Episode 6: Why oneAPI?
Host: Nicole Huesman, Intel; Moderator: Sanjiv Shah, Intel
Guests: Andrew Richards, Codeplay Software; Hal Finkel, Argonne National Lab

---

Nicole Huesman: Welcome to *Code Together*, an interview series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Nicole Huesman.

Hey, Nicole here. Thanks for joining us again. I just wanted to take a moment to let you know what's in store today on a very special episode of *Code Together*. In a year of what seems like never-ending firsts, Intel participated in the first digital International Supercomputing Conference. As part of our participation, Sanjiv Shah, Intel GM and VP of Developer Software Engineering, moderated a chat between previous *Code Together* guests, Andrew Richards of Codeplay and Hal Finkel at Argonne National Lab. Let's listen in.

Sanjiv Shah: Hi, I'm Sanjiv Shah of Intel, and I'm in charge of the software developer tools at Intel Corporation. And I'm here to welcome you to the oneAPI industry initiative fireside chat. And I have with me Andrew Richards of Codeplay Software and Hal Finkel from Argonne National Lab. Andrew, do you want to introduce yourself, and Hal?

Andrew Richards: Yes, I'll start. So, my name's Andrew. I run a company called Codeplay, based here in Edinburgh in Scotland, and we were originally supposed to be a video games company. So we did developer tools for graphics processors. And I think it was always the plan that you could do other things with graphics processors so they're incredibly powerful, but I am absolutely stunned that people want to do things like drive cars with graphics processors, run supercomputers with graphics processors, and so we kind of rode the wave and ended up working in this field of AI and supercomputers.

Hal Finkel: Great, and I'm Hal Finkel. I lead the compiler technology and programming languages team at Argonne's Leadership Computing Facility. I'm also the vice chair of the C++ Standards Committee. And I'm really excited about the developments in the programming model space, and as Andrew said, we are excited to have future supercomputers that are driven by GPUs.

Sanjiv Shah: Oh, well welcome, Andrew and Hal. So, let's talk about oneAPI. So, oneAPI is designed to be a cross-industry, standards-based, open programming model for all architectures. There has been a proliferation of accelerator architectures coming in, and the need for a cross-industry standard is very high at this point. oneAPI promises to offer a portable model that is performant, that is productive, and that lets you write very maintainable code across your different architectures. We're intending to have oneAPI code developed with the hardware and software community, and we're working very openly, both on a specification for oneAPI and on liberally licensed open source implementations. Data Parallel C++ is the central element of oneAPI, and it's based on ISO C++ and the Khronos Group SYCL standards. We've made a lot of progress on oneAPI since the announcement last C19 with implementations from Intel and Nvidia already available. Argonne and Codeplay have been our core travelers, and oneAPI is going to be the primary program, the model for programming Aurora, the first U.S. exascale computer. Hal and Andrew, would you share your experiences in engaging with the oneAPI ecosystem?

Episode 6: Why oneAPI?
Host: Nicole Huesman, Intel; Moderator: Sanjiv Shah, Intel
Guests: Andrew Richards, Codeplay Software; Hal Finkel, Argonne National Lab

---

Andrew Richards: Obviously the way we came from, uh, Codeplay is because we were doing programming models, we led the development of the SYCL programming model. So that's the industry standard that's an important part of oneAPI, provides the C++ programmability of oneAPI. Since then, our main involvement in oneAPI is we've been working to get oneAPI running on top of Nvidia GPUs. So, if somebody is using an Nvidia GPU today, what we've done is we've committed to the open source community a version of oneAPI that will run on Nvidia GPUs, so it will give you all of the performance that you can get out of an Nvidia GPU. We benchmark it regularly against CUDA, but it also gives you that industry-standard programming model, and it's open source so you can add your own capabilities to it as well.

