PGI® COMPILERS & TOOLS

Flang Update

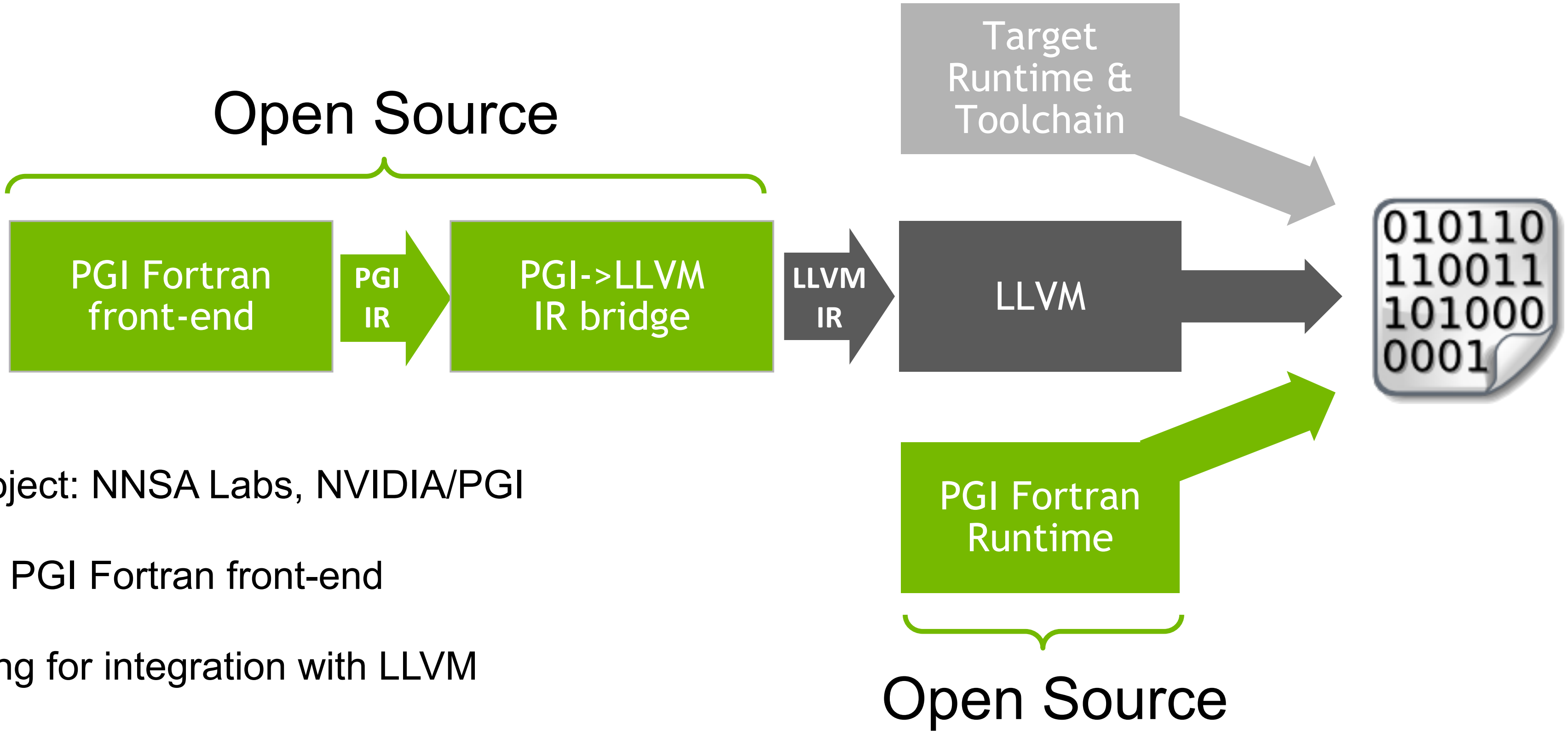
2019 LLVM Developers' Meeting

8 April, 2019

NVIDIA.

