Steve Lewis: Hello. My name is Steve Lewis. I'm the assistant director of MIT CSAIL Global Strategic Alliance program. I'm here today with Professor Saman Amarasinghe. Professor Amarasignhe leads the Commit compiler research group in MIT's Computer Science & Artificial Intelligence Laboratory (CSAIL), which focuses on programming languages and compilers that maximize application performance on modern computing platforms. He is a world leader in the field of high-performance domain-specific languages for targeted application domains such as image processing, stream computations, and graph analytics and he also pioneered the application of machine learning for compiler optimizations. Prof. Amarasinghe's entrepreneurial activities include founding the company Determina, Inc. (which was acquired by VMWare in 2007 to beef up the security capabilities of hypervisors and virtualized workloads). He also co-founded Lanka Internet Services, Ltd., the first Internet Service Provider in Sri Lanka. Prof. Amarasinghe is the faculty director of MIT Global Startup Labs, whose summer programs in 17 countries have helped to create more than 20 thriving startups. Prof. Amarasinghe received a bachelor's degree in electrical engineering and computer science from Cornell University in 1988, a master's degree and PhD in electrical engineering from Stanford University in 1990 and 1997 respectively, and he joined the MIT faculty as an assistant professor in 1997 and was elected as an ACM fellow in 2019.

Saman Amarasinghe: Thank you for inviting me.

Steve Lewis: So can you just describe in layman's terms? Uh, just in case anybody out there doesn't know what a compiler does or its purpose.

Saman Amarasinghe: Most of the tools, especially compilers, what they do is it is the way that a programmer can instruct a computer, what to do. The same day, you can use a natural language like English to explain to a student what they should be doing in their problem. Set. How do you tell the computer what to do? You use a programming language? Unlike English, they are a lot more precise. There's not much ambiguity in a programming language and what a compiler do. It takes the high level description of a program and map it in a way that it can be run on that computer. So it brings that program from a high level programming language to assembly language. That's what the compiler do in a nutshell. I see. So, you know, the compilers have been around for a long time.

Steve Lewis: Why is there a need for another compiler, a better compiler?

Saman Amarasinghe: So let me explain. When I started doing compilers in late eighties, my good friends who were working on natural language processing. So if you look at something like natural language processing, about 30 years ago and compilers 30 years ago, They had very similar structure so what you do is you take this high-level program, a new do bunch of analysis of that program and keep transforming it slowly from one level to another and slowly lower it to the assembly language in compilers or in natural language processing do slowly lower to a point. You can understand the program. How will, what has happened in last 30 years is a natural language pipeline has no similarity to what was done 30 years ago. Right now it's completely done using machine learning. So the pipeline has completely changed how the compilers, the today's compilers still look very similar to compile it 30 or 30 years ago. So one thing I am pushing hard these days is code. Can we bring compilers up to date with modern systems like machine learning and modern solvers, modern algorithms? How can you make it lot more efficient, lot more reliable, not faster. So that is a big push that I am doing in my research group.

Steve Lewis: I see. And does every computer language need a separate compiler? Is there not one universal compiler, is that what you're driving toward or actually moving away from providing a one, you knew a sector, but in fact, even a one you knew as a language.

Saman Amarasinghe: So the issue is if you try to build something so common, it's not going to be that efficient of that fast. This is what domain, if you.Last 30, 40 years, we have tried to use a single language to roll everything. But what we are finding is if you go to a domain. Whether it be physics, whether it be math processing, whether it go natural language processing or whether it go machine learning, if you create something that is focused on that domain, you can get much better performance. But also, it makes it easier for those domain experts to explain their program in that domain because that language in that domain would be much closer to how they think about their problem. Whereas if have had to map it into C or Java or some common language, you have to do a lot more changes to your thinking before you are able to write it in that programming language. So we are not only trying to provide better performance, but we are also trying to provide better productivity for these domain experts.

Steve Lewis: How would you describe your research vision and some of the bold aspirations of your work?

