# Intel oneAPI 2023 Toolkits and Codeplay Software – New Plug-In Support for NVIDIA and AMD GPUs

**Tony** [00:00:04] Welcome to Code Together, a podcast for developers by developers where we discuss technology and trends in industry.

**Tony** [00:00:11] I'm your host, Tony Mongkolsmai

**Tony** [00:00:17] Over the past few months, you've heard us talk to various people on this podcast about oneAPI and how it's being used in supercomputers for A.I. and in consumer applications like Blender. You've heard myself and our guest talk about how oneAPI is, among other things, an open multi architecture, multi-vendor programing model. Today we have an exciting announcement related to the multi-vendor part of that description. To talk about what it means for you, the developer, I'm joined by two engineers, Gordon Brown and James Reinders. Gordon Brown is product owner for the one API core team at Codeplay and has been there for nearly ten years. Welcome to the podcast, Gordon.

**Gordon** [00:00:50] Hi. Thanks for having me.

**Tony** [00:00:52] James Reinders is a technical evangelist and engineer at Intel who has authored ten technical books about high performance programing. Welcome back, James.

**James** [00:01:02] I'm happy to be here.

**Tony** [00:01:03] So today we have a big announcement, and I guess I'll go to James because James actually wrote a blog about this, which we will also link. James, won't you tell us what our big announcement is today around oneAPI and Codeplay?

**James** [00:01:15] Well, it's our annual release of Intel oneAPI tools. And so each year we do, you know, a major release gives us an opportunity to do bigger upgrades than we do during the quarterly releases and introduce new, new and exciting functionality. So we've got fantastic support for the latest Intel hardware on strong support for standards, including using the LLVM to get awesome support for C++, SYCL and FORTRAN, great AI capabilities, optimization. But my favorite of all of this is we've been talking, as you said, about multi architecture, multi-vendor support for a long time and we've had bits and pieces of that in open source and Codeplay has been doing an incredible job of doing some of that support for the industry. We bring that together today. We've opened up our tools to the ability for plug ins to be added to them, and Codeplay is following through with some exciting plug in capabilities that really help us deliver on that multi architecture, multi-vendor vision that we've been talking so much about. So to me, that's just amazing and I encourage people to take a look and give us feedback.

**Tony** [00:02:45] Yeah. And so Gordon's been working on this on the Codeplay side of things. So Gordon, why don't you tell us a little bit about what the plugin actually means from the Codeplay side of things and what can developers actually do with this plugin?

**Gordon** [00:02:57] So the product that's been released from the site as the oneAPI for NVIDIA GPUs and oneAPI for AMD GPUs. And as James said, the latest release of the Intel oneAPI toolkits now opens up the possibility for other plugins and effectively to add support for further architectures in addition to the Intel architectures. So these these new plugins from Codeplay will add support for NVIDIA and AMD GPUs to the existing DPC++, C++ compiler.

**Tony** [00:03:36] And what are the requirements then to actually make this work? So we've talked about we have oneAPI 2023, which is our latest release which should go out today. And then what do I need to actually go get from Codeplay? Is it just a small download? What does that look like from your side?

**Gordon** [00:03:53] So the way this this works is that Codeplay is just distributing the oneAPI for NVIDIA and AMD GPUs which are plugins which are supplementary to the Intel oneAPI toolkit download. So the Intel toolkit is required at first and then the the Codeplay plugins add to that by installing in addition to the compiler, the NVIDIA and AMD plugins. So these are these are designed to work together.

**Tony** [00:04:25] Awesome. And are there specific requirements in terms of the the types of NVIDIA software stack or AMD software stack that we support?

**Gordon** [00:04:35] So there are specific platforms and devices that we have tested as part of this release and these are described in the release documentation on the website. However, generally, the NVIDIA GPUs support supports everything from SM35 upwards and the AMD GPU support should support and generally any architectures which support the ROCm platform.

**James** [00:05:11] One thing that that's notable that I saw just this week before the release was, you know, there's always a matrix of what it supports and doesn't support. But I know there was recently a new introduction of a new SDK for CUDA and the team was working quickly to make sure that that was compatible and take care of any adjustments necessary. So I know the teams are eager for feedback. If there are any platforms that don't work seamlessly, the objective here is to make it work. And since this is based on open source support that, you know, really goes back to helping the community support these devices the best.

