MAQAO Use Cases

Cédric Valensi, Université de Versailles Saint-Quentin

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

MAQAO

- MAQAO is a performance analysis framework developed at Université of Versailles Saint-Quentin
- Relies on sampling-based profiling and static binary analysis
- Focuses on performance at the node/core level
- Reports available in HTML format
  - High-level reports characterising the whole application
  - Low-level reports down to loop granularity
- Additional reports for analyses encompassing multiple executions
- www.maqao.org

Code: Openradioss

MAQAO Global Metrics

MAQAO provides a set of global metrics characterising the whole application performance.

- Total Time: Execution time of the application
- Max (Thread Active Time): Time of the longest thread
- Activity Ratio: Average percentage of time during which the threads are active

Global Metrics	
Total Time (s)	3.90 E3
Max (Thread Active Time) (s)	2.82 E3
Average Active Time (s)	2.59 E3
Activity Ratio (%)	66.4
Average number of active threads	63.729
Affinity Stability (%)	100.0
Time in analyzed loops (%)	48.0
Time in analyzed innermost loops (%)	40.6
Time in user code (%)	50.9
Compilation Options Score (%)	100
Array Access Efficiency (%)	73.6

Large gap indicates a large amount of thread inactive time

Low numbers: poor resource usage

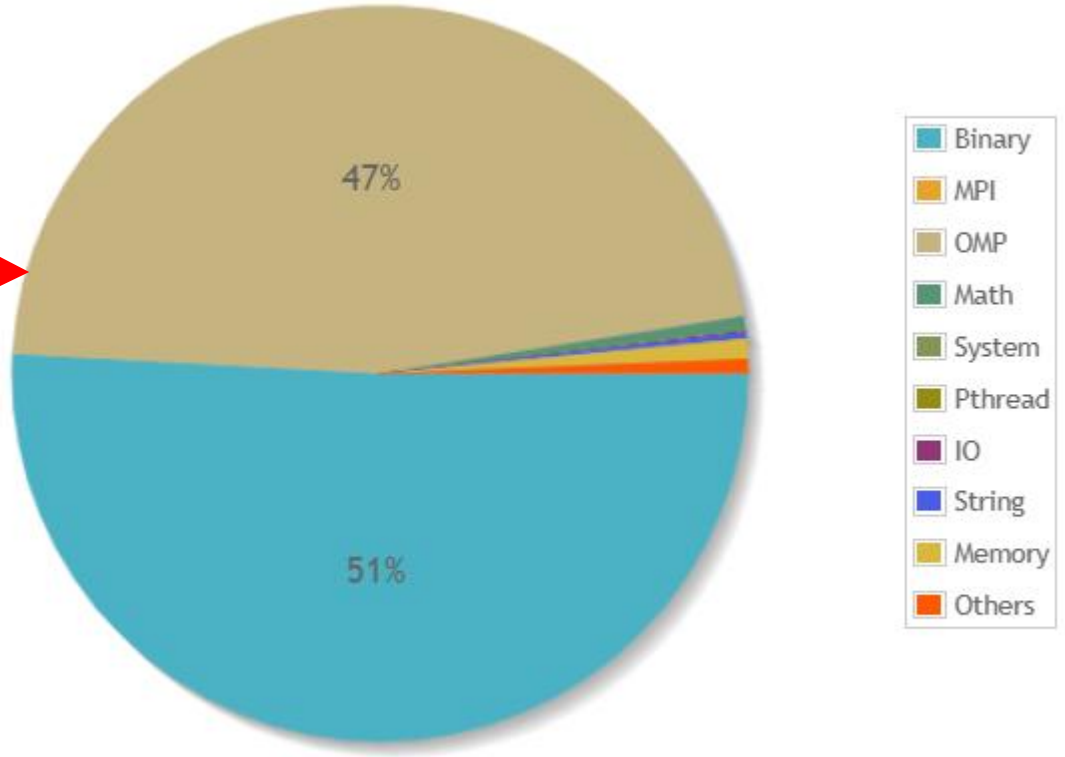
Experiment: Openradioss code (NEONM11 dataset) on AWS G4 with 1 MPI Rank, 96 OpenMP threads, ACFL –O3

MAQAO Time Categorisation and Potential Speedups

MAQAO also provides a breakdown of the time spent in various categories of codes and estimations of the gain to be expected if some optimisations were applied

