Episode 14: At the OpenMP Forefront
Host: Nicole Huesman, Intel
Guests: Tim Mattson, Intel; Bronis de Supinski, Lawrence Livermore 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.

The first OpenMP specification was released over two decades ago. Today, it's a mature, shared memory, multiprocessing programming API that supports Fortran, C and C++ across multiple platforms. It's become foundational for HPC application programmers.

Today's guests have truly *lived* at the forefront of this amazing journey.

Bronis de Supinski is chief technology officer for Livermore Computing at Lawrence Livermore Lab. In this role, he's responsible for formulating the lab's large-scale computing strategy and overseeing its implementation. He's also chair of the OpenMP Language Committee. We're so happy to have you with us today.

Bronis de Supinski: I'm happy to join you.

Nicole Huesman: And Tim Mattson, senior principal engineer, and manager of the programming systems research group, at Intel. His goal, in his own words, is to make serial software rare, and for the last 25 years, he has focused on creating programming languages and technologies to help people create parallel software. So great to have you on today's program.

Tim Mattson: Oh, it's my pleasure. I enjoy talking about OpenMP.

Nicole Huesman: We're excited to hear how OpenMP has progressed, what challenges you're grappling with today and what's on the horizon. With that, I'll let the two of you take it from here.

Tim Mattson: Well, okay, that sounds really good. I'm very excited about OpenMP. I'm very excited about Intel's oneAPI initiative and how OpenMP is a central part of that. So I'm very excited, especially since you know, we're over 20 years into this journey. It's rather fascinating to me that OpenMP's going strong and it's really showing no signs of slowing down. So that's pretty cool. So I'm curious. We came out with 5.0, gosh, when was that …

Bronis de Supinski: … two years ago …

Tim Mattson: … two years ago …

Bronis de Supinski: We're on a real regular cadence. Every five years we come out with a major release. And then two years later, we come out with a minor release. So we're going to shake that up a little.

Tim Mattson: Well, that's good. That's good. Of course, as we all know, the number of pages in a spec directly correlates to its quality. And so we started at 45 pages, and 5.0 is 600 pages … What do you think about the criticism that OpenMP has just gotten too complicated? Because when we started, this was the way to make it as simple as possible, and I don't think it's simple right now with 600 pages.

Bronis de Supinski: Well, if you want to use the simple stuff and nothing else, it still works just like it always did, but there's a lot of things that you can't do. You're not going to be able to use tasking. You're not going to be able to offload to accelerators, like GPUs. You need to have richer specification if you want to capture all possible patterns of parallelism. OpenMP started out as fork-join parallelism and there's, as you well know, far more patterns than just the one.

Tim Mattson: Oh yes, yes there is. And I have my book, if I'm allowed to plug it, you know, *The OpenMP Common Core: Making OpenMP Simple Again* from MIT press, which focuses on that simple core. So, 5.1 is coming up, right?

Episode 14: At the OpenMP Forefront
Host: Nicole Huesman, Intel
Guests: Tim Mattson, Intel; Bronis de Supinski, Lawrence Livermore National Lab

---

Bronis de Supinski: Indeed. OpenMP 5.1 will be approved on November 12th. It's all done in the language committee and being delivered to the ARB for its final vote and I have no reason to think that it won't pass.

Tim Mattson: And just for people listening, the ARB is the architecture review board, which is the governing body of OpenMP.

Bronis de Supinski: OpenMP is actually a corporation. It's technically not a nonprofit corporation because it doesn't have the legally filed paperwork to be a nonprofit, but it is a nonprofit, nonetheless. As I learned years ago, it's actually the equivalent of a bowling league in terms of its nonprofit status.

Tim Mattson: That's surprisingly new to me because I was the CEO of the ARB.

Bronis de Supinski: Well, that's a colloquial way of expressing it, but its legal status is roughly equivalent to that.

Tim Mattson: Interesting. Okay. Okay. So, will 5.1 be 800 pages, or …

Bronis de Supinski: No! It's not going to be 800 pages. We're looking at 711 pages in the PDF, front to back. So that includes several pages of appendices and front matter. So it's more, like, about 650 … 660 … pages of actual specifications. It's only 45 pages longer than 5.0.

Tim Mattson: I mean, I'm genuinely curious because even though I was heavily involved for the first half of OpenMP's life, right now, I just listen in and throw pot shots over email from time to time.

Bronis de Supinski: Yeah, I noticed that!

Tim Mattson: You would! What are the big things you're going to see in 5.1?

Bronis de Supinski: Well, you know, as a minor release, primarily what it's doing is making corrections and clarifications to 5.0. It does add a few things that'll make it easier to use.

