Numerical Optimization Targeting Sustainable Scientific Computing

A Case Study on LULESH and Reactor Simulator Benchmarks Using Verificarlo

Gülçin GEDİK

Supervisors : Roman IAKYMCHUK
Pablo DE OLIVEIRA CASTRO

Key Challenge

- Finding the balance between :
  - Accuracy
  - Time-to-solution
  - Energy-to-solution
- Our approach :
  - Mixed-Precision
    - Precision as needed
    - Better performance
    - Less energy consumption

Outline

- Floating Point Arithmetic
- Verificarlo
- Measuring Energy on European HPC Systems
- Methodology
- Reactor Simulator
- LULESH
- Conclusion and Future Work

Floating Point Arithmetic (FP)

- A FP number is represented on computers in IEEE-754 format:

$$x = mantissa \times 2^e$$

- The classical FP Error Model :

$$fl(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta), \quad |\delta| \le u, \quad \mathrm{op} \in \{+,-,\times,/,\sqrt{}\}$$

u: the unit round-off

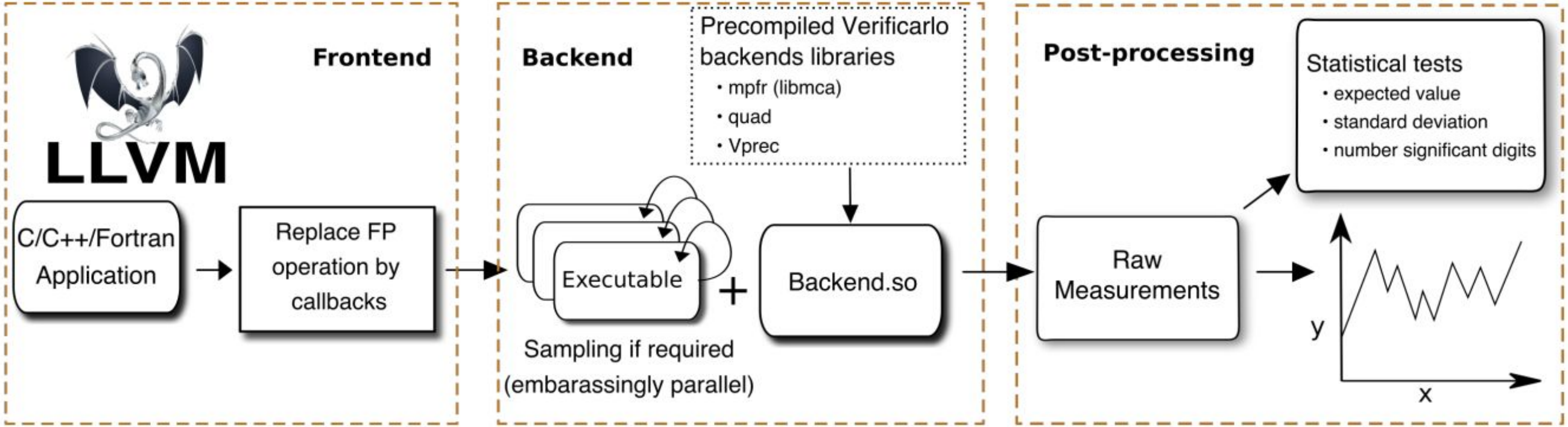
Relative Error :

$$\delta = \frac{\mathrm{fl}(a \,\mathrm{op}\, b) - (a \,\mathrm{op}\, b)}{(a \,\mathrm{op}\, b)}$$

(Source: Nicholas J. Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra)

Verificarlo - Part I

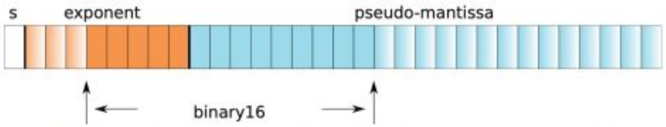
- FP Arithmetic Manipulation and Analysis
- Open-source, built upon LLVM Compiler
- Works at Intermediate Representation level
  - Captures compiler optimizations
- https://www.github.com/verificarlo/verificarlo



(Source: Automatic Exploration of Reduced Floating-Point Representations in Iterative Methods)

Verificarlo - Part II

- Manipulates FP operations with the help of different runtime library calls
  - No source code change
- Variable Precision Backend (VPREC)



VPREC simulates a binary16 inside a binary32 by truncating bits

- Monte Carlo Arithmetic (MCA) Backend
  - Random Rounding (RR) Mode : only introduce perturbation on the output of the FP operation
  - Full MCA Mode : introduce perturbations on the operands and output

Measuring Energy on European HPC Systems

Centre of Excellence in Exascale CFD

Best Practice Guide

Harvesting energy consumption on European HPC systems: Sharing Experience from the CEEC project

Best Practice Guide

- Challenges
  - More complex than measuring time-to-solution.
  - *No universal tool* exists.
  - Measurements require *elevated privileges*.

