GPU Programming Models

Subil Abraham

HPC Engineer – User Assistance

Frontier Workshop

August 21, 2023

ORNL is managed by UT-Battelle LLC for the US Department of Energy

Models We Will Cover

Focusing on brief overview of each and how to compile on Crusher:

- Kokkos
- OpenMP Offload
- HIP

Follow along with examples: https://github.com/olcf/frontier_gpu_programming_models_examples

=

HIP

What is HIP?

- AMD’s API for GPU programming.
- Gives low level control (relative to other models I will talk about) to write code for computing on GPUs
- Almost 1 to 1 replacement of CUDA (cudaAbcCall -> hipAbcCall)
  - Includes replacements for some CUDA libraries like cufft (hipfft) and cublas (hipblas)
  - Some CUDA calls not supported, because they are deprecated or not yet implemented for HIP
- Existing tools (hipify-perl, hipify-clang) for converting your CUDA code to HIP
- Uses .hip file extension (can use the .cpp file extension too, just let your build system know)

Example: parallelizing a for loop

```
int a[N];
for(int i = 0; i<N; i++) {
  a[i] = i+2;
}
```

```
__global__ void fill_array(int *a) {
  int i = blockDim.x * blockIdx.x +
          threadIdx.x;
  a[i] = i + 2;
}
...
int *a;
hipMalloc(&a, N*sizeof(int));
fill_array<<<dim3(N/256), dim3(256),
0, hipStreamDefault>>>(a);
```

This is if you were writing your own HIP kernel. There are also a lot of prebuilt functionality in libraries like hipblas and hipfft. You may not need to write that matrix multiplication routine by hand!

Things to Note

- For PrgEnv-cray and PrgEnv-gnu, make sure you also load amd-mixed to get the HIP libraries. Not needed for PrgEnv-amd.
- Load craype-accel-amd-gfx90a for PrgEnv-cray and PrgEnv-amd if using the CC compiler wrapper for compiling HIP
- No native Fortran API. You have to write your GPU code in C++ and import it to Fortran through ISO_C_binding
  - AMD also provides hipfort library with a bunch of those bindings made for you
- For CMake:
  - Can use project(myproj LANGUAGES HIP)
  - Can only default ROCm clang compiler used in `hip-lang-config.cmake`. CC won't work. (Can get around this by setting .cpp file extension for hip files and use CC by using the legacy cmake steps)
  - Make sure to set –DCMAKE_HIP_ARCHITECTURES="gfx90a"

Resources

- Documentation: https://rocm.docs.amd.com/en/latest/
- Basic tutorial if you have no CUDA knowledge (work in progress)
  - https://github.com/olcf-tutorials/HIP_from_scratch
- HIP Training Series (currently ongoing, sign up now!): https://www.olcf.ornl.gov/hip-training-series/
- HIP Tutorial if you're already familiar with CUDA
  - olcf page: https://www.olcf.ornl.gov/calendar/hip-for-cuda-programmers/
  - repo: https://github.com/olcf/HIP_for_CUDA_programmers
- hipfort (HIP bindings for fortran)
  - https://github.com/ROCmSoftwarePlatform/hipfort
  - talk on hipfort: https://www.olcf.ornl.gov/calendar/hip-for-cuda-programmers/ (see Presentation and Repository section)
- hipify (tool to convert cuda code to hip)
  - https://github.com/ROCm-Developer-Tools/HIPIFY (already available in the amd-mixed and amd modules)
- HIP-CUDA API support table
  - https://github.com/ROCm-Developer-Tools/HIPIFY#cuda-apis
- Cuda training series (most of the knowledge still applies for HIP)
  - https://www.olcf.ornl.gov/cuda-training-series/
- CMake for ROCm documentation:

https://rocm.docs.amd.com/en/latest/understand/cmake_packages.html

HIP Training Series

- Teach HIP from basics, how to convert from CUDA, profiling and debugging
- Open to current Frontier and Perlmutter users
- 1 hour lecture, 1 hour hands on. All will be recorded.
- Currently five sessions, with possibly more in the future.
- Includes info on using HIP on Nvidia GPUs, and how to do portable build scripts for HIP code for Nvidia and AMD
- https://www.olcf.ornl.gov/hip-training-series/

OpenMP Offload

What is OpenMP?

