EVIDEN

Advancing the ARM Ecosystem for European Scientific Flagship Codes

Arm HPC User Group Workshop @ ISC25

Erwan Raffin, CEPP Eviden
UVSQ LI-PaRAD lab
SiPearl

June 13rd, 2025 @ 9:00am - 1:00pm
Hall X5 - 1st floor, Congress Center Hamburg (CCH), Germany

EMOPASS project

Exascale Microprocessor and Tools for Strategic and Scientific Application Performance

- Objectives: **Advancing the ARM Ecosystem for European Scientific Flagship Codes**

- Consortium:

EMOPASS impact

Exascale Microprocessor and Tools for Strategic and Scientific Application Performance

- Context: **European Scientific Flagship Codes from EU Center of Excellence Galaxy**

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

ChEESE

T-REX
Targeting Real chemical accuracy at the EXascale

SPACE

MAX

EMOPASS application landscape

Exascale Microprocessor and Tools for Strategic and Scientific Application Performance

- Each EU CoE represents a scientific community with its own flagship codes…

EVIDEN

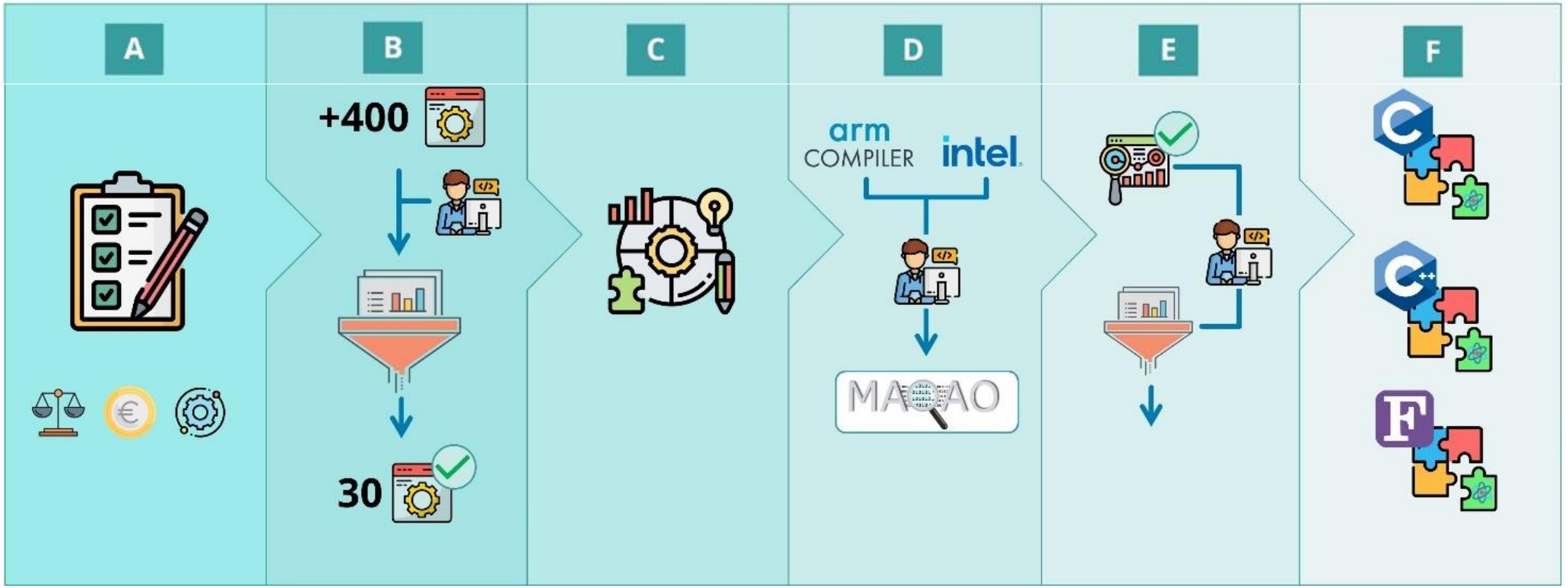
EMOPASS application landscape

Exascale Microprocessor and Tools for Strategic and Scientific Application Performance

- … and pathfinders

EMOPASS methodology

Exascale Microprocessor and Tools for Strategic and Scientific Application Performance



MAQAO

In brief

- **MAQAO (Modular Assembly Quality Analyzer and Optimizer)** is a performance analysis and optimization framework
  - Operates at binary level
  - Focus on core/node performance
  - Guides application developers along the optimization process through synthetic reports and hints

Quantum Espresso

Overview

- Quantum ESPRESSO
  - major open-source (set of) code(s) for quantum materials modelling using the plane-wave pseudopotential method.

