Reduced-Precision Floating-Point Analysis via Binary Modification

Mike Lam, UMD

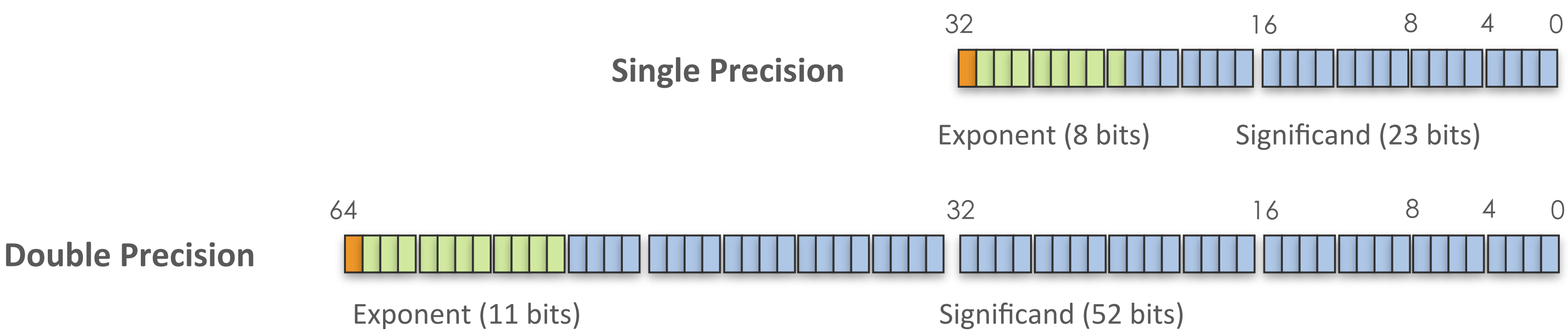
Jeff Hollingsworth, UMD

Context

Floating-point arithmetic represents real numbers as

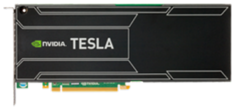
$$(\pm\ 1.frac \times 2^{exp})$$

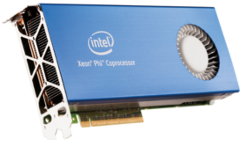
- – Sign bit
- – Exponent
- – Significand (“mantissa” or “fraction”)



Context

- Floating-point is ubiquitous **but problematic**
  - Rounding error
    - Accumulates after many operations
    - Not always intuitive (e.g., non-associative)
    - Naïve approach: higher precision
  - Lower precision is preferable
    - Tesla K20X is 2.3X faster in single precision
    - Xeon Phi is 2.0X faster in single precision
    - Single precision uses 50% of the memory bandwidth



