Exploring compiler behavior on an industrial application (OpenRadioss) on modern arm processors

Hugo Bolloré, Emmanuel Oseret, Kévin Camus, Cédric Valensi, William Jalby, Université de Versailles Saint-Quentin-en-Yvelines

ACKNOWLEDGEMENTS: ALTAIR, AWS and CALMIP

HORIZON-EUROHPC-JU-2023-COE

1 January 2024– 31 December 2026

Grant Agreement No 101143931

Objectives

The main goal is to analyze and assess quantitatively, compiler and compiler options impact on performance.

- Identify strengths/weakness of various compilers options and compilers
- Leverage optimizations between compilers
- Study ISA impact
- Develop corresponding methodology and test on a real industrial strength application

Methodology

ISSUE: Compiler A can outperform compiler B for loop 47 and the situation can be reversed for loop 52 ➡ Analysis/assessment has to be automated because we need to go down at least to the function level and very often down to the loop level.

Timing will be performed at three levels: whole app, function level, loop level

Goals require detailed analysis of compiler outputs: assembly code

Relying on MAQAO/ONE View capabilities

- To evaluate capabilities, weakness and strengths of various ASM codes using simplified simulators
- To perform matching between source code and different ASM variants: essential requisite to analyze advanced compiler strategies multiversioning

Analyzing Code Quality (1)

Focus on loops: innermost/in between/outermost

Evaluate ASM using CQA (Code Quality Analysis) included in MAQAO.

- Generic topics of interest
  - Port / Functional Units usage
  - Vectorization
  - Instruction set use
  - Vectorization Roadblocks
  - Data access
- Two types of analysis: **Static** at the ASM level and **Dynamic** requiring measurement

WARNING: By looking directly at ASM, both compiler mistakes but also source code issues will be taken into account.

Analyzing Code Quality (2)

Classify performance issues into 5 main categories

1. **Loop computation:** issues related to the computation organization (FMA, SQRT/DIV, etc…)
2. **Control Flow:** issues relevant to control (branches, call, ….)
3. **Data access:** issues essentially related to memory operations (stride, indirect, spill/fill, ….)
4. **Vectorization roadblocks:** issues preventing vectorization (complex control flow, …)
5. **Inefficient vectorization:** issues related to vectorization quality (vector length, masked, …)

Loop ID	Analysis	Penalty Score
▼ Loop 29963 - engine_linuxa64_gf_ompi	Execution Time: 1 % - Vectorization Ratio: 21.05 % - Vector Length Use: 59.87 %	
▼ Loop Computation Issues		6
○	[SA] Less than 10% of the FP ADD/SUB/MUL arithmetic operations are performed using FMA - Reorganize arithmetic expressions to exhibit potential for FMA. This issue costs 4 points.	4
○	[SA] Presence of a large number of scalar integer instructions - Simplify loop structure, perform loop splitting or perform unroll and jam. This issue costs 2 points.	2
▼ Control Flow Issues		2
○	[SA] Several paths (2 paths) - Simplify control structure or force the compiler to use masked instructions. There are 2 issues (= paths) costing 1 point each.	2
▼ Data Access Issues		12
○	[SA] Presence of constant non unit stride data access - Use array restructuring, perform loop interchange or use gather instructions to lower a bit the cost. There are 2 issues (= data accesses) costing 2 point each.	4
○	[SA] Presence of indirect accesses - Use array restructuring or gather instructions to lower the cost. There are 2 issues (= indirect data accesses) costing 4 point each.	8
► Vectorization Roadblocks		14

Hardware/Software platforms

Model Name	Frequency (GHz)	Number of cores/socket	Number of sockets	L1D (KB)	L1I (KB)	L2 (KB)	L3 (MB)	L3/Core (MB)
Neoverse N1 Ampere ALTRA Q80-30 (CALMIP)	3	80	1	64	64	1024	32	0,40
Neoverse V1 G3E (AWS)	2,6	64	1	64	64	1024	32	0,50
Neoverse V2 G4 (AWS)	2,8	96	1	64	64	2048	36	0,38

Neoverse N1: Arm C/C++/Fortran Compiler version 22.1 (build number 12) (based on LLVM 13.0.1)