- Mini-app: FFTXlib_wave

- Work done by Marc Sergent (Eviden CEPP)

Quantum Espresso

MAQAO profiling ACFL – NEON vs SVE on Grace

Loop id	Source Location	Source Function	Exclusive coverage run_0 (%)	Vectorization Ratio (%)	Vector Length Use (%)
236	miniapp2 - fft_types.f90:1249-1262	fft_types_grid_set	21.47	97.14	47.5
481	miniapp2 - fft_scalar.FFTW3.f90:135-135	fft_scalar_fftw3_cft_1z	5.13	100	75
517	miniapp2 - fft_scatter.f90:198-200	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L186_2_	3.78	66.67	41.67
515	miniapp2 - fft_scatter.f90:214-215	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L186_2_	3.43	0	25
571	miniapp2 - fft_scatter.f90:762-764	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L742_12_	2.87	66.67	41.67
519	miniapp2 - fft_scatter.f90:245-247	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L234_3_	2.67	66.67	41.67
123	miniapp2 - stick_base.f90:676-694	stick_base_hpsort	2.52	0	19.64
574	miniapp2 - fft_scatter.f90:787-789	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L775_13_	2.00	66.67	41.67
287	miniapp2 - fft_helper_subroutines.f90:1035-1036	fft_helper_subroutines_fftx_psi2c_k	1.05	100	50
572	miniapp2 - fft_scatter.f90:747-748	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L742_12_	1.01	0	12.5

ACFL NEON

Loop id	Source Location	Source Function	Exclusive coverage run_0 (%)	Vectorization Ratio (%)	Vector Length Use (%)
250	miniapp2 - fft_types.f90:1249-1262	fft_types_grid_set	21.55	97.14	47.5
497	miniapp2 - fft_scalar.FFTW3.f90:135-135	fft_scalar_fftw3_cft_1z	5.01	92.31	94.23
534	miniapp2 - fft_scatter.f90:198-200	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L186_2_	3.96	66.67	41.67
532	miniapp2 - fft_scatter.f90:214-215	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L186_2_	3.17	0	25
585	miniapp2 - fft_scatter.f90:762-764	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L742_12_	2.99	66.67	41.67
536	miniapp2 - fft_scatter.f90:245-247	__nv_fft_scatter_fft_scatter_xy_impl_xy_F1L234_3_	2.58	66.67	41.67
136	miniapp2 - stick_base.f90:676-694	stick_base_hpsort	2.54	0	19.64
588	miniapp2 - fft_scatter.f90:787-789	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L775_13_	2.18	66.67	41.67
303	miniapp2 - fft_helper_subroutines.f90:1035-1036	fft_helper_subroutines_fftx_psi2c_k	1.05	100	50
586	miniapp2 - fft_scatter.f90:747-748	__nv_fft_scatter_fft_scatter_yz_impl_yz_F1L742_12_	1.01	0	12.5

ACFL SVE

Quantum Espresso

MAQAO profiling ACFL – NEON vs SVE on Grace

Assembly code of grid_set routine in fft_types.f90: 1249-1262

NEON — Assembly Code

```
0x19ab8 SSHLL   V11.2D, V10.2S, #32
0x19abc SSHLL2  V12.2D, V10.4S, #32
0x19ac0 SUBS    W22, W22, #4
0x19ac4 ORR     V13.16B, V27.16B, V27.16B
0x19ac8 ORR     V14.16B, V27.16B, V27.16B
0x19acc ORR     V15.16B, V0.16B, V0.16B
0x19ad0 ORR     V29.16B, V0.16B, V0.16B
0x19ad4 ORR     V31.16B, V1.16B, V1.16B
0x19ad8 ORR     V3.16B, V1.16B, V1.16B
0x19adc SCVTF   V11.2D, V11.2D
0x19ae0 SCVTF   V12.2D, V12.2D
0x19ae4 FMLA    V13.2D, V12.2D, V26.2D
0x19ae8 FMLA    V14.2D, V11.2D, V26.2D
0x19aec FMLA    V15.2D, V11.2D, V28.2D
0x19af0 FMLA    V29.2D, V12.2D, V28.2D
0x19af4 FMLA    V31.2D, V12.2D, V30.2D
0x19af8 FMLA    V3.2D, V11.2D, V30.2D
0x19afc FMUL    V14.2D, V14.2D, V14.2D
```

