Exascale Quantum Mechanical Simulations: Navigating the Shifting Sands of Hardware and Software

Ravindra Shinde, Claudia Filippi, Anthony Scemama, William Jalby

Exascale Quantum Mechanical Simulations: Navigating the Shifting Sands of Hardware and Software

Ravindra Shinde[1*], Claudia Filippi[1], Anthony Scemama[2], William Jalby[3*]

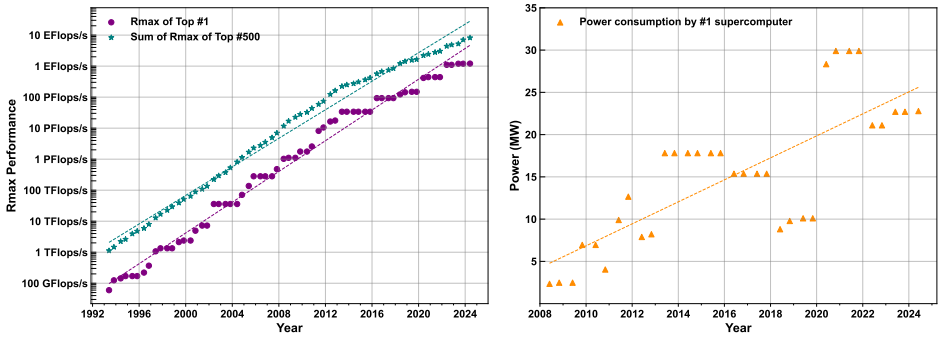
[1]MESA+ Institute for Nanotechnology, University of Twente, P.O. Box 217, Enschede, 7500 AE, Overijssel, The Netherlands.
[2]Laboratoire de Chimie et Physique Quantiques (LCPQ), Université de Toulouse (UPS) and CNRS, 118, route de Narbonne, Toulouse, 31062, France.
[3]Université Paris-Saclay, UVSQ, 9 Boulevard d'Alembert, Guyancourt, 78280, France.

*Corresponding author(s). E-mail(s): r.l.shinde@utwente.nl; william.jalby@uvsq.fr;
Contributing authors: c.filippi@utwente.nl; scemama@irsamc.ups-tlse.fr;

Abstract

The era of exascale computing presents both exciting opportunities and unique challenges for quantum mechanical simulations. While the transition from petaflops to exascale computing has been marked by a steady increase in computational power, the shift towards heterogeneous architectures, particularly the dominant role of graphical processing units (GPUs), demands a fundamental shift in software development strategies. This review examines the changing landscape of hardware and software for exascale computing, highlighting the limitations of traditional algorithms and software implementations in light of the increasing use of heterogeneous architectures in high-end systems. We discuss the challenges of adapting quantum chemistry software to these new architectures, including the fragmentation of the software stack, the need for more efficient algorithms (including reduced precision versions) tailored for GPUs, and the importance of developing standardized libraries and programming models.

Keywords: Quantum Mechanical Simulations, High-Performance Computing, Heterogeneous Architectures, Software Stack, Exascale



(a) (b)

Fig. 1: **(a) Progress in high-performance computing power**. LINPACK performance (Rmax) trend for the fastest supercomputer and the cumulative performance of Top 500 supercomputers over the years. **(b) Power consumption by the top supercomputer**. Power consumption trend for the fastest supercomputer over the years. Data from reference [34].

1 Rationale and scope

Until recently, the trajectory of top-performing computers has been fairly predictable, guided by a handful of leading hardware vendors [1]. These vendors have consistently provided scientists with clear paths (via stable languages, compilers, libraries, and tools) to harness escalating computational power, culminating in the era of petaflops [2–4]. The move to exascale computing comes with unique challenges, including a rapidly changing array of hardware options [5–9] and the once-reliable software stack that is now becoming increasingly fragmented and inconsistent [10–16]. After traditional applications like cryptocurrency mining and gaming, artificial intelligence (AI) has effectively harnessed the power of Graphical Processing Units (GPUs). This success is due to a strong alignment between AI algorithms and the architectural strengths of GPUs, coupled with the development of specialized libraries [17–20]. The transition to exascale for quantum mechanical simulations uncovers an intricate web of prospects and predicaments for physicists and chemists [4, 21, 22] and typically requires large community efforts (e.g. US Exascale Computing Project [23], EU Centers of Excellence [24] and National Competence Centers [25]) well beyond individual capabilities. This review explores the uneven path toward exascale simulations, highlighting both promising developments and persistent challenges [26–33] and offering a nuanced understanding of this important shift in computational science.

2 The transition to exascale

Quantum mechanical (QM) simulations are essential for understanding and predicting the behavior of complex materials and molecular systems across various scientific

disciplines. The accuracy of these simulations hinges on precisely describing electronic structures, a task that is computationally demanding and has been dramatically enhanced by advancements in high-performance computing (HPC). The progress in HPC is marked by milestones tracked by the Top500 list [34], which monitors peak computational power through a benchmark known as LINPACK [35]. This algorithm measures the performance of systems in solving a dense linear system, a task optimized for high computational throughput with minimal data movement and communication. Remarkably, LINPACK achieves around 70% of the theoretical maximum speed, derived from the total number of cores multiplied by their nominal speed. Figure 1(a) exhibits the steady progress of computational power, culminating in crossing the exaflop barrier approximately two years ago.

However, LINPACK's numbers, while impressive, are not fully representative of real-world performance. A more challenging and realistic benchmark is the High-Performance Conjugate-Gradient (HPCG) test, which evaluates the ability to solve sparse linear systems and is therefore representative of a different type of numerical applications, for instance, based on finite elements and finite difference schemes [36]. The performance here is starkly different, typically about 20 times lower than LINPACK, utilizing less than 5% of the nominal speed [37]. This contrast underscores the difference between idealized benchmarks and practical workloads.

