Special Analysis

Preparing for Tomorrow: Future GPU Programming Strategies

Mike Heroux and Tom Sorensen
October 2025

OVERVIEW

As GPU architectures broaden in scope, ranging from long-established players like NVIDIA to newer alternatives from AMD and Intel, developers increasingly are pursing programming approaches that allow for code portability. In addition, the ongoing evolution in GPU-based computing now encompasses not just traditional GPU architectures from vendors including like NVIDIA, AMD, and Intel, but also specialized architectures including Cerebras' Wafer-Scale Engine (WSE), Graphcore's Intelligence Processing Unit (IPU), and Google's Tensor Processing Unit (TPU).

These emerging processors aim to accelerate AI and HPC workloads by addressing well-known performance bottlenecks—particularly memory bandwidth, interconnect overhead, and parallel scalability. Cerebras, for instance, implements an entire wafer as a single chip, allowing on-chip memory and low-latency communication beyond traditional multi-GPU systems. As new architectures emerge, the software stack users rely upon will need to be available, making portability an important consideration in the design and porting of software to new hardware.

- For AI users, frameworks like PyTorch and TensorFlow are available on today's GPUs and are some of the first software capabilities offered on new accelerators, providing portability for AI users who depend on these frameworks for their own software capabilities and computational workflows. The high level, adaptable functionality of these frameworks can be leveraged for a variety of problems while still providing hardware vendors sufficient latitude to optimize implementations for efficiency.
- For HPC users, the US government funded Exascale Computing Project (ECP) produced a collection of scientific libraries and tools that work across today's GPUs, curated as a single collection in the E4S project. Users of these products can develop and maintain a single code base that works on many current and future systems. But E4S is still emerging and there is no widely used HPC focused framework with the reach of PyTorch and TensorFlow.

While AI frameworks provide portability for many needs, and E4S is emerging in the HPC-AI space, applications that need additional GPU portability can use other approaches. ECP, in collaboration with computer system vendors, sponsored efforts in several portable programming systems. Three of the most popular are OpenACC, OpenMP, and Kokkos. One emerging approach is standard C++.

OpenACC is a directive-based offloading model that relies on special lines of embedded source code to guide compilation for accelerated computations on GPUs. By reducing the need for deep parallel programming expertise, OpenACC allows scientists and engineers to offload CPU-based workloads with minimal code changes. Although OpenACC is backed by a consortium, it is most heavily optimized for NVIDIA hardware.

OpenMP, which began as a standard for shared-memory parallelism on CPUs, now supports GPU offloading via directives, similar to OpenACC. OpenMP is widely recognized across a range of architectures and has production-quality support on NVIDIA, AMD, and Intel GPUs.

Kokkos provides a C++ library framework for performance portability. Kokkos is designed to run on NVIDIA, AMD, and Intel hardware by mapping its abstractions to native APIs such as CUDA, HIP, and DPC++ respectively. This technique allows a single codebase to achieve near-native performance across different GPU architectures. Kokkos requires developers to adopt a modern C++ style and embrace its concepts for execution and memory spaces. If a critical function is too slow using Kokkos, a vendor-specific version can be swapped in for that function, enabling an effective blending of portability and vendor optimization.

New C++ features, such as support for parallel algorithms in C++17 and further enhancements in C++20/23, enable accelerated standard C++ code. Widespread availability of new C++ features tends to take time if a user wants to have portability across a broad range of hardware but in the long run can reduce friction for developers unfamiliar with proprietary APIs, enabling them to leverage vendor compiler optimizations.

