Welcome to MIT's *Computer Science and Artificial Intelligence Labs Alliances* podcast series. My name is Steve Lewis. I'm the assistant director of Global Strategic Alliances for CSAIL at MIT. In this podcast series, I will interview researchers at CSAIL to discover what they're working on and how it will impact society.

Christina Delimitrou is an assistant professor in the Electrical Engineering and Computer Science Department at MIT where she leads the SAIL group. Previously, Christina was an assistant professor in the Electrical and Computer Science Engineering Department at Cornell University.

Her main interests are in computer architectures and computer systems-- specifically, work on improving the resource efficiency of large-scale data centers through QoS-aware scheduling and resource management techniques. Other interests include designing efficient server architectures, distributive performance debugging, and cloud security.

She earned a master's and PhD in electrical engineering from Stanford University. Christina has been the recipient of several research awards from Google, Microsoft, Facebook, and Cornell. Her work has also received five IEEE Micro Top Pick awards, one Top Picks honorable mention, and several Best Paper Awards.

Thanks for your time today, Christina. Can you bring our listeners up to speed on the landscape of cloud computing and what innovation is going on in the space?

Sure. So thanks, first of all, for having me. So when we talk about cloud computing and data centers, we usually talk about systems that have at least 10,000 servers-- typically more than that-- and that holds distributed internet scale applications. So there's two aspects to this. There are private systems-- for example, the internal data centers that Google, Microsoft, Facebook operate, which run applications that usually they write and then users interface with them from a client perspective.

And there's also the public cloud, which is something that grows significantly and has grown significantly over the past few years, and is expected to continue to grow over the next few years as well, and that is the case where any user-- anyone can go and rent a number of resources on a data center that a company operates and run whatever applications they may want to.

And there aren't strict performance guarantees in this case, but there are guarantees that you will receive the resources you reserved, and you can run your application at a much lower cost compared to if you were to set up your own resources, have to maintain those resources, and so on.

As far as changes to the landscape of cloud computing, there are two main ones I can think of. One is on the hardware side and the other is on the software side. So the first thing to point out is that cloud computing got a lot of cost benefits by taking advantage of economies of scale through the use of commodity equipment.

So if we were to compare cloud computing to, let's say, supercomputing or high-performance computing, the last two use much more specialized equipment-- so they have specialized cores, specialized memory, definitely networks-- and the applications are designed specifically for those systems, so that leads to a higher cost.

By comparison, data centers go to grow so quickly and scale so quickly because they primarily use commodity equipment. So these are servers that are not unlike what you would have for your desktop or your laptop machine. It does have more resources and then hundreds of thousands of copies of that server.

So over the past few years, we've seen a significant change from that. We've seen a lot more heterogeneity on the hardware side of data centers. And there's a lot of good reasons why that happens. Primarily, it's the fact that Moore's law, which we always talk about. When we talk about compute architecture is slowing down, so that means that we cannot get this exponentially improving performance without also increasing the power of the device or the cost of the device.

And for that reason, people have looked for more specialized solutions for how to continue to scale the system without also increasing the cost significantly. So that is the one recent trend that there is a lot more heterogeneity. Some of this heterogeneity is through special purpose chips.

One of the more popular examples is the Google TPU, which is for machine learning, for deep learning. And then the other approach is through reconfigurable computing, like FPGAs, where you can customize the hardware to the computation that you're running at each point in time. So that is the one significant trend. The other one that's on the software side has to do with how we write cloud applications today.

And the way cloud applications were written, let's say three or five years ago, was as monolithic designs. Monolithic means that if you want to design a web service application, you design it as one large codebase, and you deploy it as a single application. And if at any point in time you want to scale up that application, you scale the whole thing.

Now, the disadvantage of that is it's not a very scalable approach because it's very complex to design these large applications. And it also makes it very difficult to debug them because if you have some issue in them, you have to go and dig through the whole codebase to find what the problem is. So a lot of cloud providers have moved to more fine-grained programming frameworks, like microservices, like serverless, which promote this idea of faster development, faster deployment, easier debugging, and then more elastic scale.

So those are the main two trends that we see in cloud systems today. Both of them come with significant challenges, and a lot of the work that we do tries to address those challenges. But there's good reasons why they have happened.

So let's talk about hardware. What are the current limitations that you're working on to redesign cloud hardware?

So like I said before, because we have this increased heterogeneity on the hardware side, that comes with a lot of challenges. So one difficulty of heterogeneity is that now the system needs to be aware of the different performance characteristics that each heterogeneous platform has. So you need a smarter system-- a smarter scheduler, let's say-- that is more careful about where it places computation.