V registers => NEON ISA

SVE — Assembly Code

```
0x1a054 SSHLL   V11.2D, V10.2S, #32
0x1a058 SSHLL2  V12.2D, V10.4S, #32
0x1a05c SUBS    W22, W22, #4
0x1a060 ORR     V13.16B, V27.16B, V27.16B
0x1a064 ORR     V14.16B, V27.16B, V27.16B
0x1a068 ORR     V15.16B, V0.16B, V0.16B
0x1a06c ORR     V29.16B, V0.16B, V0.16B
0x1a070 ORR     V31.16B, V1.16B, V1.16B
0x1a074 ORR     V3.16B, V1.16B, V1.16B
0x1a078 SCVTF   V11.2D, V11.2D
0x1a07c SCVTF   V12.2D, V12.2D
0x1a080 FMLA    V13.2D, V12.2D, V26.2D
0x1a084 FMLA    V14.2D, V11.2D, V26.2D
0x1a088 FMLA    V15.2D, V11.2D, V28.2D
0x1a08c FMLA    V29.2D, V12.2D, V28.2D
0x1a090 FMLA    V31.2D, V12.2D, V30.2D
0x1a094 FMLA    V3.2D, V11.2D, V30.2D
0x1a098 FMUL    V14.2D, V14.2D, V14.2D
```

V registers => NEON ISA

Quantum Espresso

MAQAO profiling ACFL – NEON vs SVE on Grace

Assembly code of cft_1z routine in fft_scalar.FFTW3.f90: 135-135

NEON — Assembly Code

```
0x4bcfc LDP     Q1, Q2, [X12, #2016]
0x4bd00 SUBS    X13, X13, #4
0x4bd04 LDP     Q3, Q4, [X12]
0x4bd08 FMUL    V1.2D, V1.2D, V0.D[0]
0x4bd0c FMUL    V2.2D, V2.2D, V0.D[0]
0x4bd10 FMUL    V3.2D, V3.2D, V0.D[0]
0x4bd14 FMUL    V4.2D, V4.2D, V0.D[0]
0x4bd18 STP     Q1, Q2, [X12, #2016]
0x4bd1c STP     Q3, Q4, [X12], #64
0x4bd20 B.NE    4bcfc
```

V registers

=> NEON ISA

SVE — Assembly Code

```
0x4c398 LD1D    {Z3.D}, P0/Z, [X11, X14,LSL #3]
0x4c39c LD1D    {Z4.D}, P0/Z, [X11, X13,LSL #3]
0x4c3a0 SUBS    X16, X16, #4
0x4c3a4 LD1D    {Z5.D}, P0/Z, [X11, X15,LSL #3]
0x4c3a8 LD1D    {Z6.D}, P0/Z, [X11, MUL VL]
0x4c3ac FMUL    Z3.D, Z3.D, Z1.D
0x4c3b0 FMUL    Z4.D, Z4.D, Z2.D
0x4c3b4 FMUL    Z5.D, Z5.D, Z1.D
0x4c3b8 FMUL    Z6.D, Z6.D, Z2.D
0x4c3bc ST1D    {Z3.D}, P0, [X11, X14,LSL #3]
0x4c3c0 ST1D    {Z4.D}, P0, [X11, X13,LSL #3]
0x4c3c4 ST1D    {Z5.D}, P0, [X11, X15,LSL #3]
0x4c3c8 ST1D    {Z6.D}, P0, [X11, MUL VL]
0x4c3cc ADD     X11, X11, #64
0x4c3d0 B.NE    4c398
```

Z registers

=> SVE ISA

Quantum Espresso

Optimization of grid_set routine

- Compiler unable to vectorize the innermost loop
- Due to if(...) MAX construct
- Compiler does not know the reduction operator

```
1227        DO k = -nr3, nr3
1228          !
1229          ! ... me_image = processor number, starting from 0
1230          !
1231          IF( MOD( k + nr3, dfft%nproc ) == dfft%mype ) THEN
1232            DO j = -nr2, nr2
1233              DO i = -nr1, nr1
1234
1235                g( 1 ) = DBLE(i)*bg(1,1) + DBLE(j)*bg(1,2) + DBLE(k)*bg(1,3)
1236                g( 2 ) = DBLE(i)*bg(2,1) + DBLE(j)*bg(2,2) + DBLE(k)*bg(2,3)
1237                g( 3 ) = DBLE(i)*bg(3,1) + DBLE(j)*bg(3,2) + DBLE(k)*bg(3,3)
1238
1239 ! ...            calculate modulus
1240
1241                gsq =  g( 1 )**2 + g( 2 )**2 + g( 3 )**2
1242
1243                IF( gsq < gcut ) THEN
1244
1245 ! ...              calculate maximum index
1246                  nb(1) = MAX( nb(1), ABS( i ) )
1247                  nb(2) = MAX( nb(2), ABS( j ) )
1248                  nb(3) = MAX( nb(3), ABS( k ) )
1249                END IF
1250
1251              END DO
1252            END DO
1253          END IF
1254        END DO
```