- OpenMP is the standard for thread based parallelism on shared memory systems
- Code looks like normal serial code, with directives annotating the code to give hints on how to parallelize.

```
int a[N];
#pragma omp parallel for
for(int i = 0; i<N; i++) {
  a[i] = i+2;
}
```

What is OpenMP Offload?

- Offload was introduced in OpenMP 4.0 standard
  - New directives to offload data and computation to devices like GPUs
- Directives specified as comments in Fortran, and #pragma in C
  - Supported compilers will determine how to parallelize the code based on your directives
  - If compiler doesn’t support, it will fallback to compiling for normal serial.
- Offload will take care of transferring data from host to device, perform compute on device, and transfer data back to host.
  - Based on the directives you specify

Example: parallelizing a for loop

```
int a[N];
for(int i = 0; i<N; i++) {
  a[i] = i+2;
}
```

```
int a[N];
#pragma omp target teams distribute parallel for
for(int i = 0; i<N; i++) {
  a[i] = i+2;
}
```

```
// fortran would look like
!$omp target teams distribute parallel do
<do loop>
!$omp target teams distribute parallel do
```

Things to Note

- GCC currently doesn't support offloading for MI250X accelerators yet. Only Cray and AMD.
- Clang based compilers (Cray, AMD) don't support loop directives yet.
- When compiling with hipcc for the examples, you get "loop not vectorized" warnings from the LLVM optimizer because hipcc add –O3 by default

Resources

OpenMP offload tutorial series from OLCF (includes Summit instructions):

- https://github.com/olcf/openmp-offload
- https://www.olcf.ornl.gov/calendar/introduction-to-openmp-offload-part-1/
- https://www.olcf.ornl.gov/calendar/introduction-to-openmp-offload-part-2/
- https://www.olcf.ornl.gov/calendar/preparing-for-frontier-openmp-part3/

text tutorial: https://enccs.github.io/openmp-gpu/

Kokkos

What is Kokkos?

- C++ library for offloading onto various backends (CUDA, OpenMP, HIP, potentially others)
- Unlike others, not part of the compiler. You manage the source (or module load it)
- Aims to be descriptive, not prescriptive
  - Less fine grained control, but fewer footguns
  - maps work to resources
- Many different backends supported, including HIP for GPU and OpenMP on CPU (as well as serial)
- Influences and is influenced by the C++ standard
- Primarily developed by Sandia, a number of applications written
- RAJA is similar : https://raja.readthedocs.io/en/develop/index.html

Example: parallelizing a for loop

```
int a[N];
for(int i = 0; i<N; i++) {
a[i] = i + 2;
}
```

```
// defaults to allocating and
// running on GPU if
// compiled for GPU
Kokkos::View<double*> a( “a”, N );
Kokkos::parallel_for(“label”, N,
  KOKKOS_LAMBDA(int i) {
    a( i ) = i + 2;
  }
);
```

Things to Note

- For PrgEnv-cray and PrgEnv-gnu, make sure you also load amd-mixed to get the HIP libraries. Not needed for PrgEnv-amd.

Resources

- Tutorial repo: https://github.com/kokkos/kokkos-tutorials
- Condensed short tutorial video: https://www.youtube.com/watch?v=6Ts6k2Nas5w (slides: https://github.com/kokkos/kokkos-tutorials/tree/main/Intro-Short)
- Long tutorial (slides also in the github) modules 1-8: https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series
- main documentation: https://kokkos.github.io/kokkos-core-wiki/index.html
- Kokkos source code on Github: https://github.com/kokkos/kokkos

Questions?

- Email help@olcf.ornl.gov