Significant time spent in OMP runtime + Load balancing issues

Potential Speedups		
Perfect Flow Complexity		1.01
Perfect OpenMP + MPI + Pthread		1.87
Perfect OpenMP + MPI + Pthread + Perfect Load Distribution		2.04
No Scalar Integer	Potential Speedup	1.11
	Nb Loops to get 80%	17
FP Vectorised	Potential Speedup	1.02
	Nb Loops to get 80%	7
Fully Vectorised	Potential Speedup	1.11
	Nb Loops to get 80%	19
FP Arithmetic Only	Potential Speedup	1.29
	Nb Loops to get 80%	26



Experiment: Openradioss code (NEONM11 dataset) on AWS G4 with 1 MPI Rank, 96 OpenMP threads, ACFL –O3

MAQAO Function Profiling

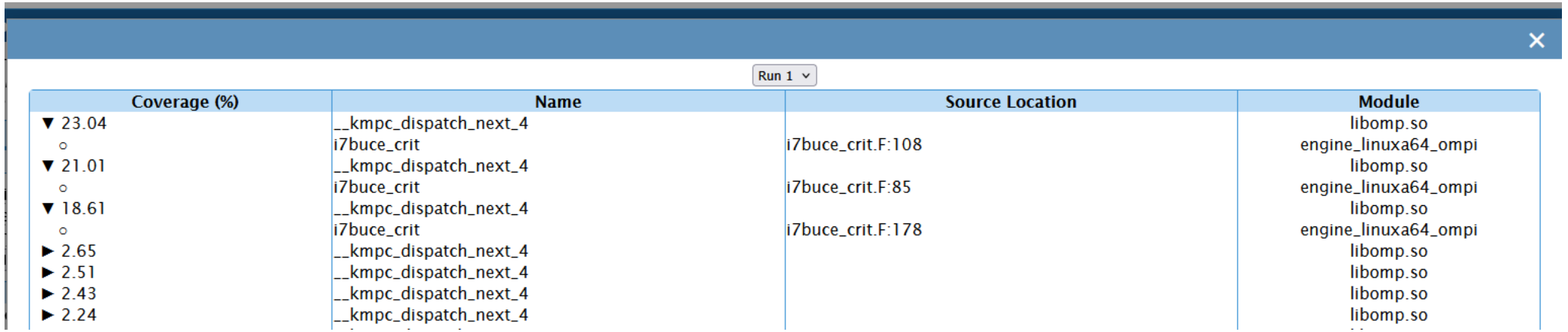
Name	Module	Coverage mpi_1_rank_omp_96_threads (%)	Nb Threads mpi_1_rank_omp_96_threads
○ __aarch64_cas4_acq_rel	libomp.so	14.96	96
○ int __kmp_dispatch_next_algorithm<int>(int, dispatch_private_info_template<int>*, dispatch_shared_info_template<int> volatile*, int*, int*, int*, traits_t<int>::signed_t*, int, int)	libomp.so	14.21	96
○ __kmp_wait_4	libomp.so	7.74	96
○ __aarch64_ldadd4_acq_rel	libomp.so	7.28	96
▶ asspar4	engine_linuxa64_ompi	6.63	96
▶ mulawc	engine_linuxa64_ompi	6.56	96
▶ i7buce_crit	engine_linuxa64_ompi	5.12	96
▶ m2cplr	engine_linuxa64_ompi	3.95	96
▶ cupdt3p	engine_linuxa64_ompi	3.26	96

The Function profile confirms that the hottest functions originate from the OpenMP runtime

Experiment: Openradioss code (NEONM11 dataset) on AWS G4 with 1 MPI Rank, 96 OpenMP threads, ACFL –O3

Call Chains Analysis

- Call chains allow tracing the source of the OMP wait



Coverage (%)	Name	Source Location	Module
▼ 23.04	__kmpc_dispatch_next_4		libomp.so
○	i7buce_crit	i7buce_crit.F:108	engine_linuxa64_ompi
▼ 21.01	__kmpc_dispatch_next_4		libomp.so
○	i7buce_crit	i7buce_crit.F:85	engine_linuxa64_ompi
▼ 18.61	__kmpc_dispatch_next_4		libomp.so
○	i7buce_crit	i7buce_crit.F:178	engine_linuxa64_ompi
► 2.65	__kmpc_dispatch_next_4		libomp.so
► 2.51	__kmpc_dispatch_next_4		libomp.so
► 2.43	__kmpc_dispatch_next_4		libomp.so
► 2.24	__kmpc_dispatch_next_4		libomp.so

