26-05-2022

D3.3 – Initial report on the performance characteristics on relevant hardware for upcoming supercomputers

Version 1.0

GA no 952165

Dissemination Level

- ☒ PU: Public
- ☐ PP: Restricted to other programme participants (including the Commission)
- ☐ RE: Restricted to a group specified by the consortium (including the Commission)
- ☐ CO: Confidential, only for members of the consortium (including the Commission)

Document Information

Project Title	Targeting Real Chemical accuracy at the EXascale
Project Acronym	TREX
Grant Agreement No	952165
Instrument	Call: H2020-INFRAEDI-2019-1
Topic	INFRAEDI-05-2020 Centres of Excellence in EXascale computing
Start Date of Project	01-10-2020
Duration of Project	36 Months
Project Website	https://trex-coe.eu/

Deliverable Number	D3.3
Deliverable title	D3.3 – Initial report on the performance characteristics on relevant hardware for upcoming supercomputers, as seen in GA
Due Date	M18 – 31-03-2022 (from GA)(from GA)
Actual Submission Date	26-05-2022

Work Package	WP3 – Code modularization and interfacing
Lead Author (Org)	Dirk Pleiter (KTH)
Contributing Author(s) (Org)	Kévin Camus (UVSQ), Johan Hellsvik (KTH), William Jalby (UVSQ), Joe Jordan (KTH), Daniel Medeiros (KTH), Anthony Scemama (CNRS), Alessandra Villa (KTH),
Reviewers (Org)	Cedric Valensi (UVSQ), Axel Auweter (Megware)
Version	1.0
Dissemination level	PU
Nature	Report
Draft / final	Final

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Disclaimer

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Versioning

Version	Date	Authors	Notes
0.1	2022-03-11	J. Hellsvik, W. Jalby, J. Jordan, D. Medeiros, D. Pleiter, A. Scemama	Draft for internal review
0.2	2022-05-06	J. Hellsvik, W. Jalby, J. Jordan, D. Pleiter, A. Scemama, A. Villa	New draft for internal review
1.0	2022-05-26	W. Jalby, J. Jordan, D. Pleiter, A. Scemama, A. Villa	Final version

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Abbreviations

CoE	Center of Excellence
EPI	European Processor Initiative
FCIQMC	Full Configuration Interaction quantum Monte Carlo
GPU	Graphics Processing Unit
IPP	Intel Performance Primitives
QMC	Quantum Monte Carlo
UVSQ	Université de Versailles Saint-Quentin-en-Yvelines
WP	Work Package

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table of Contents

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

List of Figures

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

List of Tables

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

1 Executive Summary

This deliverable documents the initial performance analysis results obtained for all 6 TREX flagship applications. The focus was, in particular, the scaling of the application and the ability to exploit parallelism at all the different levels of modern HPC architectures. This ranges from the efficient use of SIMD instructions to the use of highly parallel compute accelerators like Graphics Processing Units (GPUs).

For assessing the applications in terms of scalability, it needs to be taken into account that they differ in terms of their principle ability to be highly parallelized. Some of the applications, e.g. QMC=Chem, implement scalable methods with a strong focus on scalability, which could be demonstrated using up to 32,768 CPU cores. Furthermore, very encouraging results have been obtained for TurboRVB from GPU acceleration using Europe's currently fastest supercomputer, i.e. JUWELS Booster.

The performance results collected for this deliverable will help to guide further work and optimisations during the second half of the project.

2 Introduction

In this deliverable, we focus on the following objective of Work Package (WP) 3: *Provide empirical data and knowledge related to performance characteristics of Quantum Monte Carlo (QMC) applications and more specifically the QMCkl library*. We do this by documenting results from an initial performance analysis of the TREX flagship applications on currently available HPC systems. For initial performance results for the QMCkl library, we refer to the deliverable D3.2.

Like in the earlier deliverable (see D3.1), the results have been obtained for specific workloads, which can be considered reasonably realistic for real-life workloads while being executable with a moderate amount of resources.

The analysis focuses on the following aspects:

- Systematic evaluation of the ability to exploit SIMD instructions
- Systematic analyses of unicore and multicore behavior for driving code optimization
- Systematic gathering of performance measurements (including hardware counters) which can be used in co design
- The scaling behaviour of the applications.
- Assessment of the attainable performance on given hardware architectures.
- Exploration of acceleration on GPUs.

These aspects have been analysed for as many of the flagship codes as possible. However, porting of codes to GPUs has only been started and (encouraging) results for only a single application can be provided at this point.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

The current state of the analysis is nevertheless useful for assessing performance on upcoming architectures, e.g. future EuroHPC exascale systems. This allows identifying bottlenecks in the codes when deployed to (pre)exascale system, which is a key step to move forward to achieve objective 1 of the Center of Excellence (CoE): *Co-design of computational kernels of flagship QMC codes with efficient algorithms that are scalable for HPC applications, are flexible to adapt to future architectures, and can cater to a large base of HPC users and players in synergy with existing CoEs.* The hardware platforms used for this deliverable have been chosen such that they can be seen as precursors for such exascale systems. Furthermore, the analysis results can also be used as co-design input for hardware development projects like the European Processor Initiative (EPI). It should be noted that Université de Versailles Saint-Quentin-en-Yvelines (UVSQ) is collaborating with Sipearl and Atos in a common project on performance and optimization tools for Sipearl processor developed within EPI.

Some of the application source codes as well as workloads have not been published under an open-source license, yet. This document, therefore, includes references to repositories that may not be accessible to the reader, but have been included with the intention to comprehensively document all information needed to reproduce the presented results. Access to these repositories may be given upon request.

The document is organized as follows: After presenting more details on the methodology in section 3, we document the used hardware platforms in section 4. Next, we present the results for the different applications in section 5. Finally, conclusions are presented in section 6.

3 Methodology

The approach to performance analysis and characterization of the applications has been the following:

1. Generate a performance profile using one or few threads or ranks within a single node for a given workload to identify performance relevant regions,

2. Generate detailed performance metrics and measurements of the loops first to optimise them and second to contribute to the co design repository

3. Evaluate the scaling properties of the given workload. That is to evaluate the scaling properties of the given workload in terms of things like load balance, communication efficiency, and memory access patterns

For these different steps, the following tools have been used:

- HPCToolkit:[1] A sampling-based performance measurement tool [1] essentially has been used for step 1

- MAQAO:[2] A sampling-based performance measurement tool that also performs static analysis of the compiler-generated instructions and provides code developer with guidelines for optimization instructions [2, 3]. It has been used for steps 1, 2 and 3.

[1] http://hpctoolkit.org/
[2] http://www.maqao.org/

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

- Nsight:[3] A performance analysis tool designed to identify scale efficiently across any quantity or size of CPUs and GPUs developed by Nvidia has been used for step 3

Furthermore, most of the applications do have internal timers, which have also been used for measuring execution times.

The different tools are based on different approaches to collecting performance information and also have very different capabilities. The different tools are thus used in a complementary manner. HPCToolkit has been primarily used for obtaining profiles for full workload executions using a single or a small number of processes within a single node. This tool was in parts used for measuring hardware performance counters like cache miss counters. Nsight was used for generating detailed profiles for TurboRVB, which is currently the only application that can successfully leverage GPUs.

MAQAO toolset was used with three main goals in mind: first profile and analyse applications, second, guide code owners for code optimisation and finally set up a repository with detailed application behavior. All of the profiling and analysis results were carried out on the six TREX applications and on QMCkl. The same reference machine with a controlled software environment was used for all the experimentations allowing meaningful comparisons. The analysis results allowed us to assess code quality, vectorization capabilities, memory hierarchy usage, and intranode parallelism characteristics of the codes, constituting an excellent basis for co-design activities. MAQAO analysis capabilities also allow for an estimation of the performance benefits, which might be achieved from further vectorisation, as well as the required efforts. All of these results have been made available at a common public website (https://datafront.maqao.exascale-computing.eu/public_html/oneview2020/) for codes with no access restriction, while for other codes requiring access protection profiling, results were made available at a protected site (https://datafront.maqao.exascale-computing.eu/private_html/oneview2020/).

Analyzing performance requires an understanding of the hardware performance characteristics. For this purpose, we collected micro-benchmark information on several supercomputing platforms considered in the context of this deliverable. The results have been documented in appendix A.

Definitions

In Table 1 we collect a set of symbols, which are used in this document, and their meaning. In Table 2 we define some of the terminologies that are used in the remaining part of this deliverable. Finally, Table 3 documents the key metrics on which we focus in this deliverable.[4]

Table 1: Notation

t_{loop}	Execution time spent in a given loop
t_{total}	Execution time of the full workload

[3] https://developer.nvidia.com/nsight-systems

[4] For a more comprehensive overview see http://www.maqao.org/documentation.html.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

N_{core}	Number of CPU cores
N_{node}	Number of compute nodes

Table 2: Definitions

Speed-up $S(n)$	The speed-up is defined as the ratio of execution time $t(1)$ on a single core or node versus the execution time $t(n)$ on n cores or nodes, i.e. $S(n) = t(1)/t(n)$.
Parallel efficiency $\epsilon(n)$	Parallel efficiency is defined as the speed-up per number of cores or nodes, i.e. $\epsilon(n) = S(n)/n$.
Strong scaling analysis	For a strong scaling analysis the execution time is investigated for a fixed workload W using n cores or compute nodes, i.e. the execution time $t(n, W)$ is considered for different n.
Weak scaling analysis	For a weak scaling analysis the workload W is increased by a factor n when increasing the amount of compute resources by n, i.e. the execution time $t(n, n \cdot W)$ is considered for different n.
Ideal strong scaling	Ideal strong scaling refers to the expectation that an n-fold increase of the amount of the used parallel compute resources should result in an n-fold reduction of the execution time if the amount of work W is kept fixed, i.e. $t(n, W) = t(1, W)/n$ or $\epsilon(n) = 1$.
Ideal weak scaling	Ideal weak scaling refers to the expectation that an n-fold increase of the amount of the used parallel compute resources and an n-fold increase of the amount of work leaves the execution time constant, i.e. $t(n, n \cdot W) = t(1, W)$.
Arithmetic Intensity	Ratio of the number of floating-point operations I_{fp} versus the amount of data I_{mem} transferred from or to memory, i.e. $\text{AI} = I_{\text{fp}}/I_{\text{mem}}$, usually expressed in flops/byte.
Stride 1 memory access	This term denotes a memory access pattern resulting from accessing an array $x(i)$ such that the array index i is monotonically incremented (or decremented) by 1 during a next access. For example, if the size of an array element $x(i)$ is 8 bytes and a denotes the address of the array element $x(1)$ then reading the array elements $x(1), x(2), x(3), \ldots$ results in load instructions accessing the (virtual) addresses $a, a + 8, a + 16, \ldots$.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Compiler auto-vectorization	When a compiler is able to issue SIMD instructions in loops which independently use the same instruction for multiple data items

Table 3: MAQAO metrics

Time in analysed loops	Percentage of time that an application spends in loops (excluding loops present in modules not analysed by the profiler).
Time in analyzed innermost loops	Percentage of time that an application spends in innermost loops.
Perfect Flow Complexity	An optimistic estimate of the speed-up available by reducing the number of paths in loops. Having multiple paths in a loop can prevent the compiler to vectorize the code, decreasing performance. The metric provides an estimation of a global speed-up that can be achieved if all loops with more than two paths were fully vectorized. If no speed-up can be obtained the loop has an optimal control flow, hence lower is better (1 being optimal)
Array Access Efficiency	Percentage of processor friendly data layout. Accessing contiguous data is faster. Higher is better (100 being optimal).
Fully Vectorized: Potential speed-up	Optimistic speed-up if all instructions are vectorized in all analysed loops. Lower is better (1 being optimal).
Fully Vectorized: Number of loops	Number of loops to optimize to get 80% in speed-up. Lower is better (0 being optimal).
Vectorization Ratio	The percentage of the loop instructions that have been vectorized. (100 being optimal)
Vectorization Efficiency	Ratio of the average width of the registers used within the loop over the maximum width of a vector register, in percent. Higher is better (100 is optimal).

4 Hardware Platforms

Emerging hardware platforms

For assessing the performance of TREX codes, the goal is to take into account the dominant architectures and technologies. For this reason not only current top-tier systems, but also future exascale systems are considered. Based on the recently concluded procurement of HPC systems by EuroHPC we make the following observations:

- With one exception, namely Deucalion in Portugal, all systems are using x86-based processors. Deucalion will have a partition with Arm-based A64FX processors.
- The used processors feature a large number of cores (up to 128 cores per node) and support SIMD instructions with a width of 256 or 512 bits.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

- All systems integrate GPUs as compute accelerators (to a different extent).

