SYCL – Empowering Developers in Academia and Enterprise Transcript

Radhika (00:04):

Welcome to Code Together, a discussion series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Radhika Sarin.

Radhika (00:14):

SYCL allows developers the flexibility to use common or customized code for their development. Engineers and developers can embrace common, standards-based cross-architecture code and take advantage of specialized code.

Today we will talk to our guests about SYCL's evolution and its adoption. We will discuss how academia and enterprise are learning to program with SYCL, in addition to what the future holds for this open standard.

Our first guest today is Roberto Di Remigio Eikås.

Roberto holds a PhD in theoretical and computational chemistry. He works as a research engineer at ENCCS, where he helps optimize quantum-chemical simulation software. Roberto is also the co-author of the CMake Cookbook. Welcome Roberto. It's great to have you here.

Roberto (01:34):

Thanks Radhika. Thanks for having me.

Radhika (01:37):

Our next guest is Noah Clemons from Intel.

For twenty years, Noah has had a passion for solving large- and small-scale computational problems. He says, the new oneAPI initiative is the most exciting and challenging work of his career.

Welcome, Noah. Great to have you here.

Noah (01:56):

Thanks so much for inviting me.

Radhika (01:58):

Perfect. Let's get started. So, Noah, why did we need a unified programming language?

Noah (02:05):

Well, the way I see it is that right now in the heterogeneous world, you have many different architectures, each of which are well suited to solve specific types of problems. But if you're used to one architecture and one programming environment for that architecture, all you're going to do is keep trying to solve all problems with that architecture and it leads to some ill-posed solutions to various problems. So, what I really like right now is this SYCL initiative in order to enable people to try lots of different architectures to solve lots of different problems. It's going to lead to a lot better solutions to problems and a much more enjoyable environment from which to program. So, Roberto, what is your experience with SYCL?

Roberto (02:56):

I come from a research academic background. So my experience in general with coding for high performance is that it's something that takes a lot of time and a lot of experience. SYCL really can help with ameliorating this costs because, as you said, you can use the same framework, which is based on an open standard, to target different architectures, different hardware, and this makes it much more portable and it can also ensure that you get a satisfactory performance in these different architectures. So, it frees an academic researcher from the need to spend a lot of time learning the tools to get a performance that can lead to more publications and more exciting science.

Noah (03:42):

Oh yes. So speaking of more exciting science, it sounds like you had a really strong chemistry background. How did your background in chemistry lead to SYCL?

Roberto (03:55):

So I have a PhD in theoretical and computational chemistry and the field has really evolved in parallel with the ability of more capable hardware. So it's a field that is really connected to the need for high performance computing and it really wants to leverage these new architectures. So that's why I started learning SYCL because it's a way to be able to provide this new architectures in a way that doesn't need me to reinvent the wheel and relearn tools over and over again to solve the same problem on different architectures. What about you? I'm very curious.

Noah (04:33):

Well, the way we work as technical consulting engineers at Intel is that we consider ourselves to be optimization experts in the field. So we work with a lot of enterprise and more commercial customers developing software for a certain release during which they need to get either a certain performance number or they need to do more in the same amount of time. So my interest in SYCL is out of professional necessity, is that we're usually a year or two ahead of the trends in order to teach the customers how to effectively get the most out of the architecture they chose. So we're trying to stay ahead of innovation and give the customers what they want a year or two ahead of what they're expecting. So not only was this a personal interest, but it was also a professional necessity for me to get brushed up on this.

Noah (05:30):

So, I know that you're teaching academia right now with SYCL. My group teaches a lot of enterprise and commercial customers. What does your audience want to learn from SYCL and what sort of goals do they have when they approach you in trying to learn this?

Roberto (05:48):

So, our teaching offer is targeted towards researchers. So people with background in academia, or that are still in academia. These people have their own codes or develop their own codes and they're always interested in squeezing as much performance as possible from the algorithms that they've implemented or from the hardware that they have access to. This is because essentially better algorithm or access to better hardware guarantees that you can run larger simulations and that unlocks access to new science, so new publications and basically satisfying interesting academic questions. So when they come to us to learn SYCL, they come to us hoping to be able to find the framework that allows them to be productive in their programming, productive, both from the point of view of getting the performance they want on the hardware they have access to, and also not reinventing the wheel every time that there's a new, exciting architecture that comes out.

