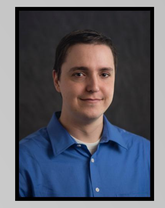
Quantifying Compiler Effects on Code Performance and Reproducibility using FLIT

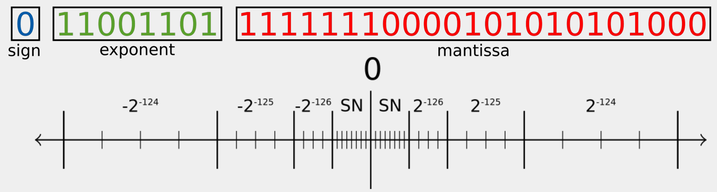
Michael Bentley[1], Ganesh Gopalakrishnan[1] (Advisor), Dong H. Ahn[2] (Advisor)

[1]University of Utah - Computer Science, [2]Lawrence Livermore National Laboratory - Livermore Computing



Floating-Point

6.022×10^{23}

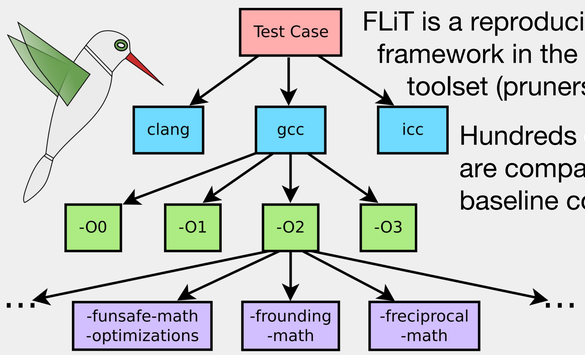


Compiler optimizations sometimes do "unsafe" things giving *different answers*.

Example: `a*b + a*c != a*(b + c)`

5% of the time the above is not equal (for 32-bit floats)

FLiT: Part of the PRUNERS Toolset



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

Hundreds of compilations are compared against a baseline compilation.

FLiT helps find good performance with reproducibility.

Question: Is FLiT's approach useful to real-world libraries and applications using their own existing test cases?

Litmus Tests:

FLiT comes with *Litmus Tests* that demonstrate problems with floating-point optimizations.

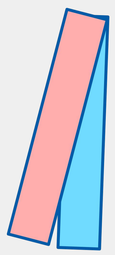
Example: keeping track of error from addition

```
float val = x + y;
float err = y - (val - x);
```

optimization ↓

```
float val = x + y;
float err = y - y;
```

err will *always* be zero here.