While at this point x86-based processors are dominant, Arm-based processors are expected to become more relevant with the EPI and its coordinator, Atos, being positioned as a provider of Arm-based exascale technology. Performance evaluation on Arm-based systems is not part of this deliverable but is planned for the future.

Used hardware platforms

Dardel

Dardel[5] is a supercomputer funded by the SNIC and hosted at the PDC at KTH (Sweden). Dardel is a Cray EX system which currently has 554 nodes each with two AMD EPYC 7742 processors having 256 GiByte of memory. An addition of a GPU partition based on nodes with one AMD EPYC Trento generation processor plus 4 AMD MI250x GPUs will be added by the end of 2022, resulting in a final peak performance of more than 13.5 PFlop/s. The interconnect uses a Slingshot network with Dragonfly topology.

This machine is similar in many regards to the upcoming exascale system Frontier at ORNL (US) and the EuroHPC pre-exascale system LUMI at CSC (Finland) so it is an important target for the TREX project

Joliot-Curie (AMD Irene Partition)

TGCC Joliot Curie[6] is a system procured by GENCI and operated by CEA (France). The Sequana XH2000 system comprises different partitions. In the context of this deliverable, the AMD Irene ROME was used, which comprises 2,292 nodes with 2 AMD EPYC 7H12 processors and 256 GiByte of memory each. In total the 293,376 cores provide a peak performance of 11.75 Pflop/s. The interconnect uses a Mellanox HDR100 network with a Dragonfly+ network topology.

Galileo100

Galileo100[7] is a cluster at CINECA (Italy) comprising 528 computing nodes. Each of these nodes is equipped with 2 Intel Xeon Platinum 8260 (Cascade Lake) processors and at least 384 GiByte main memory. The interconnect uses Mellanox HDR100 technology and a Dragonfly+ network topology.

Marconi100

Marconi100[8] is currently the fastest system installed at CINECA (Italy). It comprises 980 nodes with 2 IBM POWER9 processors and 4 NVIDIA V100 GPUs each. The V100 GPUs have a memory capacity of 16 GiByte. The overall system has a peak performance is 21.6 PFlop/s. The interconnect uses Mellanox Infiniband EDR links using a Dragonfly+ topology.

[5] https://www.pdc.kth.se/hpc-services/computing-systems/about-dardel-1.1053338
[6] http://www-hpc.cea.fr/en/complexe/tgcc-JoliotCurie.htm
[7] https://www.hpc.cineca.it/hardware/galileo100
[8] https://www.hpc.cineca.it/hardware/marconi100

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

JUWELS Booster

The JUWELS Booster at JSC (Germany) is a GPU-accelerated Sequana XH2000 system comprising 936 compute nodes. Each node hosts 2 AMD EPYC processors with 24 cores each and 4 NVIDIA A100 GPUs. The A100 GPUs have a memory capacity of 40 GiByte. The interconnect uses Mellanox Infiniband HDR links using Dragonfly+ topology.

MPG-FKF Cluster

Parts of the benchmarking of the NECI code were done on a local cluster at the Max Planck Institute for Solid State Research. Each node hosts 2 Intel Xeon E5-2680 v2 (Ivy Bridge) processors with 10 cores each.

HPC Server at UVSQ

For a detailed analysis of the single-node performance also an HPC server at UVSQ has been used. The system is equipped with two Intel Xeon Platinum 8170 processors with 24 cores each.

Processor features

Table 4 summarizes some key features of the processors used on the earlier presented systems.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 4: Processor hardware parameters.

Processor type	Intel Xeon			AMD EPYC		IBM
	E5-2680 v2 (Ivy Bridge)	Platinum 8170 (Skylake)	Platinum 8260 (Cascade Lake)	7H12 (Rome)	7742 (Rome)	POWER9
Number of cores	10	26	24	64	64	16
Base clock (GHz)	2.80	2.1	2.4	2.6	2.25	3.1
SIMD ISA	AVX	AVX2, AVX512	AVX2, AVX512	AVX2	AVX2	VMX
SIMD width (bits)	256	512	512	256	256	128
SIMD pipelines	2	2	2	2	2	2
FP64 (flops/cycle)	16	32	32	16	16	8
L1D / core (kiBytes)	32	32	32	32	32	64
L2 / core (kiBytes)	256	256	1024	512	512	512
L3 (MiBytes)	25	35.75	35.75	256	256	80

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

5 Results

In this section results are presented for each of the six TREX applications (CHAMP, GammCor, NECI, QMC=Chem, Quantum Package, TurboRVB) considered for this deliverable. Among the six codes, we can distinguish three different application profiles. The first one is characterized by a single application running on all the resources of a supercomputer including GPU accelerators. Today, only TurboRVB fits in this profile. The second profile is similar to the first one, without GPU acceleration. CHAMP and QMC=Chem belong to this second profile. In the third profile, the supercomputer is used in a high-throughput mode where many independent jobs are run independently on a relatively small number of compute nodes orchestrated by a workflow manager such as AiiDA. NECI, Quantum Package and GammCor belong to this third class.

CHAMP

Application and workload description

The Cornell-Holland Ab-initio Materials Package (CHAMP) is a package with different capabilities for Monte Carlo electronic structure calculations of molecular systems [4]. Here we focus on the Variational Monte Carlo (VMC) capability. Key properties of the current code are summarized in Table 5.

In the following analysis, commit 8a5cf87 of CHAMP was used. All results have been obtained using a VMC test workload related to Butadiene compounds, which is part of CHAMP's continuous integration test suite. In the following a modified version of workload, `butadiene_cipsi15K_T_optWF`[9] has been used, where the optimization option was removed.

For a given workload CHAMP increases the number of simultaneous Monte Carlo evaluations of integrals as the number of parallel processes is increased. More integral evaluation entails better statistical accuracy and computing more in a parallel fashion means that time to scientific solution is reduced. Since the amount of work is scaled with the number of processes the weak-scaling case is considered.

Single-thread performance profiles

A single-thread analysis has been performed using the HPC server at UVSQ, which is equipped with Intel Skylake processors. The code has been compiled using the Intel Fortran compiler version 2021.5.0 with `-O2` compiler optimizations. As shown in 1, most of the execution time of CHAMP is spent in the CHAMP binary itself, with little time spent in numerical library calls (math).

The total execution time is $t_{\text{total}} = 21\,\text{s}$. Figure 2 shows selected MAQAO global metrics. It shows that by far most of the execution time, around 80 %, is spent in loops. Most of them are innermost loops which are much simpler to optimize. The metric *Perfect Flow Complexity* indicates that innermost loops do not have a complex control flow, which is good. The metric *Array Access Efficiency* is high (over 70%), which means that the data layout is processor friendly with mostly

[9] https://github.com/filippi-claudia/champ/tree/main/tests/CI_test/VMC-Butadiene-ci1010_pVTZ-15000-dets

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 5: Selected properties of the CHAMP code.

Programming language	Fortran 2008 / Fortran 77
Process-level parallelism	MPI
Thread-level parallelism	–
SIMD parallelism	Compiler auto-vectorization
GPU Acceleration	–
Parallel libraries	–

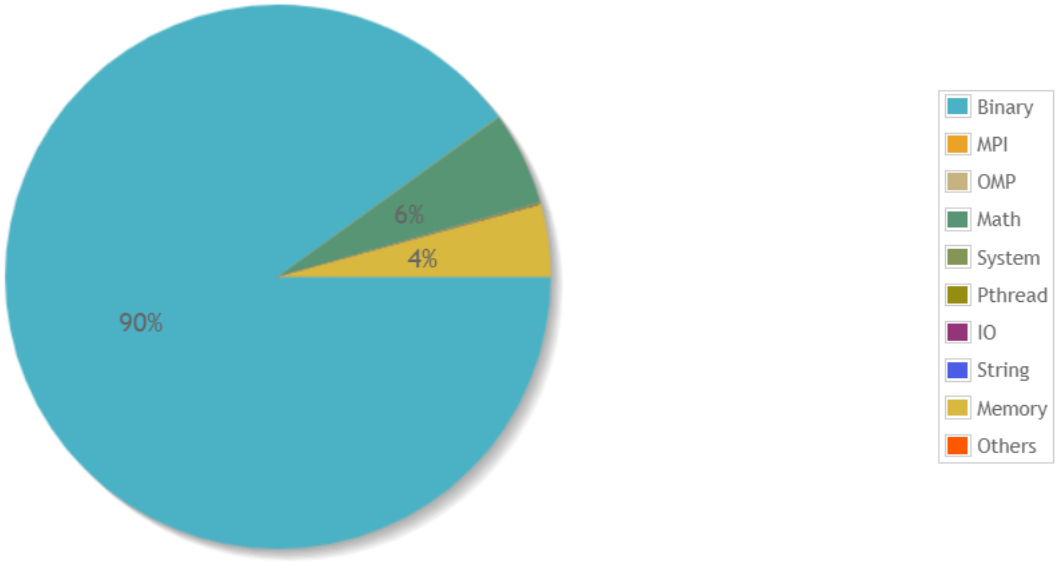

Figure 1: Categorization of CHAMP single-thread execution time based on MAQAO measurements.

stride 1 memory accesses. A further dynamic analysis indicated that most of the data access are performed out of L2 indicating a very good use of memory hierarchy. Finally, Figure 3 shows that many loops are well vectorized (see column *Vectorization Ratio*): the hottest first and fifth loop are not vectorized only due to the presence of indirect access: this will be solved by forcing the compiler to vectorize using gather instructions. Similarly, the column *Vectorization Efficiency* is a bit disappointing due a very conservative compiler strategy which by default prefers to use 256 bits vectors instead of the 512 bits ones available: again this will be solved by using appropriate compiler flags.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Global Metrics		
Total Time (s)		21.32
Profiled Time (s)		20.92
Time in analyzed loops (%)		79.7
Time in analyzed innermost loops (%)		43.8
Time in user code (%)		90.0
Compilation Options		OK
Perfect Flow Complexity		1.00
Iterations Count		1.03
Array Access Efficiency (%)		71.8
Perfect OpenMP + MPI + Pthread		1.00
Perfect OpenMP + MPI + Pthread + Perfect Load Distribution		1.00
No Scalar Integer	Potential Speedup	1.19
	Nb Loops to get 80%	10
FP Vectorised	Potential Speedup	1.27
	Nb Loops to get 80%	14
Fully Vectorised	Potential Speedup	2.73
	Nb Loops to get 80%	41
Data In L1 Cache	Potential Speedup	1.07
	Nb Loops to get 80%	5
FP Arithmetic Only	Potential Speedup	1.65
	Nb Loops to get 80%	28

Figure 2: Advanced MAQAO profiling of CHAMP

Source Location	Source Function	Level	Coverage run_0 (%)	Vectorization Ratio (%)	Vectorization Efficiency (%)
- multideterminant.f:249-254	compute_ymat	Single	6.17	0	12.5
- orbitals.f:148-153	orbitals	Innermost	5.76	100	50
- nonloc.f:435-436	nonloc	Innermost	3.49	100	50
- orbitals.f:371-375	orbitalse	Innermost	3.28	100	50
- multideterminante.f:96-99	multideterminante	Single	2.58	0	12.5
-	__powr8i4	Single	2.37	NA	NA
- multideterminante.f:66-67	multideterminante	Innermost	1.63	33.33	16.67
- multideterminant.f:264-266	compute_ymat	Innermost	1.63	0	12.5
- multideterminante.f:90-90	multideterminante	Innermost	1.27	33.33	16.67
- determinante_psit.f:25-32	determinante_psit	Single	0.88	100	50
- basis_fns.f:86-86	basis_fns	Innermost	0.86	33.33	16.67
- multideterminante.f:194-19	multideterminante_grad	Innermost	0.81	33.33	16.67
- multideterminante.f:84-87	multideterminante	Innermost	0.79	0	12.5
- determinante_psit.f:25-32	determinante_psit	Single	0.79	100	50
- multideterminante.f:95-95	multideterminante	Single	0.76	100	50

Figure 3: MAQAO vectorization metrics for CHAMP.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Scaling analysis

Most of the performance-critical regions are embarrassingly parallel and do not involve communication.

The single-node weak scaling plot shown in Figure 4 compares weak scaling of two code versions: Previous in red, Current in Blue being the latest code variant. First by comparing globally blue bars versus red bars, performance gains close to 2X are observable. Second, focusing on the most recent code version, execution time remains constant up to 32 cores showing an ideal behavior: in weak scaling execution should ideally remain constant. Beyond 64 cores and all of the way to 128, there is some performance loss culminating at 20% when all of the cores within the node are used. This simply reflects that use of all of the cores within a node generates some contention/saturation which is to be expected.