- Objectives
  - Facilitate energy measurements on the European HPC systems
  - Teach the community how to conduct such measurements
  - Provide transparent and easy-to-use guides

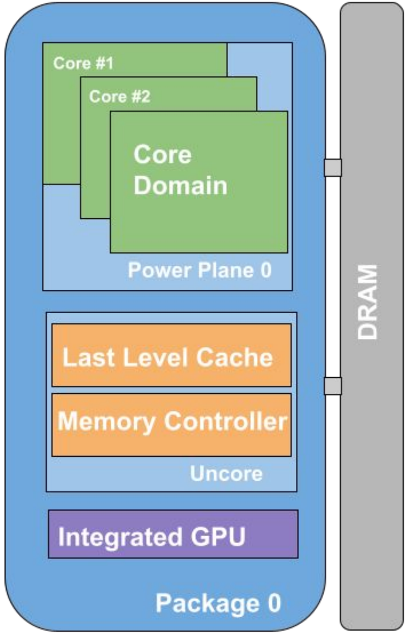
Measuring Energy on EuroHPC

Running Average Power Limit (RAPL) :

- ➢ Hardware energy measurement counters exposed with Model Specific Registers
- ➢ Provides measurements and capping control over different *power domains*
- ➢ Cumulated counters with a wrap-around time
- ➢ Requires *elevated privileges* to read

In our research :

- ➢ Used a custom tool that reads MSRs with the help of pref_event_open()
- ➢ Measured energy consumption on *cores domain* every 5 ms and computed the total
- ➢ Took care of wrap-around



RAPL Power Domains

Methodology

Step 1

Check Performance Hotspots

With the help of performance profiling tools, check the bottlenecks.
With Roofline Model check speed-up.

Step 2

Check Numerical Hotspots

With the help of numerical quality assessment tools, such as Verificarlo, check the functions that contribute the most to the final error at reduced precision and verify their stability.

Step 3

Select Candidate Code Section

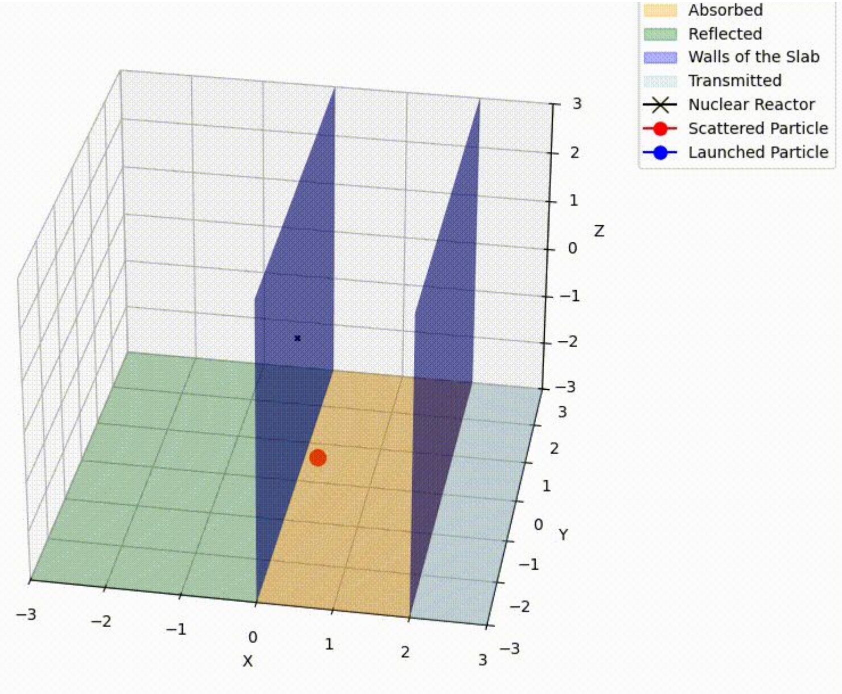
With the help of the previous steps, prune the sections with low return of investment value, and the ones providing inconsistent results with reduced precision.

Step 4

Import to Mixed Precision

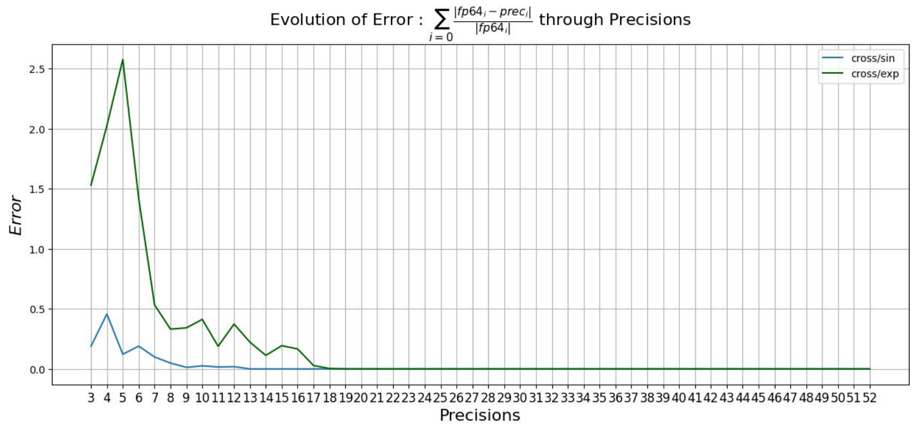
Import the selected code sections to mixed precision. Check the accuracy of the new implementations.