Saman Amarasinghe: So there are a few things driving my research vision. One is we are at the end of Moore's law. Moore's law had unprecedented impact for the entire industry, in fact, the entire world. And now we are very used to every two years getting twice the amount of compute power. However, that's not happening anymore. So if you want to keep our momentum going in computing, we have to do something different. So I am approaching this from the angle of programming languages and compilers. So one idea that we are really pursuing hard is, there are many domains of computation, like image processing, image compression, signal processing, machine learning, computational biology-- these are the emerging domains that is going to take a huge amount of compute power when you go into the future.

Can we do something that is very specific for these domains that will make it able for them to get much better utilization of current machines, so we can write the Moore's law curve a little bit longer? So I am looking into new programming languages in different domains and how to get really good performance for engineers in this domain. In many of these domains, what you find is there are researchers who want to do research in that domain, from image processing to quantum chromodynamics.

But they have such heavy computational requirements-- they are spending so much of their time not doing research, but writing and optimizing code to get the performance they want.

Steve Lewis: I see.

Saman Amarasinghe: And I want to make it much easier for them. So instead of they are spending 80% of their time writing code, I want them to spend 80% of their time thinking and doing the research, and use domain-specific compilers and high-performance compilers there so they will get the performance they need to do their experiments.

Steve Lewis: Why are you teaching performance engineering?

Saman Amarasinghe: So when I was in college, every undergraduate had to learn at a level that they cared about performance. Because those days, machines were not that fast. So if you wanted to get good performance, you had to care about it. One repercussion of long run of the Moore's law is people got a little bit lazy or distracted about performance.

And so, if you look at an undergraduate coming out of any major university today, there will be great programmers, but a lot of them don't understand how their program is going to get executed and what kind of performance you get out of their program. So they are writing their programs in a very high level abstraction. And what that means is they are leaving a lot of performance in the ground.

So in this class, we are starting showing the matrix multiply, how would you write matrix multiply in Python. And what we are showing at the end is what you write in Python, the simple matrix multiply, versus the best you could get today in the machine could be about 100,000 times slower, the Python code. Think about it.

Wow.

If you have a car that is 100,000 times worse in miles per gallon, that would be almost a criminal offense to have a car like that, OK? The comparison we make is, if you can take a supertanker and make its energy consumption at the level of a scooter, that is 100,000. And that is the kind of performance we are leaving on the floor today because we got so used to Moore's law just giving us free performance.

So if anything is free, you get a little bit lazy and carried away, and you get into gluttony. So what we have done is now we have to train people to go back and think about performance, understand that what they just normally write might not be that performant, and how to understand how to get better performance and go about that.

So it is a kind of a mindset that, if you go through that, even if you're writing a normal piece of code, not something high-performance, you think a little bit performance-wise. You won't do the crazy things that might be a little bit simpler, but you know it will be too slow. So we are pushing people in that direction, and I think a lot of times it's eye-opening for a lot of students to see how much performance they are leaving behind by just using the normal tools that they have been taught, like the high-level languages, like Python and stuff like that.

Steve Lewis: I see. So I mentioned in our intro that you pioneered the application of machine learning for compiler optimizations. Why are you using machine learning to optimize compilers?

Saman Amarasinghe: So if you look at a compiler, what does compilers do? So it takes a high-level programming language and map it into low-level assembly code. But about 80% of the compiler, also about 80% of the compiler code that if you look at the compiler, is a search problem. So what happens is, you have this high-level programming language.

There are billions and billions of ways to generate different assembly sequences to make that program work. And the compiler's task is to find the assembly sequence that will probably be the fastest way to run that piece of code. And that's basically a search problem. And so how did this search problem was done before?

Till now, the state of the art is to write a bunch of heuristics. So some engineers sits down and say, for this piece of code, if you do this, this, this is the best assembler code I can get. So this, this gets written into the compiler. The compiler is full of heuristics. The problem is some of these heuristics are 20 years old. So there are certain things in the compilers, if you look at it, you realize this was probably the best thing to do when the Intel had Pentium, but they are very bad today because compilers are too big.

It's very hard to keep updating these heuristics every time a new machine comes out. This is really sad because companies like Intel spend billions of dollars creating this new microprocessor with a lot of new capabilities, and the compilers are still not taking advantage. They are old-school. They are sitting in this very old transformations written 20 years ago.

Steve Lewis: So they're not using multi-threaded, multicore, any parallelism?

