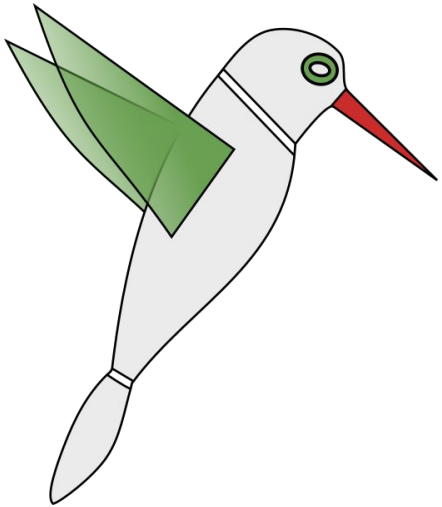


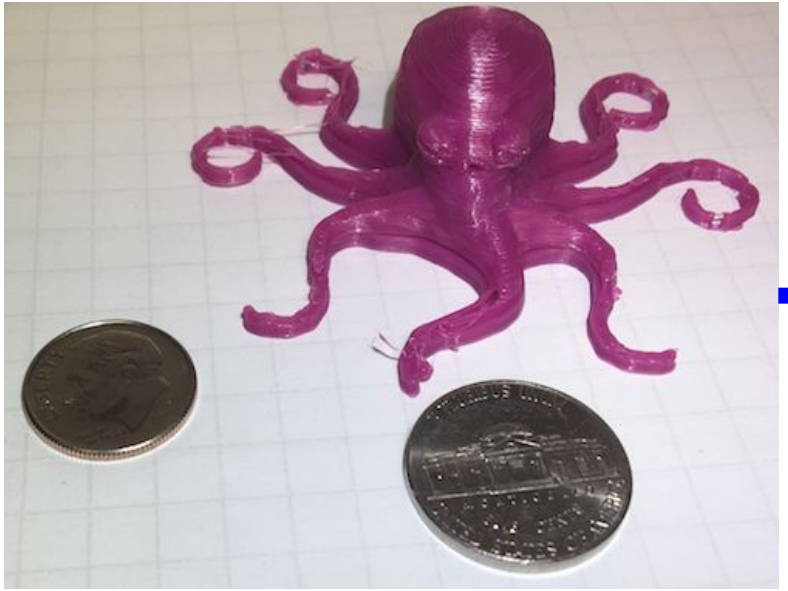
FLiT

Floating-Point Reproducibility Testing Framework

Michael Bentley, Ian Briggs, Ganesh Gopalakrishnan
University of Utah

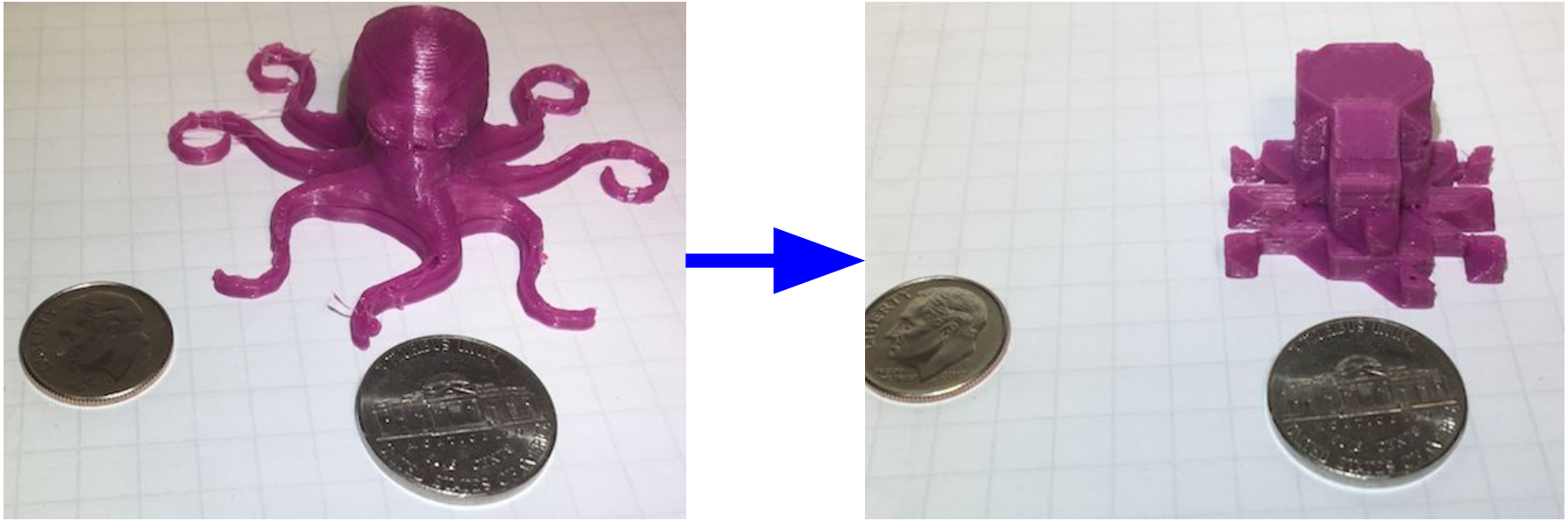
Ignacio Laguna, Greg Lee, Dong H. Ahn
Lawrence Livermore National Laboratory

