Improving Reliability Through Analyzing and Debugging Floating-Point Software

Ignacio Laguna
Computer Scientist
Center for Applied Scientific Computing

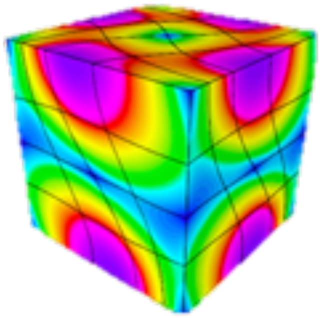
2020 ECP Annual Meeting, Feb 4, 2020

LLNL-PRES-802189
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

A Hard-To-Debug Case

Hydrodynamics mini application



Early development and porting to new system (IBM Power8, NVIDIA GPUs)

clang	–O1:	|e| = 129941.1064990107
clang	–O2:	|e| = 129941.1064990107
clang	–O3:	|e| = 129941.1064990107
gcc	–O1:	|e| = 129941.1064990107
gcc	–O2:	|e| = 129941.1064990107
gcc	–O3:	|e| = 129941.1064990107
xlc	–O1:	|e| = 129941.1064990107
xlc	–O2:	|e| = 129941.1064990107
xlc	–O3:	**|e| = 144174.9336610391**

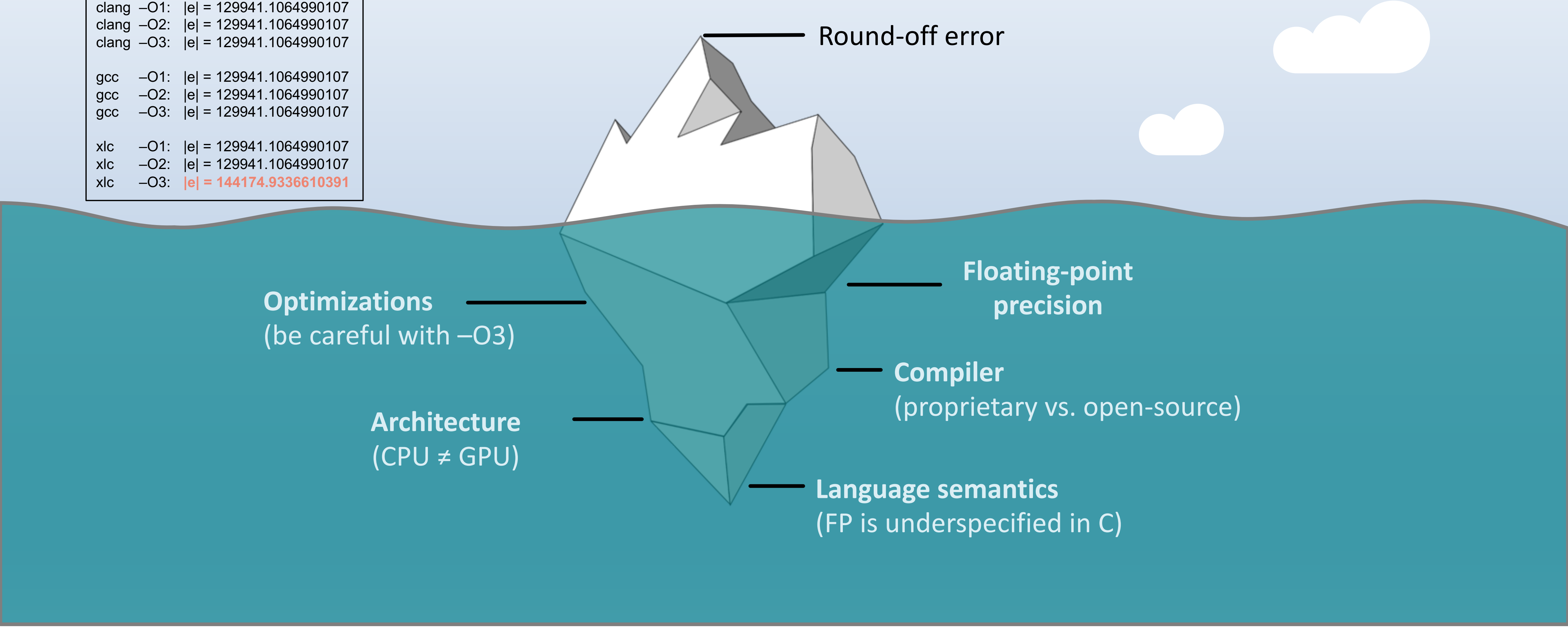
It took several weeks of effort to debug it

