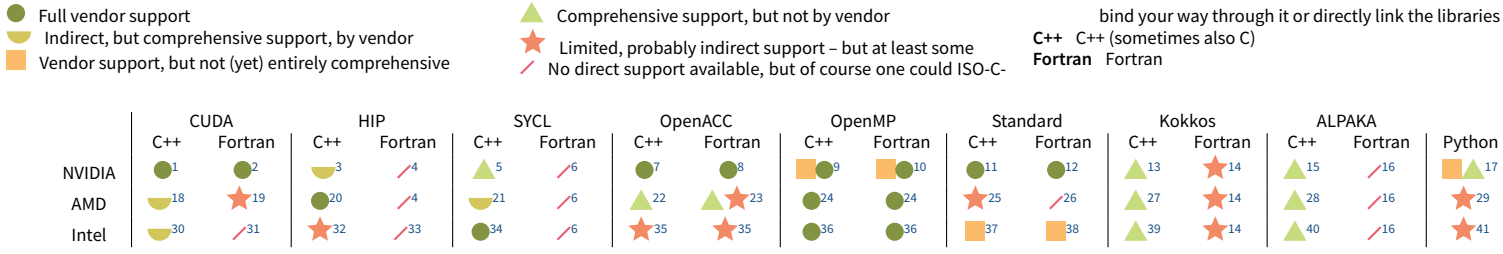


- 1: CUDA C/C++ is supported on NVIDIA GPUs through the CUDA Toolkit
- 2: CUDA Fortran, a proprietary Fortran extension, is supported on NVIDIA GPUs via the NVIDIA HPC SDK
- 3: HIP programs can directly use NVIDIA GPUs via a CUDA backend; HIP is maintained by AMD
- 4: No such thing like HIP for Fortran, but AMD offers Fortran interfaces to HIP and ROCm libraries in hipfort
- 5: SYCL can be used on NVIDIA GPUs with *experimental* support either in SYCL directly or in DPC++, or via hipSYCL
- 6: No such thing like SYCL for Fortran
- 7: OpenACC C/C++ supported on NVIDIA GPUs directly (and best) through NVIDIA HPC SDK; additional, somewhat limited support by GCC C compiler and in LLVM through Clacc
- 8: OpenACC Fortran supported on NVIDIA GPUs directly (and best) through NVIDIA HPC SDK; additional, somewhat limited support by GCC Fortran compiler and Flacc
- 9: OpenMP in C++ supported on NVIDIA GPUs through NVIDIA HPC SDK (albeit with a few limits), by GCC, and Clang; see OpenMP ECP BoF on status in 2022.
- 10: OpenMP in Fortran supported on NVIDIA GPUs through NVIDIA HPC SDK (but not full OpenMP feature set available), by GCC, and Flang
- 11: pSTL features supported on NVIDIA GPUs through NVIDIA HPC SDK
- 12: Standard Language parallel features supported on NVIDIA GPUs through NVIDIA HPC SDK
- 13: Kokkos supports NVIDIA GPUs by calling CUDA as part of the compilation process
- 14: Kokkos is a C++ model, but an official compatibility layer (*Fortran Language Compatibility Layer*, FLCL) is available.
- 15: Alpaka supports NVIDIA GPUs by calling CUDA as part of the compilation process; also, an OpenMP backend can be used
- 16: Alpaka is a C++ model
- 17: There is a vast community of offloading Python code to NVIDIA GPUs, like CuPy, Numba, cuNumeric, and many others; NVIDIA actively supports a lot of them, but has no direct product like *CUDA for Python*; so, the status is somewhere in between
- 18: hipify by AMD can translate CUDA calls to HIP calls which runs natively on AMD GPUs
- 19: AMD offers a Source-to-Source translator to convert some CUDA Fortran functionality to OpenMP for AMD GPUs (gpufort); in addition, there are ROCm library bindings for Fortran in hipfort OpenACC/CUDA Fortran Source-to-Source translator
- 20: HIP is the preferred native programming model for AMD GPUs
- 21: SYCL can use AMD GPUs, for example with hipSYCL or DPC++ for HIP AMD
- 22: OpenACC C/C++ can be used on AMD GPUs via GCC or Clacc; also, Intel's OpenACC to OpenMP Source-to-Source translator can be used to generate OpenMP directives from OpenACC directives
- 23: OpenACC Fortran can be used on AMD GPUs via GCC; also, AMD's gpufort Source-to-Source translator can move OpenACC Fortran code to OpenMP Fortran code, and also Intel's translator can work
- 24: AMD offers a dedicated, Clang-based compiler for using OpenMP on AMD GPUs: AOMP; it supports both C/C++ (Clang) and Fortran (Flang, example)
- 25: Intel's DPC++ (oneAPI) can be compiled with an experimental HIP AMD backend, allowing to launch STL algorithms to AMD GPUs; caveats from Intel's STL support apply
- 26: Currently, no (known) way to launch Standard-based parallel algorithms on AMD GPUs
- 27: Kokkos supports AMD GPUs through HIP
- 28: Alpaka supports AMD GPUs through HIP or through an OpenMP backend
- 29: AMD does not officially support GPU programming with Python (also not semi-officially like NVIDIA), but third-party support is available, for example through Numba (currently inactive) or a HIP version of CuPy
- 30: SYCLomatic translates CUDA code to SYCL code, allowing it to run on Intel GPUs; also, Intel's DPC++ Compatibility Tool can transform CUDA to SYCL
- 31: No direct support, only via ISO C bindings, but at least an example can be found on GitHub; it's pretty scarce and not by Intel itself, though
- 32: CHIP-SPV supports mapping CUDA and HIP to OpenCL and Intel's Level Zero, making it run on Intel GPUs
- 33: No such thing like HIP for Fortran
- 34: SYCL is the prime programming model for Intel GPUs; actually, SYCL is only a standard, while Intel's implementation of it is called DPC++ (*Data Parallel C++*), which extends the SYCL standard in various places; actually actually, Intel namespaces everything *oneAPI* these days, so the *full* proper name is Intel oneAPI DPC++ (which incorporates a C++ compiler and also a library)
- 35: OpenACC can be used on Intel GPUs by translating the code to OpenMP with Intel's Source-to-Source translator
- 36: Intel has extensive support for OpenMP through their latest compilers
- 37: Intel supports pSTL algorithms through their DPC++ Library (oneDPL; GitHub). It's heavily namespaced and not yet on the same level as NVIDIA
- 38: With Intel oneAPI 2022.3, Intel supports DO CONCURRENT with GPU offloading
- 39: Kokkos supports Intel GPUs through SYCL
- 40: Alpaka v0.9.0 introduces experimental SYCL support; also, Alpaka can use OpenMP backends
- 41: Not a lot of support available at the moment, but notably DPNP, a SYCL-based drop-in replacement for Numpy, and numba-dpex, an extension of Numba for DPC++.