Neoverse V1/V2: ACfL 24.10, GNU Fortran2008 13.2.0

Compiler options: O2, O3, O3 + no-sve, O3 + no-sve2

TALK FOCUS: AWS Graviton 4 Results

Target code: OpenRadioss

Altair® Radioss® & OpenRadioss™
The Industry Standard Open Platform for Crash & Impact

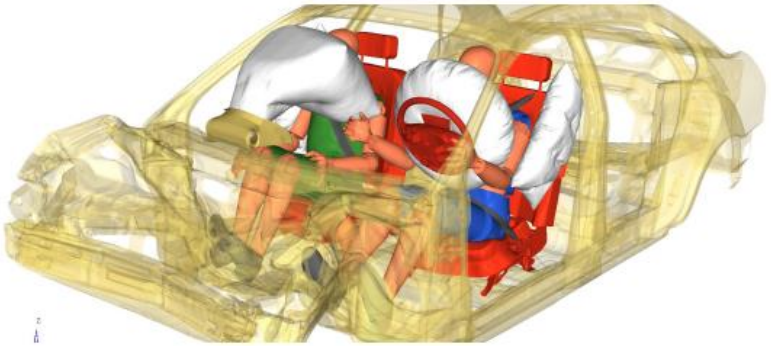
OpenRadioss™ Open-Source Version

- Source code publicly accessible from: https://github.com/OpenRadioss
- Upstream version, with contributions from a fast-growing worldwide community
- Precompiled Linux & Windows executables to run latest builds with no license check
- Support from the community, via forum

www.openradioss.org

Altair® Radioss® Commercial Version

- Commercial releases with extensive QA, professional support, documentation and maintenance priority
- Available under Altair Units license
- Encrypted models for dummies & barriers
- Channels valuable community contributions into industrial release



www.altair.com/radioss

ALTAIR

Dataset and code parallelism

Realistic data sets have been used. Due to domain decomposition, data set size varies depending upon the number of MPI ranks used

- TAURUS (Public): 10 millions elements, reference 1.7 GB, after partitioning for 96 MPI ranks 17 GB

Code in Fortran combining MPI + OpenMP parallelism

- Excellent MPI scaling due to very good domain decomposition
- More limited OpenMP parallelism

OUR TWO REFERENCE TESTS: 96 MPI Ranks and 48 MPI Ranks + 2 OpenMP threads per rank

Experimental constraints

NUMERICAL ACCURACY

The code uses explicit method with very small time steps.

Accuracy and reproducibility is major concern. Therefore any code transformation which has an impact on numerical accuracy is prohibited. For example, fast math compiler flag and even the option enabling fused multiply add (FMA) is prohibited

EXECUTION TIME

Due to the nature of the code (mechanical crash), various code segments are used throughout the whole simulation => to be realistic (involving all key code segments) the run has to be long enough…

TAURUS on 96 cores Neoverse V2 takes over 2 hours!

Global Characteristics Taurus GFortan –O3 on G4

Global Metrics	
Total Time (s)	7.51 E3
Max (Thread Active Time) (s)	7.42 E3
Average Active Time (s)	6.77 E3
Activity Ratio (%)	90.2
Average number of active threads	86.565
Affinity Stability (%)	100.0
Time in analyzed loops (%)	85.8
Time in analyzed innermost loops (%)	81.3
Time in user code (%)	87.7
Compilation Options Score (%)	74.9
Array Access Efficiency (%)	70.6

48 MPI Ranks + 2 OMP Threads/ranks

Link to MAQAO report

4 essential metrics:

1. **Average active time:** Sum over all threads of their active time divided by thread count. **IDEAL = total execution time**
2. **Activity ratio :** Sum over all threads of their active time divided by the sum of their wall time. **IDEAL: 100%**
3. **Average active number of threads :** Sum over all threads of their active time divided by longest wall clock time: **IDEAL = number of threads used**
4. **Affinity stability:** evaluates percentage of time spent without thread migration between physical cores: **IDEAL 100%**

