Code Together Podcast Episode 27: ***Building Cross-architecture, Cross-vendor Tools for HPC Application Developers***
June 9, 2021
Host: Nicole Huesman, Intel
Guests: Xiaozhu Meng, Rice University research scientist; Sujata Tibrewala, Intel oneAPI Developer Community manager

Nicole (00:04): Welcome to Code Together, an interview series exploring the possibilities across architecture development with those who live it. I'm your host, Nicole Huesman.

Nicole (00:16): Just as a carpenter needs tools to build a house, a developer needs tools to write code and programs. This often means relying on other coders to develop tools and building on that foundation. Today, we'll talk to Xiaozhu Meng, a research scientist at Rice University with a focus on tools for developers in high performance computing or HPC. With both a Master's and PhD in computer science from the University of Wisconsin at Madison, his research interests include binary code analysis and instrumentation, performance analysis, and software security. Great to have you with us, Xiaozhu.

Xiaozhu (01:04): Thanks, glad to be here.

Nicole (01:06): I'd also like to welcome Sujata Tibrewala, Intel's oneAPI's developer community manager, back to the program.

Sujata (01:15): Thank you, Nicole. Pleasure to be here.

Nicole (01:17): So Xiaozhu, Can you tell us more about what you're doing at Rice University?

Xiaozhu (01:23): Sure. I work in the HPC Toolkit Project lead by the Professor John Mellor-Crummey. HPC Toolkit is an integratory tools that analyze and manage performance of programs on computers, ranging from desktop systems to supercomputers. Our recent focus is to develop support for applications accelerate by GPUs, including support for the Intel GPUs and its programming models, such as a DPC++.

Sujata (01:52): Great. Thank you Xiaozhu for coming here. So just to build up on that point, when you said you are working on tools for developers, what does that actually mean? What are they focused on, and what are they doing?

Xiaozhu (02:07): So software tools are the ones that aim to help application developers understand and identify problems in their applications. So two typical examples are performance profiling tools, which application you are working can use to understand where the program is slow or how to make the program faster. The other one is a debugger, where programmer use them to identify chronic issues. The HPC Toolkit, or where I'm working at, is focusing on performance tuning.

Sujata (02:41): And so you actually already answered my question, "What is important to developers?" So you said performance tuning and debugging of the program. Did I get that right? Those are the two things important to developers?

Xiaozhu (02:54): Sure, but we can definitely expand that a bit more. So for example, I'm focusing on the performance aspect. So fundamentally, the first issue, what developers need to understand, is to have a phase for understanding of the execution. Now that sounds fundamental and easy, but it's not necessarily always the case due to the complicated hardware architecture, instructions are executed out of order, compilers modify program dramatically to optimize the code. So when you ask your code, the performance normally you'll get may or may not actually match what your expectation. So there's a lot of things going on underneath that we need to sort of untangle and then present the information in a useful way.

Sujata (03:37): That's excellent. So you're talking about the different layers, right? The hardware compiler, and then the software itself, right?

Xiaozhu (03:44): Yeah, so performance numbers are mostly measured at the machine-code level. That's because hardware executes machine code, they're not executing source code. So when we collect performance numbers on the machine code level, those are not the level where application developers are familiar with.

Xiaozhu (04:04): So a key problem for us is how to map the performance numbers from machine-code back to a higher level programming language construct. So example, if you write code in DPC++, then you're not going to see machine instructions or those anomaly you want to know, you want to have those performance number mapped back to source lines, loops, understand inline functions, or those language construct that you understand, you're familiar with, from a developer perspective.

Sujata (04:34): Yeah, that's an excellent point. And being a developer myself, I can't go to the assembly language program development days, right? I love being able to program at C or DPC++ or C++ level. So yeah, you're absolutely right.

Sujata (04:52): Now, one of the other things that I always wonder is you mentioned hardware. So as a tools developer, probably you would have to support a variety of hardware, right?

Xiaozhu (05:04): Mm-hmm (affirmative).

Sujata (05:04): So why is it important for the developers and for a tools developer?

Xiaozhu (05:10): So in terms for hardware, as a tool developer we are sort of at a reactionary position wherein based on what's the commonly used hardware, we support them. So we are sort of not leading the fashion in terms of that, because we want to support certain hardware, then application will start to use that. We're on the other end where some hardware becomes more popular and then there is a desire to have tools, then we support them.

Xiaozhu (05:38): And practically it is the case that within, on a CPU side, we have different computing architecture, including x86, Power, Arm. On the GPU side, they are GPUs from different vendors. So it's definitely very interesting and a challenging path to support all different kinds of hardware. And for us, it's really just that because different applications, in order to have different needs, you may only have access to a particular type of machine. So we typically want to support all these different kind of hardware by building as common functionality as possible, and then support the architecture of specific things.

