OAK RIDGE National Laboratory

The Frontier Programming Environment at OLCF

Wael Elwasif

Frontier Programming Environment Task Lead
Oak Ridge Leadership Computing Facility

Oak Ridge National Laboratory

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

U.S. DEPARTMENT OF ENERGY

OAK RIDGE National Laboratory

Overview

Contributors to Frontier Programming Environment

Vendor-Provided

- Cray Programming Environment (CPE)
  - Includes Cray compiler for C, C++, and Fortran plus GCC compiler. All the Cray profiling, tuning, and debugging tools. OpenMP and Cray MPI optimized for AMD GPU direct.
- AMD ROCm programming environment
  - Includes LLVM compiler to generate optimized code for both the AMD Trento CPU and MI250X GPU.
  - Support: C, C++, and Fortran and have GPU offload support. HIP, a CUDA-like direct GPU programming model (with CUDA to HIP conversion utilities).

Other Sources

- ECP
  - LLVM enhancements: Flang (Fortran front-end), OpenMP, OpenACC
  - Kokkos and RAJA
  - HIP LZ (HIP support for Aurora)
  - MPI, HPCToolkit, PAPI enhancements
  - …
- ALCF + OLCF
  - Pilot implementation of DPC++/SYCL for Frontier
- OLCF
  - GCC enhancements to better support OpenACC, OpenMP, Fortran on Summit and Frontier

Programming Environment

- Compilers Offered
  - Cray PE (C/C++ LLVM-based; Cray Fortran)
  - AMD ROCm (LLVM-based)
  - GCC
- Programming Languages & Models Supported (in which compilers)
  - C, C++, Fortran (all)
  - OpenACC (Cray Fortran OpenACC 2.0+ & GCC 2.6 substantially complete, 2.7 planned)
  - OpenMP (all) 5.0-5.2 in progress – most priority features complete, details vary
  - HIP (Cray, AMD)
  - Kokkos/RAJA (all)
  - UPC (Cray, GCC)
- Transition Paths
  - CUDA: semi-automatic translation to HIP
  - CUDA Fortran: HIP kernels called from Fortran (a more portable approach)
    - CUDA Fortran kernels need to be translated to C++/HIP (manual process)
    - Fortran bindings to HIP and ROCm libraries and HIP runtime available through AMD's hipfort project

Items in green are also available on Summit

Programming Tools

Debuggers and Correctness Tools

Tool
System-Level Tools
Arm DDT
Cray CCDB
Cray ATP
STAT
Node-Level Tools
ROCgdb
Cray GDB4HPC

Performance Tools

Tool
System-Level Tools
Arm MAP/Performance Reports
CrayPat/Apprentice2 (Cray)
Reveal (Cray)
TAU
HPCToolkit
Score-P / VAMPIR
Node-Level Tools
gprof
PAPI
ROCprof
ROC-profiler & ROC-tracer libraries

Items in green are also available on Summit

Scientific Libraries and Tools

Functionality	CPU	GPU	Notes
BLAS	**Cray LibSci**, **AMD BLIS**, PLASMA	**Cray LibSci_ACC**, **AMD roc/hipBLAS**, AMD rocAMD ROCm Tensile, **MAGMA**	MAGMA and PLASMA are open source software led by the UTK Innovative Computing Laboratory
LAPACK	**Cray LibSci**, **AMD libFlame**, PLASMA	**Cray LibSci_ACC**, **AMD roc/hipSolver**, **MAGMA**	
ScaLAPACK	Cray LibSci	ECP SLATE, Cray LibSci_ACC	
Sparse		**AMD roc/hipSparse**, AMD rocALUTION	
Mixed-precision iterative refinement	Cray IRT, MAGMA	MAGMA	
FFTW or similar	**Cray**, **AMD**, ECP FFTX, FFT-ECP	**AMD rocFFT**, ECP FFTX, FFT-ECP	FFT-ECP focuses on 3D FFTs
PETSc, Trilinos, HYPRE, SUNDIALS, SuperLU			Spack recipes from ECP xSDK

Functionality in **green** is also available on Summit

OAK RIDGE National Laboratory

Digging a Little Deeper

For C/C++ Codes

- Multiple compilers available
  - AMD
  - Cray
  - LLVM
- But they're all based on LLVM
  - HPE and AMD are among the many organizations contributing to the development of LLVM
  - Most work is "upstreamed" (contributed to the core LLVM source)
    - But not everything is accepted (immediately), or may be held back as proprietary
  - Capabilities (and bugs) are likely to be generally similar at any point in time…
  - But not identical!
- LLVM is also available on Summit

Upstream LLVM @ OLCF

- Summit:
  - OLCF deployed modules (with offloading): latest **`llvm/14.0.0`**
  - Periodic **`main`** snapshots :

```
module use /sw/summit/modulefiles/ums/stf010/Core
module load llvm/17.0.0-latest   # Also from specific dates
```

- Crusher:
  - Periodic **`main`** snapshots (maintained by the ECP SOLLVE project)

```
module use /sw/crusher/ums/ums012/modules
module load llvm/17.0.0-20230213  # Also from other dates
```

- Frontier :
  - **TBD**

For Fortran Codes

- One useful compiler available at present
  - Cray
    - *Not* based on LLVM
- AMD provides a Fortran implementation, but we don't recommend it
  - It is based on "classic Flang", in the LLVM ecosystem
  - Support for both the latest language standards and OpenMP offload are limited
- There is extensive work underway in the LLVM community on Flang, but it will be some time before it is production quality

But What About GCC?