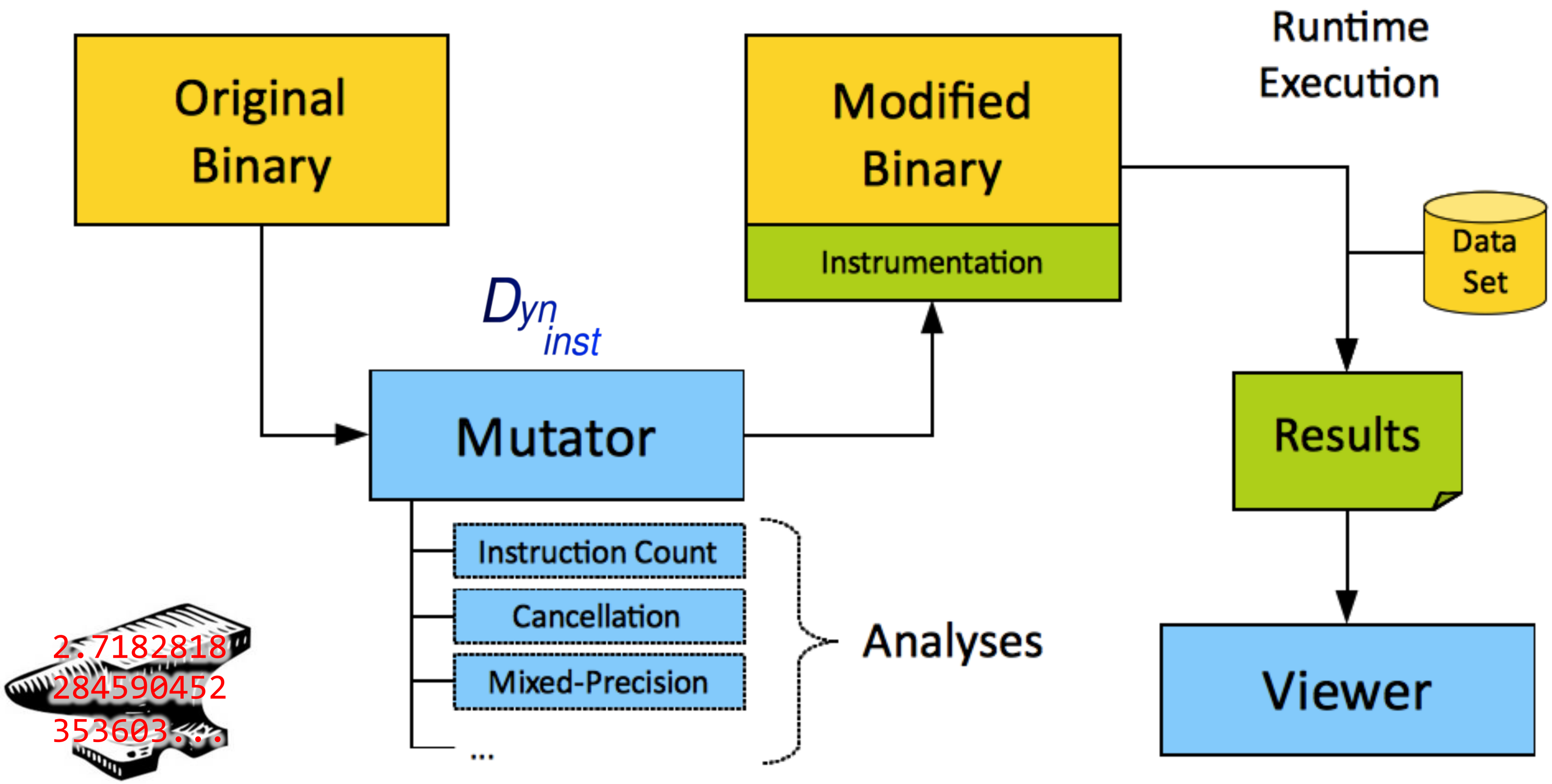


Research Contributions

- Software framework (CRAFT)
- Previous work
  - Cancellation detection [PARCO2012]
  - Mixed-precision configuration [ICS'13]
- Recent work
  - Reduced-precision analysis
- Future work

Framework

CRAFT: Configurable Runtime Analysis for Floating-point Tuning



Framework

- Dyninst: a binary analysis library
  - Parses executable files (InstructionAPI & ParseAPI)
  - Inserts instrumentation (DyninstAPI)
  - Supports full binary modification (PatchAPI)
  - Rewrites binary executable files (SymtabAPI)

Dyninst

- XED instruction decoder (from Intel's Pin)

Framework

- CRAFT framework
  - Dyninst-based binary mutator (C/C++)
  - Swing-based GUI viewers (Java)
  - Automated search scripts (Ruby)
  - Over 30K LOC total
  - LGPL on Sourceforge: `sf.net/p/crafthpc`

Cancellation Detection

- Loss of significant digits due to subtraction

```
  2.491264  (7)        1.613647  (7)
- 2.491252  (7)      - 1.613647  (7)
  0.000012  (2)        0.000000  (0)
```

(5 digits cancelled) (all digits cancelled)

- Cancellation detection
  - Instrument every addition and subtraction
  - Report cancellation events

Mixed Precision

- Frequent operations use single precision
- Crucial operations use double precision