Many Factors are Involved in Unexpected Numerical Results

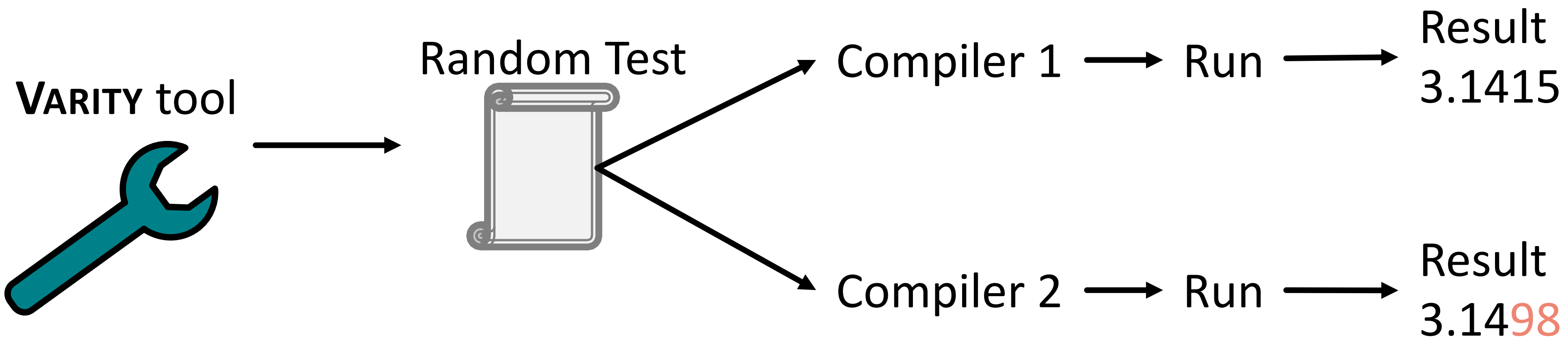
```
clang  –O1:   |e| = 129941.1064990107
clang  –O2:   |e| = 129941.1064990107
clang  –O3:   |e| = 129941.1064990107

gcc    –O1:   |e| = 129941.1064990107
gcc    –O2:   |e| = 129941.1064990107
gcc    –O3:   |e| = 129941.1064990107

xlc    –O1:   |e| = 129941.1064990107
xlc    –O2:   |e| = 129941.1064990107
xlc    –O3:   |e| = 144174.9336610391
```



What Floating-Point Code Can Produce Variability?



https://github.com/LLNL/Varity

Variability Examples Found by Varity

Example 1: variability between host and device

```
void compute(double comp,int var_1,double var_2,
double var_3,double var_4,double var_5,double var_6,
double var_7,double var_8,double var_9,double var_10,
double var_11,double var_12,double var_13,
double var_14) {
  double tmp_1 = +1.7948E-306;
  comp = tmp_1 + +1.2280E305 - var_2 +
    ceil((+1.0525E-307 - var_3 / var_4 / var_5));
  for (int i=0; i < var_1; ++i) {
    comp += (var_6 * (var_7 - var_8 - var_9));
  }
  if (comp > var_10 * var_11) {
    comp = (-1.7924E-320 - (+0.0 / (var_12/var_13)));
    comp += (var_14 * (+0.0 - -1.4541E-306));
  }
  printf("%.17g\n", comp);
}
```

Input

```
0.0 5 -0.0 -1.3121E-306 +1.9332E-313 +1.0351E-306
+1.1275E172 -1.7335E113 +1.2916E306 +1.9142E-319
+1.1877E-306 +1.2973E-101 +1.0607E-181 -1.9621E-306
-1.5913E118-03
```

clang -O3

```
$ ./test-clang
NaN
```

nvcc -O3 (V100 GPU)

```
$ ./test-nvcc
-2.3139093300000002e-188
```

Example 2: variability even with –O0

```
void compute(double tmp_1, double tmp_2, double tmp_3,
  double tmp_4, double tmp_5, double tmp_6) {
  if (tmp_1 > (-1.9275E54 * tmp_2 + (tmp_3 - tmp_4 * tmp_5)))
  {
      tmp_1 = (0 * tmp_6);
  }
  printf("%.17g\n", tmp_1);

  return 0;
}
```

