CHEMICAL REVIEWS

pubs.acs.org/CR

Editorial

Editorial: Modern Architectures and Their Impact on Electronic Structure Theory

Cite This: *Chem. Rev.* 2020, 120, 9015−9020

Read Online

ACCESS | Metrics & More | Article Recommendations

In 1929, P. A. M. Dirac asserted that "The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble".[1] Less than 15 years later, the physicists Atanasoff and Berry at Iowa State College constructed the first automatic digital computer,[2] a replica of which can be viewed on the Iowa State University campus. It is interesting to wonder what Dirac would think about the dramatic advances in the capabilities of both computer architectures and software paradigms which, taken together, have enabled theoretical and computational chemists to transform those "underlying physical laws" into important applications in chemistry, physics, biology, materials, and many other fields.

The computing world is on the brink of the exascale era, with computers that are capable of exaflop (10^{18} floating point operations/second) calculations anticipated to become available within the next two years. "Pre-exascale" computers; e.g., Summit at Oak Ridge National Laboratory, are already in operation. These computers have, or will have, the ability to successfully attack important problems that relate directly to current experiments and that could not have been solved otherwise. Examples include understanding heterogeneous catalysis in mesopores (including solvent and enough of the pore to be realistic), the design of new materials with desired properties, such as materials for quantum computing or ligands for selective separations, the study of condensed phase phenomena, and potentially designing new vaccines to combat viruses.

The most advanced computers (including the top 500 computers)[3] almost all have heterogeneous architectures that include both the latest computer processing units (CPUs) and graphical processing units (GPUs) and potentially other accelerators. GPUs, in particular, speed up calculations because of their high density of threads that greatly enhance parallelism for operations such as matrix multiplies that are ubiquitous in many electronic structure functionalities. All of the planned exascale computers (e.g., Aurora at Argonne National Laboratory, Frontier at Oak Ridge National Laboratory, and El Capitan at Lawrence Livermore National Laboratory—all in the U.S.[4]) will have a significant GPU presence. This heterogeneity necessitates the development of novel software engineering and the design of ubiquitous libraries, since the CPU and accelerator architectures often require the use of different languages, as well as different approaches to programming. Complicating this situation further, there are multiple GPU vendors (e.g., NVIDIA, AMD, Intel), each of which has its own language and software requirements. Indeed, Aurora and Frontier will be based on GPUs from different vendors. Consequently, many electronic structure functionalities (e.g., integrals[5] over basis functions, Fock builds for Hartree−Fock calculations,[6] second order perturbation theory (MP2),[7,8] multiconfigurational methods,[9] and coupled cluster (CC) methods[10]) that have been implemented primarily for NVIDIA GPUs must be translated by the application developer (e.g., quantum chemist) for use on another type of GPU. There are at present no efficient translators from one type of GPU to another, although significant effort is ongoing in this area.[11,12] Indeed, chemists themselves have been developing software generators that take quantum mechanical quantities (such as integrals)[13,5] and methods (such as CC)[14−16] and generate the code to be run by the end user. These software generators offer a potential pathway to faster generation of GPU specific code; however, the initial implementations are frequently computationally inefficient.

Other accelerators have been developed, notably the various versions of the Intel Phi,[17] especially Knights Landing (KNL) which is an important component of Cori at NERSC (the National Energy Research Scientific Computing Center at Berkeley). Others include ARM and field programmable gate arrays (FPGA), but none of these alternatives appear to be as computational effective as GPUs.[18] Nonetheless, one of the top 500 computers (Astra at Sandia National Laboratory, #198 on the list) is configured with ARM accelerators. In addition, it is noteworthy that the most recent Japanese system at the RIKEN Center for Computational Chemistry is based on the ARM architecture, and it is rumored that China's Tianhe-3 will also be based on ARM technology. Quantum computers are also making an appearance with limited qubits available to the scientific community.[19,20] The programming models are currently quite different from those that are commonly used within quantum chemistry (e.g., hybrid approaches in which classical computers and quantum computers each do that part of the computation for which they are most well suited).

Published: September 9, 2020

https://dx.doi.org/10.1021/acs.chemrev.0c00700
Chem. Rev. 2020, 120, 9015−9020

Consequently, quantum computers offer yet another challenge to quantum chemistry software development.