Floating-Point is non-intuitive



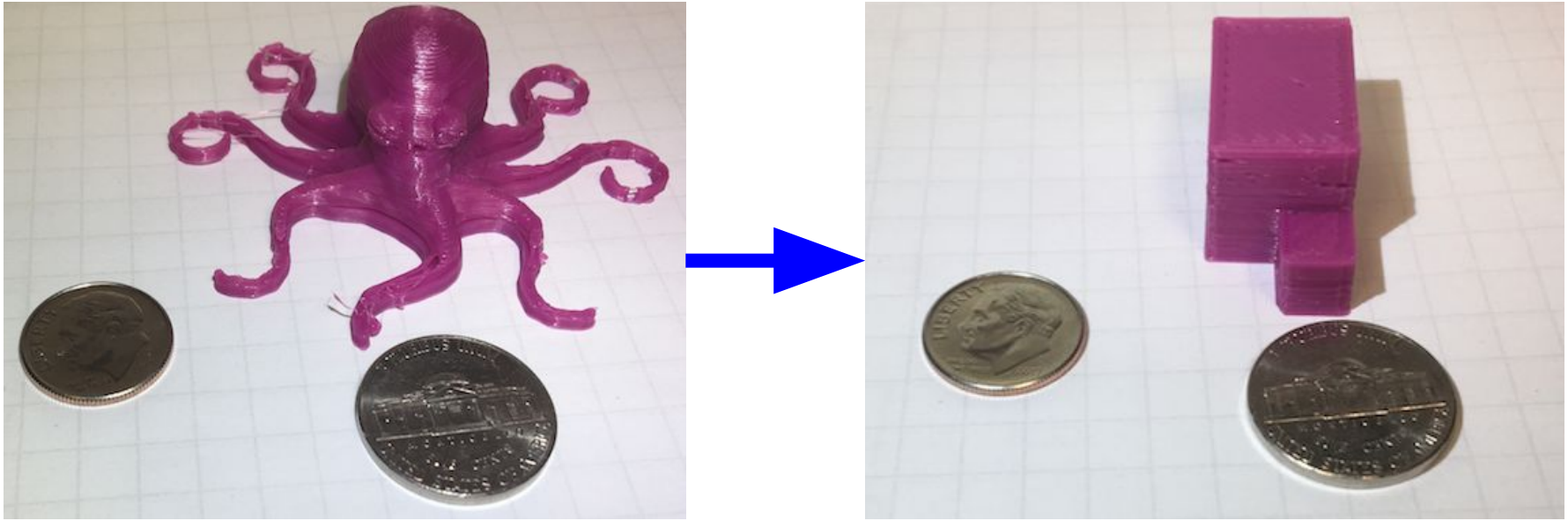
The Traveling Octopus Paradox: `(a + b) - b ?= a` for very large `b`

Floating-Point is non-intuitive



The Traveling Octopus Paradox: `(a + b) - b != a` for very large `b`

Floating-Point is non-intuitive



The Traveling Octopus Paradox: `(a + b) - b != a` for even larger `b`

Optimizations may be climate-changing

From “A new ensemble-based consistency test for the Community Earth System Model” By A. H. Baker, et. al.

