OAK RIDGE National Laboratory

OpenACC Support in LLVM

Joel E. Denny, Valentin Clement,
Seyong Lee, Jeffrey S. Vetter

Oak Ridge National Laboratory

https://csmd.ornl.gov/project/clacc

September 14, 2021: OpenACC Summit

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

U.S. DEPARTMENT OF ENERGY

Acknowledgement

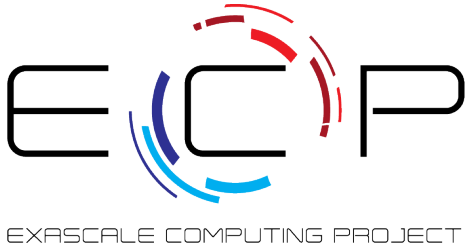
This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources of the Experimental Computing Laboratory (ExCL) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

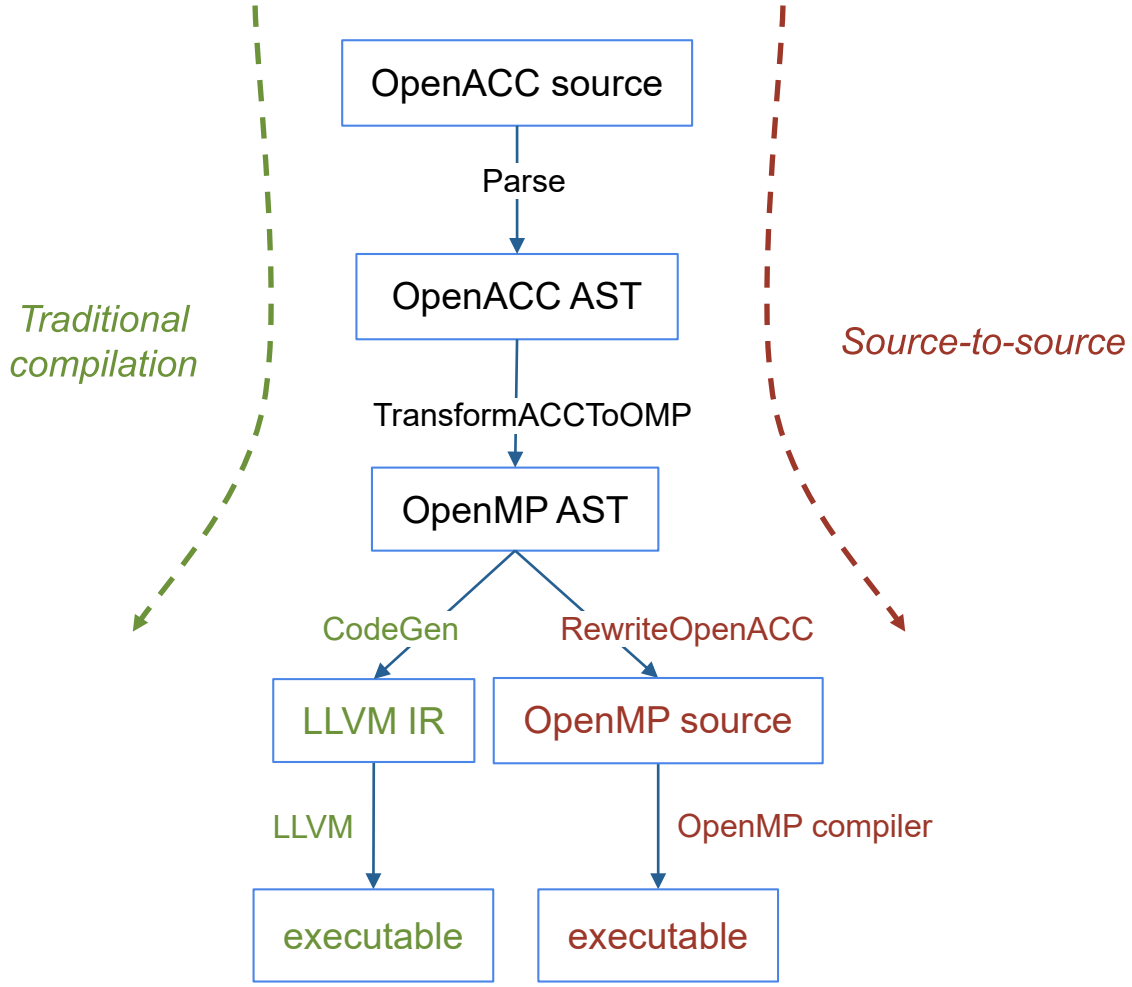
What is Clacc?

