Flang Update

2018 European LLVM Developers' Meeting

17 April, 2018
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

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

- Source code released in May, 2017
- NVIDIA continually updates source to match PGI front-end
- First CLAs have been received; several submission queued
- Github, slack, mailing list in place; github issues are active
- Roadmap, submission policies, documentation published on github
- https://github.com/flang-compiler/flang
Compiler-callable math intrinsics library (*libpgmath*) designed to underlie the numerical intrinsics in Fortran, C and C++

- Designed to augment *libm* with fast and relaxed implementations
- Scalar and SIMD versions of elemental functions (sin, cos, ...)
- Current implementations for X86-64 and POWER CPUs
- Mostly written in C; possible to tune with assembly code when needed
- Open-sourced in March 2018 as part of Flang in github.com/flang-compiler/flang
FLANG SINGLE-CORE PERFORMANCE

All runs on Intel Xeon Skylake

SPEC FP RATE 2017
Application | Flang (Dev) secs | PGI 18.3 secs | GNU 7.3.0 secs
--- | --- | --- | ---
503.bwaves | 338.5 | 294.3 | 313.7
507.cactuBSSN | 257.8 | 255.9 | 257.8
521.wrf | 334.1 | 294.0 | 487.6
527.cam | 357.1 | 333.9 | 372.0
549.fotonik3d | 373.0 | 363.4 | 413.5
554.roms | 227.4 | 231.7 | 304.7

Geo Mean | 309.7 | 292.2 | 350.3

Flang Dev: -O2 -ffast-math -march=native
GNU: -O3 -ffast-math -fpeel-loops -funroll-loops
PGI 18.3: -fast -Bstatic_pgi -Mfprelaxed -Mhugetlb -Mnouniform -Mstack_arrays

Performance measured March, 2018 and are considered ESTIMATES per SPEC run and reporting rules.
Two 20 core Skylake Intel® Gold 6148 CPU @ 2.40GHz w/ 256GB memory. SPEC® is a registered trademark of the Standard Performance Evaluation Corporation (www.spec.org).
FLANG OPENMP 3.1 PERFORMANCE

All runs on dual-socket Intel Xeon Skylake

<table>
<thead>
<tr>
<th>SPEC OMP 2012 Application</th>
<th>Flang (Dev) secs</th>
<th>PGI 18.3 secs</th>
<th>GNU 7.3.0 secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>350.md</td>
<td>226.8</td>
<td>246.4</td>
<td>352.0</td>
</tr>
<tr>
<td>351.bwaves</td>
<td>261.4</td>
<td>249.1</td>
<td>253.8</td>
</tr>
<tr>
<td>357.bt331</td>
<td>287.3</td>
<td>267.0</td>
<td>306.2</td>
</tr>
<tr>
<td>360.libdc</td>
<td>308.6</td>
<td>328.6</td>
<td>321.3</td>
</tr>
<tr>
<td>362.fma3d</td>
<td>337.1</td>
<td>304.2</td>
<td>304.2</td>
</tr>
<tr>
<td>363.swim</td>
<td>324.0</td>
<td>326.4</td>
<td>322.7</td>
</tr>
<tr>
<td>370.mgrid331</td>
<td>354.4</td>
<td>352.4</td>
<td>350.7</td>
</tr>
<tr>
<td>371.applu331</td>
<td>244.1</td>
<td>244.9</td>
<td>250.2</td>
</tr>
<tr>
<td><strong>Geo Mean</strong></td>
<td><strong>289.7</strong></td>
<td><strong>287.1</strong></td>
<td><strong>305.4</strong></td>
</tr>
</tbody>
</table>

Flang Dev: -O2 -ffast-math -march=native -mp
GNU: -O3 -ffast-math -fpeel-loops -funroll-loops -fopen-mp
PGI 18.3: -fast -Bstatic_pgi -Mprelaxed -Mhugetlb -Mnoinform -Mstack_arrays -mp

Performance measured March, 2018 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).
FEEDBACK - SOURCE CODE, DATA STRUCTURES

- LALR Parser has no hooks for tools
- C program; monolithic without structured libraries
- Written in C, not C++; makes code sharing and reuse difficult
- Driver uses obscure -x flags instead of flags with meaningful names
- Error messages do not convey much context or detail compared to Clang
- AST lowered early; not suitable for source-to-source translation
- LLVM IR is generated using printf, not IRBuilder classes
RESPONDING TO FEEDBACK

- Refactor the pre-processor, scanner, and parser in C++
  - High-quality source locations
  - ATSs as C++ classes with tooling in mind
  - ASTs follow the Fortran standard very closely
- Restructure to improve support for tools
  - Defer lowering until the AST is complete and checked
  - Organize as libraries; expose support routines
- Convert flang2 to C++ to accommodate IRBuilder

https://github.com/flang-compiler/flang