THE FLANG PROJECT

An open source Fortran front-end for LLVM



Open Source

PGI Fortran front-end → PGI IR → PGI->LLVM IR bridge → LLVM IR → LLVM

Target Runtime & Toolchain

PGI Fortran Runtime

Open Source

Multi-year project: NNSA Labs, NVIDIA/PGI

Based on the PGI Fortran front-end

Re-engineering for integration with LLVM

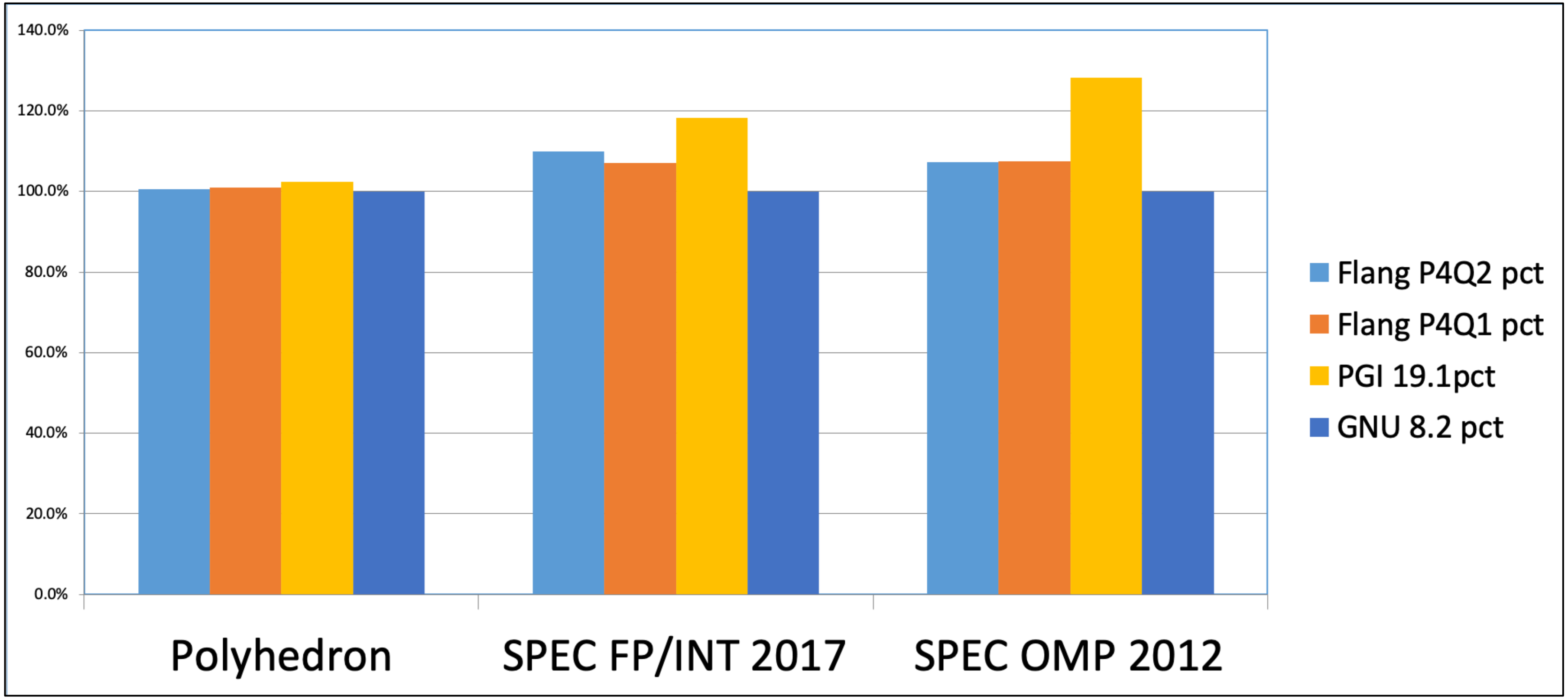
Develop CLANG-quality Fortran front end

STATE OF THE PROJECT

- Open source on github since May, 2017
- NVIDIA and others adding features and fixing bug; ~1400 commits
- Actively adding Fortran 2008 features this year
- Github, slack, mailing list in place; github issues are active
- Roadmap, submission policies, documentation published on github
- https://github.com/flang-compiler/flang