- Goal
  - OpenACC C/C++ support for Clang and LLVM
- Design
  - Translate OpenACC to OpenMP to build on OpenMP support
- Availability
  - Web page: https://csmd.ornl.gov/project/clacc
  - Source code: https://github.com/llvm-doe-org/llvm-project/wiki
- Funding
  - Exascale Computing Project (ECP)
- Contact
  - Joel E. Denny (dennyje@ornl.gov)



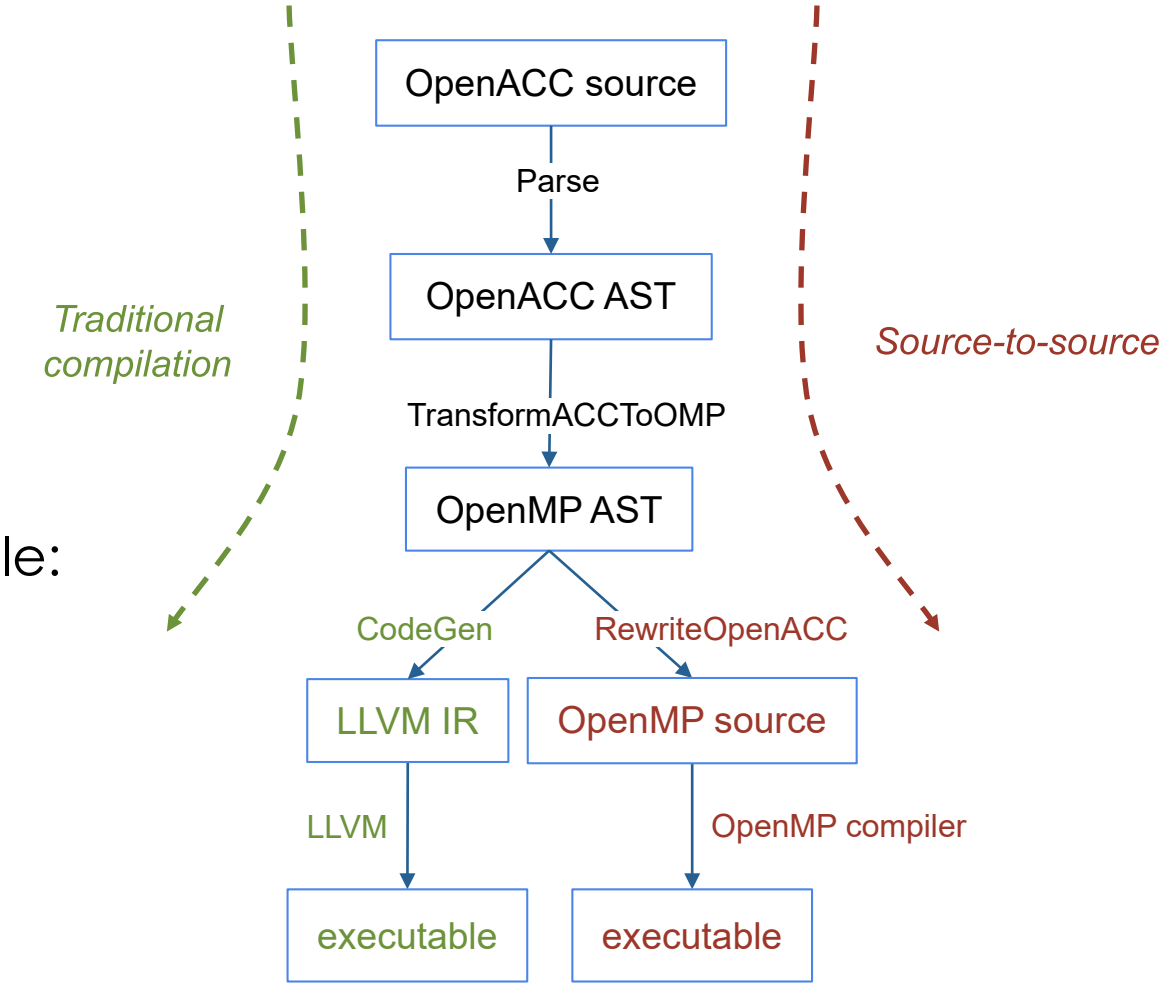
Clacc: Two Compilation Modes

- Traditional compilation
  - OpenACC source → executable
    - Similar to NVHPC or GCC