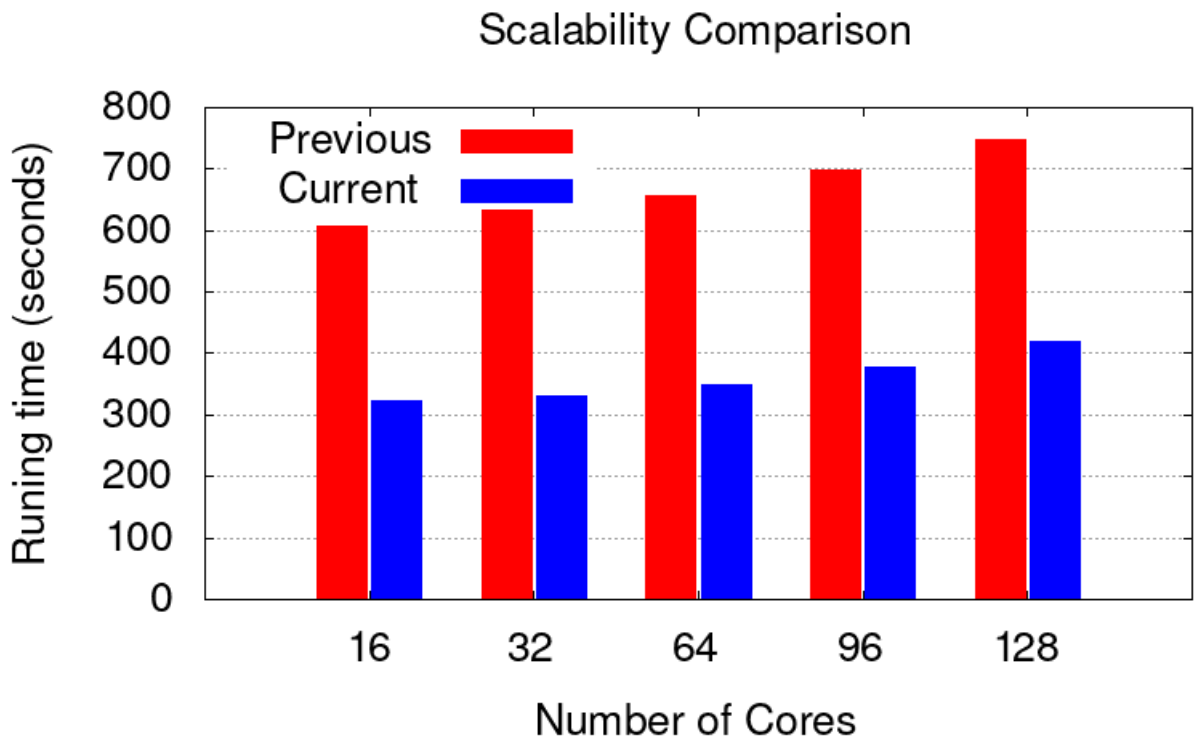

Figure 4: Weak scaling of two CHAMP versions : measurements on AMD Rome 7H12(2x) 64 Codes/Socket. The test calculation is energy only Butadiene 15000 determinants.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

GammCor

Application and workload description

GammCor is an application for calculating interaction energies using a method based on perturbation theory [5]. It is designed to interface with other *ab initio* quantum chemistry applications like Dalton and in the future other TREX applications. During the project, a new variant of the application has been developed that makes use of Cholesky decomposition. This version of the code[10], which has not been published yet, is used here. Here the branch `IntCholesky` is used. Key properties of the current code are summarized in Table 6.

The following analysis is based on workloads concerning the computation of the interaction energy between an ethylene and an argon atom. The (larger) input deck `LONG_CD` has been made available by the application developers internally.

Table 6: Selected properties of the GammCor code.

Programming language	Fortran 77 / Fortran 90
Process-level parallelism	–
Thread-level parallelism	OpenMP
SIMD parallelism	Compiler auto-vectorization
GPU Acceleration	–
Parallel libraries	–

Profiles for the single-thread case

Furthermore, a single-thread analysis has been performed using the workload `LONG_CD` using the HPC server at UVSQ with an Intel Skylake processor.[11] The code has been compiled using the Intel Fortran compiler version 2021.5.0 with `-O3` compiler optimizations. The total execution time is $t_{\text{total}} = 4540\,\text{s}$. Table 7 shows that most of the computations are happening in BLAS3 (more precisely DGEMM: matrix multiply operations) and (to a significantly lesser extent) in BLAS2 functions. Dense matrix multiplies (provided that matrix sizes are not too small) are top performers on most recent CPUs making an excellent use of vector units and memory hierarchy. Therefore, automatically GAMMCOR performance is benefiting from the use of these routines.

[10] https://gitlab.com/qchem/gammcor.git

[11] https://datafront.maqao.exascale-computing.eu/public_html/oneview2020/GAMMCOR/IntCholesky90c8a0/LONG-CD/skl/ov3/GAMMCOR_IntCholesky90c8a0_LONGCD_skl_o1_m1_c1_ov3/

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

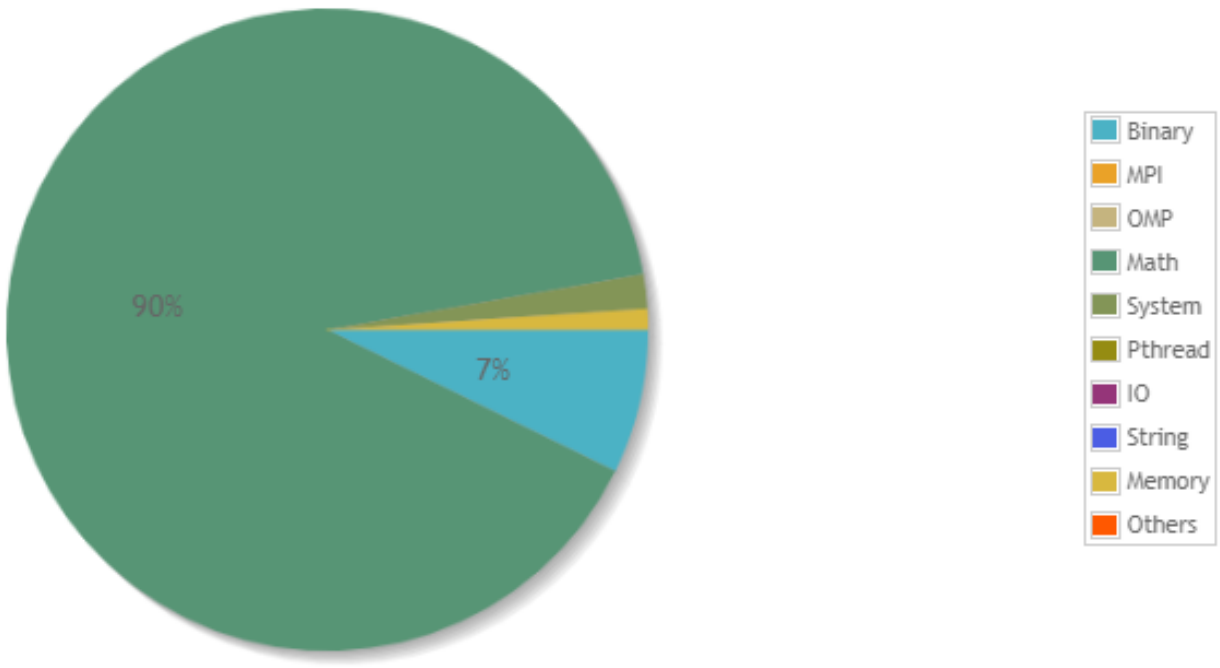

Figure 5: Time distribution of GammCor

Table 7: Single-thread timing results for the GammCor `LONG_CD` workload were collected using MAQAO.

Function name	t/t_{total}
`mkl_blas_avx512_dgemm_kernel_0`	74.5 %
`mkl_blas_avx512_dgemv_t_intrinsics`	6.2 %
`get_tr_aoreaderchol`	2.5 %

Profiles for the multi-threaded case

A multi-thread profile has been generated on Dardel. The code has been compiled using GNU Fortran 11.2 with -O3 compiler optimizations. The results are shown in Table 8. The potential for scaling in GammCor is inherently limited by the nature of the dense matrix operations it performs. However, before the scaling limit is reached, the time spent in I/O becomes dominating. For this workload about 28 GByte of data is read, which means that I/O happens with an effective bandwidth of 70 MByte/s. Micro-benchmark results on the same system (see IOR in appendix A) indicates that a much higher bandwidth can be achieved. The performance difference is likely due to the small transfer size and lack of concurrency of I/O operations. One possibility to address the I/O bottleneck would be to use MPI based file reading. This would be a logical step as well since there

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

are numerous BLAS libraries with MPI bindings, meaning that the code would thus benefit both from shorter startup times and from the ability to run on multiple nodes.

Table 8: HPCToolkit trace timings for the GammCor `LONG_CD` workload.

Number of threads	Time spent in I/O	Total execution time
1 thread	499 seconds	5220 seconds
8 threads	424 seconds	1157 seconds
32 threads	388 seconds	816 seconds
64 threads	403 seconds	868 seconds

Performance analysis

Only a small fraction, about 5-6 %, of the computations are performed outside of mathematical libraries as can be seen from Table 9, which documents key MAQAO global metrics (for definition see 3). The large value of the *Array Access Efficiency* metric indicates that most of the data accesses are with stride 1, i.e. the data layout is chosen well for exploiting spatial data localities. Finally, the metric *Fully vectorized* indicates rather limited potential for performance improvements through vectorization essentially due to the fact less than 5% of execution time is spent in loops. . Details on the vectorization metrics generated by MAQAO can be found in Figure 6.

Figure 7 displays the performance evolution (unicore and multicore) across the different code versions. All in all, the performance gain is 3X.

Loop id	Source Location	Source Function	Level	Coverage run_0 (%)	Vectorization Ratio (%)	Vectorization Efficiency (%)
4038	gammcor - sorter_Cholesky.f90:359-484 [...]	get_tr_aoreaderchol	Innermost	2.47	9.89	11.22
455	gammcor - initia.f:1295-1419 [...]	Idinteg	Innermost	0.98	11.11	12.5
16316	gammcor - Cholesky.f90:680-681	chol_rk	Innermost	0.57	100	100
24093	gammcor -	__intel_avx_rep_memcpy	Single	0.26	100	50
5373	gammcor - abfofo.f90:2376-2390	jk_loop	InBetween	0.14	84.34	82.19

Figure 6: Vectorization of GammCor

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 9: MAQAO global metrics for the GammCor `LONG_CD` workload.

Time in analysed loops	5.5 %
Time in analysed innermost loops	4.9 %
Perfect Flow Complexity	1.02
Array Access Efficiency	75.3 %
Fully vectorized: Potential speed-up	1.04
Fully vectorized: number of loops	2

Version	1	2	4	8	16	24	36
0	— 01:38:16	— 01:28:14	— 01:23:33	— 01:19:48	— 01:28:42	— 01:43:47	— 01:34:40
1b	▼ 00:58:48	▼ 00:49:50	▼ 00:42:34	▼ 00:41:23	▼ 00:41:35	▼ 00:55:00	▼ 00:56:14
2b	— 00:58:56	— 00:49:24	— 00:42:17	— 00:41:08	— 00:41:15	— 00:55:30	▲ 00:57:59
3	— 00:59:16	▼ 00:47:34	— 00:41:51	▼ 00:38:22	▼ 00:39:44	▼ 00:49:00	▼ 00:51:00
4	— 00:59:28	▼ 00:42:59	▼ 00:33:57	▼ 00:29:50	▼ 00:31:46	▼ 00:44:37	▼ 00:45:24
5	▼ 00:55:39	▲ 00:44:09	▲ 00:38:38	▲ 00:34:45	▲ 00:36:43	▲ 00:47:36	▲ 00:50:02

Figure 7: Evolution of the optimization of GammCor

NECI

Application and workload description

NECI implements the Full Configuration Interaction quantum Monte Carlo (FCIQMC) algorithm [6]. Key properties of the current code are summarized in Table 10.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 10: Selected properties of the NECI code.

Programming language	Fortran 2003
Process-level parallelism	MPI
Thread-level parallelism	–
SIMD parallelism	Compiler auto-vectorization
GPU Acceleration	–
Parallel libraries	LAPACK, Parallel HDF5, FFTW

Single-thread vectorisation analysis

For the analysis described in this subsection, a 10-electron workload `Neatom-pchb` was used.

The analysis was performed using the HPC server at UVSQ with an Intel Skylake processor.[12]. The code has been compiled using the Intel Fortran compiler version 2021.5.0 with `-O3` compiler optimizations. The total execution time is $t_{\text{total}} = 301.06\,\text{s}$.

Table 11: MAQAO global metrics for NECI.