In concert with the addition of GPUs, the core counts per node have been steadily rising with the latest nodes having up to 48 cores per node. However, even with these large core counts, much of the compute power of the top 500 machines comes from the GPUs on each node. Many of these nodes also support at least three levels of cache and have relatively large memory available for computation. Currently, the CPU and GPU memories are mostly separate, with NVLink being the notable exception.[21] Programmers need to think carefully about data locality and reuse to minimize data movement between the CPUs and GPUs. In addition, since communication networks between nodes have not been able to keep up with the advances in raw compute power, data movement between nodes also needs to be minimized. This data locality has led to a resurgence of methods that take advantage of the local features within molecular chemistry that are discussed below.

For serial or low parallelism programming, disk-based methods were often useful since the communication to local disk was faster than having to recompute quantities on a single CPU that were too large to hold in memory. However, the increased computing power, especially parallel computing capability with multiple cores and multiple threads, has outstripped most of the disk capabilities—especially for large computers where the large disk storage is on a distributed file system rather than on the node (local disk space). This has led to the reframing of many problems in terms of direct computations (recalculation of quantities too large to store) and to algorithms that require less intermediate data, such as fragmentation and reduced scaling methods. However, even for I/O, new technology is continually becoming available. For example, solid-state drives, flash memory, and burst buffers could enable programmers to again use large storage space for intermediate quantities or for restart. However, most of the large computer centers have not taken advantage of this technology. So, most programmers do not count on having this capability available for their algorithmic developments.

While the computer architecture issues are complex on their own, there are also significant computer software stack and language requirements to navigate. Most electronic structure programs are written in Fortran, C++, or Python, or in some cases, combinations of these languages. An example of the latter is Psi4,[22] which is written partly in C++ and partly in Python. This is motivated by the fact that while Python is very good at the "traffic directing" aspects of a complex code, it is not very performant, whereas the opposite is true of C++ or Fortran. It is common for "legacy" codes, such as GAMESS,[23] Molpro,[24] Molcas[25] and NWChem[26] to be written primarily in Fortran77 or Fortran90, while newer codes are commonly written in an object oriented language like C++ (e.g., Q-Chem,[27] NWChemEx,[28] QMCPack[29]), but possibly modern versions of Fortran. For example, associated with GAMESS is a C++ code called LibCChem.[30] LibCChem currently contains closed shell HF, MP2, RI-MP2, and CCSD(T) functionalities. A novel programming language, Julia, combines the best features of the lower level languages (e.g., Fortran, C++) and the higher level languages such as Python. A recent Hartree–Fock implementation in Julia has been shown to be surprisingly competitive with the HF code in GAMESS.[31]

One attractive feature of C++ is that it is integrated with many high-performance computing frameworks. C++ has language-specific bindings for CUDA, OpenMP, OpenCL, and OpenACC. For this reason, programs written in C++ can readily make use of accelerators. Several quantum chemistry codes are being written with GPUs as the primary target using C++, including TeraChem,[32] LibCChem,[30] and NWChemEx.[28] This is very important since, as noted above, accelerators are playing an increasingly central role in high performance computing. The Department of Energy Exascale Computing Project (ECP) has had an increasing focus on GPUs over the past three years, and the anticipated exascale computers will all have a significant GPU presence. In addition, the latest C++ releases include features that support thread and task-based parallel processing, thereby reducing the need for using thread libraries such as Pthreads. It is also possible, however, to offload Fortran code directly onto GPUs as well. This has been accomplished for the RI-MP2 code in GAMESS.[23]

In addition to the base programming language for a code, there are multiple models for parallel programming that one must consider. MPI[33] is still considered the general standard for communication between nodes. Multiple programming models have been built within the chemistry community on top of MPI and/or native hardware communication libraries that allow the chemistry programmer to effectively worry more about the data layout and communication as opposed to the specifics of the MPI interface. Two of the notable examples here are the Global Arrays (GA) program that was codesigned as the nonuniform memory architecture with NWChem and the generalized distributed data interface (GDDI) that performs the same role for GAMESS. GA, an early example of a partitioned global address space (PGAS) language, is now being used in other codes such as MolPro,[24] GAMESS-UK,[34] Columbus,[35] LibCChem,[30] and MolCAS.[25] While these tools have been invaluable in the community, they do not have the full flexibility that is required for exascale computing.