Hal Finkel: So, our experience with oneAPI started, you know, a while ago and we were engaging with Intel around our future supercomputer, Aurora. And from my perspective, I'm really excited to see this programming model, which is based on modern C++, which can integrate well with the frameworks and libraries and applications, which are being developed now. And, you know, even for applications that are, say, still primarily in Fortran, we're seeing a push towards taking the performance-critical parts of those applications and moving them into C++. And either from there they could use oneAPI directly, or they can use a framework like Kokkos or Raja that sit on top of oneAPI, but all of these things are written in modern C++, and, you know, we need a programming environment that integrates well into that kind of ecosystem, and oneAPI and DPC++ are really important steps in that direction.

Sanjiv Shah: You're both members of the technical advisory board that Intel has set up, particularly for the Data Parallel C++ language. How have interactions been in that body, and is oneAPI really living up to this promise of openness?

Hal Finkel: Well, I mean, it is from my perspective. I think that having a technical advisory board is really important. And of course that implies that it includes people who are, you know, outside of the community of core developers of the software, the specification. And the technical advisory board for oneAPI certainly meets that criteria. And I think that the interactions on the phone calls have been really good. There's been a lot of feedback from people who have experience with other kinds of programming models and developing other C++ applications. And in addition, you know, that feedback is incorporated, but, you know, more than that, the notes from the technical advisory board calls are online, and, you know, there's an increasing amount of visibility into how the specification is being evolved. And I expect that as the adoption of the programming model increases, there'll be an increased interest in the workings of the technical advisory board and the feedback that's being provided. And I expect this to grow and I think this will be a very good thing.

Sanjiv Shah: And Andrew, anything from your perspective?

Andrew Richards: Yeah, yeah, no, I, I think the openness is really good. I'm sure we're going to come back and talk about this, but, uh, you know, one of the big challenges in having a committee is it then slows you down. So sometimes you need to be really cutting edge and very responsive to users and try something very new and solve a problem very quickly. And

sometimes you have a much longer time period where you have to really standardize things across the whole industry and get more people involved. And what's really great about oneAPI is it fits in its place and integrates with and connects with other standards. So, oneAPI can be very, very responsive to the needs of people, like Hal now, who needs to get a bunch of stuff running on a supercomputer now. But at the same time oneAPI also feeds back into other standards bodies, like it uses the SYCL standard, the C++ standard, it gives you integration into things like OpenMP. It also uses the SPIR-V standard, which is a compiler standard that's so good from Khronos. And so, I think that's what's really good about oneAPI, is it has its place, and I know that the vision of oneAPI says it does everything for everyone, you run every bit of software everywhere, but it's also very programmatic in the sense that oneAPI understands that it's part of a wider ecosystem of technology. And it solves the problems that it needs to solve, but it doesn't reinvent the wheel when it doesn't need to, and so it integrates well with everything else. I think that's really important for those of us actually building real technology with that.

Sanjiv Shah: So, you've been deeply involved, Andrew, with the Khronos SYCL body. How have interactions been with the SYCL committee and with the oneAPI Data Parallel C++ been?

Andrew Richards: Yeah, it's been really good. I suppose, to try and put all these different things in their place, right, SYCL is designed as a C++ programming model for accelerators. So, it's kind of specifically there for these parallel accelerators, like a GPU or an FPGA or a DSP or something like that. And it's only a programming model. It tries to learn from and feed back into ISO C++, which is much slower. ISO C++ managed to solve all of the problems with C++, the CPUs, the new programmers, advanced programmers, all of those different challenges where SYCL is just focused on the programming model. If you want to build really fast software that's going to run fast on a GPU and an FPGA and a CPU, you need the programming model so you can write your own code, but you also need highly optimized libraries that can do specific things. So if you want to do a matrix multiplier operation, for example, SYCL allows you to write your own matrix multiplier operation. But what it doesn't do is actually work out exactly the right algorithms to run on each processor. Whereas oneAPI does provide that, it actually provides you matrix multipliers, convolutions, all sorts of standard, high-performance software libraries. It provides you that, but isn't part of the SYCL spec, and it provides you that on top. But it all fits together, and I think that's what's been really good. It also has been closer to the bleeding edge, so it is moving faster than SYCL, but we have seen a really good flow through of ideas from DPC++ is the programming model of oneAPI. And those ideas feed back into SYCL, and it takes a bit longer to get them in SYCL because it's more of a committee, and then we're going to be feeding those ideas back into ISO C++. I think that works really well.

