TECHNISCHE UNIVERSITÄT DARMSTADT

SCIENTIFIC COMPUTING

THINGS ARE COMING TOGETHER FOR FORTRAN TOOLING

As experienced by Peter Arzt and Tim Heldmann

TECHNISCHE UNIVERSITÄT DARMSTADT

SC SCIENTIFIC COMPUTING

WHO ARE WE

- Peter Arzt
  - PhD Student

- Tim Heldmann
  - PhD Student

- Tooling for performance analysis
  - Automatic instrumentation selection for low-overhead measurements
- Applied performance engineering
  - Mostly in aerospace / space safety contexts

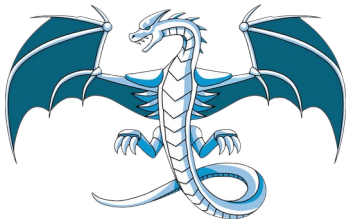
- C++ Compiler Tooling
  - Analysis
  - Transformation
  - Optimization
- Using both Clang-Tooling and IR-Passes

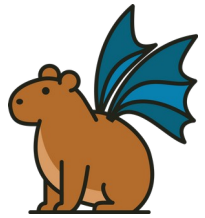
TECHNISCHE UNIVERSITÄT DARMSTADT

SC SCIENTIFIC COMPUTING

WHAT DO WE USUALLY DO

- Program Analysis: MetaCG / CaGe, ALPACA
- Program Transformation: MiniApex
- Program Optimization: Recursion Elimination
- Program Instrumentation: PIRA [1], CaPI [2], (FLIP)





  - These are all developed for C/C++, using Clang/LLVM

[1] Originally developed by JP Lehr

[2] Developed by Sebastian Kreutzer

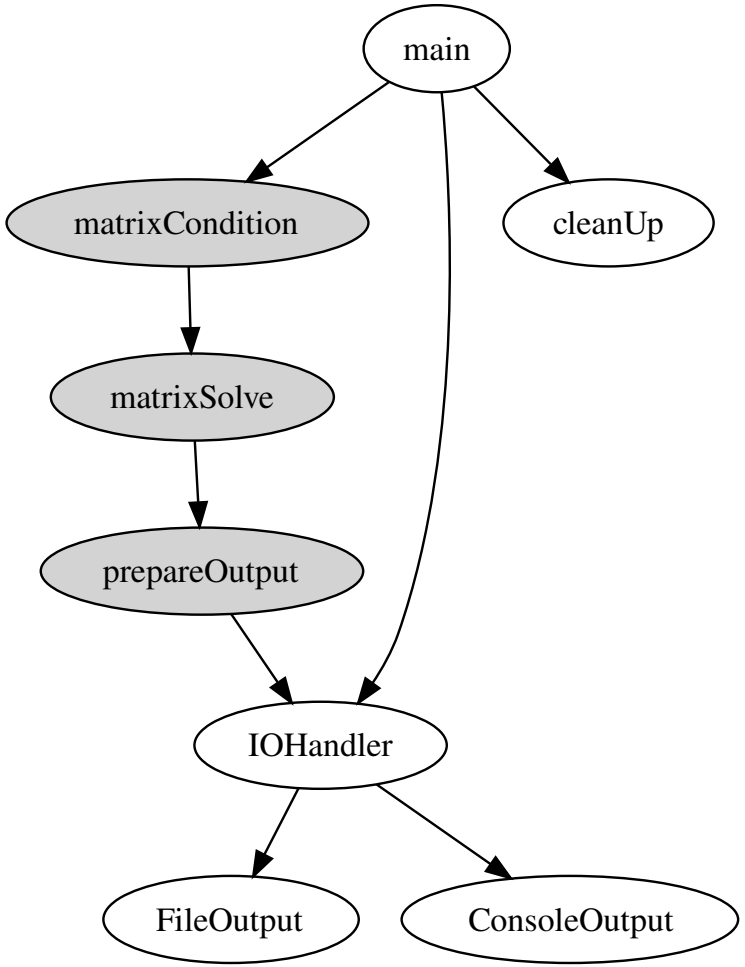
TECHNISCHE UNIVERSITÄT DARMSTADT

SCIENTIFIC COMPUTING

METACG/CAGE & PIRA

- Most of the tools use MetaCG in some capacity
  - It is a callgraph handling library
  - Can attach arbitrary metadata to a function node
  - CaGe is the IR based client tool