The situation on the node is not nearly so straightforward and no single standard has emerged. Threading on the CPUs (and to the GPUs) is becoming more common—running only a few MPI processes on the node to handle communication and data transfer issues. C++ threads, Pthreads, Intel TBB, and OpenMP are common models for threading on CPUs—although none have emerged as a standard that is portable across all platforms and computing environments. For accelerators, the situation is even more complex since each GPU has a standard language associated with it. The compute unified device architecture (CUDA)[36] software for NVIDIA GPUs has attracted a large following in the chemistry community due to the ubiquity of NVIDIA GPUs. However, in the exascale computers, AMD (Frontier and El Capitan) and Intel (Aurora) GPUs will be used. AMD uses HIP[37] as its primary programming language. Since there is a mostly one-to-one-mapping between CUDA and HIP, conversion software seems to do a reasonable first pass at converting CUDA to HIP. Intel uses SYCL[38] (a version of OpenCL) and Data Parallel C++ (DPC++)[12] within the oneAPI software.[39] While the conversion from CUDA to SYCL is not a straightforward one-to-one mapping, there are several tools available and they are continuously improving. The other possible advantage of SYCL is that it uses the OpenCL convention as its base. Therefore, if an algorithm is written in OpenCL, then, in theory, the code will run across multiple platforms. Other models such as OpenMP and OpenACC are also meant to be portable across multiple platforms. However, all of these codes

https://dx.doi.org/10.1021/acs.chemrev.0c00700
Chem. Rev. 2020, 120, 9015−9020

work better on some platforms than others and no standard has emerged. In fact, one model may perform the best on one platform and perform quite poorly on another (or not at all). At this point, it is unclear which GPU architecture(s) will survive in the long run. As noted above, the use of NVIDIA GPUs is now widespread, but they will not play a significant role in the exascale computers that are planned by the Department of Energy. Time will tell!

In addition to the challenges listed above, it is also clear that keeping all of the computational resources busy with work (i.e., balancing the computational load across multiple heterogeneous computer architectures to avoid idle components) has become a much greater challenge than it was on homogeneous systems. Multiple chemistry codes have been turning to task-based management systems such as CHARM++[40] (NAMD[41] and OpenAtom[42]), MADWorld[43] (MADNESS,[43] MPQC,[44] and NWChemEx), and PaRSEC[45] (NWChem and GAMESS). In each of these codes, there is a runtime environment that can adapt to the machine parameters and execution to provide dynamic load balancing and fault tolerance—maximizing communication and execution overlap to decrease overall execution time. For example, MADWorld uses futures (i.e., anticipated possibilities for asynchronous execution later in the code) to enable asynchronous execution of tasks. The feature of directed task graphs is also common to task-based management tools and allows the programmer to break down work into appropriately sized tasks for the computational resource. The tasks can then be scheduled to run on available nodes and to migrate to nodes where the data is located—having the effect of optimizing the job to the architecture. For example, PARSEC[45] has been used to reduce load imbalances in the coupled cluster component of NWChem[46] and is currently being used for similar purposes in the GAMESS code.

As computer capabilities have expanded to the petascale and pre-exascale, power costs and power consumption have become important issues. Consequently, considerable effort has been invested in exploring potential solutions such as reducing clock speeds in ways that have minimal impact on time to solution.[47,48] Sosonkina has shown that if one lowers the clock speed when cores are idle, power consumption can be reduced at virtually no loss in computational speed.[49] In addition, time to solution and power consumption can both be decreased in certain cases by oversubscription methods where more processes than processors are requested for a given computation.[50]