  - OpenMP serves as an internal IR
    - Diagnostics and profiling data expressed in terms of original OpenACC source not OpenMP
    - Maximizes reuse of OpenMP implementation
- Source-to-source
  - OpenACC source → OpenMP source
    - Target other OpenMP compilers and tools
    - Port apps or benchmarks
  - Uses Clang's Rewrite facility
    - Remains human-readable
    - Appropriate for other OpenMP compilers, perhaps targeting other architectures



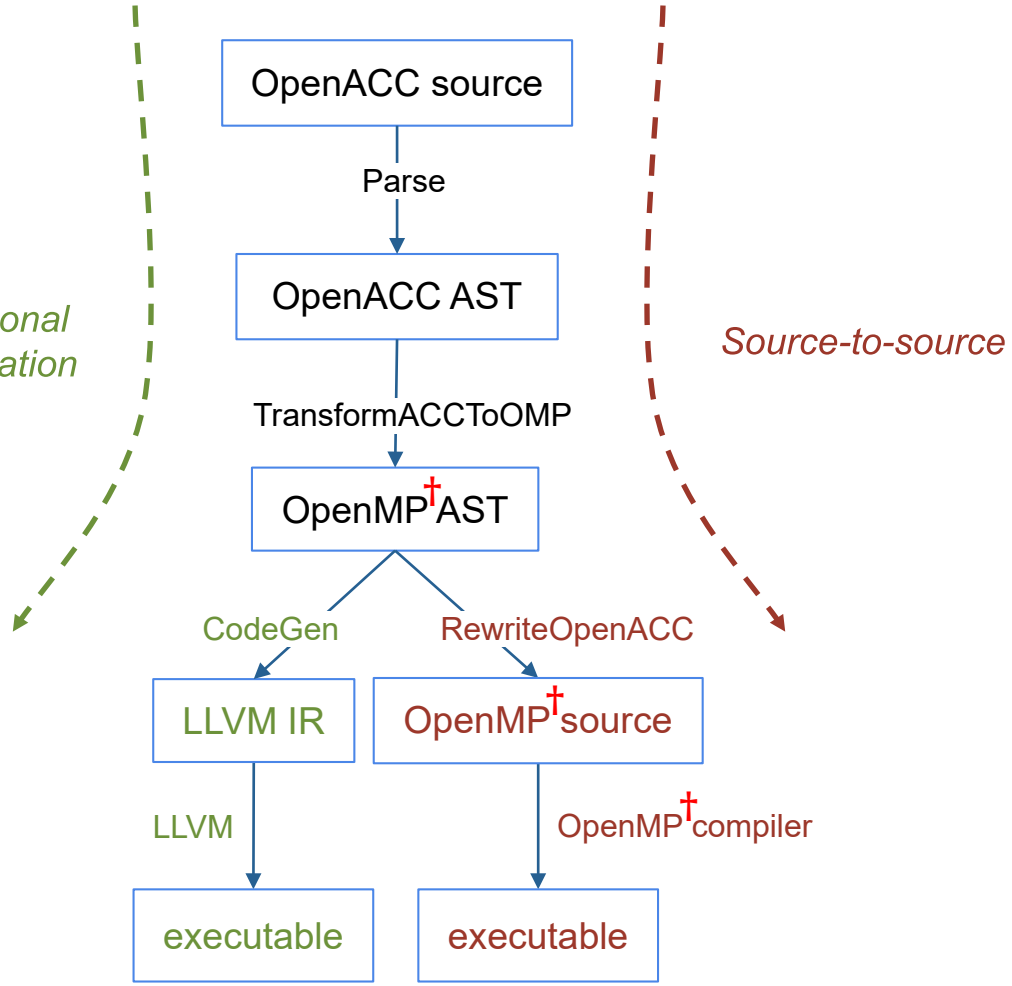
Clacc: Does OpenMP have what OpenACC needs?