Reactor Simulator

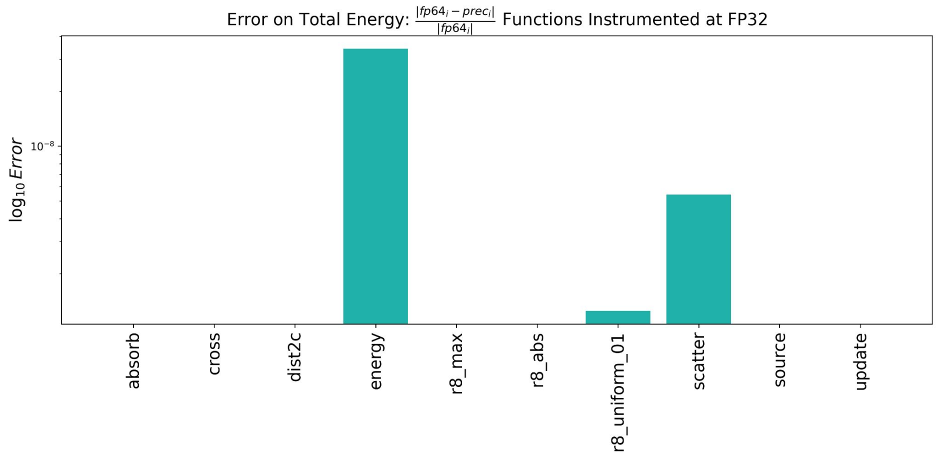



```
for each particle:
    Generate a new particle
    while (1){
        - Compute the distance that the particle
          can travel through, based on its energy.
        - Update the particle's position.
        if reflected by the shield :
            + save the energy of the particle
              er = er + e;
            break;
        if thransmitted :
            + save the energy of the particle
              et = et + e;
            break;
        if collided and absorbed :
            + save the energy of the particle
            ea = ea + e;
            break;
        if collided and scattered :
            + find scattering angle and energy
            continue;
    }
    etot = ea + er + et;
```

Code Inspection



- ➢ Performance hotspots *are not* numerical hotspots.
- ➢ Only one significant variable, *Total Energy*.



- ➢ Performance hotspots are coming from *shared mathematical library functions*, sinus and exponent and integer operations.
- ➢ They can be implemented *without accuracy loss* in single precision.

Main Takeaways

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Functions	sin(), exp() cos()	cross(), update()	except energy(), scatter() r8_uniform_01()	except energy()	all in FP32
Error[3]	0	0	0	10^{-8}	10^{-7}
Time Savings	8.4%	13.4%	16.1%	12.6%	14%
Energy Savings	8,7%	12.8%	14.9%	11.6%	13.7%

Table 1.5: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Local Machine

Strategy :

➢ Gradually include single precision at each stage

Results :

➢ No compromise on accuracy,
➢ 16% *performance gain*,
➢ 15% *less energy* consumption

Main Takeaways

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Functions	sin(), exp() cos()	cross(), update()	except energy(), scatter() r8_uniform_01()	except energy()	all in FP32
Error[3]	0	0	0	10^{-8}	10^{-7}
Time Savings	8.4%	13.4%	16.1%	12.6%	14%
Energy Savings	8,7%	12.8%	14.9%	11.6%	13.7%

Table 1.5: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Local Machine

Strategy :

- Gradually include single precision at each stage

Results :

- No compromise on accuracy,
- 16% *performance gain*,
- 15% *less energy* consumption

Main Takeaways

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Functions	sin(), exp() cos()	cross(), update()	except energy(), scatter() r8_uniform_01()	except energy()	all in FP32
Error[3]	0	0	0	10^{-8}	10^{-7}
Time Savings	8.4%	13.4%	16.1%	12.6%	14%
Energy Savings	8,7%	12.8%	14.9%	11.6%	13.7%

Table 1.5: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Local Machine

Strategy :

- Gradually include single precision at each stage

Results :

- No compromise on accuracy,
- 16% *performance gain*,
- 15% *less energy* consumption

Main Takeaways

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Functions	sin(), exp() cos()	cross(), update()	except energy(), scatter() r8_uniform_01()	except energy()	all in FP32
Error[3]	0	0	0	10^{-8}	10^{-7}
Time Savings	8.4%	13.4%	16.1%	12.6%	14%
Energy Savings	8,7%	12.8%	14.9%	11.6%	13.7%

Table 1.5: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Local Machine

Strategy :

- ➢ Gradually include single precision at each stage

Results :

- ➢ No compromise on accuracy,
- ➢ 16% *performance gain*,
- ➢ 15% *less energy* consumption