Novel architectures, such as GPUs and other accelerators, and associated languages (e.g., CUDA) and middleware (MPI, OpenMP) are an important and necessary step toward advanced high-performance computing in quantum chemistry. An equally important necessity is the development of novel quantum chemistry software that can take advantage of the architectural advances. One broad category of methods that can take advantage of massively parallel computers is fragmentation approaches.[51] One type of fragmentation approach physically divides a molecular system into pieces (fragments) each of whose properties (e.g., energies) can essentially be computed on separate nodes, thereby taking advantage of coarse-grained parallelism. Since modern compute nodes have many cores, if the underlying quantum chemistry method (e.g., HF, MP2, CCSD(T)) is implemented with a parallel (fine-grained) algorithm, then the developer can take advantage of multilevel parallelism. One such method is the fragment molecular orbital (FMO) method,[52] a many-body expansion approach that can be terminated at one-body (monomers: FMO1), pairs of fragments (dimers: FMO2), etc. The higher in the expansion one goes, the greater the accuracy and computational cost of the calculation. In systems such as water, in which three-body interactions are important, FMO3 becomes essential. Analytic FMO2 and FMO3 gradients and Hessians have been implemented in GAMESS for both HF and DFT, and the method has been shown to scale linearly up to more than 250,000 cores.[53] Recently, a version of the FMO method has been merged with the semiclassical effective fragment potential (EFP) method to form the effective fragment molecular orbital (EFMO)[54] method. An advantage of EFMO is that it uses the EFP self-consistent induction term to capture many-body interactions, thereby avoiding the need to include explicit three-body interactions (i.e., FMO3). The computational bottleneck in EFMO is the computation of the dimer interactions. However, these dimer computations can be minimized by choosing a cutoff R_{cut} so as to ensure that most dimers are computed with the much less demanding EFP method.[55] This has little effect on the accuracy since EFP interactions are generally equivalent to that of MP2.[56,57]

A different approach to fragmentation divides the wave function, rather than groups of atoms into fragments. This is often accomplished via the use of localized molecular orbitals[51] (LMOs) that then facilitate the establishment of LMO domains upon which electron correlation can be built. The efficacy of LMO-based approaches was first demonstrated by Pulay[58] for MP2. Others, especially Werner and co-workers,[59] have extended this approach to coupled cluster theory. Likewise, Piecuch and co-workers developed the cluster-in-molecule (CIM)[60] approach based on localized orbitals and then reduced the computational demand even further by combining CIM with the FMO method so that one only needs to localize the subsets of fragment orbitals rather than the entire valence orbital space.[61] Another approach in this vein is reduced active space methods such as the ORMAS[62] (occupation restricted multiple active space) and RAS[63] (reduced active space) methods that divide MCSCF active spaces into more tractable subspaces. Recent developments in domain local paired natural orbitals have allowed CC computations for large systems at approximately the same order of cost as HF and DFT methods.[64] While reduced active space methods do not explicitly rely on the computation of LMOs, they do rely, as do most fragmentation methods, on the inherent locality of the systems to be studied.

One of the great revolutions that has occurred in chemistry over the past decade is the widespread use of better computer engineering techniques. Version control with git,[65] GitHub,[66] and/or GitLab[67] is common in almost all quantum chemistry codes. With appropriate control of merges to the master code, these tools have allowed for much faster development cycles and a facile ability to revert back to previous working versions as needed. In addition, the web-based tools provide many other features in computer engineering such as the ability to have automated building and unit testing, such as those provided by Jenkins[68] and Travis-CI,[69] software review, issue tracking, community input, and documentation harnesses. Of course, managing a large community-based code is still difficult, requiring active software management and engagement with the community. Training students to use all of these tools is both challenging and important. Fortunately, there are many great online resources. For the molecular sciences

https://dx.doi.org/10.1021/acs.chemrev.0c00700
Chem. Rev. 2020, 120, 9015−9020

community, the NSF Molecular Sciences Software Institute (MolSSI)[70,71] has taken on an extensive training mission to continue to engage the community in learning excellent computer engineering and programming skills in the context of common programming methods and paradigms used in the community. The community as a whole, especially the academic community, needs to recognize more broadly the importance of software engineering and include such endeavors in the academic reward system when it comes to promotion and tenure.

Mark S. Gordon orcid.org/0000-0001-6893-553X
Theresa L. Windus orcid.org/0000-0001-6065-3167

AUTHOR INFORMATION

Complete contact information is available at:
https://pubs.acs.org/10.1021/acs.chemrev.0c00700

Notes

Views expressed in this editorial are those of the authors and not necessarily the views of the ACS.

