use and perform vectorized operations on top of it. Where we stand right now, it's 512 bits. Compared to matrices in a GPU, this is tiny.

However, when you perform a last mile computation like dot product, it allows you to perform that computation much faster than the memory bandwidth that you have. You actually don't need more for that last mile computation than AVX-512, because your limitation is not your compute, your limitation is memory bandwidth.

This is very, very different when you train models and perform computations for building deep learning. That's where you perform a lot of tensor multiplication. The amount of computes compared to the size of your data set is tremendously more when you're performing something like dot product. That's where you want GPUs. But once the model is built and the model is trained, you actually don't need GPUs to score that model in many, many cases.

What that gives you is it gives you the ability to democratize that computation, because Intel CPU is everywhere. If a machine has a GPU, it also has an Intel CPU. The point that I'm making that in many cases, all you need is a CPU and you're not going to get the computation faster if you add a GPU to the system, simply because it's the memory bandwidth that you bottleneck in the computation.

Now we use AVX-512 for vector dot product, for Euclidean distance and for other vector operations that we built into the database. We also use AVX-512 for general purpose SQL computations. That allows you to just deliver on orders of magnitude, faster SQL query processing that our competitors have and certainly what Hadoop has, or Spark. We're huge fans of that technology and we can't wait what Intel allows us to perform computations on larger vectors. Not just 512 bits.

[0:35:49.6] SC: Can you be more specific in terms of some of the actual results you've seen in production systems in terms of performance differences?

[0:35:57.7] NS: Yeah, absolutely. With MemSQL 6.0 that we just released in October, we can do things like a group by operation on a 100 billion raw table in a sub-second.

[0:36:13.2] SC: A 100 billion?

[0:36:14.2] NS: A 100 billion. Yes.

[0:36:15.7] SC: Wow.

[0:36:16.6] NS: It's a sub-second operation that runs something, like give me an average stock price over a 100 billion data points and group it by security or by a stock value. The computation itself is relatively straightforward, because we perform that computation on "compressed data." We're using those vectorized operations.

We achieve performance of a billion operations per second per core. If you throw a 100 cores into the system and MemSQL is extremely scalable, so you can throw a 100 cores or you can throw a 1,000 cores. You can achieve this type of performance on your system.

[0:37:03.7] SC: That's using the AVX-512 instructions, what were you seeing prior to that?

[0:37:09.6] NS: Well, there is a combination of two techniques that goes into this type of performance. The first is how can you perform operations on compressed data? If like I said earlier, oftentimes it's the memory bandwidth that's the bottleneck of your computation. The better you compress, but the better off you are in the final computation, because at the end of the day you scan a fewer bytes.

[0:37:35.7] SC: What exactly do we mean by compressor? Is it your traditional binary compression, or are we also talking some kind of de-duplication or something like that?

[0:37:45.0] NS: It's a combination. First of all, the compression is called column store compression. We also in the code similar values. Let's say in that example that is said was stock trading, you have 10,000 different stocks. If you have 10,000 different stocks, then each individual stock can be encoded was just a few bits.

Then you only use these many bits to represent each stock value. As you perform your computations, you never go back decoding those bits into, let's say an integer value, or God forbid a string value of a stock, because those computation will become much more expensive.

Now that you perform computations on compressed data, the second step that you do is you optimize your computation for as few branch mispredictions and as few cash misses as possible. Those are a big deal, because every time you have a branch misprediction you flash your CPU pipeline, so that slows down your computation. If you express your computation so there are no branches basically. There are some very, very cool papers out there and there are some innovation that we've done here at MemSQL as well. That will give you the second boost.

Then the third boost is now you take all those bits that you represent your values and you perform vector operations on top of those encoded values. Then that's where you use AVX-512. That gives you another, I would say 3 to 5X performance improvement just by using AVX-512. But if you take that, plus you take operations on compressed data, plus you take care of branch misprediction. You multiply all those together, and then you can routinely see to your orders of magnitude improvement and performance.

[0:39:47.9] SC: Wow. That's pretty fantastic.

[0:39:49.6] NS: The point here is fancy hardware is cool and you want to use it where it's needed. But also, there is a lot of potential in the hardware you have at hand, and if you take full advantage of that hardware, you will have remarkable results.

[0:40:06.2] SC: That's great. I guess as we wrap-up, do you have any additional advice to folks that are looking to get better – just to take better advantage of whether it's their data storage systems, their hardware systems and to use these to make better predictions and better your life's machine learning and AI.

[0:40:35.7] NS: Well, my advice here is don't settle. There are systems out there such as MemSQL that allows you to take pretty much your end-to-end machine learning AI and application development pipeline and make them completely real-time. Whatever you do in batch, whatever you have to wait for, with the right technology can be squished down to zero.

[0:41:02.6] SC: Is that the technical term?

[0:41:04.6] NS: Squished to zero? Yeah, you can say that. That's where the world is going. We'll live in the world in the future where a compute and storage are going to be utilities. If you're willing to pay for compute, you will have as much compute as you need at any concurrency as you need, and any latency as you need. The only thing that is going to be bounded by is the amount of money you pay for that compute.

[0:41:35.9]SC: Great. Well, Nikita thank you so much for spending the time to chat with me. I really appreciate it.

[0:41:41.7] NS: Yeah, absolutely. Thanks for having me.

[END OF INTERVIEW]

[0:41:47.2] SC: All right everyone, that's our show for today. Thanks so much for listening and for your continued feedback and support. For more information on Nikita or any of the topics covered in this episode, head on over to twimlai.com/talk/84. To follow along with this AWS re:Invent series, visit twimlai.com/reinvent.

Of course, we would love to receive your feedback or questions either via a comment on the show notes page or via Twitter to @twimlai or @samcharrington.

Thanks once again to Intel Nirvana for their sponsorship of this series. To learn more about DeepLens and the other things they've been up to, visit intelnirvana.com.

Thank you once again for listening and catch you next time.

[END]