Quantum Espresso

Optimization of grid_set routine

- Optimization applied:
  - Change computations of intermediate variables to associated loop nest level
  - Change IF(...) MAX construct with equivalent MAX(..., MERGE(...)) construct
  - Numerical correctness checked

```
1228        max_nr = MAX(nr1, MAX(nr2, nr3))
1229
1230        DO k = -nr3, nr3
1231          !
1232          ! ... me_image = processor number, starting from 0
1233          !
1234          kg(1) = DBLE(k)*bg(1,3)
1235          kg(2) = DBLE(k)*bg(2,3)
1236          kg(3) = DBLE(k)*bg(3,3)
1237          absk = ABS( k )
1238
1239          IF( MOD( k + nr3, dfft%nproc ) == dfft%mype ) THEN
1240            DO j = -nr2, nr2
1241
1242              jg(1) = kg(1) + DBLE(j)*bg(1,2)
1243              jg(2) = kg(2) + DBLE(j)*bg(2,2)
1244              jg(3) = kg(3) + DBLE(j)*bg(3,2)
1245              absj = ABS( j )
1246
1247              DO i = -nr1, nr1
1248
1249                g( 1 ) = jg(1) + DBLE(i)*bg(1,1)
1250                g( 2 ) = jg(2) + DBLE(i)*bg(2,1)
1251                g( 3 ) = jg(3) + DBLE(i)*bg(3,1)
1252
1253 ! ...            calculate modulus
1254
1255                gsq =  g ( 1 )*g( 1 ) + g( 2 )*g( 2 ) + g( 3 )*g( 3 )
1256
1257                gsq_gcut = gsq < gcut
1258
1259 ! ...            !   calculate maximum index
1260                nb(1) = MAX( nb(1), ABS( i ) + MERGE( 0, -max_nr, gsq_gcut))
1261                nb(2) = MAX( nb(2), absj + MERGE( 0, -max_nr, gsq_gcut))
1262                nb(3) = MAX( nb(3), absk + MERGE( 0, -max_nr, gsq_gcut))
1263              END DO
1264            END DO
1265          END IF
1266        END DO
```

Quantum Espresso

Optimization of grid_set routine

Ampere Altra (Neoverse N1)
1 MPI process 1 OpenMP thread

Global metrics between:

- non-optimized ACFL (r0)
- ACFL with with grid_set optimization (r1)