Main Takeaways

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Functions	sin(), exp() cos()	cross(), update()	except energy(), scatter() r8_uniform_01()	except energy()	all in FP32
Error[3]	0	0	0	10^{-8}	10^{-7}
Time Savings	8.4%	13.4%	16.1%	12.6%	14%
Energy Savings	8,7%	12.8%	14.9%	11.6%	13.7%

Table 1.5: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Local Machine

Strategy :

➢ Gradually include single precision at each stage

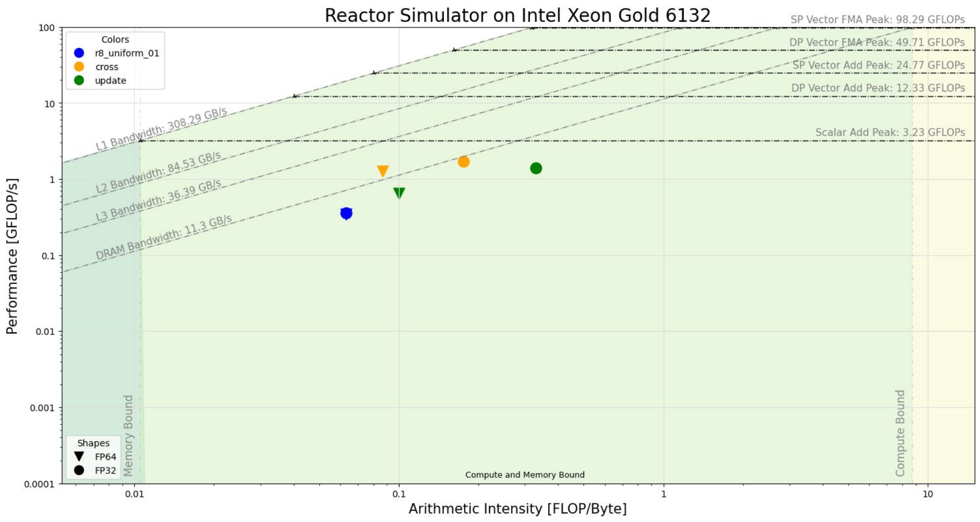
Results :

➢ No compromise on accuracy,

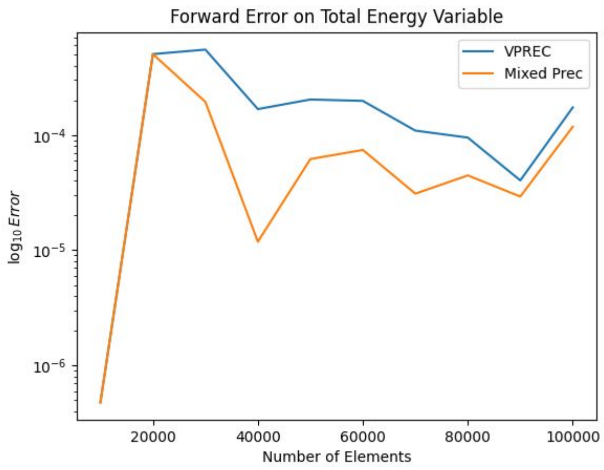
➢ 16% *performance gain*,

➢ 15% *less energy* consumption

Main Takeaways



Forward error comparison of whole application instrumentation at FP32 versus Stage 5.

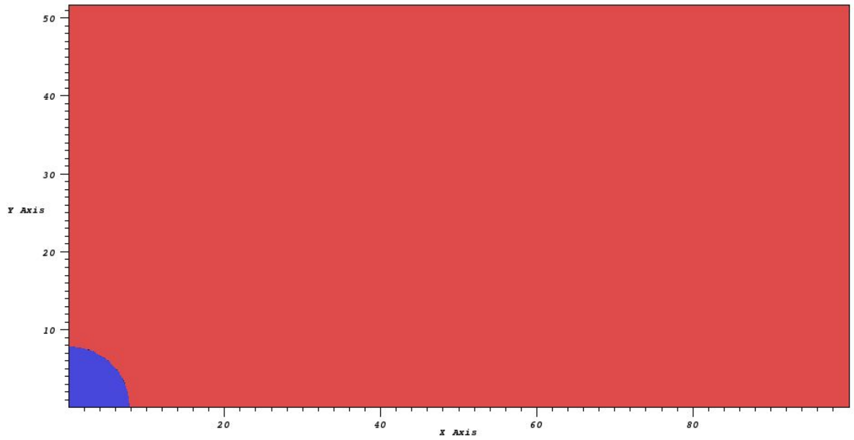


- ➢ Increased compute-bound performance

- ➢ Consistent error estimation with Verificarlo up to 50 thousands elements
- ➢ Differences to investigate