- PIRA: **P**erformance **I**nstrumentation **R**efinement **A**utomation
  - Tries to find a instrumentation selection that minimizes overhead and maximises information
  - Is intended to do hotspot/kernel detection, load-imbalance detection
  - Uses with Score-P for the actual performance messurements



TECHNISCHE UNIVERSITÄT DARMSTADT

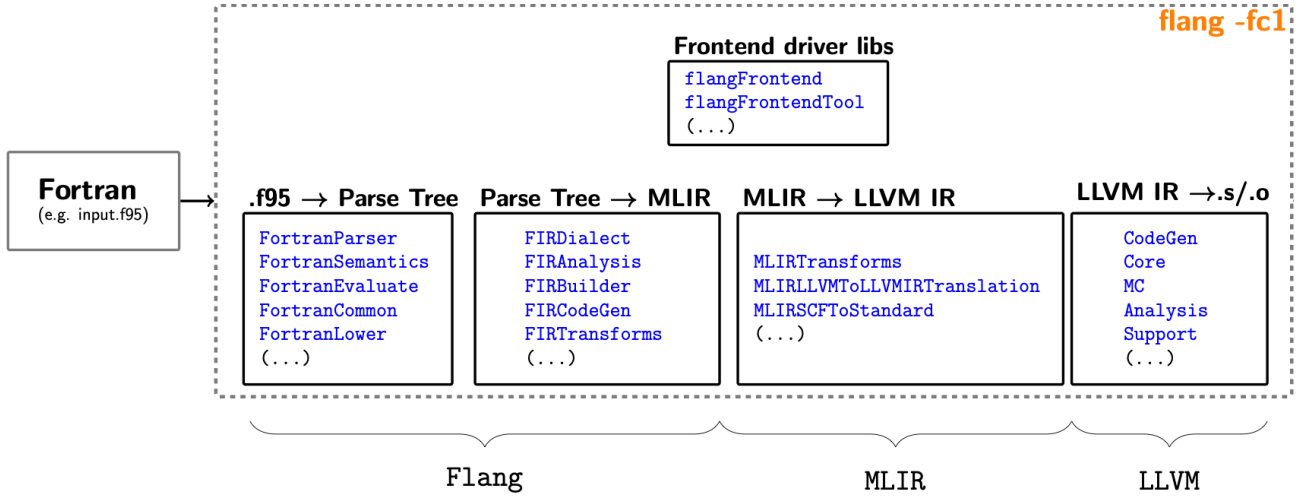
SCIENTIFIC COMPUTING

WELL THEN: FORTRAN?

- First things first, we don't know Fortran
  - We are not the only ones

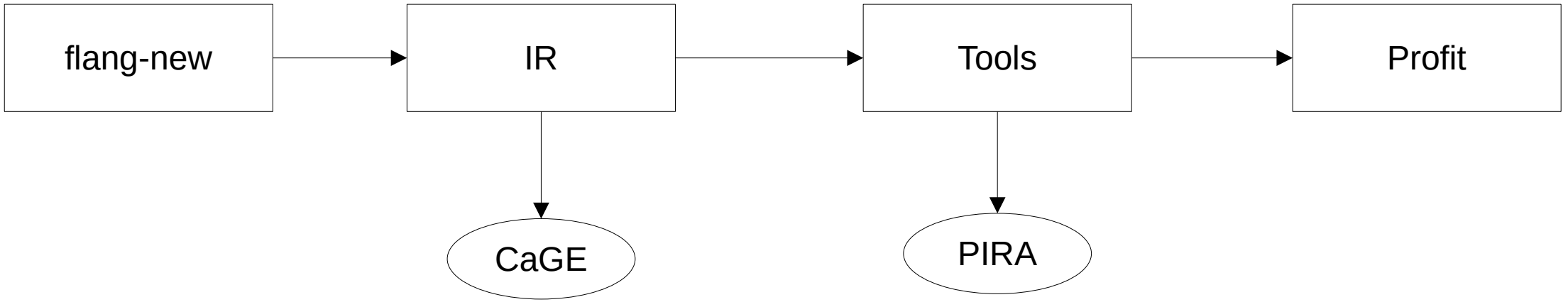
```
      ****************************************
      *             NRLMSIS-00               *
      *    NEUTRAL TERMOSPHERIC MODEL        *
      *                                      *
      ****************************************

COPIED FROM ORIGINAL SOURCE, TREATED AS BLACK BOX   ONLY
CHANGES INTRODUCED IN ORDER TO CORRECTLY INTERFACE WITH NAPEOS AND
MANTRA:
```

