Episode 1: A Shift to Modern C++ Programming Models
Host: Nicole Huesman, Intel
Guests: Alice Chan, Intel; Hal Finkel, Argonne National Lab

Nicole: Welcome to Code Together, an interview series where we explore the possibilities of cross-architecture development with those who live it. I'm your host, Nicole Huesman, developer and community advocate at Intel. With the rise of data-intensive workloads like artificial intelligence and machine learning, developers have a real need to develop on one hardware platform and target others. And with this need for cross-architecture development comes an exciting evolution in programming languages and compiler technologies. With us today is Alice Chan, vice president and general manager of compiler engineering at Intel, who is going to share her experiences with us. Hi, Alice.

Alice: Hi, it's great to be here.

Nicole: Excellent. We also have with us today Hal Finkel, lead for compiler and programming languages at Argonne National Lab who will share his insights. Hal, welcome to the program.

Hal: Thanks. Nice to be here.

Nicole: Hal, can you talk a little bit about the challenges that are facing developers today?

Hal: Sure. We live in a very interesting time because developers have to target a wide variety of different kinds of hardware. And the hardware is becoming, in some sense, more specialized for different kinds of applications and different kinds of algorithms. In addition to all of that, there are challenges associated with exploiting the hardware, even hardware that is notionally targeted at the kinds of applications or algorithms that a developer is dealing with, and this comes because the hardware has enormous amounts of parallelism that's available. It has unique characteristics that affect the way that data has to be laid out in memory, has to be moved between different parts of the memory. And with all of those different unique characteristics across different devices, application developers can't write many different versions of their application. That's just not practical. And so they have a need for portability. They have a need for that portable code to attain reasonable performance across many different kinds of architectures and yet those architectures have unique characteristics. And so providing developers with a programming environment that lets them write code in an efficient and productive manner and have that code run with high performance across many different kinds of architectures is a challenge. It's a challenge for us on the side of the technology providers. And it's definitely a challenge for developers.

Nicole: Alice, you sit at Intel, can you talk a little bit about Intel's approach to solving some of these challenges.

Alice: Oh yeah. I just want to echo almost everything that Hal said. Intel is really in a unique position. There is very few company in the world that have the variety of hardware that Hal was describing. And when we look at our roadmap moving forward, yeah, there are all kinds of different things from our traditional CPU to accelerators, GPUs, FPGA, all kinds of devices. And from the Intel point of view, we're saying, you know, we need to come up with a way for our users to be able to effectively program towards all these different varieties of hardware and do that with performance, right? And if you look at the market today, there

---

are some proprietary ways in doing so, and there are some cumbersome way to doing so. And we're saying that we need to do better than that. And on top of that, and it's really not just for us, right? It's really for the community itself. What is the best way to write the most effective heterogeneous program across hardware? Not just Intel across all the hardware out there. So we set off to say, besides helping ourselves, we really want to look into this problem and invite the community and the programmer at large to come help define this language or this programming model or new paradigm, whatever you call it, to say, hey, how can we do this better? And that's the start of oneAPI, basically.

Nicole: And the industry has really shifted to modern C++. You both have first-hand experience with this. Why does modern C++ matter?

Hal: Modern C++ matters because that's where the community is. That's where the developers are right now. And it's really the only programming language that across many vendors across different implementations consistently provides a high-performance solution and a portable solution across many different platforms. So when we look at C++, and we look at the community of developers and the sets of features that they demand in order to target the hardware that they need to target, modern C++ is really the only programming language provides that ecosystem that they can use in a consistent and widely available way. In addition, in part because of the wide use of C++, C++ is of course continuing to evolve. And one of the reasons why C++ is a popular language within HPC, is because it's clear that it is evolving in order to meet not only current needs of developers, but also will meet future needs of developers. Modern C++ has a number of modern language features. You know, templates and Lambdas. We're adding networking code routines, parallelism. All of these things are really important in performance-sensitive spaces. And so when we look at the languages that people are using in HPC, there are of course, a large number of them. People still write code in Fortran. People write code in Python, and so on. But most new code being written for HPC applications is being written in modern C++. And that reality is really what we need to reflect when we develop programming models for HPC.

Nicole: Excellent. And Alice, you've really been focused in this area. Can you give us your insights and perspective on this?

