Code Together Podcast
Episode 3: Porting Math Libraries Across Heterogeneous Architectures
Host: Nicole Huesman, Intel
Guests: Julia Sukharina, Intel; Mehdi Goli, Codeplay Software

---

Nicole Huesman: 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.

It's really no secret that we produce extraordinary amounts of data every day, driving the need for performant, scalable algorithms that can process it. Compute-intensive applications depend on serious math routines and libraries. And this landscape is quickly evolving.

I'm excited to welcome Julia Sukharina, Senior Engineering Manager at Intel to today's discussion. Hi, Julia.

Julia Sukharina: Hi, Nicole.

Nicole Huesman: And joining Julia today to share his experiences with us is Mehdi Goli, Principal Software Engineer at Codeplay Software. Welcome, Mehdi.

Mehdi Goli: Thank you.

Nicole Huesman: Mehdi, let's start with you. You've done some great work with libraries. Can you talk about the role that libraries play in the overall software ecosystem?

Mehdi Goli: Yes. I think libraries provide an essential role in successfully establishing an ecosystem among users. Libraries, in my point of view, help people who want to write high-level programs to make function calls instead of implementing those functions repeatedly. So usually these functions are written by developers who are experts in domain and in hardware architecture. They're highly optimized critical algorithms for specific hardware. Take an example of GEMM algorithm in Intel end products for both CPU or GPU. So I think other hardware vendors provide libraries their own ecosystem as well, and it's very common for each vendor to provide continuous release of core libraries in their ecosystem.

Nicole Huesman: Can you talk a little bit about your experience in developing math libraries and the advances that you're seeing?

Mehdi Goli: Yes, so my experience in math libraries goes back to developing linear algebraic kernel for eigen, and then it continued to developing SYCL BLAS libraries. So although SYCL BLAS libraries can run on any SYCL-supported devices, the main goal of creating it was to provide highly optimized math library for devices which doesn't have vendor-optimized, or vendor-specified implemented math libraries. We use C++ templates in programming on that algorithm in order to easily tune it for various embedded devices, which does not use any vendor-specified libraries. For example, the SYCL BLAS libraries highly optimized for Renesas devices.

So in my point of view, we need to have a purpose to write a math library. So if you're going to write a math library, which, at best, is going to give the same performance as a vendor's specified driver, you're reinventing the wheel. So unless you're writing a kernel which is

significantly faster than the vendor-optimized library. So these days, almost all major hardware vendors provide highly optimized libraries for their devices. So in my point of view, the future will be going towards performance portability of math libraries across heterogeneous architectures, while using existing vector-optimized math libraries instead of implementing it in different languages.

Nicole Huesman: Julia, you sit at Intel and you're focused on the Intel Math Kernel Library (Intel MKL), and the Math Kernel Library has been around for so many years. A vast majority of supercomputers that use math libraries rely on Intel MKL. Can you talk about how it's evolved over the years?

Julia Sukharina: Sure. Indeed, 95% of the world's top 500 supercomputers that use math libraries are actually using MKL. The library provides optimized math vectors for science, engineering, financial applications, and includes such core math functionality as BLAS, LAPACK, Fast Fourier Transform, and many others. We looked at the current situation and in recent years, we increased our focus on data science. We introduced Python integration in addition to standard interfaces for C and Fortran languages. All this work helps us to be a library of choice for developers, and actually confirms that MKL is among the most popular libraries for years. However, so far we were focused mostly on CPU optimizations. But this year, I think, is probably the most important year for MKL because, as part of the oneAPI industry initiative, Intel MKL actually becomes oneMKL. And it's not only a new name, but it's also a new concept. Now we develop Data Parallel C++ interfaces, which are actually hardware-agnostic. And with this, we also extend our CPU optimizations to our GPU optimizations.

Nicole Huesman: So you've both talked about the future of libraries in being able to work across architectures. Mehdi, can you talk about some of the challenges that you're seeing and what developers are facing?

Mehdi Goli: Yes. I think the missing key feature for existing math kernel libraries is portability across heterogeneous platforms. And I think this is because there's no common language to abstract out the memory model and execution model from various heterogeneous devices. So, in other words, how to glue together different parts of code written in different hardware. And I believe this problem can be solved by using SYCL as the unifying programming model. So, in that case, it is possible to develop performance-portable libraries among various hardware architectures, sharing the same library interface. I think from that point of view, oneMKL is in the right direction.

Nicole Huesman: Julia, can you talk about how Intel is working to address the challenges that Mehdi is talking about and dive a little bit deeper with us into recent developments around oneMKL and how we're working to solve these challenges?