Compiler flags: ACfl versus GFortran on AWS G4

Limited impact of Compiler Options

Global metric	*GNU*				*AcfL*			
	O2	*O3*	*O3 no SVE2*	*O3 no SVE*	*O2*	*O3*	*O3 no SVE2*	*O3 no SVE*
Total Time (s)	7.89 E3	7.51 E3	7.65 E3	7.41 E3	7.92 E3	7.91 E3	7.92 E3	8.26 E3
Max (Thread Active Time) (s)	7.71 E3	7.42 E3	7.47 E3	7.33 E3	7.89 E3	7.88 E3	7.88 E3	8.07 E3
Average Active Time (s)	7.02 E3	6.77 E3	6.78 E3	6.66 E3	7.77 E3	7.77 E3	7.77 E3	7.86 E3
Activity Ratio (%)	89.0	90.2	88.7	90.0	98.2	98.1	98.1	95.9
Average number of active threads	85.4	86.7	85.1	86.3	94.210	94.193	94.179	91.284
Affinity Stability (%)	99.2	100.0	99.1	100.0	100.0	100.0	100.0	98.5
Time in analyzed loops (%)	86.4	85.8	85.7	85.4	78.1	78.0	77.8	76.3
Time in analyzed innermost loops (%)	78.0	81.3	81.2	80.9	70.5	71.4	71.2	69.6
Time in user code (%)	88.1	87.7	87.6	87.4	80.9	80.8	80.6	79.2
Compilation Options Score (%)	62.5	74.9	74.9	74.9	99.5	99.5	99.5	99.5
Array Access Efficiency (%)	62.0	70.6	70.6	74.0	66.9	65.1	65.1	66.8

GFortran compiler flags: impact on AWS G4

Limited impact of Compiler Options

Global metric	GNU				AcfL			
	O2	O3	O3 no SVE2	O3 no SVE	O2	O3	O3 no SVE2	O3 no SVE
Total Time (s)	7.89 E3	7.51 E3	7.65 E3	7.41 E3	7.92 E3	7.91 E3	7.92 E3	8.26 E3
Max (Thread Active Time) (s)	7.71 E3	7.42 E3	7.47 E3	7.33 E3	7.89 E3	7.88 E3	7.88 E3	8.07 E3
Average Active Time (s)	7.02 E3	6.77 E3	6.78 E3	6.66 E3	7.77 E3	7.77 E3	7.77 E3	7.86 E3
Activity Ratio (%)	89.0	90.2	88.7	90.0	98.2	98.1	98.1	95.9
Average number of active threads	85.4	86.7	85.1	86.3	94.210	94.193	94.179	91.284
Affinity Stability (%)	99.2	100.0	99.1	100.0	100.0	100.0	100.0	98.5
Time in analyzed loops (%)	86.4	85.8	85.7	85.4	78.1	78.0	77.8	76.3
Time in analyzed innermost loops (%)	78.0	81.3	81.2	80.9	70.5	71.4	71.2	69.6
Time in user code (%)	88.1	87.7	87.6	87.4	80.9	80.8	80.6	79.2
Compilation Options Score (%)	62.5	74.9	74.9	74.9	99.5	99.5	99.5	99.5
Array Access Efficiency (%)	62.0	70.6	70.6	74.0	66.9	65.1	65.1	66.8

- **FOR GFortran: O2 is worse (6%), O3 no-sve is the best**

ACfl compiler flags: impact on AWS G4

Limited impact of Compiler Options ?

Global metric	GNU				AcfL			
	O2	O3	O3 no SVE2	O3 no SVE	O2	O3	O3 no SVE2	O3 no SVE
Total Time (s)	7.89 E3	7.51 E3	7.65 E3	7.41 E3	7.92 E3	7.91 E3	7.92 E3	8.26 E3
Max (Thread Active Time) (s)	7.71 E3	7.42 E3	7.47 E3	7.33 E3	7.89 E3	7.88 E3	7.88 E3	8.07 E3
Average Active Time (s)	7.02 E3	6.77 E3	6.78 E3	6.66 E3	7.77 E3	7.77 E3	7.77 E3	7.86 E3
Activity Ratio (%)	89.0	90.2	88.7	90.0	98.2	98.1	98.1	95.9
Average number of active threads	85.4	86.7	85.1	86.3	94.210	94.193	94.179	91.284
Affinity Stability (%)	99.2	100.0	99.1	100.0	100.0	100.0	100.0	98.5
Time in analyzed loops (%)	86.4	85.8	85.7	85.4	78.1	78.0	77.8	76.3
Time in analyzed innermost loops (%)	78.0	81.3	81.2	80.9	70.5	71.4	71.2	69.6
Time in user code (%)	88.1	87.7	87.6	87.4	80.9	80.8	80.6	79.2
Compilation Options Score (%)	62.5	74.9	74.9	74.9	99.5	99.5	99.5	99.5
Array Access Efficiency (%)	62.0	70.6	70.6	74.0	66.9	65.1	65.1	66.8

