CHEMICAL REVIEWS

pubs.acs.org/CR

Review

Many-Body Quantum Chemistry on Massively Parallel Computers

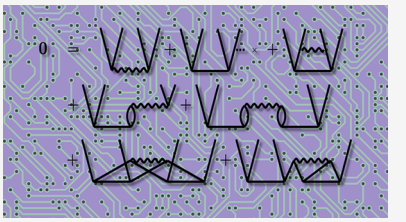
Justus A. Calvin, Chong Peng, Varun Rishi, Ashutosh Kumar, and Edward F. Valeev*

Cite This: https://dx.doi.org/10.1021/acs.chemrev.0c00006

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: The deployment of many-body quantum chemistry methods onto massively parallel high-performance computing (HPC) platforms is reviewed. The particular focus is on highly accurate methods that have become popular in predictive description of chemical phenomena, such as the coupled-cluster method. The account of relevant literature is preceded by a discussion of the modern and near-future HPC landscape and the relevant computational traits of the many-body methods, in their canonical and reduced-scaling formulations, that underlie the challenges in their HPC realization.



CONTENTS

1. Introduction A
2. Overview of the Many-Body Formalisms of Quantum Chemistry C
 2.1. Notation and Conventions C
 2.2. Coupled-Cluster and Many-Body Perturbation Theory C
  2.2.1. Ground-State Formalism C
  2.2.2. Excited States and Properties D
  2.2.3. Higher-Order Coupled-Cluster Methods and Approximations E
 2.3. Numerical Representation and Approximations E
  2.3.1. Explicitly Correlated Many-Body Methods E
  2.3.2. Integral Approximations E
 2.4. Reduced-Scaling Formulations F
3. Computational Traits of Many-Body Methods G
 3.1. Traditional Many-Body Methods G
 3.2. Reduced-Scaling Many-Body Methods I
4. Overview of High-Performance Computing Platforms J
5. Deployment of Many-Body Quantum Chemistry to Parallel Computers K
 5.1. Many-Body Perturbation Theory K
 5.2. CCSD L
  5.2.1. Pioneering Work L
  5.2.2. NWChem L
  5.2.3. PQS M
  5.2.4. CFOUR M
  5.2.5. Molpro M
  5.2.6. GAMESS M
  5.2.7. ACES M
  5.2.8. CTF N
  5.2.9. MPQC N
  5.2.10. FHI-aims O
  5.2.11. MRCC O
 5.3. CCSD(T) O
  5.3.1. Pioneering Work O
  5.3.2. NWChem O
  5.3.3. GAMESS P
  5.3.4. ACES P
  5.3.5. PQS P
  5.3.6. MPQC P
  5.3.7. FHI-aims P
  5.3.8. MRCC P
 5.4. CC Excited States and Properties P
  5.4.1. NWChem P
  5.4.2. ACES Q
  5.4.3. MPQC Q
 5.5. Higher-Order CC Q
 5.6. Reduced-Scaling CC R
 5.7. Heterogeneous Platforms R
  5.7.1. NWChem S
  5.7.2. CTF S
  5.7.3. MPQC S
6. Outlook S
Author Information T
 Corresponding Author T
 Authors T
 Notes T
 Biographies T
Acknowledgments U
References U

Received: January 3, 2020

1. INTRODUCTION

The goal of this article is to review the progress in deploying accurate *many-body* electronic structure methods of interest to chemists onto modern and emerging parallel computer platforms. The many-body electronic structure methods, by describing explicitly the correlations between states of two or more electrons in an *n*-electron system, are computationally more costly than the mainstream Kohn−Sham density functional theory (DFT) methods,[1,2] but can surpass the limitations of DFT by allowing systematic improvement toward the exact solution. The many-body electronic structure methods in use in chemistry, physics, and materials science can be loosely grouped according to their computational and formal traits into methods related to the many-body perturbation theory (e.g., coupled-cluster, Green's function), quantum Monte Carlo methods, tensor network methods, and combinations thereof. While each group of methods has strengths and weaknesses, methods based on many-body perturbation theory, and in particular the coupled-cluster method,[3−5] have had the most impact so far in the field of chemistry, enabled by efficient computer implementations nowadays available in most fully featured open-source and commercial quantum chemistry packages (ACES,[6] CFOUR,[7] Molpro,[8] Orca,[9] Q-Chem,[10] Turbomole,[11] Gaussian,[12] GAMESS,[13] and NWChem,[14] just to name a few prominent examples); thus, our focus will be almost exclusively on such methods.

Two key factors limit the usefulness of the accurate many-body electronic structure methods, like coupled-cluster, to practical problems of interest to experimentalists:

- First, the rapid (high-order polynomial) growth of the computational cost with the molecular size *N*. For example, the conventional coupled cluster singles and doubles (CCSD)[15] and CCSD with perturbative treatment of triples [CCSD(T)][16] methods have computational complexity of $O(N^6)$ and $O(N^7)$, respectively. Hence, doubling the system size increases the computational expense by roughly 2 orders of magnitude.
- Second, the slow convergence of the correlation energy with respect to the size of the atomic orbital basis set. It is known empirically that huge basis sets are required (100+ basis functions/atom) to reduce the basis set error of standard wave functions such as CCSD to "chemical" accuracy levels. Furthermore, decreasing the basis set error through brute force is futile because the computational cost of such methods depends on the basis set error ϵ as approximately $O(\epsilon^{-4})$.[17,18] Hence, reducing the basis set error by a factor of 2 increases the computational expense by more than an order of magnitude.

The high computational costs of the many-body methods can be overcome somewhat by taking advantage of modern massively parallel computers, but this is barely a remedy: even a million-fold increase in concurrency would allow an increase of the system size by only a factor of 10. Fortunately, the high polynomial complexities with size and precision issues of the many-body methods were solved to a certain extent in recent years by the emergence of efficient reduced-scaling formulations of these methods and the explicitly correlated formalisms, respectively. This opens a pathway to rapid increase in the utilization of many-body methods as an alternative to the mainstream DFT methods in everyday computational chemistry experiments. However, to realize the great promise of reduced-scaling many-body methods, they need to be deployed efficiently to modern computer platforms to compete with DFT on cost while offering much greater accuracy.

The deployment of the most recent advances in the many-body electronic structure toolkit onto modern hardware can to some appear successful, as evidenced by the presence of robust many-body capabilities in most comprehensive quantum chemistry software suites, with most capable of efficient execution of modern multicore processors and some even on heterogeneous platforms. On the other hand, the rate of deployment is unfortunately slow and does not match the rate of evolution of the hardware platforms and the theories themselves. This can be partially attributed to the complexity of the many-body formalisms and algorithms, both conventional and reduced-scaling, and partially to the increasing complexity of the high-performance computing (HPC) platforms themselves. Modern HPC platforms are characterized by critical importance of data parallelism, many execution units within each memory domain, complex memory hierarchies, and heterogeneity (increasingly so at the low end, almost exclusively so at the high end). Tackling this complexity is made harder by the relative immaturity and rapid evolution of the programming models, particularly within the node; the asynchronous nature of the intranode programming today is a massive change from the way most computational scientists are used to program.

As we continuously need to adapt our implementation, algorithms, and even electronic structure methods themselves to the realities of the rapidly evolving HPC hardware, it is useful to review the evolution of many-body methods on the high-end hardware of years past. This is the main objective of this review. As argued above, our primary (but not exclusive) focus will be on the many-body methods that evolved in the chemistry context (coupled-cluster and perturbation theory). Of course, the many-body methods of chemistry are also used in other contexts (e.g., nuclear physics), and furthermore, their computational traits are shared by related methods like Green's function methods less common in chemistry but very popular in physics. Because the efficient (i.e., early crossover) reduced-scaling many-body formalisms have emerged relatively recently, much of the prior work that we discuss will deal with the conventional (full-scaling) variants of these methods. The emergence of the reduced-scaling methods does not make the full-scaling methods obsolete. First, there is still the need to be able to benchmark the reduced-scaling methods. Second, the conventional methods become faster than the reduced-scaling counterparts as the target precision increases. Third, the advantages of reduced-scaling formalisms may not hold for all molecular properties equally. Lastly, in some contexts, like high-order coupled-cluster methods, it is not obvious whether the current reduced-scaling methodologies will be viable anyway. Thus, it is essential to understand the computational traits of both full- and reduced-scaling formalisms and how they relate to the hardware evolution.

The primary focus of this review will be on distributed-memory and accelerated (e.g., heterogeneous) computed platforms that are representative of the modern and likely future HPC environments. Albeit the vast majority of chemical applications of the many-body methods to date have not used either distributed-memory or accelerated platforms, such HPC platforms are becoming (or, rather, have become) the norm rather than the exception for most users, primarily due to their

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

greater price-to-performance ratio and better power efficiency. Thus, in the near future, the practitioners of many-body electronic structure methods will expect to be able to routinely and efficiently utilize what may appear to some as esoteric HPC platforms. Lastly, we decided to group together the discussion of distributed-memory architectures and accelerated platforms due to the great overlap in considerations that factor into efficient programming of both; namely, explicit management of data and computation flow across multiple disjoint execution and memory spaces, the need to optimize for data movement and load balance, and the importance of balancing task granularity and performance, among many. Increasingly, the same considerations apply even to programming modern multisocket CPU nodes. Thus, the lessons learned from deploying conventional many-body methods on massively parallel clusters of yesteryear will continue to be relevant to deployment of modern reduced-scaling many-body formalisms onto the complex HPC platforms of today and tomorrow.

2. OVERVIEW OF THE MANY-BODY FORMALISMS OF QUANTUM CHEMISTRY

To discuss the algorithmic and implementation aspects of the many-body quantum chemistry, it is necessary to introduce, however briefly, its formalism. Because the main focus of the review is on the technical developments, here we discuss only the essential elements of the many-body formalism that are relevant to the discussion of the literature. For details, interested readers are referred to the original literature as well as the relevant monographs and reviews.

2.1. Notation and Conventions

In this work we largely follow the long-established[19,20] tensor notation of the many-body quantum chemistry. Einstein summation convention will be implied, unless noted otherwise: namely, summation is implied over every symbol that appears once in a contravariant (upper) position and once as in a covariant (lower) position in a given tensor product, with the summation range defined by the symbol.

The following convention for the orbital spaces and their index labels will be utilized in this paper. i, j, k, and l will denote active occupied orbitals (i.e., orbitals that occur in the reference determinant and are correlated); their rank will be denoted o. m and n will denote all occupied orbitals. The union of occupied orbitals with an orthogonal complement of unoccupied (or virtual) orbitals a, b, c, and d (rank = v) will be denoted by p, q, r, and s (rank = n); in the textbook formulation of many-body theory, these are typically represented by canonical Hartree−Fock orbitals. κ, λ, μ, and ν will denote a formal complete set of orbitals that includes the $\{p\}$ orbital set. The orbitals in $\{\kappa\}$ that are not occupied will be denoted α, β, and γ. Automatically generated equations will utilize additional positive integer subscripts to distinguish indices from the same space, e.g., a_1 and a_2 will refer to two vectors from the unoccupied orbital set.

Dependent orbital spaces, such as orbital- and orbital-pair-specific subsets ("domains") of projected atomic orbitals, pair-natural orbitals, etc., occur as subsets or linear transformations of a base space. They will be denoted by annotating the base space index with one or more indices on which they depend on. The annotation will be shown, as is traditional in the literature, as subscript or, in automatically generated equations, as superscript.

Matrix elements of one-, two-, and higher-body operators will be denoted as follows:

$$\langle p|\hat{o}(1)|q\rangle \equiv \int \phi_p^*(1)\hat{o}(1)\phi_q(1)\mathrm{d}1 \equiv o_p^q \tag{1}$$

$$\langle p_1p_2|\hat{o}(1,2)|q_1q_2\rangle \equiv \int \phi_{p_1}^*(1)\phi_{p_2}^*(2)\hat{o}(1,2)\phi_{q_1}(1)\phi_{q_2}(2)\mathrm{d}1\mathrm{d}2 \equiv o_{p_1p_2}^{q_1q_2}, \text{ etc.} \tag{2}$$

All operators encountered in this article are particle-symmetric, i.e. $o(1,2) = o(2,1)$, hence $o_{q_1q_2}^{p_1p_2} = o_{q_2q_1}^{p_2p_1}$. The antisymmetrized versions of the two- and higher-body operators will be denoted by an overbar, etc.:

$$\bar{o}_{q_1q_2}^{p_1p_2} \equiv o_{q_1q_2}^{p_1p_2} - o_{q_2q_1}^{p_1p_2} = o_{q_1q_2}^{p_1p_2} - o_{q_1q_2}^{p_2p_1} \tag{3}$$

Matrix elements of the one- and two-particle parts of the Hamiltonian and the Fock operator (of the Hartree−Fock method) will be denoted by h_λ^κ, $g_{\mu\nu}^{\kappa\lambda}$, and f_λ^κ, respectively. Overlap (inner product) of two functions will be denoted by $S_p^q \equiv \langle p|q\rangle$.

The index set antisymmetrizers will be defined as follows:

$$\hat{A}_{p_1p_2}f(p_1,p_2) \equiv f(p_1,p_2) - f(p_2,p_1) \tag{4}$$

$$\hat{A}_{p_1p_2p_3}f(p_1,p_2,p_3) \equiv f(p_1,p_2,p_3) - f(p_2,p_1,p_3) - f(p_1,p_3,p_2) - f(p_3,p_2,p_1) + f(p_2,p_3,p_1) + f(p_3,p_1,p_2), \text{ etc.} \tag{5}$$

Composite antisymmetrizers will also be used:

$$\hat{A}_{p_1p_2}^{q_1q_2} \equiv \hat{A}_{p_1p_2}\hat{A}_{q_1q_2} \tag{6}$$

The convention in the quantum chemistry literature is to only include multiplies in floating point operation (FLOP) counts, not the additions (due to the lower effective throughput of multiplies on older HPC platforms). To be able to compare the measured FLOPS (1 FLOPS ≡ 1 FLOP per second) to the hardware peak performance, we will count both multiplies and additions [a pair of these is typically implemented in modern hardware as a single fused multiply add (FMA) instruction]. Thus, for example, the naïve algorithm for multiplication of two square matrices of size n will involve $2n^3$ FLOPs. In discussing the asymptotic computational costs we will use the standard "Big O" notation convention used in the quantum chemistry literature which is identical to the Θ notation used in the computer science literature, namely that:

$$O(g(n)) = \{f(n): \exists\, k_1 > 0, \exists\, k_2 > 0, \exists\, n_0, \forall\, n > n_0, k_1g(n) \le f(n) \le k_2g(n)\}$$

The ranks of all independent index spaces are assumed to grow proportionally to the system size N; thus, for example, $o^2v^4 = O(N^6)$. The meaning of generalized "Big O" notation, such as $O(o^2v^4)$, is also clear.

2.2. Coupled-Cluster and Many-Body Perturbation Theory

2.2.1. Ground-State Formalism. Our narrative starts with a quick introduction to the coupled-cluster (CC) method; the

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

reader is referred to the existing reviews[21,22] and monographs[5,23] for complete discussion.

In its standard single-reference formulation, the coupled-cluster wave function for a system of N electrons is obtained by the action of exponentiated cluster operator $\hat{T}$ onto the reference determinant $|0\rangle$:

$$|\Psi_{CC}\rangle = e^{\hat{T}}|0\rangle \tag{7}$$

The cluster operator is parametrized to be able to connect the reference determinant $|0\rangle$ to every determinant producible from the given finite (orthonormal) basis of orbitals (one-electron states); thus, it is written systematically as a linear combination of operators that substitute 1, 2, ..., N orbitals in the reference determinant:

$$\hat{T} \equiv \hat{T}_1 + \hat{T}_2 + ... + \hat{T}_N \tag{8}$$

$$\hat{T}_1 \equiv t_a^i a_a^\dagger a_i \tag{9}$$

$$\hat{T}_2 \equiv \frac{1}{(2!)^2} t_{ab}^{ij} a_a^\dagger a_b^\dagger a_j a_i \tag{10}$$

$$\hat{T}_3 \equiv \frac{1}{(3!)^2} t_{abc}^{ijk} a_a^\dagger a_b^\dagger a_c^\dagger a_k a_j a_i, \text{ etc.} \tag{11}$$

the cluster amplitudes t_a^i, t_{ab}^{ij}, t_{abc}^{ijk}, ... are thus classified according to their rank as one-body or singles ("S"), two-body, or doubles ("D"), three-body, or triples ("T"), and so on. Fermionic annihilators a_κ and creators $a^\kappa \equiv a_\kappa^\dagger$ remove and add the corresponding orbitals from the determinants with the usual canonical anticommutator relations,

$$[a_\kappa, a_\lambda]_+ = 0 \tag{12}$$

$$[a_\kappa, a_\lambda^\dagger]_+ = \delta_\kappa^\lambda \tag{13}$$

enforcing the Fermi−Dirac statistics. In the traditional formulation of the coupled-cluster method, the energy and cluster amplitudes are determined by the following projections:

$$\langle 0|\bar{H}|0\rangle = E_{CC} \tag{14}$$

$$\langle S|\bar{H}|0\rangle = 0 \tag{15}$$

$$\langle D|\bar{H}|0\rangle = 0 \tag{16}$$

$$\langle T|\bar{H}|0\rangle = 0, \text{ etc.} \tag{17}$$

where S, D, and T denote projections on all singly-, doubly-, and triply-substituted determinants. In eqs 14−17 we introduced the similarity-transformed Hamiltonian,

$$\bar{H} \equiv e^{-\hat{T}} \hat{H} e^{\hat{T}} \tag{18}$$

where:

$$\hat{H} \equiv h_\lambda^\kappa a_\lambda^\dagger a_\kappa + \frac{1}{4} g_{\mu\nu}^{\kappa\lambda} a_\mu^\dagger a_\nu^\dagger a_\lambda a_\kappa \tag{19}$$

The programmable version of the CC equations are derived straightforwardly by expanding $\bar{H}$ as

$$\bar{H} = \hat{H} + [\hat{H}, \hat{T}] + \frac{1}{2!}[[\hat{H}, \hat{T}], \hat{T}] + \frac{1}{3!}[[[\hat{H}, \hat{T}], \hat{T}], \hat{T}] + \frac{1}{4!}[[[[\hat{H}, \hat{T}], \hat{T}], \hat{T}], \hat{T}] \tag{20}$$

where the series truncates at the fourfold commutator due to the quartic dependence of the Hamiltonian (eq 19) on the creators/annihilators. The matrix elements in the coupled-cluster equations are then evaluated by the application of the canonical anticommutation relations or related techniques (Wick's theorem, diagrammatics, etc.). Additional simplifications such as tracing out the spin degrees of freedom can be performed on the programmable equations[24] directly or by starting with a spin-adapted ansatz for the cluster operator;[25−28] spin-adaptation for open-shell states is in general far more complicated.[29] Even more complex is the extension of these ideas to the case of multideterminantal reference state: although implementations of the multireference coupled-cluster methods have been long explored,[30−38] but the practical use is largely limited to the linearized coupled-cluster approximations[39,40] as well the related methods of multi-reference perturbation theory[41−43] and multireference configuration interaction.[44]

The key feature of the coupled-cluster method for chemistry is its rapid and systematic convergence toward the exact solution when the reference state is a good approximation to the exact ground-state wave function; this is in stark contrast with the older configuration interaction (CI) method which employs linear (in $\hat{T}$) rather than the exponential parametrization. Specifically, the magnitude of the coupled-cluster amplitudes decreases with their rank roughly geometrically,[45,46] and chemically accurate energies and other properties are often obtained with cluster operator including up to three-body amplitudes; thus, the methods of primary interest to chemists span CCSD ("coupled-cluster singles, doubles"),[15] CCSDT,[47] and CCSDTQ.[48] Unfortunately, the computational cost of these truncated coupled-cluster methods, when formulated in the conventional form, grows with the system size N as $O(N^6)$, $O(N^8)$, and $O(N^{10})$, respectively. The high computational complexity of the coupled-cluster methods is related to their high storage complexity, defining the number of computational intermediates that appear in the method, namely $O(N^4)$, $O(N^6)$, and $O(N^8)$. Thus, reductions in scaling via the reduced-scaling formalisms, discussed later, are essential for the chemical application of these methods.