> We expect that the following tests on Yellowstone will not be climate-changing, and thus, will be consistent with our initial ensembles distribution:

- NO-OPT: changing the Intel compiler to remove optimization (-O0)
- INTEL-15: changing the Intel compiler version to 15.0.0
- NO-THRD: compiling CAM without threading (MPI-only)
- PGI: using the CESM-supported PGI compiler (13.0)
- GNU: using the CESM-supported GNU compiler (4.8.0)

Case study: Uintah

A library for portable simulations on heterogeneous systems

- Ran a simulation on a new system
- Deadlock occurred

What?

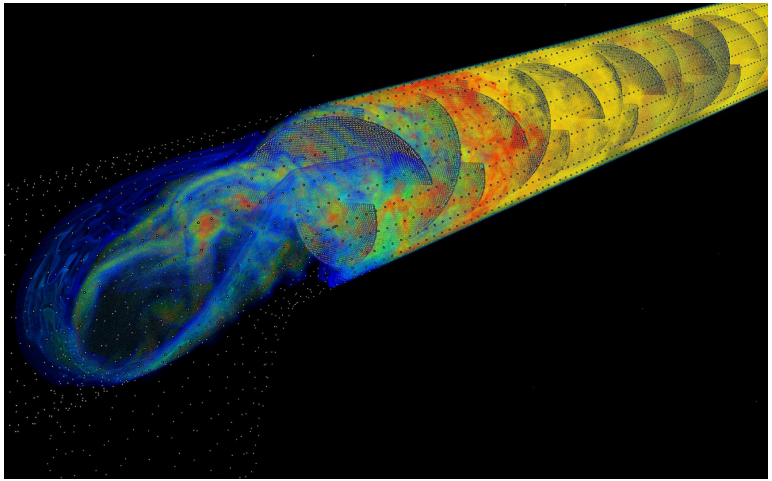
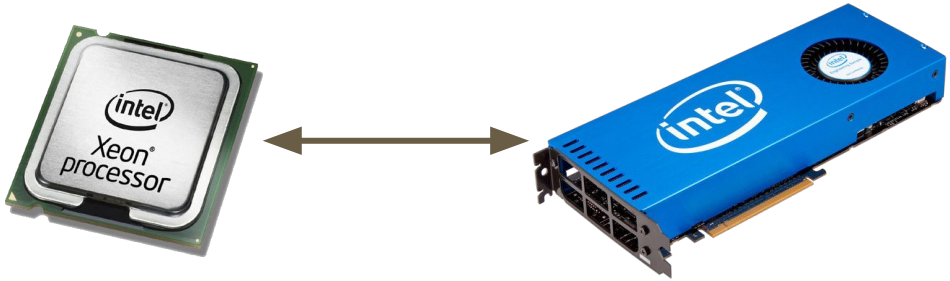


Image courtesy SCI Institute (http://uintah.utah.edu)

Case study: Uintah



Xeon Phi had a different division algorithm.

MPI Sends != # MPI Receives

Deadlock!