→ 10.6% performance gain

▶ **Compared Reports**

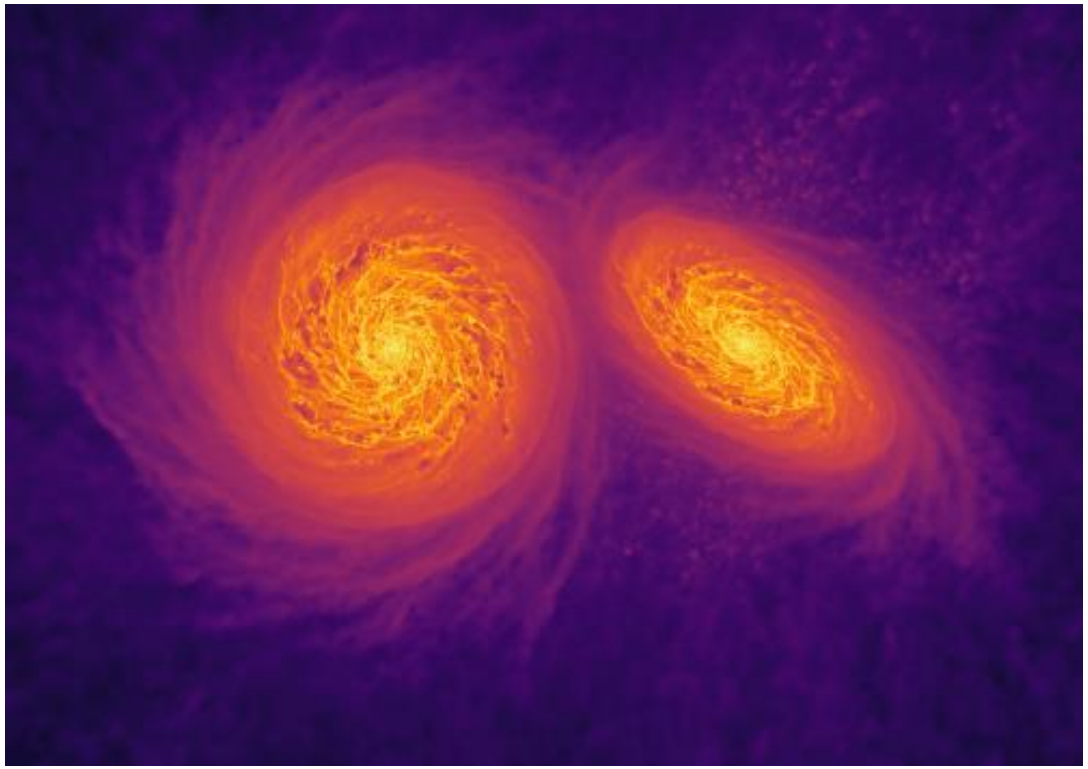
Global Metrics

Metric		r0	r1
Total Time (s)		56.54	50.74
Profiled Time (s)		54.80	49.00
Time in analyzed loops (%)		55.9	50.8
Time in analyzed innermost loops (%)		50.9	45.5
Time in user code (%)		55.8	50.9
Compilation Options Score (%)		100	100
Array Access Efficiency (%)		Not Available	Not Available
Potential Speedups			
Perfect Flow Complexity		1.00	1.00
Perfect OpenMP + MPI + Pthread		1.00	1.00
Perfect OpenMP + MPI + Pthread + Perfect Load Distribution		1.00	1.00
No Scalar Integer	Potential Speedup	1.07	1.08
	Nb Loops to get 80%	5	5
FP Vectorised	Potential Speedup	1.00	1.00
	Nb Loops to get 80%	1	1
Fully Vectorised	Potential Speedup	1.12	1.06
	Nb Loops to get 80%	5	8
Only FP Arithmetic	Potential Speedup	1.31	1.30
	Nb Loops to get 80%	7	8

ChaNGa

Overview

- ChaNGa (Charm N-body GrAvity solver)
- An adaptable N-body and gas dynamics code
- Work done by Matthieu Kuhn (Eviden CEPP)

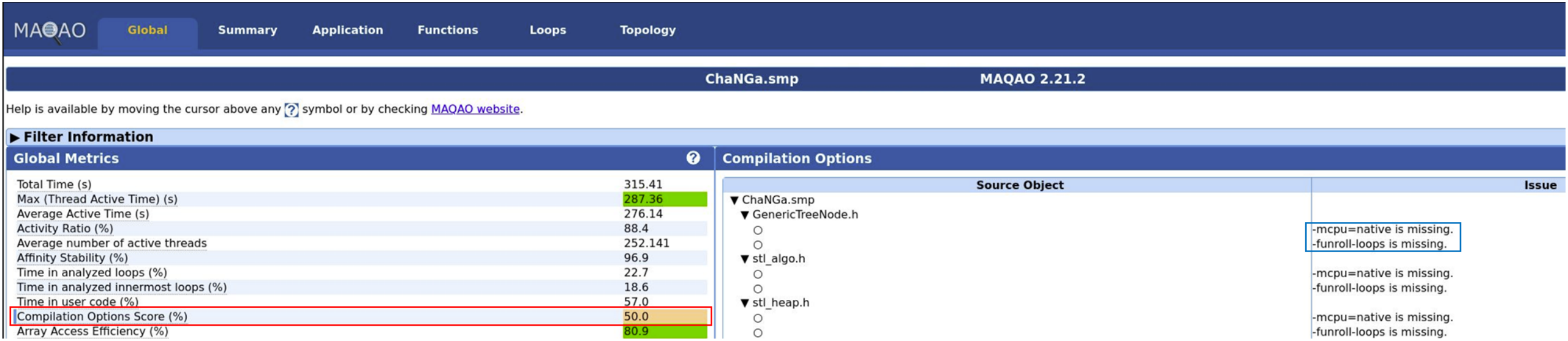


Source: https://www.space-coe.eu/codes/changa.php

ChaNGa

Compilation score

- Compilation score available and clickable into global metrics
- Enables to check the code was compiled with the appropriate flags
- Can be further inspected in case of low score to see which flags are missing
- Improve until you reach 100%!



ChaNGa

Compiler comparison

- Use the right compilation flags to achieve 100% compilation score
- Then try different compilers and compare global metrics on Grace node with SVE enabled

▶ **Filter Information**

Global Metrics

GCC 14.2.0

Total Time (s)	175.00
Max (Thread Active Time) (s)	168.82
Average Active Time (s)	166.33
Activity Ratio (%)	98.3
Average number of active threads	273.736
Affinity Stability (%)	99.2
Time in analyzed loops (%)	37.8
Time in analyzed innermost loops (%)	30.9
Time in user code (%)	88.7
Compilation Options Score (%)	100
Array Access Efficiency (%)	70.2

~6% difference of Total time

▶ **Filter Information**

Global Metrics

LLVM 20.1.1

Total Time (s)	164.76
Max (Thread Active Time) (s)	160.03
Average Active Time (s)	156.22
Activity Ratio (%)	97.4
Average number of active threads	273.065
Affinity Stability (%)	98.5
Time in analyzed loops (%)	40.5
Time in analyzed innermost loops (%)	18.5
Time in user code (%)	90.4
Compilation Options Score (%)	100
Array Access Efficiency (%)	80.9

ChaNGa

Vectorization analysis

- Comparison report between different configurations (e.g. compilation here)
- Metrics about vectorization with remaining speedups to be achieved
- Can be further inspected to find the most important loops to be optimized