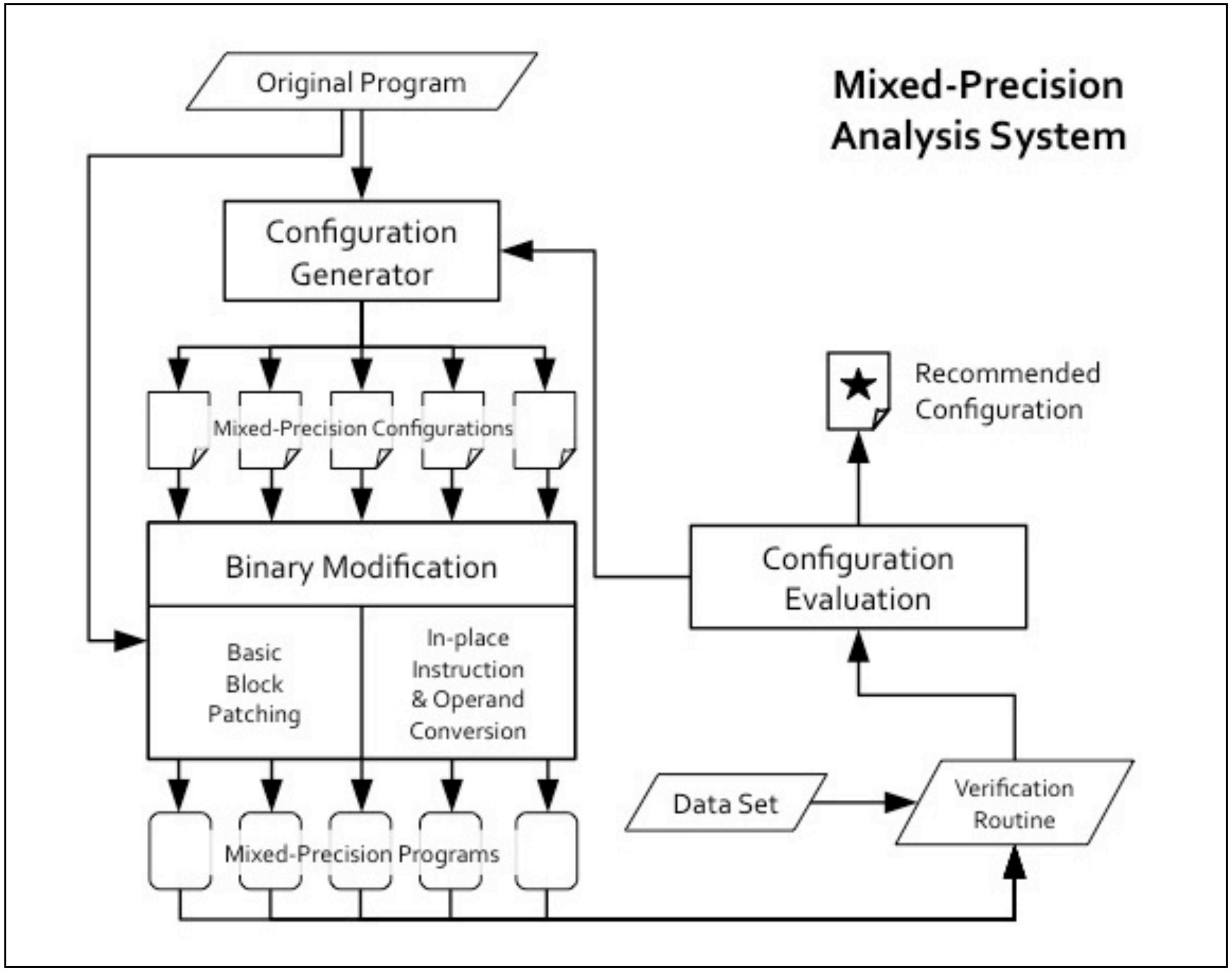
```
LU ← PA
solve Ly = Pb
solve Ux_0 = y
for k = 1, 2, ... do
    r_k ← b – Ax_{k-1}
    solve Ly = Pr_k
    solve Uz_k = y
    x_k ← x_{k-1} + z_k
    check for convergence
end for
```

Mixed-precision linear solver
[Buttari 2008]