Biographies

Mark S. Gordon, Frances M. Craig Distinguished Professor of Chemistry at Iowa State University, was born and raised in New York City. After completing his B.S. in Chemistry in 1963, Professor Gordon entered the graduate program at Carnegie Institute of Technology, where he received his Ph.D. in 1967 under the guidance of Professor John Pople. Following a postdoctoral research appointment with Professor Klaus Ruedenberg at Iowa State University, Professor Gordon accepted a faculty appointment at North Dakota State University in 1970, where he rose through the ranks, eventually becoming distinguished professor and department chair. He moved to Iowa State University and Ames Laboratory in 1992. He is the Frances M. Craig Distinguished Professor of Chemistry. Professor Gordon's research interests are broadly based in electronic structure theory, computational science, and related fields. He has authored more than 640 research papers and is an elected member of the International Academy of Quantum Molecular Science. He received the 2009 ACS Award for Computers in Chemical and Pharmaceutical Research and the 2015 ACS Award for Theoretical Chemistry. He is a Fellow of the APS, the ACS, and the American Association for the Advancement of Science.

Theresa L. Windus is a Distinguished Professor of Chemistry at Iowa State University (ISU), an Associate with Ames Laboratory, an ISU Liberal Arts and Sciences Dean's Professor, and a Fellow of the American Chemical Society. Theresa received her B.A. degrees in chemistry, mathematics, and computer science from Minot State University. She then completed her Ph.D. in physical chemistry at Iowa State University, where she focused on developing high performance algorithms. Theresa has contributed to multiple chemistry packages and currently develops new methods and algorithms for high performance computational chemistry as the director of the NWChemEx project as well as applying those techniques to both basic and applied research.

ACKNOWLEDGMENTS

The authors have been supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, the Department of Energy Basic Energy Sciences Computational Chemical Sciences program, both of which are administered by Ames Laboratory, as well as by a National Science Foundation Software Infrastructure (SI2) grant (OCI-1047772).

REFERENCES

(1) Fifth Solvay International Conference (October 1927), quoted in *Physics and Beyond: Encounters and Conversations*; W. Heisenberg, 1971; pp 85−86.
(2) Atanasoff-Berry computer https://en.wikipedia.org/wiki/Atanasoff-Berry_computer (Accessed 2020-08-01).
(3) Top 500. The List. https://www.top500.org/ (accessed 2020-06-18).
(4) Alexander, F.; Almgren, A.; Bell, J.; Bhattacharjee, A.; Chen, J.; Colella, P.; Daniel, D.; DeSlippe, J.; Diachin, L.; Draeger, E.; et al. Exascale applications: skin in the game. *Philos. Trans. R. Soc., A* **2020**, *378*, 20190056.
(5) Asadchev, A.; Allada, V.; Felder, J.; Bode, B. M.; Windus, T. L.; Gordon, M. S. Uncontracted Rys Quadrature Implementation of up to g Functions on Graphical Processing Units. *J. Chem. Theory Comput.* **2010**, *6*, 696−704 and references therein.
(6) Asadchev, A.; Gordon, M. S. New Multithreaded Hybrid CPU/GPU Approach to Hartree-Fock. *J. Chem. Theory Comput.* **2012**, *8*, 4166−4176 and references therein.
(7) Tomlinson, D. G.; Asadchev, A.; Gordon, M. S. A New Approach to Second Order Perturbation Theory. *J. Comput. Chem.* **2016**, *37*, 1274−1282.
(8) Katouda, M.; Naruse, A.; Hirano, Y.; Nakajima, T. Massively Parallel Algorithm and Implementation of RI-MP2 Energy Calculation for Peta-Scale Many-Core Supercomputers. *J. Comput. Chem.* **2016**, *37*, 2623−2633.
(9) Snyder, J. W.; Hohenstein, E. G.; Luehr, N.; Martinez, T. J. An atomic orbital-based formulation of analytical gradients and non-adiabatic coupling vector elements for the state-averaged complete active space self-consistent field method on graphical processing units. *J. Chem. Phys.* **2015**, *143*, 154107.
(10) Asadchev, A.; Gordon, M. S. A Fast and Flexible Coupled Cluster Approach. *J. Chem. Theory Comput.* **2013**, *9*, 3385−3392.
(11) Advanced Micro Devices, Inc. HIP Porting Guide. https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-porting-guide.html (accessed 2020-06-18).
(12) https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/dpc-compiler.html (accessed 2020-06-18).
(13) Valeev, E. F. Libint: high-performance library for computing Gaussian integrals in quantum mechanics, https://github.com/evaleev/libint (accessed 2020-06-18).
(14) Hirata, S. Tensor contraction engine: abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *J. Phys. Chem. A* **2003**, *107*, 9887−9897.
(15) Mutlu, E.; Kowalski, K.; Krishnamoorthy, S. Toward generalized tensor algebra for ab initio quantum chemistry methods. *ARRAY 2019: Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming* **2019**, 46−56.
(16) Sanders, B. A.; Bartlett, R. J.; Deumens, E.; Lotrich, V.; Ponton, M. A Block-Oriented Language and Runtime System for Tensor Algebra with Very Large Arrays. *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* **2010**, 1−11.
(17) Leang, S. S.; Rendell, A. P.; Gordon, M. S. Quantum Chemical Calculations using Accelerators: Migrating Matrix Operations to the NVIDIA Kepler GPU and the Intel Xeon Phi. *J. Chem. Theory Comput.* **2014**, *10*, 908−912.
(18) Keipert, K.; Mitra, G.; Sunriyal, V.; Leang, S. S.; Sosonkina, M.; Rendell, A.; Gordon, M. S. Energy Efficient Computational Chemistry: Comparison of X86 and ARM Systems. *J. Chem. Theory Comput.* **2015**, *11*, 5055−5061.