While LINPACK and HPCG represent reasonable upper and lower bounds of performance, respectively, neither reflects the complexity of actual applications. The Gordon Bell Prize, awarded for outstanding achievements in high-performance computing applications, provides a more relevant measure [38]. These applications, often solving real scientific problems, have demonstrated performance nearing the exaflop threshold [39, 40]. Achieving this requires monumental efforts: entire codebases are rewritten by interdisciplinary teams of physicists, chemists, mathematicians, and computer scientists. New algorithms are developed, and implementations are tailored to exploit the specific features of the target architectures. In contrast, legacy codes, which constitute the bulk of real-world applications, lag significantly behind. These codes were not designed with exascale capabilities in mind and thus fall short of the exaflop performance. The challenge ahead lies in bridging this gap, ensuring that applications needing exascale capabilities can harness the full potential of exascale systems.

3 Hardware: The changing landscape of computational power

The transition from petaflops to exascale computing marked a significant departure from the consistent and predictable trajectory of high-performance computing. While the climb to petaflops was characterized by steady increases in processor core count and clock speeds, exascale computing brings a fundamental shift in hardware architectures and software development strategies. One of the most significant changes is the increasing adoption of heterogeneous architectures. Unlike the homogeneous central processing unit (CPU)-based systems that dominated petaflops, exascale systems heavily rely on accelerators like GPUs, which offer massive parallelism for specific tasks but come with unique challenges. This heterogeneous landscape presents a major

hurdle for software developers as traditional algorithms and software stacks designed for CPUs need significant modifications to utilize the capabilities of GPUs efficiently.

Before exploring this hardware evolution, it is crucial to identify the primary application areas driving these advancements, as they fundamentally determine the level of investment and the trajectory of hardware development. Scientific HPC, including QM applications, remains a niche market and does not significantly influence major hardware innovations. Instead, scientific computing and exaflop architectures often repurpose software and hardware technology developed for other, larger markets. Two key segments, that are increasingly shaping the practices of the scientific community, are cloud and accelerator computing, driven primarily by their adoption in conventional companies and the field of AI, respectively. The general-purpose laptop segment also indirectly impacts high-end processor computing. For instance, Apple, a major customer of the Taiwan Semiconductor Manufacturing Company (TSMC), exerts significant pressure on high-end chip production for both cloud computing and AI applications due to their high volume demand, as these components share the same fabrication lines [41].

In analyzing hardware evolution, we focus on three major components: CPUs, accelerators, and supercomputing systems.

On the CPU front, changes have been incremental over the past decade. Major CPU manufacturers (AMD, Intel, ARM) have converged towards a similar generic architecture: multicore processors based on out-of-order and superscalar technology. A significant development has been the increase in the number of cores per processor, now exceeding one hundred. While other technologies have emerged, none have achieved widespread adoption. Wider vectors (up to 512 bits) have been embraced by x86 architectures (initially by Intel, followed by AMD), whereas ARM continues to use shorter vectors (128/256 bits) without major performance penalties. Notably, memory technology has evolved more significantly, with very large level 3 caches using three-dimensional stacking and High Bandwidth Memory (HBM) offering an alternative to the standard Double Data Rate Synchronous Dynamic Random Access Memory (DDR-SDRAM). The performance gains from these new memory technologies depend heavily on the application's data access characteristics.

On the accelerator front, the primary advancement has been the massive increase in core count, now reaching thousands – nearly two orders of magnitude more than CPUs. The number of cores and peak performance depend on the floating-point (FP) format; typically, double-precision (DP) cores are half as numerous as single-precision (SP) cores. The shift toward narrower FP formats has been driven by AI, which tolerates reduced precision (16-bit, 8-bit, and even 4-bit formats are in use). Modern GPUs achieve remarkable performance using smaller FP formats and Tensor Cores, which are optimized for dense matrix multiplications. To illustrate these differences, Table 1 compares the computational performance of a high-performance CPU and a GPU released within the same time frame. Peak performance reflects execution speed only when computation is the bottleneck, not memory transfers or inter-node communications, which are too slow to keep up with the execution units. With standard DP operations (FP64), GPU peak performance is about five times that of a high-end CPU. When Tensor Cores are used, this increases to a factor of ten. For SP (FP32)

Table 1: Comparison of the peak performance with different precision modes, the amount of memory, the memory bandwidth and the thermal design power (TDP) of an Nvidia H100 GPU [42] and an AMD EPYC (Bergamo) CPU [43].

	Nvidia H100 SXM 16896 cores, 1.6 GHz	↔	AMD EPYC™ 9754 128 cores, 2.25 GHz
Release date	September 2022		June 2023
FP64	34 TFlop/s		6.9 TFlop/s
FP64 (tensor core)	67 TFlop/s		
FP32	67 TFlop/s		13.8 TFlop/s
FP32 (tensor core)	989 TFlop/s		
FP16 (tensor core)	1979 TFlop/s		
FP8 (tensor core)	3958 TFlop/s		
Memory	80 GB		max 6000 GB
Memory Bandwidth	3350 GB/s (HBM3)		460.8 GB/s (DDR5)
Interconnection Nvlink	900 GB/s		
Interconnection PCIe		128 GB/s	
Quantum X800 Infiniband network	100 GB/s		100 GB/s
TDP	700 W		360 W

operations, both CPU and GPU performance doubles, but with Tensor Cores, the GPU reaches 989 TFlop/s, over 70 times that of the CPU. Reducing precision to 8 bits (FP8) adds another factor of four. While smaller FP formats can sometimes be used for standard FP32 and FP64 tasks, Tensor Cores mainly benefit computations reliant on dense matrix multiplication. Achieving GPU peak performance is harder than on CPUs, as it requires high levels of parallelism. Consequently, matrices on the GPU must be much larger [44], or numerous, requiring batching algorithms. Furthermore, GPUs have less memory than CPUs [45] and large workloads necessitate memory transfers from the host to the accelerator, which must be managed in parallel with computations to prevent communication bottlenecks. We further elaborate below on the implications of these CPU and GPU performance differences for quantum simulations.

Finally, on the supercomputing systems front, the changes have been fairly radical, using Top500 as a reference, most of the major changes can be easily monitored. First, the amount of parallelism has drastically increased: the top runner in Top500 had 200 000 cores in June 2010, and now it has 9 million cores (a 45× increase). The overall system organization has also radically changed: in 2010, GPUs were used in less than 40 systems among the 500, now over half of the systems are equipped with GPUs. Today's top systems exhibit a heterogeneous architecture, integrating both CPUs and GPUs, though the bulk of computational capability predominantly stems from the GPUs. Moreover, the total power consumption of these systems has increased by an order of magnitude, rising from 3 MW to 30 MW, albeit with a notably irregular progression, as depicted in Figure 1(b). This underscores the substantial enhancements in system design, particularly in terms of energy efficiency.