- Unrepresentable individual behaviors. For example:
  - Reference counters for device allocations
    - OpenACC has two: structured and dynamic
    - OpenMP has one: dynamic
  - `no_create` clause
- Unrepresentable range of behaviors, each of which is individually representable. For example:
  - `auto` clause
  - `kernels` construct



Clacc: Solution is OpenMP Extensions†

- Clean design
  - Supports unrepresentable individual behaviors and unrepresentable ranges of behaviors
  - Supports traditional compilation mode and source-to-source mode (with a caveat we'll discuss next)
  - Distinct OpenACC vs. OpenMP representations with full translation in one compiler phase
  - Complex analyses and transformation passes can be implemented on LLVM IR instead of Clang AST
- Improves OpenMP
  - Following OpenACC's history, leads to contributions to the OpenMP specification
  - Encourages prototyping new OpenMP features (for OpenACC support) before standardizing



Clacc: User Impact of OpenMP Extensions

- Traditional compilation mode
  - OpenMP is just an internal IR
  - Clacc compiler quietly uses OpenMP extensions where needed
- Source-to-source mode
  - Compile-time error diagnostic if translation uses OpenMP extension
  - Option to disable diagnostic
    - Useful if OpenMP compiler supports extension
  - Option to convert error to warning
    - Useful to find all occurrences to manually adjust
  - Option to choose alternative, good enough translation to standard OpenMP
    - User not compiler must verify if it's good enough per application
    - Can be used in traditional compilation mode to help test the alternative translation

Clacc: OpenACC Runtime Library and Profiling Interface

- Again, build on OpenMP plus extensions
  - Clearly defined relationship between OpenACC and OpenMP representations
  - Provides support for source-to-source and using other OpenMP runtimes
  - Following OpenACC's history, leads to contributions to the OpenMP specification
- libacc2omp: OpenACC runtime
  - Wrapper around OpenMP runtime
  - Currently tested with LLVM's OpenMP runtime only
  - Carefully defined interfaces to facilitate extending support to other OpenMP runtimes in the future
- OpenACC Runtime Library Routines
  - libacc2omp wraps OpenMP runtime library routines plus original extensions
- OpenACC Profiling interface
  - libacc2omp wraps OpenMP's OMPT plus original extensions
- OpenACC Environment Variables
  - libacc2omp requires OpenMP runtime to call handler routines provided by libacc2omp
- Clacc's compiler currently does *not* translate runtime library routine calls or profiling libraries to OpenMP

