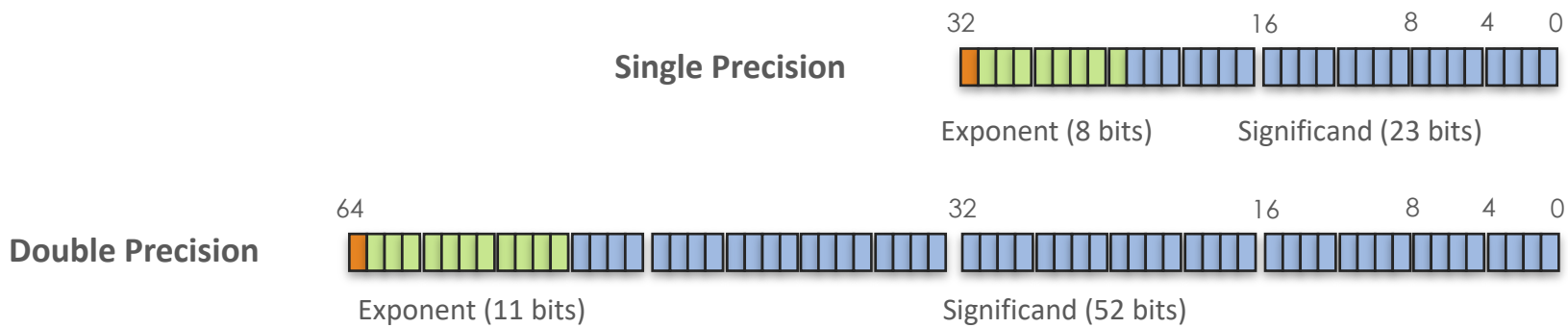
Floating-Point Analysis Tools

Mike Lam

James Madison University

Motivation

- IEEE floating-point arithmetic
  - Ubiquitous in scientific computing
  - More bits => higher accuracy (usually)
  - Fewer bits => higher performance (usually)

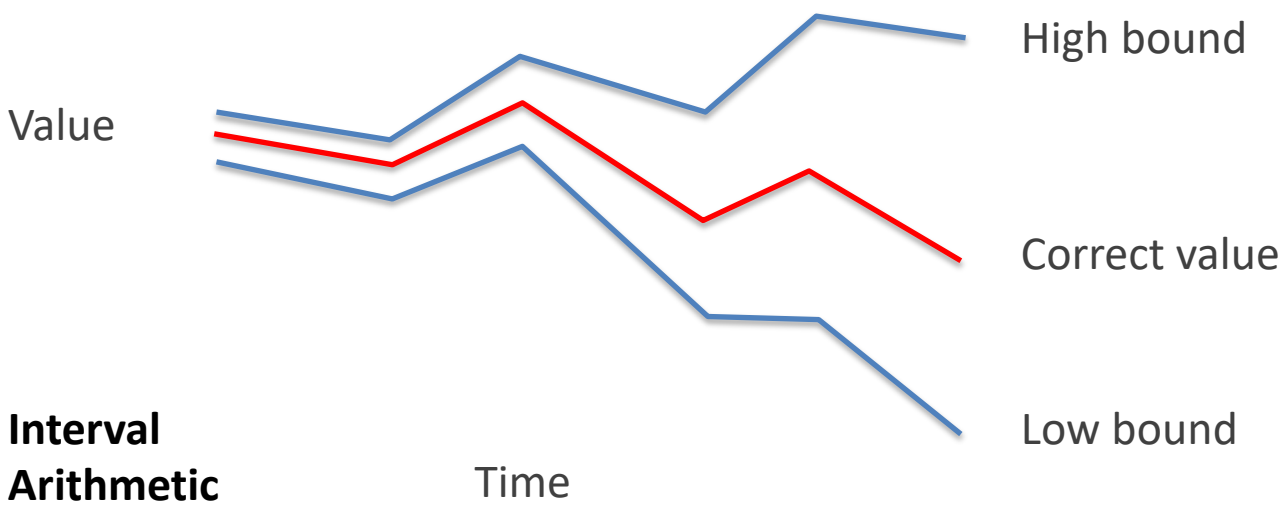


Issues

- Roundoff error
  - Finite-width field; must round to fit
  - Not uniform across range
- Cancellation
  - Loss of significant digits due to subtraction
- Non-associativity
  - Optimization can impact accuracy

Historical Approaches

- Forwards and backwards error analysis
- Interval and affine arithmetic
- “Just use doubles”



Modern Tools

- Static vs. dynamic
- Level of analysis
  - AST (“source-level”)
  - Linear IR
  - Assembly/machine code
- Sound vs. unsound
- Varying degrees of scalability

Tool Categories

- **Error bound**
  - Sound rounding error guarantees on small subroutines
- **Accuracy improvement**
  - Re-write portions of your program to reduce rounding error
- **Dynamic analysis**
  - “Quick & dirty” info about your program (e.g., cancellations)
- **New format prototyping**
  - Experiment with alternative representations (e.g., float or posit)
- **Mixed precision**
  - New version of program that uses single precision in some places for a speedup (e.g., 20% improvement on LULESH)

Tools

- Error Bound
  - FLUCTUAT
  - Gappa
  - CompCert
  - Astrée
  - FPTaylor
  - Rosa
  - Real2Float
  - PRECiSA
- Accuracy Improvement
  - Herbie/Herbgrind
  - Salsa
- Dynamic Analysis
  - CRAFT
  - S3FP
  - FLiT
- New Format Prototyping
  - SHVAL
  - Template Library
- Mixed Precision
  - CRAFT
  - Precimonious
  - FPTuner
  - Daisy
  - ADaPT

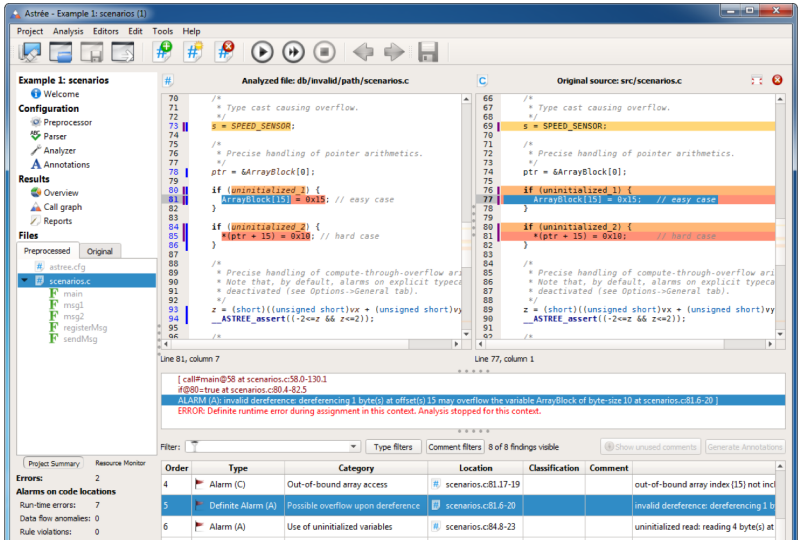
tinyurl.com/fpanalysis

Error Bound Tools

- Sound upper bound on output error
  - Sometimes other properties as well
- Mostly from PL community
  - Thus, mostly based on static analysis

- FLUCTUAT (2002)
- Gappa (2005)
- CompCert & Astrée (2015)
- FPTaylor (2015)
- Rosa (2017)
- Real2Float (2017)
- PRECiSA (2017)

Astrée



Error Bound Tools

Table IV. Comparisons of Informal and Formal Execution Times to Certify Roundoff Error Bounds Obtained with `Real2Float`, `FPTaylor`, and GAPPA

Benchmark	id	Informal timings		Formal timings		Speedup	
		$t_{\texttt{Real2Float}}$	$t_{\texttt{FPTaylor}}$	t_{Coq}	$t_{\text{HOL-LIGHT}}$	Informal	Formal
`rigidBody1`	a	0.58	1.84	0.36	10.2	3.18	12.9
`rigidBody2`	b	0.26	3.01	4.81	32.3	11.6	6.97
`kepler0`	c	0.22	4.93	0.29	45.5	22.5	98.9
`kepler1`	d	17.6	9.33	449	90.5	0.54	0.22
`kepler2`	e	16.5	19.1	297	274	1.16	0.94

$$\texttt{let obj\_kepler2}\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6 = [(x_1 * x_4 * (-x_1 + x_2 + x_3 - x_4 + x_5 + x_6) + x_2 * x_5 * (x_1 - x_2 + x_3 + x_4 - x_5 + x_6) + x_3 * x_6 * (x_1 + x_2 - x_3 + x_4 + x_5 - x_6) - x_2 * x_3 * x_4 - x_1 * x_3 * x_5 - x_1 * x_2 * x_6 - x_4 * x_5 * x_6,\ 0)];;$$

- **Mostly** unusable on HPC codes
  - May be useful for proving properties of microkernels or numerical library routines

From https://dl.acm.org/citation.cfm?doid=3034774.3015465

Accuracy Improvement Tools

- Transform code to improve accuracy
- Herbie (2015)
  - Improve accuracy of Scala code
  - Web interface: https://herbie.uwplse.org
  - Herbgrind: Valgrind plugin for C codes

- Salsa (2016)
  - Improve accuracy of C-like DSL
  - Unavailable

FPBench

Compare FP tools: ~ 100 accuracy benchmarks

Benchmark sources		Features used		Domains	
Rosa	37	Arithmetic	111	Textbooks	28
Herbie	28	Temporaries	57	Mathematics	24
Salsa	25	Comparison	33	Controls	10
FPTaylor	21	Loops	28	Science	10
		Exponents	16	(unknown)	39
		Trigonometry	15		
		Conditionals	10		

Interop to compose tools : FPCore, FPImp (w/ C & Scala transl)

Growing community: Utah, JM, MPI, UW, etc.