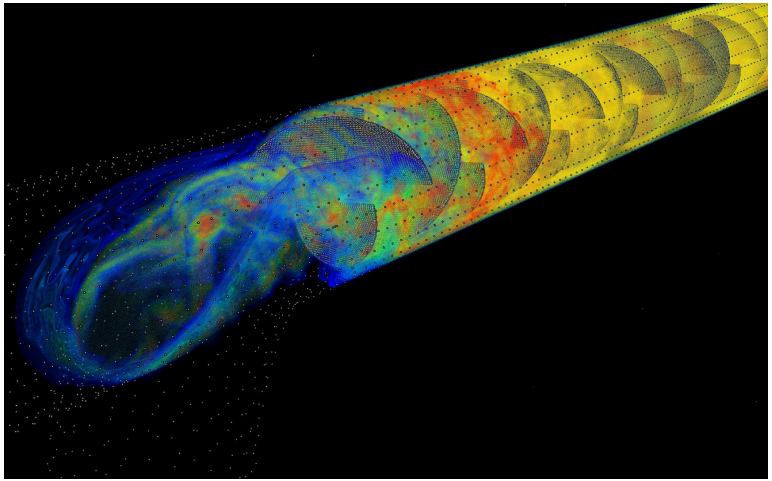
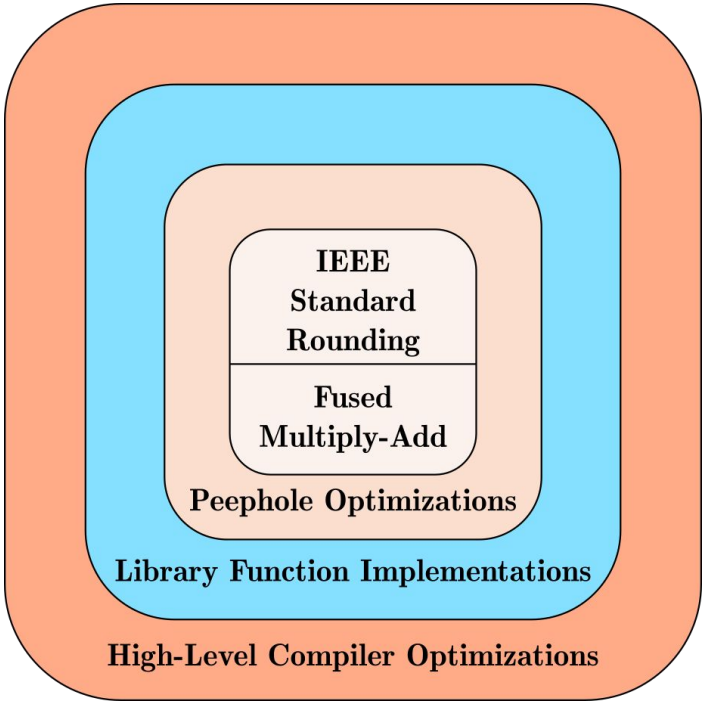


Image courtesy SCI Institute (http://uintah.utah.edu)

What causes variability?



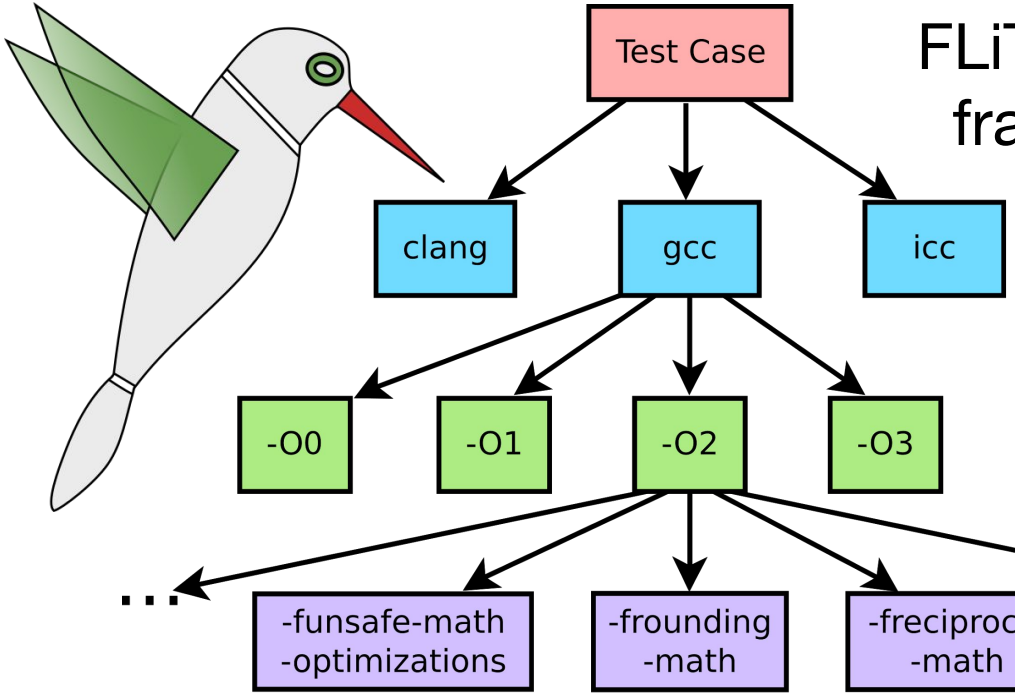
Compiler and hardware variability:

1. Associativity violations
   - Vectorization
2. Software implementations
3. Hardware (e.g. FMA)
4. Compile time vs runtime
5. Higher precision intermediates

FLiT

https://pruners.github.io/flit

Our Contribution: FLiT



FLiT is a reproducibility test framework in the PRUNERS toolset (pruners.gihub.io).

Hundreds of compilations are compared against a baseline compilation.

Profiles test cases too!

What is a FLiT Test?

Must implement these

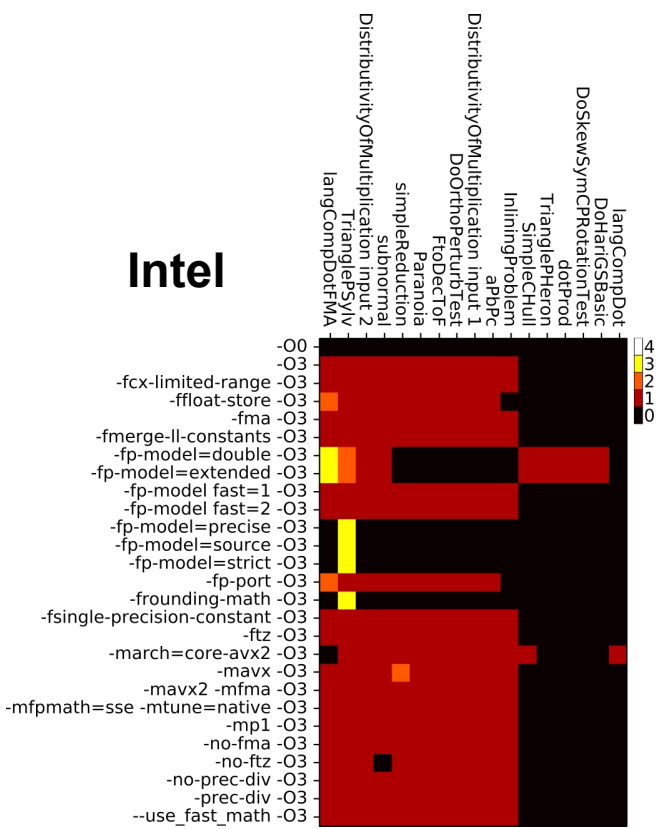
- `size_t getInputsPerRun()`
- `vector<T> getDefaultInput()`
  - Can give more inputs
- `flit::Variant run_impl(vector<T> input)`
  - Must be **deterministic**