- Second things second, we never worked with Fortran Frontend Tooling or MLIR



HOW WE EXPECTED IT TO GO

- We do have tools that only rely on IR, which we can get from flang-new



FORTRAN → IR

- A colleague from Aachen provided us with jukkr-kloop as a test code
- Load available flang-new and GO!
  - Configure Failed: No Fortran compiler found
    - flang-news compiler identification might not be recognized by the buildsystem
      - Hack compiler identification into the build system
  - Compile failed: Unsupported Intrinsic
    - This was in January 2024, with LLVM 16
    - LLVM-Trunk had a fix, that was merged in December 2023 → compile LLVM from source
- We got LLVM IR!

```
[ 12%] Building Fortran object CMakeFiles/test_kloop.x.dir/source/KKRhost/invsupercell.f90.o
[ 20%] Building Fortran object CMakeFiles/test_kloop.x.dir/source/KKRhost/surfgf.f90.o
[ 20%] Building Fortran object CMakeFiles/test_kloop.x.dir/source/KKRhost/invslab.f90.o
error: loc(" .../source/external/NPY-for-Fortran/src/npy.F90":49:9): .../clang/16.0.6/
llvm-project/flang/lib/Lower/IntrinsicCall.cpp:1643: not yet implemented: intrinsic:
execute_command_line
```

IR → TOOLS

- With LLVM 18, we now have opaque pointer
  - CaGe relied on typed pointers for some analysis (VTables mostly)
    - We don’t need VTables for Fortan, just strip the whole logic
  - CaGe now can provide Callgraphs for Fortran Codes

- Now use PIRA and Score-P to iteratively instrument the binary

SCORE-P

- Profiling and tracing toolkit for parallel codes
- Multiple options for instrumentation
  - `-finstrument-functions` and friends: Less fine grain control
  - GCC plugin shipped with Score-P

- Since Score-P 9.0 (currently a release candidate): LLVM instrumentation plugin
  - Works with flang-new!

- Load OpenMPI 4.1.4, run jukkr-kloop and profile
  - mpif90 compiler wrapper is incompatible with flang-new

OPENMPI 5.0.5

- OpenMPI reads available compilerflags at compiletime
  - You need to have all compilers available that you want to have supported
  - OpenMPI 4.1.4 can not be compiled with flang-new
    - But OpenMPI 5.0.5 can
      - Compile your own OpenMPI, with flang-new

- Recompile Score-P because it is tied to a specific MPI version

- Run PIRA!

INSTRUMENTATION

- Performance measurement technique based on inserting **measurement hooks**
  - Here: Compiler instrumentation, function level
- Can produce precise, reliable measurements
- Introduces overhead
  - Potentially slow-down > 1000x

➢ Need to find **trade-off between coverage and overhead**

```
1  void __cyg_profile_func_enter(void *current_func, void *callsite);
2  void __cyg_profile_func_exit(void *current_func, void *callsite);
3
4  int square(int num) {
5      __cyg_profile_func_enter(square, 0);
6      int res = num*num;
7      __cyg_profile_func_exit(square, 0);
8      return res;
9  }
10
11 int main() {
12     __cyg_profile_func_enter(main, 0);
13     int res = square(42);
14     __cyg_profile_func_exit(main, 0);
15     return res;
16 }
```