Time in analysed loops	38.9 %
Time in analyzed innermost loops	21.5 %
Perfect Flow Complexity	1.06
Array Access Efficiency	76.6 %
Fully vectorized: Potential speed-up	1.47
Fully vectorized: Number of loops	34

The MAQAO analysis shows that most of the time (over 97%) is spent in the binary. A very limited amount of time is spent in innermost loops (21 % for Neatom-pchb workload), as shown in Table 11. Similarly, the time spent in analysed loops (39% for Neatom-pchb) is quite low, limiting the payoff of standard loop optimization. For the Neatom-pchb most of the time is actually spent in various functions with 40% of time spent in 10 functions and 80% in 45 functions.

The performance indicators of MAQAO are well adapted to applications that are expected to reach high flops/s rates. However, the FCIQMC algorithm is such that it is dominated by binary operations

[12] https://datafront.maqao.exascale-computing.eu/private_html/oneview2020/NECI/base/Neatom-pchb/skl/ov3/NECI_Neatom-pchb_skl_o1_m1_c1_ov3_release_togglelprof-profilenone/

on integers and indirect memory access, generating by nature very low flops/s rates. Vectorization can't be leveraged, but the algorithm can take advantage of large-scale parallelism.

Single-node profiles

The NECI executable has been used with the input deck `testset_B/c2_cas`[13] with 100 iterations.

For the runs on Galileo100 we used the following toolchain: GCC 10.2.0, OpenMPI 4.1.1, MKL 2021, and FFTW 3.3.9.

Table 12 shows results obtained on 4 cores of a Galileo100 node listing the routines that based on HPCToolkit profiles account for about 70 % of the cycles.

Table 12: Execution time breakdown for NECI on 4 Galileo100 cores using HPCToolkit.

Subroutine/function	File	$10^{10}\, N_{\text{cycle}}$	Fraction
`subroutine gen_exc`	`pchb_excitgen.F90`	10.1	36.05 %
`function dyn_sltcnd_excit_old`	`sltcnd.fpp`	2.7	9.40 %
`pure function FindBitExcitLevel`	`DetBitOps.F90`	1.6	5.70 %
`subroutine extract_bit_rep_avsign_no_rdm`	`rdm_general.fpp`	1.5	5.47 %
`subroutine walker_death`	`fcimc_helper.F90`	1.4	5.11 %
`subroutine ReturnAlphaOpenDet`	`HPHFRandExcit.F90`	1.1	3.79 %
`function dyn_sltcnd_excit`	`sltcnd.fpp`	1.1	3.89 %

Scaling analysis

The weak scaling behaviour of NECI depends due to possible load imbalance effects on the number of walkers used per MPI rank. Figure 8 shows the parallel efficiency, which is defined as $t(1)/\left(n \cdot t(n)\right)$ with n being the number of MPI ranks, for different number of walkers defined in the input file. The large variability of the data is due to the implicit weak scaling behavior of the algorithm. First, the code is run on a small number or nodes and the number of basis functions is gradually increased. Eventually, the number of basis functions saturates the nodes and the job is restarted on a larger number of nodes. The number of walkers is managed automatically by the code. With other words, the amount of work is not strictly constant when changing the number of walkers in the input file. Nevertheless, we consider this as an indication that setting the number of walkers to $\gtrsim 10,000$ is sufficient to obtain a reasonable parallel efficiency.

A full weak scaling analysis has been performed on the MPG-FKF Cluster (See Section 4) using $P = 1, \ldots 640$ MPI ranks. For this analysis, 10,000 walkers are used per processor and the initial

[13] https://gitlab.jsc.fz-juelich.de/trex/neci-performance-tests/-/tree/master/testset_B/c2_cas

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

phase was discarded to avoid the effects discussed earlier. An all-electron CO_2 molecule with cc-pVTZ basis had been selected as workload. The timing results have been obtained using NECI's internal timing routines.

In order to take into account that for the different runs the number of updated walkers is fluctuating, we do not consider the execution time but rather the update rate b_{walker} at which the walkers are updated. We defined this rate as follows:

$$b_{\text{walker}}(P) = \frac{\overline{N}_{\text{walker}}(P) \cdot N_{\text{step}}}{t(P)}\,, \tag{1}$$

where $\overline{N}_{\text{walker}}(P)$ is the average number of walkers that have been updated during the run with P MPI ranks.

The weak scaling can now be explored by considering the ratio

$$\frac{b_{\text{walker}}(P)}{b_{\text{walker}}(P_0)}\,. \tag{2}$$

While typically a single core ($P_0 = 1$) is used as the reference, we use instead a single compute node ($P_0 = 20$). The ratio can now be interpreted as a node weak scaling. The results are shown in Figure 9.

The weak scaling parallel efficiency can now be defined in the following way:

$$\epsilon(P_0, P) = \frac{P_0 \cdot b_{\text{walker}}(P)}{P \cdot b_{\text{walker}}(P_0)}\,. \tag{3}$$

We observe for the given workload and system a weak parallel efficiency $\epsilon(20, 640) = 86\,\%$.

The deviation from ideal scaling is due to the increasing amount of time, which is spent in communication, more specifically in MPI collectives. This can be seen from Figure 10, which plots the time spent per iteration in communication as a function of the number of MPI ranks P. We attribute the non-monotonic behaviour to effects of the network topology, which would need further investigation, e.g. using synthetic benchmarks. The communication time has been observed to significantly increase for $P > 640$, which is possibly due to a bottleneck in the network of the used system rather than the application.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

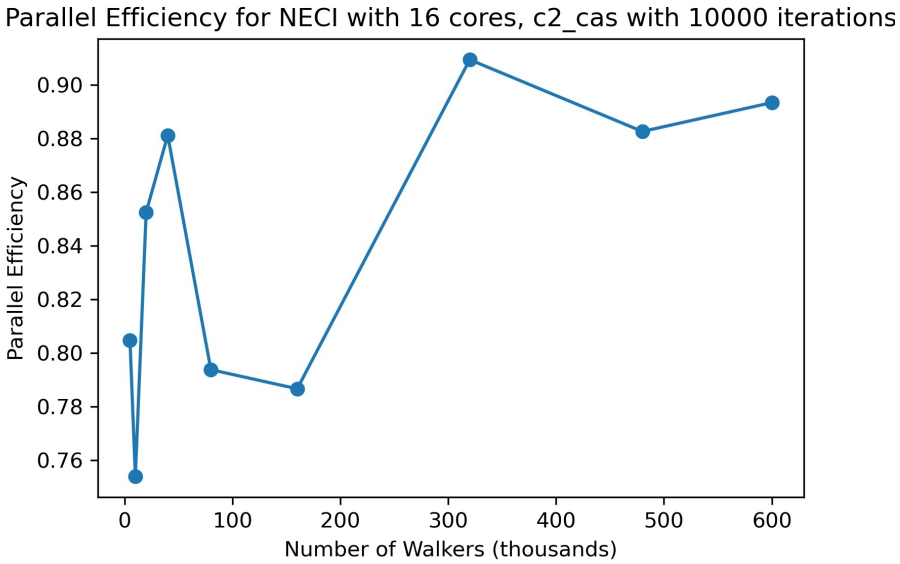

Figure 8: NECI parallel efficiency on Galileo100 using $P = 16$ MPI ranks as a function of the number of walkers defined in the input file.

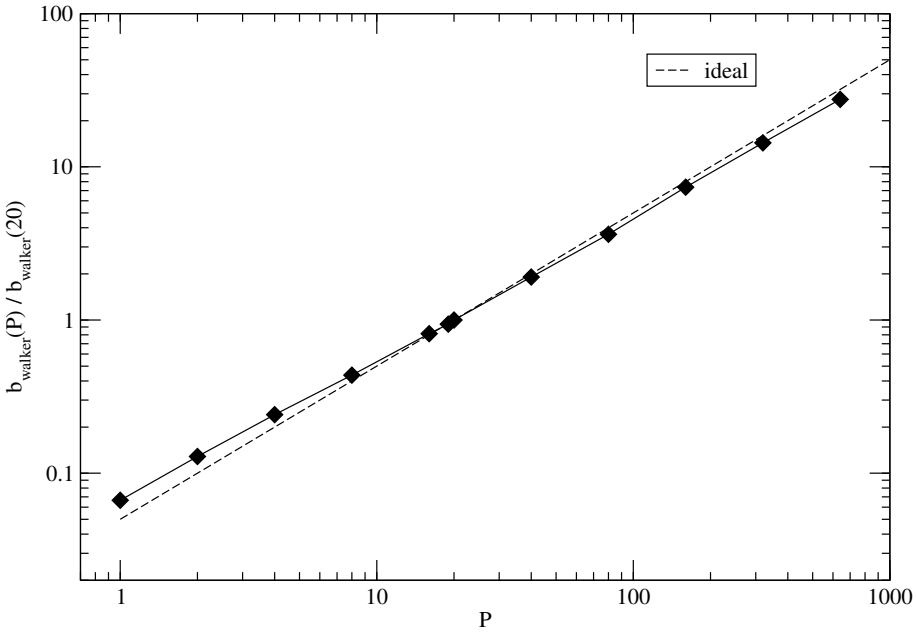

Figure 9: Weak scaling of the update rate of NECI for $P_0 = 20$.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

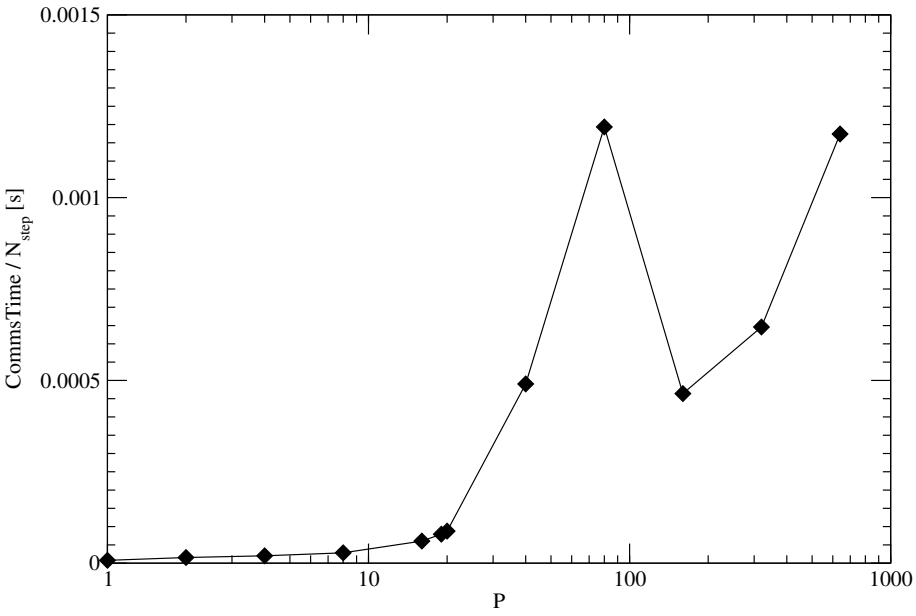

Figure 10: Communication time as measured with the CommsTime counter of NECI.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

QMC=Chem

Application and workload description

QMC=Chem[14] is a Quantum Monte Carlo application that has been developed for extreme-scale scalability based on a master-worker approach [7]. Key properties of the current code are summarized in Table 13.

Table 13: Selected properties of the QMC=Chem code.

Programming language	Fortran 77/90 (based on IRPF90 [8]), OCaml
Process-level parallelism	ZeroMQ
Thread-level parallelism	–
SIMD parallelism	Compiler auto-vectorization + compiler directives
GPU Acceleration	–
Parallel libraries	BLAS/LAPACK

Single-node profiles

For the single-node profile, the Benzene workload has been used.[15] In the following we report on an analysis using the HPC server at UVSQ using MAQAO. The Intel Fortran compiler version 19.0.1.144 was used with `-O2` compiler optimizations. The workload was executed with 1 master and 6 worker threads with a total execution time $t_{\text{total}} = 54.47\,\text{s}$. The most recent version of the code has been used.

Figure 11 shows that most of the computations are performed by the application code and only 4 % of the time is spent in mathematical libraries. A more detailed view is given in Table 14, which shows that the library calls are mainly for a BLAS2 operation. Table 15 shows that 71.3 % of the execution time is spent in loops and 52.4 % in innermost loops. The MAQAO metric `Perfect Flow Complexity` suggests some potential for improving performance by optimising the control flow within the innermost loops. The metric `Array Access Efficiency` indicates that most of the data access in the innermost loop is processor friendly, i.e. stride-1 access. Finally, the metric *Fully vectorized* suggests a rather big potential for improving performance by means of vectorization. However, to achieve 80 % of this performance gain, MAQAO estimates that 21 loops need to be vectorized.