2.2.2. Excited States and Properties. Excited eigenstates and transition properties can be obtained via the linear response formalism,[49−51] the related equation of motion (EOM) CC formalism[52] (independently developed under the name "symmetry-adapted-cluster configuration interaction"[53]), or via the propagator approach.[54] Due to the computational similarity of these formalisms, we will only discuss the EOM-CC methods. Only a brief recap of the formalism is given here; the reader is referred to the aforementioned reviews and monographs of coupled-cluster as many excellent reviews dedicated to the EOM[52,55] and response CC formalisms.[56]

The (right-hand) kth excited-state wave function is obtained in a CI fashion by the action of a linear excitation operator acting on the ground-state CC wave function:

$$|k\rangle \equiv \hat{R}_{(k)} e^{\hat{T}}|0\rangle \tag{21}$$

where $\hat{R}_{(k)}$ is parametrized analogously to the cluster operator $\hat{T}$ (eqs 8−11):

$$\hat{R}_{(k)} \equiv \delta_{k0} + \hat{R}_{1(k)} + \hat{R}_{2(k)} + ... + \hat{R}_{N(k)} \tag{22}$$

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

$$\hat{R}_{1(k)} \equiv (r_{(k)})^{i}_{a} a^{\dagger}_{a} a_{i} \tag{23}$$

$$\hat{R}_{2(k)} \equiv \frac{1}{(2!)^2} (r_{(k)})^{ij}_{ab} a^{\dagger}_{a} a^{\dagger}_{b} a_{j} a_{i}, \text{ etc.} \tag{24}$$

where δ_{k0} is the Kronecker delta. Extension to particle number nonconserving $\hat{R}_{(k)}$ is also straightforward, leading to the Fock-space extensions (EOM-IP-CC, EOM-EA-CC, etc.). $\hat{R}_{(k)}$ and the corresponding energies $E_{(k)}$ are obtained by diagonalizing the similarity-transformed Hamiltonian (eq 18):

$$\bar{H}\hat{R}_{(k)}|0\rangle = E_{(k)}\hat{R}_{(k)}|0\rangle \tag{25}$$

In practice, the diagonalization uses Krylov subspace methods, with the time-determining step being evaluation of the action of the similarity-transformed Hamiltonian onto each Krylov subspace vector. Similar considerations apply to the response and propagator approaches, in which application of $\bar{H}$-like linear operator to the guess vector is the time-consuming step.

For computational feasibility, most applications limit expansion eq 22 to singles and doubles only (EOM-CCSD). The storage and operation complexities of the CCSD and EOM-CCSD methods are $O(N^4)$ and $O(N^6)$, respectively, with prefactors linear in the Krylov subspace size and with the number of solutions (roots) sought.

2.2.3. Higher-Order Coupled-Cluster Methods and Approximations. The coupled-cluster methods limited to the double substitutions are not sufficiently accurate to describe energies and other properties with sufficient accuracy for chemical applications. The introduction of triples is therefore mandatory, but comes with a steep increase in the storage ($O(N^6)$) and operation ($O(N^8)$) complexities compared to its lower-order counterparts. The most popular treatment of triples in the coupled-cluster framework is by perturbation theory with respect to the CCSD reference state; this gives rise to the CCSD(T) method[16] and other related methods.[57−59] The computational complexity of these methods is $O(N^7)$, i.e. lower than that of the full CCSDT method, and more importantly, the storage of three-body amplitudes is avoided, thus the storage complexity is the same as that of CCSD. Similar extensions of CCSDT, namely CCSDT(Q), are also available.[60] A rich variety of iterative approximations to CCSDT have also been considered, with $O(N^6)$ storage complexity and $O(N^7)$ computational complexity.[61−63] Similar approximate treatment of high-order clusters for excited states and response properties have also been considered.[64−68]

2.3. Numerical Representation and Approximations

2.3.1. Explicitly Correlated Many-Body Methods. The basis set problem of the traditional many-body methods can be efficiently addressed by the explicit use of the interelectronic distances to express the wave function/operator. For example, in the context of CC, this can be achieved by adding explicitly correlated terms to the cluster operator:

$$\hat{T}_{\mathrm{F12}} \equiv \frac{1}{(2!)^2} (R^{ij}_{\alpha\beta} a^{\dagger}_{\alpha} a^{\dagger}_{\beta} a_{j} a_{i} - R^{ij}_{ab} a^{\dagger}_{a} a^{\dagger}_{b} a_{j} a_{i}) \tag{26}$$

with R denoting integrals of a correlation function (a spherically symmetric function of interparticle distance). The role of these terms is to approximate the analytic cusp behavior of exact eigenstates at short interelectronic distances.[69,70] Inclusions of such terms in the electronic wave functions is almost as old as the quantum mechanics itself.[71] Due to the appearance of the numerous and expensive 3- and higher-body integrals (versus the 2-body integrals in the conventional methods), practical application to molecules did not begin until Kutzelnigg proposed the R12 formalism[72] in which such integrals are treated approximately. Further improvements of the original ideas[73−76] culminated in the F12 formalism[77−79] which can robustly decrease the basis set error of wave function and Green's function many-body methods.

The explicitly correlated R12/F12 formalism of the coupled-cluster method was first developed in simplified form by Noga, Kutzelnigg, and others.[80,81] More rigorous modern F12 variants of CCSD and higher-order CC were realized with the help of specialized computer algebra systems.[82−85] Although the F12 extensions of CCSD does not change its storage or computational complexity, due to the dramatically greater cost of the rigorous formulation of the CCSD-F12 compared to CCSD, practical incorporation of the F12 terms into the coupled-cluster hierarchy must involve approximations. Iterative approximations to CCSD-F12, such as the CCSD(F12),[86] CCSD-F12{a,b},[87] CCSD(F12*), and CCSD-[F12][88] variants, retain only the most essential F12 terms in the CCSD-F12 amplitude equations. In contrast, perturbative approximations to CCSD-F12 are obtained by low-order expansion of the CCSD-F12 Lagrangian with respect to the CCSD zeroth-order state.[89] Both styles of approximations have similar costs (albeit CCSD(F12) is substantially more expensive than others) and their performance is comparable. For chemical energy differences (atomization energies, reaction energies, reaction barrier heights) and noncovalent interaction energies, the use of explicitly correlated terms in CC results in a basis set error reduction that approximates that the effect of increasing by two the cardinal number of the basis set (two for double-ζ, three for triple-ζ, etc.), thus resulting in savings of one to two orders of magnitude.[90,91]

2.3.2. Integral Approximations. Significant computational advantages can be gained in many-body methods by various approximations to the matrix elements of the Hamiltonian (eq 19), specifically the two-electron integrals $g^{\kappa\lambda}_{\rho\sigma}$. Although the motivation of these approximations is best understood from the perspective of approximating individual integrals, for our purposes it is sufficient to view these approximations as various factorizations (or more generally, decompositions[92]) of the two-electron Coulomb integral tensor.

The oldest of such approximations is the density-fitting (DF) approximation, also for historical reasons known in quantum chemistry as the resolution of the identity (RI). In the DF approximation, products of two or more orbitals are approximated by a linear combination of density-fitting (also referred to generically as auxiliary) basis functions:[93−95]

$$g^{pq}_{rs} \overset{\mathrm{DF}}{\approx} C^{pX}_{r} g_{XY} C^{qY}_{s} = \sum_{Z} (C^{pX}_{r} (\mathbf{g}^{1/2})_{XZ})(\mathbf{g}^{1/2})_{ZY} C^{qY}_{s}) \tag{27}$$

$$= \sum_{Z} B^{pZ}_{r} B^{qZ}_{s} \tag{28}$$

where the density-fitting coefficients C^{pX}_{r} are determined by solving a least-squares problem at the $O(N^4)$ cost. Thus, from the tensor factorization perspective, DF represents the order-4 Coulomb integral tensor as a contraction of two order-3 tensors. In the basis of molecular orbitals, evaluation of the

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

Coulomb integrals and their DF evaluation both cost $O(N^5)$. Thus, the use of DF in many-body methods does not lower their overall complexity. However, it can significantly lower their prefactor, as is the case of DF-MP2,[96,97] and DF can help reduce the complexity of some (but not all) terms in the CCSD[98] and other methods. Another use case for DF is the computations utilizing basis sets including AOs with high angular momenta, in which the cost of the AO integral evaluation contributes significantly to the cost of MO integral evaluation, the DF approximation can greatly reduce the overall cost; this is why, for example, the F12 methods in practice require the use of the DF approximation.[99] The use of local DF is also crucial in the context of some reduced-scaling coupled-cluster methods (see below).[100−106] Careful engineering of the density-fitting basis sets[107,108] allows to make the DF error sufficiently small and systematically improvable, and largely canceling in practically relevant energy differences.

Similar in spirit to DF is the use of Cholesky decomposition of the Coulomb integral tensor g_{rs}^{pq} represented as an pr by qs matrix (matricized).[109−111] The apparent connection between the two techniques is obvious from the form of eq 28; note, however, that tensor B in the DF case does not have a lower-triangular shape as it would in the case of Cholesky. Despite the formal similarity, the Cholesky approximation cannot be applied directly to indefinite operators whereas the DF can be easily applied to this case.[112] DF is also technically simpler, albeit its reliance on fitting basis set optimization does not permit control of the fitting error as robustly as for the Cholesky case (this is not in practice a significant drawback).

As noted, DF and Cholesky approximations alone cannot reduce complexity of many-body methods without additional approximations. Thus, these techniques are often discussed together with other cost-saving measures that do not fall under the umbrella of integral approximation, like the frozen natural orbitals (FNO) method[113−116] and variable/mixed precision.[117−122] The former is related to (and was motivated by) the use of natural orbitals (e.g., pair-natural orbitals) in reduced-scaling formalisms of many-body methods discussed later.

More general factorizations of the Coulomb integral tensor have also been considered, such as the CP decomposition (or separated representation),[123,124] Cholesky-plus-SVD,[125] Clustered Low-Rank,[126] and tensor hypercontraction,[127−129] which is closely related to the pseudospectral approximation.[130] By numerically approximating the Hamiltonian these techniques can reduce the complexity of many-body methods by one degree, but numerical approximations for the wave function are also needed to be able to reduce the complexity more significantly, as we discuss next.

2.4. Reduced-Scaling Formulations

Strategies that reduce the asymptotic complexity of the many-body methods fall into one of the two camps:

- Fast Methods: In this group of approaches, the wave function of the whole system is computed using fast techniques for application of the Hamiltonian to the wave function, e.g., by changing the representation to maximize data-sparsity of the Hamiltonian and the wave function. This general strategy can appear in many disguises, such as the use of localized bases to reveal sparsity of states and operators, and fast resummations like Fast Fourier Transform (FFT)[131] or Fast Multipole Method (FMM).[132,133] In the molecular context, the sparse representations take many forms, e.g. atomic basis sets, localized molecular orbitals, finite-element representations and grids. Among the most successful approaches of this kind were the correlation methods expressed in terms of localized molecular orbitals (conventional localized occupied orbitals and redundant nonorthogonal projected atomic orbitals (PAOs) for the unoccupied basis) proposed by Pulay and Saebø.[134,135] Its further development and large-scale use has been made possible by the work of Werner, Schütz, and coworkers.[136−141] Another alternative is to express many-body methods in terms of canonical (hence delocalized) orbitals and attain reduced scaling by exploiting sparsity in the AO basis representation; these methods were pioneered by Almlöf[142,143] and developed further by Scuseria, Ochsenfeld, Head-Gordon, and others.[144−146] Combined use of localized occupied orbitals and atomic orbitals for the unoccupied space has also been considered.[147] Other choices of bases for unoccupied orbitals that have been used over the years: pair-natural orbitals (PNOs) investigated in 1960s and 70s by Edmiston, Krauss, Meyer, and others[148−154] and recently redeployed in the context of local correlation by Neese and others;[102−106] orbital-specific virtuals (OSVs) of Chan and Manby;[155] and localized virtual orbitals of Jørgensen.[156,157] Other tensor factorization techniques, such as tensor hypercontraction, can also be used in a similar vein.[128] Lastly, we should mention scaling reduction via the use of (non-LCAO) numerical representations, e.g., real-space/reciprocal-space grids, that permit fast application of operators.[158−162]

- Divide-and-Conquer: In this group of approaches, the whole system is divided into small fragments and the properties of the whole system are patched up (assembled) from the contributions of its fragments. The defining feature of these methods is that the computations on the fragments are completely independent of one another. The fragment definition can be made ad hoc (e.g., using chemical intuition or following the natural partitioning that exists in liquids and molecular crystals), as is done for example in the fragment molecular orbital (FMO)[163] and its many cousins,[164−171] or using an approximate (e.g., mean-field) description of the system as a whole. Examples of such approaches include the divide-and-conquer method,[172] the incremental scheme,[173] the cluster-in-molecules approach,[174] and the divide-expand-consolidate (DEC) scheme,[175,176] among others. These approaches are closely related to the quantum embedding approaches[177−182] that usually aim at accurate description of a single fragment of the whole system, with the rest of the system (bath) described at a lower level of theory. A significant challenge posed by these methods is how to increase the size of fragments for systematic control of the error. For the approaches with ad hoc fragment definitions, the many-body expansion[183−186] is a popular strategy to systematically grow the fragment size.

For iterative (infinite-order) many-body methods like coupled-cluster or Green's function methods, the divide-and-conquer strategy usually requires redundancy in the fragment

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

definition, i.e. the fragments must be overlapping because the independent description of each fragment needs to span the relevant correlation length scales. Thus, iterative methods are in principle more efficiently implemented using the fast techniques which avoid the redundancy. However, perturbative methods, such as MP2 and the perturbative triples treatment in CCSD(T), can avoid the need for redundant fragments. Namely, perturbative energy corrections in these methods can be expressed via a Laplace transform as a sum (trace) over local basis states (AOs and/or localized MOs); note that when expressed in a local basis without the Laplace transform, these methods are iterative. The sum can then be perfectly divided into independent fragment contributions without any additional approximations. Not only the energy but also the wave functions in these methods can be perfectly divided into independent contributions. It is of course possible to combine the two strategies; because the fragment size is commensurate with the correlation length scales, as the correlation length scale grows, fast methods become viable for describing individual fragments.[187−191] Fast methods also have advantages for treatment of excited states and properties due to the availability of a consistent single representation of the wave function.

Impressive production realization for reduced complexity coupled-cluster computations on molecules with thousands of atoms have been demonstrated recently via both types of strategies. Namely, methods combining ideas of local correlation with PNO-style compression have been demonstrated with reduced[192,193] and asymptotically linear complexity[105,194−201] for coupled-cluster and single- and multireference perturbation theories.[202−205] These techniques have also been extended to treatment of analytic nuclear forces,[206−208] excitation energies,[209−212] and electron attachment/detachment energies.[213−215] Divide-and-conquer methodologies have been demonstrated for ground-state energies up to the CCSD(T) level[189,190] and forces up to MP2 level.[216]

3. COMPUTATIONAL TRAITS OF MANY-BODY METHODS

The iterative coupled-cluster method in the context of which we discussed the computational aspects of many-body quantum chemistry shares most computational characteristics with the other traditional many-body methods (algebra of dense high-dimensional tensors, importance of symmetries, ability to reduce complexity by change of representation). In this section, we will discuss the central computational traits of many-body methods that will be needed to understand the design of algorithms and implementation on parallel computer platforms.