Sanjiv Shah: That's great to hear. Hal, and you've got a huge community of scientists and users off your computers. Do you have any early experience with oneAPI?

Episode 6: Why oneAPI?
Host: Nicole Huesman, Intel; Moderator: Sanjiv Shah, Intel
Guests: Andrew Richards, Codeplay Software; Hal Finkel, Argonne National Lab

---

Hal Finkel: So, we do have early experience with oneAPI. As part of our preparation activities for Aurora, we have several different programs that are designed in order to get early adoption, early users, onto our systems, interact with the programming models for the new computers, port applications, and so on. So as part of these programs, some of which fall under our early science program, some of which are activities that are just driven internally by having our own staff go and port proxy apps and benchmarks and other things, you know, we have started gaining experience with oneAPI. In addition, we are developing at Argonne the backend implementations for the Raja and Kokkos abstraction layers. And those backends sit on top of SYCL and oneAPI. So, you know, we have a lot of experience now developing with oneAPI, and based on that experience, we've been able to provide feedback to Intel, and we've been able to sort of develop our own intuition about, you know, how things can work well and where we would like to see things go in the future. And I just wanted to add something to Andrew's comments earlier, which is the committee processes work really well when you come to the committee with implementation experience for particular things. This is true for C++, and it's true for SYCL and so on. And so, the development work that's going on for oneAPI, which includes a lot of things which are really important for our users as reinforced by our experience, doing all of this preparatory work, things like support for unified memory and reductions, and so on. These things we can then bring to the SYCL standards body with implementation experience. And that's really the best way of evolving a standard is when you come from a position where you have an implementation that's in use for your production-level workloads, and you have a good understanding of the trade-offs involved, and then you can bring that to the standards committee and that can feed that discussion in a really productive way. So all of this ties together for me, and I, I think that's all important.

Sanjiv Shah: Yeah, excellent point. Yeah, I, I remember standards bodies going through these iterations where they basically just allow addition of new features without implementation experience. So that's an excellent point. Let me try to end with asking each of you, what is the best thing about oneAPI from your perspective? Then what is one thing that you'd like to change about oneAPI? So Andrew, let's start with you.

Andrew Richards: Yeah, so I think the best thing is the fact that it integrates different technologies together. I think, uh, you know, the one thing that I found applying these technologies in the real world is the difficulty of integration. And you can imagine a solution that does something really fast, and what you find is you integrate it with something else and all of your performance benefit disappears because the cost of moving the data or whatever, or transforming the data between two different systems. It's just so many systems break down like that. So, the challenge of integrating it is actually really tough. It's sort of, that makes the difference between, you know, actual production quality systems versus some interesting research projects. And oneAPI does that really well.

I think that the challenge in oneAPI is that if you take all of this experience and accelerated programming models, and how you get performance on lots of different systems, you do end up with something quite complicated. And oneAPI is therefore quite complicated for

somebody coming to it from outside, and that's necessary. That's just necessary because that's just a factor of all of the optimizations and all the integrations of the different technologies that people do it. You know, it's a good thing that it's like that. But I do think that the challenge that we have is giving people an easy route in, and one of the really big things that I've found is giving people an easy route in if they're from a processor background. If our future is lots of people designing different special-purpose processors for different things, and if the vision for oneAPI is it's not just one company, it's lots of companies and researchers and everyone working together on something, then I think we need to find an easier route in, or easier way of explaining it for people who've designed some brilliant new processor.

Sanjiv Shah: Excellent. Hal?

Hal Finkel: So, I think that oneAPI is obviously going to be very important for us and for our users. And, you know, as I said, I'm really excited about it because it really represents the cutting edge and where programming models are for heterogeneous computing.