The specifics of the vectorization potential are shown for 5 loops with a higher percentage of execution time in Table 16. The metric `Vectorization Ratio` shows that currently some loops are partially or not vectorized. The metric `Vectorization Efficiency` indicates that for a number

[14] https://gitlab.com/scemama/qmcchem

[15] https://gitlab.jsc.fz-juelich.de/trex/workload_qmcchem/-/blob/master/workloads/Benzene.tar.gz

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

of loops the compiler did not manage to leverage the AVX512 instruction set. This problem was investigated in depth, and we have observed that the compiler was being too conservative generating AVX2 instructions instead of AVX512 with masking. Rewriting the loops differently to enable AVX512 vectorization did not exhibit any acceleration, because of the low trip count of the loops.

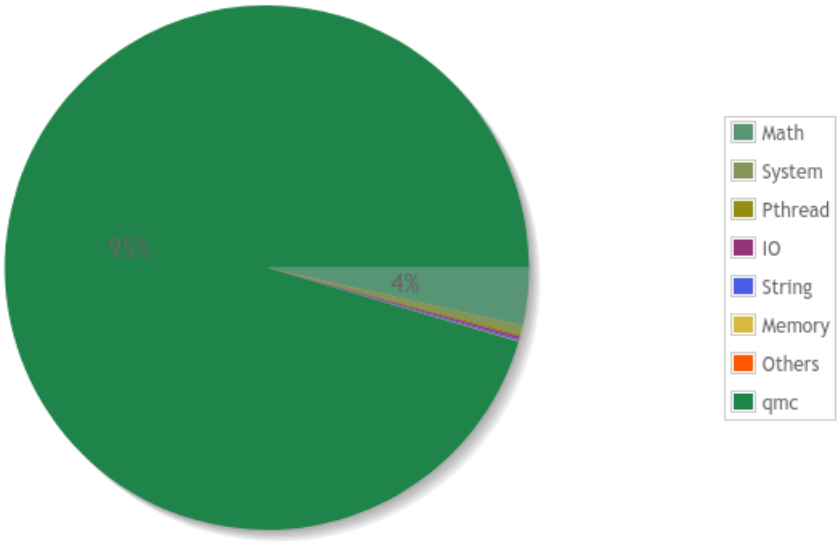

Figure 11: Categorization of QMC=Chem single-node execution time based on MAQAO measurements.

Table 14: Break-down of the execution time of QMC=Chem based on MAQAO measurements.

Function name	t/t_{total}
`det_update21`	33.81 %
`bld_det_beta_grad_lapl_curr`	17.59 %
`bld_det_alpha_grad_lapl_curr`	14.85 %
`sparse_full_mv`	4.11 %
`bld_psidet_value`	3.5 %
`bld_det_beta_value_curr`	2.85 %
`bld_det_alpha_value_curr`	2.38 %
`mkl_blas_avx512_dgemv_n_intrinsics`	2.17 %

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 15: MAQAO global metrics for QMC=Chem.

Time in analysed loops	71.3 %
Time in analyzed innermost loops	52.4 %
Perfect Flow Complexity	1.06
Array Access Efficiency	78.2 %
Fully vectorized: Potential speed-up	2.01
Fully vectorized: Number of loops	21

Table 16: MAQAO vectorization metrics for QMC=Chem.

Function name	$t_{\text{loop}}/t_{\text{total}}$	Vectorization Ratio	Vectorization Efficiency
`bld_det_beta_grad_lapl_curr`	10.78	66.7 %	37.5 %
`det_update21`	9.91	100 %	50 %
`bld_det_alpha_grad_lapl_curr`	9.44	66.7 %	37.5 %
`det_update21`	5.83	100 %	50 %
`bld_psidet_value`	3.49	0 %	12.3 %

Scaling analysis

The scalability of QMC=Chem has already been demonstrated in 2012 using up to 80,000 cores of the Curie machine [7]. To re-assess the scalability of the application, runs have been performed on the Joliot-Curie machine (for details see Section 4). The benchmarks were run for a wave function of the benzene molecule with 60351 Slater determinants in the cc-pVDZ basis set with BFD effective core potentials. Weak scaling benchmarks were run for 5 minutes each. In the following, the number of Monte Carlo samples used in the computation of the energy is reported either as a function of the number of cores or as a function of the number of full nodes (with 128 cores).

For evaluating the weak scaling we consider the throughput of work $b_{\text{sample}}(n)$, which we define as

$$b_{\text{sample}}(n) = \frac{N_{\text{sample}}(n)}{t}\,, \tag{4}$$

for fixed $t = 5$ minutes and variable amount of compute resources, which are parameterized by n. For the single-node scaling we use $n = N_{\text{core}}$ and for the multi-node case $n = N_{\text{node}}$. In Table 17

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

and in Figure 12 we show the ratio

$$\frac{b_{\text{sample}}(N_{\text{core}})}{b_{\text{sample}}(1)} \tag{5}$$

as a function of the number of cores N_{core}. In the case of perfect weak scaling, we expect this ratio to scale linearly with the number of cores. Similarly, Table 18 and Figure 13 show the scaling as a function of the number of nodes N_{node}.

To evaluate the weak scaling parallel efficiency ϵ, we calculate

$$\epsilon(n) = \frac{b_{\text{sample}}(n)}{n \cdot b_{\text{sample}}(1)} \tag{6}$$

where n is the number of cores (for the single node case) or the number of nodes (for the multi-node case). In case of single node, a drop to $\epsilon = 83\,\%$ is observed for $N_{\text{core}} = 128\,\text{cores}$ (see Table 17), while in the case of multi-node scaling a drop to $\epsilon = 72\,\%$ for $N_{\text{node}} = 256$ (see Table 18). This deviation from ideal scaling is due to the rather short runs as only initialization and termination affects the scaling, all intermediate communications being non-blocking. For real workloads, which run for longer periods of time, a significantly better scaling is expected.

Table 17: Single-node performance of QMC=Chem.

N_{core}	N_{sample}	$N_{\text{sample}}(N_{\text{core}})/N_{\text{sample}}(1)$	$N_{\text{sample}}(N_{\text{core}})/(N_{\text{core}} \cdot N_{\text{sample}}(1))$
1	85572	1.00	1.00
2	167077	1.95	0.98
4	333347	3.90	0.97
8	669300	7.82	0.98
16	1311330	15.32	0.96
32	2524136	29.50	0.92
64	4584338	53.57	0.84
128	9101106	106.36	0.83

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

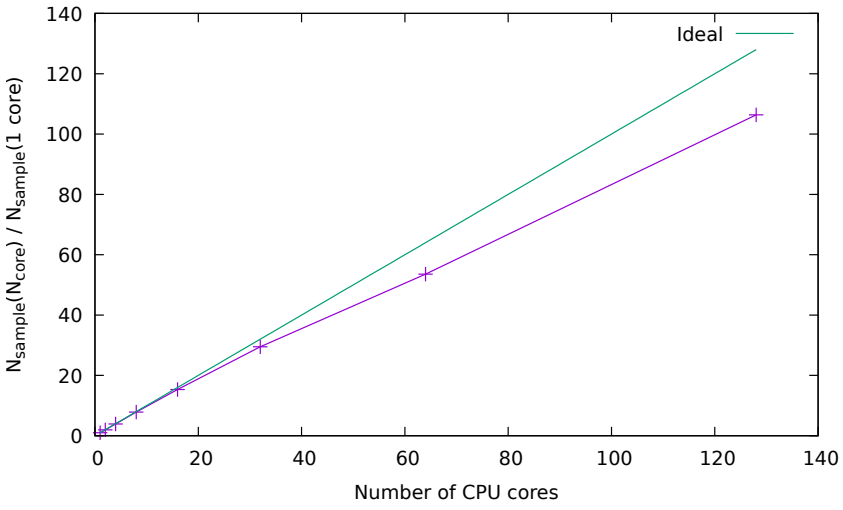

Figure 12: QMC=Chem single node scaling.

Table 18: Multi-node performance of QMC=Chem.

N_{node}	N_{core}	N_{sample}	$N_{\text{sample}}(N_{\text{node}})/N_{\text{sample}}(1)$	$N_{\text{sample}}(N_{\text{node}})/(N_{\text{node}} \cdot N_{\text{sample}}(1))$
1	128	9101106	1.00	1.00
2	256	18054405	1.98	0.99
4	512	35446069	3.89	0.97
8	1024	71119268	7.81	0.98
16	2048	143016081	15.71	0.98
32	4096	274720772	30.19	0.94
64	8192	485091764	53.30	0.83
128	16384	910805452	100.08	0.78
256	32768	1686532542	185.31	0.72

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

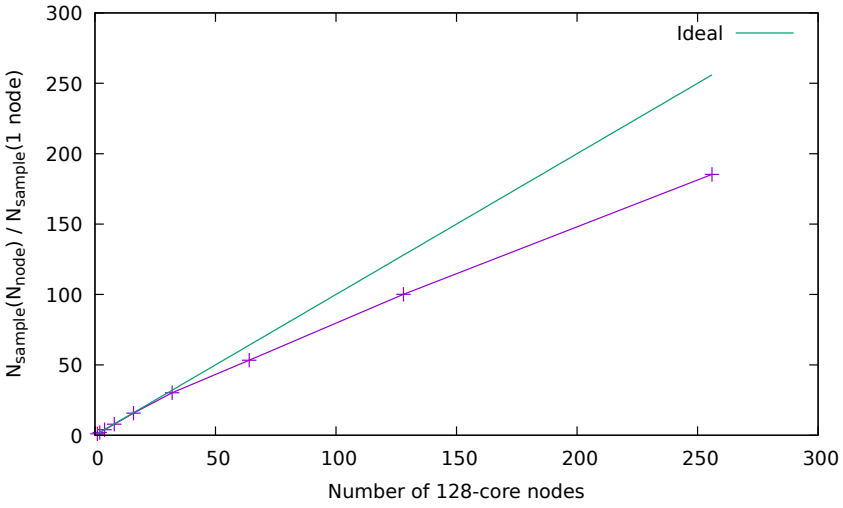

Figure 13: QMC=Chem scaling on multiple nodes with 128 cores per node.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Quantum Package

Application and workload description

Quantum Package 2.0 is an application that implements the selected configuration interaction (sCI) method based on the CIPSI (Configuration Interaction using a Perturbative Selection made Iteratively) algorithm [9]. Key properties of the current code are summarized in Table 19.

Table 19: Selected properties of the Quantum Package 2.0 code.

Programming language	Fortran 77/90 (based on IRPF90 [8]), OCaml
Process-level parallelism	MPI, ZeroMQ
Thread-level parallelism	OpenMP
SIMD parallelism	Compiler auto-vectorization
GPU Acceleration	–
Parallel libraries	BLAS/LAPACK

Single-node profiles

For a MAQAO single-node profile the HPC server at UVSQ was used for running a small workload that computes the total energy of a water molecule using the small basis set, 6-31G.[16] The workload was executed using 1 process and 12 threads with a total execution time $t_{\text{total}} = 363.38\,\text{s}$.[17]

Most of the computational time is used by the application code and only 6 % of the time is spent in libraries, as shown in Fig. 14. A more detailed view is given in Table 20, which shows that about 25 % of the execution time is spent in a sorting routine. In Table 21 selected MAQAO's global metrics are reported. The analysis shows that 70.5 % of the execution time outside the library is spent in loops, but only 22.8 % in innermost loops. The MAQAO metric `Perfect Flow Complexity` suggests some potential for improving performance by optimising the control flow within the innermost loops. The high value for the `Array Access Efficiency` indicates that most of the data access in the innermost loop is very processor friendly, i.e. stride-1 access. Finally, the metric *Fully vectorized* suggests a rather big potential for improving performance by means of vectorization. However, to achieve 80 % of this performance gain, MAQAO expects that a rather large number of 41 loops will be to be vectorized.

Table 22 reports the function and the corresponding loop where most of the execution time is spent. The vectorization potential of those loops is also reported in Table 22. Four of the five selected

[16] https://gitlab.jsc.fz-juelich.de/trex/workload_quantum_package/-/tree/master/workloads/small

[17] https://datafront.maqao.exascale-computing.eu/public_html/oneview2020/QUANTUMPACKAGE/base/h2o/skl/ov1/QUANTUMPACKAGE_base_h2o_skl_o1_m1_c1_ov1_fci/

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

loops are innermost. The metric `Vectorization Ratio` shows that currently these loops are not vectorized. Furthermore, the metric `Vectorization Efficiency` indicates that the compiler barely managed to leverage the AVX512 instruction set. The reason behind the metric values is that one of the loops is not innermost and most of them feature a complex control flow, which prevents vectorization.

Thanks to this analysis, the sorting routine has first been replaced by a routine from the Intel Performance Primitives (IPP) library. After this modification, the sorting has disappeared from the profile. Some external users complained that IPP was not readily available with their Intel compilers installation, so the developers of Quantum Package decided to remove the call to IPP and replace it with a binding to the `quicksort` function of the C standard library. This modification did not produce any performance issue, and has the advantage of being portable.