Roberto (06:55):

So not spending time doing the part of the same kind of academic code to different architectures but being able to express the concepts once and reuse multiple architectures. Of course, there's a lot of curiosity behind as well because mostly researchers are driven by curiosity and SYCL, at least for me personally, it represents a framework that allows you to express parallelism in a way that feels very natural in kind of conceptual framework. That is very interesting to learn in and of itself.

Roberto (07:34):

So I've told a little bit about how we think about teaching SYCL and what our learners come to get. What about in your case, Noah?

Noah (07:45):

Well, you pose an interesting point about a new architecture comes out and they would like to spend time with it fully ported and not spend most of the time porting it to that new architecture. With the, I wouldn't say transition, but the update and cadence of new and interesting architectures to solve problems, even across all different kinds of vendors, it's coming so fast now that we're really driven by someone coding up the fastest possible solution to see if this new architecture will solve their problem quickly. So they come to us because they want to find out the fastest way to evaluate a hardware that has just come out. So SYCL appears to be the best I've seen into a transition into a heterogeneous computing world, to where if you have multiple architectures sitting on your desk or you have access to them in some sort of cloud environment, you want to get a performance number as soon as possible so you can make a decision. So they pretty much come to our group in order to help them make the fastest possible decision so that they don't spend more time making the decision by the time the next architecture comes out to help solve their problem.

Noah (09:13):

So you were saying that for a lot of the people that you work with, this is not just an experiment and it has real life considerations for implementation and applications that they're running. There's some people who come to you that have had no other experience in targeting multiple architectures, but then there's others who may have had some prior CUDA experience. How does someone with prior CUDA experience relate to SYCL? How does it compare? What are their impressions? Especially for someone who has targeted heterogeneous hardware in the past with CUDA, or maybe even another language.

Roberto (09:48):

So people that come to our workshops to learn about SYCL and don't have previous experience have really good reception of what we teach and how we teach it and what they can get out of the course, because if you familiar with C++, and SYCL is kind of your first, let's say parallel language of framework for expressing parallelism, it feels very natural. I mean, there are high level abstractions that are available within SYCL to express memory allocations and memory movement between different devices and how you would express a computation in a way that maps to how your mental model of parallelism works.

Roberto (10:31):

So for people that do not have previous experience, SYCL is a really good first introduction to parallel programming because it has this high level picture of things that maps to the actual code that you will write. For people that come from a CUDA background, it's a revelation in another way. It is still something that they appreciate a lot, that there is this kind of framework available because it relieves them from making a lot of low level decisions. So you might not be familiar with C++, or you have some familiarity, but have not tended to use these high level abstractions because you were limited by the low level language that you were using to get performance out of the device. Now that you can actually use these abstractions and it is in some ways encouraged by the framework that you're using, then it relieves you from a really large burden in programming. Is this your experience as well with your customers?

Noah (11:30):

Yeah. When it comes to expert programmers, they're always looking for more control, but for a large swath of developers, they're not always looking for that level of control, but they are looking for some level of confidence that if they do rely on a higher level of abstraction, that lower-level optimizations will be taken care of. So, it's both our job as teachers of this standard in the industry, both to teach them on the ins and outs of how to program with SYCL, but there is a performance methodology for people to think about as well. What I like about this is that you can get the performance that you're looking for with a lot less effort than in the past.

Noah (12:19):

So that's why I think most people will find is there is a definite familiarity with C++, and it is easier to learn and adapt to. I would say finding those initial performance results, it is very rewarding for those that are not indoctrinated to a performance methodology with SYCL. So I'd say we're able to capture more novice programmers to reap performance benefits a bit faster than with other solutions we used in the past. That's what I like about it personally and that's what I find that our commercial enterprise programmers that we work with are finding as well.