LULESH

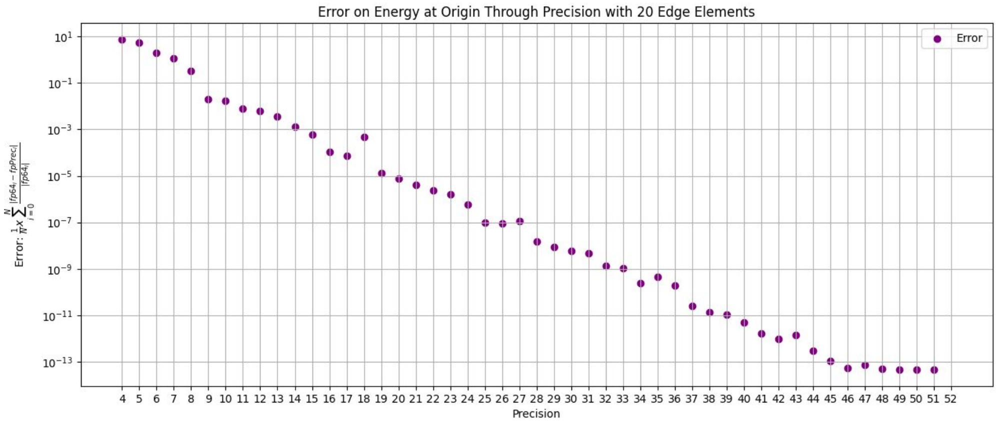
Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics



(Source: sd.llnl.gov, ALE3D Applications, Explicit Hydrodynamics)

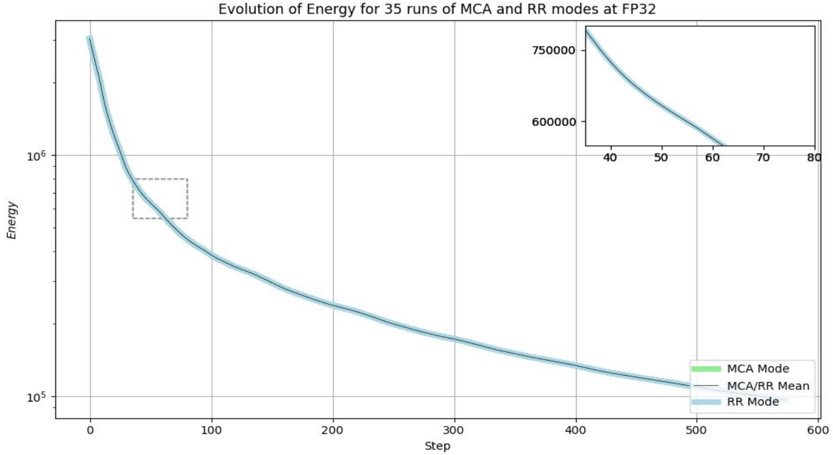
1. Create mesh domains and assign related variables for the mesh portion
2. Initialize problem state with previously allocated variables
3. Set up boundary conditions based on mesh symmetries
4. While simulation time is less than the stopping time :
   a. Calculate time increment using CFL condition
   b. Update the solution using the Lagrange Leapfrog method:
      i. Advance solution at the nodes
      ii. Advance solution at the elements

Code Inspection - Part I



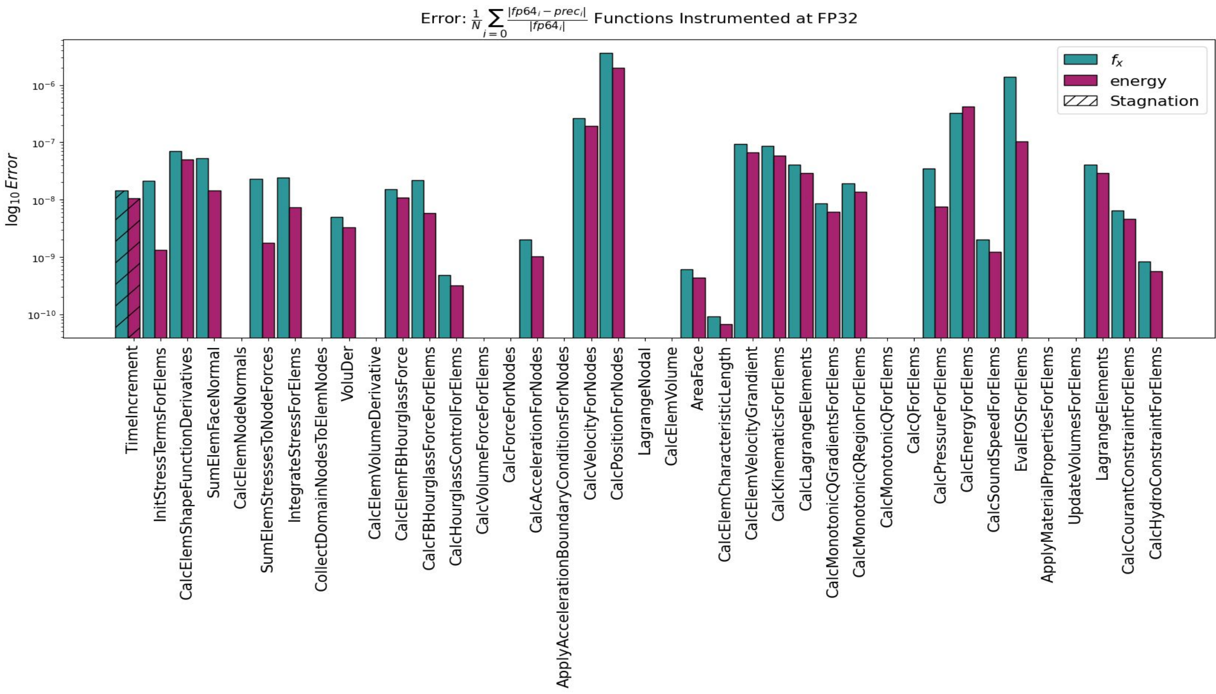
- ➢ Almost *linear* relationship between ***precision*** and ***accuracy***.
- ➢ Multiple significant variables, focused on ***energy*** variable.