▶ **Compared Reports**

Global Metrics

Metric		No vectorization	Neon vectorization	SVE vectorization
Total Time (s)		175.56	170.02	174.32
Max (Thread Active Time) (s)		171.54	166.37	170.33
Average Active Time (s)		164.31	163.75	164.76
Activity Ratio (%)		95.4	98.0	96.4
Average number of active threads		269.550	277.376	272.205
Affinity Stability (%)		99.8	99.5	99.8
Time in analyzed loops (%)		38.1	38.6	38.4
Time in analyzed innermost loops (%)		17.1	17.5	17.5
Time in user code (%)		84.9	84.7	85.0
Compilation Options Score (%)		100	100	100
Array Access Efficiency (%)		83.7	84.1	83.7
Potential Speedups				
Perfect Flow Complexity		1.01	1.01	1.01
Perfect OpenMP + MPI + Pthread		1.03	1.07	1.07
Perfect OpenMP + MPI + Pthread + Perfect Load Distribution		1.11	1.11	1.13
No Scalar Integer	Potential Speedup	1.26	1.20	1.20
	Nb Loops to get 80%	2	2	2
FP Vectorised	Potential Speedup	1.21	1.16	1.15
	Nb Loops to get 80%	2	2	2
Fully Vectorised	Potential Speedup	1.21	1.17	1.17
	Nb Loops to get 80%	2	2	2
Only FP Arithmetic	Potential Speedup	1.05	1.09	1.09
	Nb Loops to get 80%	6	4	4

ChaNGa on GRACE CPU, LLVM compiler, vectorization support comparison

ChaNGa

Loop profiling on NEON

- Find the most time-consuming loops
- See vectorization status into details and remaining speedup if fully vectorized
- Further inspect each loop and see recommendations to improve performances (vectorization, memory access patterns, …)
- Assembly code available (for the braves!)

Loop id	Source Location	Source Function	Max Inclusive Time Over Threads run_0 (s)	Vectorization Ratio (%)	Vector Length Use (%)	Speedup If FP Vectorized
175	ChaNGa.smp - gravity.h:151-302 [...]	partBucketForce(ExternalGravityParticle*, Tree::GenericTreeNode*, GravityParticle*, Vector3D<double>, int)	25.55	25.52	61.38	1.21
270	ChaNGa.smp - Ewald.cpp:149-242 [...]	TreePiece::BucketEwald(Tree::GenericTreeNode*, int, double)	5.88	8.02	53.48	1.85

- Looking at the code of first loop → several vectorization inhibitors are present:
  - conditional branches, function calls within loop, data dependencies in accumulation loop
- Recommendations:
  - Restructure the computations to minimize branching, inlining the called function, reorganizing the data layout into SoA

Focus on MAQAO

New Features

MAQAO new features

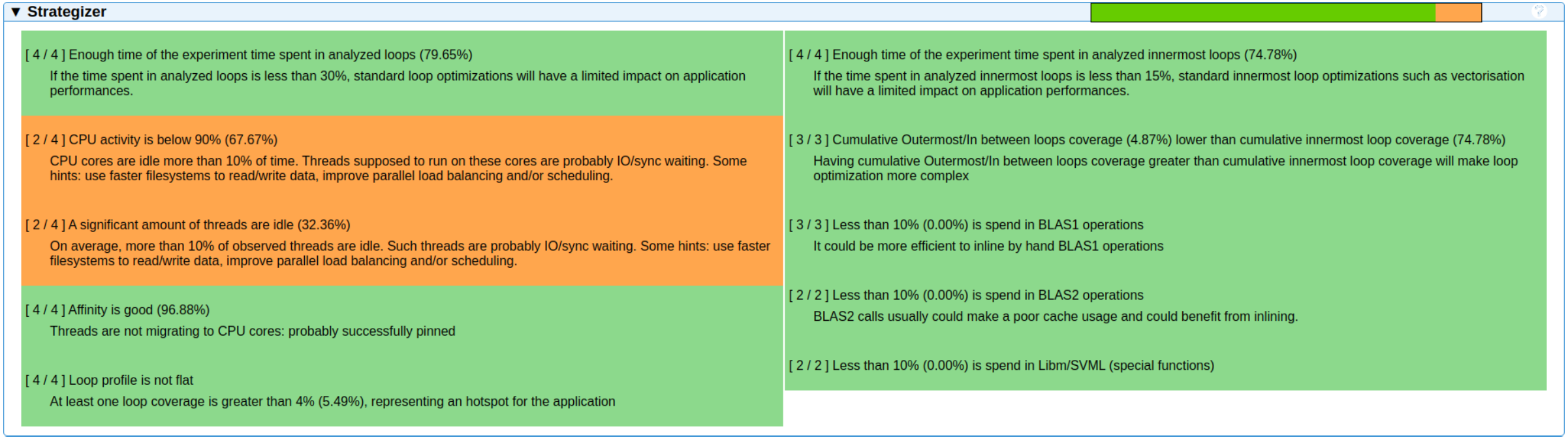
When collaboration brings ideas

- The EMOPASS project was the occasion for implementing new features in the MAQAO Performance Analysis Framework, such as:
  - Support of Arm Neoverse N1, Arm Neoverse V1 and Arm Neoverse V2,
  - High-level reports providing clear feedback on key issues in the application performance,
  - Detection of the activity of each thread during application execution,
  - Detection of threads placement, migration and pinning.