3.1. Traditional Many-Body Methods

The traditional formulation of the many-body methods reduces to algebra of dense tensors. To demonstrate this more concretely, consider the equations for the CCSD method:

$$\begin{aligned}\langle \Phi_{i_1}^{a_1}|\bar{H}|0\rangle = {} & f_{a_1}^{i_1} - \bar{g}_{i_2a_1}^{i_1a_2} t_{a_2}^{i_2} - f_{i_2}^{i_1} t_{a_1}^{i_2} + f_{a_1}^{a_2} t_{a_2}^{i_1} - \frac{1}{2}\bar{g}_{i_2i_3}^{i_1a_2} t_{a_1a_2}^{i_2i_3} \\ & - \frac{1}{2}\bar{g}_{i_2a_1}^{a_2a_3} t_{a_2a_3}^{i_1i_2} + f_{i_2}^{a_2} t_{a_1a_2}^{i_1i_2} + \bar{g}_{i_2a_1}^{a_2a_3} t_{a_2}^{i_2} t_{a_3}^{i_1} + \bar{g}_{i_2i_3}^{i_1a_2} t_{a_2}^{i_2} t_{a_1}^{i_3} \\ & - f_{i_2}^{a_2} t_{a_2}^{i_1} t_{a_1}^{i_2} \\ & - \frac{1}{2}\bar{g}_{i_2i_3}^{a_2a_3} t_{a_1}^{i_2} t_{a_2a_3}^{i_1i_3} - \frac{1}{2}\bar{g}_{i_2i_3}^{a_2a_3} t_{a_1}^{i_1} t_{a_2a_3}^{i_2i_3} + \bar{g}_{i_2i_3}^{a_2a_3} t_{a_2}^{i_2} t_{a_1a_3}^{i_1i_3} \\ & - \bar{g}_{i_2i_3}^{a_2a_3} t_{a_2}^{i_2} t_{a_3}^{i_1} t_{a_1}^{i_3}\end{aligned} \tag{29}$$

$$\begin{aligned}\langle \Phi_{i_1i_2}^{a_1a_2}|\bar{H}|0\rangle = {} & \hat{A}_{i_1i_2}^{a_1a_2}\Big(\frac{1}{4}\bar{g}_{a_1a_2}^{i_1i_2} + \frac{1}{8}\bar{g}_{a_1a_2}^{a_3a_4} t_{a_3a_4}^{i_1i_2} - \frac{1}{2} f_{a_1}^{a_3} t_{a_2a_3}^{i_1i_2} + \frac{1}{2} f_{i_3}^{i_1} t_{a_1a_2}^{i_2i_3} \\ & + \frac{1}{8}\bar{g}_{i_3i_4}^{i_1i_2} t_{a_1a_2}^{i_3i_4} - \bar{g}_{i_3a_1}^{i_1a_3} t_{a_2a_3}^{i_2i_3} + \frac{1}{2}\bar{g}_{a_1a_2}^{i_1a_3} t_{a_3}^{i_2} + \frac{1}{2}\bar{g}_{i_3a_1}^{i_1i_2} t_{a_2}^{i_3} \\ & + \frac{1}{2} f_{i_3}^{a_3} t_{a_3}^{i_1} t_{a_1a_2}^{i_2i_3} + \frac{1}{2} f_{i_3}^{a_3} t_{a_1}^{i_3} t_{a_2a_3}^{i_1i_2} + \frac{1}{4}\bar{g}_{i_3i_4}^{i_1a_3} t_{a_3}^{i_2} t_{a_1a_2}^{i_3i_4} - \frac{1}{2}\bar{g}_{i_3i_4}^{i_1a_3} t_{a_3}^{i_3} t_{a_1a_2}^{i_2i_4} \\ & - \bar{g}_{i_3i_4}^{i_1a_3} t_{a_1}^{i_3} t_{a_2a_3}^{i_2i_4} - \bar{g}_{i_3a_1}^{a_3a_4} t_{a_3}^{i_1} t_{a_2a_4}^{i_2i_3} - \frac{1}{2}\bar{g}_{i_3a_1}^{a_3a_4} t_{a_3}^{i_3} t_{a_2a_4}^{i_1i_2} + \frac{1}{4}\bar{g}_{i_3a_1}^{a_3a_4} t_{a_2}^{i_3} t_{a_3a_4}^{i_1i_2} \\ & + \frac{1}{4}\bar{g}_{a_1a_2}^{a_3a_4} t_{a_3}^{i_1} t_{a_4}^{i_2} + \bar{g}_{i_3a_1}^{i_1a_3} t_{a_3}^{i_2} t_{a_2}^{i_3} + \frac{1}{4}\bar{g}_{i_3i_4}^{i_1i_2} t_{a_1}^{i_3} t_{a_2}^{i_4} - \frac{1}{4}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_1a_3}^{i_1i_2} t_{a_2a_4}^{i_3i_4} \\ & + \frac{1}{2}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_1a_3}^{i_1i_3} t_{a_2a_4}^{i_2i_4} - \frac{1}{4}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3a_4}^{i_1i_3} t_{a_1a_2}^{i_2i_4} + \frac{1}{16}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3a_4}^{i_1i_2} t_{a_1a_2}^{i_3i_4} + \frac{1}{8}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3}^{i_1} t_{a_4}^{i_2} t_{a_1a_2}^{i_3i_4} \\ & - \frac{1}{2}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3}^{i_1} t_{a_4}^{i_3} t_{a_1a_2}^{i_2i_4} - \bar{g}_{i_3i_4}^{a_3a_4} t_{a_3}^{i_1} t_{a_1}^{i_3} t_{a_2a_4}^{i_2i_4} + \frac{1}{2}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3}^{i_3} t_{a_1}^{i_4} t_{a_2a_4}^{i_1i_2} + \frac{1}{8}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_1}^{i_3} t_{a_2}^{i_4} t_{a_3a_4}^{i_1i_2} \\ & + \frac{1}{2}\bar{g}_{i_3a_1}^{a_3a_4} t_{a_3}^{i_1} t_{a_4}^{i_2} t_{a_2}^{i_3} + \frac{1}{2}\bar{g}_{i_3i_4}^{i_1a_3} t_{a_3}^{i_2} t_{a_1}^{i_3} t_{a_2}^{i_4} + \frac{1}{4}\bar{g}_{i_3i_4}^{a_3a_4} t_{a_3}^{i_1} t_{a_4}^{i_2} t_{a_1}^{i_3} t_{a_2}^{i_4}\Big),\end{aligned} \tag{30}$$

where $\Phi_{i_1}^{a_1} \equiv a_{a_1}^{\dagger} a_{i_1} |0\rangle$ and $\Phi_{i_1i_2}^{a_1a_2} \equiv a_{a_1}^{\dagger} a_{a_2}^{\dagger} a_{i_2} a_{i_1} |0\rangle$. The key computational aspects include the following:

Technical Complexity: Even for a relatively low-order method like CCSD, explicit tensor algebra is complex and automated techniques for derivation and evaluation of these equations are advised. This is especially true if additional approximations are introduced, such as Hamiltonian factorization or wave function compression. For more complex methods—and especially when a new method is considered—manual implementation of tensor algebra rapidly crosses from "error prone" to "intractable". Thus, automated manipulation of quantum many-body algebra has a long history in quantum chemistry[217] and is employed in related

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

areas (e.g., nuclear physics[218] and high-energy physics[219]). Nevertheless, the majority of many-body methods are still implemented manually; technical complexity of implementing complex tensor algebra nowadays is usually lowered by the use of high-level array/tensor libraries[220−227] or languages[228] that keep the abstraction level of the code as close to the math as necessary.

Optimization: As written, the expressions are not suitable for implementation due to suboptimal computational complexity. The singles and doubles equations include terms with $O(N^6)$ and $O(N^8)$ complexity. Techniques like strength reduction (factorization), common subexpression elimination (CSE), and fusion can be used to reduce the complexity of these expressions to $O(N^5)$ and $O(N^6)$. Indeed, it is very easy to see that even trivial strength reduction of each term in eqs 29 and 30, namely evaluation of $ABC...$ as $((AB)C)...$, will reduce the complexity to optimal; e.g., evaluating the last term in eq 30 as $\frac{1}{4}((((\bar{g}^{a_3a_4}_{i_3i_4} t^{i_1}_{a_3}) t^{i_2}_{a_4}) t^{i_3}_{a_1}) t^{i_4}_{a_2})$ reduces its complexity from $O(N^8)$ to $O(N^5)$ at the cost of introducing intermediate tensors of size $O(N^4)$ (i.e., the storage complexity is not increased, albeit the total storage size does). In general, the optimization problem is hard and open-ended. It is hard because the search space is factorial: e.g., the number of possible factorizations of each term is factorial, and so is the number of candidates for CSE and fusion. The optimization problem is open-ended because it permits an arbitrary number of objective functions (optimize for time, for space, etc.) and constraints, and in general optimization for time requires complete specification of the performance model for each operation type. Due to the latter consideration, the expression optimization should be, if possible, performed at runtime, when the problem dimensions and runtime parameters (number of processors, details of memory hierarchy, kernel performance model, etc.) are known. Due to the complexity of writing a compiler that can perform the expression optimization, implementation of basic methods like CCSD is performed manually using prefactorized heuristically optimized expressions available in the literature (e.g., for CCSD[15,27,229,230]). For higher-order methods, both manual and automated approaches have been applied.[47,48,231] With a few exceptions,[231] usually optimization of coupled-cluster expressions is done at "compile" (code generation) time and transformed expressions are optionally transformed into compilable code.[29,217,232−234]

Large Tensors: In the traditional formulation of the many-body methods, the tensors (Hamiltonian matrix elements, amplitudes, and expression intermediates) are treated as dense. This results in high-order polynomial storage complexity, namely $O(N^4)$ for CCSD, $O(N^6)$ for CCSDT, etc. More specifically, the number of CC amplitudes of rank r is $o^r v^r$. Using as a concrete example of the computation of the CC energy of a 20-molecule water cluster performed by Apra et al.,[235] with $o = 80$ and $v = 920$, the two- and three-body amplitudes in 64-bit representation occupy 43.3 GB and 3.2 PB, respectively. The shocking rise of the size of amplitude tensor with the particle rank highlights the corresponding shocking rise of the computational cost of the coupled-cluster methods with the rank, and the increasing importance of numerical approximations to achieve reduced scaling of storage and cost. Whereas the rank-2 amplitudes can fit into RAM of a typical node of a modern HPC platform, multiple quantities of this size appear in the course of solving CCSD equations and thus even on "fat" types of nodes, it may be necessary to distribute the 2-body amplitudes and other tensors of this size. In CCSD equations, there are also larger tensors, of size ov^3 and v^4, occupying 498 GB and 5.7 TB, respectively which do not fit into memory of a typical HPC node of today and must be distributed; alternative formulations with numerical approximations like density fitting can help avoid these intermediates (see below). Clearly the triples amplitudes can only be stored in the aggregate memory of a large parallel platform and thus must be distributed among nodes.

Most Operations are in Tensor Contractions: Most of the arithmetic operations in many-body methods like coupled-cluster occur in tensor contractions. Normally this term is applied to purely covariant products in which common indices of 2 or more tensors are summed over and thus do not appear in the final result (Hadamard product, mixed covariant-Hadamard product (e.g., the Khatri-Rao product), hyper-contraction, and other generalized products[92,236] also appear in nonstandard formulations of many-body methods). For example, consider the second term in eq 30, due to its diagrammatic structure referred to as the particle−particle ladder (PPL):

$$\bar{g}^{a_3a_4}_{a_1a_2} t^{i_1i_2}_{a_3a_4} \equiv \sum_{a_3a_4} \bar{g}^{a_3a_4}_{a_1a_2} t^{i_1i_2}_{a_3a_4} \tag{31}$$

This is the most expensive step in CCSD with realistic basis sets. Its cost is o^2v^4 floating-point additions and o^2v^4 floating-point multiplications, for the total of $2o^2v^4$ floating-point operations, or FLOPs (note that in the quantum chemistry literature it is more common to see the operation count written as o^2v^4, without the factor of two; see section 2.1 for more details); for the 20-water example we considered above, this translates into 9.2×10^{15} FLOPs $\equiv$ 9.2 PFLOPs. In contrast, addition of individual terms on the r.h.s. of eq 30 costs o^2v^2 or only 5.4 GFLOPs. Binary covariant tensor contraction can be viewed as a generalization of matrix multiplication and can be mapped onto matrix multiplication by appropriate matricization of the tensors. Because high-performance matrix multiplication routines, via the standard BLAS library,[237] are available on all modern HPC platforms, tensor contraction and by extension traditional many-body formalism have potential to reach high fraction of the peak performance on modern architectures.

Symmetry: Due to the Fermionic symmetry of the electronic states, as well as the geometric symmetry of the molecular framework, the tensors in many-body methods possess a variety of symmetries. For example, tensor $\bar{g}^{a_3a_4}_{a_1a_2}$ is antisymmetric with respect to the permutation of the contravariant/covariant indices, as well as Hermitian with respect to the permutation of the contravariant indices with the corresponding covariant indices:

$$\begin{aligned} \bar{g}^{a_3a_4}_{a_1a_2} &= -\bar{g}^{a_4a_3}_{a_1a_2} = -\bar{g}^{a_3a_4}_{a_2a_1} = \bar{g}^{a_4a_3}_{a_2a_1} \\ &= (\bar{g}^{a_1a_2}_{a_3a_4})^* = -(\bar{g}^{a_1a_2}_{a_4a_3})^* = ... \end{aligned} \tag{32}$$

This means that only $\approx 1/8$ of the elements of this tensor are unique. Utilization of this permutational symmetry is thus essential to minimize the prefactor (but not the complexity) of storage and operation costs. For finite large systems, geometric symmetry is rarely exploitable, but periodic infinite crystals are a notable exception where the translational symmetry of the (reciprocal) lattice can be exploited very effectively.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

Freedom to Choose Basis: The covariant property of eqs 29 and 30, namely that every sum is over a {covariant,contravariant} pair of indices is the result of the invariance of the coupled-cluster method to basis transformation within occupied and unoccupied subspaces. The freedom to change the representation allows to choose orbital basis that maximizes the data sparsity of the Hamiltonian and amplitude tensors. This simple idea is the foundation of reduced-scaling formulations in which the use of localized orbital basis is the prerequisite to exploiting the data sparsity and lowering the storage and operation complexity of the many-body methods. Their computational traits will be discussed next.

3.2. Reduced-Scaling Many-Body Methods

Near-optimal exploitation of the data sparsity significantly increases the formal and computational complexity of the reduced-scaling many-body theory. Due to the great variety of the approaches for complexity reduction in the many-body theory, we can only broadly discuss the relevant computational traits; in this section, we will use the categorization of the reduced-scaling methods into fast and divide-and-conquer that we introduced in section 2.4.

In fast methods, we typically have to solve different and more complex equations that involve sparse data, with potentially many changes in representation needed between steps. For example, in PNO-style many-body methods, the equations themselves become more complicated due to the hierarchical dependencies between index spaces; this is best demonstrated by the PNO−CCSD example:

$$
\begin{aligned}
\langle \Phi_{i_1}^{a_1^{i_1}} | \bar{H} | 0 \rangle = \quad & f_{a_1^{i_1}}^{i_1} - g_{i_2 a_1^{i_1}}^{i_1 a_2^{i_2}} t_{a_2^{i_2}}^{i_2} - \frac{1}{2} g_{i_2 a_1^{i_1}}^{a_2^{i_2 i_2} a_3^{i_2 i_2}} t_{a_2^{i_2 i_2} a_3^{i_2 i_2}}^{i_1 i_2} \\
& + f_{a_1^{i_1}}^{a_2^{i_1}} t_{a_2^{i_1}}^{i_1} - f_{i_2}^{i_1} t_{a_2^{i_2}}^{i_2} S_{a_1^{i_1}}^{a_2^{i_2}} + g_{i_2 a_1^{i_1}}^{a_2^{i_2} a_3^{i_1}} t_{a_2^{i_2} a_3^{i_1}}^{i_2} \\
& + \frac{1}{2} g_{i_2 i_3}^{i_1 a_2^{i_2 i_3}} t_{a_2^{i_2 i_3} a_3^{i_2 i_3}}^{i_2 i_3} S_{a_1^{i_1}}^{a_3^{i_2 i_3}} - f_{i_2}^{a_2^{i_2 i_2}} t_{a_2^{i_2 i_2} a_3^{i_2 i_2}}^{i_1 i_2} S_{a_1^{i_1}}^{a_3^{i_2 i_2}} \\
& - g_{i_2 i_3}^{a_2^{i_2} a_3^{i_1 i_3}} t_{a_2^{i_2} a_3^{i_1 i_3} a_4^{i_1 i_3}}^{i_2 i_3} S_{a_1^{i_1}}^{a_4^{i_1 i_3}} \\
& - g_{i_2 i_3}^{i_1 a_2^{i_2 i_3}} t_{a_2^{i_2} a_3^{i_2 i_3}}^{i_2} t_{a_2^{i_2} a_3^{i_2 i_3}}^{i_3} S_{a_1^{i_1}}^{a_3^{i_2}} - f_{i_2}^{a_2^{i_1}} t_{a_2^{i_2} a_3^{i_1}}^{i_2} t_{a_2^{i_1}}^{i_1} S_{a_1^{i_1}}^{a_3^{i_2}} \\
& + \frac{1}{2} g_{i_2 i_3}^{a_2^{i_1} a_3^{i_2 i_3}} t_{a_2^{i_1}}^{i_1} t_{a_3^{i_2 i_3} a_4^{i_2 i_3}}^{i_2 i_3} S_{a_1^{i_1}}^{a_4^{i_2 i_3}} \\
& - \frac{1}{2} g_{i_2 i_3}^{a_2^{i_1 i_3} a_3^{i_1 i_3}} t_{a_2^{i_2} a_3^{i_1 i_3} a_4^{i_1 i_3}}^{i_2} t_{a_3^{i_1 i_3} a_4^{i_1 i_3}}^{i_1 i_3} S_{a_1^{i_1}}^{a_4^{i_2}} + g_{i_2 i_3}^{a_2^{i_3} a_3^{i_1}} t_{a_4^{i_2} a_2^{i_3}}^{i_2} t_{a_3^{i_1}}^{i_3} t_{a_2^{i_1}}^{i_1} S_{a_1^{i_1}}^{a_4^{i_2}}
\end{aligned}
\tag{33}
$$