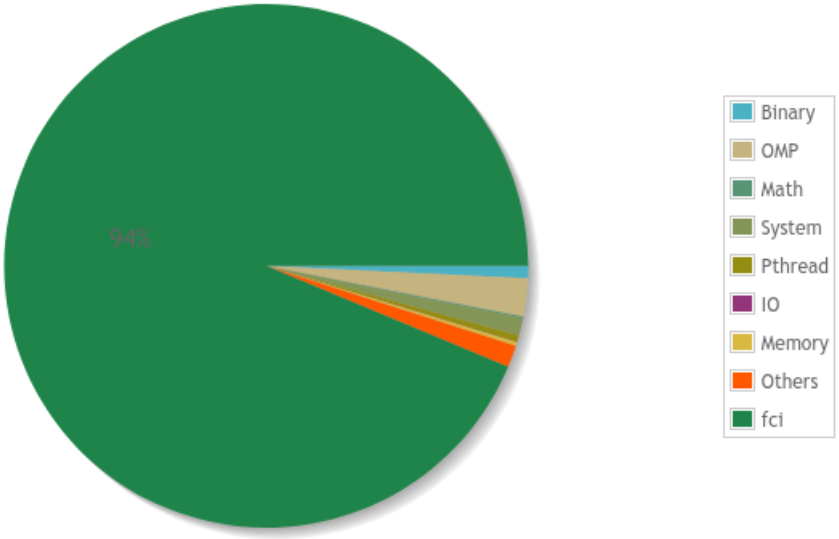

Figure 14: Categorization of Quantum Package single-node execution time based on MAQAO measurements.

Scaling analysis

The single-node benchmarks were run for a CIPSI calculation of the Benzene molecule, starting from a Hartree-Fock determinant to a wave function with up to 1.6 million Slater determinants in the cc-pVDZ basis set. In the CIPSI algorithm, as the size of the calculation increases with the simulation time the multi-node efficiency is low at the beginning of the simulation and increases with the size of the wave function. Therefore, the multi-node calculations were run as a restart from the single-node calculation making the wave function grow from 1.6 million to 26.7 million determinants. The benchmark was performed on the Joliot-Curie system (for details see Section 4).

The CIPSI algorithm is an iterative algorithm with a succession of steps. Each step is parallelized but synchronization is required between the different phases. Although some steps have a good parallel efficiency, the overall performance suffers from these synchronizations. A better scalability

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 20: Break-down of the execution time of Quantum Package based on MAQAO measurements.

Function name	t/t_{total}
`select_singles_and_doubles`	27.6 %
`rec_i_quicksort`	24.9 %
`get_d1`	7.3 %
`splash_pq`	6.7 %
`bitstring_to_list_in_selection`	4.5 %
`get_mo_two_e_integrals`	3.6 %
`get_mask_phase`	3.5 %

Table 21: MAQAO global metrics for Quantum Package.

Time in analysed loops	70.5 %
Time in analyzed innermost loops	22.8 %
Perfect Flow Complexity	1.06
Array Access Efficiency	86 %
Fully vectorized: Potential speed-up	2.86
Fully vectorized: Number of loops	41

is expected for very large workloads. For common workloads, the best compromise is around 20 compute nodes.

The speed-up factor is defined as $S(n) = t(1)/t(n)$ where $t(n)$ is the wall-clock time required using n cores or nodes. For measurements in this subsection the internal times of Quantum Package are being used. Table 23 and Fig. 15 show the single-node performance results as a function of the number of cores N_{core}. A significant drop in the parallel efficiency $\epsilon(N_{\text{core}}) = S(N_{\text{core}})/N_{\text{core}}$ to 30 % for $N_{\text{core}} = 128$ can be observed. A possible explanation is that successive forking and joining of threads with interleaved I/O operations cause scaling to break down. For larger problems, the time spent in the parallel sections will increase and thus the speed-up should improve. Moreover, the inclusion in a near future of the TREXIO library developed in WP2 is expected to improve dramatically the I/O performance, reducing the I/O penalty observed in this benchmark.

The multi-node scaling is documented in Table 24 and Fig. 16. In this second benchmark, the workload is larger than for the single-node scaling described before. The scaling is very good

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 22: MAQAO vectorization metrics for Quantum Package.

Function name	$t_{\text{loop}}/t_{\text{total}}$	Vectorization Ratio	Vectorization Efficiency
`rec_i_quicksort`	4.43	0 %	9.38 %
`rec_i_quicksort`	4.21	0 %	9.38 %
`get_mo_two_e_integrals`	2.71	1.0 %	8.57 %
`select_singles_and_doubles`	2.20	0 %	9.38 %
`bitstring_to_list_in_selection`	2.00	0 %	9.03 %

Table 23: Single-node performance of Quantum Package.

N_{core}	t (minutes)	$t(1)/t(N_{\text{core}})$	$t(1)/\left(N_{\text{core}} \cdot t(N_{\text{core}})\right)$
1	784.50	1.00	1.00
2	381.83	2.05	1.03
4	197.10	3.98	1.00
8	118.77	6.61	0.83
16	69.80	11.24	0.70
32	39.68	19.77	0.62
64	39.45	19.89	0.31
128	20.72	37.86	0.30

up to 8 nodes, but for a larger number of nodes scaling breaks down. As the size of the wave function increases along with the calculation, and as the speed-up improves with the size of the wave function, Quantum Package offers the possibility to dynamically increase the number of CPUs as the computation is running, such that the users can always stay in the domain in which the speed-up is satisfactory.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

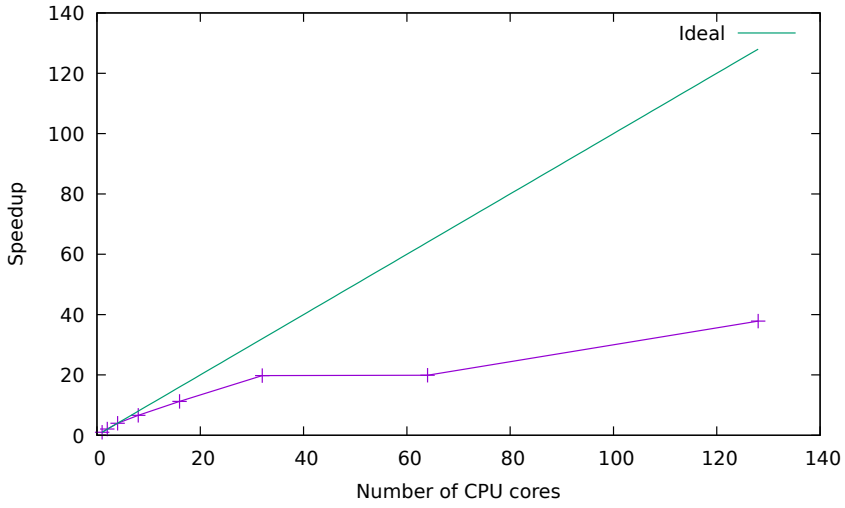

Figure 15: Quantum Package speed-up ratio $S(n) = t(1)/t(N_{\text{core}})$ as a function of the number of cores N_{core}.

Table 24: Multi-node performance of Quantum Package.

N_{node}	N_{core}	t (minute)	$t(1)/t(N_{\text{node}})$	$t(1)/\left(N_{\text{node}} \cdot t(N_{\text{node}})\right)$
1	128	2594.25	1.00	1.00
2	256	1351.17	1.92	0.96
4	512	688.63	3.77	0.94
8	1024	344.23	7.54	0.94
16	2048	231.88	11.19	0.70

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

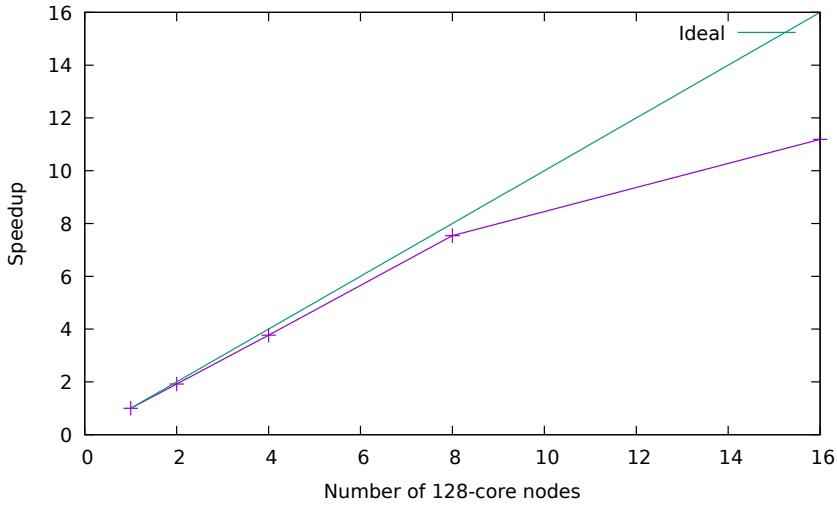

Figure 16: Quantum Package speed-up ratio $t(1)/t(N_{\mathrm{node}})$ as a function of the number of nodes N_{node}.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

TurboRVB

Application and workload description

TurboRVB is a computational package for QMC simulations of both molecular and bulk electronic systems using either Variational Monte Carlo (VMC) or diffusion Monte Carlo [10]. Key properties of the current code are summarized in Table 25. The code is available both as a CPU-only version as well as a branch supporting GPUs using OpenMP directives. In the following, both code versions will be considered separately.

Table 25: Selected properties of the TurboRVB code.

Programming language	Fortran 90
Process-level parallelism	MPI
Thread-level parallelism	OpenMP
SIMD parallelism	Compiler auto-vectorization
GPU Acceleration	OpenMP
Parallel libraries	–

CPU Version: Single-thread profiles

For a MAQAO single-node profile the HPC server at UVSQ was used for running a small workload consisting of 300 hydrogen atoms.[18] The Intel Fortran compiler version 2021.5.0 was used with -O3 compiler optimizations. The workload was executed using 1 process and 1 threads with a total execution time $t = 56.44\,\mathrm{s}$.

Fig. 17 shows that 66 % of the computations are performed in mathematical libraries. The breakdown in Table 26 shows that mainly BLAS2 and BLAS3 routines are called. In Table 27 selected MAQAO's global metrics are documented. It shows that only 25 % of the execution time outside the library is spent in loops with only half of that time in innermost loops. The MAQAO metric `Perfect Flow Complexity` suggests that most of the innermost loops have a perfect control flow, i.e. there are no branches and subroutine calls. The high value for the `Array Access Efficiency` indicates that most of the data access in the innermost loop is very processor friendly, i.e. stride-1 access.

The metric *Fully vectorized* suggests a limited potential for improving performance by means of vectorization. Exploiting this potential would require working on 24 loops. The specifics of the vectorization potential are shown in Table 28. The metric *Vectorization Ratio* indicates that for a few loops the percentage of the loop that was vectorized is large, while for some loops there is no vectorization. The metric *Vectorization Efficiency* with values under 50 clearly reveals that the

[18] https://datafront.maqao.exascale-computing.eu/public_html/oneview2020/TURBORVB/master6e1dab/fn/skl/ov1/TURBORVB_master6e1dab_fn_skl_o1_m1_c1_ov1/

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

compiler was not able to generate full AVX512 instructions. However, the performance benefits are expected to be limited as most of the time is spent in dense BLAS2 or BLAS3 operations.

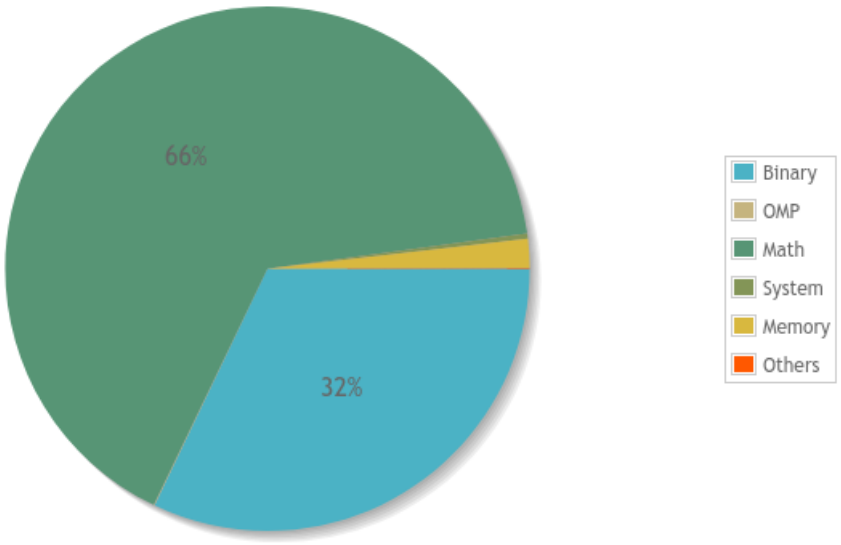

Figure 17: Categorization of TurboRVB single-node execution time based on MAQAO measurements.