Probably the most significant thing that's added is the first in the loop transformations is what we're calling them. You could argue that work sharing loops. So parallel loops were loop transformations as well. But we're adding, in particular, sequential loop transformations—tile and unroll. And in 6.0, we expect to add a fair number more of those. The reason that we're adding those is that, [with] those optimizations, you can have the compiler apply them automatically, normally, or most compilers have fragments to support them that are specific to that compiler. But how they interact with the OpenMP pragmas is not well defined in that situation, so we're giving clear semantics that allow you to specify, you know, basically when in the generation of your executable, when those transformations are applied. We've added a few other things.

Some of the implementers complained that we added too much for a minor release. We added something that I think many users will find very useful, which is the error directive, which allows you to, either at compile time or at runtime, instruct the compiler to emit a message, and you can control whether encountering that directive is either fatal or just merely a warning. The reason you would use them at compile time is that you can use them with things like the meta directive to say that you're encountering a chunk of code that you never expected to do to the way the meta directive works. And obviously the utility at runtime is pretty clear.

Tim Mattson: Well, that's pretty exciting that you're adding an error directive, 'cause I started work on an error model for OpenMP in 2001 and we didn't really come up with anything. So this is good.

Bronis de Supinski: Yeah. I wouldn't call this exactly an error model because basically you have the ability to emit a message, and either it just keeps on going on its merry way or it will terminate execution for you.

My favorite addition though, for OpenMP 5.1, is the nothing directive.

Tim Mattson: Oh, I can think about my good friend Michael Wolf and what a heyday he'd have with a nothing directive!

Bronis de Supinski: I think he hates it, but that's kind of part of the fun of it!

Tim Mattson: So what does the nothing directive do?

Bronis de Supinski: It does nothing! It's actually explicitly replaced with nothing.

Tim Mattson: So don't let anyone say OpenMP doesn't have a sense of humor!

Bronis de Supinski: Yeah. It's particularly useful with the meta directive. So rather than having a directed variant that's empty—so, nothing is specified—you can actually specify that what you want is nothing. So subsequently, people don't come along that are maintaining your code and think that you have an error there where you didn't type anything.

Tim Mattson: Yeah, any of us who are software engineers will immediately see the value of that. Others might scratch their head, but no, I can see the great value in that. So I'm curious, you spend an awful lot more time these days than I do with DOE labs and the HPC community in that world. What do you see within the user community you work with, with [respect to the] use of OpenMP—is it growing, shrinking, are people happy with it?

Bronis de Supinski: It's definitely growing, we're getting more and more systems with GPUs, and there's a few different ways to use GPUs, but most of them people find cumbersome and non-portable. Probably the least cumbersome ways are basically building things on top of C++ and using Lambdas, and then underneath that, implementing on some sort of substrate that might be system-specific, or it might be OpenMP. In fact, two of the most popular frameworks take that approach—so, Kokkos and RAJA have OpenMP backends and use them regularly.

Tim Mattson: Right. Actually I'm very excited about Kokkos and RAJA. We have yet another example of the C++ community really getting out of hand and thinking hard about how to express concurrency. So yeah, that's cool stuff. What do you think of SYCL then, 'cause SYCL just wraps a Lambda?

Bronis de Supinski: From the C++ programmers at Livermore [National Lab] that I talk with, they're interested in SYCL, but it's not yet widely supported enough to really get their interest. What I think the C++ programmers most want to see is everything gets pulled right into the language and then they just use language features. Realistically there's something that like OpenMP is willing to do that C++ won't ever do, which provides you more direct control of optimization. I don't see OpenMP going away, but for that reason, and also because frankly, at least for the foreseeable future, we'll still have Fortran and C and having a mechanism that's consistent across them is pretty desirable. We have a lot of people that end up writing programs that touch all three of those languages.

Tim Mattson: Right? I mean, you're absolutely right. This is a conversation I have inside Intel right now because we've gone, I think it's fair to say, all in on SYCL, because SYCL is at the core of our Data Parallel C++ system.

Bronis de Supinski: So I've heard. So, like, you know, oneAPI, what's up with that, right? Like if I understand correctly, you've got threading building blocks, you've got SYCL, you've got OpenMP. You must have like three or four other programming, like, models in there, right? So, like, if it's oneAPI, what is that, are you counting binary and there's an odd number?

Tim Mattson: No, no, no. Let me tell you, let me tell you. I work at Intel, so of course, I have an answer for you, and I even believe it. So the idea is not that there's only one API. The idea is that if you pick one of the APIs, you can program from that one API, any of our products, whether it's the GPU, the CPU or the FPGA.

Bronis de Supinski: And here, I thought it was more like this, where it was like 'one API to rule them all'.