Julia Sukharina: About a couple of weeks ago, we actually released oneMKL interfaces on GitHub, and basically, the project consists of two parts. Interfaces, which are DPC++

interfaces and extensions, which allow you to code to different hardware implementations underneath. This project is currently limited to the BLAS domain, but we have really big plans and we'll introduce more domains. And the main two goals we are trying to solve is actually what Mehdi mentioned before: the first one is to establish industry-standard interfaces for math functions. There are many libraries, but all of them have different interfaces. We are working on oneMKL specification to actually define interfaces and implement them through oneMKL open source project. The second goal for which we actually created this open source project is to enable developers to efficiently port portable math-intensive applications and allow them to run across multiple vendor architectures. Just to mention, so far, developers need to significantly modify or completely rewrite their code if they wanted to switch from one hardware architecture to another. And some of them will abandon the idea of doing so because that's huge work. But now, with oneMKL open source interface project, we hope to make this work easier for developers, and they can switch actually now between different hardware implementations just by specifying what target device they plan to use. They don't need to update interfaces—everything will work as it is.

Nicole Huesman: And Mehdi, you've had direct, hands-on experience with what Julia is talking about. Can you tell our listeners about your insights?

Mehdi Goli: Yes. So, we have added the CuBLAS back into oneMKL. This is the first math library implementation for oneAPI on Nvidia GPUs. It uses the SYCL interoperability feature implemented by DPC++. It is a 2020 SYCL features, and these features enables us the integrations of third-party vendor-specified libraries for SYCL-supported back-end, like C++ host, OpenCL and CUDA. While using this feature, we can get the underlying CUDA memory and CUDA context from SYCL memory and SYCL context. And then we retrieve the CuBLAS handle for the specified contexts and directly call CuBLAS routine. The DPC++ runtimes will handle the kernel dependency in their multiple CuBLAS routine calls. And this will also achieve the performance portability for oneMKL on Nvidia GPUs while using the same industry standard that oneMKL introduce for BLAS.

Nicole Huesman: This is really exciting to see Codeplay be one of the first to enable some of the oneMKL functions for cross-vendor GPUs. Can you talk about what challenges you encountered? How difficult was it?

Mehdi Goli: The main challenge was integrating CuBLAS without losing performance. So we believe that it's crucial for a user application that embed oneMKL routines to have comparable performance as opposed to directly using CuBLAS. During this process, however, there was some other interface challenges, basically. The NVIDIA CuBLAS library is designed for the CUDA programming model, which is mostly C-based. This limits the ability of the library to integrate with modern code bases. So we had to work closely with both DPC++ team and Codeplay, both on Codeplay side and Intel side, basically, to be able to have the capability that we needed and the features that we needed to add in the oneAPI side. And since CuBLAS is a closed-source and vendor-locked library, it was difficult for us to find all

the documentation we needed. So we found out there are a lot of undefined and unclear behavior and some corner cases in the CuBLAS library. But however, at the end we managed to perform this integration in a way that oneMKL and SYCL users don't have to be bothered with these details.

Nicole Huesman: It's great to see the collaboration to truly make this cross-vendor and really bring out that portability. Julia, if I were a developer wanting to use Intel oneMKL, what other things would I need?

Julia Sukharina: Well, first of all, you will need to get a compiler, and for now it can be Intel Data Parallel C++ compiler, or an Intel LLVM compiler, which supports CPU, GPU, and actually also supports NVIDIA, which was also contributed by Codeplay. And depending on target hardware, you will need to also download the library with the implementation for this hardware. At this moment in the open source interface project, we support Intel oneMKL, which is binary product and supports Intel CPU and GPU. And as it was mentioned, CuBLAS for NVIDIA. And that's it. You're actually ready to go.

Nicole Huesman: Can you give us a vision of what's next for support of additional math functions, libraries, other hardware?

Julia Sukharina: We plan to enable Netlib libraries on the oneMKL interfaces so the community will have an option to play with some reference implementation. We also will continue to add more interfaces for math domains. LAPACK, Fast Fourier Transforms, random number generators, sparse solvers and vectors math. And we see it will be much harder to introduce domains which don't have any standard interfaces in the industry. And it will be really challenging to come up with something which will become a standard. And we also have made some prototypes for FPGA support to make sure that our interfaces are really extendable to any hardware, and also helps to achieve the best performance.

Nicole Huesman: So for developers who want to get started, want to get involved, participate, what do you suggest for them?

Julia Sukharina: That's quite simple. They actually need to go to the github.com/oneAPI-SRC/oneMKL and what they can do is they can enable more libraries implementation under existing interfaces. It can be your favorite libraries for CPU computation, GPU or any other device. You can also contribute proposals for new math interfaces to drive the standardization across multiple hardware. This project can be really huge and for us it's very important that industry developers would see great future in it and help us to grow this project.

Nicole Huesman: Julia, Mehdi, it’s been a pleasure to have you on today’s program.

Julia Sukharina: Thanks, Nicole, it was a very good conversation.

Mehdi Goli: It has been a very good experience. Thank you, guys.

Code Together Podcast
Episode 3: Porting Math Libraries Across Heterogeneous Architectures
Host: Nicole Huesman, Intel
Guests: Julia Sukharina, Intel; Mehdi Goli, Codeplay Software

---

Nicole Huesman: We'd love to have you both back on the program. For all of you listening, thanks for tuning in. You can join the conversation at oneapi.com. Until next time!