I'll highlight two challenges. One challenge, which I think echoes something that Andrew talked about is that modern C++, which of course is the underpinnings for SYCL and oneAPI, is still challenging for a lot of our user community. Most of our user community are not people who have formal training in computer science, but rather are scientists who have learned to write computer programs. And so, you know, the extent to which they can pick up modern C++ and use it effectively varies widely. There are, some of our users who are really, really excited about modern C++ and use it very effectively, and there are other users who find that more challenging. And so, you know, that education process is going to be a challenge for us, even though I think it's definitely the right direction, and we're very excited to see it. A second challenge is the support for systems that have multiple GPUs and multiple kinds of accelerators. This is an area where I think oneAPI and SYCL itself and C++ are all going to have to grow as we further embrace the reality of heterogeneous computing. And that is that heterogeneous computing does not just mean one CPU and one accelerator, but, you know, the systems we're seeing have multiple different kinds of devices in some case even, but even if you're not going to that extreme, we certainly see systems even in the field today that have CPUs and then multiple GPUs. And the challenge is going to be—and this is a challenge across the entire programming model space, oneAPI is no exception in this regard, but also not unusual—is that we're going to have to see how we evolve programming models that really embrace this heterogeneous future.

Sanjiv Shah: Okay, very good.

Andrew Richards: Hey, can I jump in and ask Hal, do you have an idea yet of whether you're able to get the kind of performance you want out of these programming models? 'Cause I think that's the question people have, and they're very worried about this modern C++ style that may look nice, but does it go fast? And I get some great results that I can't often talk about, but do you have results?

Episode 6: Why oneAPI?
Host: Nicole Huesman, Intel; Moderator: Sanjiv Shah, Intel
Guests: Andrew Richards, Codeplay Software; Hal Finkel, Argonne National Lab

---

Hal Finkel: We do have some results, and, you know, we're getting more. We are not all that concerned about the performance overhead introduced by modern C++ features, at least not in the way that we are using the programming model. The compilers seem to be effective at doing enough inlining and other kinds of optimizations, so that we're not seeing a lot of particular overheads there. We do have, you know, some challenges with programming model, you know, economy, in some sense, when we compare to models like CUDA, for example, that have a lot of kind of built in magic variables and so on. The oneAPI and SYCL models are much cleaner, but, you know, sometimes the magic variables are convenient, and they're convenient until they're not convenient. And of course, that's always the challenge of reality. But, you know, I think that as we move forward, I really see us overcoming all of these kinds of things. The one place where we have seen, you know, potential performance-related implications to the model, is really been in comparing to OpenCL, for what it's worth. You know, most of our users don't use OpenCL. We have very few OpenCL applications in our HPC ecosystem, but the ones that exist sometimes get performance because they take advantage of the just-in-time compilation capabilities of the OpenCL stack, which you don't necessarily get in the same way with SYCL and oneAPI and DPC++, and so on. Taking advantage of it in OpenCL, of course, it's kind of ugly because it involves having the program create, you know, strings of code and then passing them into the compiler, and so on. But nevertheless, it does have its advantages in places, but that's really the one place that we've seen where, you know, we've looked at DPC++ and said, "It's not really clear how to exactly replicate that within the model." For everything else, I think we've been really satisfied with what we've seen as far as performance goes.

Sanjiv Shah: Well, thank you, Andrew and Hal. I really appreciate your thoughts on areas where we can improve, and I look forward to working with you guys and the technical advisory board as we evolve the specification. We're at version 0.8 of the spec, we're headed toward a 1.0 release sometime in the September, October timeframe. I'm really looking forward to evolving, to address these challenges you've pointed out, and work together and make oneAPI a success. Thanks very much.

Andrew Richards: Thank you, Sanjiv. We always like working on very ambitious projects, and this is a great, ambitious project to be involved in.

Hal Finkel: Yeah, and thank you. And I certainly second that we also live for our ambitious projects, and we've been quite happy on this road.

Sanjiv Shah: Okay. Thank you, gentlemen.

Nicole Huesman: Thank you for listening to another episode of *Code Together*. Let's continue the conversation at oneAPI.com. See you next time.