$$
\begin{aligned}
\langle \Phi_{i_1 i_2}^{a_1^{i_1 i_2} a_2^{i_1 i_2}} | \bar{H} | 0 \rangle = \hat{A}_{i_1 i_2}^{a_1^{i_1 i_2} a_2^{i_1 i_2}} \Bigg(& \frac{1}{4} g_{a_1^{i_1 i_2} a_2^{i_1 i_2}}^{i_1 i_2} - \frac{1}{2} f_{a_1^{i_1 i_2}}^{a_3^{i_1 i_2}} t_{a_2^{i_1 i_2} a_3^{i_1 i_2}}^{i_1 i_2} + \frac{1}{8} g_{a_1^{i_1 i_2} a_2^{i_1 i_2}}^{a_3^{i_1 i_2} a_4^{i_1 i_2}} t_{a_3^{i_1 i_2} a_4^{i_1 i_2}}^{i_1 i_2} + \frac{1}{2} g_{a_1^{i_1 i_2} a_2^{i_1 i_2}}^{i_1 a_3^{i_2}} t_{a_3^{i_2}}^{i_2} + \frac{1}{2} g_{i_3 a_1^{i_1 i_2}}^{i_1 i_2} t_{a_3^{i_3} a_2^{i_1 i_2}}^{i_3 a_3^{i_3}} - \frac{1}{2} g_{i_3 a_1^{i_1 i_2}}^{a_3^{i_3} a_4^{i_1 i_2}} t_{a_3^{i_3} a_2^{i_1 i_2} a_4^{i_1 i_2}}^{i_3} \\
& + g_{i_3 a_1^{i_1 i_2}}^{i_1 a_3^{i_2 i_3}} t_{a_3^{i_2 i_3} a_4^{i_2 i_3}}^{i_2 i_3} S_{a_2^{i_1 i_2}}^{a_4^{i_2 i_3}} + \frac{1}{4} g_{a_1^{i_1 i_2} a_2^{i_1 i_2}}^{a_3^{i_1} a_4^{i_2}} t_{a_3^{i_1} a_4^{i_2}}^{i_1} t_{a_4^{i_2}}^{i_2} + g_{i_3 a_1^{i_1 i_2}}^{i_1 a_3^{i_2}} t_{a_4^{i_3}}^{i_3} t_{a_3^{i_2}}^{i_2} S_{a_2^{i_1 i_2}}^{a_4^{i_3}} - \frac{1}{2} f_{i_3}^{a_3^{i_1 i_2}} t_{a_4^{i_3}}^{i_3} t_{a_1^{i_1 i_2} a_3^{i_1 i_2}}^{i_1 i_2} S_{a_2^{i_1 i_2}}^{a_4^{i_3}} + \frac{1}{8} g_{i_3 i_4}^{i_1 i_2} t_{a_3^{i_3 i_4} a_4^{i_3 i_4}}^{i_3 i_4} S_{a_1^{i_1 i_2}}^{a_3^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_4^{i_3 i_4}} \\
& + \frac{1}{4} g_{i_3 a_1^{i_1 i_2}}^{a_3^{i_1 i_2} a_4^{i_1 i_2}} t_{a_5^{i_3}}^{i_3} t_{a_3^{i_1 i_2} a_4^{i_1 i_2}}^{i_1 i_2} S_{a_2^{i_1 i_2}}^{a_5^{i_3}} + g_{i_3 a_1^{i_1 i_2}}^{a_3^{i_1} a_4^{i_2 i_3}} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2 i_3} a_5^{i_2 i_3}}^{i_2 i_3} S_{a_2^{i_1 i_2}}^{a_5^{i_2 i_3}} + \frac{1}{2} f_{i_3}^{i_1} t_{a_3^{i_2 i_3} a_4^{i_2 i_3}}^{i_2 i_3} S_{a_1^{i_1 i_2}}^{a_3^{i_2 i_3}} S_{a_2^{i_1 i_2}}^{a_4^{i_2 i_3}} - \frac{1}{4} g_{i_3 i_4}^{a_3^{i_3 i_4} a_4^{i_1 i_2}} t_{a_3^{i_3 i_4} a_5^{i_3 i_4}}^{i_3 i_4} S_{a_1^{i_1 i_2} a_4^{i_1 i_2}}^{a_5^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_4^{i_3 i_4}} \\
& + \frac{1}{16} g_{i_3 i_4}^{a_3^{i_1 i_2} a_4^{i_1 i_2}} t_{a_5^{i_3 i_4} a_6^{i_3 i_4}}^{i_3 i_4} t_{a_3^{i_1 i_2} a_4^{i_1 i_2}}^{i_1 i_2} S_{a_1^{i_1 i_2}}^{a_5^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_6^{i_3 i_4}} + \frac{1}{2} f_{i_3}^{a_3^{i_1}} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2 i_3} a_5^{i_2 i_3}}^{i_2 i_3} S_{a_1^{i_1 i_2}}^{a_4^{i_2 i_3}} S_{a_2^{i_1 i_2}}^{a_5^{i_2 i_3}} + \frac{1}{4} g_{i_3 i_4}^{i_1 i_2} t_{a_3^{i_3} a_4^{i_4}}^{i_3 i_4} S_{a_1^{i_1 i_2}}^{a_3^{i_3} a_4^{i_4}} - g_{i_3 i_4}^{i_1 a_3^{i_2 i_4}} t_{a_4^{i_3} a_3^{i_2 i_4} a_5^{i_2 i_4}}^{i_3 i_2 i_4} S_{a_1^{i_1 i_2}}^{a_5^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_4^{i_3}} \\
& - \frac{1}{2} g_{i_3 i_4}^{i_1 a_3^{i_3}} t_{a_3^{i_3}}^{i_3} t_{a_4^{i_2 i_4} a_5^{i_2 i_4}}^{i_2 i_4} S_{a_1^{i_1 i_2}}^{a_4^{i_2 i_4}} S_{a_2^{i_1 i_2}}^{a_5^{i_2 i_4}} - \frac{1}{4} g_{i_3 i_4}^{a_3^{i_2 i_4} a_4^{i_2 i_4}} t_{a_5^{i_3 i_4} a_6^{i_3 i_4}}^{i_1 i_3} t_{a_3^{i_2 i_4} a_4^{i_2 i_4}}^{i_2 i_4} S_{a_1^{i_1 i_2}}^{a_5^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_6^{i_3 i_4}} + \frac{1}{4} g_{i_3 i_4}^{i_1 a_3^{i_2}} t_{a_3^{i_2}}^{i_2} t_{a_4^{i_3 i_4} a_5^{i_3 i_4}}^{i_3 i_4} S_{a_1^{i_1 i_2}}^{a_4^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_5^{i_3 i_4}} \\
& + \frac{1}{2} g_{i_3 i_4}^{a_3^{i_1 i_3} a_4^{i_2 i_4}} t_{a_5^{i_3} a_6^{i_3}}^{i_1 i_3} t_{a_4^{i_2 i_4} a_6^{i_2 i_4}}^{i_2 i_4} S_{a_1^{i_1 i_2}}^{a_5^{i_1 i_3}} S_{a_2^{i_1 i_2}}^{a_6^{i_2 i_4}} + \frac{1}{2} g_{i_3 i_4}^{a_3^{i_1} a_4^{i_2}} t_{a_5^{i_3}}^{i_3} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2}}^{i_2} S_{a_2^{i_1 i_2}}^{a_5^{i_3}} + \frac{1}{2} g_{i_3 i_4}^{a_3^{i_4} a_4^{i_1 i_2}} t_{a_5^{i_3}}^{i_3} t_{a_3^{i_4}}^{i_4} t_{a_1^{i_1 i_2} a_4^{i_1 i_2}}^{i_1 i_2} S_{a_2^{i_1 i_2}}^{a_5^{i_3}} \\
& + \frac{1}{8} g_{i_3 i_4}^{a_3^{i_1} a_4^{i_2}} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2}}^{i_2} t_{a_5^{i_3} a_6^{i_3}}^{i_3 i_4} S_{a_1^{i_1 i_2}}^{a_5^{i_3 i_4}} S_{a_2^{i_1 i_2}}^{a_6^{i_3 i_4}} - g_{i_3 i_4}^{a_3^{i_1} a_4^{i_2 i_4}} t_{a_5^{i_3}}^{i_3} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2 i_4} a_6^{i_2 i_4}}^{i_2 i_4} S_{a_1^{i_1 i_2}}^{a_6^{i_2 i_4}} S_{a_2^{i_1 i_2}}^{a_5^{i_3}} + \frac{1}{2} g_{i_3 i_4}^{i_1 a_3^{i_2}} t_{a_4^{i_3}}^{i_3} t_{a_5^{i_4}}^{i_4} t_{a_2^{i_2}}^{i_2} S_{a_3^{i_3} a_4^{i_4}}^{a_5^{i_4}} \\
& - \frac{1}{2} g_{i_3 i_4}^{a_3^{i_1} a_4^{i_3}} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_3}}^{i_3} t_{a_5^{i_2 i_4} a_6^{i_2 i_4}}^{i_2 i_4} S_{a_1^{i_1 i_2}}^{a_5^{i_2 i_4}} S_{a_2^{i_1 i_2}}^{a_6^{i_2 i_4}} + \frac{1}{8} g_{i_3 i_4}^{a_3^{i_1 i_2} a_4^{i_1 i_2}} t_{a_5^{i_3} a_6^{i_4}}^{i_3 i_4} t_{a_3^{i_1 i_2} a_4^{i_1 i_2}}^{i_1 i_2} S_{a_1^{i_1 i_2}}^{a_5^{i_3}} S_{a_2^{i_1 i_2}}^{a_6^{i_4}} + \frac{1}{4} g_{i_3 i_4}^{a_3^{i_1} a_4^{i_2}} t_{a_5^{i_3} a_6^{i_4}}^{i_3 i_4} t_{a_3^{i_1}}^{i_1} t_{a_4^{i_2}}^{i_2} S_{a_1^{i_1 i_2}}^{a_5^{i_3}} S_{a_2^{i_1 i_2}}^{a_6^{i_4}} \Bigg)
\end{aligned}
\tag{34}
$$

Although these expressions are seemingly far more complex than their respective counterparts eqs 29 and 30, the structure of these equations is very similar due to the identical physics; the only difference is the change of representation of the unoccupied orbitals, but there are extra overlaps due to the nonorthonormality of the new basis. The more serious issue is that the algebra of dense tensors has been replaced by the algebra of data-sparse tensors with irregular structure. The latter is more difficult to implement, to optimize, and to parallelize.

Specifically, the covariance of eqs 29 and 30 is lost due to the orbital dependence of the unoccupied indices on the occupied. In computational terms, the perfectly nested loops in eqs 29 and 30, which can be optimized and parallelized by a variety of powerful polyhedral model techniques,[238] become nonaffine loop nests that cannot be optimized by the existing automatic tools.[239] Furthermore, the inner loop ranges can vary widely due to the physics of electron correlation, e.g., range of a^{ij} is typically >100 when orbital i is spatially close to orbital j; in contrast, the range can approach 0 for ij pairs composed of spatially distant orbitals. This translates into massive variance of the computational load of block operations, and the elementary block operations themselves (like contractions) unable to reach significant proportion of the hardware peak throughput due to the limited amount of data parallelism and increased logic overhead.

Next, the tensors in eqs 33 and 34 can be represented as a number of data structures (hierarchical tensors, sparse tensors, etc.), but irrespective of the representation, these are less amenable to parallelization. Their sparse structure is determined by dynamical factors such as properties of the particular molecular system (e.g., effective dimensionality, such as 1-d for linear molecules vs 3-d for globular molecules), variations between iterations, etc. Higher effective surface to volume ratio means that the ratio of communication to computation is lower, just like in, e.g., sparse matrix algebra. Thus, attaining high performance even on a single processor becomes more difficult; scalable parallel implementation is also more complicated than that of the traditional formalism.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

Note that some fast methods, for example, the AO-based methods,[144,145] do not involve the hierarchical space dependencies. But the equations are still more complex than the standard approaches, and the need to deal with the sparse data causes similar issues that make them more difficult to implement, optimize, and parallelize.

Divide-and-conquer approaches are typically significantly simpler to implement than their fast counterparts, because for each fragment, we solve the same or a trivially modified (due to the embedding) "base" conventional-scaling method. Thus, the implementation complexity is greatly diminished and usually only modest modifications to the dense implementation are sufficient to compose the divide-and-conquer methods. The often-cited advantage of the fragment methods is their strong scalability (i.e., their parallel efficiency does not decay with the number of processors), due to the independence of the individual fragment computations and lack of the communication during the fragment calculations. Note that there are global communication costs related to the work distribution and aggregation of the results, which limit scalability of these methods, but typically these costs are small relative to the cost of the fragment computations. The load imbalance is the primary factor that limits strong scaling of these methods, nevertheless truly massive computations are possible.[240] Note that the individual fragment computations may need to be parallelized also due to their size.

4. OVERVIEW OF HIGH-PERFORMANCE COMPUTING PLATFORMS

Although the expected computational environment will vary greatly from user to user, the high-end HPC platforms of today are likely to become the typical HPC platforms of tomorrow. Thus, it is instructive to review typical and high-end HPC platforms with an eye toward the ongoing and anticipated trends in their evolution.

A typical HPC platform used by today's scientist is likely to be a commodity cluster.[241] A commodity cluster is an ensemble of independent computers (nodes) connected by a network fabric (interconnect), with both built from off-the-shelf (commodity) components. The popularity of commodity clusters is primarily due to their low cost, which is driven down by the economy of scale for the mass-market components. Commodity clusters are ubiquitous in everyday computing in academia, government laboratories, and industry alike: for example, our institution's campus research computing facility features 4 commodity clusters, each containing as few as 48 and as many as 408 nodes.[242,243] The same is true for high-end computing: as of November 2019, more than 90% of the systems on the Top500 list[244] are commodity clusters (with thousands and even tens of thousands of nodes each), and they account for more than 80% of the total performance on the list. (The term "cluster" as utilized by the Top500 list[245] for our purposes is synonymous with "commodity cluster". The noncluster systems on Top500 for our purposes will be classified as the "high-end" systems.) Although these figures include systems not used for academic and government research, it is clear that the vast majority of research HPC platforms today—and likely to be true in the future as well—are commodity platforms.

The conventional processors ("CPU") account for the vast majority of the total throughput of the commodity systems, and of the Top500 list as a whole. By far the most popular are CPUs of the x86 family, primarily from Intel and AMD: they account for >95% of systems on the list and >73% of the total throughput. Other CPU families represented on the Top 500 List include PowerPC (e.g., IBM POWER9), ARM (e.g., SPARC64 XIfx), and others (e.g., Sunway SW26010).

All CPUs represented on the Top500 list are multicore CPUs, with at least 4 cores per CPU chip (socket) and as many as 260 cores per socket (Sunway SW26010). Most common (over 35% system share) are the systems with 20-core x86 CPUs; coupled with the typical use multiple sockets per node, a typical node of an HPC system contains on the order of 30−50 cores. To extract the peak performance from a modern CPU requires utilization of vector arithmetic logic units (ALUs) using data-parallel single-instruction-multiple-data (SIMD) instructions. Vector ALUs typically process vectors composed of 256/512 bits (4/8 double- or 8/16 single-precision floating point values). Because a typical core can issue up to 4 vector floating point operations (in the form of 2 fused multiply add instructions), thus, a typical HPC node can execute 40 cores × 8 SIMD lanes × 4 instructions/clock tick × 2 billion clock ticks/second ≈ 2.5 TFLOPS from its CPU(s). Of course, the peak performance depends on the instruction mix and thus on the details of the computational task itself.

Not only the cluster nodes but also the interconnect fabrics represented on the Top500 list are dominated (more than 80% of system share) by two commodity technologies, Gigabit Ethernet and Infiniband, with the rest of the systems utilizing custom or proprietary interconnects.

A significant and increasing part of the total performance of the Top500 list is attributed to the systems equipped with various accelerators: ≈29% of all systems on the November 2019 list include accelerators, whereas 10 years ago the corresponding figure was only ≈1.4%. The top 10 is dominated by the heterogeneous systems even more thoroughly: 6 out of 10 systems are heterogeneous, including the top 2 systems. The vast majority of accelerators are commodity components, namely the general-purpose graphical processing units (GPGPUs, or, simply, GPUs) produced almost exclusively by NVIDIA and accelerator cards based on Intel Xeon Phi processor. The recent rate of transition to the heterogeneous HPC platforms is nothing short of stunning: for example, 94 systems (18.8% of all systems) on the Top500 list use NVIDIA GPUs based on the Volta microarchitecture as accelerators, compared to 1 system with NVIDIA Volta only 2 years ago. This suggests that the majority of new entrants to the list are heterogeneous systems. This trend is only likely to accelerate, and we can expect that soon the majority of the list, and the HPC ecosystem as a whole, will be dominated by heterogeneous systems.

The transition to accelerators is driven largely by the cost and power considerations. It is useful to perform rough comparison using as an example the latest cluster at our institution's campus research computing facility.[243] This cluster includes 40 nodes, each equipped with 2 Intel Xeon Gold 6136 processors and 2 NVIDIA Titan V100 cards. The peak double-precision performance of a single NVIDIA Tesla V100 PCIe GPU is ≳7 TFLOPS with peak power utilization of 250W and the list price of ≈ $8,500.[246] One 12-core Intel Xeon Gold 6136 processor is capable of ≈1.15 TFLOPS in double precision utilizing 150W and priced at $2460.00. Thus, the V100 accelerator delivers ≈80% more FLOPS per dollar. Another important criterion is the energy efficiency, or FLOPS/watt, because the total power consumption is already a significant portion of the total cost of ownership of HPC

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

resources. Here, the V100 also has the advantage over its CPU counterpart by delivering ≈3.7 more FLOPS per watt (this is a merely theoretical estimate; the practical FLOPS/watt figures listed in the Green500 list[247] are much lower). This is clearly a very crude comparison of relative costs of commodity GPUs to commodity CPUs: a more precise total cost comparison must take into account the cost of other system components, the average system load, the cost of powering and maintaining the system and, most importantly, the cost of porting software applications to the GPU. Nevertheless, it is safe to say that at least in the commodity segment, the heterogeneous platforms are significantly more cost efficient than their CPU counterparts.