Saman Amarasinghe: Like for example, Intel every generation introduced this SIMD instructions. MMX, SSE, AVX, AVX-512, now basically machine learning instructions, all these things. If you-- that's about 3,000-odd instructions. If you look at something like LLVM compiler, it will only generate about 1,000 of them. The last 2,000 that they introduce, it's never been even automatically generated by the compiler.

So Intel spent billions of dollars adding these instructions. Perhaps few people had assembled some critical kernels using them, but that's it. The compiler can't do that. However, if you can build a compiler that can learn how to optimize, how to generate code, we have shown that it's possible for the compiler to learn to generate these pieces of code.

So what it does is, so we can keep up to date with all these changes happening in architecture, all these changes happening to languages, by learning to do that. So you can be a lot more responsive to changes by building a compiler using machine learning than a compiler that has a lot of human-written heuristics.

Steve Lewis: So how do you train that machine learning model?

Saman Amarasinghe: Actually, it's very interesting. To train machine learning models, you need a lot of data.

Right.

One thing we have in programming is a lot of programming data. You go to GitHub, you can get millions and millions of lines of code, and also generated assembly, there's millions and millions of lines of generated assembly. And to even figure out if they're good or bad, you just have to run it. That's a machine. You can easily run things in there and evaluate.

So kind of the data side is not that difficult. Still there's a lot of work required to basically collect these things, put it into evaluatable benchmark suites, and run through that. But it's not like you need humans to hand-generate or hand-annotate. There's a lot of good data out there that you can use to train a compiler.

Steve Lewis: I see. Interesting. Can you talk about your work with the Global Startup Labs? You were the Faculty Director. Can you tell me about the lab and sort of its mission?

Saman Amarasinghe: So Global Startup Labs got started about 20 years ago. It has this very interesting model. What we do is, every summer, we take two computer science students, two business students, probably Sloan students or some other business students, and send them to a country, mostly a developing country, and they spend about eight weeks in a university or some organization with about 20 to 40 local students or aspiring entrepreneurs training them how to become entrepreneurs, how to start a company, especially a technical company.

This is not about just producing some slides about doing a company. They will actually think about a product idea. And we are doing computer science because most of these products are information technology products, so they will also be the prototype. We will show them how to build a prototype, and they will also create a business model around it, and do presentations, and see how they can launch a company.

So last few years, even in the remote corners of the world, people knew what entrepreneurship was. I mean, they have seen enough news, there's enough internet, that people understand entrepreneurship now. So now it's becoming something-- we go tell them something they know they want to understand. However, when we started, people didn't know what entrepreneurship was.

In fact, when we went the first time to Sri Lanka, it was advertised as, there's a bunch of people from MIT coming to teach you how to do programming. The entrepreneurship side was not even in the advertisement. And a lot of students were really upset that they are told this thing about how to sell products and stuff like that. They did not get it.

But at the end of that session, a bunch of people took it to the heart, and saying, look, instead of going to go get a salaried job, we can actually start creating jobs. So I will tell you a couple of startups. There's this company that

came out of that program called 4 Axis Solutions. It's a company out of Sri Lanka started by four undergraduates who had never left Sri Lanka. They were all local students who were only grown up locally.

Now there is this company that is producing a drawing app on the Apple marketplace that has ranked higher than the Adobe and Microsoft apps. They have multiple million downloads. It's in about 200 different geographic regions in there they are selling this app, and they are making I think a few million dollars revenue at this point. In a country like Sri Lanka, it's a huge amount. They are basically employing about 40 people.

And this is because, when they joined this program, they have no intention of going to go start a company. But they caught that bug of entrepreneurship. They realized this is something they can do. It's possible. And they took it to the heart, and they waited 'till they graduated and started this company.

We have many stories like that across the globe. From Africa, Asia, Latin America, these students go and inspire these young, brilliant students, mainly computer science students, because that's what we are targeting. And also, in these countries, very hard to raise money, and information technology startups are the things that they can do with very little capital. So because of that, that's why we are focusing on that.

So it's a very inspiring thing. I think the students at MIT, they get a lot out of it, too, because a lot of times, when they land in a country, they suddenly become the experts in the area. In fact, I have seen the prime ministers of a country talking to our freshmen and junior graduate students, asking them how they should encourage entrepreneurship in that country.