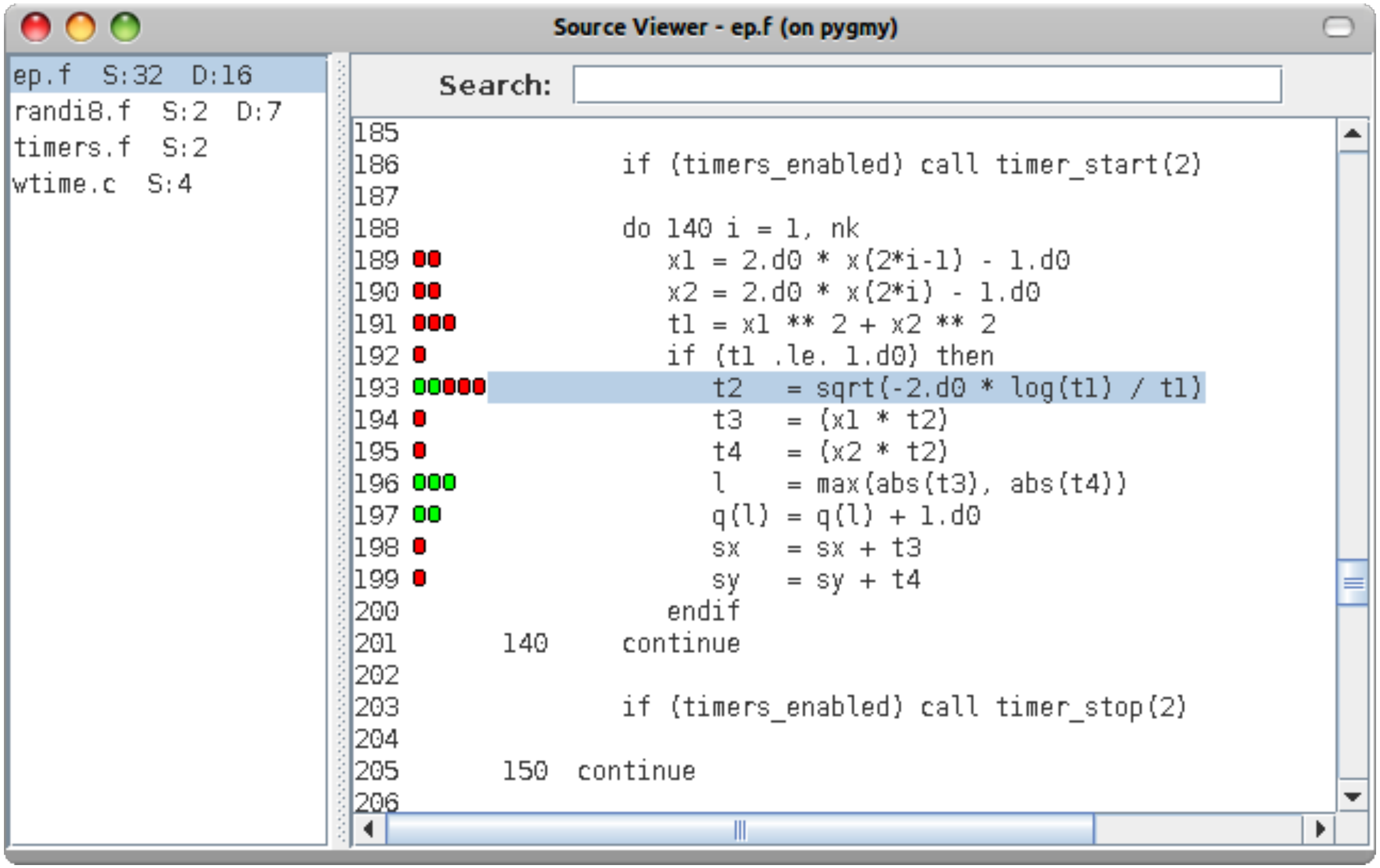
Red text indicates double-precision
(all other steps are single-precision)

50% speedup on average
(12X in special cases)

Automated Search



Automated Search



Mixed Precision: Results

Benchmark (name.CLASS)	Candidate Instructions	% Dynamic Replaced
bt.A	6,262	78.6
cg.A	956	5.6
ep.A	423	45.5
ft.A	426	0.2
lu.A	6,014	57.4
mg.A	1,393	36.6
sp.A	4,507	30.5

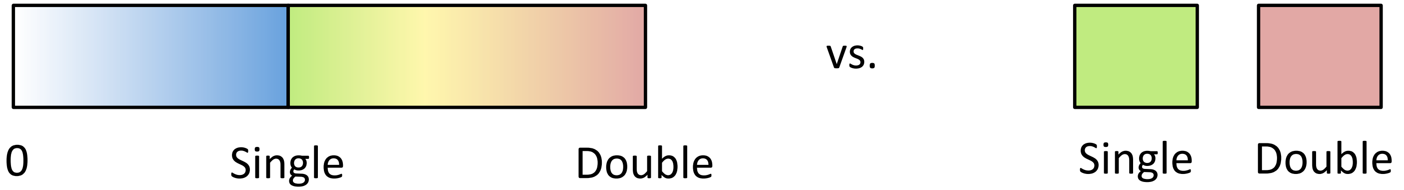
Mixed Precision: Conclusions

- Automated analysis can illuminate cancellation behavior
- Automated search can provide precision-level replacement insights
  – Still very coarse-grained w/ binary decision-making

Reduced Precision

- Simulate reduced precision with truncation
  - Truncate result after every operation
  - Allows zero up to double (64-bit) precision
  - Less overhead (fewer added operations)