- For ACfl: O3 no-sve is the worst (5%), O2/O3/O3 no-sve are identical and the best

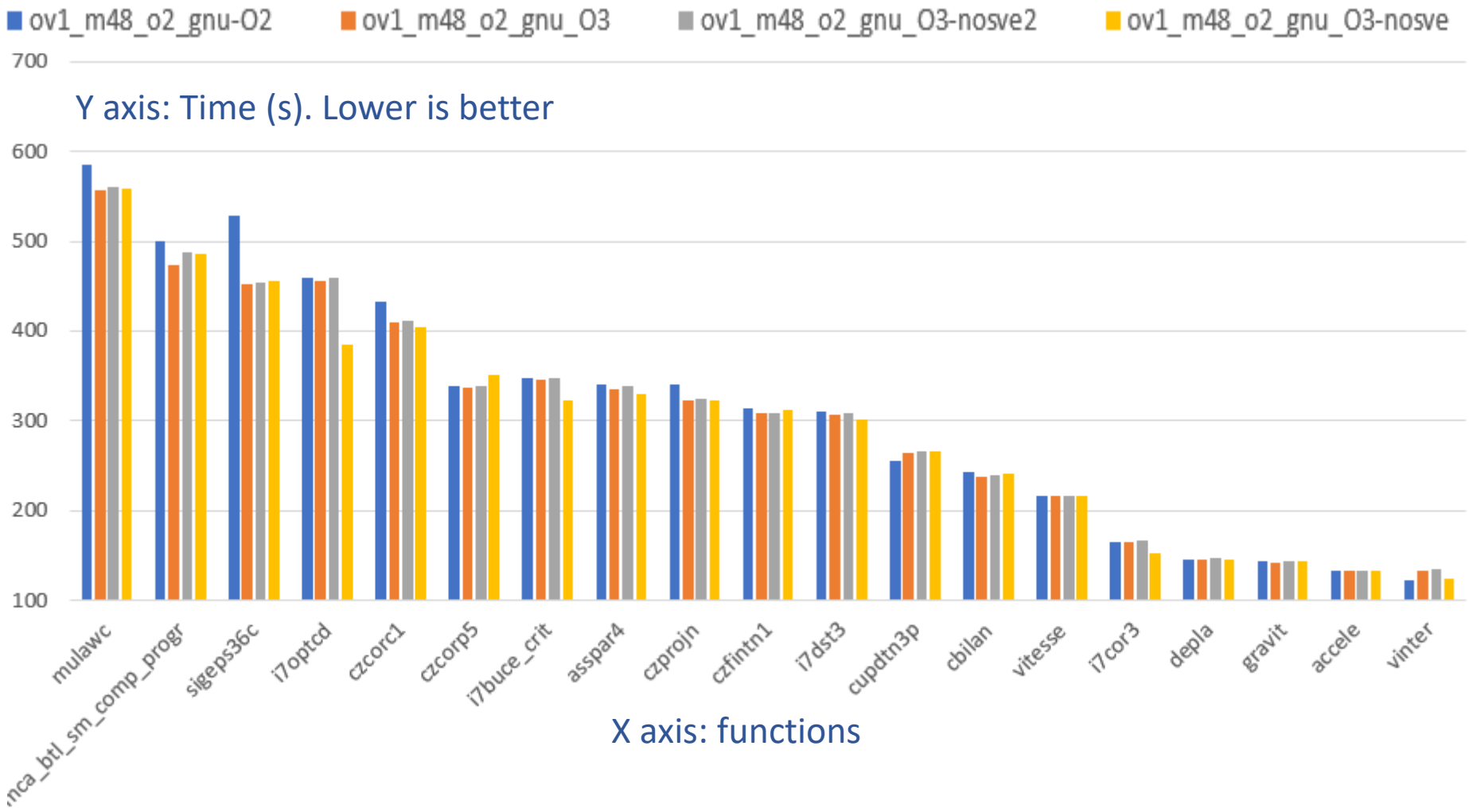
Compiler flags: ACfl versus GFortran on AWS G4

Limited impact of Compiler Options ?