fpbench.org

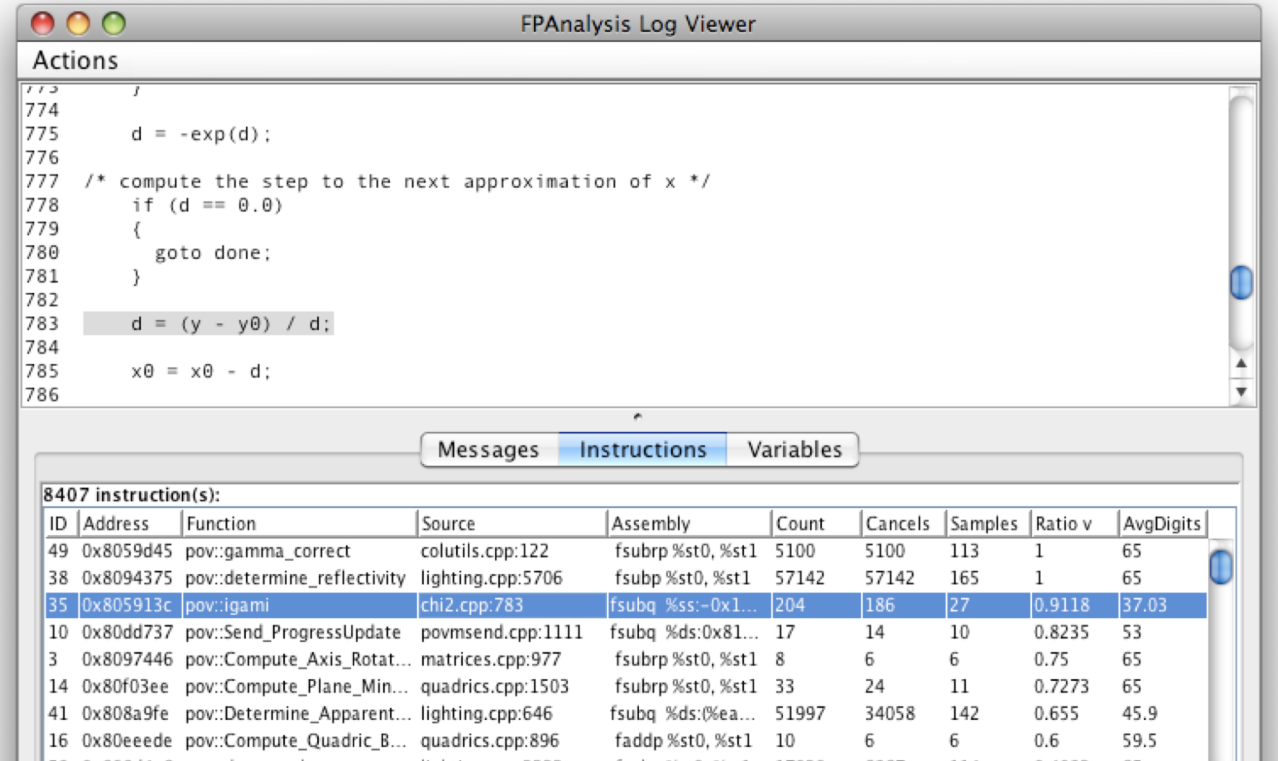
Credit: Pavel Panchekha (me@pavpanchekha.com)

Dynamic Analysis Tools

- Runtime heuristic tools
  - Usually at a compiled/binary level
- Unsound “quick and dirty” approaches
  - Results only valid for tested inputs
- Can inform mixed-precision efforts or help find bugs and numerical instabilities

CRAFT (2011)

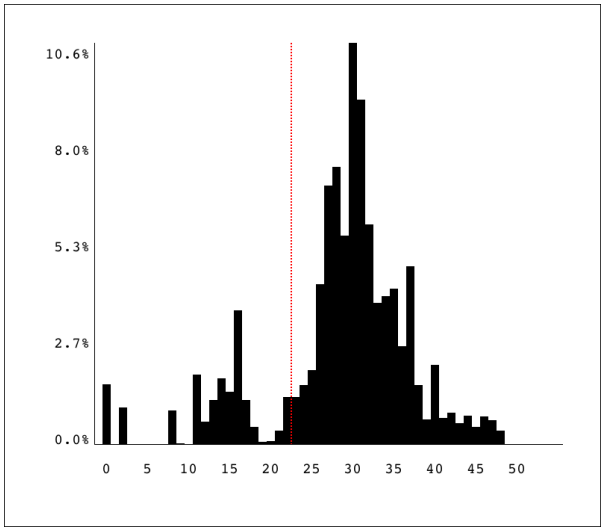
- Dynamic binary analysis via Dyninst
- Cancellation detection
- Range (exponent) tracking



CRAFT (2013)

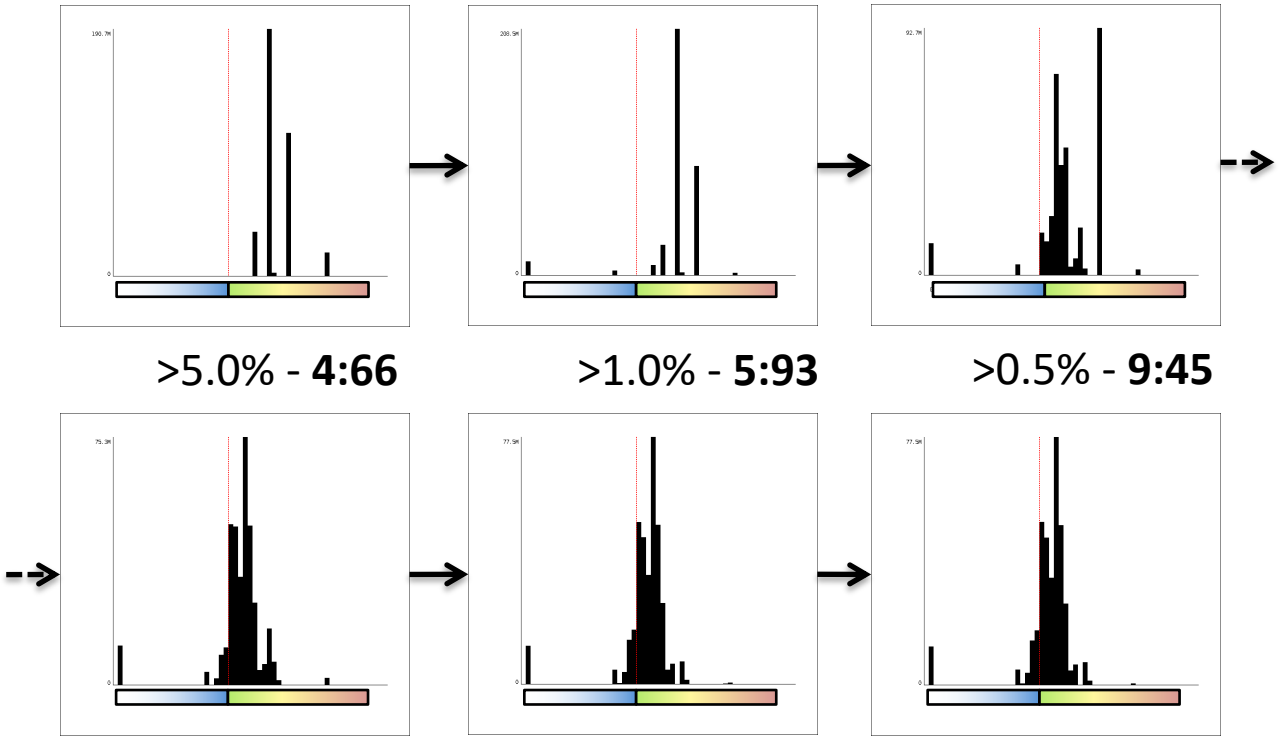
- Reduced-precision analysis
  - Simulate reduced precision via truncation
  - Report minimum precision for output of each instruction (w/ user-provided verification routine)
  - Generate histograms of precision requirements

```
▼ MODULE: 0x400000 "wtime.c"  Prec=51                [51 instruction(s)]
  ▼ FUNC: 0x400b60 "MAIN__"  Prec=51                  [49 instruction(s)]
    ▼ BBLK: 0x401088   Prec=40
        INSN: 0x401096 "mulsd xmm6, xmm10  [ep.f:189]"  Prec=39
        INSN: 0x40109b "mulsd xmm8, xmm10  [ep.f:190]"  Prec=37
        INSN: 0x4010a0 "subsd xmm6, xmm9  [ep.f:189]"  Prec=29
        INSN: 0x4010a5 "subsd xmm8, xmm9  [ep.f:190]"  Prec=27
        INSN: 0x4010b1 "mulsd xmm1, xmm6  [ep.f:191]"  Prec=25
        INSN: 0x4010b5 "mulsd xmm2, xmm8  [ep.f:191]"  Prec=26
        INSN: 0x4010ba "addsd xmm1, xmm2  [ep.f:191]"  Prec=27
    ▼ BBLK: 0x4010f2   Prec=51
        INSN: 0x401106 "addsd xmm0, xmm0  [ep.f:193]"  Prec=25
        INSN: 0x40110a "subsd xmm2, xmm1  [ep.f:193]"  Prec=25
        INSN: 0x40110e "divsd xmm0, xmm2  [ep.f:193]"  Prec=25
        INSN: 0x401112 "sqrtsd xmm0, xmm0  [ep.f:193]"  Prec=26
        INSN: 0x401136 "mulsd xmm6, xmm0  [ep.f:194]"  Prec=27
        INSN: 0x40113a "mulsd xmm8, xmm0  [ep.f:195]"  Prec=26
        INSN: 0x40113f "addsd xmm7, xmm6  [ep.f:198]"  Prec=51
        INSN: 0x40115d "addsd xmm6, xmm8  [ep.f:199]"  Prec=51
        INSN: 0x401178 "addsd xmm5, xmm9  [ep.f:197]"  Prec=0
```



CRAFT (2013)

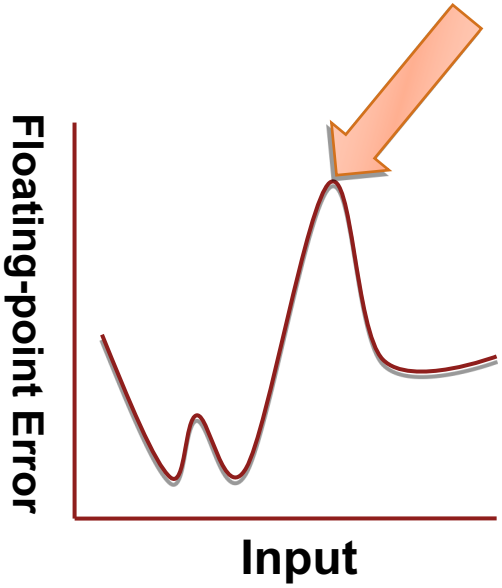
- Scalability via heuristic search: focus on most-executed instructions



https://github.com/crafthpc/craft

S3FP (2014)