- ➢ ***Negligible*** FP error accumulation with both MCA backend modes.
- ➢ Increase in the number of elements results in an increase in error.



Code Inspection - Part II

- 40 functions that has floating point operations.
- Half of the functions has *less than 1% self-time* each.
- 25% of functions *do not contribute* error in single precision.



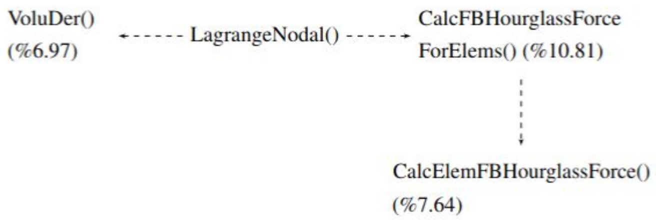


Figure 6.5: Self Times and Call Relations on LULESH

Inclusion Approach

Type	Stage 1	Stage 2	Stage 3
Functions	CalcElemFB-HourglassForce()	CalcFB-HourglassForceFE()	CalcHourglassControlFE() CollectDomainNtoElemN() VoluDer()
Error[3]	1.2×10^{-9}	6.9×10^{-9}	9.6×10^{-9}
Time Savings	-6.5%	8.5%	4.5%
Energy Savings	-7.3%	10%	3.19%

Table 1.4: Achieved Savings With Mixed Precision Versions on LULESH on Laptop

Strategy:

- ➢ Gradually include single precision at each stage
- ➢ Each stage includes the previous one
- ➢ *Avoid type-castings* in the process

Results:

- ➢ Correlations between time-to-solution and energy-to-solution
- ➢ 20% *more performance* on Kebnekaise and 8.5% on Laptop
- ➢ 10% *less energy consumption* on Laptop

Exclusion Approach

Groups	Function Name
Base	Complete FP64 Application
Group 0	Complete FP32 Application
Group 1	Time Increment()
Group 2	CalcPositionFN()
Group 3	CalcEnergyFE() CalcPressureFE()
Group 4	EvalEOSFE() CalcSoundSpreedFE()
Group 5	CalcKinematicFE() CollectDomainNtoElemN() CalcElemVolume() CalcElemShapeFuncDeriv() CalcVelocityGradient()
Group 6	CalcVelocityFN()

Stagnation with Verificarlo

Strategy :

- ➢ Maintain single precision in the rest of the application
- ➢ Gradually include double precision at most error prone locations

Results :

- ➢ Defined upper limit for performance gain
- ➢ Defined lower limit for accuracy
- ➢ To investigate the source of the differences between the implementation and Verificarlo instrumentation

Conclusion

- Provided a Best Practice Guide to facilitate energy measurements on EuroHPC systems.
- Followed a well established methodology.
- Explored mixed-precision opportunities in scientific codes.
- Improved time-to-solution and energy-to solution.
  - With Reactor Simulator, achieved 15% gain in both.
  - With LULESH, achieved 10% in both.
- Investigated possible implementation strategies.

Thank you for listening !

I sincerely thank the jury for their time and efforts in understanding my approach. I am happy to answer your questions.

Acknowledgements

Special thanks to my supervisors for their guidance and feedback, and to my friends Yanxiang Chen and Sude for their valuable contributions.

Backups

Reactor Simulator

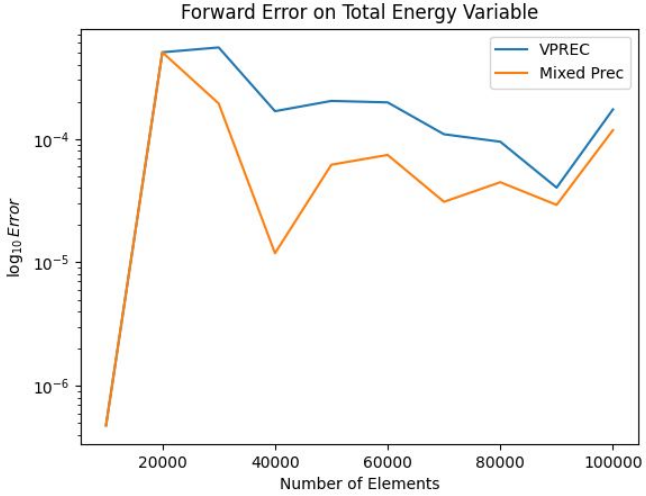
Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Double Median	2.2	2.2	2.2	2.19	2.23
Mixed Median	2.09	1.95	1.83	1.88	1.85
Savings	5%	11.2%	16.8%	14.1%	17.7%
Std. Dev.	1.1%	1.1%	1.2%	1.5%	2.6%
Error[3]	0	0	0	10^{-8}	10^{-7}