FLANG PERFORMANCE

All runs on dual-socket Intel Xeon Skylake



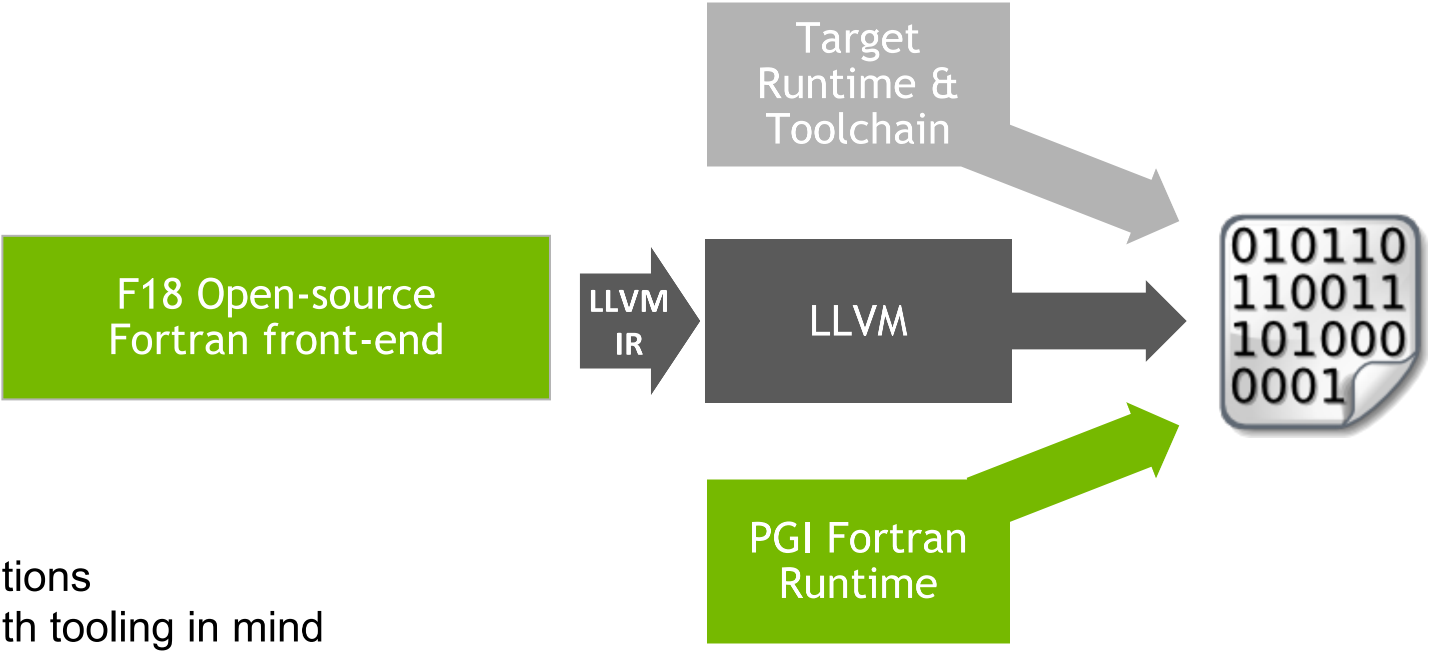
Performance measured March, 2019 and are considered ESTIMATES per SPEC run and reporting rules.
Two 20 core Skylake Intel® Gold 6148 CPU @ 2.40GHz CPUs @ 2.4GHz w/ 256GB memory. SPEC® is a registered trademark of the Standard Performance Evaluation Corporation (www.spec.org).

OPENMP TARGET OFFLOAD

- Tracking the OpenMP target offload work for Clang
- Flang target offload supports only the combined constructs of OpenMP
    - Generates code in SPMD/SIMT mode as this seems to be the most common pattern used in performance-oriented OpenMP programs targeting GPUs
    - The LLVM OpenMP runtime has a special case implementation for combined constructs that delivers much better GPU performance than the general-case code used to support non-combined constructs.
    - Johannes Doerfert is working on improving performance; his talk “Compiler Optimizations for (OpenMP) Target Offloading to GPUs” is later this afternoon.

THE F18 PROJECT

A NEW open source Fortran front-end for LLVM



- Modern C++
- High-quality source locations
- ASTs as C++ classes with tooling in mind
- ASTs follow the Fortran standard very closely
- Defer lowering until the AST is complete and checked
- Organize as libraries; expose support routines
- https://github.com/flang-compiler/f18