➔ MAIN ISSUES

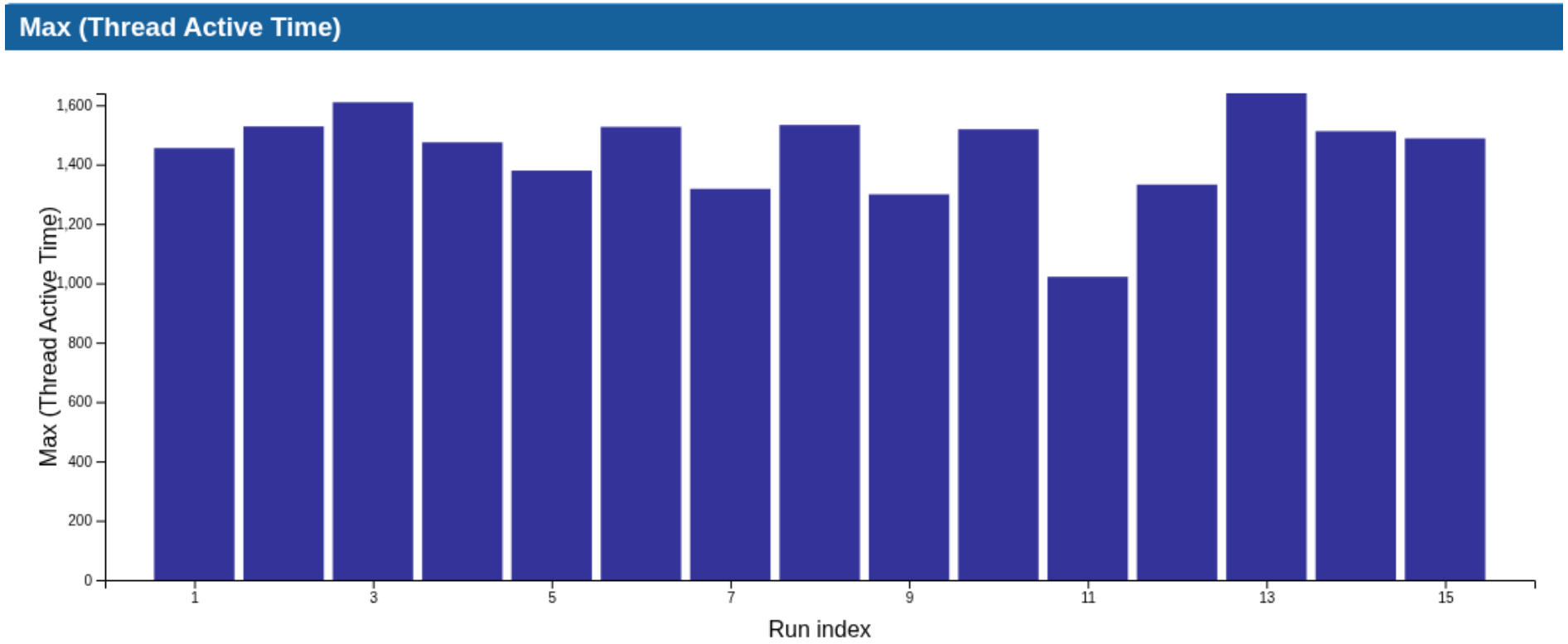
- Critical section (OMP CRITICAL) in i7buce_crit
- Locks (OMP_SET_LOCK) mostly from i7optcd (lockon.inc:28)

Experiment: Openradioss code (NEONM11 dataset) on AWS G4 with 1 MPI Rank, 96 OpenMP threads, ACFL –O3

Code Quantum Package

Measurements Stability

- MAQAO stability mode allows to detect time variations between runs



Experiment: Quantum Package code on Zen4 with 1 MPI Rank, 192 OMP threads, gfortran –O3