(19) IBM, Quantum Starts Here. https://www.ibm.com/quantum-computing/ (accessed 2020-06-23).
(20) Google, Quantum, https://research.google/teams/applied-science/quantum/ (accessed 2020-06-23).
(21) NVLINK AND NVSWITCH https://www.nvidia.com/en-us/data-center/nvlink/ (accessed 2020-08-01).
(22) Smith, D. G. A.; Burns, L. A.; Sirianni, D. A.; Nascimento, D. R.; Kumar, A.; James, A. M.; Schriber, J. B.; Zhang, T.; Zhang, B.; Abbott, A. S. Psi4NumPy: An Interactive Quantum Chemistry Programming Environment for Reference Implementations and Rapid Development. *J. Chem. Theory Comput.* **2018**, *14*, 3504−3511.
(23) Barca, G. M. J.; Bertoni, C.; Fedorov, D. G.; Ivanic, J.; Leang, S. S.; Pham, B. Q.; Piecuch, P.; Slipchenko, L. V.; Xu, P.; Gordon, M. S.; et al. Recent Developments in the General Atomic and Molecular Electronic Structure System. *J. Chem. Phys.* **2020**, *152*, 154102.
(24) Molpro Quantum Chemistry Software https://www.molpro.net (accessed 2020-08-01).
(25) Molcas 8.4 http://www.molcas.org (accessed 2020-08-01).
(26) Apra, E.; Bylaska, E. J.; deJong, W. A.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Valiev, M.; van Dam, H. J. J.; Alexeev, Y.; Anchell, J.; et al. NWChem: Past, Present, and Future. *J. Chem. Phys.* **2020**, *152*, 184102.
(27) Q-CHEM 5.3 https://www.q-chem.com (accessed 2020-08-01).
(28) Richard, R. M.; Bertoni, C.; Boschen, J. S.; Keipert, K.; Pritchard, B.; Valeev, E. F.; Harrison, R. J.; de Jong, W. A.; Windus, T. L. Developing a Computational Chemistry Framework for the Exascale Era. *Comput. Sci. Eng.* **2019**, *21*, 48−58.
(29) Kim, J.; Baczewski, A. D.; Beaudet, T. D.; Benali, A.; Bennett, M. C.; Berrill, M. A.; Blunt, N. S.; Borda, E. J. L.; Casula, M.; Ceperley, M. D.; et al. QMCPACK: an open source *ab initio* quantum Monte Carlo package for the electronic structure of atoms, molecules and solids. *J. Phys.: Condens. Matter* **2018**, *30*, 19.
(30) https://www.msg.chem.iastate.edu/gamess/index.html.
(31) Poole, D.; Galvez Vallejo, J. L.; Gordon, M. S. A New Kid on the Block: Application of Julia to Hartree-Fock Calculations. *J. Chem. Theory Comput.* **2020**, *16*, 5006.
(32) PetaChem. http://www.petachem.com/products.html (accessed 2020-08-01).
(33) The MPI Forum, CORPORATE. MPI: A Message Passing Interface. *Proceedings of the 1993 ACM/IEEE conference on Supercomputing. Supercomputing '93*; ACM: Portland, Oregon, USA, pp 878−883.
(34) Guest, M. F.; Bush, I. J.; van Dam, H. J. J.; Sherwood, P.; Thomas, J. M. H.; van Lenthe, J. H.; Havenith, R. W. A.; Kendrick, J. The GAMESS-UK electronic structure package: algorithms, developments and applications. *Mol. Phys.* **2005**, *103*, 719−747.
(35) Lischka, H.; Shepard, R.; Pitzer, R. M.; Shavitt, I.; Dallos, M.; Müller, Th; Szalay, P. G.; Seth, M.; Kedziora, G. S.; Yabushita, S.; Zhang, Z. *Phys. Chem. Chem. Phys.* **2001**, *3*, 664.
(36) NVIDIA, NVIDIA Toolkit; https://developer.nvidia.com/cuda-toolkit (accessed 2020-06-18).
(37) HIP: C++ Heterogeneous-Compute Interface for Portability, https://github.com/ROCm-Developer-Tools/HIP (accessed 2020-06-18).
(38) The Kronos Group, Inc. SYCL overview. https://www.khronos.org/sycl/ (accessed 2020-06-18).
(39) Intel, Intel OneAPI Toolkits. https://software.intel.com/content/www/us/en/develop/tools/oneapi.html (accessed 2020-06-18).
(40) Kale, L. V.; Krishnan, S. CHARM++: A Portable Concurrent Object Oriented System Based on C++. In *Proceedings of OOPSLA'93*; Paepcke, A., Ed.; ACM Press, 1993; pp 91−108.
(41) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(42) OpenAtom. http://charm.cs.illinois.edu/OpenAtom/ (accessed 2020-06-18).
(43) Harrison, R. J.; Fann, G. I.; Yanai, T.; Gan, Z.; Beylkin, G. Multiresolution quantum chemistry: Basic theory and initial applications. *J. Chem. Phys.* **2004**, *121*, 11587−11598.