MAQAO new features

When collaboration brings ideas

- High level summary: overview of key points of the application profile and assessment of their impact on overall performance



▼ Strategizer

[4 / 4] Enough time of the experiment time spent in analyzed loops (79.65%)

If the time spent in analyzed loops is less than 30%, standard loop optimizations will have a limited impact on application performances.

[2 / 4] CPU activity is below 90% (67.67%)

CPU cores are idle more than 10% of time. Threads supposed to run on these cores are probably IO/sync waiting. Some hints: use faster filesystems to read/write data, improve parallel load balancing and/or scheduling.

[2 / 4] A significant amount of threads are idle (32.36%)

On average, more than 10% of observed threads are idle. Such threads are probably IO/sync waiting. Some hints: use faster filesystems to read/write data, improve parallel load balancing and/or scheduling.

[4 / 4] Affinity is good (96.88%)

Threads are not migrating to CPU cores: probably successfully pinned

[4 / 4] Loop profile is not flat

At least one loop coverage is greater than 4% (5.49%), representing an hotspot for the application

[4 / 4] Enough time of the experiment time spent in analyzed innermost loops (74.78%)

If the time spent in analyzed innermost loops is less than 15%, standard innermost loop optimizations such as vectorisation will have a limited impact on application performances.

[3 / 3] Cumulative Outermost/In between loops coverage (4.87%) lower than cumulative innermost loop coverage (74.78%)

Having cumulative Outermost/In between loops coverage greater than cumulative innermost loop coverage will make loop optimization more complex

[3 / 3] Less than 10% (0.00%) is spend in BLAS1 operations

It could be more efficient to inline by hand BLAS1 operations

[2 / 2] Less than 10% (0.00%) is spend in BLAS2 operations

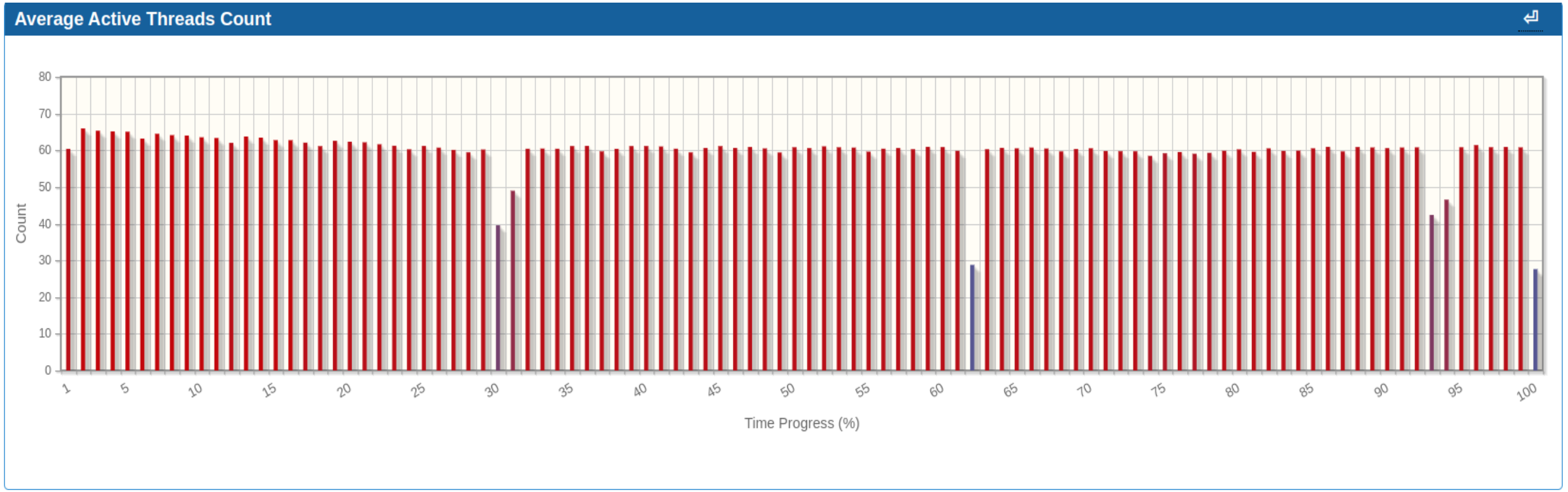
BLAS2 calls usually could make a poor cache usage and could benefit from inlining.