Global metric	GNU				AcfL			
	O2	O3	O3 no SVE2	O3 no SVE	O2	O3	O3 no SVE2	O3 no SVE
Total Time (s)	7.89 E3	7.51 E3	7.65 E3	7.41 E3	7.92 E3	7.91 E3	7.92 E3	8.26 E3
Max (Thread Active Time) (s)	7.71 E3	7.42 E3	7.47 E3	7.33 E3	7.89 E3	7.88 E3	7.88 E3	8.07 E3
Average Active Time (s)	7.02 E3	6.77 E3	6.78 E3	6.66 E3	7.77 E3	7.77 E3	7.77 E3	7.86 E3
Activity Ratio (%)	89.0	90.2	88.7	90.0	98.2	98.1	98.1	95.9
Average number of active threads	85.4	86.7	85.1	86.3	94.210	94.193	94.179	91.284
Affinity Stability (%)	99.2	100.0	99.1	100.0	100.0	100.0	100.0	98.5
Time in analyzed loops (%)	86.4	85.8	85.7	85.4	78.1	78.0	77.8	76.3
Time in analyzed innermost loops (%)	78.0	81.3	81.2	80.9	70.5	71.4	71.2	69.6
Time in user code (%)	88.1	87.7	87.6	87.4	80.9	80.8	80.6	79.2
Compilation Options Score (%)	62.5	74.9	74.9	74.9	99.5	99.5	99.5	99.5
Array Access Efficiency (%)	62.0	70.6	70.6	74.0	66.9	65.1	65.1	66.8

- For GFortran: O2 is worse (6%), O3 no-sve is the best
- For ACfl: O3 no-sve is the worst (5%), O2/03/O3 no-sve are identical and the best
- ACfl is using active waiting counted as activity while GFortran is using passive waiting counted as inactivity
- GFortran is better than ACfl: 6%
- For both compilers and options: a large amount of time is spent in loops