Compiler instrumentation, e.g., using Clang/GCC's `-finstrument-functions`

No instrumentation
→ No overhead
→ No coverage

Instrumentation selection

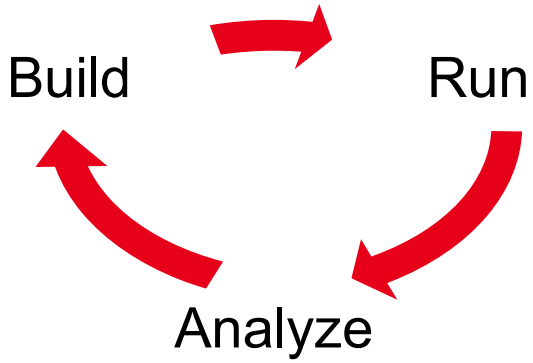
Full instrumentation
→ High overhead
→ Perfect coverage

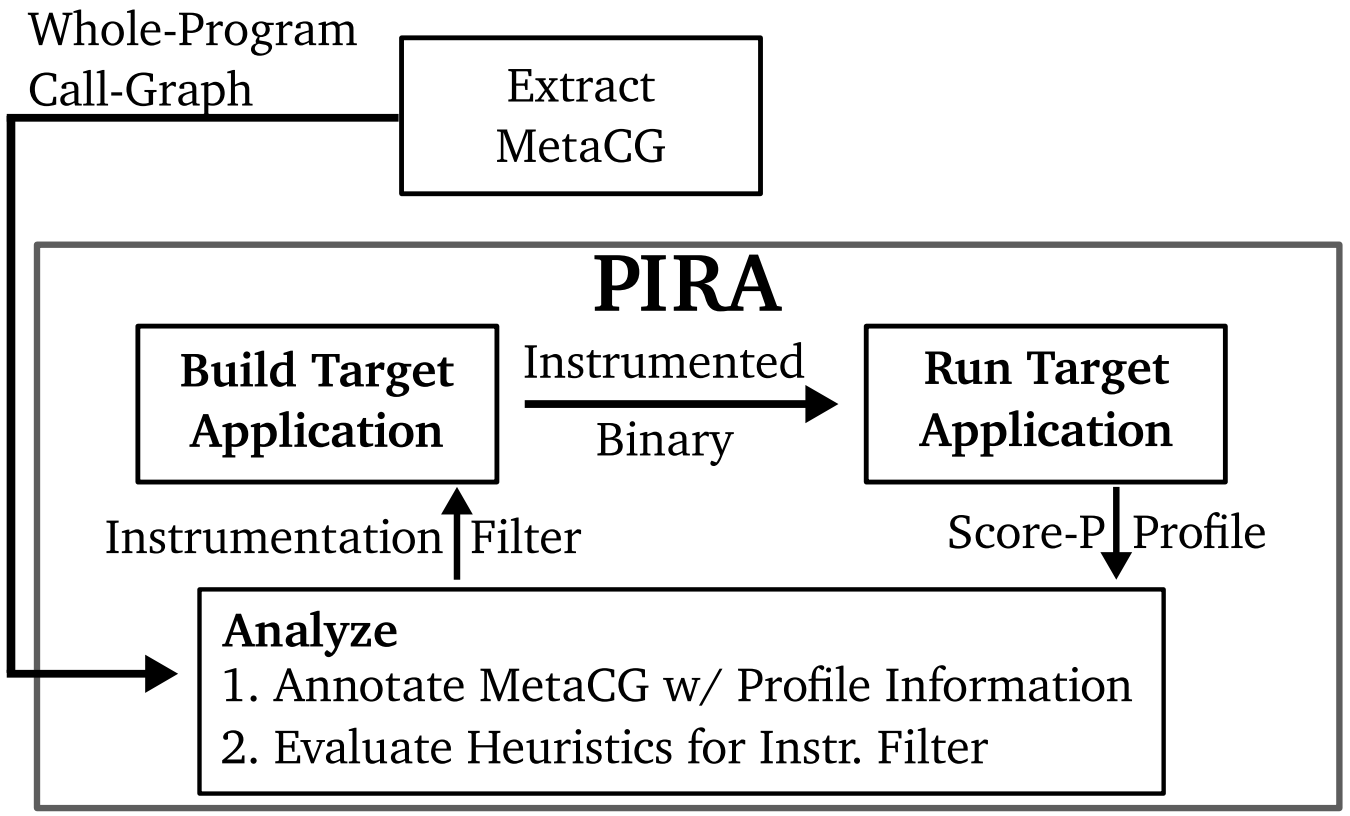
TECHNISCHE UNIVERSITÄT DARMSTADT

SCIENTIFIC COMPUTING

ITERATIVE REFINEMENT AUTOMATION

- Find trade-off between measurement coverage and overhead by **iteratively** improving instrumentation





TECHNISCHE UNIVERSITÄT DARMSTADT

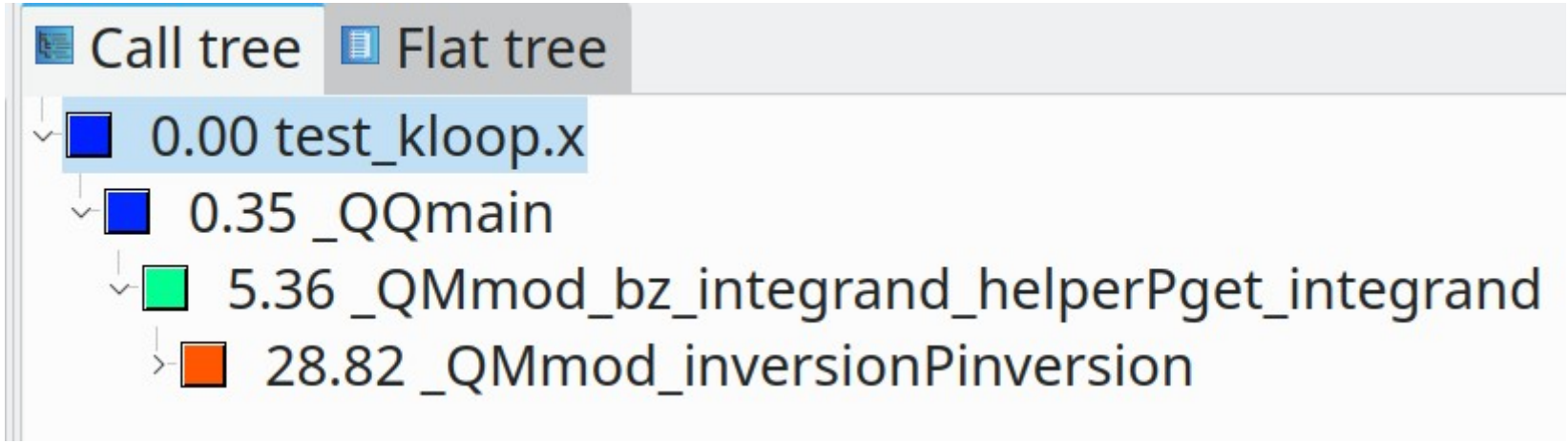
SCIENTIFIC COMPUTING

JUKKR-KLOOP

- Is intended to be compiled with intels fortran compiler and uses intel mkl
  - hack mkl linking into the buildscript for flang-new, and:

IT WORKS!!!

- PIRA succesfully instruments and profiles the binary
- Correctly identifies hotspot



CONCLUSION

- You can use your LLVM-IR based tools with flang-new right now
  - you might need to move to a newer version
  - you might need to slightly modify some build scripts
  - you might need to compile some things from source
- But you can do it

- We demonstrated this using jukkr-kloop

- Things are really coming together!

OUTLOOK

- We rely solely on LLVM-IR when it comes to Fortran Tooling:
- Also do Sourcecode based analysis:
  - Flang-new has a frontend-plugin interface
    - similar to clang-compiler plugins
  - currently under active development

- Improve heuristics PIRA heuristics for and profiling overhead
  - Use FLIP for exact call counts and runtime estimation
- Maybe move to CaPI and XRAY to eliminate rebuilding step