4 Software: The challenge of increasing fragmentation

The transition from petascale to exascale computing has brought unprecedented computational power but has also introduced significant challenges in adapting software to efficiently utilize the new architectures. One of the most prominent challenges is the increasing fragmentation of the software stack, particularly with GPU programming. Nvidia had an early advantage for being a pioneer in this area. This created a relatively unified ecosystem where developers could rely on a single set of tools and libraries to harness the power of GPUs. However, as GPU computing gained wider adoption, other vendors such as AMD and Intel entered the market, each with their own programming models (HIP and SYCL, respectively). This proliferation of models has led to a fragmented landscape, making it increasingly difficult for developers to write portable and efficient code that can run seamlessly across different GPU platforms or heterogeneous systems.

OpenCL was introduced to address portability, providing an open standard for CPUs, GPUs, and field programmable gate arrays (FPGAs) across vendors. However, it has not gained wide adoption in HPC due to performance and usability issues. While OpenCL offers portability, its abstraction often results in suboptimal performance compared to specialized frameworks like CUDA. Its lower-level programming model is also cumbersome, requiring significant manual effort for memory management and kernel optimization. Fragmented implementations across vendors further undermine OpenCL's portability, as performance and features can vary, forcing developers to fine-tune code for each platform.

The lack of an efficient standardized programming model for GPUs has resulted in a number of issues for developers. Firstly, it creates vendor lock-in, where codes written for one platform cannot easily be migrated to another without substantial code refactoring. For example, adopting NVIDIA programming models involves significant trade-offs between performance and portability across different hardware environments [46]. Another issue stemming from this fragmentation is the lack of compiler support for all programming models on all platforms. For instance, (Fig. 2), while CUDA is well supported on Nvidia GPUs through the CUDA Toolkit, its support on AMD and Intel hardware is limited and relies on indirect means like translation layers or third-party libraries and utilities [47, 48]. This forces developers to rely on vendor-specific compilers, which often lack the universality and feature-richness of their CPU counterparts.

The compatibility between various GPU programming models and vendors is complex, largely due to the growing number of choices involving GPU platforms, programming models, and languages. As recently analyzed by Herten [49], while OpenMP is natively supported across the three major platforms—AMD, Intel, and Nvidia—and works with both C++ and Fortran, other popular models like OpenACC have limited support on Intel GPUs. This underscores the importance of thoroughly assessing a programming model's level of support on a chosen platform before beginning code development. Additionally, standardized benchmarks and evaluation tools are needed to effectively compare the performance and capabilities of different programming models.

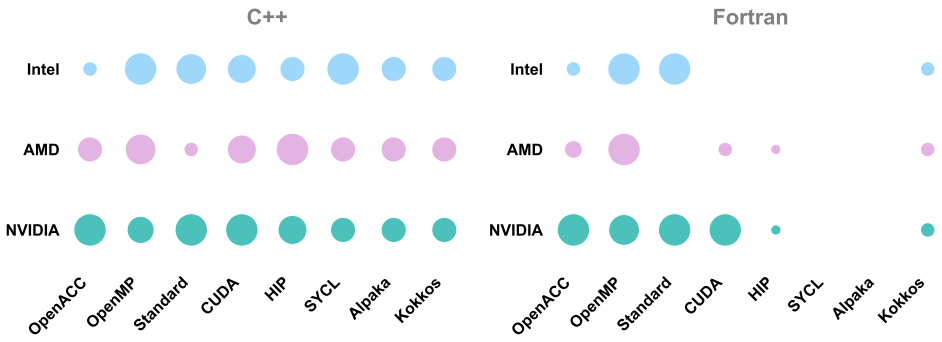


Fig. 2: **Vendors and support compatibility charts**. (a) C++ (b) Fortran. The size of the circle represents the software and toolset supported by vendors in the following order - full vendor support, indirect but comprehensive support by vendor, comprehensive support but not by vendor, non-comprehensive vendor support, limited but some support, and a missing circle represents no direct support available.

The lack of standardization in GPU programming has also led to a variety of community-driven, higher-level models that aim to abstract away vendor-specific details and provide a more portable programming experience. Examples of such models include Kokkos [50], RAJA [51], Alpaka [52], and, to a lesser extent, hipSYCL [53] which is limited to AMD and Intel. These abstraction models often utilize vendor-native infrastructure in the background, enabling developers to write code that can be deployed on multiple platforms without significant code changes. However, the support and standardization of these higher-level models can vary significantly, and relying on community-driven efforts for critical software infrastructures can introduce uncertainties and complexities in the long run. Comparing the performance of various GPU programming models on the LUMI supercomputer [54] reveals the potential of community-driven models like Kokkos in achieving portable performance but also highlights the need for further development and community support to ensure their long-term viability.

Another important aspect of the software stack is the role of low-level libraries for key computational kernels, such as linear algebra, fast Fourier transforms, and communication routines. These libraries play a critical role in achieving high performance and efficiency on exascale systems. However, the use of different libraries across different platforms can also contribute to software fragmentation. For example, while Nvidia provides the highly optimized cuBLAS library for linear algebra operations on their GPUs, AMD offers the RocBLAS library. These libraries, despite providing similar functionality, often have different APIs and performance characteristics, further hindering code portability.