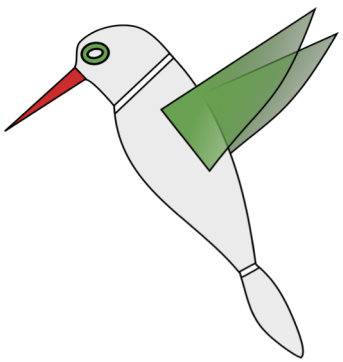
- Guided fuzzing finds program inputs that maximize floating-point error
  - Improves scalability to “real” codes (1200x1200 LAPACK routines on GPU)
  - Finds (much) higher errors than random testing



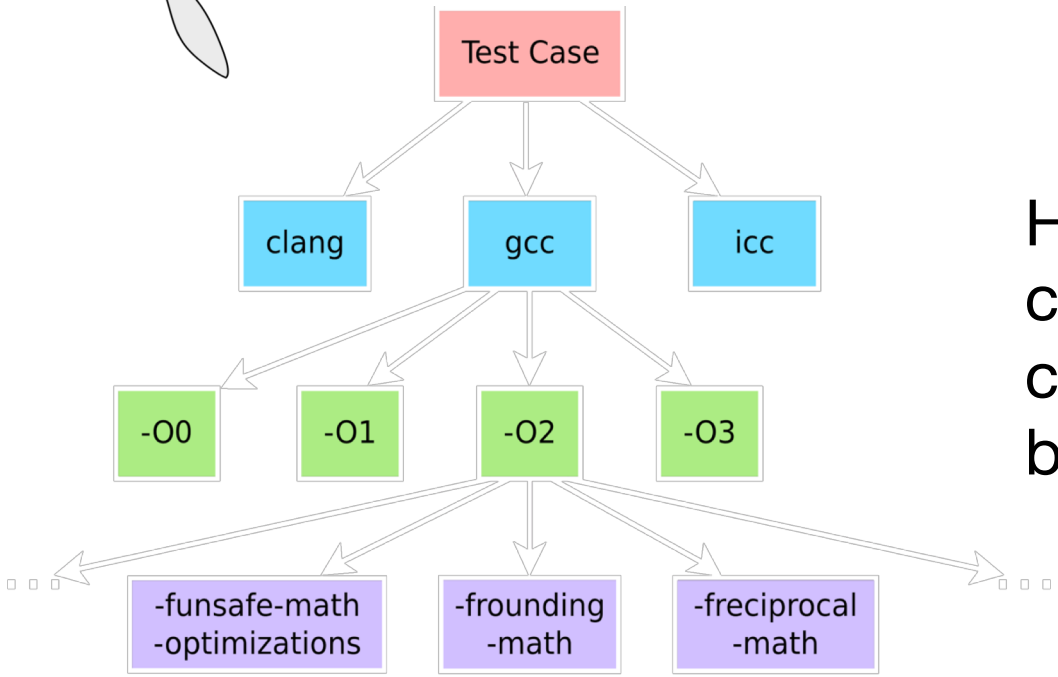
github.com/soarlab/S3FP

Credit: Ganesh Gopalakrishnan (ganesh@cs.utah.edu)

FLiT (2017)



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



Hundreds of compilations are compared against a baseline compilation.

Credit: Ganesh Gopalakrishnan (ganesh@cs.utah.edu)

Prototyping New Formats

- IEEE single or half precision
- Arbitrary precision (e.g., MPFR)
- “Next generation” formats
  - John Gustafson’s Unum and Posit formats
  - Peter Lindstrom’s ZFP compression
  - See CoNGA’18 talk
    - https://www.youtube.com/watch?v=snbBRzu4LVI

SHVAL (2016)

- SHadow Value Analysis Library (github.com/lam2mo/shval)
- Pintool for simulating alternative representations
- Maintains shadow values for all memory locations
- Executes shadow operations for all computation

```
double sum = 0.0;
for (int i = 0; i < 10; i++) {
    sum += 0.1;
}
printf("%25.20f\n", sum);
```

Fig. 3. Sample C program

```
Original machine code:                                  Inserted shadow code:

  pxor    xmm0, xmm0          (set to 0.0)               xmm[0] = convert(0.0)
  mov     eax, 10
  movsd   xmm1, 0x400628      (load 0.1)                 xmm[1] = convert(*(0x400628))
loop:
  sub     eax, 1
  addsd   xmm0, xmm1          (increment)                xmm[0] += xmm[1]
  jne     loop
  movsd   0x8(rsp), xmm0      (store sum)                mem[rsp+0x8] = xmm[0]
```

Fig. 4. Compiled assembly of program from Figure 3

Shadow Value Type	Exp Size	Frac Size	Final Shadow Value	Relative Error
32-bit (native single)	8	23	1.000000	1.19e-07
64-bit (native double)	11	52	1.000000000000000	0
128-bit GNU MPFR	15	112	1.00000000000000005551e+00	1.11e-16
Unum (3,2)	8	4	(0.9375, 1.1875)	0.059
Unum (3,4)	8	16	(0.9999847412109375, 1.0000457763671875)	1.53e-05
Unum (4,6)	16	64	1.00000000000000005551...182	1.11e-16

TABLE I
ANALYSIS RESULTS ON SAMPLE PROGRAM

SHVAL (2017)

- Aggregate error by instruction or memory location over time
  - Higher overhead, more information

- Apriltags case study using mixed precision
  - 1.7x speedup on average with only 4% error
  - 40% energy savings in embedded experiments

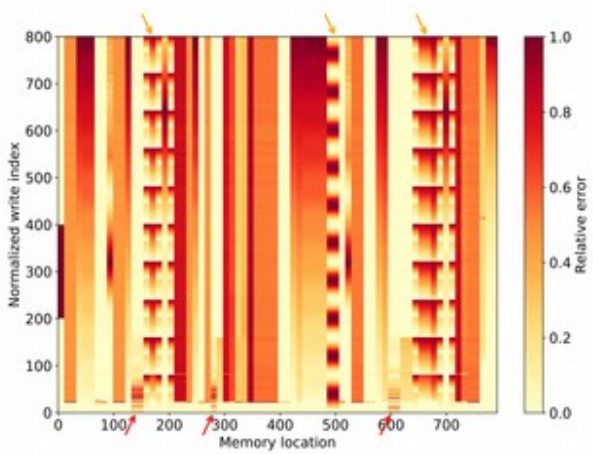


Fig. 1. Error trace per memory location. A darker pixel indicates higher error.

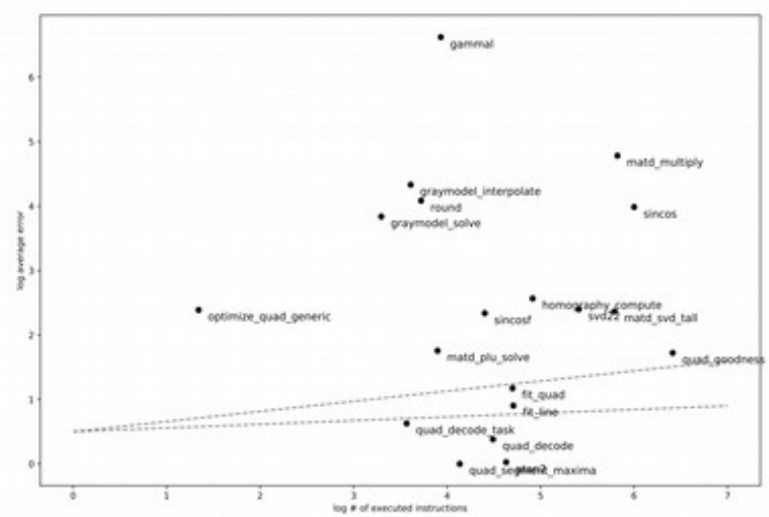
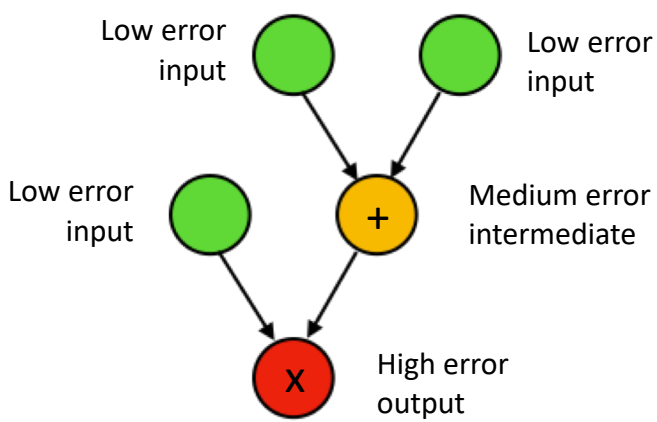


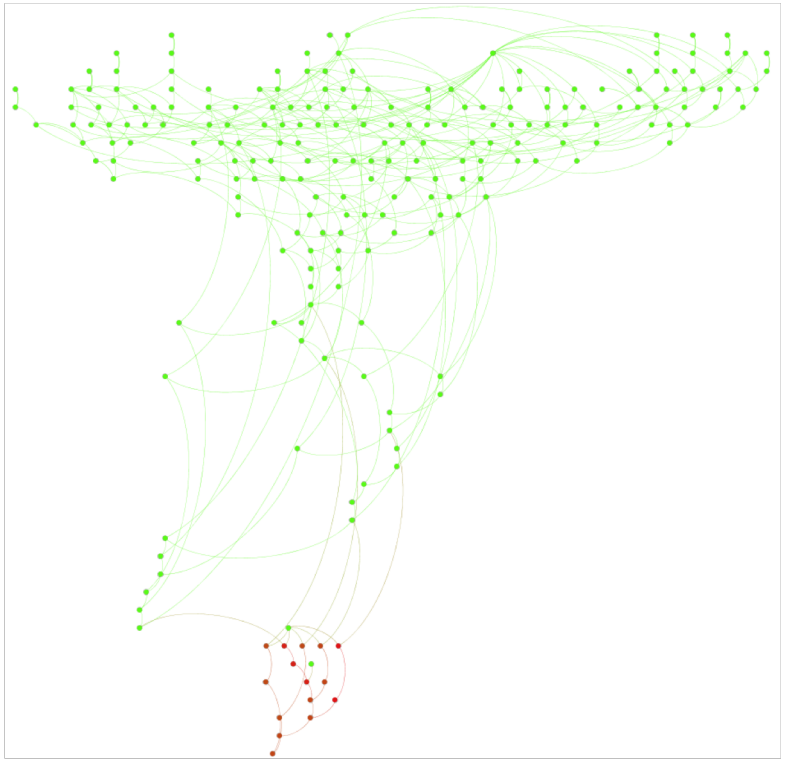
Fig. 8. Tradeoff between performance gain and error for every function in the Apriltags library.