Table 26: Break-down of the execution time of TurboRVB.

Subroutine	t/t_{total}
`mkl_blas_avx512_dgemv_t_intrinsics`	29.4 %
`makefun_grid`	13.1 %
`mkl_blas_avx512_zgemm_kernel_0`	10.5 %
`mkl_blas_avx512_zgemm_zcopy_right12_ea`	8.1 %
`__svml_sincos4_l9`	4.4 %
`mkl_blas_avx512_dgemm_kernel_nocopy_TN_b1`	2.9 %
`cvtas_a_to_t`	2.8 %
`zgemvtry`	2.8 %

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 27: MAQAO global metrics for TurboRVB.

Time in analysed loops	24.9 %
Time in analyzed innermost loops	14.6 %
Perfect Flow Complexity	1.01
Array Access Efficiency	76.9 %
Fully vectorized: Potential speed-up	1.25
Fully vectorized: Number of loops	24

Table 28: MAQAO vectorization metrics for TurboRVB.

Subroutine	$t_{loop}/t_{\text{total}}$	Vectorization Ratio	Vectorization Efficiency
`zgemvtry`	2.7 %	75 %	21.9 %
`makefun_grid`	1.4 %	0 %	12.5 %
`makefun_grid`	1.2 %	100 %	41.7 %
`cvtas_a_to_t`	1.0 %	0 %	9.2 %
`makefun_grid`	1.0 %	0 %	6.3 %
`makefun_grid`	0.8 %	100 %	37.5 %

GPU Version: Multi-node scaling

Weak scaling analysis was performed on the Marconi100 system using the CPU+GPU version for a system of 300 H atoms and a system of 64 P atoms for bot the variational and diffusional Monte Carlo cases. This is shown in 29.

GPU Version: Profiling results for Marconi100

The FNC workload with 300 electrons has been profiled using NVIDIA Nsight on Marconi100. The binary was generated using IBM's XL compiler. To keep the profiling data manageable, only a single MPI rank and 16 threads have been used, i.e. 2 threads per core. The application trace was manually annotated using NVTX markers. A small example is shown in Fig. 18.

The following observations can be made:

- The trace shows 85.9 s wall-time being spent in the main application loop, which is 49.9 % of

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 29: TurboRVB weak scaling on Marconi100.

Number of cores/walkers	300-H system VMC (s)	300-H system DMC (s)	64-P system VMC (s)	64-P system DMC (s)
32	31.4	85.4	24.0	95.7
64	31.5	85.4	24.6	98.3
128	34.7	85.1	23.9	97.6
256	31.8	85.8	24.2	97.9
512	34.0	85.7	24.4	105.7
1024	50.8	120.4	25.6	110.5
2048	49.5	123.5	28.6	104.7
4096	50.5	142.6	29.7	107.8
8192	39.1	107.8	27.2	94.3

Figure 18: Nsight profile for TurboRVB.

the total execution. The remaining time was due to the startup and finalization phases of the application and are not included in the analysis.

- The average GPU utilization during the main loop is 15.7 %, which is a rather low utilization.
- The GPU kernels and data movement operations are launched asynchronously but are typically followed by synchronization calls, which effectively renders all of these synchronous. Therefore, CPU and GPU computation, as well as data movement, is serialized, hence there is a limited amount of overlap.

The CPU sampling data indicates that the majority of the useful application time is spent in the routine `makefun_grid` and `uptabpip`. On average 46 % and 12 % of the time is spent there.

On the GPU side Table 30 summarizes the breakdown of compute kernels (with $> 1\,\%$ contribution relative to the total GPU usage) and wall-time used. The CPU to GPU `memcpy` operations are among

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

the most time-consuming tasks. They take overall 2.4 s and 1110872 invocations, most of which are not overlapped suggesting that this data movement has significant overhead. GPU to CPU copies add less overhead and account for only 0.4 s. Table 31 shows the breakdown of wall-time spent in CUDA API calls, which shows the effect of frequent CPU-GPU data movement and synchronization calls.

Table 30: TurboRVB timings obtained on Marconi100 using Nsight.

Kernel name	t/t_{total}	Total time [s]	Instances
volta_dgemm_64x64_tn	30.7 %	3.25	24055
gemv2T_kernel_val	28.5 %	3.02	24000
zgemm_largek	18.1 %	1.92	25500
__xl_zgemvtry_l1149_OL_122	5.7 %	0.61	24000
gemv2N_kernel	3.6 %	0.38	71864
gemv2T_kernel_val	2.3 %	0.25	48136
__xl_uptabpip_l61_OL_1	1.9 %	0.20	96000
__xl_uptabtot_l818_OL_3	1.0 %	0.10	24000
Other	7.9 %	0.87	–

GPU Version: Performance results for JUWELS Booster

One solution to address low GPU utilization is to use Nvidia's Multi-Process Service (MPS)[19]. This allows a larger number of MPI ranks to connect to the same device.[20] With multiple instances of the application running in parallel it becomes possible for the scheduler to fill gaps. A challenge of this approach is the increased memory capacity requirements on the GPUs, as MPS requires each MPI process to reserve memory on the GPU.

For TurboRVB this approach has been explored on a single node of the JUWELS Booster. The NVIDIA A100 GPUs in that system provide a sufficient memory capacity. In Table 32 the execution times are shown (with initialization is discarded) using different settings for a system of H atoms:

- Runs with a different number of electrons N.
- Runs with a different number of MPI ranks N_{rank}.
- Runs only on the CPUs as well as on CPUs plus GPUs.

[19] https://docs.nvidia.com/deploy/mps/index.html
[20] Since the Volta generation of GPUs, up to 48 clients per device are supported.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 31: TurboRVB timings for the CUDA API calls obtained on Marconi100 using Nsight.

Kernel name	t/t_{total}	Total time [s]
`cudaDeviceSynchronize`	24.2 %	9.7
`cuMemcpyHtoDAsync_v2`	22.0 %	8.8
`cuEventSynchronize`	17.0 %	6.8
`cuLaunchKernel`	12.6 %	5.0
`cudaLaunchKernel`	7.2 %	2.9
`cudaFree`	4.1 %	1.7
`cuMemcpyDtoHAsync_v2`	3.2 %	1.3
`cuEventRecord`	2.7 %	1.1
`cuEventCreate`	2.2 %	0.9
`cuEventDestroy_v2`	1.8 %	0.7
Other	3.0 %	1.9

- Runs with and without MPS.

The following observations can be made:

- The use of MPS has a significant performance impact. Using $N_{\text{rank}} = 96$ MPI ranks and $N = 1024$ electrons the execution time reduces by a factor $4\times$ when using MPS.
- Increasing the number of MPI ranks per node from 48 to 96 ranks still improves the performance.
- Speed-up compared to using only the available CPUs increases significantly for larger problem sizes, i.e. larger number of electrons.

Using MPS and a larger problem size with $N = 1024$ electrons the GPU-enabled version shows a very significant speed-up compared to the CPU-only version.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table 32: TurboRVB execution times (without initialization) on JUWELS Booster for a different number of electrons N using the application's timers.

	N	$N_{\text{rank}} = 48$	$N_{\text{rank}} = 96$
t_{CPU}	128	20.1	17.3
t_{GPU}		13.5	11.2
Speed-up		1.5	1.5
t_{CPU}	300	183.8	186.3
t_{GPU}		49.5	43.9
Speed-up		5.7	7.2
t_{CPU}	1024		3604.4
t_{GPU}			195.9
t_{GPU} (no MPS)			781.8
Speed-up			18.4

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

6 Summary and Conclusions

In this deliverable, all TREX codes have been executed on different high-end HPC systems. The performance analysis overall indicates an efficient use of the different processor architectures. The evaluation of various loops using the MAQAO tool suggests a potential for further improvements by means of vectorization.

At this point scalability of the codes has been explored more carefully only for a subset of the codes that can be expected to be highly scalable. Most notably, QMC=Chem is showing a reasonable parallel efficiency $\epsilon = 72\,\%$ when using 32,768 cores, with the reduction of parallel efficiency due to to the short run time. For some of the applications, efforts will be required to leverage communication hiding by means of asynchronicity.

TurboRVB is currently the only TREX application that can exploit the performance of GPUs. While the use of a high-end GPU using a single instance of the application shows relatively low device utilization, using multiple instances per GPU allows mitigating this problem. For larger problem sizes a speed-up of about $18\times$ has been observed on a single JUWELS Booster node.

For one of the applications, I/O speed is limiting overall application performance. By increasing the concurrency of I/O operations, better exploitation of the capabilities of the I/O subsystem can be expected.

In the future, it is planned that QMC=Chem and CHAMP will benefit from GPU acceleration via the inclusion of the QMCkl library developed within the CoE (see deliverable D3.2 for more details). In the specific context of QMC where the independent processes are loosely coupled, communication is not a bottleneck: it can be adjusted at will by the user. In this context, all the gains obtained by single-node optimization are directly transferred to the complete application. For applications designed to run on smaller scales (Quantum Package, NECI and GammCor), single-node optimization will also be extremely beneficial. Hence, for the second period of the project, we plan to focus on the single-node performance of all the TREX codes.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

References

[1] N. Tallent, J. Mellor-Crummey, L. Adhianto, M. Fagan, and M. Krentel, "HPCToolkit: performance tools for scientific computing," *Journal of Physics: Conference Series*, vol. 125, p. 012088, jul 2008. [Online]. Available: https://doi.org/10.1088/1742-6596/125/1/012088

[2] L. Djoudi, D. Barthou, P. Carribault, C. Lemuet, J.-T. Acquaviva, W. Jalby *et al.*, "Maqao: Modular assembler quality analyzer and optimizer for itanium 2," in *The 4th Workshop on EPIC architectures and compiler technology, San Jose*, vol. 200, no. 5. Citeseer, 2005.

[3] S. Koliaï, Z. Bendifallah, M. Tribalat, C. Valensi, J.-T. Acquaviva, and W. Jalby, "Quantifying performance bottleneck cost through differential analysis," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ser. ICS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 263–272. [Online]. Available: https://doi.org/10.1145/2464996.2465440

[4] C. Amovilli, C. Filippi, and F. M. Floris, "Quantum Monte Carlo formulation of volume polarization in dielectric continuum theory," *The Journal of Chemical Physics*, vol. 129, no. 24, p. 244106, 2008. [Online]. Available: https://doi.org/10.1063/1.3043804

[5] P. H. Kowalski, A. Krzemińska, K. Pernal, and E. Pastorczak, "Dispersion interactions between molecules in and out of equilibrium geometry: Visualization and analysis," *The Journal of Physical Chemistry A*, vol. 126, no. 7, pp. 1312–1319, Feb 2022.

[6] K. Guther, R. J. Anderson, N. S. Blunt, N. A. Bogdanov, D. Cleland, N. Dattani, W. Dobrautz, K. Ghanem, P. Jeszenszki, N. Liebermann, G. L. Manni, A. Y. Lozovoi, H. Luo, D. Ma, F. Merz, C. Overy, M. Rampp, P. K. Samanta, L. R. Schwarz, J. J. Shepherd, S. D. Smart, E. Vitale, O. Weser, G. H. Booth, and A. Alavi, "NECI: N-electron configuration interaction with an emphasis on state-of-the-art stochastic methods," *The Journal of Chemical Physics*, vol. 153, no. 3, p. 034107, 2020.

[7] A. Scemama, M. Caffarel, E. Oseret, and W. Jalby, "QMC=Chem: A Quantum Monte Carlo program for large-scale simulations in chemistry at the petascale level and beyond," in *High Performance Computing for Computational Science - VECPAR 2012*, M. Daydé, O. Marques, and K. Nakajima, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 118–127.

[8] A. Scemama, "IRPF90: a programming environment for high performance computing," *CoRR*, vol. abs/0909.5012, 2009. [Online]. Available: http://arxiv.org/abs/0909.5012

[9] Y. Garniron, T. Applencourt, K. Gasperich, A. Benali, A. Ferté, J. Paquier, B. Pradines, R. Assaraf, P. Reinhardt, J. Toulouse, P. Barbaresco, N. Renon, G. David, J.-P. Malrieu, M. Véril, M. Caffarel, P.-F. Loos, E. Giner, and A. Scemama, "Quantum Package 2.0: An open-source determinant-driven suite of programs," *Journal of Chemical Theory and Computation*, vol. 15, no. 6, pp. 3591–3609, Jun 2019. [Online]. Available: https://doi.org/10.1021/acs.jctc.9b00176

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

[10] K. Nakano, C. Attaccalite, M. Barborini, L. Capriotti, M. Casula, E. Coccia, M. Dagrada, C. Genovese, Y. Luo, G. Mazzola, A. Zen, and S. Sorella, "TurboRVB: A many-body toolkit for ab initio electronic simulations by Quantum Monte Carlo," *The Journal of Chemical Physics*, vol. 152, no. 20, p. 204121, 2020.