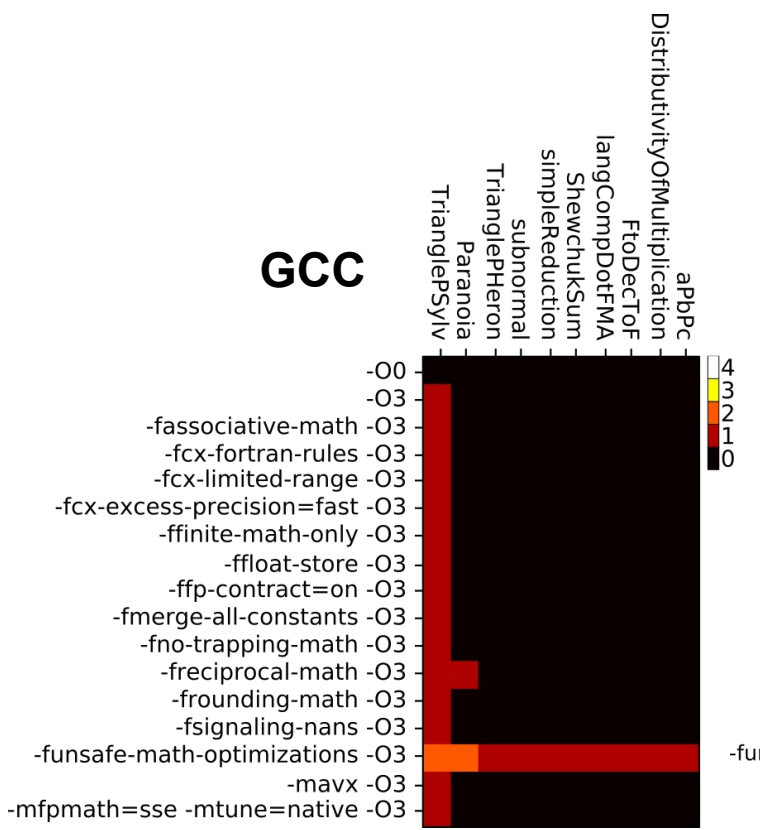
Optionally

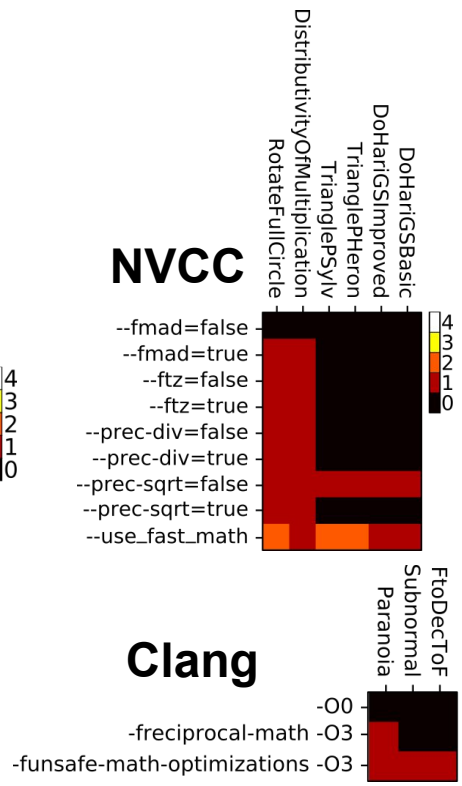
- `long double compare(gt-result, other-result)`

What else?