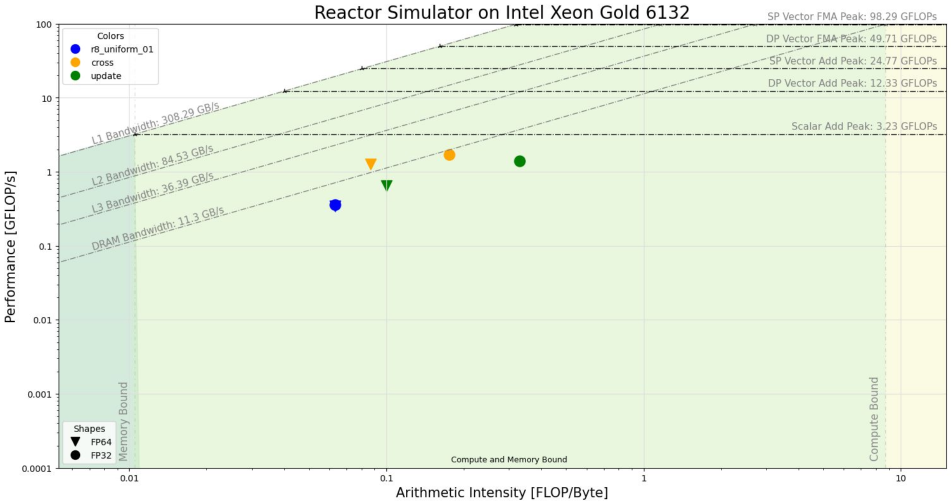
Table 5.4: Achieved Savings With Mixed Precision Versions on Reactor Simulator on Laptop

Type	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Double
Cores Median	7372.93	7034.02	6869.165	7138.684	6966.302	8075.75
Savings	8,7%	12.8%	14.9%	11.6%	13.7%	-

Table 5.5: Achieved Energy (in Joules) Savings With Mixed-Precision Versions on Reactor Simulator on Laptop

Reactor Simulator





Reactor Simulator

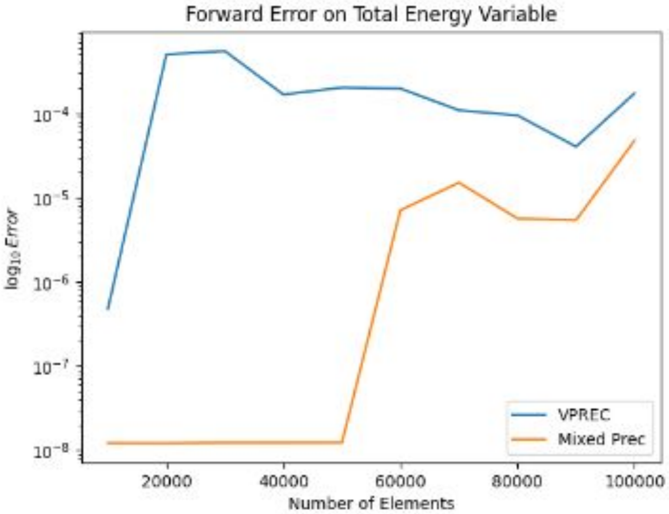


Figure 5.6: Forward Error Comparison VPREC-FP32 versus Stage 4.

LULESH :

Time Savings on Kebnekaise

Type	Stage 1	Stage 2	Stage 3
Functions	CalcElemFB-HourglassForce()	CalcFB-HourglassForceFE()	CalcHourglassControlFE() CollectDomainNtoElemN() VoluDer()
Double Median	5.27	5.27	5.25
Mixed Median	5.51	4.21	4.6
Savings	-4.5%	20.2%	12.3%
Std. Dev.	2.8%	3.5%	1.2%
Error[1]	1.2×10^{-9}	6.9×10^{-9}	9.6×10^{-9}

Table 6.2: Achieved Savings With Mixed-Precision Versions on LULESH on Kebnekaise with Inclusion Approach

LULESH :