Credit: Ramy Medhat (ramy.medhat@uwaterloo.ca)

SHVAL (2018)

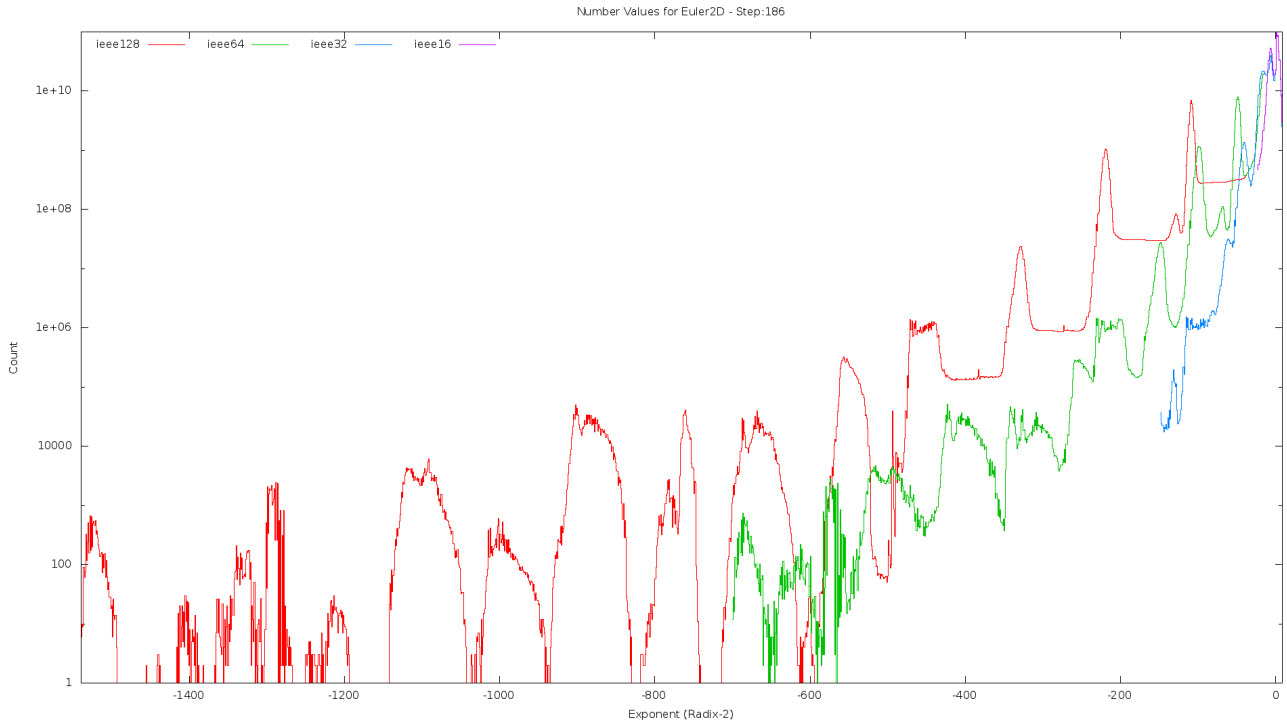
- Trace execution and build data flow graph
- Keep both single and double precision versions
- Color nodes by error (single vs. double)
- Inherent scaling issues
  - Idea: aggregate!





Template Library (2016)

- Configurable floating-point types
- Template metaprogramming for efficiency
- Optional instrumentation to measure statistics

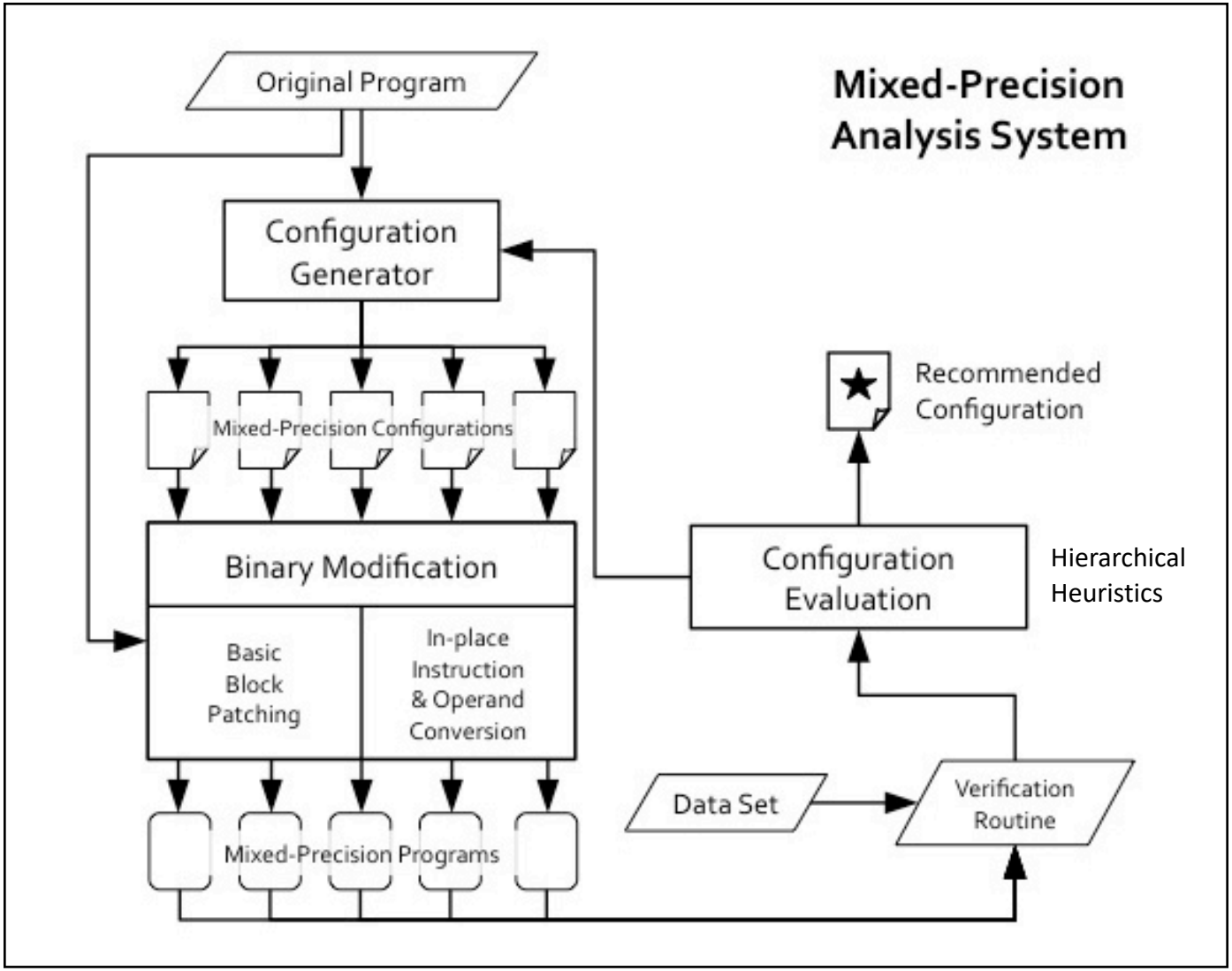


Credit: Scott Lloyd (lloyd23@llnl.gov)