Tim Mattson: Yeah, well, there's some people early on who probably thought about it that way. But to be honest with you, I was quite surprised when all the other approaches joined the oneAPI label, because when those of us inside the company's engineers started the process, it was just SYCL. And we weren't going to try to do everything under one label. But you know, I think Intel came around and decided, right or wrong, that, you know, it's clean to have this one umbrella. And as I said, the goal—well, I'll be honest, I'm not convinced it's there yet across all of the different APIs—but the goal is that you pick one of the APIs, and from that one API, you can target all the hardware.

Bronis de Supinski: And what do you do when you want to use a library that uses a different one?

Tim Mattson: Oh gosh!

Bronis de Supinski: So that reminds me of another thing we added in 5.1, which is the interrupt construct. It returns you an interrupt object that, if you're using something like CUDA or OpenCL, allows you to, you know, pick the stream or the queue that's appropriate for it, and it gets you that object back, and then allows you to synchronize from CUDA with your OpenMP, or from OpenCL with your OpenMP, allows you to really mix the APIs.

Tim Mattson: Wow, I love that!

Bronis de Supinski: Some of the Intel folks really worked to shape how that works, so that it can interoperate with multiple different APIs like threading building blocks and OpenCL, say, in the same program.

Tim Mattson: Right, 'cause I would never use CUDA, but I would use OpenCL all the time.

Bronis de Supinski: I know, that's why I picked it.

Tim Mattson: So we're actually doing some work deep in the bowels of Intel on the parallel composability problem and how do runtimes have to be changed to make that really, really work. So we're aware of how hard it is to compose across APIs. The interop capability you talked about would help, but basically you fundamentally need a shared resource manager, so that different instances of OpenMP can know what even different instances of OpenMP are doing.

Bronis de Supinski: Yeah. It's been a long-time weakness of OpenMP. And I wouldn't claim that this interop construct solves it, but it does improve the situation.

Tim Mattson: It does, it does.

Bronis de Supinski: So I haven't mentioned OpenMP 5.2. So normally the next OpenMP specification after 5.1 would be OpenMP 6.0, and that would come out three years from now in November of 2023. We're still planning on that, but we're going to also come out with something that we're going to call OpenMP 5.2. And what that's going to do is rearrange some things from 5.1 and reorganize how we

specify the OpenMP syntax. It should consolidate the syntax across the different constructs and help us find some places where things are not consistent. Right now we have some things kind of buried in the syntax, in fact, that are probably not obvious to people. But basically the syntax says, you know, this has to occur at a specific point in the use of a construct or it's not included in what we say is allowed as the parameters. So specific clause. So if you give it while you're not conforming the syntax and we're going to clean that up. We also expect to find other restrictions that are not consistent across constructs. And we'll either add those restrictions or possibly do things to clarify that coming in 6.0, things will be made more consistent.

Tim Mattson: And you're hoping to do that in about a year from now.

Bronis de Supinski: Actually, we hope to have it out through comment in a few months. And ideally, you know, we'd like to get it released by June or July of next year. Realistically, yeah, it'll probably take us till next November, just because we like to release things for comment and then do a very careful proofreading round on the spec and find as many errors as we can, although you never find all.

Tim Mattson: Well, you know, I'll be really impressed if you pull that off. Looking back across the history of OpenMP, between 2.0 and 2.5, we did a major restructuring and I remember Sanjeev Shah and I, at the time going, 'Yeah, we'll budget six months for this. Yeah, that's about all it'll take, 'cause we're just putting together the Fortran and the C spec and consolidating, not adding anything new.' It took us between two and three years. So, it's a big job!

Bronis de Supinski: Yeah, I'm aware of that history and that's why I'm hedging a little bit. But I think what you undertook was a lot more difficult, frankly. We have everything already in one place and it's just a matter of wishing it all together … rather than spread throughout.

Tim Mattson: Right, right. Well, I know we're almost out of time, but I wanted to make sure to talk to you about a topic that's near and dear to my heart right now. And that's Python. Once again, you know, you move around inside the DOE HPC community more than I am right now. What do you hear about parallelism from Python these days among your user bases that you talk to?

Bronis de Supinski: I haven't been hearing about it. They always used Python sort of a parallel way, but our applications typically use Python as kind of an outer shell that provides for steering, allows them to pull together what they call packages. So different kinds of physics into a single run. And they do make use of Py MPI to be able to talk to different MPI processes from their Python shell. But typically they avoid having compute-intensive work in Python because you know, I've talked to them about like, 'Oh, would an OpenMP and Python or something like that be useful?' And they generally said, 'No, if we have that much computation going on, we basically move it to lower language at that point.' But you know, people move quickly, so maybe I just haven't caught up with something that's coming down the pike.