Clacc: Development Status

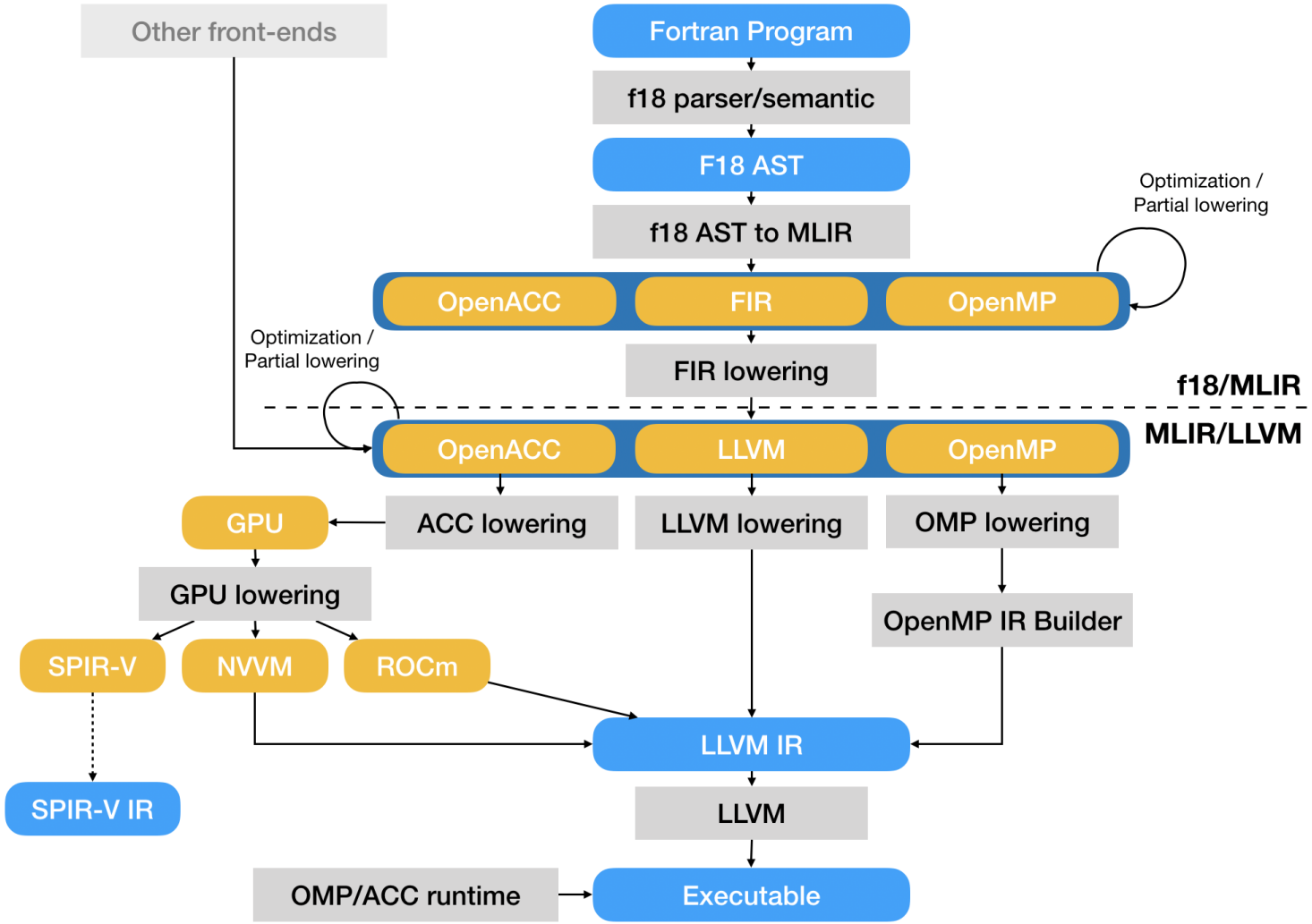
The most noteworthy new efforts from the past year are shown in **bold**