The exascale era has reiterated the need for a more unified and standardized software stack for GPU programming. The fragmentation creates barriers to code portability, hinders performance optimization, and increases the development effort required for scientific applications. As we move forward, the community must address

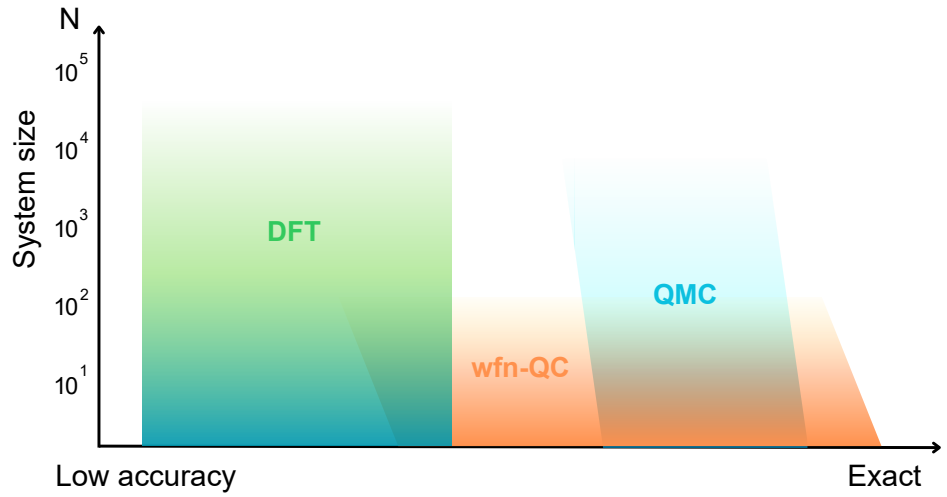


Fig. 3: **Landscape of electronic-structure calculation methods**. Different areas of applicability of density functional theory (DFT), wavefunction-based quantum chemical methods (wfn-QC), and quantum Monte Carlo (QMC).

these challenges. Initiatives like SYCL, OpenMP, and community-driven efforts like Kokkos and RAJA look promising toward achieving a more unified programming experience. However, vendors, developers, and researchers must collaborate closely to establish common standards and standardize a more cohesive and interoperable software ecosystem for GPU computing.

5 QM computation in the current HPC landscape

Quantum mechanical (QM) methods encompass a broad spectrum of applications, each with varying computational demands and levels of accuracy. As depicted in Fig. 3, density functional theory (DFT) represents a computationally efficient approach, offering a favorable balance between precision and performance, making it a popular choice for many systems. On the other end of the spectrum, wave-function-based methods are indispensable when higher accuracy is required to capture the intricate properties of more complex systems. Quantum Monte Carlo (QMC) methods also offer a path to high precision for larger systems by employing stochastic techniques to solve the Schrödinger equation.

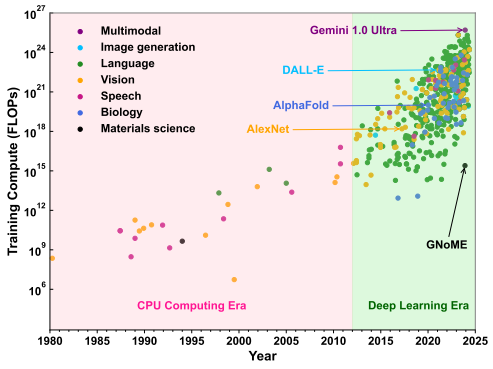
The nominal computational cost of these methods varies significantly. DFT is generally efficient, scaling with the number of electrons N as $O(N^3)$, primarily due to the diagonalization of the Kohn-Sham eigenvalue problem. In contrast, wave-function-based methods such as configuration interaction (CI), density matrix renormalization group (DMRG), and coupled-cluster (CC) methods demand considerably more computational resources, with scaling complexities that increase steeply depending on the method's sophistication. For example, the computational cost of the "gold standard"

in quantum chemistry, coupled-cluster singles and doubles with perturbative triples (CCSD(T)), scales as $O(N^7)$. QMC methods, particularly in their commonly used real-space variants, occupy an intermediate position in terms of computational cost, with a scaling of $O(N^3)$ per Monte Carlo step, largely governed by the repeated evaluation of the Slater determinant.

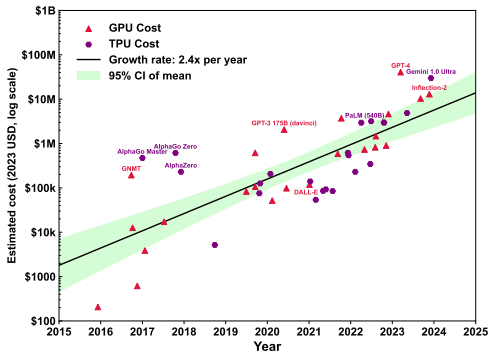
Each QM method relies on distinct computational techniques (e.g., fast Fourier transforms, matrix-vector operations, diagonalization of sparse or dense matrices), which determine the scaling behavior, while the specific hardware implementation governs the prefactor associated with this scaling. For instance, matrix multiplication kernels are central to CC methods that utilize large tensors, enabling efficient use of tensor cores and resulting in a relatively small prefactor despite the high scaling complexity. On the other hand, although QMC methods are inherently massively parallel, they often operate on relatively small matrices, leading to a large prefactor; therefore, scaling these approaches on accelerators requires significant effort [55].

The computational physics and chemistry communities have implemented these and other QM approaches in various software packages, facilitating discoveries across multiple disciplines. Among the widely used electronic structure codes, ABINIT [56], BerkeleyGW [57], BigDFT [58], CP2K [59], FHI-aims [60], NWChemX [61], Q-Chem [62], Quantum ESPRESSO (QE) [63], and VASP [64] have been particularly successful in harnessing HPC advancements for QM calculations, each with unique strengths and target applications. These codes have made significant strides in simulating larger systems, incorporating advanced algorithmic and methodological developments, and enhancing parallel efficiency, particularly on pre-exascale architectures [29, 65].

Recent efforts to adapt to the heterogeneous architectures of HPC systems highlight the critical importance of balancing performance with portability. Some codes, such as ABINIT and QE, have adopted hardware-agnostic offload programming models (OpenMP or OpenACC), while others, like CP2K, have opted for non-portable, kernel-based models (CUDA and HIP). BerkeleyGW provides both CUDA and OpenACC alternatives, with the latter achieving performance within ten percent of the former [57]. Additionally, different communities are actively developing specialized performance libraries to execute computationally intensive tasks on accelerators. Examples include the Distributed Block Compressed Sparse Row Matrix (DBCSR) library [66] for NVIDIA and AMD GPUs and the LibintX [67, 68] library, which provides efficient implementations of Gaussian integral evaluation, central to integral-direct implementations. The QMCkl library, developed within the TREX CoE, offers high-performance computation kernels for QMC simulations using OpenMP, OpenACC, and SYCL [69]. The wave-function community [27, 61, 70–73] has been extensively developing tensor libraries that automatically parallelize and accelerate operations on multidimensional arrays, taking advantage of modern HPC architectures [74–76].