- On this slide “GCC” refers to the whole suite, including gfortran
  - With support for offloading using OpenMP/OpenACC
- OLCF is working with Siemens to implement OpenMP in GCC
- OLCF will provide recent release and development versions of GCC on Frontier
- For various reasons, you should *not* expect gcc-generated executables to be performant for offload at this time
  - Results will vary
  - We are interested in improving the performance of gcc. If you have a troublesome case, reach out to me. (No guarantees, however)
- GCC is also available on Summit

GCC+offloading

- Summit:
  - OLCF deployed modules (with offloading): latest **`gcc/12.1.0`**
  - Periodic development snapshots :

```
module use /sw/summit/modulefiles/ums/stf010/Core
module load gcc/12.2.1-latest   # Also from specific dates
```

- Crusher:
  - Periodic development snapshots

```
module use /sw/crusher/ums/compilers/modulefiles
module load gcc/12.2.1-latest  # Also from specific dates
```

- Frontier :
  - **TBD**

For HIP (and CUDA) Codes

- HIP runs today on AMD and NVIDIA GPUs
- An ECP project is working on supporting HIP on Intel GPUs
- Recommend a one-time translation of CUDA codes to HIP and make the HIP version primary from then on
- Both Cray and AMD compilers support HIP
  - They both use the AMD runtime
- More on HIP available in the OLCF Training Archive
- HIP is also available on Summit

- AMD provides tools to translate CUDA to HIP
  - **`hipify-perl`** and **`hipify-clang`**
  - Not fully automatic
- **`hipfort`**
  - https://github.com/ROCmSoftwarePlatform/hipfort
  - Fortran bindings to HIP and ROCm libraries and HIP runtime
  - Build depends on ROCm version & Fortran compiler used
- Related: SYCLomatic - Intel tool to translate CUDA to Sycl
  - https://github.com/oneapi-src/SYCLomatic
  - Intel® DPC++ Compatibility Tool
  - Not fully automatic

For OpenMP Codes

- OpenMP is very much a work in progress in the LLVM community
  - Most of 5.0 is implemented
  - Parts of 5.1, 5.2 are implemented
- We (DOE labs, including ORNL/OLCF) are trying to help prioritize the order of implementation based on what users tell us they need/want
  - So if you could really use features that aren't available yet, please let us know!
- Cray and AMD compilers use different OpenMP runtimes
- Remember that Cray Fortran is not based on LLVM
- OpenMP implementation in GCCis also a work in progress
- More on OpenMP available in the OLCF Training Archive
- OpenMP is available on Summit, but different progress on impl

For OpenACC Codes

- Cray Fortran supports OpenACC 2.0+
  - *"CCE supports full OpenACC 2.0 and partial OpenACC 2.x/3.x for Fortran (OpenACC is not supported for C and C++)"*
  - Work is underway to 3.2 (latest)
    - but no timeline has been given
- OLCF provides OpenACC support via GCC
  - 2.6 currently supported --- 2.7 planned
  - 3.x not currently planned – let us know if there are particular features that you could really use
  - Don't expect this to be performant at present
- Work is also underway in the LLVM community on OpenACC
  - Unknown when these will be production
- OpenACC is also available on Summit

CCE OpenACC 2.x/3.x features – CCE/15:

- **atttach/detach** behavior and clauses
- **`default(present)`** clause
- Implied present-or behavior for **`copy`**, **`copyin`**, **`copyout`**, and **`create`** data clauses
- **`if_present`** clause on **`acc update`**
- **`if`** clause on **`acc wait`**
- **`async`** and **`wait`** clauses on **`acc data`**
- **`acc_attach`** and **`acc_attach_async`** APIs
- **`finalize`** clause on **`exit data`**
- **`no_create`** clause on structured data and compute constructs
- if clause on **`host_data`**

What about SYCL?

- OLCF and ALCF have partnered with Codeplay on a pilot implementation of the Intel DPC++ compiler for AMD GPUs
  - ALCF has also partnered with NERSC on NVIDIA support
- Pilot implementation is complete
  - ~"50%" level of support
  - Tested with a small set of benchmarks and mini-apps
  - Spock: **`module load ums ums015 dpcpp`**
  - Crusher: **`module load dpcpp`** ***`#dpcpp/22.09`***
- Seeking interested users to try out the pilot implementation
  - Provide feedback
  - Shake out issues
  - Provide motivation to complete the port

Help Us Help You…

- If you have a liaison, work with them
- ***If you encounter an issue, file a ticket with OLCF*** – otherwise the facility won't (necessarily) know about it, and can't track it
  - Summit, Spock, Crusher, Frontier…
- Take advantage of training events like this one
  - *Preparing for Frontier* series in the OLCF Training Archive
  - If you have Crusher access: office hours, hackathons, additional trainings