Input

```
+1.3438E306 -1.8226E305 +1.4310E306 -1.8556E305 -
1.2631E305 -1.0353E3
```

clang -O0

```
$ ./test-clang
1.3437999999999999e+306
```

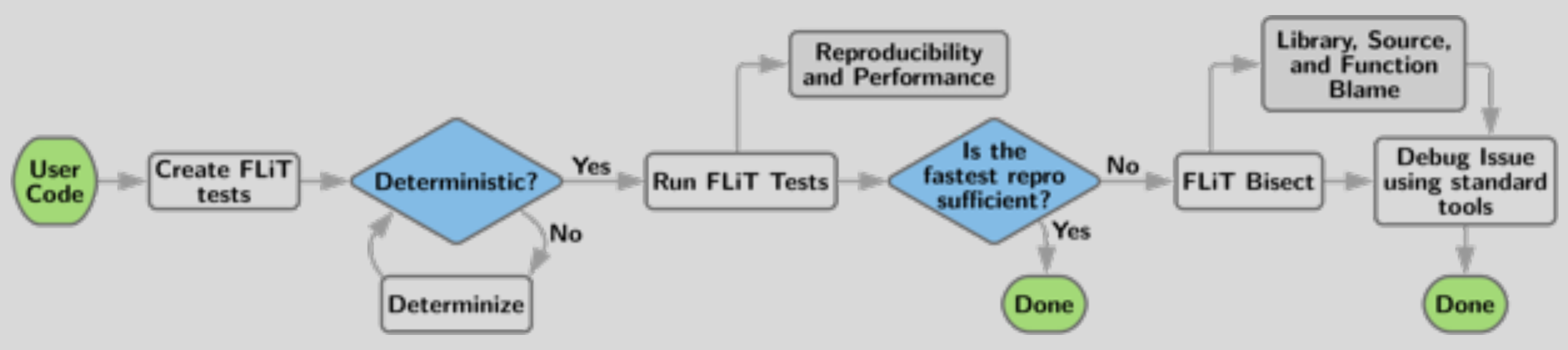
gcc -O0

```
$ ./test-gcc
1.3437999999999999e+306
```

xlc -O0

```
$ ./test-xlc
-0
```

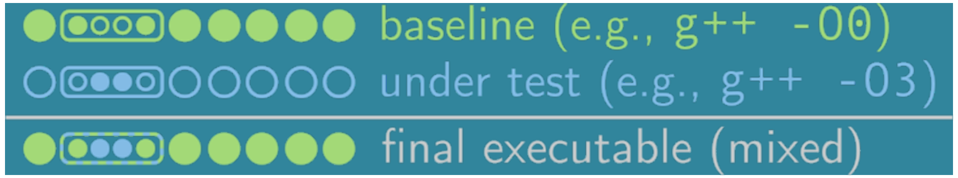
FLiT: Floating-Point Litmus Tester



Multiple Levels:

- Determine variability-inducing compilations
- Analyze the tradeoff of reproducibility and performance
- Locate variability by identifying files and functions causing variability

Bisection Method



Michael Bentley, Ian Briggs, Ganesh Gopalakrishnan, Dong H. Ahn, Ignacio Laguna, Gregory L. Lee, and Holger E. Jones. **Multi-Level Analysis of Compiler-Induced Variability and Performance Tradeoffs**. In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '19).