In fact, I was I was part of this meeting, where-- if you look at MIT, we do a lot of teaching that are basically very interactive, participatory ways of teaching. Many countries has this one-directional teaching in there. And then, actually it's happened in Sri Lanka. A couple of university professors came to see this class, and they were very impressed the way that that interactive teaching is working.

They invited these students to go give a talk to the entire faculty of that university on how to give a good lecture. And imagine a freshman of MIT, who was the instructor, telling a bunch of faculty who are probably 50, 60, 70 years old, how to teach. And actually, she did an amazing job giving an example of a couple of freshman classes and how this, like, interactive teaching methodology worked, why it was really good. And you had a rapt audience of these old faculty listening to this student. So they get this amazing experience, too. And it has worked very well for both the MIT students who go as instructors and also the local participants who come attend these classes.

Steve Lewis: That's wonderful. So you're going to these developing countries. You're inspiring students. You're teaching them, not only on a technical basis, but also sort of the business side of things, about how to start a company. Just giving them the ideas to think bigger, right? To think--

Saman Amarasinghe: It's-- many countries has very hierarchical kind of social structure, so it's very hard for a 19-year-old to think that they can be the boss, and to tell them, give them that confidence. In this arena, it doesn't matter. The fact that a lot of these people that come to these schools might come from poor backgrounds. The fact that you or your parents are not millionaires doesn't matter. That if you have a good idea and if you put a lot of hard work, you can make it work and show them the examples-- it can be very inspiring, and a lot of people take it to heart, and a fair amount have been successful.

Steve Lewis: Are you providing any funding for them, or?

Saman Amarasinghe: We are not providing funding. What we will do is, when a group of MIT people go to a country, it's easy to open doors. So our students will basically have sessions with the students, and sometimes, either one student, or after the session, they will walk into all the top cooperation’s of that country, government

offices-- and basically, we will create an ecosystem. We will basically invite a lot of, like, people, rich individuals, in the country, top business leaders, to basically-- to our classes, to talk to our students, show what they are doing.

And many times, when we leave, they have found sponsors to their work. Because normally, these people might not go and see this, but because MIT is involved, they are excited to see what MIT is doing in their country. And that helps us a lot. So that opens the door, and we will basically highlight these students to the leaders of this country, both industry leaders and sometimes political leaders. And that has helped the students to basically take their project to the next level after we leave.

Steve Lewis: I see. I'm interested to know sort of your opinion on, you know, how would this-- what does the future look like? What should these students be looking at as far as coding?

Saman Amarasinghe: So one thing very interesting these days is, building application and infrastructure is very easy. There's a lot of frameworks, languages, a lot of them are available for free. There's web services, things like AWS or local counterparts for that. So what we are showing them is, unlike good old days, where it took a lot of resources, money, talent needed to get a product out, especially information technology product out, there's a lot of resources available today to do something easy and cheap.

And showing them parts, showing them how to do some of these things, guide them through, giving them examples, actually inspires them. So good old days, when we started, 10, 15 years ago, most of the demonstrations they can do after eight weeks were very minimal. A lot of them were just like simple prototypes, simple website, stuff like that. Today, in eight weeks, people can build fairly sophisticated applications because they're using a lot of frameworks.

And what happens in a lot of these countries, these kind of practical things are not taught in universities. They're a lot more book learning or theoretical material. So one thing our students tried to do is go teach them about these practical, available resources that they can use to build applications and build things that they want to prioritize.

Steve Lewis: I see. [INAUDIBLE] sort of what inspires you in this field. Obviously, working with these young entrepreneurs is a source of inspiration, especially when they have success. Can you tell us about what else inspires you in the field and some of the projects that you're working on?

Saman Amarasinghe: So if somebody told me that-- I had the entire hindsight when I was a graduate student or even undergrad to enter into a field that will become the center of what activity that's happening these days in computer science-- I mean, they would think I was very smart. I feel I'm very lucky that I made some random decisions 30 years ago that made me here.

First of all, computer science is an amazing field. Things that we do today was never even done 10 years, or never even imagined, 10 years ago. It's about imagining the future. And not just imagining the future. We don't write science fiction novels. We make this happen. And being in that, being in the center of that, is really, really, really exciting. That's the first thing in there.