Because depending on where computation gets placed, it will get very different performance. So that is one aspect that we're trying to address. Another aspect has to do with programmability. These hardware accelerators, whether they are special purpose chips or they are configurable accelerators, they're much harder to program compared to your typical CPU, which supports all different programming languages, different compilers, and so on.

So that is another challenge. There's a lot of groups that are working in that area, but-- including my own-- and what we try to do is provide a more programmable interface, a better programming model, where you don't expose the user to all the complexity of hardware heterogeneity but you still allow the cloud user to get the benefits as far as performance is concerned and as far as power efficiency is concerned.

And what about microservices? How is that helping meet this challenge?

So the benefit of microservices is that they offer better elasticity. Essentially, you decouple the concerns of your application. So one popular example for microservices is social networks. So if we look at Facebook, we look at Twitter, those are both large-scale cloud applications that are built using this programming model.

So what does using microservices means? It means that the little widget that you see that tells you who you might want to follow, that is one microservice. Another widget that tells you how many followers you have, that's another microservice. So all those are loosely coupled components that communicate with each other over the network, but they don't have strong dependencies between them.

So the benefit of that is that if today I want to change the machine learning model I use to provide other recommendations, I can easily do that by only changing that microservice, and I can leave the rest of the network topology, the rest of the application topology unchanged. So it promotes this idea of very agile, very fast development and very fast deployment.

At the same time, even though I said that these applications are loosely coupled, they're not completely independent from each other. So that complicates things as far as resource management is concerned. So resource management is the idea of deciding how many resources to dedicate to each of your cloud applications so that they achieve the performance they care about.

And because these applications depend on each other, if I get one decision wrong-- so if I give insufficient resources to one of the microservices-- I won't only hurt that microservice, I will also have dependent services that rely on that microservice for data or for computation. And that includes both upstream and downstream services.

So that is one aspect that we have spent some time in my group researching because it's something that is not easy to solve in an empirical way or in a manual way just because of the scale of this application. So to give you an example, a typical cloud scale microservice topology can have hundreds of unique microservices, and then each of those scaled over tens of thousands of instances. It's a really complex system. It's very difficult to debug manually.

So can you tell us how machine learning is affecting cloud performance debugging? How is it used today?

So that's one very promising approach. The motivation is that manual debugging is very difficult. And even though there are people with high expertise that manage these data centers that know what the applications look like and what the system looks like and typically what went wrong when something goes wrong-- because of the scale of the system, it's hard to do that in an empirical way or to continue to do that in an empirical way.

So one approach that we've been looking at is applying machine learning to the idea of performance debugging. So the goal is, if something goes wrong in the system, if you get performance that is worse than what you expected, can you quickly diagnose where the performance-- where the issue started from before it gets to spread to other neighboring microservices?

Because if you wait until it spreads to neighboring microservices, then it's very hard to restore performance and it can take a long time for the system to recover. So we've looked for-- we've looked at a few different approaches for that purpose. Some are using supervised learning, which assumes you have some information about what the system looks like, what are critical factors with respect to the application.

And some are using unsupervised learning, which is a more scalable, more practical approach. Usually, it's hard to achieve the same accuracy as supervised learning, but we've had a recent project in collaboration with Google where we show that this is a system that can scale across a cloud scale system, achieve good performance, and quickly diagnose what the root causes of performance issues are in microservices specifically.

And could you tell our listeners a little bit about your project Ditto?

Sure. So one major challenge when it comes to research related to cloud computing or data centers is that it's very hard to get access to realistic cloud applications, cloud services. And the reason is self-explanatory, because cloud providers do not want to release proprietary code-- open source, to open source.

So what Ditto tries to do is create a synthetic proxy-- so create a clone-- of a production cloud service without revealing in detail what is the design of that service to the point where somebody can reverse engineer it and copy it. So essentially, we want to clone the performance and resource characteristics but not clone the individual instructions or individual memory accesses, which is where a lot of information leakage would come from.

There is previous work on application cloning. It has primarily focused on simpler applications that run on a single machine that mostly run at user level-- so the more traditional, small scientific benchmarks that we tend to use in computer architecture research. The innovation with Ditto is that it looks at the end-to-end stack of an entire microservice topology.

So that includes from the moment a request is sent from the client as it goes through the whole microservice topology-- so the frontend, the different logic services, the backend databases, and then back to the client. And that includes both user level activity and kernel level activity, including the network stack.