Alice: Yeah, I agree with everything that Hal said again. It is a very powerful language that gave us the performance that is needed, especially in HPC space and many other areas also. So at Intel, we have been a longstanding member of the ISO standard C++ community. We recognize the language, and especially with the new development, all the modern features is really pushing the language forward. However, a while back, we actually recognize the need of adding threading, parallelism and such things into the language, or at least help the language. Many, many years ago we started a product called Threading Building Blocks (TBB) to help in threading, it's a template library to sit on top of C++. So a lot of people are very familiar with it and it's pretty powerful. But then we look at it and say, you know, we need to do a little bit more to the standard part of it. And we look at the STL, the standard template libraries, and we say we would like to introduce parallelism, or at least push more of it onto

---

it. And so we started to at least participate in the development of parallel STL, not only for threading, but for vector authorization also. And as Hal said, we have a lot of opportunities and the new C++ 20 standard have more features that are coming into this area. And we think that this is a really good foundation for some of the things that we would like to do in the future.

Nicole: Excellent. And can you talk about how DPC++ really helps support and advance modern C++?

Alice: Right. So, DPC++ for us is C++. It has specifications that do a little bit more, but it is sitting on top of a specification called SYCL and C++ and both SYCL and C++ is open standard. One is supported by the Khronos community and one of course ISO community behind that. And Khronos also is the organization who introduce OpenCL to the world and OpenCL was supposed to be a portable standard across different hardware. Very similar to the idea that what we want DPC++ for us to be doing. However, you know, it's not really getting the traction that is needed. And that's when DPC++ got developed. And it's abstracted at a higher level. But also it has C++ as a foundation. So it provides a lot of templates and specifications to really help us to write heterogeneous programming. And what we're trying to do, as I said before, is to really move this into the community and introduce that to not just Intel, but across all the developers. And why DPC++? Why don't we do SYCL instead? What we tried to do is really get a little bit ahead of the standard. Invite the community to come together, write the language spec, try it out, introduce an open source implementation. And if that works out, and if we are all confident with it, we will absolutely try to get that to the next SYCL standard. And it is actually happening with the next one. And eventually, we would absolutely want that to be part of ISO C++, and it's a long goal, but DPC++ was to enable us to do this in the long run.

Nicole: Excellent. Hal, what has your experience been with DPC++?

Hal: Our experience with DPC++ has been very positive. When we first started working with Intel on programming models for future systems, and we've been doing this for quite some time, we looked at SYCL and we looked at many of the other alternatives that are out there. And you know, the value of having this kind of interaction is that we can look at the various options and we can say that, you know, these various features are very good and there are some gaps that we need to address. And Intel has worked on DPC++ and addressed those gaps that we had identified. So when we looked at SYCL, we really liked the fact that it was a modern C++-based programming model that would integrate well with applications and libraries that were written in modern C++. And as I mentioned, we have more code being written in modern C++ than in any other language for HPC, for performance-sensitive applications. And so that's an important point of integration for us. But there were things that SYCL needed to address and we address those in DPC++. And in addition, those features that have been developed on top of SYCL for DPC++, such as the ability to have nameless Lambdas and reductions and so on, are things that we are now talking to the SYCL standards committee itself about incorporating.

Nicole: Excellent. I think you've both talked about really collaborating with the community. Can you talk about why community-driven collaboration is so important?

Hal: Community-driven collaboration in this space is really essential to getting wide adoption. And the community-driven collaboration really exists on several levels. So there's collaboration on the high level design that encompasses both Intel users such as ourselves at Argonne National Laboratory and within the Department of Energy, and a whole wide variety of people who have been involved in this process. And that's really essential because you know, like in any other case, no one person or small group of people think of everything and it's very important to reach out and get a wide set of perspectives, when you're trying to build a model on which many, many millions of people are going to end up writing applications. So that kind of wide involvement and outreach has been essential to making sure that the wide variety of different use cases can be represented within this model.