Sujata (06:20): That's great. So when you're doing that, when you're supporting multiple hardware, is that any wishlist for you or any limitations or road blocks that you face?

Xiaozhu (06:31): Yeah. So let's focus on a GPU side a little bit because on the CPU side, the hardware is more well-developed, because CPU has a longer history. So on the GPU side, we discuss Nvidia, AMD and Intel GPUs. Now, Nvidia has a longer history in GPU computing, so their hardware has relatively speaking, most of their support in terms of performance management on GPU code. What distinguished that Nvidia GPU from the other two are the support of a fine-grained management. Here, the fine-grained management means the capability to manage your performance inside kernels. That should be a function of source lines of kernels. Currently for Intel or AMD GPU, we can only know the performance numbers for a kernel. With more complicated applications a kernel can be huge, but then we have seen a kernel that includes hundreds of thousands of source lines. Now, if I tell you that this kernel is slow, you won't really be able to do anything because the kernel is a huge, and we would like to have the support from all of the vendors to be able to have this capability of doing fine-grained management so that application developers can know exactly where their code is slow. So I would say, in terms of GPU computing, that will be the topmost wishlist, to have the capability of fine-grained management.

Sujata (08:08): That's great. And then support from the vendor who's providing the hardware as well, right? That was another thing that you mentioned.

Xiaozhu (08:16): Right. Typically, GPU computing is the hardware vendor that provide the GPU, will be responsible for providing the monitoring and support in terms of performance management.

Sujata (08:27): So is that something that plays into your decision on whether to include a particular hardware or not in the Rice HPC toolkit?

Xiaozhu (08:37): Not really. We will try our best, even though we definitely would urge and we constantly engage with vendors to stress the need of being able to have a such fine-grained management capability. But typically we will say we won't support any particular hardware because of the lacking of certain management capability. Typically, if we start to look at a particular type of hardware, it is important because it has already gained sufficient traction in terms of field, so it will exist and it will be important regardless of hardware management support.

Sujata (09:16): That's great. So maybe that's an excellent segue to your oneAPI Level Zero support. Can you elaborate on the reasons why you included that support in the HPC Toolkit.

Xiaozhu (09:29): Sure. So from an administrative perspective data, Intel is going to deliver the exascale computing, supercomputer route which will equivalent with Intel GPUs. So as part of the ECP project, the HPC Toolkit will support the performance and management for Intel GPUs and its program models. But besides this administrative reasons in terms of technically speaking, is that Intel GPUs sort of program using a higher-level program model DPC++, which is based on SYCL, which is an expansion to C++. Comparing with what Nvidia or AMD have, their program models are CUDA and HIP, which are more of a C-style, lower-level programming construct. In CUDA or in HIP, you directly say, "I want to launch in kernel. I want to do memory allocation. I want to do some synchronizations." Those are the language closer to lower level.

Xiaozhu (10:37): On the other hand for Intel, for DPC++, it's more speaking at a higher level for them, where you want to have a parallel for loop, you want to do things in parallel, or maybe even better than that you want to create task graphs, which have some dependencies, and then you want to execute that in parallel. So I think a unique distinct characteristic, in terms of Intel GPUs program model, is that it's built on top of a more advanced C++, so it provides a higher-level programming construct, compared to the other two vendors.

Sujata (11:17): Okay. So it's basically because of the difference in the programming paradigm, is why you have chosen to include oneAPI and DPC++ support in the Rice HPC Tooklit?

Xiaozhu (11:29): Right. That's definitely one of the interesting aspects of oneAPI, compared to CUDA or HIP, which are the ones from the two other vendors.

Sujata (11:40): How has your experience been? Both in terms of providing that support and second, and that may be premature, and feel free to say that, in terms of developers experience with that.

Xiaozhu (11:56): Yeah. Overall in terms for wanting support for oneAPI, it is really early. Now the reason we started with building with the Level Zero, which is the lower-level interface inside oneAPI, that actually looks similar to the C language programing model like CUDA or HIP, where you have some kernels, I want to execute them rather than the higher-level language, such as DPC++. We got some basic support for Level Zero. So now we can know how much time is spent for each different kernels, what's the time spent for memory copy, what's the time spent for synchronization. But application developers don't write applications in Level Zero, they will use DPC++. So there is a natural gap between the data we collected at the Level Zero level, compared to what the application wants at a DPC++ level.

Xiaozhu (12:56): Our experience is that this is fine if you look at a simple program where for them when you have the power of four, and then we say, "Okay, this kernel takes this much in time, and because of the parallel four loop, so the kernel is all the same." So you can make some connection between performance data and the source code. But what if the application is more complicated, uses task graphs, then we can only know that there are certain dependencies, there are certain synchronization going on at a lower level, but we don't know why there is such a dependency or how to make it better.