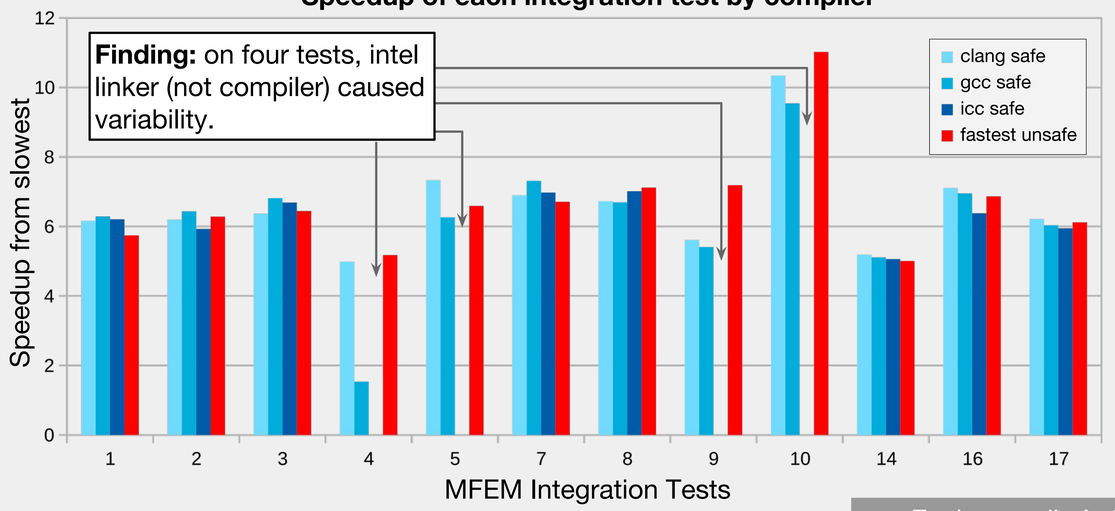
Case Study: MFEM

MFEM was chosen as a **real-world library** to show FLiT's utility

- MFEM (mfem.org) is a finite element library used in a variety of large-scale simulations at LLNL.
- FLiT was applied to 12 of the 17 MFEM integration tests on the Cab cluster - x86_64 Intel Xeon CPU E5-2670
- Baseline is set as "**gcc -O0**".

Compiler	Version	Compilations	Test Runs	Different Answers
clang	4.0.0	67	804	18 (2.2%)
gcc	4.9.2	66	792	82 (10.4%)
icc	16.0.1	101	1212	740 (61.1%)

Speedup of each integration test by compiler

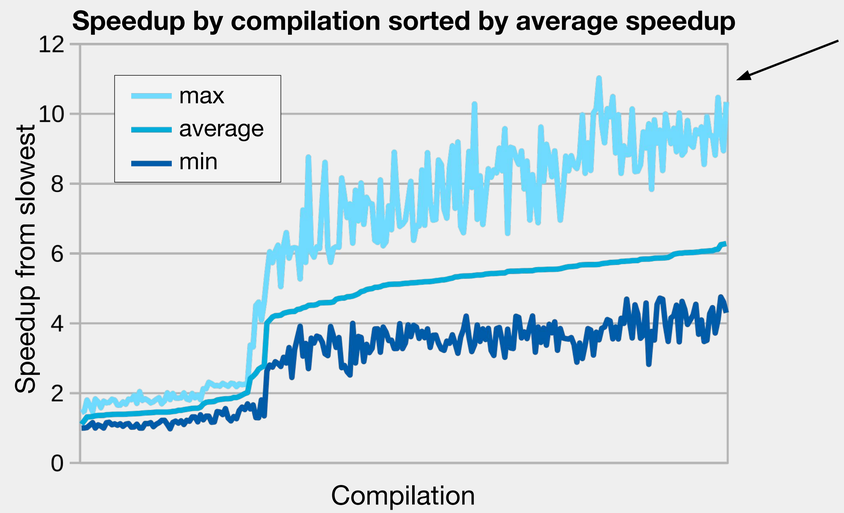


Finding: when using gcc 5.4.0 to compile FLiT, clang compilations would not run with optimization level -O2 or -O3. This is a clang bug and has been reported to their team.

The red bar is the fastest runtime among all compilations (including all compilers) with different answers than the baseline.

Finding: 8 of 12 tests had better performance with safe compilations. Of the other four, only one of them had a performance boost higher than 7% (integration test 9 had a 28% boost with an unsafe compilation).

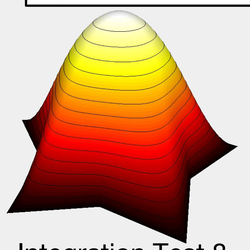
Speedup by compilation sorted by average speedup



The max, average, and min are over the speedup of the 12 integration tests. Compilations are sorted by average speedup.

Each compiler's flags for fastest average speedup			
Compiler	Opt. Level	Flag	Avg speedup
clang	-O2	-fexcess-precision=standard	6.29
gcc	-O2	-mfpmath=sse -mtune=native	6.27
icc	-O3	-fp-model fast=2	6.08

Finding: Both clang's and gcc's fastest flags were reproducible in all 12 integration tests ported to FLiT.



Integration Test 8 from MFEM

Finding: integration test 8 had an ℓ_2 norm difference of 10^{-6} even though it is an iterative algorithm stopping at 10^{-12}. *Caused by vectorization of dot product.*

- Takes 0.8 secs on a single core
- Magnified for large tasks, error may become huge

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-POST-735966