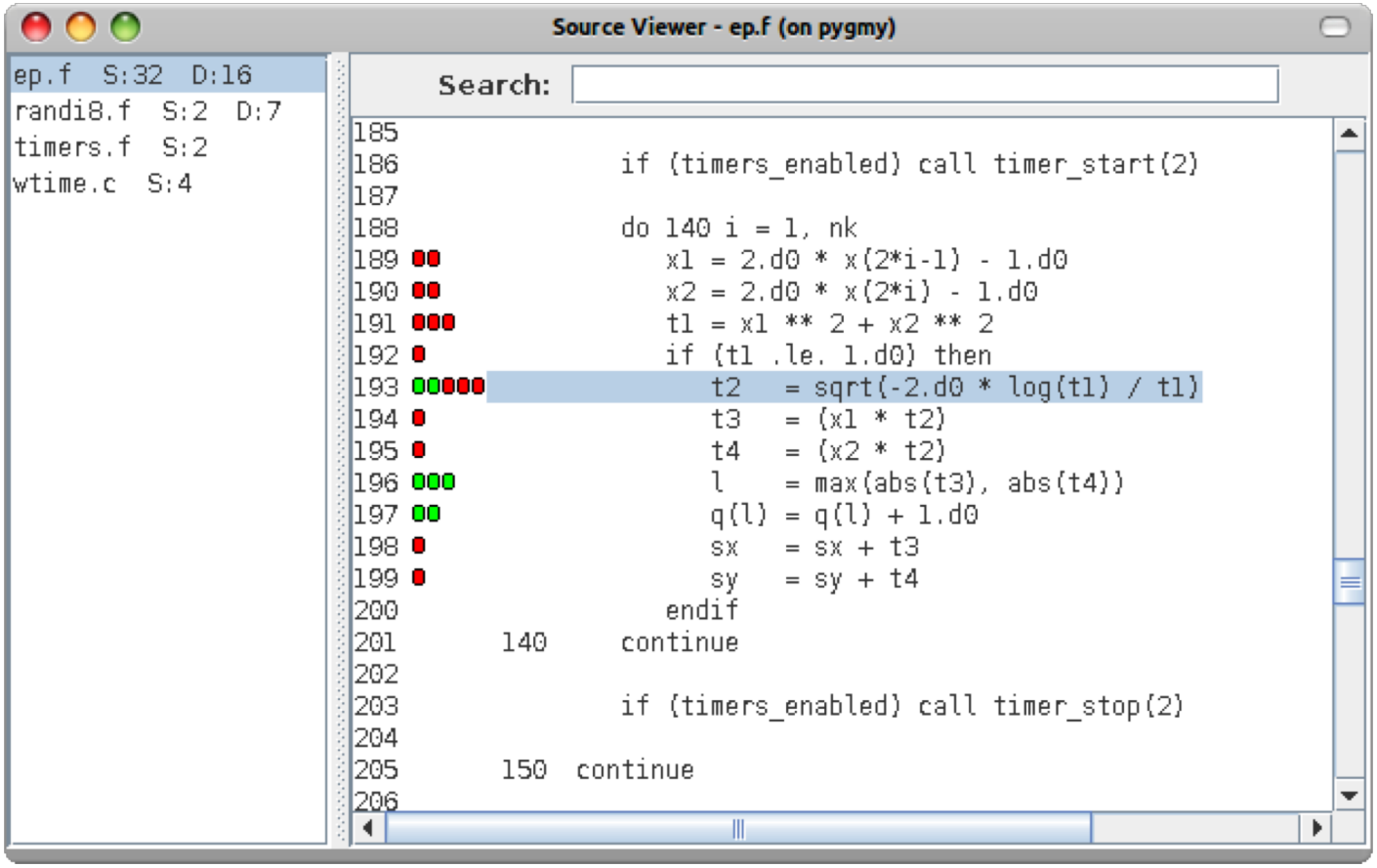
Mixed Precision Tools

- Early precision auto-tuning projects
  - CRAFT [ICS'13]
    - Mike Lam, UMD (now JMU)
    - Instruction-centric search using Dyninst
    - github.com/crafthpc/craft
  - Precimonious [SC'13]
    - Cindy Rubio-Gonzáles, UC Berkeley (now UC Davis)
    - Variable-centric search using LLVM
    - github.com/corvette-berkeley/precimonious
- Influx of new projects 2017-present
  - FPTuner, Daisy, ADaPT

CRAFT (2013)


Mixed-Precision
Analysis System
Original Program
Configuration
Generator
Mixed-Precision Configurations
Recommended
Configuration
Binary Modification
Basic
Block
Patching
In-place
Instruction
& Operand
Conversion
Configuration
Evaluation
Hierarchical
Heuristics
Data Set
Verification
Routine
Mixed-Precision Programs

CRAFT (2013)


Source Viewer - ep.f (on pygmy)
ep.f S:32 D:16
randi8.f S:2 D:7
timers.f S:2
wtime.c S:4
Search:
185
186 if (timers_enabled) call timer_start(2)
187
188 do 140 i = 1, nk
189 x1 = 2.d0 * x(2*i-1) - 1.d0
190 x2 = 2.d0 * x(2*i) - 1.d0
191 t1 = x1 ** 2 + x2 ** 2
192 if (t1 .le. 1.d0) then
193 t2 = sqrt(-2.d0 * log(t1) / t1)
194 t3 = (x1 * t2)
195 t4 = (x2 * t2)
196 l = max(abs(t3), abs(t4))
197 q(l) = q(l) + 1.d0
198 sx = sx + t3
199 sy = sy + t4
200 endif
201 140 continue
202
203 if (timers_enabled) call timer_stop(2)
204
205 150 continue
206

CRAFT (2013)

NAS Benchmark (name.CLASS)	Candidate Instructions	Configurations Tested	% Dynamic Replaced
bt.A	6,262	4,000	78.6
cg.A	956	255	5.6
ep.A	423	114	45.5
ft.A	426	74	0.2
lu.A	6,014	3,057	57.4
mg.A	1,393	437	36.6
sp.A	4,507	4,920	30.5

- Issue: does not translate well to a mixed-precision implementation
  - (difficult to go from instruction-level to source-level transformations)
- Issue: each instruction is tested in isolation
  - (combinations of replacements are often not passing)
- Issue: cannot ensure speedup
  - (instrumentation + casting overhead + lack of vectorization)

CRAFT (2013)

- Memory-based analysis
  - Replacement candidates: output operands
  - Analysis found several valid variable-level replacements
  - Does not consider computation (only storage)

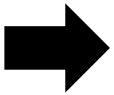
NAS Benchmark (name.CLASS)	Candidate Operands	Configurations Tested	% Executions Replaced
bt.A	2,342	300	97.0
cg.A	287	68	71.3
ep.A	236	59	37.9
ft.A	466	108	46.2
lu.A	1,742	104	99.9
mg.A	597	153	83.4
sp.A	1,525	1,094	88.9

CRAFT (2018)

- Basic handling of variable-level source tuning
  - Initial approach: look for and modify double-precision declarations in source code using pattern matching
  - Current work: use TypeForge (a Rose tool) for AST-level modifications

- Search for best speedup
  - Output is rewritten version of the fastest configuration that passed verification

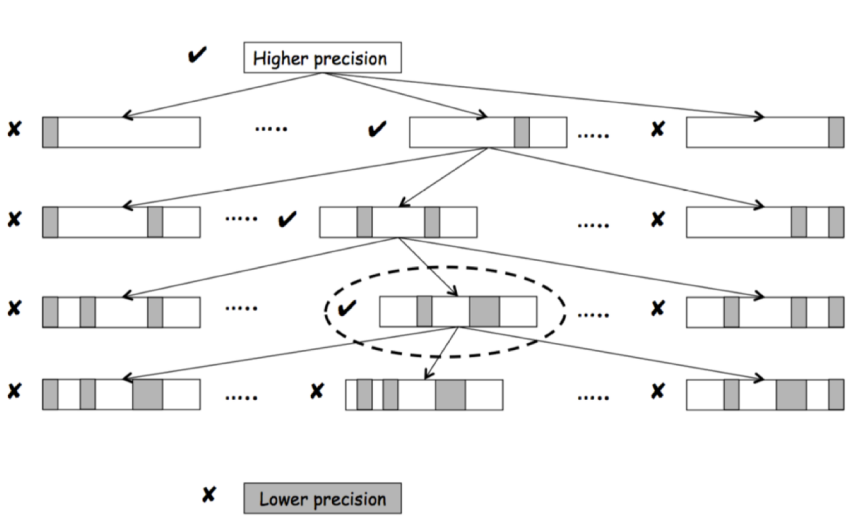
```
double x[];
double y[];
double dx[];
double dy[];
double alpha;
double beta;
double gamma;
```



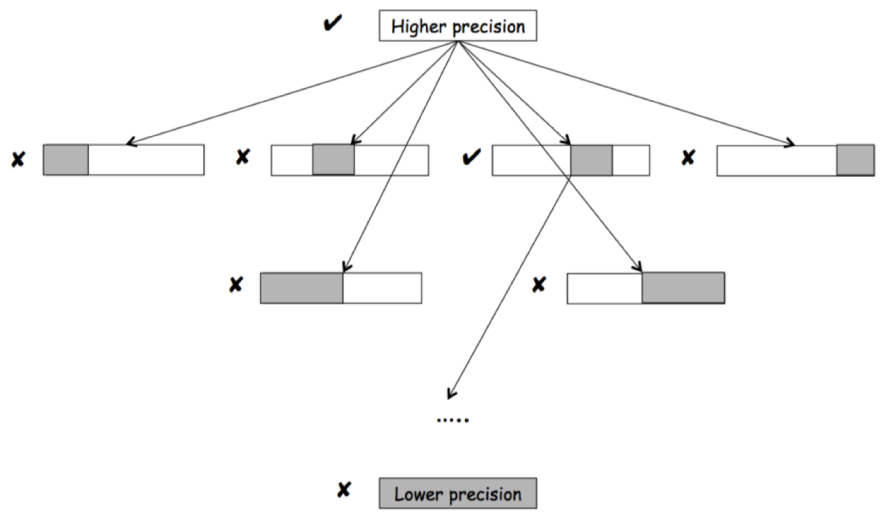
```
double x[];        (non-replaceable)
double y[];        (non-replaceable)
float dx[];        (replaceable)
float dy[];        (replaceable)
double alpha;      (non-replaceable)
double beta;       (replaceable)
float gamma;       (replaceable)
```

Precimonious (2013)

- Dynamic LLVM IR variable tuning
- Delta-Debugging search algorithm



(a) Quadratic number of variants



(b) Binary search on quadratic search space

Precimonious (2013)

		Original Type Configuration				Proposed Type Configuration					
	Program	L	D	F	Calls	L	D	F	Calls	# Config	mm:ss
GSL	bessel	0	18	0	0	0	18	0	0	130	37:11
GSL	gaussian	0	52	0	0	0	52	0	0	201	16:12
GSL	roots	0	19	0	0	0	0	19	0	3	1:03
GSL	polyroots	0	28	0	0	0	28	0	0	336	43:17
GSL	rootnewt	0	12	0	0	0	4	8	0	61	16:56
GSL	sum	0	31	0	0	0	9	22	0	325	28:14
GSL	fft	0	22	0	0	0	0	22	0	3	1:16
GSL	blas	0	17	0	0	0	0	17	0	3	1:06
NAS Class A	EP	0	13	0	4	0	5	8	4	111	23:53
NAS Class A	CG	0	32	0	3	0	2	30	3	44	0:57
NAS Class A	arclength	9	0	0	3	0	2	7	3	33	0:40
NAS Class A	simpsons	9	0	0	2	0	0	9	2	4	0:07

Credit: Cindy Rubio Gonzalez (crubio@ucdavis.edu)

Blame Analysis (2016)