- Directives
  - Basic features are supported (e.g., data, parallel, loop directives, **acc routine seq**)
  - Some important features missing (e.g., kernels directive, C++ support, async/wait)
- **Runtime Library Routines, Preprocessor, Environment Variables**
  - **Most features supported (e.g., async/wait routines are missing)**
- Profiling interface
  - All events supported except wait events
  - Some profiling data missing (e.g., kernel_name, num_gangs, num_workers, vector_length)
  - **var_name support**
  - **Integrated with TAU**
- Supported Architectures
  - x86_64, Power 9, NVIDIA GPU
  - **AMD CPU and GPU underway**
- Ongoing activities
  - CI testing on ORNL's ExCL cluster and **Ascent (Summit training system)**
  - **Issue tracking on LLVM DOE Fork in github**
    - **Identifies potential external contributions**
  - Upstream Clang/LLVM
    - Merging into Clacc
    - **Contributed dual ref count support: OpenMP ompx_hold map type modifier extension**
    - Contributing various other improvements
  - Contributing improvements to OpenACC spec (eventually OpenMP spec)

What is Flacc?

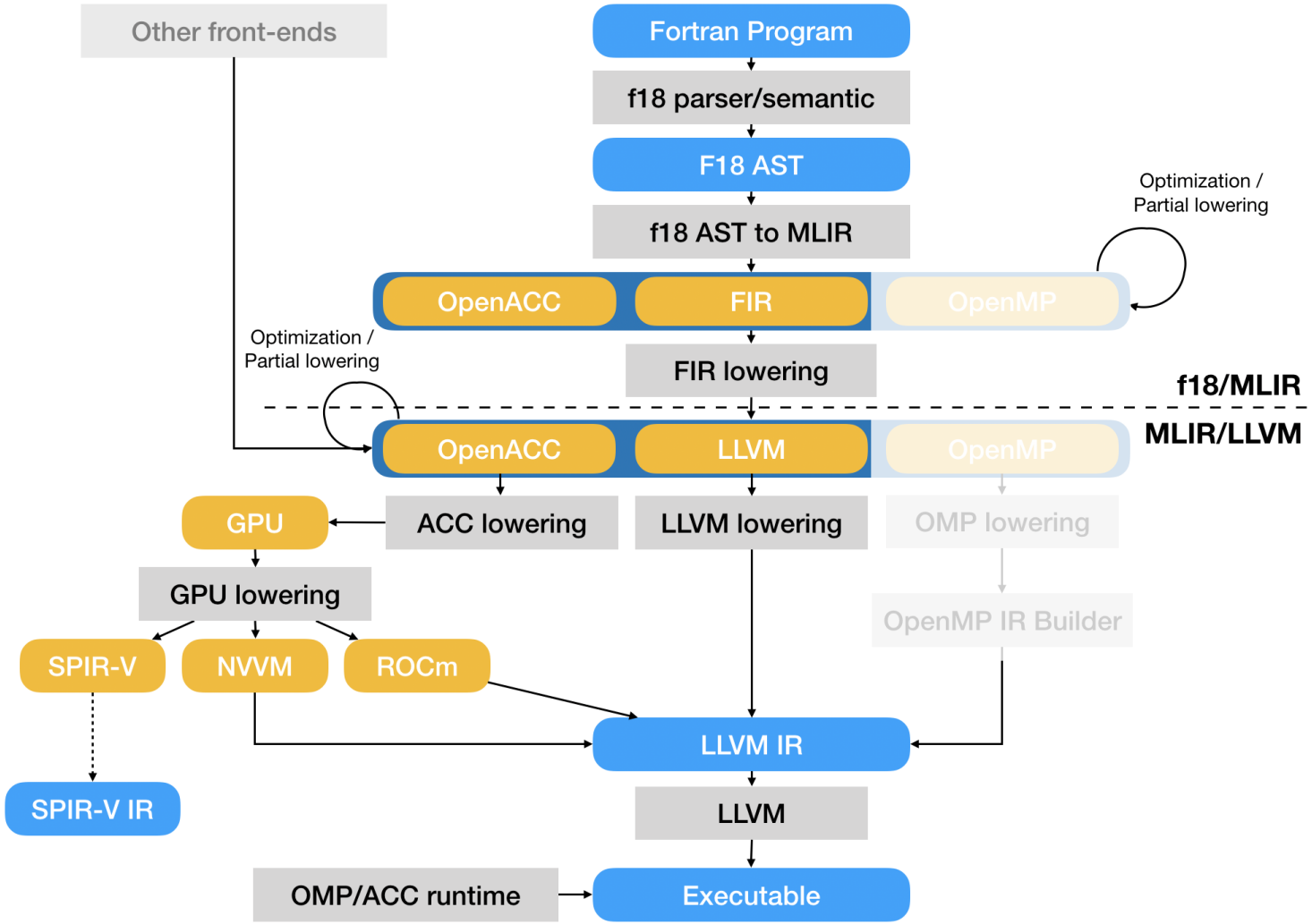
- Goal
  - OpenACC Fortran support for upstream LLVM Flang
- Design
  - Lowers OpenACC to mix of FIR and OpenACC dialects in MLIR
- Availability
  - Upstreamed to LLVM Flang as developed
- Funding
  - Exascale Computing Project (ECP)
- Contact
  - Valentin Clement