Heat Maps: So many answers

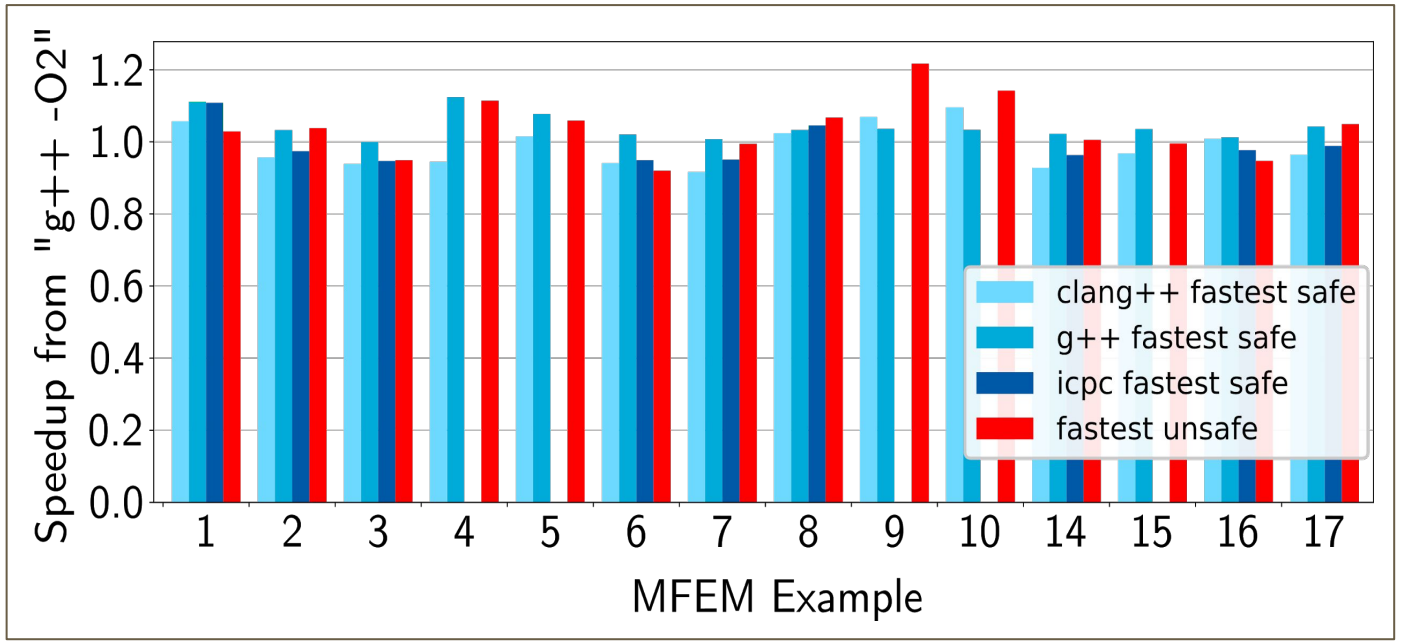




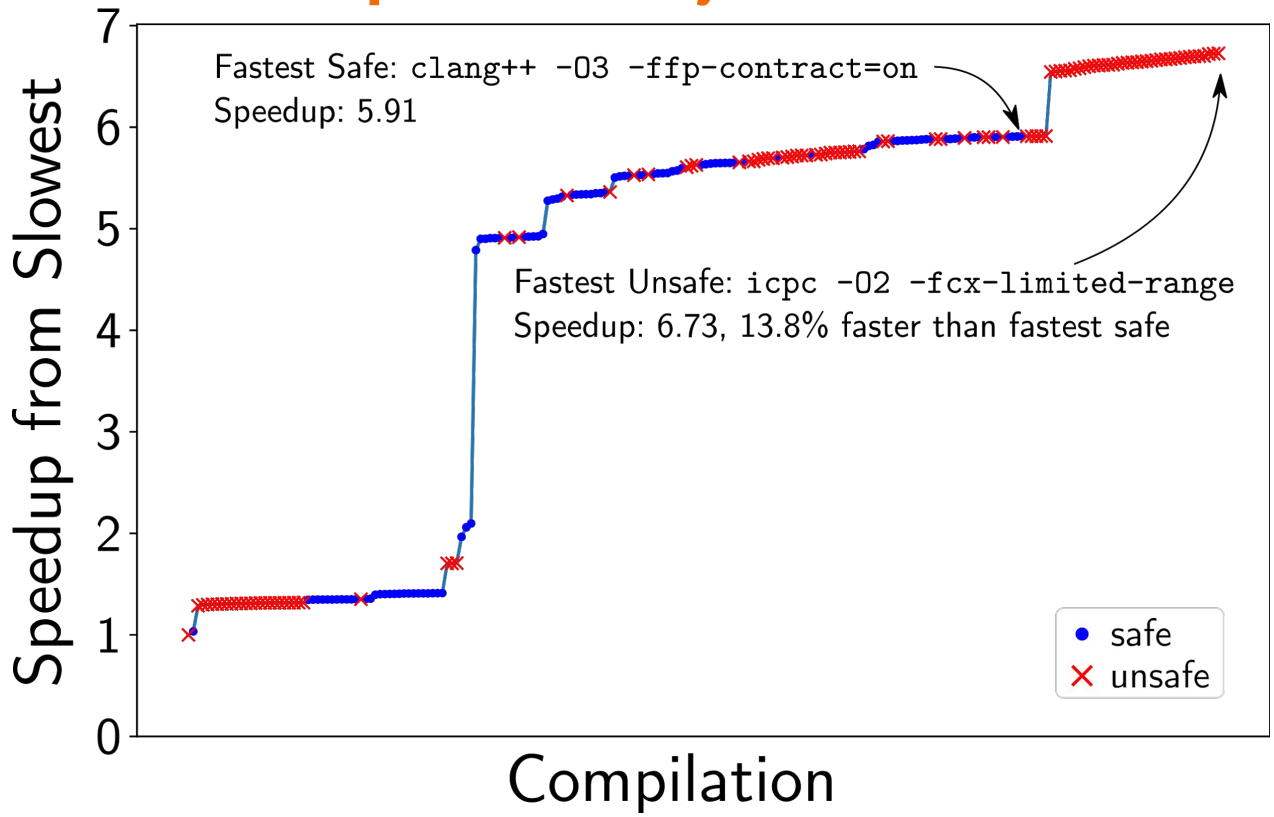


MFEM: finite element library

Fastest safe compilation almost always wins



Tradeoff Between Reproducibility and Performance



Debugging the Variability

Suppose knowing which compilations to avoid is not good enough for you. You want to identify where and why this is happening?