Detecting the Result of Exceptions in a CUDA Program

- Place **printf** statements in the code (as many as possible)

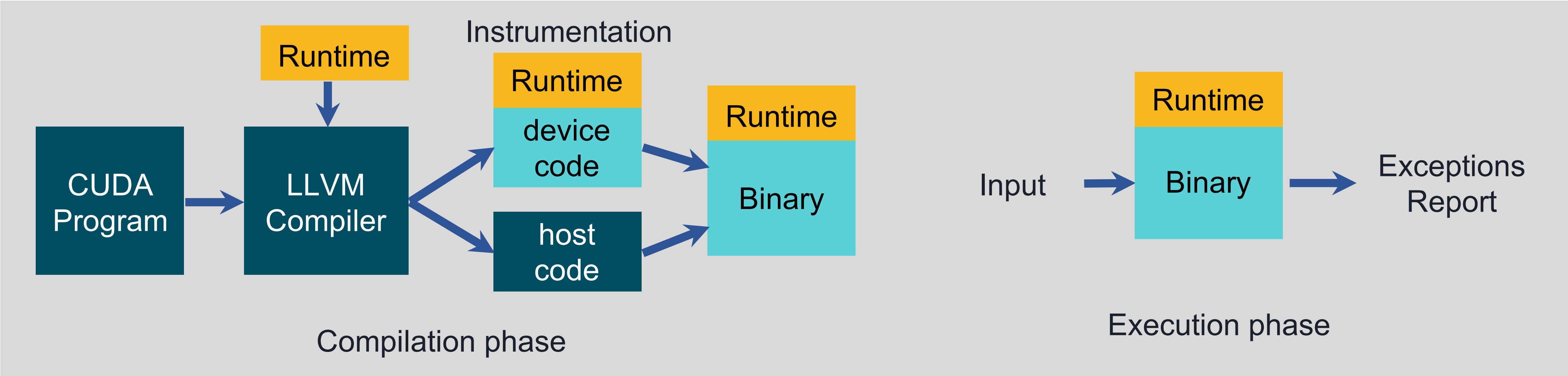
```
double x = 0;
x = x/x;
printf("res = %e\n", x);
```

- Programing checks are available in CUDA:

```
__device__   int isnan ( float  a );
__device__   int isnan ( double  a );
```

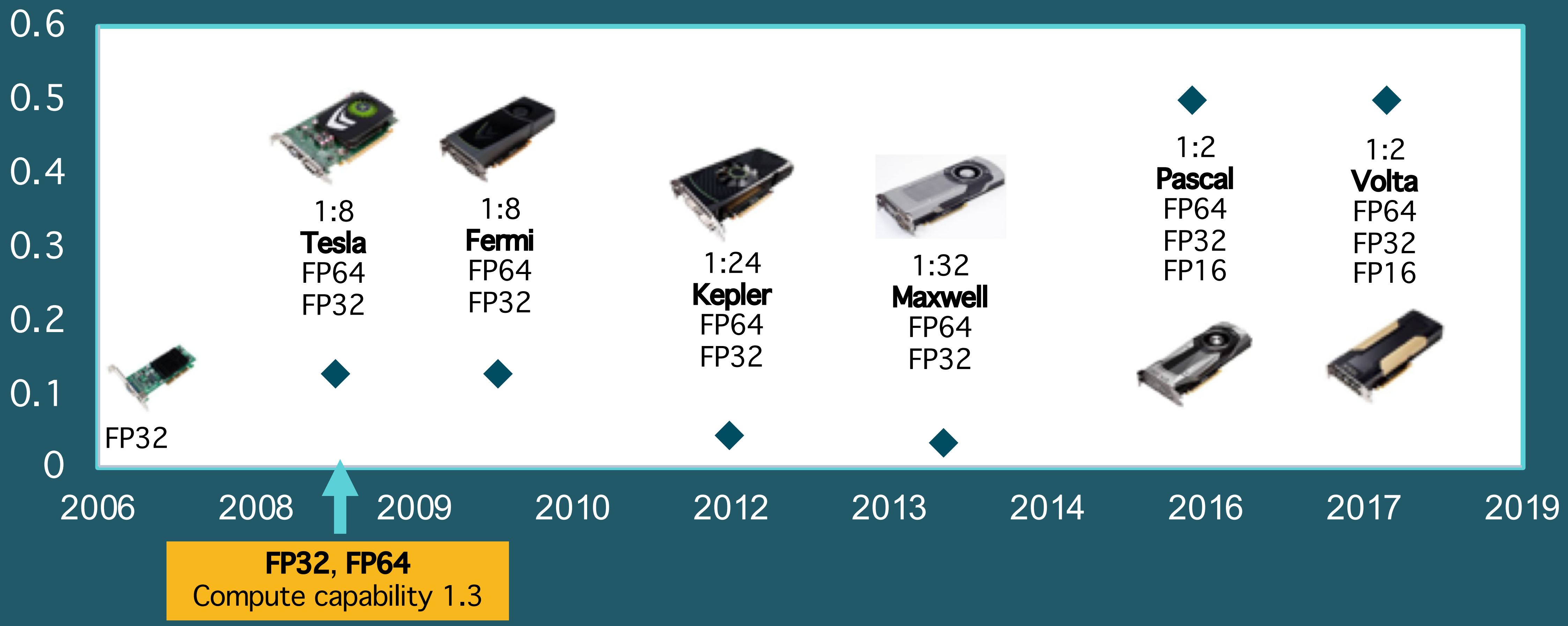
These solutions are not ideal;   they require significant programming effort