[2 / 2] Less than 10% (0.00%) is spend in Libm/SVML (special functions)

MAQAO new features

When collaboration brings ideas

- Threads activity summary: this view displays the average number of active threads over of the course of the application's execution



Focus on LLVM

Outer-Loop-vectorisation and other improvments

https://sipearl.com/wp-content/uploads/2025/03/LLVM_Dev_Meeting_2023_Poster_OLV_SiPearl.pdf

LLVM

When collaboration brings ideas

Outer-Loop-vectorisation (Poster presented at LLVM'23) :

- Often vectorizing the inner-most loop is the best thing to do
- However outer-loop vectorization is a better choice

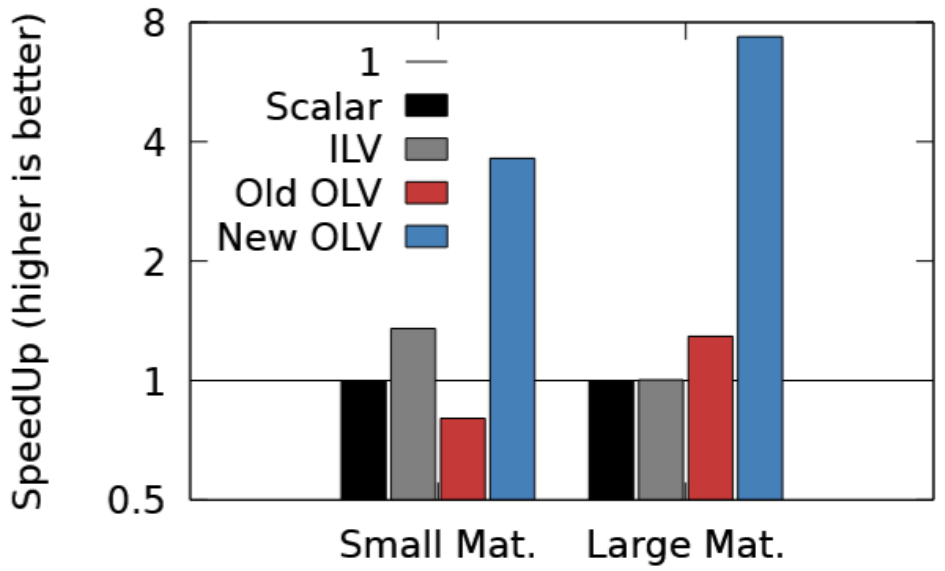
Inner-Loop Vectorization

```
for (size_t i = 0; i < N; i++) {
  float sum = 0.;
  // Pseudo-Vectorized inner loop:
  for (size_t j = 0; j < M; j += 8) {
    float[8] vec1 = B[j..j+8];
    float[8] vec2 =
        strided_load(&C[j][i], N); // Slow!
    sum += reduce_add(vec1 * vec2);
  }
  A[i] = sum;
}
```

Outer-Loop vectorization

```
// Pseudo-Vectorized outer loop:
for (size_t i = 0; i < N; i += 8) {
  float[8] sum = { 0., ... };
  for (size_t j = 0; j < M; j++) {
    float[8] vec1 = dup(B[j]);
    float[8] vec2 = C[j][i..i+8];
    sum += vec1 * vec2;
  }
  A[i..i+8] = sum;
}
```

Results for Matrix Multiplication on aarch64 (Graviton3e)



Small Matrix: 10^5 Entries (Everything fits in L1 Cache, Tiling),
Large Matrix: 10^8 Entries

LLVM

When collaboration brings ideas

Other improvments

- Improved vectorization and Fortran support in the new flang front-end
  - Support of many (mostly intel one) pragmas:
    - UNROLL, INTERLEAVE, DISTRIBUTE,
    - IVDEP
  - Help upstream to compile ABINIT with flang-new
  - Add missing flags for: –m64,
  - Add missing intrinsics :
    - PERROR
    - DERF/C
    - CHDIR
    - TAN, COTAN, COTAND,
- New, more accurate, scheduling model for Neoverse-V1
- …. **and more :** Branch on Superword, Data Dependent Exit, Scalable Interleave Groups,
- … And extensive evaluation of Vectorization Interleaving Factor impact on AArch64 (ISC'25 research paper)

Great Collaboration for Advancing the ARM Ecosystem for European Scientific Flagship Codes

Amazing team of experts

Great Collaboration Advancing the ARM Ecosystem

emopass

Thank you for your attention!

EVIDEN

Acknowledgement

MaX - Materials design at the Exascale has received funding from the European High Performance Computing Joint Undertaking and Participating Countries in Project (Czechia, France, Germany, Italy, Slovenia and Spain) under grant agreement no. 101093374.

Centres of Excellence for HPC Applications – Horizon-EuroHPC-JU-2021-COE-01

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Belgium, Czech Republic, France, Germany, Greece, Italy, Norway, and Spain under grant agreement No 101093441