Computer architects are in the middle of a massive shift. During this shift, heterogeneous programming becomes a critical skill.

The objective of oneAPI is to help abstract the enormous diversity that’s coming in hardware, so software engineers can take advantage of it higher up in the stack and get the most out of the hardware. James, a software engineer who also works closely with the hardware, loves what oneAPI can contribute in light of trending heterogeneous computing.

The word heterogeneous here means that different system devices can do the computation, but they don’t run the same instruction set. For example, the CPU has one way of running instructions, and the GPU has a different way. FPGA, other ASICs, and specialty devices each have different programmed ways. Taking advantage of all of those is the goal.

John Hennessy and David Patterson, leaders in the computer architecture field, called this the new golden age of computer architecture. For computer architects, it’s an amazing time to design all sorts of specialty devices to do better things for specific domains. Still, it can be difficult and even scary for programmers because of the complexity. This is where the oneAPI initiative and oneAPI products come in.

The name, oneAPI, is both an initiative with a specification and an implementation. The initiative is a simple concept: software developers should have the freedom to use any device they want with full performance. Their coding should preserve its value; they should not have to rewrite for every new device. These qualities can be applied to compilers, libraries, debuggers, and any tool.

A great deal of software to implement certain specifications has been open-sourced. Much of it originated from Intel, but Intel also makes it available as downloadable toolkits that are already pre-built and ready to go to support Intel hardware. Other vendors creating parts of oneAPI have their implementations, so everyone comes together on specifications, but each gets to generate support for their hardware in an open multi-vendor fashion.

The high-level goal is to write code once, and it runs well everywhere. It is, however, a complex problem that will require some performance tuning. For example, if an algorithm is working well on a GPU, it will similarly run on almost any GPU. If you switch the algorithm to run on an FPGA, you can maximize the reuse of your program, but you have the option to recode parts of the algorithm for it to run as well. It’s a complex approach that allows you to take advantage of any hardware with varying degrees of hopefully very isolated rewrites.
You can build one code for different devices. Even more, sophisticated runtime approaches allow you to detect what’s there and run various pieces of code so you can have a common source code that dynamically decides. You can apply for a deployer program, and it can figure out what’s on your machine and use it dynamically rather than tell the user to run a different binary depending on what device you are on.

This is exciting because, for a long time, engineers used a software stack that was the best for their machines. Nowadays, they want to compile a program that may use devices from multiple vendors. The program should react to that. For that to work, the compiler has to be able to spit out code for these various devices in a reliable fashion. This is where openness comes into play.

Some will argue that a closed system will improve performance since the libraries and language are tuned specifically to the hardware capabilities. But the question is, how can you get the best of both worlds? If, for example, there is a vendor-specific implementation for a math library, there can be a common program. A big part of oneAPI is not trying to reinvent the entire world, but trying to organize it to take advantage of the best on every platform that’s possible.

OneAPI has the capabilities of moving memory and moving data. Different programming models sit on top of oneAPI, and it’s your choice of how much you want to get involved in managing the memory. Moving data around is expensive and consumes power, so you can’t escape that, but oneAPI gives you the tools to manage that by querying the system and letting your program at runtime make the right, dynamic decisions that will get you the best performance.

Intel has decades of experience building tools to help with tuning and migration and has highly optimizing compilers. V2 has helped evolve the industry around hardware counters on processors becoming the norm. There are various other analysis tools to give feedback, from the structure of your program to finding deadlock and parallel applications to seeing where you need to add some locks. Intel is making all of these tools available in an oneAPI fashion more versatile than just being about a CPU.

The first groups adopting oneAPI include high-performance computing (HPC). With the explosion of new computer architecture ideas, there will be greater diversity and innovation in this space. The large codes that can help solve the most significant engineering problems in the world, or for example, solve pharmaceutical problems and evaluate new drugs, demands the latest and greatest hardware. So this concept of performance portability is getting in front of national labs, universities, and research centers.

Today’s HPC is tomorrow’s department servers, so the capability to use different hardware requires software engineers to plan and pay attention to how portable the code is because code does not die quickly; it lasts for decades. Within a few years, heterogeneous systems will touch everyone, and now is the time to get educated about them.

Software engineers do not need to parallel programs to take advantage of oneAPI. It’s about utilizing things that are open and multi-vendor, multi-architecture capable. Even engineers at the top of the stack should understand what is in the stack and what it’s capable of in terms of portability and performance portability.

The IoT community is another group that has been way ahead in programming across multiple heterogeneous compute devices and using different methods to manage them. Their compute capabilities continue to rise as technology moves on. So oneAPI applies here and can help formalize or standardize things that have been innovated first in the embedded world.

The idea of one API has moved from being a crazy idea that only a few people were talking about to now more and more recognizing that it makes sense and solves problems in their organization.

The most straightforward place to learn about the initiative is at the oneapi.io website. To learn about implementation, click on the implementation tab for a link to follow to find Intel implementations. There, you can download the different toolkits. Search for the Intel dev cloud to try out the tools in the cloud for free, including on other hardware. The oneapi.io website also has a variety of tutorials and resources.