So that is a significant innovation because for this type of applications, you might be spending the majority of your time processing network requests or spending the majority of your time at kernel level. So that's not something that you want to miss. And what we were able to show with Ditto is that you can clone complex applications.

So you can reproduce their performance and resource characteristics, and that synthetic code can be released in the open source community and can be used for a lot of architecture and system studies. So it has a couple of benefits. One is for academic researchers because, obviously, we don't have access to proprietary code.

But there are benefits for cloud providers as well because if you think about a software cloud provider-- like Facebook, for example-- that wants to buy a new generation of servers, they have to share some version of their application to the hardware provider to get an idea of what performance they will get from the next generation of servers.

So that is, again, a use case where they don't want to share their proprietary code. But if they have a synthetic version of that code that still mimics the performance and resource characteristics, it would serve that purpose. So it has this dual advantage both for researchers and for cloud providers.

And is that available as open source on GitHub, or how could people find out more about Ditto?

Yeah, yeah. So that is open source. We actually have a paper coming out in three weeks, I think, about a month, in the ASPLOS conference where this work will be presented, and then the work is-- the project is open source as well on GitHub.

What do you see as some of the biggest misconceptions surrounding cloud computing?

I think a major misconception is that academia perhaps is not well-positioned to do research on cloud computing because for the one reason that I was saying before, which is we don't have access to cloud applications. And also, it's very difficult to do experiments at a scale that is representative of what a real data center looks like-- so tens or hundreds of thousands of machines.

And even though those two are both valid concerns, I think academia can play a very important role in advancing what cloud computing systems look like for the simple reason that we can try much more radical ideas when it comes to changing the cloud system stack compared to what is easy to try from an industry perspective, from a cloud provider perspective.

Because at the end of the day, they operate a business, and it's very hard to make radical changes in how that system operates either on the hardware or the software side. So that is a misconception. With this work, including Ditto, we try to remove the first part-- so the first disadvantage that we have as academic researchers, which is getting access to more realistic representative applications.

And then when it comes to experimenting with large scale systems, pulic clouds, environments have been very helpful with that. So for most of the projects that we work on in our group, we usually use some public cloud provider to experiment with larger scale systems. And in any cases, you find bottlenecks and you find issues with your solution that you wouldn't have seen if you only run it on a smaller scale.

And you've done a lot of work on cloud computing and data centers. What advice would you give to businesses to make their data centers more efficient or to improve their data centers? And what do you think are the biggest inefficiencies today with data centers?

Yeah. So one issue that data centers have had for a long time-- it's not as bad anymore with the advancement of machine learning, but it used to be bad for a long time-- is they were running at very low utilization. So what I mean by that is you have a system that has hundreds of thousands of machines and can run up to 100% utilization.

But if you look at the actual utilization of the systems, it's closer to 20% to 25%. So that means that you are leaving a 3x to 4x of compute capabilities on the table. Basically, your system can support a lot more computation. But for whatever reason, you're not taking advantage of the computation.

Now, there's a lot of valid reasons why utilization used to be-- and in some cases, still is-- fairly low, including the fact that the systems often provision for the worst-case scenario-- so for the case where everybody goes online, something happens, everybody checks Twitter. There need to be enough resources for that.

But there's also other reasons why utilization is low that don't necessarily need to be there. One of those reasons is that, for a long time, data centers relied on their users-- not the end user; these are the people that deploy applications to data centers-- to tell the system how many resources they think they will need.

So for example, if I go and rent some resources on AWS, I have to tell the system I need 10 machines with this much memory, this much hardware. And this is very hard for people to get correctly, so usually they err on the side of being more conservative so that they-- even if something happens, they still have enough resources.

And that is one place where machine learning can really help-- and in some of our previous work we've shown that it really does help-- in the sense of automating this process-- so instead of requiring the user to tell you how many resources they think they will need, ask the user to tell you what kind of performance this application will need.

And that is something where conservativeness doesn't play as much overall because these applications have very well-defined quality of service requirements or service-level agreements that they have to meet either in terms of latency for interactive applications or in terms of throughput.

So the user can specify that performance target to the system and then rely on the system to automatically figure out what resources are needed for that performance target to be met. And with that kind of an approach, we were able to push utilization much higher.

So you can never go to 100%, because you don't want to run close to saturation-- performance gets very unpredictable in that case-- but you can go from a 25% to 80%, which means that you can accommodate a significantly larger amount of computation in your system without needing to pay extra in terms of hardware cost.