(a)



(b)

Fig. 4: **The exponential growth of AI and its cost**. (a) The explosion of AI models has caused a substantial demand for computing power for training. (b) The estimated cost of hardware and energy required to train the top 10 AI models and the predicted growth. Data from [77, 78].

6 AI and quantum simulations

While both AI and QM simulations extensively need the power of modern computing, the nature of computations involved in each field and their respective challenges in reaching exascale performance are fundamentally different. AI, particularly machine learning (ML), has exhibited significant progress in recent years due to its ability to extract patterns and learn complex relationships from data [79, 80]. The success (see Fig. 4(a)) is mainly due to the availability of large datasets, advancements in algorithm development, and the accessibility of HPC resources, especially those equipped with GPUs and tensor processing units (TPUs).

The computations involved in AI training primarily revolve around optimizing the weights and biases of a neural network to minimize a specific loss function, which measures the difference between predicted and actual target values. Such tasks often

involve reduced-precision matrix multiplications, convolutions, and activation function evaluations. Optimization techniques like stochastic gradient descent and its variants are also widely employed, involving iterative updates to network parameters based on gradients of the loss function calculated using backpropagation [81, 82]. These computations are inherently parallel, as the gradients for different data points can be computed independently and then averaged. Furthermore, model parallelism, where different parts of a model are handled by different processors, allows for efficient training of complex models with billions of parameters [83]. Figure 4(a) illustrates the exponential growth in computational requirements for training large AI models, that double every 3.4 months compared to conventional Moore's law [84]. The inherent parallelism in AI training makes it particularly well-suited for GPUs, which excel at executing the same operation on multiple data points simultaneously.

Consequently, the impressive acceleration of almost 300×, illustrated in Table 1, for matrix multiplications using reduced precision on GPUs can be readily exploited by AI, while QM simulations typically require double precision where only a tenfold speedup can be achieved for matrix multiplications using vendor libraries. For kernels written in OpenMP, OpenACC, or OpenCL, lacking the tensor core boost, an acceleration factor of less than five is usually expected in QM simulations. In the current QM landscape, many algorithms exhibit low arithmetic intensity, largely due to their historical development and the limitations of earlier hardware. As a result, the potential advantages of GPUs are not always clear for these traditional algorithms. To fully harness the computational power of modern GPUs, particularly their ability to perform dense matrix multiplications at reduced precision, new algorithms must be developed. These new approaches would need to align more closely with GPU strengths, optimizing for parallelism and reduced precision where possible, rather than relying on the double-precision calculations and sparse matrix operations that have characterized QM methods to date.

Interestingly, scientists have started to utilize the power of AI/ML to address the accuracy-efficiency dilemma of quantum mechanical calculations. For example, machine learning potentials, which are generally trained on *ab initio* data, can predict potential energy surfaces and forces with the accuracy of the underlying quantum method, but at a much lower computational cost [85]. An efficient materials discovery and design using high-throughput calculations is now possible [86] with public datasets [87] and efficient optimization techniques [88]. Furthermore, neural network wavefunctions in combination with quantum Monte Carlo methods have shown promise in achieving highly accurate solutions of the electronic Schrödinger equation for electronic systems [81, 82, 89–91].

7 Outlook and conclusions

The future of quantum mechanical simulations at the exascale level hinges on effectively navigating the shifting landscape of high-performance computing hardware and software. A critical aspect of this adaptation is the development of algorithms that go beyond simply exploiting the brute force of massive parallelism provided by GPUs. While GPUs offer significant acceleration, the impressive performance gain is limited

to algorithms using matrix-matrix multiplies with reduced precision. Designing alternative algorithms for QM simulations leveraging this particular strength of GPUs is a must. Furthermore, given the complexity of adapting legacy codes to hybrid architectures, the community must unite in embracing and standardizing modular libraries for quantum chemical calculations, providing reusable, optimized, and versatile implementations of complex algorithms [92]. Some specialized libraries have already demonstrated significant success in enabling efficient implementations on diverse hardware platforms, and their continued development and community-wide adoption are crucial for simplifying code development and ensuring long-term code sustainability. Finally, AI has already proven its broad potential, generating machine-learning force fields with *ab initio* accuracy, faithfully describing quantum states, and speeding up various tasks within quantum simulations [93]. As a result, the growing integration of AI with quantum simulations offers a promising path for future breakthroughs.

However, a few notes of caution are necessary regarding the widespread adoption of GPUs. CPUs remain essential for workloads that require flexibility, general-purpose computation, or that are memory-bound. For example, CPUs equipped with HBM are competitive with GPUs in some cases, particularly in memory-limited tasks. Although in the near future, we can expect CPUs to integrate AI acceleration features, they will likely remain complementary to GPUs. In addition, while their computational power is undeniable, the energy consumption of GPUs is a growing concern. While a laptop requires less than 100W to function, the peak energy consumption of a modern supercomputer like Frontier is around 22MW. The community must prioritize energy efficiency as a core design principle for future exascale simulations. Time-to-solution should be scaled by power usage when comparing simulations on different architectures. Re-evaluating the blind pursuit of raw performance at the cost of enormous energy consumption (*cf.* Figs. 1(b) and 4(b)) is of paramount importance. The focus should shift towards a more balanced approach where algorithmic efficiency, software optimization, and energy-aware hardware design work in synergy.

Importantly, while the exascale era offers exciting prospects for quantum mechanical simulations, it also demands a cohesive effort from the scientific community to overcome the associated challenges. Collaborative initiatives, such as those driven by global research collaborations and industry partnerships, will be vital in shaping the future of computational chemistry and materials science. By embracing energy-efficient computing, stronger collaboration between academia and industry, and prioritizing the development of standardized software tools (e.g. languages, compilers, debuggers, libraries), we can ensure that exascale computing delivers on its promise as a cornerstone of scientific innovation, enabling us to take on grand challenges and push the boundaries of our understanding of the quantum world.