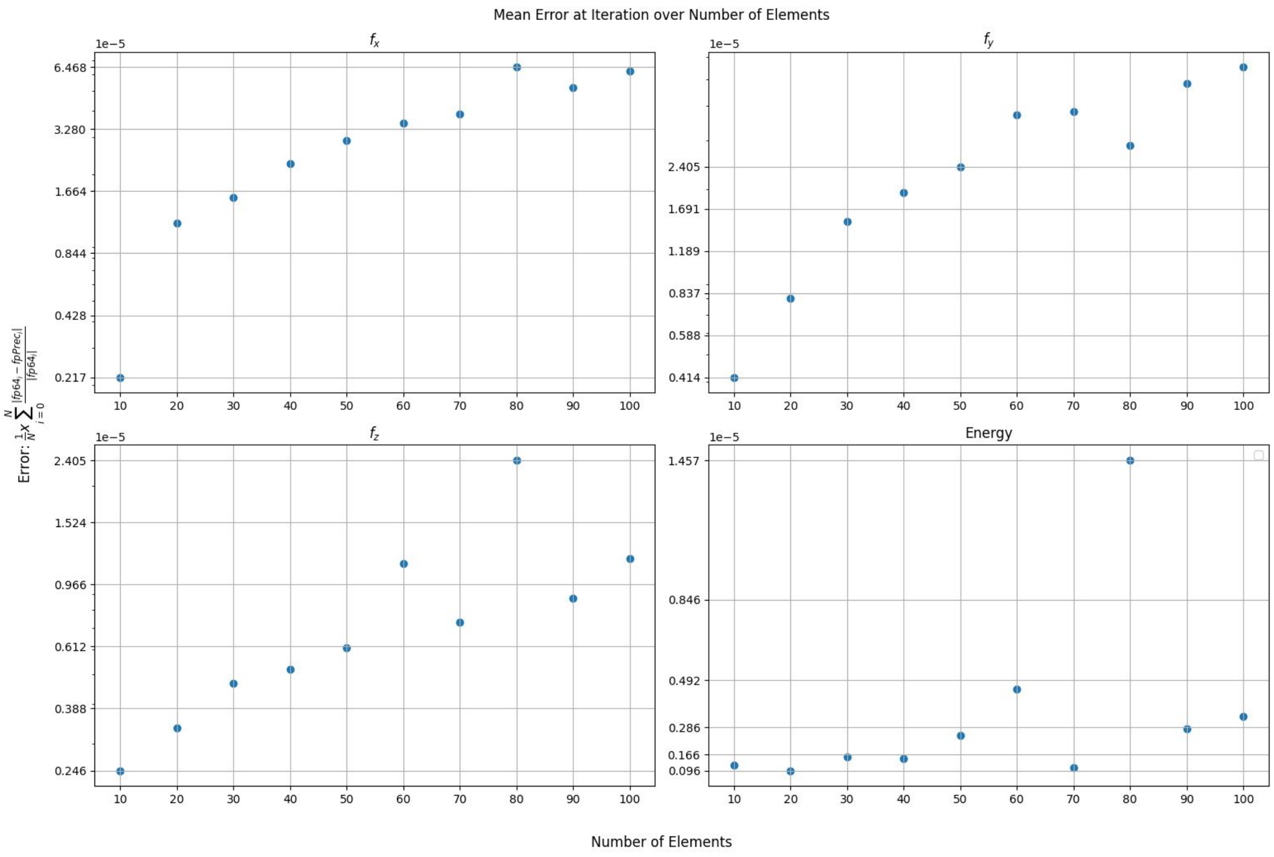
Time Savings on Laptop

Type	Stage 1	Stage 2	Stage 3
Double Median	2.91	2.92	2.8
Mixed Median	3.11	2.67	2.7
Savings	-6.5%	8.5%	4.5%
Std. Dev.	3.8%	2.9%	3.1%

Table A.1: Achieved Savings With Mixed Precision Versions on LULESH on Laptop with Inclusion Approach

LULESH :

Different Number of Elements



LULESH :

Exclusion Approach Details

Groups	Function Name	Single Error	SelfTime(%)	Time Gain
Base	Complete FP64 Application	0	-	-
Group 0	Complete FP32 Application	1.95×10^{-6}	-	18.8 %
Group 1	Time Increment()	10^{-8}	$< 0.1\%$	
Cumulative Error		1.95×10^{-6}		14.4 %
Group 2	CalcPositionFN()	1.77×10^{-6}	0.48%	
Improvement		1.95×10^{-6}		13.7%
Group 3	CalcEnergyFE()	3×10^{-7}	5.08%	
	CalcPressureFE()	3.5×10^{-8}	2.91%	
Improvement		1.89×10^{-6}		11%
Group 4	EvalEOSFE()	1.2×10^{-6}	2.47%	
	CalcSoundSpreedFE()	10^{-9}	$< 0.1\%$	
Improvement		2.08×10^{-6}		10.3%
Group 5	CalcKinematicFE()	7.8×10^{-8}	3.47%	
	CollectDomainNtoElemN()	0	4.51%	
	CalcElemVolume()	0	1.79%	
	CalcElemShapeFuncDeriv()	6.5×10^{-8}	3.47%	
	CalcVelocityGradient()	8×10^{-8}	1.71%	
Improvement		1.33×10^{-6}		7.9%
Group 6	CalcVelocityFN()	1.8×10^{-7}	0.81%	
Improvement		1.8×10^{-6}		6.8%

Table 6.4: Detailed Function Statistics and Cumulative Implementation Steps of Mixed-Precision Application with Achieved Error Improvements and Time Gain on Kebnekaise

LULESH :

Complete FP32 implementation