Xiaozhu (13:31): So those are where we are, and Intel is actually trying to help us by providing some higher-level performance management interface. We'll then be able to trace or identify the events in terms of those higher-level constructs, to be able to trace the graphs, to be able to identify the operations associated with the tasks. Which tasks is waiting for which task, or which task is being executed. Those information can really be helpful when application developers tries to understand their code. It's still in early phase, and we are actively engaged with Intel, looking forward to their higher-level interface, and then incorporate that newer support to be able to have a higher-level performance measurement presentation for application developers. Right now is sort of a relatively lower level, you can see some numbers, but there are many layers between the actual numbers and your programming construct.

Sujata (14:42): So do you have any advice for others who are working with oneAPI and DPC++, from a tool developer's standpoint, and also from developer standpoint?

Xiaozhu (14:49): I think that my first suggestion... So first, let's talk about tool developers. My biggest suggestion will be then even though we are tool developers our goal is to develop our tools. It's actually quite important sometimes to put us into the application developers aspect, to really use our tool. So numerous times, we had the situation where we think we get some performing numbers and we are happy about that, we support a new language or a new model, we are happy about that. But if we really look at data, if we really think inside application developer's perspective, what if we really want to use our data to optimize our program? Then we realized, "Oh, there are some results that is not really correct. That doesn't make sense." Those sort of thinking, from a developers' perspective typically can help us identify issues in our own tools, and sometimes you may actually suggest that our tools are fine. And then we may actually find real problems, which we can report to application developers. And then that's a win-win for all of us.

Xiaozhu (16:00): So I would say this is one of our biggest lessons, is that we don't really know our tools are good or not until we put them into use. And even though we want other people to use our tools, but sometimes we have ourselves as tool developers have to use our own tools, otherwise it's just going to be difficult to identify issues. Second, I'm going to say then from an application developer's perspective, even though I'm not an application developer, I feel that is definitely because they understand their program better than anyone else. So sometimes we may feel like some issues, maybe say we want to present from numbers in a particular way which we think is useful. That may not necessarily the case for developers.

Xiaozhu (16:54): So we definitely did benefit from engagement or discussion with application developers to talk about what they think, what they can get from our measurement results. So I would say if an application developer is using someone's software tools, not just HPC Toolkit, it's always good to get in contact with the tool developers, because tool developers would love to hear feedback, would love to know how to improve. And we as tool developers only know that much about different applications, so application developers feedback is usually quite inspiring to us.

Sujata (17:36): Great. So as a tools developer, what are you looking ahead for? What is something that you're looking most forward to?

Xiaozhu (17:45): So still focused on the performance aspect, particularly on a GPU side. What we are looking for most is the capability of doing fine-grained management on the GPU code. We are seeing more and more applications that are writing kernels content many months ago, that if you just have a summary result of your applications, that's not really helpful. So we are definitely looking into the fine-grained management capabilities, appearing in how we run the vendors.

Sujata (18:21): Yeah. Makes sense with the push to bigger, better, faster. It does make sense that developers will get that fine-grained visibility so that they can optimize their programs much better.

Nicole (18:35): Xiaozhu, Sujata, such a great discussion. Getting the right tools into the hands of developers is so critical. As we wrap up today Xiaozhu, where can listeners go to learn more?

Xiaozhu (18:47): So, HPC Toolkit is an Open Source software, and it's available on GitHub, the spec software packaging manager. Or you can just go to hpctoolkit.org to understand more about what we do at Rice University.

Nicole (19:03): Great. Thank you. And Sujata, any other resources that our listeners should be aware of and check out?

Sujata (19:13): So, I will talk at the Developer Summit, oneAPI Developer Summit, in November, along with... I believe, Xiaozhu, you have a GitHub code repository, or the Rice HPC Toolkit code? Is that right?

Xiaozhu (19:25): Actually HPC Toolkit has a user organization or GitHub account. So, yeah, the code is there. So basically just search HPC Toolkit GitHub, that should show up as the first one.

Sujata (19:36): Yeah. And there is a DevMesh project also, which actually includes everything.

Xiaozhu (19:40): I think we actually have some tutorial down at NERSC National Labs, which we have some videos available, talking about how to use HPC Toolkit on YouTube actually.

Nicole (19:53): Xiaozhu, it's been so great to hear about your work at Rice University, and your experiences with oneAPI and Level Zero. And we're looking forward to hearing more in the future about your experiences. Thanks for joining us today.

Xiaozhu (20:08): Yeah, it's great to be here. Thanks for having me.

Nicole (20:10): And Sujata, always so nice to have you on the program.

Sujata (20:15): Thank you so much, Nicole. Thank you for having me.

Nicole (20:16): And a big thank you to all of our listeners for joining us. Let's continue the conversation at oneapi.com.