Due to these advantages, it is therefore not surprising that most high-end HPC platforms of today are heterogeneous, composed of relatively "fat" nodes equipped with multiple CPUs and multiple accelerators in each node. The #1 machine in the November 2019 Top500 list (and bumped to #2 in the June 2020 list) is the Summit machine[248] installed at Oak Ridge Leadership Computing Facility (OLCF) consists of 4,608 nodes, each with 2 IBM POWER9 processors and 6 NVIDIA V100 accelerators yielding ≈48 TFLOPS peak performance (only ≈1.1 TFLOPS is attributed to the CPUs). The use of complex fat nodes is a significant departure from relatively recent CPU-only high-end architectures such as the IBM Blue Gene/Q (e.g., the Mira system[249] at the Argonne LCF) with many (tens of thousands) relatively "thin" nodes. Programming of the fat multiaccelerator nodes is a key challenge due to the relative immaturity and the explicitly asynchronous character of the programming models for programming accelerators (CUDA, OpenCL, etc.), the complexity of the memory hierarchy, and the vast amount of parallelism.

Most planned high-end HPC resources in the US and elsewhere will be heterogeneous machines as well. The Aurora machine[250] to be installed at the Argonne Leadership Computing Facility (ALCF) in 2021 will feature nodes equipped Intel Xeon CPUs and multiple Xe GPGPUs. The Frontier machine[251] to be installed at the Oak Ridge LCF (OLCF) will feature nodes equipped with AMD CPUs and multiple AMD GPGPUs. Heterogeneous architectures are also considered for the China's exascale system planned for deployment in 2020.[252] However, despite the cited advantages of heterogeneous HPC platforms, homogeneous platforms are not going to disappear from the scene. For example, the pre-exascale computer Fugaku,[253] recently deployed at the RIKEN Center for Computational Science in Japan and in June 2020 Top500 list becoming the world's top supercomputer, features a "homogeneous" architecture, with a large number (158,976) of "thin" nodes equipped with custom A64FX processor based on the ARM technology.

Lastly, we should mention a number of emerging computing technologies that could have an impact on our field in a not-too-distant future. Application-specific integrated circuit (ASIC) devices, such as the tensor processing unit (TPU) of Google[254] and tensor cores in NVIDIA Volta and Ampere architectures,[255] aimed at applications in machine learning (ML) could be used for some computational tasks in electronic structure. Similar benefits can be achieved with general-purpose programmable hardware, such as Field Programmable Gate Arrays (FPGAs),[256] which could be used for custom tensor algebra, integral evaluation, etc. Of course, a quantum computer could also be viewed as a specialized coprocessor/accelerator for a classical HPC platform.

5. DEPLOYMENT OF MANY-BODY QUANTUM CHEMISTRY TO PARALLEL COMPUTERS

In this section, we will review the existing literature on distributed-memory parallel implementation of many-body electronic structure methods. (We did our best to discuss all implementations whose details of distributed-memory parallel execution we could find from public sources. In the interest of keeping the presentation focused we excluded implementations that only target shared memory platforms.) We start out by discussing canonical formulations of the many-body methods on conventional (nonheterogeneous) platforms, followed by discussions of reduced-scaling variants and the implementations for heterogeneous platforms. Because our focus is on high-performance implementations, we discuss those works that do not use global shared storage, like parallel file system, to share data due to the poor scalability of such approaches.

In discussing parallel performances, we will refer to the traditional performance measures for parallel programs, such as (parallel) speedup and efficiency,[257] only slightly modified. The speedup corresponding to an increase in the number of "processors" (nodes, unless specified otherwise) from p_i to p_f will be defined as

$$S_{p_i \to p_f} \equiv \frac{T(p_f)}{T(p_i)} \tag{35}$$

where $T(p)$ is the value of a performance metric (wall time, unless specified otherwise) for a computation utilizing p processors. The ideal value for $S_{p_i \to p_f}$ is p_f/p_i. The traditional term "speedup on p processors" thus corresponds to $S_{1\to p}$, with the ideal value for speedup equal to p. (Parallel) efficiency of metric T will be defined as

$$E_{p_i \to p_f} \equiv \frac{p_i T(p_f)}{p_f T(p_i)} \tag{36}$$

The traditional term "efficiency on p processors" thus corresponds to $E_{1\to p}$. The ideal value for efficiency is 1, or 100%. Both of these measures correspond to the computations for a fixed problem size, thus are useful to characterize strong scalability.

5.1. Many-Body Perturbation Theory

The most popular, and simplest, many-body method in chemistry is the Møller−Plesset perturbation theory truncated at the second-order for energy (MP2). Despite its historical significance and its continued utility in some contexts, it is generally not accurate enough for chemical applications. It is also fairly simple from the computational standpoint, thus only brief remarks will suffice. There are many parallel implementations of the MP2 method; a detailed analysis of several representative algorithms for the canonical MP2 method with exact evaluation of integrals and a brief review of older algorithms can be found in the book by Janssen and Nielsen.[258] Modern implementations of the MP2 method almost always use some numerical approximations, such as density fitting, Laplace transform, and related techniques. A number of massively parallel variants of DF-MP2 have been described in the literature.[259−262] Approximate variants of canonical MP2, such as DF-MP2, are much cheaper than the

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

prerequisite Hartree−Fock calculation until the system size approaches 100 atoms or so (the crossover depends on the basis and the nature of the system); by that point the reduced-scaling variants of MP2 become faster (e.g., see Pinski et al.,[104] Werner et al.[106]). Thus, for all practical purposes MP2 is never a bottleneck.

An important use case for perturbation theory are problems where single Slater determinant does not provide a good zeroth-order description of the wave function. For small molecules the multireference configuration interaction and related infinite-order methods are still the most robust approaches, with parallel implementations long established[263] and further developments continuing.[264,265] For larger systems the most robust approach is the multireference perturbation theory, or more generally, quasidegenerate perturbation theory. Few distributed memory parallel implementations of multi-reference perturbation theories exist.[266−268] Note the recent development of analytic gradient formalism[43] that in conjunction with efficient parallel implementation allowed realization of *ab initio* direct multistate dynamics.[269]

5.2. CCSD

Early efforts to deploy conventional CCSD to distributed-memory parallel machines were reviewed by Watts.[270] A more recent overview was presented by Peng et al.[98]

5.2.1. Pioneering Work. The first implementation of the conventional CCSD energy for a distributed memory computer was reported in 1992 by Rendell, Lee, and Lindh.[271] It was restricted to closed-shell systems and used the spin-adapted CCSD formulation of Scuseria, Janssen, and Schaefer.[27] Due to the limited memory available per node, the authors designed their algorithm to optimize for space, with only $O(o^2v)$ memory required on each node, with the rest of the data stored on a distributed file system. Due to this, the permutational symmetry was only partially exploited, with the most expensive PPL term (eq 31) ≈ 4 times more expensive than optimal.[27,272] The performance was evaluated on Intel i860 parallel computers with up to 128 nodes. Modest scalability was demonstrated: for the valence triple-ζ CCSD computation for the HCNO molecule, $E_{1\rightarrow16}$ = 34.5% was demonstrated, with steep decrease in efficiency upon further increase of the processor count. The parallel disk I/O was identified as the bottleneck, with the dominant contribution to the I/O costs coming from the 2-electron integrals with 3 and 4 unoccupied indices. To reduce the I/O bottleneck, the authors proposed a semidirect algorithm in which the contributions of integrals with 3 and 4 unoccupied indices to the doubles amplitude equations were computed in an integral-direct fashion, with Gaussian AO integrals computed on the fly; the rest of the terms were evaluated as in the base algorithm using the MO integrals in persistent distributed storage. The authors estimated the potential benefit of the semidirect algorithm by replacing disk I/O for the integrals with 3 and 4 unoccupied indices with using local buffers with dummy (constant) values; the simulated semidirect algorithm resulted in an excellent strong scaling, with $E_{1\rightarrow64}$ = 74.4%. Note that this is clearly an upper bound to the efficiency of the real implementation because the costs of the AO integral evaluation were not included in the simulated calculations. Nevertheless, this was an important milestone as this work demonstrated the critical importance of communication minimization in parallel many-body methods like CCSD. It also demonstrated the utility of integral-direct formalisms for communication minimization. The authors also discussed the potential benefits of replicating amplitudes and intermediates for communication minimization.

It is important to understand that the integral-direct algorithm is just another instance of a time-space trade-off. Namely, the AO integral set is computed every iteration, at the $O(N^4)$ cost (or even $O(N^2)$, if the sparsity of the AO integral tensor is taken into account), to avoid globally accessible storage of 2-electron integrals with 3 and 4 unoccupied indices and the communication costs associated with these integrals. How do these communication costs scale with the system size? For simplicity, consider the PPL term (eq 31). It can be viewed as nothing but a multiplication of an $o^2 \times v^2$ matrix by a $v^2 \times v^2$ matrix producing an $o^2 \times v^2$ matrix. With all the matrices distributed among processors, the communication cost of the parallel matrix multiplications are asymptotically the same as the operation count. Thus, the communication cost of the PPL term (and the CCSD itself) for all data fully distributed is $O(N^6)$, with the prefactor determined by the particular algorithm used to implement the distributed tensor contraction. It is clear that the extra $O(N^4)$ cost of computing the AO integrals every CCSD iteration is (asymptotically) insignificant as it allows to replace the $O(N^6)$ amount of communication.

Additional improvements to the base algorithm of Rendell et al.[271] along with larger computations were reported by Rendell, Guest, and Kendall in a follow-up paper in 1993.[273] A key advance here was the use of one-sided data accesses implemented using interrupt handlers. This allowed the use of global distributed memory rather than parallel file system to store amplitudes and intermediates.

5.2.2. NWChem. A complete distributed-memory implementation of the semidirect algorithm for closed-shell CCSD was demonstrated in 1997 by Kobayashi and Rendell[274] in the NWChem[275] program. Although the authors call this algorithm "direct", the on-the-fly evaluation of integrals is only used for some terms, namely those involving the integrals with 3 and 4 unoccupied indices, just like in the original proposal of Rendell et al.[271] Instead of the use of interrupt handlers as in ref 273, one-sided memory accesses were implemented using the Global Array (GA) library.[276,277] GA is a portable high-level abstraction for distributed multidimensional arrays that provides one-sided accesses to the data (put, get, accumulate, scatter, gather, etc.) and some collective operations such as linear algebra (the original GA API supported two-dimensional arrays only, but the recent GA API supports up to arrays with seven dimensions). GA is a Partitioned Global Address Space (PGAS) abstraction, as it represents distributed memory as a single address space partitioned between application processes; hence, the GA is often described as a "shared-memory" programming model for a distributed memory machine. Although the extension of the GA abstraction to disk-based arrays (Disk Resident Arrays) has been implemented, by default, GA arrays reside in the main memory (dynamic random-access memory, or DRAM); thus, the per-process memory use by this algorithm is $O(o^2v^2p^{-1})$, where p is the number of processors. The implementation exhibited excellent strong scaling: the all-electron CCSD computation on glycine in cc-pVDZ basis attained $E_{16\rightarrow256}$ = 71% on Cray T3D. Similar results were obtained on the faster Cray T3E machine: the same computation attained $E_{8\rightarrow64}$ = 101%, i.e. superlinear scaling.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

The CCSD implementation of Kobayashi and Rendell[274] in NWChem was further improved by Anisimov et al.[278] by replicating the (symmetrized) doubles amplitudes to reduce the communication volume in the integral-direct CCSD algorithm. This specifically eliminated all remote reads at the cost increased memory requirement per node; note that remote accumulation was still needed. The improved CCSD implementation demonstrated $E_{1100\rightarrow 20\,000} \approx 30\%$ for a 2-base DNA fragment in an augmented triple-ζ basis (63 atoms, o = 103, n = 1042) on a Cray XE6 cluster (NCSA's "Blue Waters" machine). Although the strong scaling was good, the absolute efficiency of these calculations was shown later to be relatively low by Peng et al.[279]

The Tensor Contraction Engine (TCE)[232,280] furnishes another family of implementations of the conventional coupled-cluster methods in NWChem. TCE consists of a many-body algebra compiler for derivation of programmable equations and their symbolic optimization (e.g., by common subexpression elimination, strength reduction, etc.) and a code generator that transforms the tensor expressions into low-level FORTRAN code. The generated code uses Global Array toolkit to implement tensor algebra using one-sided distributed memory operations. NWChem's TCE module supports a number of CC methods for ground and excited states, both in spin−orbital form and with partial utilization of the spin symmetry for closed-shell systems. All TCE methods are formulated in the MO basis, i.e., integral-direct evaluation is not employed. The TCE compiler was recently improved by evaluating several parts of the tensor expression in parallel. This coarse-grained parallel evaluation strategy, termed alternative task scheduling (ATS),[281] improved the strong scalability of the code. For example, the ATS-enabled CCSD exhibited 84% parallel efficiency on a commodity infiniband cluster when the core count increased from 768 to 3072.[282] The TCE-based CCSD algorithms in NWChem have their data fully distributed in memory; hence, the per-process memory requirement is $O(v^4p^{-1})$, which is much larger than that of the implementation by Kobayashi and Rendell[274] and all TCE CC computations require large-scale HPC platforms (i.e., large p).

5.2.3. PQS. The Parallel Quantum Solutions (PQS) software suite from the Pulay group includes efficient AO-driven implementation of CCSD and CCSD(T).[283,284] The implementation uses closed-shell spin-adapted generator-state[26] CCSD formalism of Hampel et al.,[230] which is closely related to that of Scuseria et al.[27] The distributed data is managed by the Array Files (AF) middleware,[285] which stores the data on each node's local disk (thus avoiding the use of complex distributed file systems); global one-sided access to the disk-resident data is provided to greatly simplify programming. Calculations on systems as large as a benzene dimer with 1512 basis functions without symmetry were demonstrated using the quadratic configuration interaction singles and doubles method (closely related to CCSD). The CCSD program exhibited 90% parallel efficiency upon increase in node count from 2 to 16.[284] Notably, the integral-direct evaluation of the CCSD doubles residual evaluation exploits the AO integral sparsity to reduce the formal arithmetic complexity.

5.2.4. CFOUR. Parallel implementation of closed- and open-shell CCSD energies and up to its second geometric derivatives was developed in the CFOUR package[7] by Harding et al.[286] The amplitudes are replicated to simplify parallelization and reduce communication. Only the time-determining steps are parallelized, which limits the parallel efficiency for smaller basis sets.

5.2.5. Molpro. Parallel implementation of coupled-cluster models in Molpro[8] exists, but its details are not described. The amplitudes seem to be replicated to simplify parallelization. The distributed data abstraction, implemented on top of the GA toolkit, used to implement the parallel CC methods in Molpro was described by Wang et al.[287]

5.2.6. GAMESS. The GAMESS package includes two implementations of parallel CC. The original CCSD and CCSD(T) implementations[288] developed by the Gordon group utilized a hybrid approach (similar to that of Kobayashi and Rendell[274]) in which some terms were evaluated in AO basis using integral-direct approach; these implementations utilized the spin-adapted CC formalism of Piecuch et al.[289] This implementation managed distributed data using the Distributed Data Interface (DDI) library and utilized hybrid parallelism (message passing + interprocess UNIX System V communication) to optimize intranode data sharing. Modest parallel efficiency was demonstrated, with the MO-basis tensor contractions in CCSD identified as the primary bottleneck.

Another implementation of parallel CC was developed by Asadchev and Gordon,[290] first as a standalone library and then integrated into GAMESS. The key innovation of their work was, similarly to the PQS implementation, to explicitly manage disk storage as part of the memory hierarchy, with one-sided access to distributed memory provided by the Global Arrays toolkit. Another innovation was the use of explicit thread-level programming for intranode parallelism. Yes another innovation was the ability to offload matrix multiplication work to CUDA-enabled NVIDIA accelerators. The CCSD implementation exhibited parallel efficiency as high as ≈83% upon core increase from 24 to 96 on a small commodity cluster; efficiency of ≈93% was attained on a high-end Cray XE6 supercomputer upon core increase from 256 to 1024.

5.2.7. ACES. The ACES III package[6] is the result of re-engineering the ACES II suite of Bartlett et al. for massively parallel architectures. It includes a full lineup of ground- and excited-state coupled-cluster methods, e.g., CCSD and CCSD-(T) energy and forces are available for both closed- and open-shell references. A custom domain-specific language (DSL), the super instruction assembly language (SIAL), was used to implement all methods; the DSL programs are executed by the super instruction processor (SIP) interpreter.[228,291] The DSL statements are implicitly parallelized by distributing "super-instructions" (i.e., operations on individual tensor blocks) to the execution agents residing on each node. The DSL does include explicit control of data I/O. Excellent strong scaling (sometimes, superlinear) was demonstrated for CCSD[292] and other CC methods.[293] On a Cray XT5 cluster 80% of parallel efficiency was attained when the number of processors was increased from 2000 to 8000.[228]

The development of the ACES series of programs continued recently with Aces4, the reengineered version of the ACES III program using updated DSL and the DSL interpreter.[294] The major development is the efficient support for block-sparse tensors with high degree of sparsity. This allowed implementation of efficient fragment-based reduced-scaling methods. A variety of conventional CC methods has been implemented. Performance of the CCSD energy was demonstrated for 1-d arrays of Ne atoms with 15, 20, 25, and 30 atoms on a Cray XC40 cluster (the Excalibur system located at the U.S. Army

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

Research Laboratory). Our conservative interpretation of the Ne_{15} aug-cc-pVTZ CCSD performance data in Figure 5 in Sanders et al.[294] is $E_{700\rightarrow 20\,000} \lesssim 25\%$ (processor counts refer to cores).

5.2.8. CTF. Several implementations of CC methods have been recently realized with the help of the Cyclops Tensor Framework (CTF) of Solomonik et al.[225] Most notable of these include Devin Matthews' Aquarius code[295] as well as the commercial Q-Chem suite.[10,296] CTF is a library that implements basic algebra of distributed-memory dense and element-sparse tensors. CTF is implemented using MPI and OpenMP. CTF includes communication-optimal algorithms for tensor contraction (e.g., implemented using the 2.5D[297] SUMMA[298] for distributed matrix multiplication). The Aquarius program includes a number of ground- and excited-state MO-basis CC methods, up to full treatment of quadruples, with all tensors fully distributed in memory. Excellent strong scaling of the CCSD implementation was demonstrated on a large-scale Cray XC30 cluster;[295] e.g., parallel efficiency is conservatively estimated at ≈50% for a cluster of 15 water molecules in cc-pVDZ basis upon increasing the node count from 16 to 256. The CTF-based CCSD exhibited better parallel efficiency than the TCE-based CCSD in NWChem; however, it is not clear if the NWChem's CCSD code included the improvements of ref 281 (for the timings reported by the NWChem team please see ref 299). The sequential performance of Aquarius was worse than that of NWChem, and modest scaling with the number of threads was demonstrated.