FPChecker: Automatic Detection of Floating-Point Exceptions in GPUs

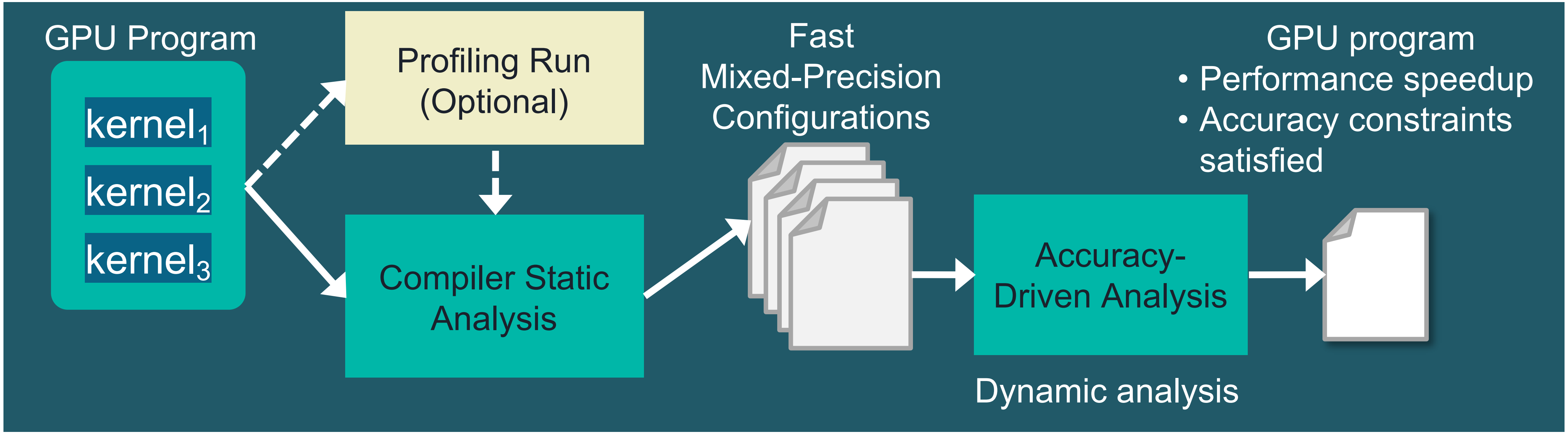


https://github.com/LLNL/FPChecker

Floating-Point Precision Levels in GPUs Are Increasing



GPUMixer: Performance-Driven Floating-Point Tuning for GPU Scientific Applications



Ignacio Laguna, Paul C. Wood, Ranvijay Singh, Saurabh Bagchi. **GPUMixer: Performance-Driven Floating-Point Tuning for GPU Scientific Applications**. ISC High Performance, Frankfurt, Germany, Jun 16-20, 2019 **(Best paper)**

Tutorial on Floating-Point Analysis Tools

http://fpanalysistools.org/

- Demonstrate several analysis tools
- Hands-on exercises
- Cover various important aspects of floating-point and repro
- Tutorials:
  - LANL, Jan 9th, 2020
  - SC19, Denver, Nov 17th, 2019
  - PEARC19, Chicago, Jul 30th, 2019

Lawrence Livermore National Laboratory