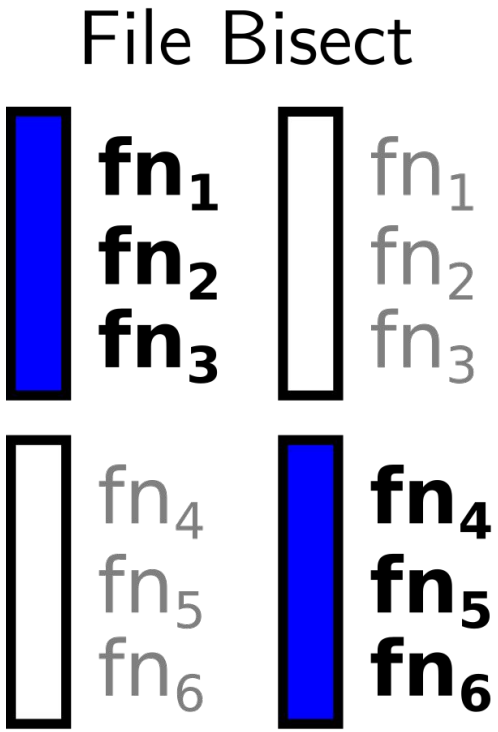
FLiT Bisect

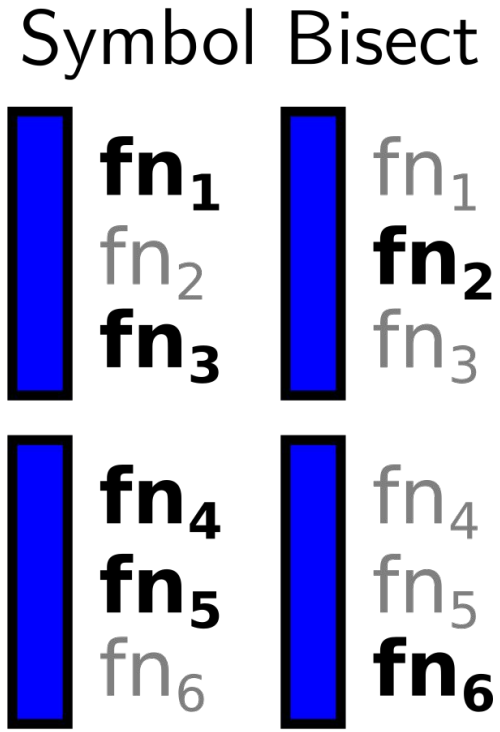
FLiT Bisect

Only works about 75% of the time!

Compile once, Link often

Assumes files cause variability independently





Recompiles with -fPIC

Uses objcopy to make symbols weak

Assumes functions cause variability independently

FLiT Workflow