Measurements Stability

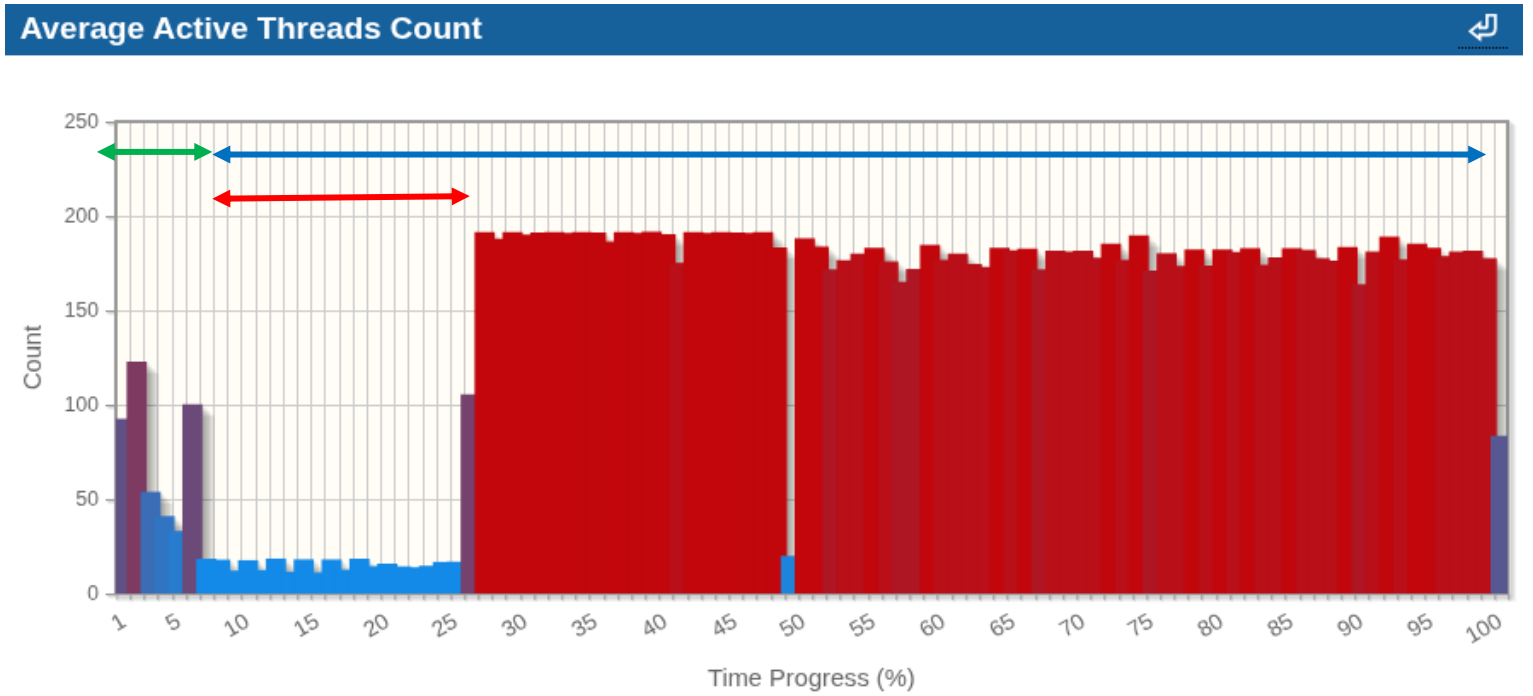
- Statistics on 15 runs:

Name	Module	min (Coverage) (%)	avg (Coverage) (%)	med (Coverage) (%)	max (Coverage) (%)
▶ select_singles_and_doubles	fci	28.66	36.69	38.10	38.92
○ get_two_e_integral_cache	fci	9.61	10.99	10.41	14.90
▶ h_s2_u_0_nstates_openmp_work_3	fci	9.19	10.48	9.87	14.13
○ compare_int32_t	fci	6.28	7.82	8.00	8.38
▶ get_mask_phase	fci	5.10	6.72	6.91	7.17
▶ get_all_spin_singles_3	fci	3.18	3.69	3.51	4.96
▶ get_d2	fci	2.41	3.08	3.16	3.29

- Variation in the duration of each run is expected due to two main factors:
  - A random generation process followed by a selection of samples
  - By the code being asynchronous by design
- Top functions stay the same
- **Reliability: following measurements were made multiple times, biggest outliers were removed and the average value was kept**

MAQAO Thread Activity

- MAQAO displays the number of average active threads over the execution



- First phase is the initialization phase which is quite short
- Beginning of the second phase is the first step showing load imbalance