CTF can also be used as a backend to libtensor,[222] a C++ tensor library designed with support for symmetries necessary for implementing quantum many-body and used to implement the coupled-cluster features of the popular Q-Chem package.[10] Ibrahim et al. assessed the parallel performance of CCSD implemented using libtensor.[296] Both the native shared-memory backend of libtensor and the CTF-based distributed memory backend were used. On a single node the native backend outperformed the CTF-based one, due to the more effective utilization of symmetry in the tensor contraction expressions. Several test systems and five different supercomputing platforms were considered. For a medium-sized system, methylated uracil−water cluster ($o = 58$, $n = 410$), the CTF-based backend demonstrated good strong scaling ($E_{32\rightarrow 128} \approx 64\%$) on the Cray XC30 Cori system at NERSC, and was roughly 4× faster than NWChem. For the largest system, a AATT nucleobase tetramer in a triple-ζ basis ($o = 98$, $v = 830$), computations were performed on the Titan supercomputer at OLCF,[300] and the Mira supercomputer at ALCF[249] with as many as 8192 nodes each (131 072 cores); at such scale, the majority of the execution time was attributed to the communication.

5.2.9. MPQC. An unfortunate fact about most implementations discussed so far is that their absolute efficiency, defined as the percentage of the effective FLOPS rate to the hardware peak, is relatively low, usually <10% (few exceptions include refs 284 and 290). Of course, most literature does not include the absolute performance analysis, so the efficiency can only be coarsely estimated. Because for realistic CCSD calculations $o \ll v$, the cost of CCSD is usually dominated by the PPL term. For the optimal implementation of closed-shell CCSD, the cost of the PPL term is $o^2v^4/2$, but some implementations above, such as the TCE-based CCSD, use spin−orbital formalism (where its cost is $5o^2v^4/2$, i.e., 5 times higher) or do not use the permutational symmetry of the tensors (then the PPL cost is $2o^2v^4$). The integral-direct implementations also need to account for the cost of AO integral evaluation, for which there is no simple performance model, but the higher-order complexity of the tensor algebra in CCSD means that most of the time, the cost of the integral evaluation is relatively small. Thus, using the $2o^2v^4$ FLOP count divided by the peak hardware performance is a reasonable a rough performance model for a single iteration of closed-shell CCSD. Interested readers are invited to use this performance model to gauge the absolute efficiency of the above implementations for themselves.

To develop a development platform for many-body quantum chemistry that is strongly scalable on the highest-end HPC systems *and* has competitive absolute performance on small clusters and even one node, we re-engineered the Massively Parallel Quantum Chemistry program (MPQC),[301] originally developed by Curtis Janssen et al., to use of the high-performance block-sparse tensor algebra framework TiledArray. An efficient implementation of closed-shell conventional and explicitly correlated CCSD was developed by Peng et al.[98] in MPQC (version 4) in 2016.

To support both memory-limited platforms as well as high-end platforms, several closed-shell CCSD formulations were implemented: conventional, where all integrals in MO basis are held in memory, with per-node memory requirement of $O(v^4p^{-1})$, density fitting, where all integrals are approximated by the density fitting, with per-node memory requirement of $O(v^4p^{-1})$, DF-AO-hybrid (where contributions from the integrals with 3 and 4 unoccupied indices are evaluated in AO basis in direct fashion, just like in Rendell et al.,[271] whereas the rest of the terms are handled via DF, with per-node memory requirement of $O(o^2v^2p^{-1})$), and DF-direct, added later,[279] in which the DF reconstruction of integrals with 3 and 4 unoccupied indices was done lazily, on a tile-by-tile basis, with the rest of the terms evaluated as in the DF approach; this variant is numerically equivalent to DF-CCSD but has a reduced per-node memory requirement of $O(o^2v^2p^{-1})$, the same as DF-AO-hybrid.

The use of DF does not change the asymptotic complexity of CCSD and only slightly reduces the $O(N^6)$ cost prefactor,[98] but it is absolutely essential to reduce the costs of the explicitly correlated (F12) variant of CCSD. The F12 terms are minor contributors to the overall cost due to their noniterative evaluation and have the same complexity as the CCSD itself.

The implementation used the factorization of Scuseria et al.[27] However, the permutational symmetry was not taken into account; thus, the PPL term has the suboptimal $2o^2v^4$ cost. Despite this drawback, the high efficiency and strong scalability makes the implementation competitive to existing optimized implementations. On a single node the conventional and DF implementations of CCSD scaled nearly perfectly with the number of threads and matched the efficiency of the popular implementations in Orca[9,302] and Psi4,[303] respectively, despite the suboptimal use of permutational symmetry in MPQC. Similar favorable comparison was found for the explicitly correlated CCSD method in MPQC when compared to Orca.

On the BlueRidge commodity cluster[304] the DF-AO-hybrid CCSD implementations demonstrated good strong scaling for a water 10-mer in a realistic basis ($o = 40$, $v = 430$): 10.4× speedup is achieved on 16 nodes, relative to 1 node, and 16.4×

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

on 32 nodes (≈ 65% and ≈51% parallel efficiency, respectively). Because the noniterative F12 energy contribution does not scale as well but due to its low cost, it lowers the overall efficiency only slightly. The absolute efficiency was high: the calculation of the PPL term utilized 85% of the machine peak on 1 node and 70% of the machine peak on 32 nodes. Later, benchmarks of the DF-direct CCSD implementation on uracil trimer produced similar parallel efficiencies.[279]

Similar performance was observed on the high-end Mira system at ALCF:[249] for the $(H_2O)_{12}$ cluster, the DF-AO-hybrid explicitly correlated CCSD method reached $E_{128\rightarrow1024} \approx 50\%$, where the processor counts refer to nodes (each node of Mira has 16 cores). Computations on much larger systems, e.g., a nanotube fragment C_{84} ($o = 168$ and $v = 924$), were performed on 1024 nodes of Mira.

High efficiency of the implementation allowed Peng et al.[98] to compute the CCSD binding energy of the uracil dimer with high precision and reveal substantial errors in the previous benchmark values. The excellent precision of the CCSD binding energies were later confirmed by the data from Brauer et al.[305]

The DF-direct CCSD implementation was compared[279] to the improved Kobayashi-Rendell CCSD implementation in NWChem using the reference GC-dDMP-B benchmark ($o = 103$, $n = 1042$) of Anisimov et al.[278] The MPQC implementation of DF-CCSD was conservatively estimated to be more efficient by a factor of ≈25.

5.2.10. FHI-aims. An efficient massively parallel implementation of DF-CCSD was reported by Shen et al.[306] in the FHI-aims package.[307] The implementation has an improved handling of the permutational symmetry, e.g., with the PPL term costing o^2v^4, i.e. half of that in MPQC. Thus, the overall efficiency was roughly twice that of MPQC. The communication minimization was achieved by the replication of data along one dimension of a 2D grid, roughly corresponding to SUMMA-like replication strategy. The t_1-based factorization of DF-CCSD was utilized.[308] Excellent intranode and strong internode scalings were observed, e.g., for the beta-carotene in a triple-ζ basis ($o = 108$, $u = 884$) $E_{32\rightarrow512} = 66\%$ was observed.

5.2.11. MRCC. An efficient parallel implementation of DF-CCSD was reported by Gyevi-Nagy et al.[309] in the MRCC package. The implementation has optimal handling of the permutational symmetry, i.e. the PPL term costs $o^2v^4/2$, or quarter of the cost in MPQC and half the cost in FHI-aims, with the rest of the $O(N^6)$ terms carefully optimized. The implementation also uses factorization of DF-CCSD due to DePrince and Sherrill.[308] Excellent intranode scaling is observed along with an excellent strong scaling up to 8 nodes, $E_{1\rightarrow8} = 66\%$. The authors noted that the strong-scaling of $O(N^6)$ terms to more nodes is possible, but the lower-scaling terms do not scale well. Also, it appears that amplitudes *and* the three-index integrals were replicated, thus contributing to the excellent strong scaling.

5.3. CCSD(T)

The (T) correction in canonical form is an ideal candidate for parallelization due to its time-determining role in the popular CCSD(T) method and relative ease of parallelization. Early efforts to deploy conventional (T) to distributed-memory parallel machines were reviewed by Watts.[270] A more recent overview was presented by Peng et al.[279]

5.3.1. Pioneering Work. First distributed-memory implementation of a CCSD(T)-like method was reported by Rendell et al.[310] Namely, their work included only the fourth-order (in the closed-shell Møller−Plesset sense) contributions to the (T) energy correction. However, their parallelization strategy and implementation can both be directly applied to the evaluation of complete (T) correction.

Evaluation of the (T) correction to CCSD energy involves a set of reductions of pairs of order-6 tensors, each with three active occupied indices and three unoccupied indices. To keep the storage requirements manageable these tensors are evaluated in blockwise fashion as needed, utilized in binary reductions, and discarded. Rendell et al.'s considered two evaluation strategies: *abcijk* and *aijkbc*, in which reductions over three unoccupied (*abc*) indices and an unoccupied-occupied index pair (*ai*) was task-parallelized, respectively. The *abcijk* algorithm was nearly perfectly scalable when all integrals were replicated. However, because the largest set of integrals has ov^3 elements, this mandated for all but the smallest computations storing integrals on a parallel file system and reading them in batches. The disk-based *abcijk* algorithm was found to be nonscalable due to the $O(ov^4)$ complexity of file I/O. Due to the lower complexity of file I/O ($O(o^2v^3)$) the *aijkbc* algorithm demonstrated excellent strong scaling, albeit having an operation count greater than the optimal by a factor of four (due to the suboptimal use of symmetry). Another notable feature of the *aijkbc* algorithm is its modest per-node memory requirement of $O(ov^2)$.

5.3.2. NWChem. Kobayashi and Rendell[274] presented first complete implementation of distributed-memory CCSD(T) in NWChem. Their implementation followed the *aijkbc* algorithm of Rendell et al.,[310] but instead of disk-based storage[310] integrals and amplitudes were stored in memory, with one-sided data operations provided by the Global Arrays (GA) library. Linear scaling was demonstrated for glycine ($C_2O_2NH_5$; $o = 20$, $v = 80$) on Cray T3E when the number of processors increased from 8 to 64. Aprà et al.[235] further improved the (T) code by reducing the communication volume and the memory footprint. The improved code was used to benchmark the CCSD(T) binding energies of water clusters. The total CCSD(T) evaluation (including CCSD) for water 18-mer in a triple-ζ basis set (918 basis functions) demonstrated speed-up of ≈2.2 from 30 000 to 90 000 processors on ORNL's Cray XT5 "Jaguar" supercomputer. Evaluation of the (T) energy for the 20-mer (1020 basis functions) took ≈2 h and used 100TB memory and 96 000 cores of XT5, reaching 487 TFLOPS (55% of theoretical peak).[235]

Another implementation of CCSD(T) in NWChem uses the Tensor Contraction Engine (TCE).[232] TCE fully distributes all of its tensors in GA-managed memory. Tensor algebra is implemented in TCE as a statically scheduled sequence of tasks on the tensor tiles; each task pulls its data from distributed memory using one-sided (blocking) reads managed by the GA toolkit. The TCE implementation differs from most other (T) implementations in that the reduction over all six indices, rather than two or three, is parallelized. Such fine-grained parallelization strategy increases the amount of exploitable parallelism at the cost of increased communication requirements. The TCE implementation of the (T) correction in the CR-EOMCCSD(T) method (closely related to the traditional (T) correction of the ground-state CCSD(T) method) was applied to the FBP-f-coronene molecule in a triple-ζ basis (780 basis functions), with the triples correction

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

reaching 84% parallel efficiency upon increasing the number of cores from 60 000 to 210 000.[281] The TCE-based implementation of the (T) corrections was extended to exploit heterogeneous systems with NVIDIA's GPGPUs[311,312] and the Intel Xeon Phi coprocessors[313] (see below).

5.3.3. GAMESS. The first parallel implementation of CCSD(T) in GAMESS was developed by Bentz and Ryan et al.[288,314] The Distributed Data Interface(DDI) library was used to manage the data distribution and (one-sided) transport. The (T) implementation uses the *ijkabc* algorithm (first described by Rendell et al.[315]) in which the reduction over $i \geq j \geq k$ is task-parallelized. The integrals were fully distributed and transported to workers as needed, however the doubles amplitudes were not distributed. The implementation demonstrated ≈72% parallel efficiency on 24 processors of an IBM SMP cluster for the T-shaped benzophenol-benzene dimer benchmark (o = 33, v = 313).[314]

Asadchev and Gordon[290] developed another CCSD(T) implementation within the *libcchem* library embedded in GAMESS. Their (T) implementation used the *abcijk* algorithm, with the *abc* reduction task-parallelized over nodes (the same algorithm was earlier utilized by Janowski and Pulay[284] in their (T) program; see below). This choice was motivated by the desire to minimize the per-node memory requirements and maximize the degree of exploitable parallelism (i.e., the number of tasks). Linear scaling was demonstrated for the $C_8H_{10}N_4O_2$ molecule in a triple-ζ basis on a commodity cluster with the number of nodes varied between 4 and 16. Speedup of 3.7 was obtained for the $SiH_4B_2H_6$ molecule in a quadruple-ζ basis on a Cray XE6 system when the number of cores increased from 256 to 1024.

5.3.4. ACES. The ACES III package[292] supports evaluation of spin-restricted and spin-unrestricted CCSD(T) energy and forces on distributed memory machines. As discussed in section 5.2.7, all electronic structure methods in ACES III are implemented using a domain-specific language. The (T) implementation in ACES III thus appears to completely distribute all tile level tasks, thus it is most similar to the NWChem TCE (T) implementation. Parallel scaling of the UHF (T) implementation was examined for the Ar_6 benchmark in a triple-ζ basis. The execution time reduced from 784 to 131 min when the processor count increased from 32 to 256; this corresponds to ≈75% parallel efficiency.

5.3.5. PQS. The PQS package provides a (T) energy implementation utilizing disk-resident data managed by the Array Files library.[284] The implementation is thus intended for the use on clusters with local disk storage. The use of the *abcijk* algorithm ensures minimal the per-node memory footprint, of only $O(o^3)$, that does not depend on the number of unoccupied orbitals and thus is suited for the (T) computations with very large basis sets. Excellent strong scaling was demonstrated for the aspirin molecule in a triple-ζ basis: speedup of 27.8 was obtained when the number of processors increased from 1 to 32. By taking advantage of the point group symmetry it was possible to perform among the largest canonical (T) energy computations ever, using modest computational resources. For example, the (T) energy of benzene dimer (C_{2h} symmetry) in a quadruple-ζ basis (1512 basis functions) was performed in 57 h using 32 nodes of a commodity cluster.

5.3.6. MPQC. An efficient closed-shell (T) implementation in the latest version of MPQC was recently developed by Peng et al.[279] To be able to deploy the (T) method on high-end machines and on compute devices with limited memory (such as accelerators), the *abcijk* strategy was chosen as it minimizes per-task size of $O(N^6)$ intermediates and maximizes the number of tasks; the trade-off is the smaller amount of work available per each task. Thus, similarly to the blocked algorithm of ref 290, this algorithm's formal maximum storage requirement per MPI process is $3 \times O^3 b_v^3$, where b_v is the tile size for the unoccupied orbital range (this can be controlled by the user). All integrals and amplitudes are distributed, but the integrals with 1 unoccupied index can be optionally replicated. Because the blocking of the unoccupied range optimal for (T) is different from that for CCSD, the doubles amplitudes and integrals are partially reblocked before (T).

The implementation demonstrated perfect intranode linear scaling with respect to the number threads on a single 16-core node of the BlueRidge commodity cluster.[304] Excellent strong scaling with respect to the number of nodes was demonstrated on BlueRidge: $E_{1\rightarrow 32} \approx 80\%$ was measured for the cc-pVDZ (T) computation on $(H_2O)_{14}$. This made very large calculations accessible on a commodity cluster, such as pentacene dimer in cc-pVDZ basis, with 72 atoms and 756 basis functions; the (T) correction took ≈25 h using 32 nodes (512 cores, with only ≈10 TFLOPS total peak performance).

The absolute CCSD(T) efficiency of MPQC was compared against NWChem's (T) code for the GC-dDMP-B benchmark reported by Anisimov et al.[278] (at the time, it was the largest conventional CCSD(T) calculation reported). The MPQC time to solution for the (T) correction was 47.4 h on 64 nodes of the BlueRidge cluster vs 1.4 h on 20 000 essentially identical nodes used by Anisimov et al.[278] The absolute efficiency of MPQC in this calculation is estimated to be ≈44%. Further improvement is possible by the optimization of the tensor contraction and permutation kernels.

5.3.7. FHI-aims. An efficient massively parallel implementation of DF-CCSD(T) was reported by Shen et al.[306] in the FHI-aims package.[307] Technical details of the (T) implementation were not discussed, but some parallel performance data was given. For the $(H_2O)_{10}$ cc-pVDZ benchmark, $E_{1\rightarrow 8} \approx 95\%$ was observed on a commodity cluster.

5.3.8. MRCC. An efficient parallel implementation of DF-based (T) was reported by Gyevi-Nagy et al.[309] in the MRCC package. This implementation uses the *ijkabc* loop nest, with 1 occupied index (k) distributed across MPI processors and the remainder partitioned among threads using OpenMP parallel constructs; the use of 1 loop for the MPI parallelization is likely due to the focus on smaller HPC setups like a typical small commodity cluster. Near-perfect scaling with the number of threads and absolute efficiency of ≈65% was observed on 1 node. Effects of NUMA on the performance were evaluated, with 1 MPI rank per NUMA region (i.e., socket) recommended to gain more than 10% efficiency. Excellent strong scaling was observed up to 8 nodes, with $E_{1\rightarrow 8} = 88\%$ for the $(H_2O)_{14}$ cc-pVDZ benchmark, also used to evaluate MPQC. A number of CCSD(T) energies were computed for systems with up to 43 atoms and 1569 basis functions on up to 224 cores.