Compiler flags impact: GFortran on AWS G4



Every compiler option is losing sometimes, winning at other times

GFortran on AWS G4: Some detailed analyses

i7optcd

Difference comes from a loop conditionally setting arrays values to zero

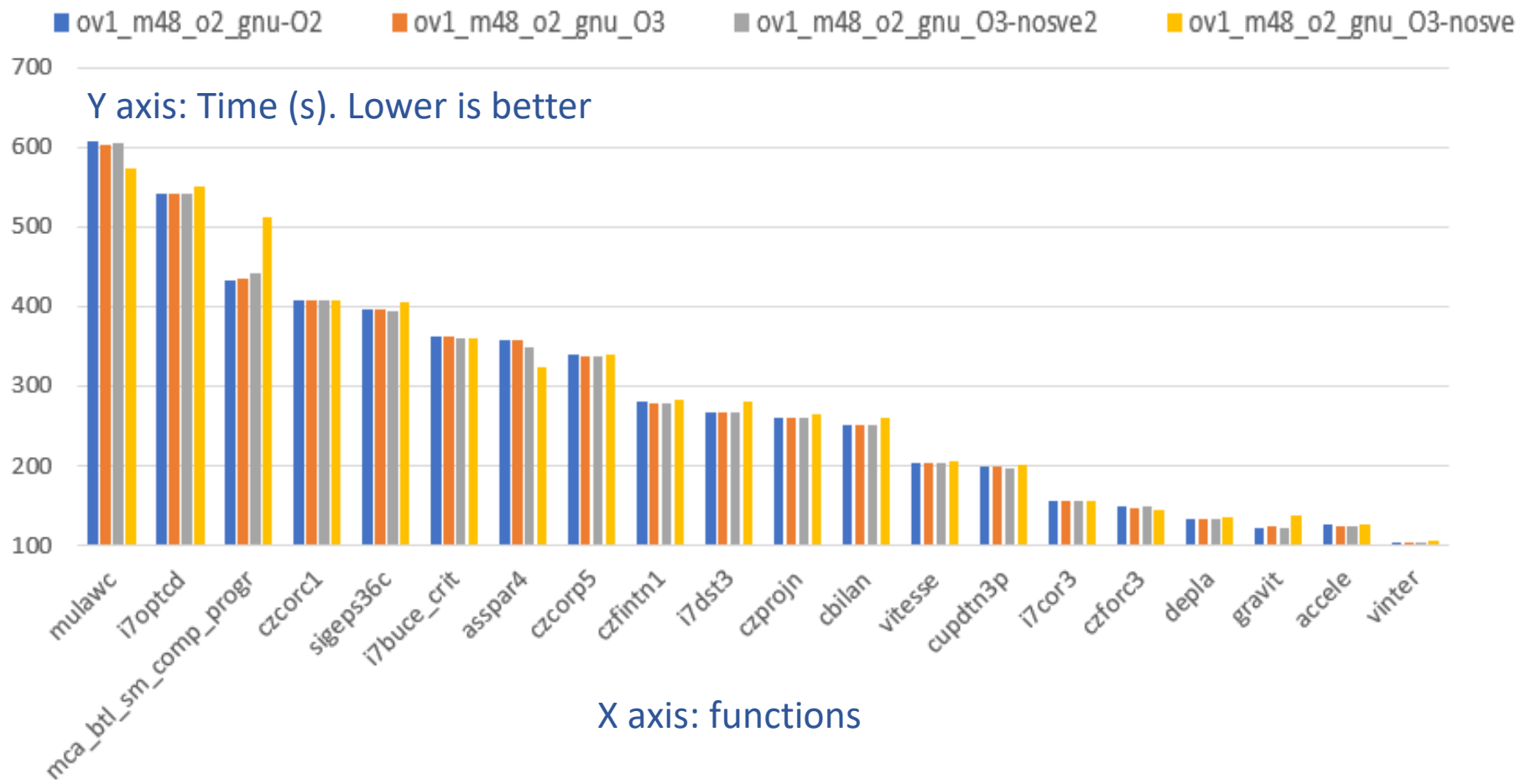
- no-sve option produces a scalar version taking ~25s
- Without excluding option, SVE version is taking ~90s

Multiple factors: scalar version can use dedicated instructions/registers when comparing/setting with zero whereas SVE instruction set lack such feature and requires to have every store instruction under predication