Second, being at CSAIL, it's an amazing place. CSAIL is kind of a center of activity. It's a beehive of activity. Everybody's thinking about some crazy idea. You bump into your colleague, and somebody is basically doing-- using machine learning to figure out the next-generation molecule, or creating this algorithm that you can see through walls or see through obstacles. There's so many things that is even beyond science fiction that I wouldn't be even thinking the science fiction writers without that much imagination that's happening around you.

So that's is really inspiring in there. Another thing that's very inspiring is the students we have. We have some of the most brilliant minds who come through MIT. I mean, every year, we are only able to take about 1% of the people who want to come here. And we are lucky to have some amazing students, and working with them is really

great. So seeing some of these students who came as junior students-- I had one of my students who graduated and became a professor at MIT-- sorry, MIT.

And another student, he became a professor at Stanford. And the third student is going to UIUC. Seeing that these students, who started as graduate students, are now becoming the next-generation leaders, training the next-generation computer scientists-- able to help them achieve that is a really exciting proposition. That's what makes me want to wake up every morning. And these days, I don't come to the office. I come in front of my Zoom camera and talk to everybody.

[CHUCKLES]

That's what motivates me.

Steve Lewis: That's fantastic. That's great. Can you talk about some of your collaborations with CSAIL's Alliance partners, some industry partners?

Saman Amarasinghe: So CSAIL Alliance has been really useful because my research-- I want my research to be useful for people. There's a lot of people who do research because there's some beauty in what they're doing, or they're just pushing a boundary. I want to actually do things that will be useful. What CSAIL Alliance has provided me is people who has the need.

So there are many types-- like, for example, we have been working with one of the alliance members, Toyota, for a while. And we had this project called TACO, which is a Tensor Algebra Compiler. So as good tensor algebra researchers, we build something that works in the C++ language and say we [INAUDIBLE] it. And then, when we talk to Toyota, they say, if we had this working in Python, my researchers would much like to adapt this thing.

And that feedback we got from the CSAIL Alliance. Because they can [INAUDIBLE] it out-- because they have the people with the need, who are using cutting-edge technology, who have the need. And then, when you talk to them, you realize small things you can do toward your research that can have a big impact in real life because they give you the real-life feedback. So I have used that multiple times.

When I have some new ideas and a new thing coming out of research, I love to go and tell the CSAIL Alliance members, here's what we are doing. And many times, they will come and say, this is very similar to one of our needs, but our need might be a little bit different. And that can actually guide the research and inspire the next things we can do. And that has been very useful in getting other CSAIL Alliance members, and that's a really fun thing to be part of CSAIL Alliance.

Steve Lewis: Excellent. And so some of the use cases, perhaps, that you've collaborated, use cases that you've collaborated with industry on, when you mentioned Toyota-- is that for autonomous vehicles? Is that for computer vision? Is that for path-planning? What type of [INAUDIBLE]?

Saman Amarasinghe: So the Toyota project was a lot of experimentation around computer vision requires computing on sparse tensors, so we are working at low-level infrastructure so their researchers at Toyota, who's playing with algorithms, and they need an easy way to implement this algorithm, send their data to see what happen. So we are building this system, and they said, look, if you do these certain few things, it will be very useful for our researchers, so that's we did.

And another project that I am working with, another one of my colleagues, [INAUDIBLE] is we are looking at synthetic data. So there are many companies in CSAIL Alliance that also have problems with basically privacy of data and data security. And we have got a lot of interesting feedback from them, and say, look, if we have synthetic data in certain ways, certain forms, it will really solve some of our immediate needs. So that part is

helping us-- which, in fact, I am working with him to basically create this new startup company to prototype some of it because the feedback we got-- we got a lot of strong feedback that says there might be a use for synthetic data for these organizations. So that inspired us to basically go try to prioritize this technology.

Steve Lewis: OK, well, that's all the time that we have today. I'd like to thank Saman for joining us. And have a great day.

Saman Amarasinghe: Thank you for having me in the show.

[MUSIC PLAYING]

Steve Lewis: 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 Alliance's podcast, and stay ahead of the curve.