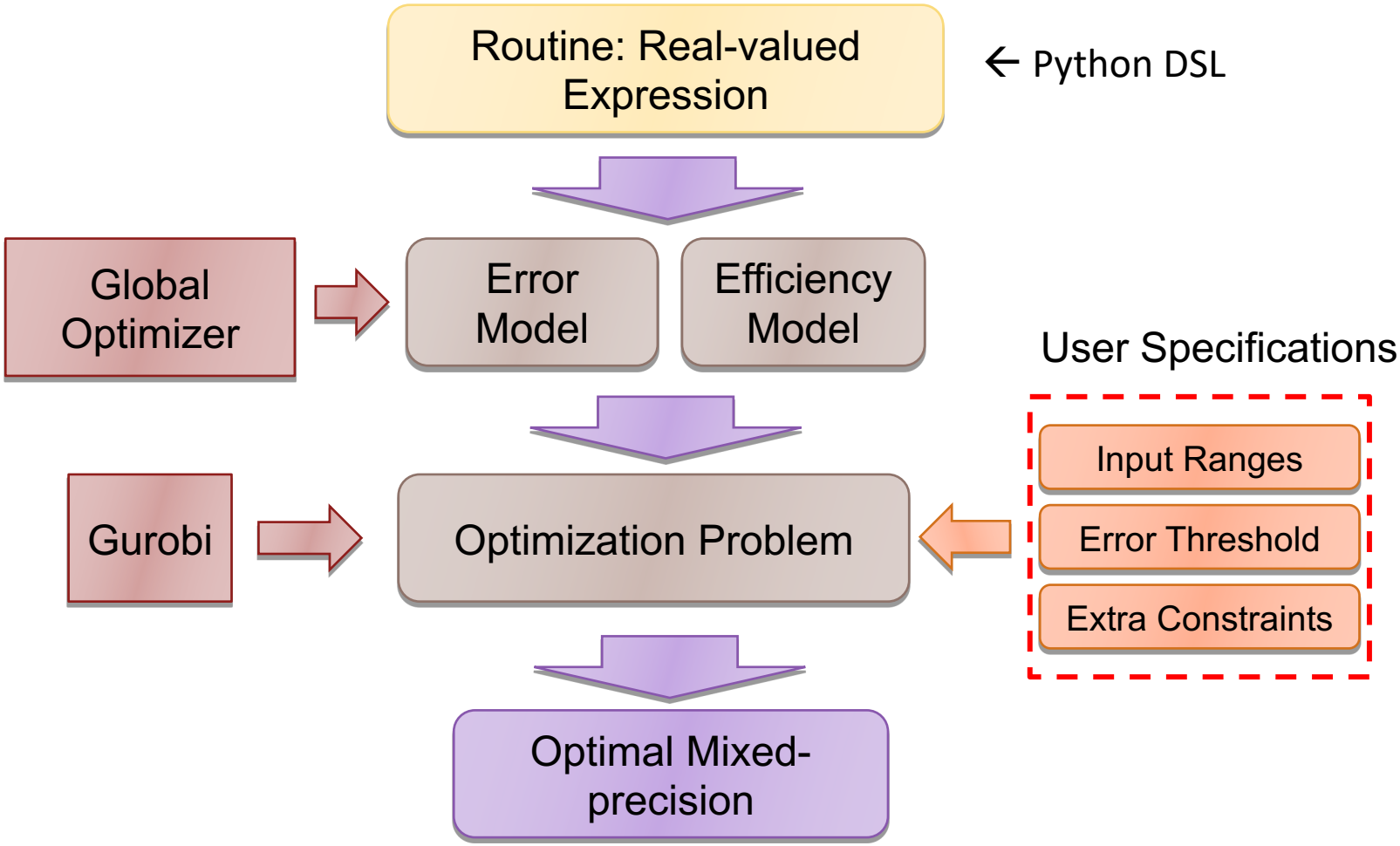
- Shadow value data-flow analysis
- Runs program once maintaining “blame sets” to infer precision requirements
- Integrating blame analysis results causes Precimonious to converge significantly faster (9x on average)
- Lack of documentation

https://github.com/corvette-berkeley/shadow-execution

HiFPTuner (2018)

- Extension of Precimonious
- Builds network of variables using dependency analysis and edge profiling
- Hierarchical mixed precision search
- Claims better scaling

FPTuner (2017)



Credit: Ganesh Gopalakrishnan (ganesh@cs.utah.edu)

https://github.com/soarlab/FPTuner

FPTuner (2017)

```
lam26@quartz2300:micro$ python3 ../../bin/fptuner.py -e 0.001 --aopt glpk --linear-tc reduction
==== error bound : 0.001 ====
Total # of operators: 1023
# of 32-bit operators: 1023
# of 64-bit operators: 0

---- alloc. ----
Group 0 : 32-bit
Group 1 : 32-bit
Group 2 : 32-bit
Group 3 : 32-bit
Group 4 : 32-bit
Group 5 : 32-bit
Group 6 : 32-bit
Group 7 : 32-bit
Group 8 : 32-bit
Group 9 : 32-bit
----------------

# L2H castings: 0
# H2L castings: 0
# Castings: 0

Expression:
(+ (+ (+ (+ (+ (+ (+ (+ (+ (v0) (v256)) (+ (v128) (v384))) (+ (+ (v64) (v320))
      (+ (v192) (v448)))) (+ (+ (+ (v32) (v288)) (+ (v160) (v416)))
     (+ (+ (v96) (v352)) (+ (v224) (v480))))) (+ (+ (+ (+ (v16) (v272))
      (+ (v144) (v400))) (+ (+ (v80) (v336)) (+ (v208) (v464))))
    (+ (+ (+ (v48) (v304)) (+ (v176) (v432))) (+ (+ (v112) (v368)) (+ (v240)
      (v496))))))) (+ (+ (+ (+ (+ (v8) (v264)) (+ (v136) (v392))) (+ (+ (v72)
      (v328)) (+ (v200) (v456)))) (+ (+ (+ (v40) (v296)) (+ (v168) (v424)))
     (+ (+ (v104) (v360)) (+ (v232) (v488))))) (+ (+ (+ (v24) (v280))
      (+ (v152) (v408))) (+ (+ (v88) (v344)) (+ (v216) (v472))))
    (+ (+ (+ (v56) (v312)) (+ (v184) (v440))) (+ (+ (v120) (v376)) (+ (v248)
      (v504))))))) (+ (+ (+ (+ (+ (v4) (v260)) (+ (v132) (v388)))
     (+ (+ (v68) (v324)) (+ (v196) (v452)))) (+ (+ (+ (v36) (v292)) (+ (v164)
      (v420))) (+ (+ (v100) (v356)) (+ (v228) (v484))))) (+ (+ (+ (+ (v20)
      (v276)) (+ (v148) (v404))) (+ (+ (v84) (v340)) (+ (v212) (v468))))
    (+ (+ (+ (v52) (v308)) (+ (v180) (v436))) (+ (+ (v116) (v372)) (+ (v244)
      (v500))))))) (+ (+ (+ (+ (v12) (v268)) (+ (v140) (v396)))
     (+ (+ (v76) (v332)) (+ (v204) (v460))) (+ (+ (+ (v44) (v300)) (+ (v172)
      (v428))) (+ (+ (v108) (v364)) (+ (v236) (v492))))) (+ (+ (+ (v28)
      (v284)) (+ (v156) (v412))) (+ (+ (v92) (v348)) (+ (v220) (v476))))
    (+ (+ (+ (v60) (v316)) (+ (v188) (v444))) (+ (+ (v124) (v380)) (+ (v252)
      (v508))))))))) (+ (+ (+ (+ (+ (+ (v2) (v258)) (+ (v130) (v386)))
     (+ (+ (v66) (v322)) (+ (v194) (v450)))) (+ (+ (+ (v34) (v290)) (+ (v162)
      (v418))) (+ (+ (v98) (v354)) (+ (v226) (v482))))) (+ (+ (+ (+ (v18)
      (v274)) (+ (v146) (v402))) (+ (+ (v82) (v338)) (+ (v210) (v466))))
    (+ (+ (+ (v50) (v306)) (+ (v178) (v434))) (+ (+ (v114) (v370)) (+ (v242)
      (v498))))) (+ (+ (+ (+ (v10) (v266)) (+ (v138) (v394)))
     (+ (+ (v74) (v330)) (+ (v202) (v458)))) (+ (+ (+ (v42) (v298)) (+ (v170)
      (v426))) (+ (+ (v106) (v362)) (+ (v234) (v490))))) (+ (+ (+ (+ (v26)
      (v282)) (+ (v154) (v410))) (+ (+ (v90) (v346)) (+ (v218) (v474))))
    (+ (+ (+ (v58) (v314)) (+ (v186) (v442))) (+ (+ (v122) (v378)) (+ (v250)
      (v506))))))) (+ (+ (+ (+ (+ (v6) (v262)) (+ (v134) (v390)))
     (+ (+ (v70) (v326)) (+ (v198) (v454)))) (+ (+ (+ (v38) (v294)) (+ (v166)
      (v422))) (+ (+ (v102) (v358)) (+ (v230) (v486))))) (+ (+ (+ (+ (v22)
      (v278)) (+ (v150) (v406))) (+ (+ (v86) (v342)) (+ (v214) (v470))))
    (+ (+ (+ (v54) (v310)) (+ (v182) (v438))) (+ (+ (v118) (v374)) (+ (v246)
      (v502))))))) (+ (+ (+ (+ (+ (v14) (v270)) (+ (v142) (v398)))
     (+ (+ (v78) (v334)) (+ (v206) (v462)))) (+ (+ (+ (v46) (v302)) (+ (v174)
      (v430))) (+ (+ (v110) (v366)) (+ (v238) (v494))))) (+ (+ (+ (+ (v30)
      (v286)) (+ (v158) (v414))) (+ (+ (v94) (v350)) (+ (v222) (v478))))
    (+ (+ (+ (v62) (v318)) (+ (v190) (v446))) (+ (+ (v126) (v382)) (+ (v254)
```