Acknowledgments. The authors acknowledge partial support from the European Centre of Excellence in Exascale Computing TREX — Targeting Real Chemical Accuracy at the Exascale. This project has received funding in part from the European Union's Horizon 2020 — Research and Innovation Program — under grant agreement no. 952165.

Declarations

- Author contribution: R.S. researched data for the article. All authors contributed to the writing of the manuscript.

References

[1] Dongarra, J. J. & Walker, D. W. The quest for petascale computing. *Comput. Sci. Eng.* **3**, 32–39 (2001).

[2] Bader, D. A. *Petascale Computing: Algorithms and Applications (Computational Science)* (Chapman and Hall/CRC, 2007). URL https://www.amazon.com/Petascale-Computing-Algorithms-Applications-Computational/dp/1584889098.

[3] Geist, A. & Lucas, R. Major Computer Science Challenges At Exascale. *Int. J. High Perform. Comput. Appl.* **23**, 427–436 (2009).

[4] Vetter, J. S. *Contemporary High Performance Computing* (Chapman and Hall/CRC, 2017). URL https://www.oreilly.com/library/view/contemporary-high-performance/9781466568358.

[5] Pedretti, K. *et al.* Chronicles of Astra: Challenges and Lessons from the First Petascale Arm Supercomputer. (2020). URL https://www.osti.gov/biblio/1822114. [Online; accessed 28. Aug. 2023].

[6] Kogge, P. M. & Dally, W. J. *Frontier vs the Exascale Report: Why so long? and Are We Really There Yet?*, 26–35 (IEEE, 2022).

[7] Sinha, P. *et al.* in *Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems* 01–15 (IEEE, 2022).

[8] Patrizio, A. ISC '22: The AMD-Intel-Nvidia HPC race heats up. *Network World* (2022). URL https://www.networkworld.com/article/3662114/isc-22-the-amd-intel-nvidia-hpc-race-heats-up.html.

[9] Loh, G. H. *et al.* in *A Research Retrospective on AMD's Exascale Computing Journey* 1–14 (Association for Computing Machinery, New York, NY, USA, 2023).

[10] Dongarra, J. *et al.* The International Exascale Software Project roadmap. *Int. J. High Perform. Comput. Appl.* **25**, 3–60 (2011).

[11] Fiore, S., Bakhouya, M. & Smari, W. W. On the road to exascale: Advances in High Performance Computing and Simulations—An overview and editorial. *Future Gener. Comput. Syst.* **82**, 450–458 (2018).

[12] Richard, R. M. *et al.* Developing a Computational Chemistry Framework for the Exascale Era. *Comput. Sci. Eng.* **21**, 48–58 (2018).

[13] Gordon, M. S. *et al.* Novel Computer Architectures and Quantum Chemistry. *J. Phys. Chem. A* **124**, 4557–4582 (2020).

[14] Luo, L. *et al.* Pre-exascale accelerated application development: The ORNL Summit experience. *IBM J. Res. Dev.* **64**, 11:1–11:21 (2020).

[15] McInnes, L. C. *et al.* How community software ecosystems can unlock the potential of exascale computing. *Nat. Comput. Sci.* **1**, 92–94 (2021).

[16] Matsuoka, S., Domke, J., Wahib, M., Drozd, A. & Hoefler, T. Myths and legends in high-performance computing. *Int. J. High Perform. Comput. Appl.* **37**, 245–259 (2023).

[17] Lu, Y., Qian, D., Fu, H. & Chen, W. Will supercomputers be super-data and super-AI machines? *Commun. ACM* **61**, 82–87 (2018).

[18] Huerta, E. A. *et al.* Convergence of artificial intelligence and high performance computing on NSF-supported cyberinfrastructure. *J. Big Data* **7**, 1–12 (2020).

[19] Gepner, P. in *Machine Learning and High-Performance Computing Hybrid Systems, a New Way of Performance Acceleration in Engineering and Scientific Applications* 27–36 (IEEE, 2021).

[20] Liang, B.-S. in *AI Computing in Large-Scale Era: Pre-trillion-scale Neural Network Models and Exa-scale Supercomputing* 1–3 (IEEE, 2023).

[21] Geist, A. & Reed, D. A. A survey of high-performance computing scaling challenges. *Int. J. High Perform. Comput. Appl.* **31**, 104–113 (2015).

[22] Evans, T. M. *et al.* A survey of software implementations used by application codes in the Exascale Computing Project. *Int. J. High Perform. Comput. Appl.* **36**, 5–12 (2021).

[23] Research - Exascale Computing Project (2020). URL https://www.exascaleproject.org/research. [Online; accessed 05 June 2024].

[24] EU HPC Centres of Excellence (2023). URL https://www.hpccoe.eu/eu-hpc-centres-of-excellence2. [Online; accessed 25 June 2024].

[25] EuroCC ACCESS (2023). URL https://www.eurocc-access.eu. [Online; accessed 25 June 2024].

[26] Kowalski, K. *et al.* From NWChem to NWChemEx: Evolving with the Computational Chemistry Landscape. *Chem. Rev.* **121**, 4962–4998 (2021).

[27] Pototschnig, J. V. *et al.* Implementation of Relativistic Coupled Cluster Theory for Massively Parallel GPU-Accelerated Computing Architectures. *J. Chem. Theory Comput.* **17**, 5509–5529 (2021).

[28] Yokelson, D., Tkachenko, N. V., Robey, R., Li, Y. W. & Dub, P. A. Performance Analysis of CP2K Code for Ab Initio Molecular Dynamics on CPUs and GPUs. *J. Chem. Inf. Model.* **62**, 2378–2386 (2022).

[29] Gavini, V. *et al.* Roadmap on electronic structure codes in the exascale era. *Model. Simul. Mater. Sci. Eng.* **31**, 063301 (2023).

[30] Kim, I. *et al.* Kohn–Sham time-dependent density functional theory with Tamm–Dancoff approximation on massively parallel GPUs. *npj Comput. Mater.* **9**, 1–12 (2023).

[31] Galvez Vallejo, J. L. *et al.* Toward an extreme-scale electronic structure system. *J. Chem. Phys.* **159** (2023).

[32] Corzo, H. H. *et al.* Coupled cluster theory on modern heterogeneous supercomputers. *Front. Chem.* **11**, 1154526 (2023).