TABLE 1

Software Portability Options for GPUs and Emerging Accelerators

Approach	Strengths	Weaknesses
High-level frameworks (PyTorch, TensorFlow, E4S)	Provides broad scope of portable functionality designed and tested for cross-platform execution	Requires coarse-grain adoption of functionality and implementation that may restrict user options
OpenACC	Relatively straightforward to adopt, especially for users without extensive parallel programming experience. Strong support on NVIDIA GPUs due to close association with NVIDIA, useful for quickly porting large existing CPU-based HPC codes	Limited vendor support on AMD and Intel GPUs, with less mature tooling. Adoption mostly limited to select HPC domains, fewer advanced parallel programming features compared to OpenMP
OpenMP	Widely recognized industry standard, originally for CPU multithreading, offers granular control over data mapping and parallel execution	More complex syntax and steeper learning curve than OpenACC, especially for offloading, performance can vary across different GPU vendors and compiler implementations
Kokkos	C++-based abstraction that targets NVIDIA (CUDA), AMD (HIP), and Intel (DPC++), high performance obtained through direct mapping to vendor APIs	Necessitates modern C++ proficiency and restructured code design, steeper initial learning curve and ecosystem unfamiliarity for some users
C++ Parallelism (Standard C++17/20/23)	Standardized features reducing vendor lock-in, integrates naturally with modern C++ codebases	GPU offloading support is still evolving and may lag proprietary APIs

Source: Hyperion Research, 2025

ANALYST COMMENT

Directive-based approaches (OpenACC and OpenMP) have proven especially valuable for HPC environments that emphasize rapid porting of existing CPU-focused codes.

- OpenACC's relatively automated parallelism may be the quickest path to GPU acceleration on NVIDIA hardware, but broader vendor support is less robust.
- OpenMP, by contrast, is considered a safe bet for longevity, given its large community and proven CPU capabilities.

Likewise, Kokkos is attractive for heterogeneous supercomputing environments or for those planning to switch hardware vendors in future procurements. Its library-based approach and emphasis on modern C++ paradigms allows developers to avoid rewriting applications for each new GPU architecture. Despite the steeper initial learning curve, HPC centers have reported strong performance across NVIDIA, AMD, and Intel GPUs once the Kokkos framework is properly integrated.

From a strategic standpoint, organizations with deep investments in a single vendor's stack (for instance, NVIDIA's CUDA or Intel's oneAPI) may find their path of least resistance by layering OpenACC or OpenMP on top of that existing infrastructure. However, those seeking to keep vendor options open—whether due to cost considerations, hardware availability, or performance requirements—should weigh the longer-term benefits of Kokkos and modern C++. Additionally, leveraging the software libraries and tools produced by the ECP and sustained by current efforts associated with E4S, provides a way to achieve portability for at least some needed functionality.

As GPU offerings continue to evolve, demand for high-level portability solutions is likely to grow, spurring further development of frameworks, standards, and libraries that unify diverse GPU architectures under a single codebase. While OpenACC, OpenMP, and Kokkos provide immediate support for portability, C++ parallelism may provide the stable long-term approach. NVIDIA encourages the use of modern C++ parallel programming features to their users.

Software portability on GPUs for AI applications is possible today using high-level frameworks like PyTorch and TensorFlow. Beyond these two frameworks, the emerging HPC software capabilities curated by E4S, and portable programming environments like OpenACC, OpenMP, and Kokkos provide a lower-level solution upon which applications can build. Ultimately, portability may be best realized by using native C++ parallel features.

About Hyperion Research, LLC

Hyperion Research provides data-driven research, analysis and recommendations for technologies, applications, and markets in high performance computing and emerging technology areas to help organizations worldwide make effective decisions and seize growth opportunities. Research includes market sizing and forecasting, share tracking, segmentation, technology, and related trend analysis, and both user & vendor analysis for multi-user technical server technology used for HPC and HPDA (high performance data analysis). Hyperion Research provides thought leadership and practical guidance for users, vendors, and other members of the HPC community by focusing on key market and technology trends across government, industry, commerce, and academia.

Headquarters

365 Summit Avenue
St. Paul, MN 55102
USA
612.812.5798
www.HyperionResearch.com and www.hpcuserforum.com