- Search routine
  - Identifies component-level precision requirements



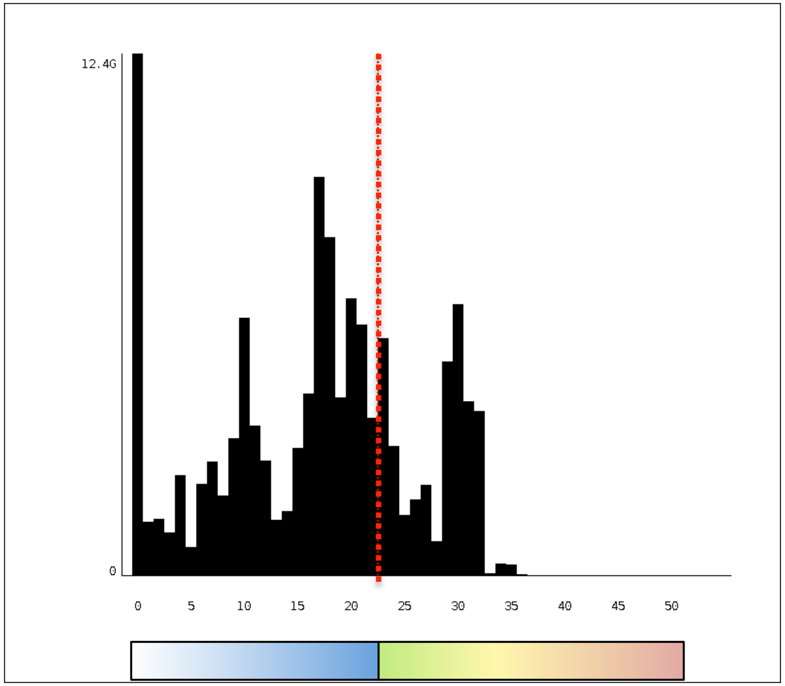
Reduced Precision

- Faster search convergence compared to mixed-precision analysis

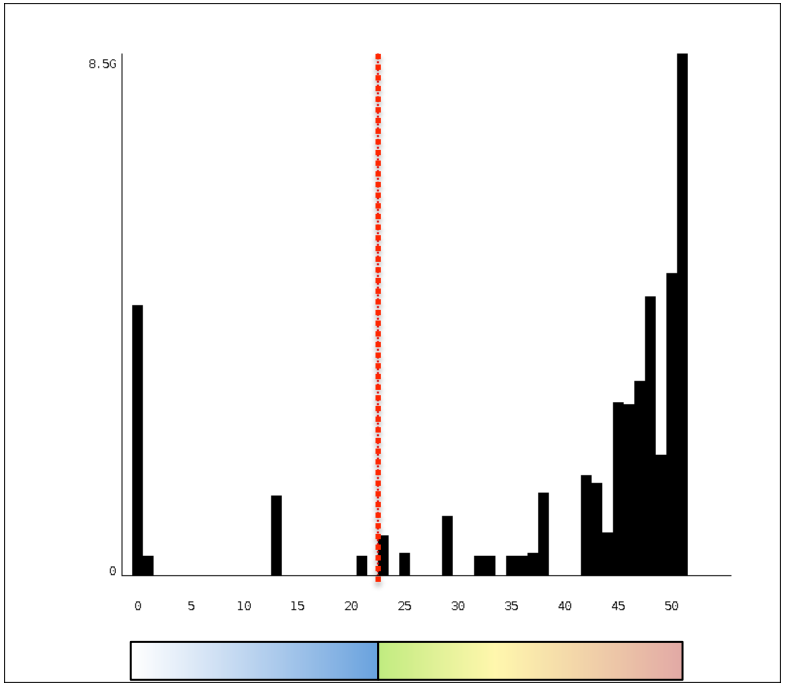
Benchmark	Original Wall time (s)	Speedup
cg.A	1,305	59.2%
ep.A	978	42.5%
ft.A	825	50.2%
lu.A	514,332	86.7%
mg.A	2,898	66.0%
sp.A	422,371	44.1%