Detecting Compiler Options

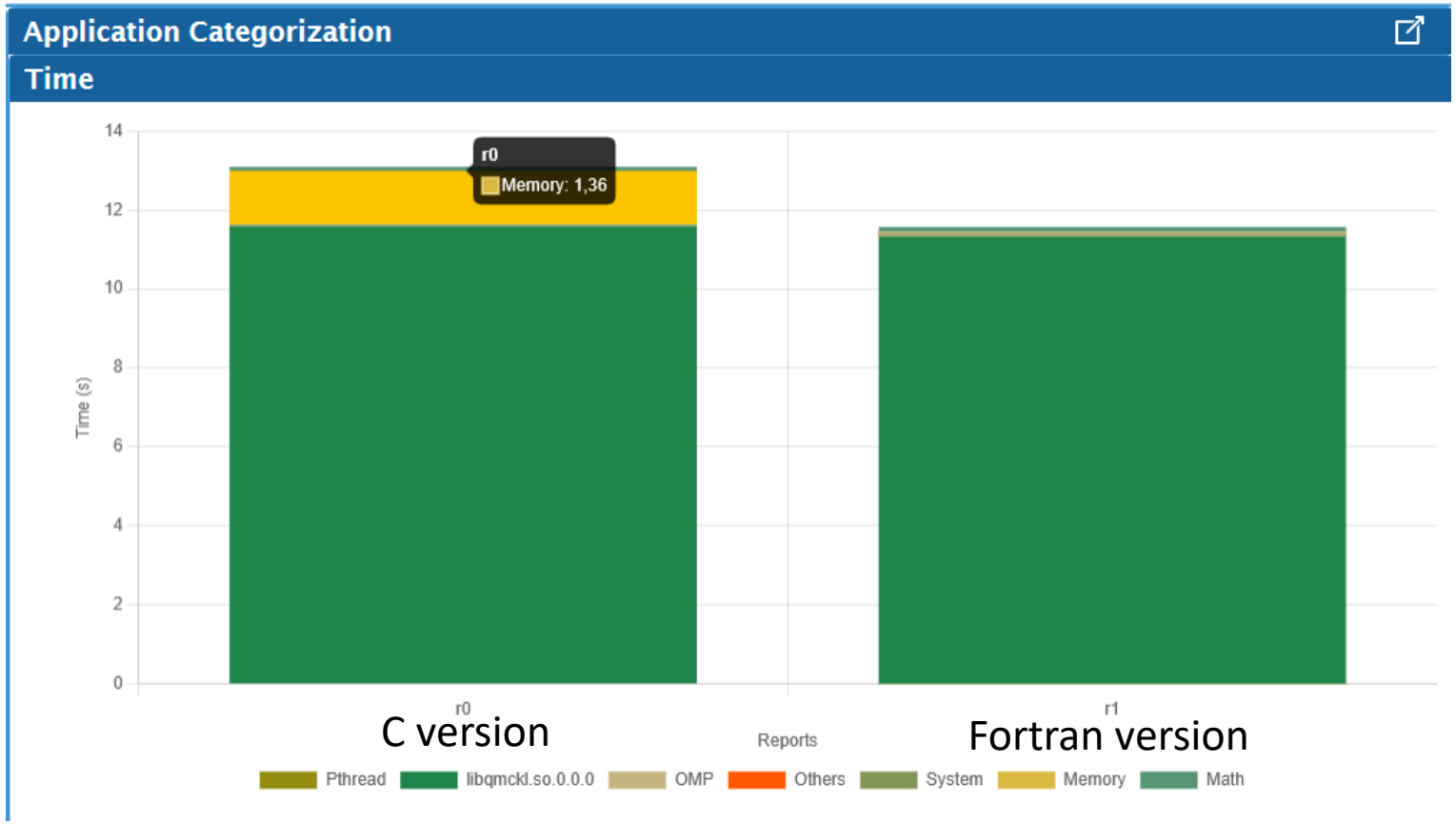
- MAQAO can provide the compilation flags at the function level
- ➔ Function *compare_int32_t* was not compiled with optimization or specialization

