(https://developer.nvidia.com)

CUDA Fortran

CUDA Fortran is a software compiler and tool chain for building performance optimized GPU-accelerated Fortran applications targeting the NVIDIA GPU platform.

GET STARTED (https://developer.nvidia.com/nvidia-hpc-sdk-d...

Powerful

CUDA Fortran includes language extension to simplify data management, the **!$cuf kernel** directive for automatically generating device code loops, and interface modules to all the NVIDIA CUDA-X math libraries.

Flexible

CUDA Fortran is designed to interoperate with other popular GPU programming models including CUDA C, OpenACC and OpenMP. You can directly access all the latest hardware and driver features including cooperative groups, Tensor Cores, managed memory, and direct to shared memory loads, and more.

Low Risk

CUDA Fortran is proven and supported on all major HPC platforms including x86-64, OpenPOWER and Arm-based servers. NVIDIA CUDA Fortran is available for use both on-premises and on all major cloud platforms including NGC. Commercial support options are available.

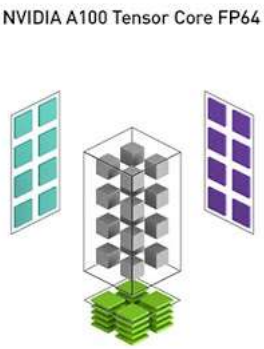
Key Features

Direct GPU Programming

CUDA Fortran is a low-level explicit programming model with substantial runtime library components that gives expert Fortran programmers direct control over all aspects of GPU programming. CUDA Fortran enables programmers to access and control all the newest GPU features including CUDA Managed Data, Cooperative Groups and Tensor Cores. CUDA Fortran includes several productivity enhancements such as Loop Kernel Directives, module interfaces to the NVIDIA GPU math libraries and OpenACC interoperability features.

CUDA FORTRAN

Access New GPU Features



CUDA Fortran gives you access to the latest CUDA features. Using CUDA Managed Data, a single variable declaration can be used in both host and device code. Variables declared with the managed attribute require no explicit data transfers between host and device. Cooperative groups provide the ability to synchronize device threads in groups larger or smaller than thread blocks. Warp-based matrix multiply accumulate operations using Tensor Cores are enabled through the wmma module which provides WMMA API functions for manipulating and operating on matrices.

CUDA Library Interfaces

CUDA Fortran provides module-defined interfaces to all the CUDA-X math libraries including cuBLAS, cuFFT, cuRAND, cuSOLVER, cuSPARSE, and cuTENSOR, as well as the NCCL and NVSHMEM communications libraries. The NVFORTRAN compiler can seamlessly accelerate many standard Fortran array intrinsics and language constructs including sum, maxval, minval, matmul, reshape, spread, and transpose on device and managed arrays by mapping Fortran statements to the functions available in the NVIDIA cuTENSOR library, a first-of-its-kind, GPU-accelerated, tensor linear algebra library providing tensor contraction, reduction, and element-wise operations. See the NVIDIA Fortran CUDA Library Interfaces (https://docs.nvidia.com/hpc-sdk/compilers/fortran-cuda-interfaces/index.html) documentation for more information.

Interoperable Programming

OpenACC
More Science, Less Programming

Combining CUDA Fortran with other GPU programming models can save time and help improve productivity. For example, you can use CUDA Fortran device and managed data in OpenACC compute constructs. Call CUDA Fortran kernels using OpenACC data present in device memory and call CUDA Fortran device subroutines and functions from within OpenACC loops. Pass device arrays moved using OpenACC data constructs directly to CUDA Fortran kernels. CUDA Fortran and OpenACC can share CUDA streams and can both use multiple devices. CUDA Fortran is also interoperable with OpenMP programs.

Kernel Loop Directive

CUDA Fortran allows automatic kernel generation and invocation from a region of host code containing one or more tightly nested loops. Launch configuration and loop mapping are controlled within the directive body using the familiar CUDA chevron syntax. CUF kernels support multidimensional arrays. The CUDA Fortran compiler recognizes a scalar reduction operation, such as summing the values in a vector or matrix, and generates the final reduction kernel, inserting synchronization as appropriate.

```
!$cuf kernel do <<< *,* >>>
```

User Quotes

> CUDA Fortran gives us the full performance potential of the CUDA programming model. While leveraging the potential of explicit data movement, !$CUF KERNELS directives give us productivity and source code maintainability. It's the best of both worlds.
>
> — Filippo Spiga, Senior Contributor, Quantum ESPRESSO Group

Resources

> HPC SDK Docs (https://docs.nvidia.com/hpc-sdk/compilers/index.html)

> CUDA Fortran Programming Guide (https://docs.nvidia.com/hpc-sdk/compilers/cuda-fortran-prog-guide/index.html)

> Blog: Using Tensor Cores with CUDA Fortran (https://developer.nvidia.com/blog/using-tensor-cores-in-cuda-fortran/)