Reduced Precision

- General precision requirement profiles

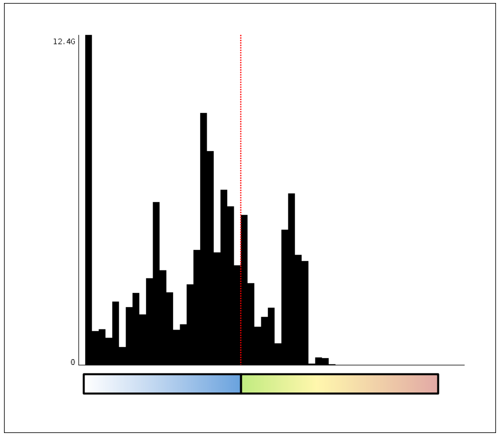


Low sensitivity

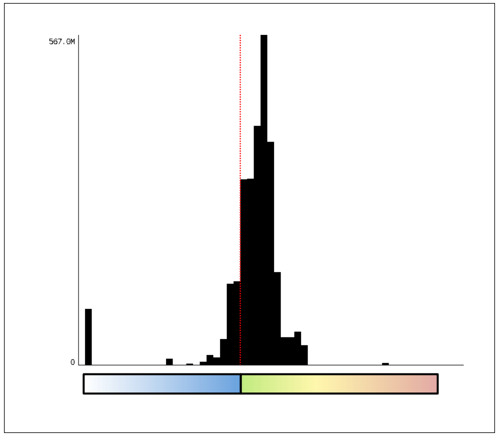


High sensitivity

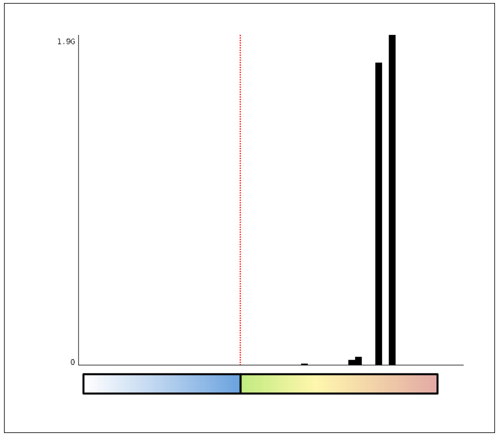
Reduced Precision: Results
NAS (top) & LAMMPS (bottom)



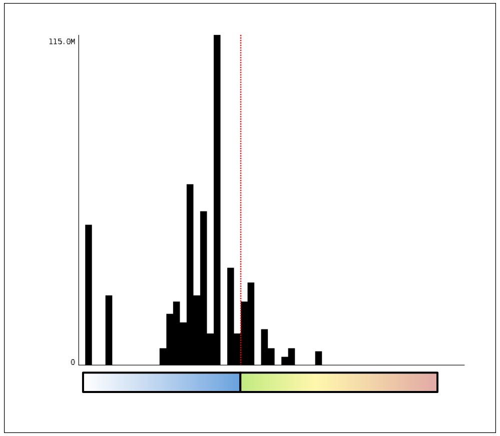
bt.A (78.6%)



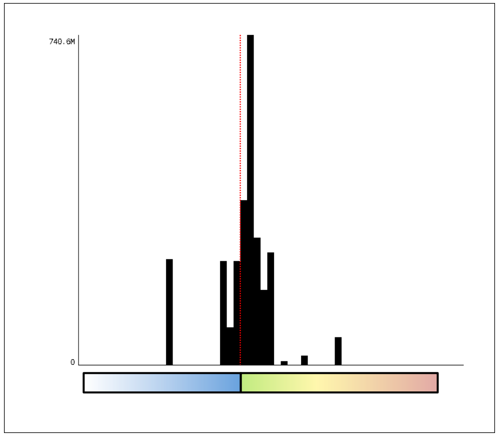
mg.A (36.6%)



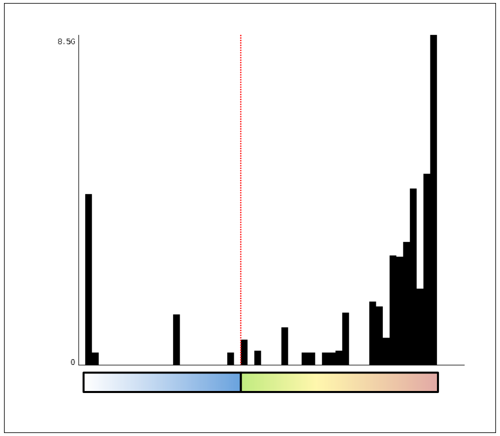
ft.A (0.2%)