Name	Module	Max Thread Time / Walltime run_0 (%)	Coverage run_0 (%)	Compilation Options
▶ select_singles_and_doubles	fci	22.37	19.15	GNU Fortran2008 14.2.0 -march=znver4 -mmmx -mpopcnt -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -mavx -mavx2 -msse4a -mno-fma4 -mno-xop -mfma -mavx512f -mbmi -mbmi2 -maes -mpclmul -mavx512vl -mavx512bw -mavx512dq -mavx512cd -mavx512vbmi -mavx512ifma -mav...
○ compare_int32_t	fci	16.95	10.56	GNU C17 14.2.0 -mtune=generic -march=x86-64 -g -fPIC
○ __memcpy_avx512_unaligned_erms	libc.so.6	16.06	15.16	
○ get_two_e_integral_cache	fci	14.14	13.59	GNU Fortran2008 14.2.0 -march=znver4 -mmmx -mpopcnt -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -mavx -mavx2 -msse4a -mno-fma4 -mno-xop -mfma -mavx512f -mbmi -mbmi2 -maes -mpclmul -mavx512vl -mavx512bw -mavx512dq -mavx512cd -mavx512vbmi -mavx512ifma -mav...
▶ h_s2_u_0_nstates_openmp_work_3_._omp_fn.0	fci	12.94	12.93	GNU Fortran2008 14.2.0 -march=znver4 -mmmx -mpopcnt -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -mavx -mavx2 -msse4a -mno-fma4 -mno-xop -mfma -mavx512f -mbmi -mbmi2 -maes -mpclmul -mavx512vl -mavx512bw -mavx512dq -mavx512cd -mavx512vbmi -mavx512ifma -mav...
▶ get_mask_phase	fci	9.07	7.22	GNU Fortran2008 14.2.0 -march=znver4 -mmmx -mpopcnt -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -mavx -mavx2 -msse4a -mno-fma4 -mno-xop -mfma -mavx512f -mbmi -mbmi2 -maes -mpclmul -mavx512vl -mavx512bw -mavx512dq -mavx512cd -mavx512vbmi -mavx512ifma -mav...
▶ get_all_spin_singles_3	fci	6.03	5.15	GNU Fortran2008 14.2.0 -march=znver4 -mmmx -mpopcnt -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -mavx -mavx2 -msse4a -mno-fma4 -mno-xop -mfma -mavx512f -mbmi -mbmi2 -maes -mpclmul -mavx512vl -mavx512bw -mavx512dq -mavx512cd -mavx512vbmi -mavx512ifma -mav...

Code: QMCkl

MAQAO Comparison Mode

- Code exists in two versions: C and Fortran
- MAQAO comparison mode showed that C version was slower than Fortran version, with the difference being due to memory functions



Pinpointing Differences

- Timing difference was due to memset function added by Intel C compiler

Functions

Name	Module	Coverage (%)		Inclusive Time w.r.t. Wall Time(s)		Max Inc. Time over Threads(s)		
		test_memory	test_memory_fortran	test_memory	test_memory_fortran	test_memory	test_memory_fortran	test_me
qmckl_compute_jastrow_champ_factor_single_een_hpc_f	libqmckl.so.0.0.0	NA	94.64	NA	10.96	NA	10.96	NA
qmckl_compute_jastrow_champ_factor_single_een_hpc_c	libqmckl.so.0.0.0	86.12	NA	11.29	NA	11.29	NA	1
__intel_avx_rep_memset	binary	10.07	NA	1.32	NA	1.32	NA	1
__svml_exp4_l9	binary	0.80	1.17	0.10	0.13	0.10	0.14	1
qmckl_distance	libqmckl.so.0.0.0	0.61	1.21	0.08	0.14	0.08	0.14	1
qmckl_compute_een_rescaled_e_hpc.omp_outlined	libqmckl.so.0.0.0	0.80	0.95	0.10	0.11	0.10	0.11	1
qmckl_compute_een_rescaled_single_e_doc	libqmckl.so.0.0.0	0.42	0.39	0.05	0.04	0.05	0.05	1
__intel_avx_rep_memcpy	binary	0.31	0.30	0.04	0.04	0.04	0.04	1
qmckl_compute_een_rescaled_single_n	libqmckl.so.0.0.0	0.19	0.13	0.03	0.02	0.03	0.01	1
qmckl_compute_een_rescaled_n	libqmckl.so.0.0.0	0.08	0.22	0.01	0.02	0.01	0.03	1
qmckl_provide_een_rescaled_single_e	libqmckl.so.0.0.0	NA	0.09	NA	0.01	NA	0.01	NA
clear_page_erms	kernel	NA	0.09	NA	0.01	NA	0.01	NA
__pthread_getspecific	libc.so.6	NA	0.09	NA	0.01	NA	0.01	NA
__tls_get_addr	ld-linux-x86-64.so.2	0.04	0.04	0.00	0.00	0.00	0.00	1

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.