Tim Mattson: Yeah, you'd be surprised. Last year, at Supercomputing, the weeks leading up to it and then at Supercomputing, I interviewed everyone in the DOE community I could nail down to talk to about it. And then there was a Birds of a Feather session at Supercomputing on Python. That room had maybe twice as many people as you get at the OpenMP Birds of a Feather sessions—the room was packed to the gills! They were interested in parallel computing with Python. And the word I got out of it is because of the GIL, the global interpreter lock, multithreading with Python is bitterly painful. So most people, if they were doing parallel computing directly in Python, as opposed to just calling PyTorch, would actually use the MTI interface and go multiprocessing. But gosh, we're doing some cool stuff at Intel. Multithreading with Python. And I don't know what I can say. So I've learned that if I don't know I can say it, I won't, but stay tuned. I am hoping to have something that will just knock your socks off!

Episode 14: At the OpenMP Forefront
Host: Nicole Huesman, Intel
Guests: Tim Mattson, Intel; Bronis de Supinski, Lawrence Livermore National Lab

---

Bronis de Supinski: Who are you, and what have you done with Tim Mattson?

Tim Mattson: I know, I know. This is weird. I hate Python, but I have to admit it!

Bronis de Supinski: No, no, I meant not saying that, that which you don't know, you can't say!

Tim Mattson: Well, I'm getting old and I'm getting more cheerful in my old age. But also Python just offends me at a deep level because it hides too much of the hardware. But I don't care. People are coming out of school these days, and it's the only language they know. And it's growing rapidly while C and C++ are fairly flat, some would say shrinking, you can argue that depending on what language survey you look at, but Python is growing so fast. I think we have to figure out how to make people get performance just as much as they would out of C writing straight Python. And we can do it. Not always, not general, you know, but it's blowing me away how far we can go, but that's going to have to wait six months, three months, not too long.

Nicole Huesman: So you guys, this has been such a fun conversation! As we wrap this up, Bronis, where can listeners go to learn more?

Bronis de Supinski: www.openmp.org. There's all kinds of material there. In a short while, there'll be a new specification there. Right now, there's TR9 [OpenMP Technical Report 9: Version 5.1 Public Comment Draft].

Nicole Huesman: Excellent. Thank you. And Tim, from your perspective, where can our listeners go to learn more?

Tim Mattson: Well, of course, if you're new to OpenMP, you should buy my book, *The OpenMP Common Core: Making OpenMP Simple Again*. If you're already experienced with OpenMP, there's a fabulous book that I did not help write, so it's okay for me to call it fabulous: *Using OpenMP, The Next Step*. It's also from MIT press. Ruud van der Pas and that group wrote it. Excellent book for looking beyond the fork-joins, and looking at NUMA and GPUs, and it's just great. And then of course, Supercomputing is coming up.

Bronis de Supinski: I was going to mention that!

Tim Mattson: There's always a lot of great content [there]. They're having a Birds of a Feather again, right, on OpenMP?

Bronis de Supinski: We're having a Birds of a Feather, and there are several tutorials. I think your *Common Core* tutorial is there [Part I | Part II] . Two of the authors of the *Next Step OpenMP* book that you just mentioned, Ruud and Christian, are there with me and with Michael Klemm, the current CEO of OpenMP, we're giving an advanced OpenMP tutorial [Part I | Part II]. I think you're also with a few other people giving one on using GPUs with OpenMP.

Tim Mattson: Programming your GPU with OpenMP [Part I | Part II]. And the funny thing is, even though I've been doing that for years, I'm not doing it this year because I'm doing a SYCL tutorial.

Oh, and of course, where can people go to learn more?! I work at Intel, and the Intel oneAPI HPC toolkit has OpenMP, MPI [Library], Python, all included. So go out and download that now!

Nicole Huesman: Absolutely. So, with this, Bronis, thank you so much for sharing your insights with us today. It's really been incredible.

Bronis de Supinski: It's been fun!

Nicole Huesman: And Tim, it is always fantastic to talk to you. So great to have you on today's program.

Tim Mattson: Well, it's a pleasure. It'll be interesting to see if I'm ever asked back.

Episode 14: At the OpenMP Forefront
Host: Nicole Huesman, Intel
Guests: Tim Mattson, Intel; Bronis de Supinski, Lawrence Livermore National Lab

---

Nicole Huesman: Actually, I was just going to give you that invitation. There's so much to talk about in this space that it would be fantastic to have you both back on the program.

Tim Mattson: Sure!

Nicole Huesman: And for all of our listeners out there, thanks so much for joining us. It's been an absolute blast with these two! Let's continue the conversation at oneapi.com. Until next time!