chute

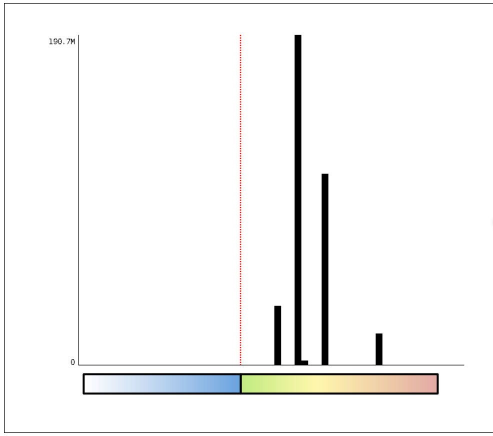


lj

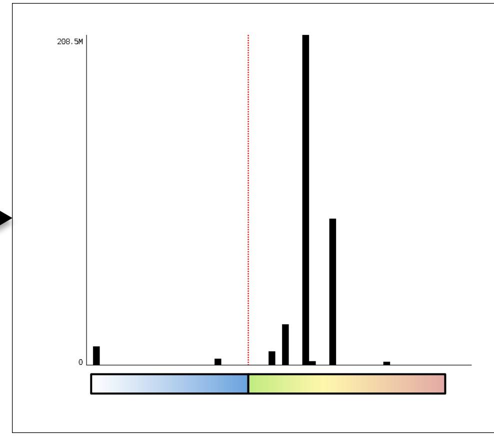


rhodo

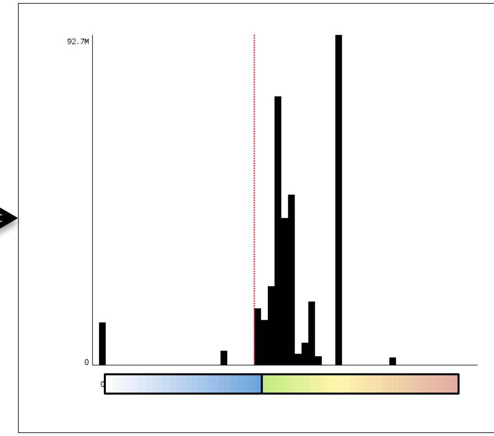
Reduced Precision: Results
NAS mg.W (incremental)