i7bucecrit

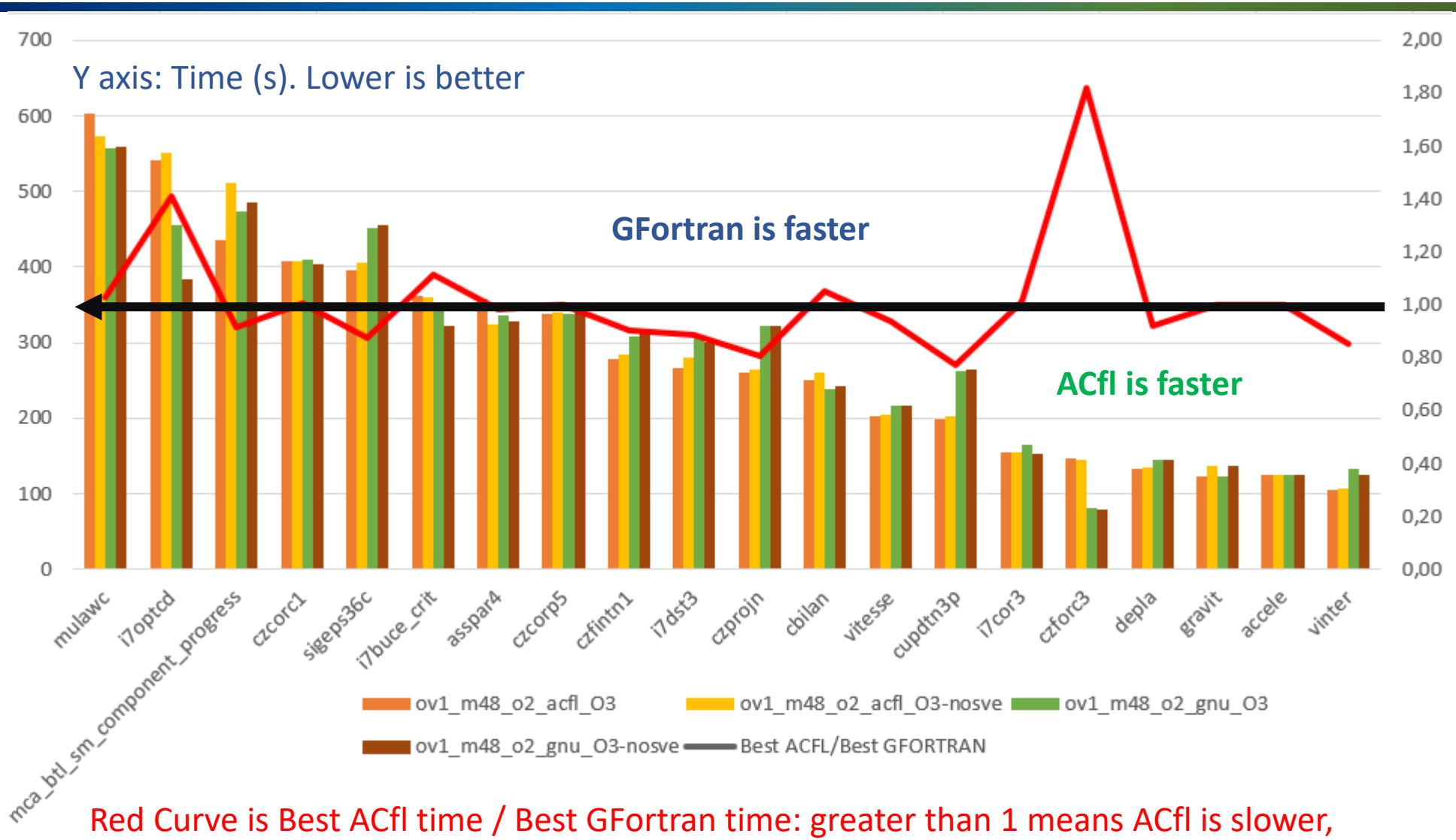
One of the loop is vectorized in NEON ONLY if SVE is disabled
Bug/Cost model ? Hard to extract or reproduce outside of the application

Compiler flags impact: ACfl on AWS G4



O3 no-sve is the compiler option with the largest impact (positive or negative !)

Best ACfl/best GFortran on AWS G4



Red Curve is Best ACfl time / Best GFortran time: greater than 1 means ACfl is slower, lower than 1 means GFortran is slower

Future work

- Analyze more compiler options: fastmath, funroll, etc….
- Automate backport from one compiler to the other: pragma insertion: if a compiler has been able to vectorize a loop, the information could be provided through pragmas to compilers which were unsuccessful in vectorizing the same loop.
- Interact with compiler developers to refine/improve cost models
- Interact with application developers to use this technology

Conclusions

- Non uniform behavior of compiler options across subroutines/loops: some options perform better with some subroutines
- Non uniform behavior between compilers across subroutines/loops: no silver bullet compiler….
- Non uniform means non negligible performance difference: these performance difference are worth exploring/exploiting

Above all of these are well known “generic facts”

Conclusions

- Non uniform behavior of compiler options across subroutines/loops: some options perform better with some subroutines
- Non uniform behavior between compilers across subroutines/loops: no silver bullet compiler….
- Non uniform means non negligible performance difference: these performance difference are worth exploring/exploiting

Above all of these are well known “generic facts” but MAQAO allows to bring in:

- Quantitative estimation : essential for driving optimization
- Explanation of performance differences: this opens the door to backport optimization between compilers

Performance Optimisation and Productivity 3

A Centre of Excellence in HPC

Contact:

https://www.pop-coe.eu

pop@bsc.es

@POP_HPC

youtube.com/POPHPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.