5.4. CC Excited States and Properties

5.4.1. NWChem. The distributed-memory EOM-CC implementation based on TCE has been applied to several large systems with hundreds of electrons and basis functions.[281,282,316] Fan et al. studied the excitation energies of zinc-porphyrin using the parallel EOM-CCSD code in

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

NWChem. They improved the performance of Davidson eigensolver used in EOM-CCSD for both single-root and multiroot version. Parallel performance is demonstrated using zinc-porphyrin in the 6-31G* basis with around 3.4× speedup from 256 to 1024 processors.[317] Kowalski et al. has developed the completely renormalized EOM-CC method CR-EOM-CCSD(T) using TCE, the TCE generated codes have been further optimized manually.[281] The performance of CR-EOM-CCSD(T) has been reported on the green fluorescent protein (GFP), free-base porphyrin (FBP), oligoporphyrin dimer (P_2TA) and FBP-f-coronene system using over 1000 cores.[316,318] The strong scaling of EOM-CCSD is tested on Bacteriochlorophyll a molecule using the 6-311G basis, with 80−90% parallel efficiency from 2048 cores to 4096 cores.[281,319] Later on, Hu et al. extended the EOM-CCSD program in NWChem to the open-shell system.[282]

5.4.2. ACES. The massively parallel ACES III program also has implementations of CC methods for excited states. Similar to its ground-state CC methods, the implementations are based on the super instruction assembly language (SIAL). Kuś et al. reported two variants of open-shell EOM-CCSD implementation in ACES III: (1) storing the entire $\bar{H}$ matrix in memory and (2) evaluating the $\bar{H}$ with four-virtual indices on the fly.[293] The latter approach was found to be more efficient, which showed 7 times speed up from 32 to 256 processors on OH cytosine adduct with 25 occupied and 245 virtual orbitals. Verma et al. introduced a new massively parallel linear response CC module within ACES III, with applications to static polarizabilities of closed-shell molecules and open-shell radicals.[320]

5.4.3. MPQC. The MPQC4 program has distributed parallel code for excited-states, which includes a series of implementations of equation-of-motion coupled-cluster methods (EOM-IP/EA/EE-CCSD) for closed-shell systems. The implementation follows the same approach to its ground state CCSD code (see section 5.2.9 for detail), which is based on top of the TiledArray framework, with all amplitudes and intermediates distributed in memory. It also contains several variants including approaches with and without density-fitting, and with stored integrals or lazily evaluated integrals. The performance was demonstrated by Peng et al. on methylated uracil dimer with water (39 atoms, 882 basis functions) and 11-cis-retinal protonated Schiff base (51 atoms, 1050 basis functions) using up to 128 nodes (2048 cores).[321] A parallel efficiency between 60 to 70% is observed from 16 nodes to 128 nodes. The EOM-IP/EA-CCSD implementation has not been reported, but the IP implementation was used to benchmark the OAM24 data set CBS limit by Teke et al.[322]

5.5. Higher-Order CC

There are significantly fewer high-performance implementations of many-body methods that include explicitly three- and higher-body correlations, due to their extreme cost. Simple parallelization strategies where the data is (partially) replicated and only work is distributed are not feasible due to the high-order storage complexity of such methods (e.g., $O(N^6)$ for CCSDT). Thus, all data, such as amplitudes and other tensors, and all work must be distributed.

Parallel implementation of a number of high-order coupled-cluster methods were developed as part of the TCE effort in NWChem.[232] As with other TCE methods, spin−orbital formalism is used, with full exploitation of permutational symmetry at the level of tiles, but zero contributions due to spin are skipped. Unfortunately, no performance benchmarks were presented. An analysis of FLOP counts for the CCSDT code in NWChem vs automatically generated equations was presented by Lai et al.[234]

Efficient distributed-memory parallel implementation of ground- and excited-state higher-order CC methods (CCSDT and CCSDTQ) were developed by Matthews in the Aquarius package using the Cyclops Tensor Framework of Solomonik et al.[225] The performance of the CCSDT code was analyzed for small water clusters $[(H_2O)_n, n = 2, ..., 4]$ in cc-pVDZ basis and compared with the NWChem's CCSDT module on the Edison system at NERSC (now retired; see ref 225 for the hardware details). The absolute performance of the CCSDT code in Aquarius and NWChem on 1 node of Edison (460.8 GFLOPS peak) for $(H_2O)_2$ and $(H_2O)_3$ were nearly identical, but the NWChem code was not scalable; speedup of only ≈2 was achieved no matter how many nodes were used. The poor scalability of the TCE code is likely due to the lack of any kind of communication optimization: in a tensor contraction each argument tile is fetched to the location of a result tile and discarded immediately. In contrast, the CCSDT in Aquarius attained speedup of ≈8 on 256 nodes for the water dimer and of ≈32 on 256 nodes for the water trimer. The absolute performance of the CCSDT code in Aquarius was lower than that of CCSD, largely due to the significantly higher cost of transpositions involved in tensor (un)matricization ("folding", in the language of Solomonik et al.[225]) performed in tensor contractions in CTF. For example, a larger calculation on $(H_2O)_8$ spent only 16% of time in matrix multiplication on 256 nodes of Edison.

Implementation of the ground-state closed-shell CCSDT and its iterative approximations was developed recently by Rishi et al. in the MPQC package.[323] Because the parallel performance of this implementation has not been documented, we are reporting a preliminary representative benchmark here to be able to assess performance to that of the CTF-based CCSDT. Intra- and internode strong scaling of the CCSDT energy implementation in MPQC was assessed for the all-electron CCSDT energy computation on $(H_2O)_4$ in the cc-pVDZ basis on the NewRiver cluster.[242] Reasonable strong scaling was obtained with respect to the number of threads (not shown) and the number of nodes (Figure 1). On a single node, scaling from 1 to 24 threads resulted in an ≈12 speedup,

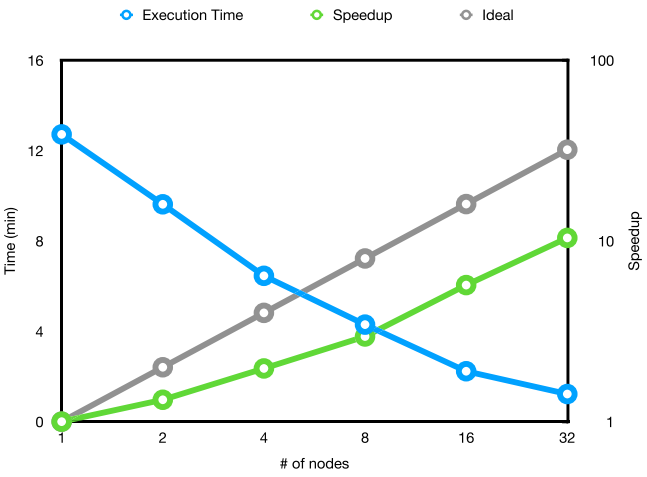


Figure 1. Internode strong scaling of the CCSDT energy implementation in MPQC (see text for details).

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

for ≈50% efficiency. Increasing the number of nodes from 1 to 32 resulted in a ≈12 speedup, or ≈38% parallel efficiency. The absolute performance of the CCSDT implementation in MPQC is comparable to that of Aquarius, even though there is no use of permutational symmetry at the moment in this code in MPQC: each CCSDT iteration using Aquarius took ≈60 s on 64 nodes of Edison Solomonik et al.,[225] which is in line with the ≈70 s per iteration using MPQC on 32 nodes of NewRiver (with identical peak performance to that 64 nodes of Edison).

5.6. Reduced-Scaling CC

Due to the complete decoupling of the fragment computations in the divide-and-conquer schemes, as we discussed in section 3.2, robust strong scaling can be achieved to hundreds of thousands of cores, if needed, with the main limitation being the load imbalance of the fragment computations.[176] Efficient parallelization (at least, intranode) of the monomer computations is also needed in general.

A more interesting challenge for high-performance implementation are fast reduced-scaling methods in which redundancies associated with fragment overlap are avoided. Only one strongly scalable implementation has appeared, due to the relative novelty of these methods. Efficient implementation of PNO-based MP2, CCSD, and CCSD(T) were reported by the Werner group in the Molpro package.[106,199−201,324] Global Arrays toolkit[276,277] is used to represent the data in distributed memory and provide one-sided operations. Each process also maintains its own local memory pool to keep (and cache, as needed) tensors. Partial replication of the data by caching it allows to save on communication costs. It appears that threads are not used for parallelization, i.e., one thread per MPI rank is used.

In all PNO methods, the two major steps are (1) generation and transformation of three-index integrals into localized occupied MO and/or the PAO bases and (2) local DF reconstruction of four-index integrals in PNO basis and evaluation of equation residuals. The first step is parallelized most efficiently over the batches of density fitting functions (in fact even serial codes decompose this step into task by density fitting batches[104]), whereas the latter is parallelized over pairs, with the redistribution of the three-index integrals in between the steps. Key to minimizing communication in the second step is caching and reuse of three-index integrals partially transformed to PNO basis;[200] this requires the use of static distribution of pairs among workers. To try to minimize the load imbalance bandwidth minimization by graph partitioning is used.

The PNO methods demonstrated nearly perfect strong intranode scaling. The internode scaling was tested on commodity small clusters with up to 16 nodes. The perturbative methods, namely MP2 and (T) (in localized orbital basis, even perturbative methods like MP2 and (T) requires iteration unless additional approximations are used), showed excellent strong scaling: PNO-MP2 attained speedup of ≈7 on 9 nodes,[106] and (T) scaled essentially perfectly from 3 to 16 nodes.[201] PNO−CCSD, however, did not scale as well: speedup of ≈4 was attained when number of nodes was increased from 2 to 12 (i.e., ≈67% parallel efficiency).[199] The worse scaling of PNO−CCSD is due to the significant communication overhead of the integral transformation and the load imbalance due to static pair distribution.

5.7. Heterogeneous Platforms

Recent emergence of heterogeneous architectures as cost-efficient compute platforms has spurred efforts to utilize such platforms in the context of electronic structure.[325] Due to the complexity of programming heterogeneous computers and the large memory footprint of many-body methods (which prevents fitting all data to accelerator memory) the deployment of many-body methods onto heterogeneous platforms has been slow and usually did not target heterogeneous distributed memory platforms. The pioneering efforts in this area utilized accelerators only for tensor contractions, implemented by matricizing the argument tensors followed by dense matrix multiplication on the device using the vendor-provided BLAS library. Dispatching the work into the accelerators can be implemented (1) by explicit management of the data and computation traffic to/from the accelerator(s), and (2) by offloading matrix multiplications to the accelerator, with the movement of the data to/from the accelerator memory performed automatically for every matrix multiplication task. The offloading approach makes the programming accelerators significantly simpler, but offers less control of data reuse and resource utilization.

DePrince et al. pioneered the use of accelerators in the context of iterative coupled-cluster methods (CCD/CCSD).[121,326] Their implementation manually managed the computation and data movement to the accelerator. Asadchev et al. also demonstrated a GPU-capable implementation of CCSD in GAMESS with explicit management of the data movement to/from the accelerator.[290] Eriksen[327] implemented MP2 and (T) methods for hybrid execution (on CPUs *and* on one or more accelerators) by offloading work via the OpenACC language extensions. Kaliman et al. extended the CCSD and EOM-CCSD implementations in the Q-Chem package by offloading the matrix multiplications to NVIDIA GPUs using the CUBLASXT API of cuBLAS library.[328] All of these developments were capable of executing on a single node with one or more NVIDIA GPU accelerators and utilized standard matrix multiplication kernels in the cuBLAS library.

Several groups have also developed more general kernels for tensor operations (contractions and permutations) arising in the context of post-CCSD methods. Ma et al.[311] improved the efficiency of tensor contractions on NVIDIA GPUs by generating optimized CUDA kernels via the use of index combining and dimension flattening. They also improved overlap of data movement to/from the GPU and the computation on the device by pipelining data motion. Lyakh introduced a standalone general-purpose library (TAL_SH) for performing basic tensor algebra on multicore CPUs and NVIDIA GPU-containing shared-memory computers.[329] TAL_SH implements tensor contractions by utilizing matricizing tensors via permutations implemented in the cuTT library[330] followed by matrix multiplication using the cuBLAS library. NVIDIA also recently provided the cuTensor library for tensor algebra primitives.[331] Another effort by Kim et al. introduced optimized device kernels for tensor contractions for the (T) method.[332] Their autogenerated CUDA kernels reached >60% of peak on NVIDIA's P100 and V100 GPUs by using loop fusion, register tiling and transposing, and shared-memory buffering. This effort culminated in the COGENT generator for high-performance tensor contraction kernels.[333] The distinguishing feature of COGENT from TAL_SH is that the latter permutes the argument tensors before their contraction [the so-called Transpose-Transpose-

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

GEMM-Transpose, (TTGT) strategy], thus consuming extra memory bandwidth; direct (permutation-free) tensor contraction in COGENT (similar approaches are also used in CPU kernels[227,334,335]) leads to the higher observed performance for the vast majority of the performance benchmarks considered by Kim et al.[333]

High-performance implementations of many-body methods for heterogeneous distributed-memory machines are reviewed next.

5.7.1. NWChem. The pioneering distributed-memory implementation of any many-body method for CUDA-enabled GPU platforms was the CUDA-enabled (T) code in NWChem demonstrated by Ma et al.[311,336] Speedup of ≈2.7 was demonstrated via hybrid execution on 7 CPU cores (Intel Xeon X5560) and 2 GPUs (NVIDIA Tesla T10) on each node, relative to using the 7 CPU cores only. Because the custom CUDA kernels written for this implementation implement tensor contraction directly (i.e., do not use the TTGT strategy), avoidance of the permutation on the CPU can be a contributor to the observed speedup, i.e., the use of non-TTGT kernels for the contraction on the CPU as well could make the CPU-only code faster as well and thus reduce the benefit from the GPU use. The data also illustrated the importance of pipelining, i.e. the data has to flow into and out of GPU in parallel with computation to maximally utilize the GPU resources. Excellent parallel efficiency was attained for both the CPU-only and hybrid code, $E_{16\rightarrow 64} \approx 98\%$ and $E_{16\rightarrow 64} \approx 89\%$, respectively. These developments were later used to evaluate the $O(N^7)$ terms in renormalized[337] and multi-reference[338] CCSD(T) methods.

A large scale application of the NWChem GPU-enabled code on a leadership platform was reported by Ma et al.:[339] the all-electron (T) energy of a single pentacene molecule (cc-pVDZ basis) was evaluated on 96 nodes of the Titan supercomputer at OLCF,[300] each node of which has one AMD Opteron 6274 CPU (281.6 GFLOPS peak) and one NVIDIA K20X GPU (1.3 TLOPS peak). The CPU-only computation utilizing all 16 cores (2 cores in the Opteron processor correspond to 1 floating-point unit) of the Opteron chip took 9240 s, while the hybrid execution took 1631 s, for an ≈5.7 speedup, which is slightly more than the theoretical speedup. The absolute efficiency of this computation seems in line with what we found in comparing the performance of the (T) CPU codes in MPQC and NWChem (e.g., see the valence computation on pentacene in Peng et al.,[279] which performs at most 4 times less work than the all-electron computation).

5.7.2. CTF. CTF can use GPU-capable BLAS library to perform tensor algebra on the GPUs. Ibrahim et al. assessed the parallel performance of CCSD implemented using GPU-capable CTF on the Titan system.[296] Unfortunately all tests on Titan were performed at such scale that the execution time was dominated purely by internode communication, thus little benefit from execution on the accelerator would be expected.

5.7.3. MPQC. Evaluation of the closed-shell (T) energy on CUDA-compatible GPUs was recently implemented in the MPQC package by Peng et al.[279] This development was powered by the CUDA extension of the TiledArray framework, to be available as a part of its first stable release. The final algorithm of the GPU implementation is the same as the CPU version; the only difference is the use of CUDA-specific tile type to parametrize the C++ type for the relevant tensors. Tensor contractions are implemented on the GPU using the TTGT approach, with permutations ("transposes", in the language of CS community) performed by a custom fork[340] of the open-source cuTT library[330] and GEMM performed by the NVIDIA cuBLAS library.[341] Management of the device memory pool is provided by the open-source Umpire library.[342]

It is instructive to examine the performance of several incremental variants of the final (T) algorithm that differed in the amount of work performed by the GPU. These variants were compared on a single node of the NewRiver cluster[242] equipped with two 14-core Intel Xeon E5-2680v4 CPUs (1075 GFLOPS total peak) and two NVIDIA P100 GPUs (≈9.4 TFLOPS total peak). Table 1). Variant **v1** only performed

Table 1. Multi-GPU Performance (in Minutes) of Different Versions of (T) Implementation of $(H_2O)_{14}$/cc-pVDZ on NewRiver

variant	v1[a]	v2[b]	v3[c]	final[d]
CPU	228			
CPU + 1×NVIDIA P100	468	336	138	67
CPU + 2× NVIDIA P100	270	186	72	39

[a]**v1** = GEMMs on GPU. [b]**v2** = **v1** + permutation on GPU. [c]**v3** = **v2** + use of CUDA UM memory pool. [d]final = entire (T) energy evaluation on GPU.

matrix multiplication (the "*G*" in "TTGT") part of the tensor contraction (nominally, this is the only step that costs $O(N^7)$), with the integrals and amplitudes transferred to the device memory and the resulting three-body intermediate copied back to the host. The initial variant was slower than the CPU-only implementation with both one and two GPUs utilized per node. Variant **v2** also performed tensor permutations (the "T"s in "TTGT") on the GPU, which provided substantial improvement. Variant **v3** used a memory pool to allocate the memory on the device, to minimize explicit allocation and deallocation of CUDA memory (the latter of which synchronizes the device and thus interrupts concurrent execution). This variant was the first in which even the use of 1 GPU provided a boost over the CPU-only code. The final algorithm contains all improvements of **v3** plus implements evaluation of outer products and the (T) energy evaluation on the GPU; this completely eliminates the need to copy any data other than the (T) energy contribution back to the host. The final implementation exhibits speedups relative to the CPU-only execution of {3.5, 5.8} with {one, two} P100 GPUs. This is encouraging performance compared to the {4.4, 8.7} ideal speedups estimated from the peak FLOPS rates of the hardware.