Is bandwidth a problem these days? I mean, in the early days of cloud computing, that was probably one of the biggest bottlenecks. What about today?

It depends a lot on the application. So for some applications, like data analytics or machine learning, training, a little bit less in case of inference, bandwidth can become a problem. Usually, network bandwidth in data centers is quite a bit overprovisioned. So you rarely run into the case where you're using 100%.

But, of course, as the resolution of these models increases, as the resolution of videos and images of all the types of data that we produce increases, that can become an issue. There are a lot of innovations in the networking community that try to avoid that-- in many cases, through the use of specialized hardware.

And there's also a lot of innovations in terms of performance debugging and cluster management that try to identify performance issues in the networking stack-- in some cases, through the use of machine learning.

And are you seeing any move from the cloud providers to make things run more energy efficiently?

So that is a big-- there is a big push for low power computation as well. And in many cases, that makes a lot of sense, and there are applications that can run on low power code-- on low power processors without sacrificing their performance. In general, when it comes to cloud applications, there's two main classes.

There are the throughput bound workloads, or batch workloads, and then there are the latency critical, or interactive applications. So throughput bound, or batch applications, are things like analytics, things like machine learning, training. And then interactive applications are things like web search, email, social networks, online document editing, and so on.

For the first class, for the batch applications, low power cores work very well because these applications have a lot of parallelism. They don't care so much about how long it takes to perform a single request. They care more about how many requests you perform per second. So for that type of application, you can distribute the work across a larger number of weaker cores and still get the same performance.

Where those type of systems don't work well is for latency critical interactive applications. Because in that case, I still care about how long it takes for a single request to go to the system and come back to the client. And even more than that, I don't care about the average latency. I care about what we call the day latency.

So day latency means, what is the 1% of worst-performing requests? I want to bound that number below some threshold. If it exceeds the threshold, it means that I'm violating the quality of service that that application has. So for those applications, it's hard to achieve performance targets, performance metrics, using low power cores.

But, again, because of how popular machine learning, especially training, has gotten there are a lot of use cases where low power cores make a lot of sense. Another way to improve energy efficiency is by just using the existing systems as efficiently as possible. So for example, the use case that I mentioned before about pushing utilization higher, that is one way to improve energy efficiency because the systems themselves, or servers, are not very energy proportional.

That means that if you are running the system at 20% utilization, you are burning most of the power that you would be burning if you were to run the system at 100% utilization. So it gives you the incentive of pushing the utilization as high as possible, of course, without sacrificing performance. So that's another way of improving energy efficiency without needing to replace the existing infrastructure.

Can you tell our listeners more about your research? Or is there anything that you'd like to tell our listeners about your research that we haven't discussed? And where could they go to find out more information about your work?

Sure. So we talked a little bit about applying machine learning to systems for performance debugging. That is a more general topic in my research group. So it's something that we apply to different areas as well, including scheduling, cluster management, security debugging, configuration debugging, even design debugging-- so automating the process of writing these cloud applications through higher level programming frameworks because a lot of the bugs that we have in cloud systems today are due to human error.

It is because-- they are because we rely on people to write these cloud applications down to the low levels of software. If we can abstract that process using a higher level programming model where, for example, a user says, this is the network topology I'm interested in. I'm interested in building a social network that has a web server, has this machine learning components, these databases, and then you pass that through a compiler and it produces the end-to-end application. That would remove a lot of bugs from the pipeline because you remove a lot of the human error.

So that's another direction that we're working on. And then I also have a research direction that looks more at the hardware side of how to build data center servers along the lines of what type of computation does it make sense to accelerate. Not so much in terms of specific applications, but more system level tasks like networking, memory operations, garbage collection-- the type of things that you need in a data center for it to operate correctly but are not part of the logic of individual applications. But they can benefit all active applications.

And we also look at what is the right type of accelerator. Do you want the ability to reconfigure that accelerator or a more specialized chip? And how should you interface those accelerators with a main general purpose CPU to make data transfer more efficient? And you can find more information about all of these projects on my website, including links to the GitHub repos for the open source projects.

Great. And they can get through your website by going to csail.mit.edu, correct?

Correct.

Excellent. Well, Christina, thank you very much for your time today. It's been a fascinating conversation.

[MUSIC PLAYING]

If you're interested in learning more about the CSAIL Alliance program and the latest research at CSAIL, please visit our website at cap.csail.mit.edu. And listen to our podcast series on Spotify, Apple Music, or wherever you listen to your podcasts. Tune in next month for a brand new edition of the *CSAIL Alliances* podcast, and stay ahead of the curve.