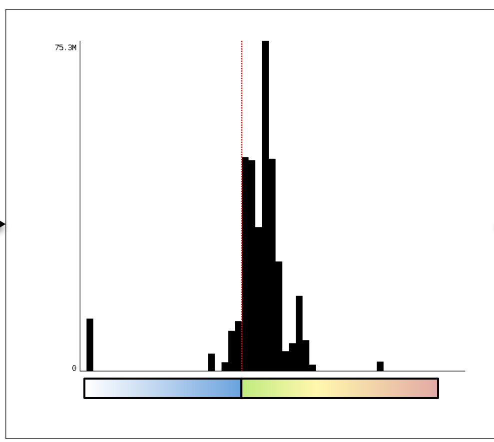
>5.0% - **4:66**



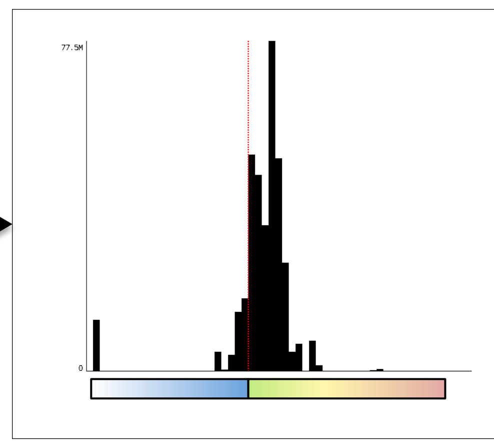
>1.0% - **5:93**



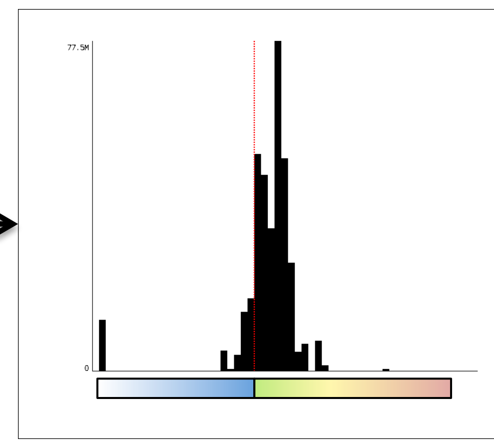
>0.5% - **9:45**



>0.1% - **15:45**



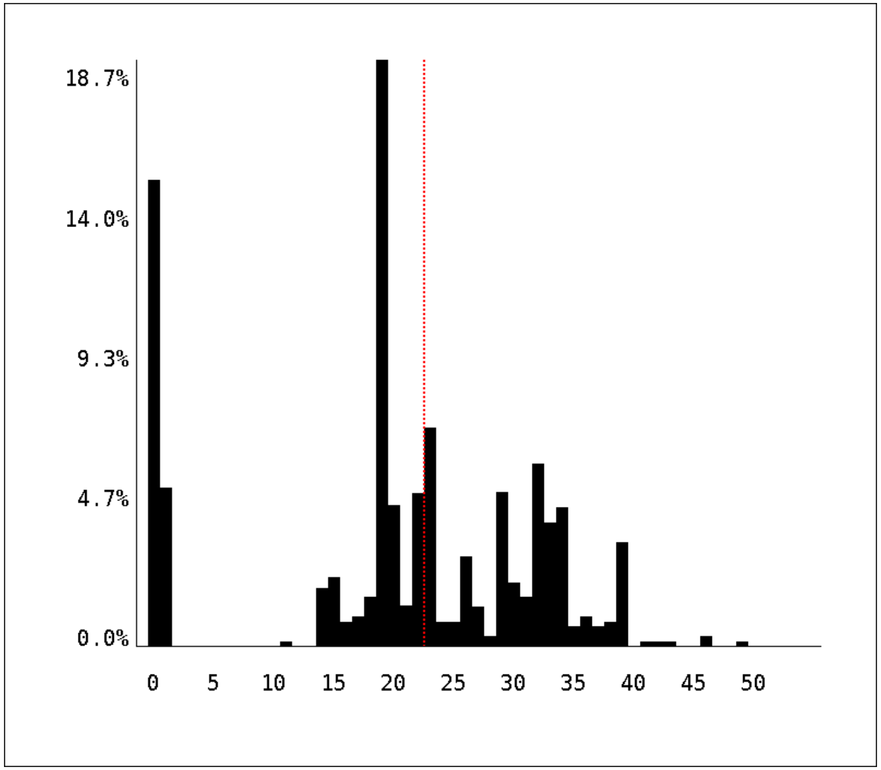
>0.05% - **23:60**

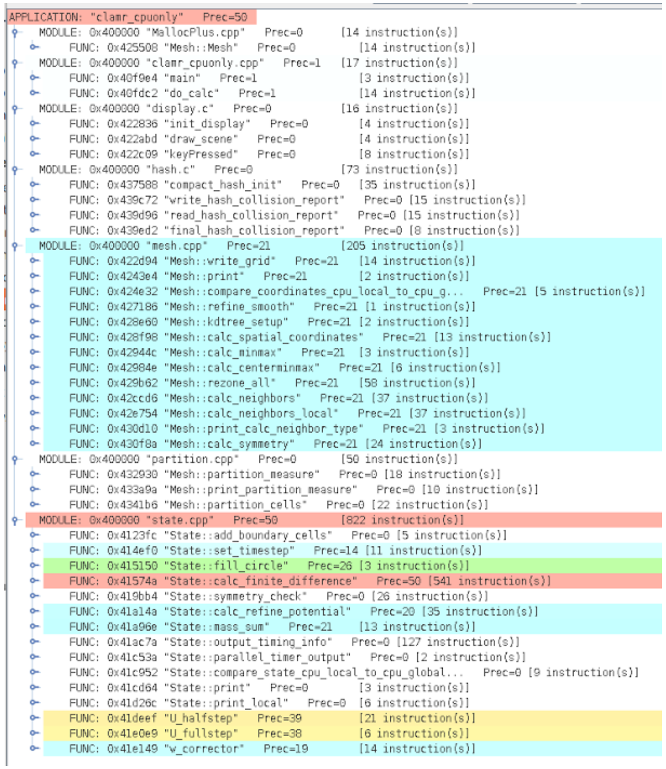


Full – **28:71**

CLAMR

- Cell-based Adaptive Mesh Refinement
  - Los Alamos National Lab





CLAMR

- Original program
  - Mass preserved to twelve digits
  - Average CPU time: 89.7s
- After CRAFT-assisted conversion
  - Mass preserved to seven digits
  - CPU time speedup (7.3%)
  - Memory usage reduced (~7%)
- After GPU mixed-precision rewrite
  - Up to 4X speed improvement in certain subroutines

Reduced Precision: Conclusions

- Automated analysis can identify fine-grained precision level requirements
- Reduced-precision analysis provides results more quickly than mixed-precision analysis

Future Work

- Short term projects + planned collaborations
  - Visualization/graphics studies (JMU)
  - Development cycle integration
    - Mixed-precision feedback in IDE
    - Correctness/accuracy integration testing
  - Machine learning techniques (UMD)
    - Can we predict low-sensitivity portions of code?
  - Energy-aware analysis (LLNL)
    - Min-maxing energy/performance with a hard bound on accuracy
  - Further case studies (LANL, JMU)

Future Work

- Long term
  - Direct GPU support and/or CUDA implementation
  - Full shadow value analysis
  - Compiler-based implementation
    - Probably LLVM (see Precimonious for similar work)
  - Automatic program transformations
  - Program modeling and verification
    - Guarantees for ALL inputs
    - Can concolic execution help here?
  - Probabilistic arithmetic

"Grand Vision"

- A suite of tools and analysis techniques
  - Understand floating-point behavior and precision-level profiles
  - Recommend or build mixed-precision variants
    - Preserve accuracy
    - Improve performance and reduce energy use
  - Encourage best practices in floating-point code

Contact Info

- Collaborators -

Jeff Hollingsworth (advisor) and Pete Stewart (UMD)

Bronis de Supinski, Matt Legendre, et al. (LLNL)

Bob Robey and Nathan DeBardeleben (LANL)

Email: **lam2mo (at) jmu.edu**

Personal website: **blog.freearrow.com/about**

CRAFT website: **sf.net/p/crafthpc**

Dyninst

Intel XED2

Lawrence Livermore National Laboratory