Error threshold: 1e-3
(more permissive)

```
lam26@quartz2300:micro$ python3 ../../bin/fptuner.py -e 0.0001 --aopt glpk --linear-tc reductio
==== error bound : 0.0001 ====
Total # of operators: 1023
# of 32-bit operators: 896
# of 64-bit operators: 127

---- alloc. ----
Group 0 : 32-bit
Group 1 : 32-bit
Group 2 : 32-bit
Group 3 : 64-bit
Group 4 : 64-bit
Group 5 : 64-bit
Group 6 : 64-bit
Group 7 : 64-bit
Group 8 : 64-bit
Group 9 : 64-bit
----------------

# L2H castings: 128
# H2L castings: 0
# Castings: 128

Expression:
(+ (+ (+ (+ (+ (+ (+ (+ (+ (v0) (v256)) (+ (v128) (v384))) (+ (+ (v64) (v320))
      (+ (v192) (v448)))) (+ (+ (+ (v32) (v288)) (+ (v160) (v416)))
     (+ (+ (v96) (v352)) (+ (v224) (v480))))) (+ (+ (+ (+ (v16) (v272))
      (+ (v144) (v400))) (+ (+ (v80) (v336)) (+ (v208) (v464))))
    (+ (+ (+ (v48) (v304)) (+ (v176) (v432))) (+ (+ (v112) (v368)) (+ (v240)
      (v496))))))) (+ (+ (+ (+ (+ (v8) (v264)) (+ (v136) (v392))) (+ (+ (v72)
      (v328)) (+ (v200) (v456)))) (+ (+ (+ (v40) (v296)) (+ (v168) (v424)))
     (+ (+ (v104) (v360)) (+ (v232) (v488))))) (+ (+ (+ (v24) (v280))
      (+ (v152) (v408))) (+ (+ (v88) (v344)) (+ (v216) (v472))))
    (+ (+ (+ (v56) (v312)) (+ (v184) (v440))) (+ (+ (v120) (v376)) (+ (v248)
      (v504))))))) (+ (+ (+ (+ (+ (v4) (v260)) (+ (v132) (v388)))
     (+ (+ (v68) (v324)) (+ (v196) (v452)))) (+ (+ (+ (v36) (v292)) (+ (v164)
      (v420))) (+ (+ (v100) (v356)) (+ (v228) (v484))))) (+ (+ (+ (+ (v20)
      (v276)) (+ (v148) (v404))) (+ (+ (v84) (v340)) (+ (v212) (v468))))
    (+ (+ (+ (v52) (v308)) (+ (v180) (v436))) (+ (+ (v116) (v372)) (+ (v244)
      (v500))))))) (+ (+ (+ (+ (v12) (v268)) (+ (v140) (v396)))
     (+ (+ (v76) (v332)) (+ (v204) (v460))) (+ (+ (+ (v44) (v300)) (+ (v172)
      (v428))) (+ (+ (v108) (v364)) (+ (v236) (v492))))) (+ (+ (+ (v28)
      (v284)) (+ (v156) (v412))) (+ (+ (v92) (v348)) (+ (v220) (v476))))
    (+ (+ (+ (v60) (v316)) (+ (v188) (v444))) (+ (+ (v124) (v380)) (+ (v252)
      (v508))))))))) (+ (+ (+ (+ (+ (+ (v2) (v258)) (+ (v130) (v386)))
     (+ (+ (v66) (v322)) (+ (v194) (v450)))) (+ (+ (+ (v34) (v290)) (+ (v162)
      (v418))) (+ (+ (v98) (v354)) (+ (v226) (v482))))) (+ (+ (+ (+ (v18)
      (v274)) (+ (v146) (v402))) (+ (+ (v82) (v338)) (+ (v210) (v466))))
    (+ (+ (+ (v50) (v306)) (+ (v178) (v434))) (+ (+ (v114) (v370)) (+ (v242)
      (v498))))) (+ (+ (+ (+ (v10) (v266)) (+ (v138) (v394)))
     (+ (+ (v74) (v330)) (+ (v202) (v458)))) (+ (+ (+ (v42) (v298)) (+ (v170)
      (v426))) (+ (+ (v106) (v362)) (+ (v234) (v490))))) (+ (+ (+ (+ (v26)
      (v282)) (+ (v154) (v410))) (+ (+ (v90) (v346)) (+ (v218) (v474))))
    (+ (+ (+ (v58) (v314)) (+ (v186) (v442))) (+ (+ (v122) (v378)) (+ (v250)
      (v506))))))) (+ (+ (+ (+ (+ (v6) (v262)) (+ (v134) (v390)))
     (+ (+ (v70) (v326)) (+ (v198) (v454)))) (+ (+ (+ (v38) (v294)) (+ (v166)
      (v422))) (+ (+ (v102) (v358)) (+ (v230) (v486))))) (+ (+ (+ (+ (v22)
      (v278)) (+ (v150) (v406))) (+ (+ (v86) (v342)) (+ (v214) (v470))))
    (+ (+ (+ (v54) (v310)) (+ (v182) (v438))) (+ (+ (v118) (v374)) (+ (v246)
      (v502))))))) (+ (+ (+ (+ (+ (v14) (v270)) (+ (v142) (v398)))
     (+ (+ (v78) (v334)) (+ (v206) (v462)))) (+ (+ (+ (v46) (v302)) (+ (v174)
      (v430))) (+ (+ (v110) (v366)) (+ (v238) (v494))))) (+ (+ (+ (+ (v30)
      (v286)) (+ (v158) (v414))) (+ (+ (v94) (v350)) (+ (v222) (v478))))
    (+ (+ (+ (v62) (v318)) (+ (v190) (v446))) (+ (+ (v126) (v382)) (+ (v254)
```

Error threshold: 1e-4

```
lam26@quartz2300:micro$ python3 ../../bin/fptuner.py -e 0.00001 --aopt glpk --linear-tc reductio
==== error bound : 1e-05 ====
Total # of operators: 1023
# of 32-bit operators: 0
# of 64-bit operators: 1023

---- alloc. ----
Group 0 : 64-bit
Group 1 : 64-bit
Group 2 : 64-bit
Group 3 : 64-bit
Group 4 : 64-bit
Group 5 : 64-bit
Group 6 : 64-bit
Group 7 : 64-bit
Group 8 : 64-bit
Group 9 : 64-bit
----------------

# L2H castings: 0
# H2L castings: 0
# Castings: 0

Expression:
(+ (+ (+ (+ (+ (+ (+ (+ (+ (v0) (v256)) (+ (v128) (v384))) (+ (+ (v64) (v320))
      (+ (v192) (v448)))) (+ (+ (+ (v32) (v288)) (+ (v160) (v416)))
     (+ (+ (v96) (v352)) (+ (v224) (v480))))) (+ (+ (+ (+ (v16) (v272))
      (+ (v144) (v400))) (+ (+ (v80) (v336)) (+ (v208) (v464))))
    (+ (+ (+ (v48) (v304)) (+ (v176) (v432))) (+ (+ (v112) (v368)) (+ (v240)
      (v496))))))) (+ (+ (+ (+ (+ (v8) (v264)) (+ (v136) (v392))) (+ (+ (v72)
      (v328)) (+ (v200) (v456)))) (+ (+ (+ (v40) (v296)) (+ (v168) (v424)))
     (+ (+ (v104) (v360)) (+ (v232) (v488))))) (+ (+ (+ (v24) (v280))
      (+ (v152) (v408))) (+ (+ (v88) (v344)) (+ (v216) (v472))))
    (+ (+ (+ (v56) (v312)) (+ (v184) (v440))) (+ (+ (v120) (v376)) (+ (v248)
      (v504))))))) (+ (+ (+ (+ (+ (v4) (v260)) (+ (v132) (v388)))
     (+ (+ (v68) (v324)) (+ (v196) (v452)))) (+ (+ (+ (v36) (v292)) (+ (v164)
      (v420))) (+ (+ (v100) (v356)) (+ (v228) (v484))))) (+ (+ (+ (+ (v20)
      (v276)) (+ (v148) (v404))) (+ (+ (v84) (v340)) (+ (v212) (v468))))
    (+ (+ (+ (v52) (v308)) (+ (v180) (v436))) (+ (+ (v116) (v372)) (+ (v244)
      (v500))))))) (+ (+ (+ (+ (v12) (v268)) (+ (v140) (v396)))
     (+ (+ (v76) (v332)) (+ (v204) (v460))) (+ (+ (+ (v44) (v300)) (+ (v172)
      (v428))) (+ (+ (v108) (v364)) (+ (v236) (v492))))) (+ (+ (+ (v28)
      (v284)) (+ (v156) (v412))) (+ (+ (v92) (v348)) (+ (v220) (v476))))
    (+ (+ (+ (v60) (v316)) (+ (v188) (v444))) (+ (+ (v124) (v380)) (+ (v252)
      (v508))))))))) (+ (+ (+ (+ (+ (+ (v2) (v258)) (+ (v130) (v386)))
     (+ (+ (v66) (v322)) (+ (v194) (v450)))) (+ (+ (+ (v34) (v290)) (+ (v162)
      (v418))) (+ (+ (v98) (v354)) (+ (v226) (v482))))) (+ (+ (+ (+ (v18)
      (v274)) (+ (v146) (v402))) (+ (+ (v82) (v338)) (+ (v210) (v466))))
    (+ (+ (+ (v50) (v306)) (+ (v178) (v434))) (+ (+ (v114) (v370)) (+ (v242)
      (v498))))) (+ (+ (+ (+ (v10) (v266)) (+ (v138) (v394)))
     (+ (+ (v74) (v330)) (+ (v202) (v458)))) (+ (+ (+ (v42) (v298)) (+ (v170)
      (v426))) (+ (+ (v106) (v362)) (+ (v234) (v490))))) (+ (+ (+ (+ (v26)
      (v282)) (+ (v154) (v410))) (+ (+ (v90) (v346)) (+ (v218) (v474))))
    (+ (+ (+ (v58) (v314)) (+ (v186) (v442))) (+ (+ (v122) (v378)) (+ (v250)
      (v506))))))) (+ (+ (+ (+ (+ (v6) (v262)) (+ (v134) (v390)))
     (+ (+ (v70) (v326)) (+ (v198) (v454)))) (+ (+ (+ (v38) (v294)) (+ (v166)
      (v422))) (+ (+ (v102) (v358)) (+ (v230) (v486))))) (+ (+ (+ (+ (v22)
      (v278)) (+ (v150) (v406))) (+ (+ (v86) (v342)) (+ (v214) (v470))))
    (+ (+ (+ (v54) (v310)) (+ (v182) (v438))) (+ (+ (v118) (v374)) (+ (v246)
      (v502))))))) (+ (+ (+ (+ (+ (v14) (v270)) (+ (v142) (v398)))
     (+ (+ (v78) (v334)) (+ (v206) (v462)))) (+ (+ (+ (v46) (v302)) (+ (v174)
      (v430))) (+ (+ (v110) (v366)) (+ (v238) (v494))))) (+ (+ (+ (+ (v30)
      (v286)) (+ (v158) (v414))) (+ (+ (v94) (v350)) (+ (v222) (v478))))
    (+ (+ (+ (v62) (v318)) (+ (v190) (v446))) (+ (+ (v126) (v382)) (+ (v254)
```