(44) Valeev, E. F. MPQC, http://github.com/ValeevGroup/mpqc (accessed 2020-06-18).
(45) Bosilca, G.; Bouteiller, A.; Danalis, A.; Herault, T.; Lemariner, P.; Dongarra, J. DAGuE: A Generic Distributed DAG Engine for High Performance Computing. *Proceedings of the Workshops of the 25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2011 Workshops)* **2011**, 1151−1158.
(46) McCraw, H.; Danalis, A.; Herault, T.; Bosilca, G.; Dongarra, J.; Kowalski, K.; Windus, T. L. Utilizing Dataflow-based Execution for Coupled Cluster Methods. *IEEE Cluster 2014*, Madrid, Spain, September 22−26, 2014, IEEE Cluster 2014 proceedings, pp 296−297.
(47) Sundriyal, V.; Sosonkina, M. Runtime power-aware energy-saving scheme for parallel applications. *Int. J. of High Performance Systems Architecture* **2017**, *7*, 129.
(48) Sundriyal, V.; Fought, E.; Sosonkina, M.; Windus, T. L. Evaluating Effects of Application Based and Automatic Energy Saving Strategies on NWChem. *Proc. 25th HPC Symposium, HPC '17* **2017**, *16*, 1−12.
(49) Sundriyal, V.; Sosonkina, M.; Poole, D.; Gordon, M. S. Runtime Power Allocation Approach for GAMESS Hybrid CPU/GPU Implementation. *Concurrency and Computation* DOI: 10.1002/cpe.5917.
(50) Fought, E. L.; Sundriyal, V.; Sosonkina, M.; Windus, T. L. Improving Efficiency of Semi-Direct Møller-Plesset Second Order Perturbation Methods Through Oversubscription on Multiple Nodes. *J. Comput. Chem.* **2019**, *40*, 2146−2157.
(51) Gordon, M. S.; Fedorov, D. G.; Pruitt, S. R.; Slipchenko, L. V. Fragmentation Methods: A Route to Accurate Calculations on Large Systems. *Chem. Rev.* **2012**, *112*, 632−672.
(52) Kitaura, K.; Ikeo, E.; Asada, T.; Nakano, T.; Uebayasi, M. Fragment molecular orbital method: an approximate computational method for large molecules. *Chem. Phys. Lett.* **1999**, *313*, 701−706.
(53) Pruitt, S. R.; Nakata, H.; Nagata, T.; Mayes, M.; Alexeev, Y.; Fletcher, G. D.; Fedorov, D. G.; Kitaura, K.; Gordon, M. S. The Importance of Three-Body Interactions in Molecular Dynamics Simulations of Water. *J. Chem. Theory Comput.* **2016**, *12*, 1423−1435.
(54) Pruitt, S. R.; Steinmann, C.; Jensen, J. H.; Gordon, M. S. A Fully Integrated Fragment Molecular Orbital/Effective Fragment Potential Method. *J. Chem. Theory Comput.* **2013**, *9*, 2235−2249.
(55) Xu, P.; Guidez, E. B.; Bertoni, C.; Gordon, M. S. Perspective: Ab Initio Force Field Methods Derived From Quantum Mechanics. *J. Chem. Phys.*, **2018**, *148*, 090901. .
(56) Slipchenko, L. V.; Sherrill, C. D. Accurate prediction of noncovalent interaction energies with the effective fragment potential method". *J. Chem. Theory Comput.* **2012**, *8*, 2835−2843.
(57) Edmiston, C.; Ruedenberg, K. Localized Atomic and Molecular Orbitals. *Rev. Mod. Phys.* **1963**, *35*, 457.
(58) Pulay, P.; Saebo, S. Orbital-invariant formulation and second-order gradient evaluation in M?ller-Plesset perturbation theory. *Theor. Chim. Acta* **1986**, *69*, 357−368.
(59) Ma, Q.; Werner, H.-J. Explicitly Correlated Coupled-Cluster Methods Using Pair Natural Orbitals. *WIREs* **2018**, *8*, No. e1371.
(60) Li, W.; Piecuch, P. Improved Design of Orbital Domains within the Cluster-in-Molecule Local Correlation Framework: Single-environment Cluster-in-Molecule Ansatz and its Application to Local Coupled-Cluster Approach with Singles and Doubles. *J. Phys. Chem. A* **2010**, *114*, 8644−8657.
(61) Findlater, A.; Zahariev, F.; Gordon, M. S. A Combined Fragment Molecular Orbital-Cluster in Molecule Approach to Massively Parallel Electron Correlation Calculations for Large Systems. *J. Phys. Chem. A* **2015**, *119*, 3587−3593.
(62) Ivanic, J. Direct configuration interaction and multiconfigurational self-consistent-field method for multiple active spaces with variable occupations. I. Method. *J. Chem. Phys.* **2003**, *119*, 9364.