[33] Schade, R. *et al.* Breaking the exascale barrier for the electronic structure problem in ab-initio molecular dynamics. *Int. J. High Perform. Comput. Appl.* 10943420231177631 (2023).

[34] TOP500.org. June 2024 | TOP500 (2024). URL https://www.top500.org/lists/top500/2024/06. [Online; accessed 1. Aug. 2024].

[35] Dongarra, J. J., Luszczek, P. & Petitet, A. The LINPACK Benchmark: past, present and future. *Concurrency Computat.: Pract. Exper.* **15**, 803–820 (2003).

[36] Dongarra, J., Heroux, M. A. & Luszczek, P. High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. *Int. J. High Perform. Comput. Appl.* **30**, 3–10 (2015).

[37] TOP500.org. HPCG - June 2024 | TOP500 (2024). URL https://www.top500.org/lists/hpcg/2024/06. [Online; accessed 1. Aug. 2024].

[38] ACM Gordon Bell Prize (2024). URL https://awards.acm.org/bell. [Online; accessed 1. Aug. 2024].

[39] Liu, Y. A. *et al.* in *Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new sunway supercomputer* 1–12 (Association for Computing Machinery, New York, NY, USA, 2021).

[40] Das, S. *et al.* in *Large-Scale Materials Modeling at Quantum Accuracy: Ab Initio Simulations of Quasicrystals and Interacting Extended Defects in Metallic Alloys* 1–12 (Association for Computing Machinery, New York, NY, USA, 2023).

[41] Bloomberg and Digitimes. TSMC top 10 customers revealed: Apple accounts for quarter of revenue (2021). URL https://www.gizmochina.com/2021/12/15/tsmc-top-10-customers-revealed-apple-accounts-for-quarter-of-revenue/.

[Online; accessed 14. Aug. 2024].

[42] GPU NVIDIA H100 Tensor Core (2024). URL https://www.nvidia.com/fr-fr/data-center/h100. [Online; accessed 2. Sep. 2024].

[43] AMD EPYC™ 9754 (2024). URL https://www.amd.com/en/products/processors/server/epyc/4th-generation-9004-and-8004-series/amd-epyc-9754.html. [Online; accessed 2. Sep. 2024].

[44] Sorokin, A., Malkovsky, S. & Tsoy, G. Comparing the performance of general matrix multiplication routine on heterogeneous computing systems. *J. Parallel Distrib. Comput.* **160**, 39–48 (2022).

[45] 4th Gen AMD EPYC Review (AMD Genoa) (2022). URL https://www.storagereview.com/review/4th-gen-amd-epyc-review-amd-genoa. [Online; accessed 2. Sep. 2024].

[46] Hammond, J. Shifting through the gears of gpu programming: Understanding performance and portability trade-offs (2022). URL https://www.nvidia.com/en-us/on-demand/session/gtcspring22-s41620/. Accessed: 2024-06-11.

[47] Intel Inc. Syclomatic. URL https://github.com/oneapi-src/SYCLomatic. Accessed: 2024-06-11.

[48] Intel oneAPI DPC++/C++ compiler (2024). URL https://www.intel.com/content/www/us/en/developer/tools/oneapi/dpc-compiler.html#gs.aek6kb. Accessed: 2024-06-11.

[49] Herten, A. *Many cores, many models: GPU programming model vs. vendor compatibility overview*, SC-W 2023 (ACM, 2023). URL http://dx.doi.org/10.1145/3624062.3624178.

[50] Carter Edwards, H., Trott, C. R. & Sunderland, D. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing* **74**, 3202–3216 (2014). URL https://www.sciencedirect.com/science/article/pii/S0743731514001257. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[51] Beckingsale, D. A. *et al. Raja: Portable performance for large-scale scientific applications*, 2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), 71–81 (2019).

[52] Matthes, A. *et al. Tuning and Optimization for a Variety of Many-Core Architectures Without Changing a Single Line of Implementation Code Using the Alpaka Library*, 496–514 (Springer International Publishing, 2017). URL http://dx.doi.org/10.1007/978-3-319-67630-2_36.

[53] Alpay, A., Soproni, B., Wünsche, H. & Heuveline, V. *Exploring the possibility of a hipsycl-based implementation of oneAPI*, IWOCL '22 (Association for Computing Machinery, New York, NY, USA, 2022). URL https://doi.org/10.1145/3529538.3530005.

[54] Markomanolis, G. S. *et al. Evaluating GPU Programming Models for the LUMI Supercomputer*, 79–101 (Springer International Publishing, 2022). URL http://dx.doi.org/10.1007/978-3-031-10419-0_6.

[55] Kim, J. *et al.* Qmcpack: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids. *Journal of Physics: Condensed Matter* **30**, 195901 (2018). URL https://dx.doi.org/10.1088/1361-648X/aab9c3.

[56] Gonze, X. *et al.* The abinit project: Impact, environment and recent developments. *Comput. Phys. Commun.* **248**, 107042 (2020). URL https://doi.org/10.1016/j.cpc.2019.107042.

[57] Ben, M. D. *et al. Accelerating large-scale excited-state gw calculations on leadership hpc systems*, 1–11 (2020).

[58] Ratcliff, L. E. *et al.* Flexibilities of wavelets as a computational basis set for large-scale electronic structure calculations. *The Journal of Chemical Physics* **152**, 194110 (2020). URL https://doi.org/10.1063/5.0004792.

[59] Kühne, T. D. *et al.* CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics* **152**, 194103 (2020). URL https://doi.org/10.1063/5.0007045.

[60] Blum, V., Rossi, M., Kokott, S. & Scheffler, M. The fhi-aims code: All-electron, ab initio materials simulations towards the exascale (2022). 2208.12335.

[61] Kowalski, K. *et al.* From nwchem to nwchemex: Evolving with the computational chemistry landscape. *Chemical Reviews* **121**, 4962–4998 (2021). URL https://doi.org/10.1021/acs.chemrev.0c00998.

[62] Shao, Y. *et al.* Advances in molecular quantum chemistry contained in the Q-Chem 4 program package. *Molecular Physics* **113**, 184–215 (2014). URL http://dx.doi.org/10.1080/00268976.2014.952696.

