rocFFT

An open-source GPU FFT Library for Exascale Systems

Malcolm Roberts, Fei Zheng, and Bragadeesh Natarajan
malcolm.roberts@amd.com
2020-02-14

Outline

- ROCm: Radeon Open Compute Platform
- The `HIP` programming language
- `rocFFT`
  - Design
  - Features
  - Usage
- Sample applications

ROCm: Radeon Open Compute

https://rocm.github.io/

- Open source drivers and libraries
  - Community engagement encouraged!
- HPC and machine learning applications
- Focus on multi-gpu computation
- Some libraries of interest to the HPC community:
  - rocBLAS
  - rocSPARSE: sparse BLAS
  - rocALUTION: sparse solvers
  - rocFFT
- Written in the HIP programming language

HIP: Heterogeneous-Computing Interface for Portability

HIP is a C++ dialect for GPU programming.

Sample differences between HIP, CUDA, and OpenCL:

Term	HIP	CUDA	OpenCL
Device kernel	`__global__`	`__global__`	`__kernel`
Thread index	`threadIdx.x`	`threadIdx.x`	`get_local_id(0)`
Kernel launch	`hipLaunchKernelGGL`	`<<< >>>`	`clEnqueueNDRangeKernel`

The hipify tool converts CUDA code to HIP code.

OpenCL continues to be supported.

Compiler is a branch of `clang`, and will be upstreamed.

rocFFT: An open-source GPU FFT Library for Exascale Systems

`github.com/ROCmSoftwarePlatform/rocFFT`

- Open source:
- Written in `C++` and `HIP`
- can also be compiled with `nvcc`
- `rocfft` interface provides flexibility
- Alternative `hipfft` interface: usage similar to `cufft`

rocFFT: design

`rocfft.h` provides a `C` interface.
Based on the input parameters, we create a tree structure:

- Input and output buffers, strides, etc are assigned recursively.
- Leaf nodes contain device kernels.
- A plan is executed by traversing the leaf nodes sequentially.

Most kernels are generated.

Kernel types:

- Stockham
- Transpose
- Bluestein
- Real/complex stages.

The host code is stored in `librocfft.so`; device kernels are in `librocfft-device.so`.

rocFFT: features

- 1D, 2D, and 3D, transforms.
- Batched transforms.
- Single and double precision.
- In-place and out-of-place transforms.
- Input and output strides.
- Complex data format: interleaved or planar (aka “split”).
- Extensive testing on several Linux distributions.

rocFFT: usage

The `rocfft` interface is defined in `rocfft.h`.

1. `rocfft_setup()`
2. Optional: create a `rocfft_plan_description` for advanced stride and other format choices.
3. Create a `rocfft_plan`, possibly using the description above.
4. Create a `rocfft_execution_info`
   - 4.1 Get the scratch memory requirements of the `rocfft_plan`
   - 4.2 Allocate the work buffer and associate it to the `rocfft_execution_info`
5. Allocate the input and output buffers and set up the input data.
6. Execute the plan with `rocfft_execute`.
7. Free the buffers, destroy the `rocFFT` structures, and `rocfft_cleanup()`.

rocFFT: rocfft interface example

```
// create the plan:
rocfft_plan_create(&plan,
                   rocfft_placement_inplace,
                   rocfft_transform_type_complex_forward,
                   rocfft_precision_double,
                   1,       // dimension
                   &length, // transform lengths (column-major)
                   1,       // batch size
                   NULL);   // optional description

// allocate the work memory and associate it to the execution info:
rocfft_execution_info_create(&planinfo);
rocfft_plan_get_work_buffer_size(gpu_plan, &wsize);
hipMalloc(&wbuffer, wsize);
rocfft_execution_info_set_work_buffer(planinfo, wbuffer, wsize);

// execute the plan:
rocfft_execute(gpu_plan, gpu_in, planinfo);
```

rocFFT: hipfft interface example

```
// Create the plan:
hipfftPlan1d(&plan,      // plan handle
             Nx,         // transform lengths
             HIPFFT_Z2Z, // transform type (HIPFFT_C2C for float)
             1);         // number of transforms
// Can also call hipfftPlanMany (row-major)

// Work memory is automatically created and associated with the plan.

// Execute the plan:
hipfftExecZ2Z(plan, x, x, direction);
```

Frontier CAAR Projects

Some CAAR (Center for Accelerated Application Readiness) applications which target Frontier and which use FFTs (`www.olcf.ornl.gov/caar/frontier-caar`):

- GESTS (GPUs for Extreme-Scale Turbulence Simulations)
  High-resolution pseudospectral code: 35 trillion grid points.
- NAMD (Nanoscale Molecular Dynamics)
  Molecular simulator; can use rocFFT as the backend.
- PIConGPU (Particle-in-cell on Graphics Processing Units)
  Plasma simulator: PIC methods often use FFTs to resolve the underlying Maxwell equations.

Conclusion

rocFFT is:

- An open-source GPU FFT library from AMD
- Part of ROCm: the Radeon Open Compute Platform
- Uses the HIP programming language
- Will be used by a number of applications for exascale computing.

Available at: `github.com/ROCmSoftwarePlatform/rocFFT`