https://dx.doi.org/10.1021/acs.chemrev.0c00700
Chem. Rev. 2020, 120, 9015−9020

(63) Malmqvist, P. A.; Rendell, A.; Roos, B. O. The complete active space self-consistent field method implemented with a split graph unitary group approach. *J. Phys. Chem.* **1990**, *94*, 5477−5482.
(64) Saitow, M.; Becker, U.; Riplinger, C.; Valeev, E. F.; Neese, F. A new near-linear scaling, efficient and accurate, open-shell domain-based local pair natural orbital coupled cluster singles and doubles theory. *J. Chem. Phys.* **2017**, *146*, 164105.
(65) git. https://git-scm.com/ (accessed 2020-06-18).
(66) GitHub. https://github.com/ (accessed 2020-06-18).
(67) GitLab. https://about.gitlab.com/ (accessed 2020-06-18).
(68) Jenkins. https://www.jenkins.io/ (accessed 2020-06-18).
(69) Travis CI. https://travis-ci.com/ (accessed 2020-06-18).
(70) Krylov, A.; Windus, T. L.; Barnes, T.; Marin-Rimoldi, E.; Nash, J. A.; Pritchard, B.; Smith, D. G. A.; Altarawy, D.; Saxe, P.; Clementi, C.; et al. Computational Chemistry Software and Its Advancement: Three Grand Challenge Cases for Computational Molecular Science. *J. Chem. Phys.* **2018**, *149*, 180901.
(71) Ringer McDonald, A.; Nash, J. A.; Nerenberg, P. S.; Ball, K. A.; Sode, O.; Foley, J. J., IV; Windus, T. L.; Crawford, T. D. Building Capacity for Undergraduate Education and Training in Computational Molecular Science: A Collaboration Between the MERCURY Consortium and the Molecular Sciences Software Institute. *Int. J. Quantum Chem.* **2020**, DOI: 10.1002/qua.26359.

https://dx.doi.org/10.1021/acs.chemrev.0c00700
Chem. Rev. 2020, 120, 9015−9020