[63] Carnimeo, I. *et al.* Quantum ESPRESSO: One Further Step toward the Exascale. *J. Chem. Theory Comput.* **2023** (2023).

[64] Kresse, G. & Joubert, D. From ultrasoft pseudopotentials to the projector augmented-wave method. *Phys. Rev. B* **59**, 1758–1775 (1999). URL https://link.aps.org/doi/10.1103/PhysRevB.59.1758.

[65] Blum, V. *et al.* Roadmap on methods and software for electronic structure based simulations in chemistry and materials. *Electronic Structure* (2024). URL http://iopscience.iop.org/article/10.1088/2516-1075/ad48ec.

[66] Borštnik, U., VandeVondele, J., Weber, V. & Hutter, J. Sparse matrix multiplication: The distributed block-compressed sparse row library. *Parallel Computing* **40**, 47–58 (2014). URL https://www.sciencedirect.com/science/article/pii/S0167819114000428.

[67] Asadchev, A. & Valeev, E. F. High-performance evaluation of high angular momentum 4-center gaussian integrals on modern accelerated processors. *The Journal of Physical Chemistry A* **127**, 10889–10895 (2023). URL https://doi.org/10.1021/acs.jpca.3c04574.

[68] Asadchev, A. & Valeev, E. F. 3-center and 4-center 2-particle gaussian ao integrals on modern accelerated processors (2024). 2405.01834.

[69] Scemama, A. QMCkl: A Unified Approach to Accelerating Quantum Monte Carlo Codes (2024). URL https://doi.org/10.5281/zenodo.10622933.

[70] Cc4s User Documentation (2022). URL https://cc4s.github.io/user-manual. [Online; accessed 5. Jun. 2024].

[71] Hasik, J., Poilblanc, D. & Becca, F. Investigation of the Néel phase of the frustrated Heisenberg antiferromagnet by differentiable symmetric tensor networks (2020). URL https://scipost.org/submissions/scipost_202011_00009v2. [Online; accessed 5. Jun. 2024].

[72] Fishman, M., White, S. & Stoudenmire, E. M. The ITensor Software Library for Tensor Network Calculations. *SciPost Phys. Codebases* 004 (2022).

[73] Menczer, A. *et al.* Parallel implementation of the Density Matrix Renormalization Group method achieving a quarter petaFLOPS performance on a single DGX-H100 GPU node. *arXiv* (2024).

[74] Calvin, J. A., Lewis, C. A. & Valeev, E. F. *Scalable task-based algorithm for multiplication of block-rank-sparse matrices* (Association for Computing Machinery, New York, NY, USA, 2015). URL https://doi.org/10.1145/2833179.2833186.

[75] Solomonik, E., Matthews, D., Hammond, J. R., Stanton, J. F. & Demmel, J. A massively parallel tensor contraction framework for coupled-cluster computations. *Journal of Parallel and Distributed Computing* **74**, 3176–3190 (2014).

[76] Matthews, D. A. High-Performance Tensor Contraction without Transposition. *SIAM J. Sci. Comput.* (2018). URL https://epubs.siam.org/doi/10.1137/16M108968X.

[77] The exponential growth of AI computation (2024). URL https://thescience.dev/the-exponential-growth-of-ai-computation/. Accessed: 2024-07-05.

[78] Cottier, B., Rahman, R., Fattorini, L., Maslej, N. & Owen, D. The rising costs of training frontier AI models (2024). URL https://arxiv.org/abs/2405.21015.2405.21015.

[79] Bhattacharya, T. *et al.* AI meets exascale computing: Advancing cancer research with large-scale high performance computing. *Frontiers in Oncology* **9** (2019). URL https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2019.00984.

[80] Ang, J. A., Barker, K. J., Vrabie, D. L. & Kestor, G. Codesign for extreme heterogeneity: Integrating custom hardware with commodity computing technology to support next-generation hpc converged workloads. *IEEE Internet Computing* **27**, 7–14 (2023).

[81] Pfau, D., Spencer, J. S., Matthews, A. G. D. G. & Foulkes, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Res.* **2**, 033429 (2020). URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.033429.

[82] Hermann, J., Schätzle, Z. & Noé, F. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry* **12**, 891–897 (2020). URL http://dx.doi.org/10.1038/s41557-020-0544-y.

[83] Abts, D. & Kim, J. Enabling artificial intelligence supercomputers with domain-specific networks. *IEEE Micro* **44**, 41–49 (2024).

[84] AI and compute (2018). URL https://openai.com/index/ai-and-compute/. Accessed: 2024-06-11.

[85] Chandrasekaran, A. *et al.* Solving the electronic structure problem with machine learning. *npj Computational Materials* **5** (2019). URL http://dx.doi.org/10.1038/s41524-019-0162-7.

[86] Batatia, I. *et al.* A foundation model for atomistic materials chemistry. *arXiv* (2023).

[87] Jain, A. *et al.* *The Materials Project: Accelerating Materials Design Through Theory-Driven Data and Tools*, 1–34 (Springer International Publishing, 2018). URL http://dx.doi.org/10.1007/978-3-319-42913-7_60-1.

[88] Fare, C., Fenner, P., Benatan, M., Varsi, A. & Pyzer-Knapp, E. O. A multi-fidelity machine learning approach to high throughput materials screening. *npj Computational Materials* **8** (2022). URL http://dx.doi.org/10.1038/s41524-022-00947-9.

[89] Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).

[90] Wilson, M. *et al.* Neural network ansatz for periodic wave functions and the homogeneous electron gas. *Phys. Rev. B* **107**, 235139 (2023). URL https://link.aps.org/doi/10.1103/PhysRevB.107.235139.

[91] Han, J., Zhang, L. & E, W. Solving many-electron schrödinger equation using deep neural networks. *Journal of Computational Physics* **399**, 108929 (2019). URL https://www.sciencedirect.com/science/article/pii/S0021999119306345.

[92] Lehtola, S. A call to arms: Making the case for more reusable libraries. *J. Chem. Phys.* **159** (2023). URL https://doi.org/10.1063/5.0175165.

[93] Kulik, H. J. *et al.* Roadmap on Machine learning in electronic structure. *Electron. Struct.* **4**, 023004 (2022). URL https://dx.doi.org/10.1088/2516-1075/ac572f.