But there's another level on which this community engagement has been really essential. And that's on the implementation level. So DPC++, the implementation that Intel has created, is open source. And because it's open source, the community of implementers has been able to build on that, to create implementations for a variety of different platforms. The DPC++ implementation already supports more than just Intel hardware. There's support for NVIDIA's hardware in addition, and other hardware support is going to be added over time and it's being integrated with the LLVM and Clang projects, which are large community-driven projects that support many different kinds of hardware and many different kinds of platforms. Those projects support many different kinds of tooling to get built on top of those compilers—static analysis tools and debuggers and all sorts of things that people rely upon to have productive programming environments and development experiences. And so, these two layers really interact in the sense that, when you have a programming model and environment that's available across many different architectures, it makes it much more appealing for developers to invest time in developing code for the programming environment because they know that they'll be able to reuse that investment over a long period of time in many different circumstances and targeting many different platforms. So the community engagement here is really essential on many different levels, and that's been an important aspect to us in driving the adoption.

Nicole: Fantastic. Thank you. And Alice, from your perspective, why is community-driven collaboration or innovation so important?

Alice: Well, one of the things that we are doing is to make sure that that can happen—that collaboration that Hal was describing. We actually have set up a technical advisory board. And the purpose of this organization, or this team or committee, is to provide governance to put a real standard together, incorporating the needs and the requirements from different hardware, different segments of applications, and making sure that we have community input to have a language that actually can stand on its own and improve adoption. And with this advisory board, [the effort] is actually going beyond the language itself. It has the well-defined APIs on the lower-level layers, APIs for libraries that other programmers or

developers will be interested in. And it is a big ecosystem that we're putting together to help program this diverse class of hardware and diverse class of applications that we are talking about. So, this is what we are doing and it's pretty exciting actually. And, this advisory board is comprised not only of companies from the industry, but there are also national labs like where Hal is coming from, and we also have invited a number of academia researchers to join us so that we have a wide view of requirements and what is needed to be done.

Nicole: Excellent—so bringing together a diversity of viewpoints to really drive this forward. Great. Can you share your vision with us for the future? What are you most excited about?

Alice: Well, we're changing the world in some way. It's really changing the way how we can do heterogeneous programming moving forward. Right now and up until now, a lot of things are very limited and fragmented. The vision is really pull everybody together to have a more effective way in doing this and also bring performance. So this is really exciting, exciting for Intel and exciting for the industry. That's, in my opinion.

Nicole: Excellent things. And Hal, from your perspective, what are you most looking forward to as you look forward?

Hal: Well, I'm looking forward to the development of this portable performance, and well-integrated ecosystem that has many different people involved. We have not only involvement in the implementations, but also in the programming and environment itself. We have involvement in DPC++, and the experience that we gained from creating DPC++. It's going to feed into what the standard C++ ends up supporting. And importantly, how it supports it. So, you know, long term we are looking at C++ as a language that can address these kinds of use cases that require heterogeneous computing and HPC hardware and all sorts of other things that exist in performance-sensitive spaces. But that requires experience, and this is a really good base to attain that experience. In addition, we're going to see a lot of applications and libraries built on top of DPC++ that provide productive abstractions for different kinds of hardware platforms. Within the DOE, we developed Kokkos and RAJA and several components that provide these kinds of abstraction layers, which will integrate with DPC++. And overall, I am really looking forward to providing our users with a programming environment, which is relatively easy to use and provides good performance across many different architectures.

Nicole: Fantastic. One final question for those interested in trying out DPC++ and oneAPI, what would you suggest as a few steps to get started?

Alice: If anyone is interested in this work, Intel has started their beta program. So it is ready for download and can be tried out. But more importantly, if you go to oneapi.com, you can find all the information about the project, about the open source implementation, about the specifications that we're talking about, and about this technical advisory board, this governance that we're putting together for the standard. So there's a lot more information that can be found there if you are interested.

Episode 1: A Shift to Modern C++ Programming Models
Host: Nicole Huesman, Intel
Guests: Alice Chan, Intel; Hal Finkel, Argonne National Lab

Nicole: Excellent. Alice and Hal, thanks so much for joining us on the program today. It's so exciting to see when a community, when the industry comes together, the kind of innovation that happens. So thanks for being here with us.

Alice: Thanks for having me.

Hal: Thanks for inviting me. It's been great.

Nicole: And thanks to all of you out there for joining us. Join the conversation at oneapi.com and until next time, thanks for listening.