**Gordon** [00:05:53] Yeah thanks thanks for pointing that out. Generally, we aim to support the latest versions of CUDA and ROCm kind of as soon as possible after they're they're released. So yeah, we're always keen to kind of keep the AMD and NVIDIA support as latest as possible.

**Tony** [00:06:14] And I think one of the key things that our developer listeners would probably be interested in is how is this different from our previous open source offering? So in the past you were able to actually go take the open source code and build the Intel DPC++ compiler and build and run on NVIDIA or AMD hardware. How is what we're talking about today different than that?

**Gordon** [00:06:35] So that's a really good question. It's still possible to take the open source Intel LLVM project and build DPC++ with NVIDIA and HIP support it manually. However, this release is the first binary release built from the Open Source Project that is built and tested and verified with the binary Intel oneAPI release. The other thing is that as along with the these these products Codeplay will be introducing user and customer support that users can sign up for.

**Tony** [00:07:11] Okay. So the key differences are it's something that's already been built and tested and also there support available for customers rather than just being open source and kind of letting people try it and submit bugs as they see them.

**Gordon** [00:07:25] That's right, yeah.

**Tony** [00:07:26] So it's part of the workflow when you're trying to build code that's going to run on both Intel, NVIDIA and AMD platforms. Do I actually get a fat binary? Do I build individual binaries for each platform? Do I have the choice? If I can build them all together, can I actually choose at runtime which GPU to use? So for instance, if I have multiple GPUs in my system like I do in my test system, what does that look like for the developer? How do they interact with the development platform?

**Gordon** [00:07:56] So that's a really, really good question. So DPC++ has been sort of designed from ground up to be multi vendor, multi architecture to provide the kind of performance portability that you want from programing with SYCL. When you're using the DPC++ C++ compiler, you can specify the different triples that you want to target, and you can specify more than once. You can target multiple architectures at once.

**Gordon** [00:08:21] And what this will do is it will generate the resulting binary executable with both, you know, the host code and the device code for each of the architectures you're targeting. And then from there, then that binary is then capable of running on all those different architectures. The other side of the of that is at runtime, you have to choose the device that you want to execute on. And whether or not you're able to execute certain devices will depend on whether you've compiled for those architectures and whether the devices available in the system. Generally for this use the SYCL device selector feature, which is a feature of the SYCL programing model where you can describe heuristics for choosing the device you want to target. So you could see any GPU or specifically NVIDIA GPU or AMD CPU or Intel CPU.

**Tony** [00:09:12] Yeah, that's going to be a lot of fun.

**James** [00:09:14] I think users of other systems like CUDA and OpenCL have seen fat binaries capabilities grow on the system in the sense that SYCL allows to be implemented, that the the LLVM compiler called DPC++ implements can allow incredibly fat binaries, but that's really about flexibility. It's about the ability to compile ahead of time to be ready for various targets. You can make decisions about how much compilation happens at compile time, how much happens at runtime. With the JIT, you have a lot of control, so obviously a fat binary takes up space on the disk and so forth, but at runtime the system actually goes and grabs, you know, what matches your hardware and gives you access to that as a programmer. So incredibly flexible system. The way that it's implemented, I think it's very exciting and worth people taking a look at and taking advantage of.

**Tony** [00:10:13] Yeah. And we mentioned that the oneAPI for AMD release is a beta release, but we don't have that beta tag on the oneAPI for NVIDIA Codeplay release. How are we differentiating? What's the difference there? If I'm a developer when I'm looking at a beta versus a non beta for us.

**Gordon** [00:10:28] So the main difference there is sort of the level of support for the SYCL programing model so that the CUDA and NVIDIA GPU support through the CUDA back end has has been in development for quite a bit longer. And at this stage it's as close to feature complete and the performance we're seeing with that is in in many cases very good kind of comparable with with native CUDA. Whereas the AMD GPU support through the HIP back end is only been development for the last year or so. So it's currently approximately sort of 50% support, which means it has of all the core cycle features. So certain applications and benchmarks will run, but it's missing some some other more

advanced features that are still to be to be implemented for the applications we have run. We are seeing good performance relative to native HIP, but there's there's still things that are missing. So users should be aware, you know, there's there's certain features that might not work yet or they may not see the performance they'd hoped for with that yet.