Flacc: Progressive Lowering



- F18 = older name for upstream LLVM Flang
- Mix of FIR and OpenACC dialects
- Optimizations can happen at multiple levels
- Multiple approach possible to lower to LLVM IR
  - Using lower level dialect like the GPU dialect
  - Leveraging the work in the OpenMP IR Builder

Flacc: Progressive Lowering



- F18 = older name for upstream LLVM Flang
- Mix of FIR and OpenACC dialects
- Optimizations can happen at multiple levels
- Multiple approach possible to lower to LLVM IR
  - Using lower level dialect like the GPU dialect
  - Leveraging the work in the OpenMP IR Builder

Clacc and Flacc: Planned Implementation Reuse

- OpenACC Directive Support in Compiler
  - Clacc is based on Clang (no MLIR)
  - Flacc is based on Flang (targets MLIR)
  - Little opportunity for reuse
- OpenACC Directive Support in Runtime
  - Both use LLVM's OpenMP runtime
  - Both benefit from Clacc's OpenMP runtime extensions (e.g., dual ref count)
- Clang's OpenMP Directive Extensions
  - E.g., ompx_hold to select 2nd ref count
  - Clacc directly benefits
  - Flacc benefits indirectly because they are used to test OpenMP runtime extensions
- OpenACC Runtime Library Routines
  - libacc2omp's C routines directly support Clacc
  - Expect that libacc2omp will grow Fortran routines that wrap the C routines
- OpenACC Profiling Interface
  - C interface for OpenACC profiling libraries (e.g., TAU) not for OpenACC apps
  - OpenACC app language is irrelevant
  - Expect Flacc to reuse libacc2omp's support from Clacc w/o modification
- OpenACC Environment Variables
  - Expect Flacc to reuse libacc2omp's support from Clacc w/o modification

Takeaways

- Clacc
  - OpenACC C/C++ support for Clang/LLVM
  - Builds on OpenMP plus extensions
  - Two compilation modes: traditional and source-to-source
- Flacc
  - OpenACC Fortran support for Flang/LLVM
  - Lowers to MLIR dialects
  - Plan to reuse Clacc's runtime and profiling support
- Join Us
  - Oak Ridge National Laboratory
  - Hiring interns, postdocs, research and technical staff
  - External collaborators welcome

- URLs
  - Web: https://csmd.ornl.gov/project/clacc
  - Source: https://github.com/llvm-doe-org/llvm-project/wiki
  - Email: dennyje@ornl.gov
- Publications
  - OpenACC Profiling Support for Clang and LLVM using Clacc and TAU, Camille Coti, Joel E. Denny, Kevin Huck, Seyong Lee, Allen D. Malony, Sameer Shende, and Jeffrey S. Vetter, ProTools, GA, USA (November 2020)
  - Clacc: Translating OpenACC to OpenMP in Clang, Joel E. Denny, Seyong Lee, and Jeffrey S. Vetter, 2018 IEEE/ACM 5th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC), Dallas, TX, USA, (November 2018).