Roberto (12:58):

Yeah. That's actually really interesting because basically framework that encourages thinking about things that is executing in parallel will help you get the correct answer fast, putting in less effort, because usually you go in stages where you first get the correct answer and then try to make it fast. And then this making it fast usually is really tied to the low-level framework that you're trying to use. That's a really steep learning curve, while with SYCL you can merge two aspects. I think this is large part of the value proposition in SYCL and having an open standard for expressing parallelism across a wide range of devices and hardware.

Noah (13:46):

Yeah, I think in previous years there was a focus on how do you get the last mile of performance? What I'm seeing teaching with SYCL is I'm focused on how do we get the customer's first mile of performance. So if I can get them to a level of seeing some performance gains early on, that's very rewarding for both the programmer that I'm working with and for me personally to see someone get those gains. That's one thing I'm liking about SYCL personally, is the speed to get the first level of good performance seems to be faster than anything I've encountered in the past.

Roberto (14:25):

Exactly. Yes.

Noah (14:27):

But I guess the next thing I would ask you is what have your observations been regarding your teaching experience of SYCL and what sort of learning resources have you utilized? Have you written everything from scratch yourself? Have you relied on any other sort of resources? What is your toolbox that you use in order to teach SYCL to others?

Roberto (14:47):

So we have prepared, together with my colleagues at the Swedish Competence Centre, a website that collects the material that we've written. This is freely available. It's on GitHub so that everyone can access it and use it for self-learning. Of course, this material was not built in a vacuum. There are already other websites that have exercises and examples. One that comes to mind is SYCL Academy GitHub repository, which has valid resources and really valid code examples and exercises. Something that is always on my desk is the open access Data Parallel C++ book, which is an invaluable resource to learn SYCL, in my opinion. Of course, the SYCL standard online, if one needs to dig into the finer points of the definition of the function or a class and so forth from the point of view of actually teaching SYCL and help people learn this exciting new standard.

Roberto (15:51):

We have adopted this kind of philosophy where we are not trying to overwhelm people with very long presentations where we go through a lot of detail, but having short presentation followed by hands-on exercises so that people can get immediately a feel for what it means to write SYCL code. What kind of resources do you use, Noah?

Noah (16:14):

Well, we use some of the resources that you mentioned, but another one is we have our own oneAPI DevCloud that anyone can have access to. We've created a lot of Jupyter* Notebook within there that someone can iteratively not only learn SYCL, but also get a handle of how to get that first mile of performance through a performance-oriented methodology. So a lot of the public resources that are out there, we definitely utilize, but our DevCould is very much a treasure trove that we've created to where someone can start from just a basic knowledge of SYCL and then expand their knowledge to definitely get that first mile of good or even great performance. So I'd say the notebooks that we've built from the dev cloud outward has definitely been one of the cornerstones of our training materials.

Roberto (17:04):

I understand. Yes. Now that you bring the DevCloud in the picture, one thing that I didn't mention is that, of course, we have a lot of examples and hands-on exercises in our training, and we need access to computational resources to run these exercises in a way that it's not just a lab experiment. For that, we have been using supercomputing clusters in and around Europe. So that kind of gives a similar experience to the cloud. Of course, it's not maybe as responsive as with **Intel® DevCloud**, because there you said you have Jupyter Notebook that guide you through an exercise with different prompts and really show this performance-oriented methodology that you were mentioning. So that would be something for us to try and improve in our materials, to bring more of this interactivity into the picture so that people are always engaged in active learning when they attend our lectures.

Radhika (18:00):

Perfect. This has been a great conversation. We are almost out of time. I wanted to thank you, Roberto, for joining us today. It's been a pleasure having you.

Roberto (18:16):

Thanks Radhika, it was really interesting to participate with Noah. Thanks for the opportunity.

Radhika (18:18):

Yeah. Thank you, Noah, for bringing insights from not just the enterprise world, but also from developer standpoints and those related to Intel as well.

Noah (18:27):

Yeah. Thanks for inviting me. It's a very interesting conversation.

Radhika (18:31):

Perfect. Awesome. Thank you. It's been a pleasure and I would love to thank all of our listeners for joining us today. Let's continue the conservation on oneAPI.com.