Error threshold: 1e-5
(more strict)

Daisy (2017)

- Input: real-valued expression DSL that is a subset of Scala
- Aggregates many previous analyses in a modular analysis system
  - Error bounding
  - Accuracy improvement
  - Mixed precision
- Online interface: http://daisy.mpi-sws.org
- “Limited support for conditionals or loops”

ADaPT (2018)

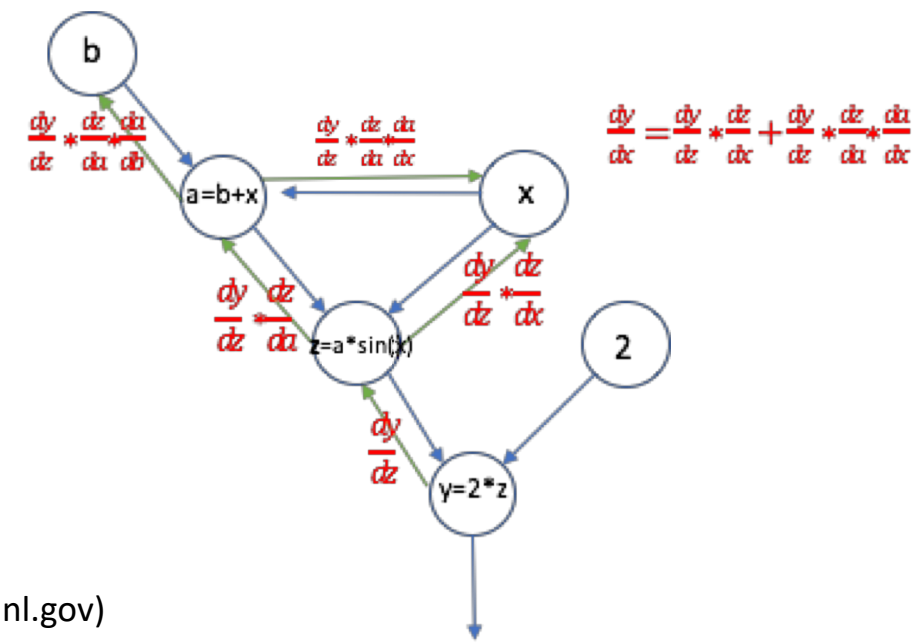
New tool developed at LLNL over the past year

Use first order Taylor series approximation to estimate the rounding errors in variables

$$\Delta y = f'(a)\ \Delta x \qquad \text{for } y=f(x) \text{ at } x=a$$

Obtain f'(a) using reverse-mode algorithmic differentiation (AD)

```
a = b+x
z = a*sin(x);
y = 2*z;
```



Credit: Harshitha Menon (gopalakrishn1@llnl.gov)

ADaPT (2018)

Original C Code

```
#include <iostream>



double sum = 0.0;
double inc = 0.1;

double do_sum() {
    int i;
    for (i = 0; i < 1000; i++) {
        sum += inc;
    }
    return sum;
}

int main() {


    do_sum();
    cout << sum << endl;



    return 0;
}
```

AD Instrumented Code

```
#include <iostream>
#include <adapt.h>
#include <adapt-impl.cpp>

AD_real sum = 0.0;
AD_real inc = 0.1;

AD_real do_sum() {
    int i;
    for (i = 0; i < 1000; i++) {
        sum += inc;
    }
    return sum;
}

int main() {
    AD_begin();
    AD_independent(inc, "inc");
    do_sum();
    cout << AD_value(sum) << endl;

    AD_dependent(sum, "sum", 8);
    AD_report();
    return 0;
}
```

– AD Libraries

– Type Changes

– Initialization

– Output

ADaPT (2018)

Used ADaPT on LULESH to create mixed precision sensitivity profile

Used the profile as a guide to develop a mixed precision version for a CUDA implementation of LULESH

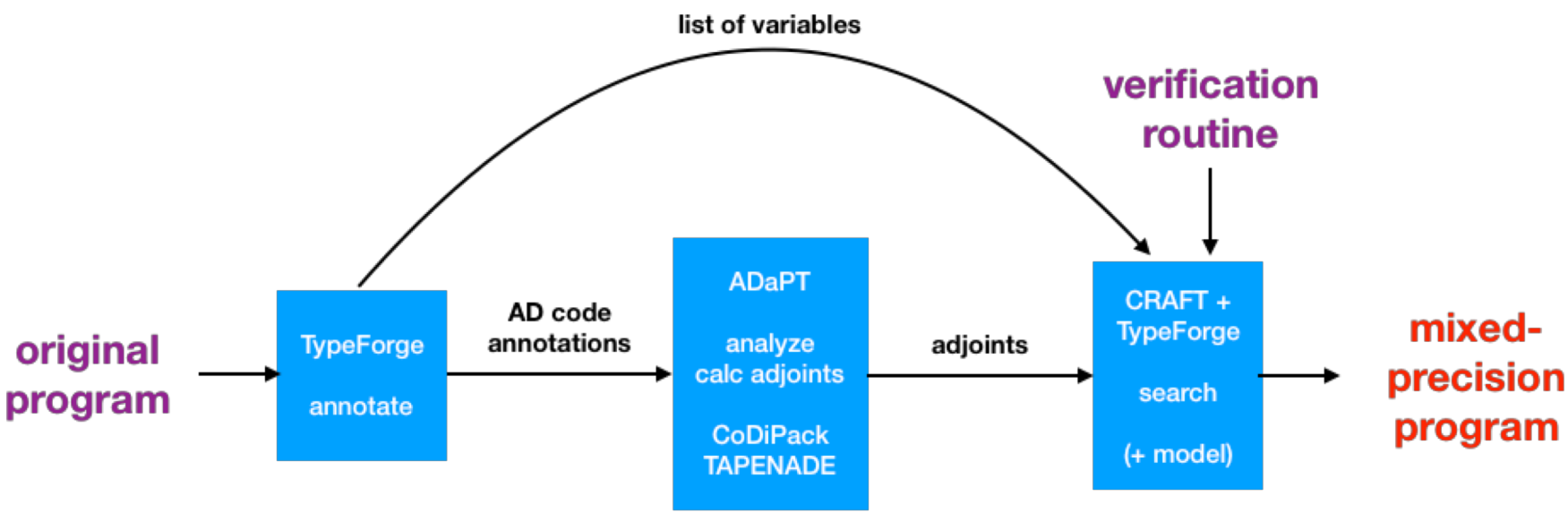
Achieved speedup of 20% within error threshold of 1e-11 on GPU

```
main
|__ TimeIncrement
|__ LagrangeLeapFrog
    |__ LagrangeNodal
    |   |__ CalcForceForNodes
    |   |   |__ CalcVolumeForceForElems
    |   |       |__ InitStressTermsForElems
    |   |       |__ IntegrateStressForElems
    |   |           |__ CollectDomainNodesToElemNodes
    |   |           |__ CalcElemShapeFunctionDerivatives
    |   |           |__ CalcElemNodeNormals
    |   |           |   |__ SumElemFaceNormal
    |   |           |__ SumElemStressesToNodeForces
    |   |          CalcHourglassControlForElems
    |   |           |__ CollectDomainNodesToElemNodes
    |   |           |__ CalcElemVolumeDerivative
    |   |           |   |__ VoluDer
    |   |           |__ CalcFBHourglassForceForElems
    |   |__ CalcAccelerationForNodes
    |   |__ ApplyAccelerationBoundaryConditionsForNodes
    |   |__ CalcVelocityForNodes
    |   |__ CalcPositionForNodes
    |__ LagrangeElements
    |   |__ CalcLagrangeElements
    |   |   |__ CalcKinematicsForElems
    |   |       |__ CollectDomainNodesToElemNodes
    |   |       |__ CalcElemVolume
    |   |       |   |__ CalcElemVolume
    |   |       |__ CalcElemCharacteristicLength
    |   |       |   |__ AreaFace
    |   |       |__ CalcElemShapeFunctionDerivatives
    |   |       |__ CalcElemVelocityGradient
    |   |__ CalcQForElems
    |   |   |__ CalcMonotonicQGradientsForElems
    |   |   |__ CalcMonotonicQForElems
    |   |       |__ CalcMonotonicQRegionForElems
    |   |__ ApplyMaterialPropertiesForElems
    |   |   |__ EvalEOSForElems
    |   |       |__ CalcEnergyForElems
    |   |       |   |__ CalcPressureForElems
    |   |       |__ CalcSoundSpeedForElems
    |   |__ UpdateVolumesForElems
    |__ CalcTimeConstraintsForElems
        |__ CalcCourantConstraintForElems
        |__ CalcHydroConstraintForElems
```

Credit: Harshitha Menon (gopalakrishn1@llnl.gov)

CRAFT+ADaPT+TypeForge

- Search process and integration via CRAFT
- Source-to-source translation via TypeForge
- Optional inclusion of ADaPT analysis to narrow search



Summary

- 10+ actively-maintained floating-point analysis projects
- Diverse goals and approaches
- Many avenues for future research

Thank you!

Acknowledgements:

- Jeff Hollingsworth
- Bronis de Supinski
- Barry Rountree
- Jeff Hittinger
- Scott Lloyd
- Harshitha Menon
- Markus Schordan
- Shelby Funk
- Ramy Medhat
- Garrett Folks
- Logan Moody
- Nkeng Atabong

For more information:

tinyurl.com/fpanalysis

Contact me: lam2mo@jmu.edu

JAMES MADISON UNIVERSITY.