**Tony** [00:11:34] Yeah, we're kind of adopting a model here where we also we're not just having people pay for support, but we also have a kind of an open model. So when people see issues like that, how can they get involved? And if developers are interested in contributing to this, how can they get involved?

**Gordon** [00:11:51] All of the work that Codeplay is doing and Intel as well is in the open. It's on the Intel LLVM GitHub project, so we work directly with that. That's already quite an active project. There's a lot of tickets that come in. So, you know, if anyone has any bug reports or performance issues or any kind of feature requests, they can come in through there and we'll we'll look at them and and try to work on those.

**Tony** [00:12:18] All right. And then looking forward, I guess I'm going to direct this question a little bit to James. So James has been a very vocal proponent of all of these things. And you can kind of hear his passion as he's talked about where we've come from. James, what are you excited about where we're going? Give us an idea of what do you think this is going to lead to in the future and why it matters?

**James** [00:12:38] I think one of the characteristics of where computing is going is that we're going to see more and more diverse compute devices on one machine. And it's not unusual to have a machine with, say, an NVIDIA graphics card plugged in to do your gaming on or some of your computation. But you also have, you know, a wonderful Intel CPU with capabilities, maybe including AVX or AVX-512 and integrated graphics. And each one of these is a compute device and that's just the beginning.

**James** [00:13:07] So we're seeing more and more diversity. And the way tooling has been headed is that tooling is usually very good at taking advantage of one or two of those at a time. But with the diversity coming, we really need solutions that bring together the ability to get the best out of each piece. And so, you know, we've been talking about oneAPI, the SYCL standard has been wonderful, you know, even OpenCL and OpenMP. There's been, you know, people of like mind thinking, gee, we really need to support this abundance of compute that we have coming towards us.

**James** [00:13:43] But if you don't have the tool chains able to come together, you know, like we do with this plugin capability, you're left with a problem that a single command line can only use one device. And so when I do my compile, I'm forced to have a binary that only uses one device. And, you know, if you build with the CUDA tools, you know, your application is going to wake up and say, Where's my NVIDIA GPU? I'm going to use it, it'll be great and it is great. Or you use the HIP tools and your application will then we'll wake up and say, Where's that AMD GPU? I'll use it. It's great. You know, use the Intel tools. It wakes up and says, Where are the Intel devices? I'll use them. It's great.

**James** [00:14:19] I want my application to wake up and say, what devices are there? I'll use all of them. And we're seeing a very real realization now of the ability to do that. When you get these these tools and apply the plugins and suddenly you can create a binary. I know you were sending around earlier this week, Tony, that you ran something and you were excited that it you were able to dispatch things to both an integrated GPU from Intel and an NVIDIA GPU from the same application. That's very exciting because that power is

sitting there on your machine. Why not use it? So that's the. Yes, you're right. I'm very passionate about this because I think this is very, very important for the future of the industry, that we build our tools that way. We build our compilers, our libraries. To do that or debugger is to allow us to debug all these together. The profiler is to show us what's going on on all devices together.  Really so we can harness the power of the machine, instead of being locked out of selected parts of it, because our tools from one vendor or another are focused only on one vendor.

**Tony** [00:15:23] All right. Any last thoughts from you, Gordon?

**Gordon** [00:15:25] Yeah, I suppose about well, I thought just be. Yeah, we're really excited to be be releasing this and we're really keen for people to get get their hands on it and start using it and start giving us feedback. So I'd just suggest people, you know, go to the corporate website and download it and check out.

**Tony** [00:15:44] Alright, so that's what we have for you today.  Thanks to Gordon and James for joining us, and thank you our listener for tuning in.  Go check out the brand new Intel oneAPI 2023 Toolkit releases at developer.intel.com and check out the Codeplay oneAPI for NVIDIA and the beta oneAPI for AMD plugin from codeplay.com.  It's really exciting stuff…