[11] H. Shan and J. Shalf, "Using IOR to analyze the I/O performance for HPC platforms," in *In: Cray User Group Conference (CUG'07*, 2007.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

A Micro-benchmark Results

Likwid Bench

Likwid Bench[21] is a micro-benchmarking framework. It allows determining a set of key CPU performance figures like throughput of floating-point operations or memory bandwidth.

In the following results from Likwid are documented that have been obtained on Dardel and Galileo100. On both platforms, Likwid version 5.2.1 was used and compiled with GCC. Table 33 shows the results obtained for Dardel. As Dardel has 1 CPU with 64 cores and 2 threads each, the results regarding the number of flops per operation (named "peakflops") was done with 1280 kB (20 kB per core). This is done so each vector can be retrieved from the cache. The peakflops benchmark was repeated multiple times using different SIMD instructions - SSE, AVX and AVX+FMA in double precision. Although Likwid has support for AVX512 benchmarks, EPYC 7742 does not have support for AVX512 instructions. For the load benchmark, a 4 GB vector size was used for those 128 threads, ensuring that the data would be totally retrieved from the memory.

Benchmark	Result
Peakflops (Scalar)	641 364.16 MFlops/s
Peakflops (SSE)	1 256 676.72 MFlops/s
Peakflops (AVX)	2 171 318.09 MFlops/s
Peakflops (AVX+FMA)	2 680 379.33 MFlops/s
Load (Memory)	161 491.14 MFlops/s

Table 33: Results for the Likwid benchmark on Dardel.

Galileo100 has 2 CPUs per node where each has 24 cores and 1 thread. Likwid was compiled with the Intel compilers (2021) and was set up to run with 960kB of vector array for the peakflops benchmark (20kB per core), and 2GB for the load benchmark. Results are displayed in Table 34.

Intel MPI Benchmark Suite

The Intel MPI Benchmarks (IMB)[22] have become a standard benchmark tool for assessing the performance of MPI on a given system. For the TREX applications, MPI collective operations like `MPI_Reduce` and `MPI_Bcast` are the most performance relevant and are therefore documented in this subsection. The used version for the benchmark was the 2021.3, and it was compiled with GCC and Cray MPI.

For Galileo, the IMB was compiled with the Intel OneAPI compilers (v2021.4), which also includes the MPI.

[21] https://github.com/RRZE-HPC/likwid/wiki/Likwid-Bench
[22] https://github.com/intel/mpi-benchmarks

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Benchmark	Result
Peakflops (Scalar)	291 110.48 MFlops/s
Peakflops (SSE)	583 541.85 MFlops/s
Peakflops (AVX)	966 787.31 MFlops/s
Peakflops (AVX+FMA)	1 928 375.45 MFlops/s
Peakflops (AVX512+FMA)	3 360 487.51 MFlops/s
Load (Memory)	240 120.40 MFlops/s

Table 34: Results for the Likwid benchmark on Galileo100.

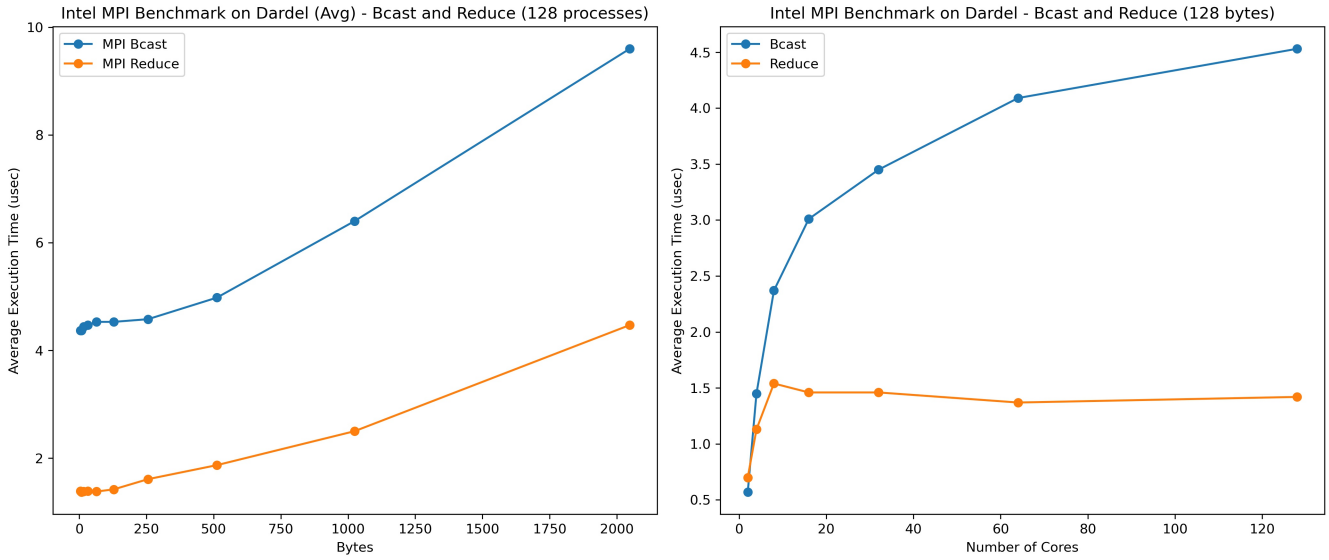

Figure 19: IMB scaling results for Dardel.

IOR

IOR [11] is a benchmark that was developed to analyse the I/O performance for HPC platforms. Figure 21 shows the time required to read a file of size 128 GiByte on Dardel as a function of the `blockSize`, which is the size of each read. It confirms that a significant performance improvement can be achieved by increasing the number of concurrent read operations.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

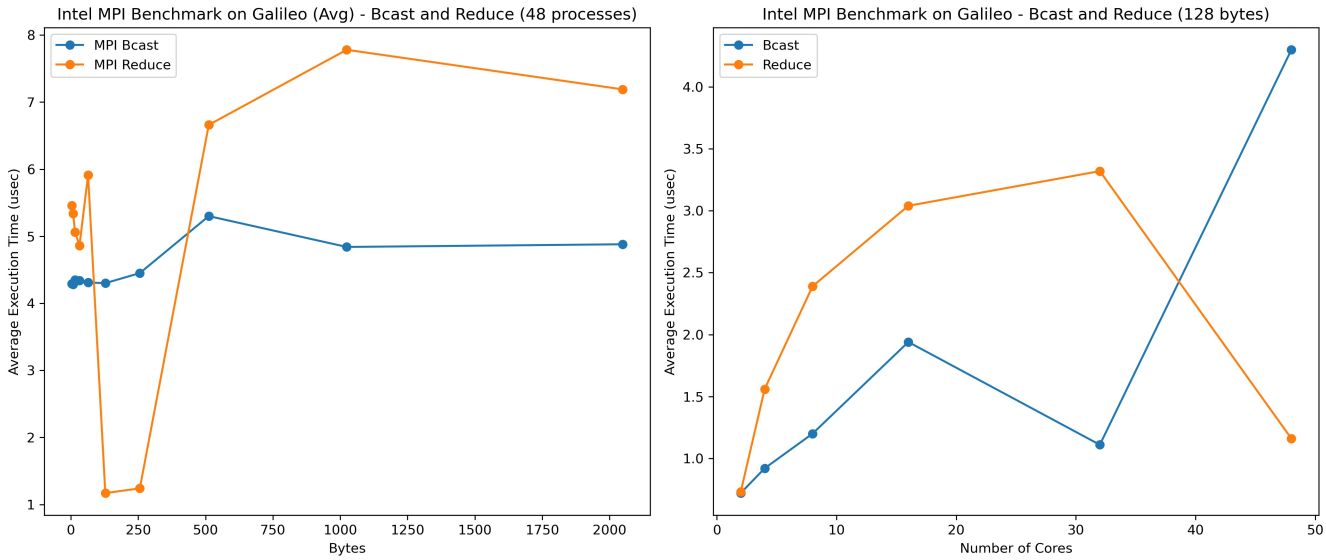

Figure 20: IMB scaling results for Galileo.

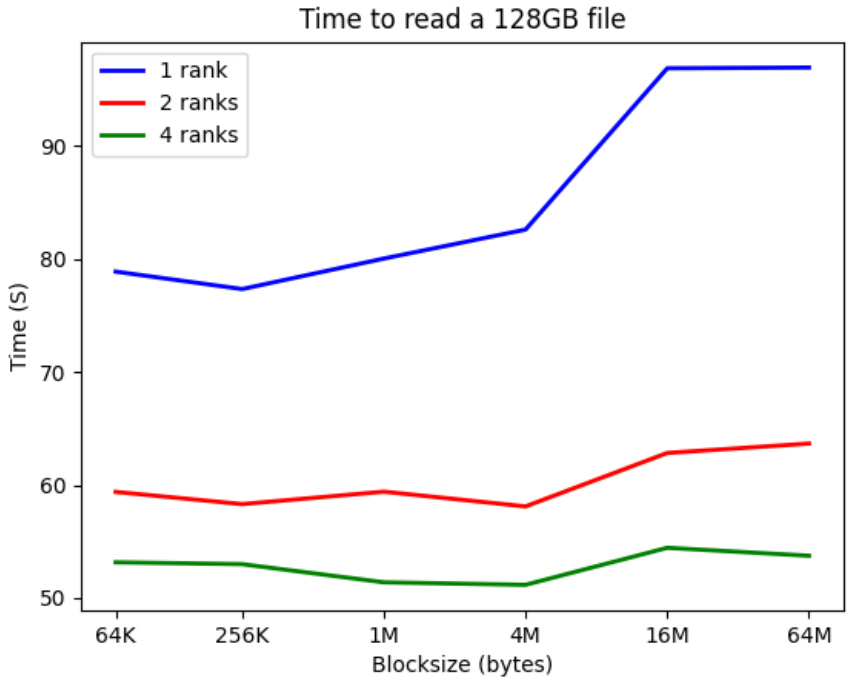

Figure 21: IOR sequential read performance on Dardel.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

B HPE Cray EX build environment

In this appendix will be highlighted some aspects on the build procedures of TREX applications on HPE Cray EX supercomputer architecture. A specification of the Dardel HPE Cray EX supercomputer at PDC can can be found in Section 4. An integrated part of the EX architecture is the Cray Programming Environment (CPE) which provides a consistent interface to multiple compilers and libraries. Parallel applications are built using compiler wrappers for C, C++ and Fortran. The compiler wrappers choose the required compiler version, target architecture options, and will automatically link to the scientific libraries, as well as the MPI and OpenSHMEM libraries.

A central component in managing the software stack on Dardel is provided by the EasyBuild framework [23]. A number of the dependency programs for TREX applications can be readily built using recipes contained in *easyconfig* files for a particular CPE toolchain. Recipes developed on one EX computer can be shared with users and staff of other HPC centers, providing a stable and rapid route for deploying code. To illustrate the workflow, the main steps [24] for building the NECI code are

```
# Build instructions NECI on Dardel

#Load the environment
ml PDC/21.11
ml PrgEnv-gnu/8.2.0
ml cray-fftw/3.3.8.12
ml CMake/3.21.2

#Create build directory
mkdir buildGnu
cd buildGnu

#Configure with Cmake
cmake ..

#make and install
make -j 32
```

This recipe was developed for Dardel, and can with minimal modification be used for building NECI on for instance LUMI. Similarly, the main steps of a build recipe for QP2 on Dardel are

```
# Build instructions for QP2 on Dardel

# Clone the QP2 git repository
git clone https://github.com/QuantumPackage/qp2
```

[23] https://easybuild.io/

[24] details on build recipes are contained in https://gitlab.jsc.fz-juelich.de/trex/build_env_pdc

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

```
cd qp2

# Load the environment for QP2 and dependencies
ml PDC/21.11
ml PrgEnv-gnu/8.2.0
ml Ninja/1.10.2-cpeGNU-21.11-python3
ml GMP/6.2.1-cpeGNU-21.11

# Run the configure script
./configure

# Install dependencies
# Use gcc 9.3.0 for the compilation of dependencies
ml gcc/9.3.0
./configure -i zeromq
./configure -i f77zmq
./configure -i ocaml
./configure -i docopt
./configure -i resultsFile

# Use gcc 11.2.0 for the compilation of QP2
ml gcc/11.2.0

# Set the quantum package environment variables
source quantum_package.rc

# Specify use of flags for Gfortran via the ftn compiler wrapper
./configure -c config/gfortran-ftn.cfg

# Compile and link QP2
make
```

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.