The GPU-capable (T) code exhibited excellent strong scaling, with speedup of 13.5 on 16 dual-P100 nodes of NewRiver for a $(H_2O)_{20}$ cluster (cc-pVDZ basis set). Further improvement is possible by the use of improved tensor contraction kernels and optimizing the data movement to/from GPU. The latter was recently implemented and the updated performance will be evaluated elsewhere.

6. OUTLOOK

Over the past 40 years, the many-body quantum chemistry methods have evolved into a viable alternative to more approximate methods like DFT for large molecules and materials. Critical to these developments were the many robust numerical approximations that lead to (near-optimally) reduced storage and computational complexities. Before these

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

advances occurred, the conventional variants of many-body methods, with high-order (sixth and higher) polynomial computational complexity, were only applicable to small systems; the deployment of conventional methods onto parallel computers was critical for their application to larger systems. The exponential rate of hardware advancement (Moore's law) was instrumental in making many of the reduced-scaling developments viable, e.g., by helping reach the crossover point from the naïve to reduced complexity with practical resources.

The emergence of the reduced-scaling formalisms does not mean that parallel implementation of many-body methods is no longer important. It is important to reduce the time to solution for shortening the feedback loop in computation exploration and design, for enabling the simulation of first-principles dynamics, and even for increasing the power efficiency by leveraging increased aggregate storage to reduce the amount of recomputation. For some methods, the reduced-scaling formalisms may not be viable, and thus, the conventional full-scaling approaches are preferred. Lastly, the full-scaling approaches are needed as the benchmark for assessing their reduced-scaling counterparts.

The biggest challenge going forward is the continuing slowdown of the exponential evolution of the classical hardware. It is causing a disruptive increase in the complexity of typical and high-end computational platforms. Massive increase in the number of independent compute units in a "single" computer, deep hierarchical and/or disjoint storage spaces, emergence of accelerators and other types of specialized hardware, focus on power efficiency and variable precision, all make the deployment of compute-intensive methods more difficult. How do we tackle these challenges while dealing with ever more technically complex formalisms? Increasing the level of composition and abstraction by domain-specific abstractions/languages and code generation can help isolate our science from the hardware. Another part of the answer is to design our methods for the hardware. Examples of such hardware-oriented algorithm design can be seen throughout this review, e.g., trading off computation to reduce communication via integral-direct algorithms, stochastic evaluation, and other forms of recomputation. Similar opportunities can be found in favoring methods suitable for implementation in lower precision. Lastly, perhaps the emergence of practical programmable hardware will allow for exciting new opportunities to codesign hardware for software.

AUTHOR INFORMATION

Corresponding Author

Edward F. Valeev − *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States;* orcid.org/0000-0001-9923-6256; Email: efv@vt.edu

Authors

Justus A. Calvin − *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States*

Chong Peng − *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States*

Varun Rishi − *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States;* orcid.org/0000-0003-0317-3815

Ashutosh Kumar − *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.chemrev.0c00006

Notes

The authors declare no competing financial interest.

Biographies

Mr. Justus Calvin enlisted into the U.S. Army in July 1998, where he worked as an electronic technician operating and maintaining automatic test equipment. He received an honorable discharge from the military in 2002 after achieving the rank of Specialist. Afterward, he continued his education at Germanna Community College in Locust Grove, VA, where he received an associate degree in Arts and Sciences in 2004. He went on to complete a Bachelor of Science degree in Chemical Engineering in 2009 with minors in Chemistry and Physics at Virginia Tech. He is currently a Ph.D. candidate in the Department of Chemistry at Virginia Tech under the supervision of Dr. Edward F. Valeev. His work has focused on the development of a high-performance, massively parallel tensor computational framework and applying this framework to quantum many-body theory. He has also worked at Apple, Inc. since 2016 as a senior software engineer on global data distribution and search platforms.

Dr. Chong Peng received his B.S. in 2012 from Southwest University in Chongqing, China. He obtained his Ph.D. in Chemistry (2012−2018) under the supervision of Dr. Edward F. Valeev at Virginia Tech in Blacksburg, United States. His work focused on developing high-performance software for coupled cluster methods and reduced-scaling approaches for excited-state coupled cluster theory. After that, he continued his research at the same research group as a postdoc, working on a parallel tensor contraction library on heterogeneous platforms.

Dr. Varun Rishi received integrated B.S./M.S. from the Indian Institute of Science Education and Research, Pune in 2011. His doctoral work, under the guidance of Prof. Rod Bartlett at University of Florida, explored the solutions to the intricate problem of strong correlation in the realm of coupled cluster (CC) theory and methods for the accurate study of excited and ionized states and was completed in 2017. As a postdoc in Prof. Ed Valeev's group at Virginia Tech until October 2019, he undertook a massively parallel implementation of higher-level CC methods and applied tensor decomposition techniques to develop a reduced scaling version of these methods. Currently a postdoctoral researcher in Prof. Tom Miller's group at Caltech, he seeks to develop machine learning approaches for accurate yet less expensive prediction of response properties of molecular systems.

Dr. Ashutosh Kumar received his integrated Masters in Chemistry (B.Sc./M.Sc.) in 2013 from the Indian Institute of Technology, Kharagpur, located in West Bengal, India. His Ph.D. work (2013−2018) under the supervision of Prof. Daniel Crawford at Virginia Tech focused on the development of reduced-scaling formalisms for coupled cluster linear response theory. Currently, he is a postdoctoral associate at Virginia Tech, working on the development of efficient reduced-scaling coupled cluster explicit correlation methods for open-shell systems, exploration of explicitly correlated transformed Hamiltonians for cheaper yet accurate quantum simulations of electronic structure, and massively parallel implementation of coupled cluster response methods in the MPQC software package.

Prof. Edward F. Valeev received an M.Sc. in Chemistry in 1996 from the Higher Chemistry College of the Russian Academy of Sciences (Moscow, Russia) and a Ph.D. in Chemistry in 2000 from the University of Georgia (Athens, GA). He held a research scientist post

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

at the School of Chemistry and Biochemistry at Georgia Tech with a joint research appointment at the Oak Ridge National Laboratory during 2004−2006. Since 2006, he has been at Virginia Tech. Prof. Valeev's interests focus on accurate (many-body) methodology for electronic and molecular structure and high-productivity and high-performance scientific computing.

ACKNOWLEDGMENTS

This work was supported by the U.S. National Science Foundation (awards 1550456, 1800348, and 1931347) and 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. This work used resources of the OLCF at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract DE-AC05-00OR22725. This research used resources of the ALCF, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We also acknowledge Advanced Research Computing at Virginia Tech (www.arc.vt.edu) for providing computational resources and technical support that have contributed to the results reported within this paper.

REFERENCES

(1) Kohn, W.; Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **1965**, *140*, A1133−A1138.
(2) Engel, E.; Dreizler, R. M. *Density Functional Theory*; Theoretical and Mathematical Physics; Springer: Berlin, Heidelberg, 2011.
(3) Coester, F. Bound states of a many-particle system. *Nucl. Phys.* **1958**, *7*, 421−424.
(4) Coester, F.; Kümmel, H. Short-range correlations in nuclear wave functions. *Nucl. Phys.* **1960**, *17*, 477−485.
(5) Shavitt, I.; Bartlett, R. J. *Many-body methods in chemistry and physics: MBPT and coupled-cluster theory*; Cambridge University Press, 2009.
(6) Perera, A.; Bartlett, R. J.; Sanders, B. A.; Lotrich, V. F.; Byrd, J. N. Advanced concepts in electronic structure (ACES) software programs. *J. Chem. Phys.* **2020**, *152*, 184105.
(7) Matthews, D. A.; Cheng, L.; Harding, M. E.; Lipparini, F.; Stopkowicz, S.; Jagau, T.-C.; Szalay, P. G.; Gauss, J.; Stanton, J. F. Coupled-cluster techniques for computational chemistry: The CFOUR program package. *J. Chem. Phys.* **2020**, *152*, 214108.
(8) Werner, H.-J.; et al. The Molpro quantum chemistry package. *J. Chem. Phys.* **2020**, *152*, 144107.
(9) Neese, F.; Wennmohs, F.; Becker, U.; Riplinger, C. The ORCA quantum chemistry program package. *J. Chem. Phys.* **2020**, *152*, 224108.
(10) Shao, Y.; et al. Advances in molecular quantum chemistry contained in the Q-Chem 4 program package. *Mol. Phys.* **2015**, *113*, 184−215.
(11) Balasubramani, S. G.; et al. TURBOMOLE: Modular program suite for ab initio quantum-chemical and condensed-matter simulations. *J. Chem. Phys.* **2020**, *152*, 184107.
(12) Frisch, M. J., et al. *Gaussian16*, Revision C.01; Gaussian Inc.: Wallingford, CT, 2016.
(13) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A., Jr General atomic and molecular electronic structure system. *J. Comput. Chem.* **1993**, *14*, 1347−1363.
(14) Aprà, E.; et al. NWChem: Past, present, and future. *J. Chem. Phys.* **2020**, *152*, 184102.
(15) Purvis, G. D.; Bartlett, R. J. A full coupled-cluster singles and doubles model: The inclusion of disconnected triples. *J. Chem. Phys.* **1982**, *76*, 1910−1918.
(16) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. A fifth-order perturbation comparison of electron correlation theories. *Chem. Phys. Lett.* **1989**, *157*, 479−483.
(17) Kutzelnigg, W.; Morgan, J. D. Rates of convergence of the partial-wave expansions of atomic correlation energies. *J. Chem. Phys.* **1992**, *96*, 4484.
(18) Helgaker, T.; Klopper, W.; Koch, H.; Noga, J. Basis-set convergence of correlated calculations on water. *J. Chem. Phys.* **1997**, *106*, 9639−9646.
(19) Harris, F. E.; Jeziorski, B.; Monkhorst, H. J. Contraction theorem for the algebraic reduction of (anti)commutators involving operator strings. *Phys. Rev. A: At., Mol., Opt. Phys.* **1981**, *23*, 1632−1638.
(20) Kutzelnigg, W. Quantum chemistry in Fock space. I. The universal wave and energy operators. *J. Chem. Phys.* **1982**, *77*, 3081−3097.
(21) Bartlett, R. J.; Musiał, M. Coupled-cluster theory in quantum chemistry. *Rev. Mod. Phys.* **2007**, *79*, 291−352.
(22) Crawford, T. D.; Schaefer, H. F. *Rev. Comput. Chem.*; John Wiley & Sons, Inc.: Hoboken, NJ, 2000; Vol. *14*; pp 33−136.
(23) Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular electronic-structure theory*, 1st ed.; John Wiley & Sons, Ltd: Chichester, UK, 2000.
(24) Čížek, J. On the correlation problem in atomic and molecular systems. Calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods. *J. Chem. Phys.* **1966**, *45*, 4256−4266.
(25) Paldus, J. Correlation problems in atomic and molecular systems. V. Spin-adapted coupled cluster many-electron theory. *J. Chem. Phys.* **1977**, *67*, 303−318.
(26) Pulay, P.; Saebø, S.; Meyer, W. An efficient reformulation of the closed-shell self-consistent electron pair theory. *J. Chem. Phys.* **1984**, *81*, 1901−1905.
(27) Scuseria, G. E.; Janssen, C. L.; Schaefer, H. F. An efficient reformulation of the closed-shell coupled cluster single and double excitation (CCSD) equations. *J. Chem. Phys.* **1988**, *89*, 7382−7387.
(28) Li, X.; Paldus, J. In *Recent advances in coupled-cluster methods: Vol. 3*; Bartlett, R. J., Ed.; World Scientific, pp 183−219.
(29) Janssen, C. L.; Schaefer, H. F. The automated solution of second quantization equations with applications to the coupled cluster approach. *Theoretica Chimica Acta* **1991**, *79*, 1−42.
(30) Jeziorski, B.; Monkhorst, H. J. Coupled-cluster method for multideterminantal reference states. *Phys. Rev. A: At., Mol., Opt. Phys.* **1981**, *24*, 1668−1681.
(31) Mahapatra, U. S.; Datta, B.; Bandyopadhyay, B.; Mukherjee, D. In *State-specific multi-reference coupled cluster formulations: Two paradigms*; Lwdin, P.-O., Ed.; Advances in Quantum Chemistry; Academic Press, 1998; Vol. *30*; pp 163−193.
(32) Mášik, J.; Hubac, I. In *Multireference Brillouin-Wigner coupled-cluster theory. Single-root approach*; Sabin, J. R., Zerner, M. C., Brändas, E.; Wilson, S., Maruani, J., Smeyers, Y., Grout, P., McWeeny, R., Eds.; Advances in Quantum Chemistry; Academic Press, 1998; Vol. 31; pp 75−104.
(33) Haque, M. A.; Mukherjee, D. Application of cluster expansion techniques to open shells: Calculation of difference energies. *J. Chem. Phys.* **1984**, *80*, 5058−5069.
(34) Piecuch, P.; Paldus, J. Orthogonally spin-adapted multi-reference Hilbert space coupled-cluster formalism: diagrammatic formulation. *Theoretica Chimica Acta* **1992**, *83*, 69−103.
(35) Hanrath, M. An exponential multireference wave-function Ansatz. *J. Chem. Phys.* **2005**, *123*, No. 084102.
(36) Evangelista, F. A.; Gauss, J. An orbital-invariant internally contracted multireference coupled cluster approach. *J. Chem. Phys.* **2011**, *134*, 114102.
(37) Hanauer, M.; Köhn, A. Pilot applications of internally contracted multireference coupled cluster theory, and how to choose the cluster operator properly. *J. Chem. Phys.* **2011**, *134*, 204111.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

(38) Evangelista, F. A. Perspective: Multireference coupled cluster theories of dynamical electron correlation. *J. Chem. Phys.* **2018**, *149*, No. 030901.
(39) Laidig, W. D.; Bartlett, R. J. A multi-reference coupled-cluster method for molecular applications. *Chem. Phys. Lett.* **1984**, *104*, 424−430.
(40) Gdanitz, R. J.; Ahlrichs, R. The averaged coupled-pair functional (ACPF): A size-extensive modification of MR CI(SD). *Chem. Phys. Lett.* **1988**, *143*, 413−420.
(41) Andersson, K.; Malmqvist, P. Å.; Roos, B. O. Second-order perturbation theory with a complete active space self-consistent field reference function. *J. Chem. Phys.* **1992**, *96*, 1218−1226.
(42) Angeli, C.; Cimiraglia, R.; Evangelisti, S.; Leininger, T.; Malrieu, J. P. Introduction of n-electron valence states for multi-reference perturbation theory. *J. Chem. Phys.* **2001**, *114*, 10252−10264.
(43) MacLeod, M. K.; Shiozaki, T. Communication: Automatic code generation enables nuclear gradient computations for fully internally contracted multireference theory. *J. Chem. Phys.* **2015**, *142*, No. 051103.
(44) Werner, H. J.; Knowles, P. J. An efficient internally contracted multiconfiguration−reference configuration interaction method. *J. Chem. Phys.* **1988**, *89*, 5803−5814.
(45) Olsen, J.; Jørgensen, P.; Koch, H.; Balkova, A.; Bartlett, R. J. Full configuration−interaction and state of the art correlation calculations on water in a valence double-zeta basis with polarization functions. *J. Chem. Phys.* **1996**, *104*, 8007−8015.
(46) Hirata, S.; Bartlett, R. J. High-order coupled-cluster calculations through connected octuple excitations. *Chem. Phys. Lett.* **2000**, *321*, 216−224.
(47) Noga, J.; Bartlett, R. J. The full CCSDT model for molecular electronic structure. *J. Chem. Phys.* **1987**, *86*, 7041−7050.
(48) Kucharski, S. A.; Bartlett, R. J. The coupled-cluster single, double, triple, and quadruple excitation method. *J. Chem. Phys.* **1992**, *97*, 4282−4288.
(49) Monkhorst, H. J. Calculation of properties with the coupled-cluster method. *Int. J. Quantum Chem.* **1977**, *12*, 421−432.
(50) Sekino, H.; Bartlett, R. J. A linear response, coupled-cluster theory for excitation energy. *Int. J. Quantum Chem.* **1984**, *26*, 255−265.
(51) Koch, H.; Jørgensen, P. Coupled cluster response functions. *J. Chem. Phys.* **1990**, *93*, 3333−3344.
(52) Stanton, J. F.; Bartlett, R. J. The equation of motion coupled-cluster method. A systematic biorthogonal approach to molecular excitation energies, transition probabilities, and excited state properties. *J. Chem. Phys.* **1993**, *98*, 7029.
(53) Nakatsuji, H. Cluster expansion of the wavefunction. Electron correlations in ground and excited states by SAC (symmetry-adapted-cluster) and SAC CI theories. *Chem. Phys. Lett.* **1979**, *67*, 329−333.
(54) Geertsen, J.; Oddershede, J. A coupled cluster polarization propagator method applied to CH+. *J. Chem. Phys.* **1986**, *85*, 2112−2118.
(55) Krylov, A. I. Equation-of-motion coupled-cluster methods for open-shell and electronically excited species: The hitchhiker's guide to Fock space. *Annu. Rev. Phys. Chem.* **2008**, *59*, 433−462.
(56) Sneskov, K.; Christiansen, O. Excited state coupled cluster methods. *WIREs Comput. Mol. Sci.* **2012**, *2*, 566−584.
(57) Kucharski, S. A.; Bartlett, R. J. Noniterative energy corrections through fifth-order to the coupled cluster singles and doubles method. *J. Chem. Phys.* **1998**, *108*, 5243−5254.
(58) Hirata, S.; Nooijen, M.; Grabowski, I.; Bartlett, R. J. Perturbative corrections to coupled-cluster and equation-of-motion coupled-cluster energies: A determinantal analysis. *J. Chem. Phys.* **2001**, *114*, 3919−3928.
(59) Piecuch, P.; Włoch, M. Renormalized coupled-cluster methods exploiting left eigenstates of the similarity-transformed Hamiltonian. *J. Chem. Phys.* **2005**, *123*, 224105.
(60) Bomble, Y. J.; Stanton, J. F.; Kállay, M.; Gauss, J. Coupled-cluster methods including noniterative corrections for quadruple excitations. *J. Chem. Phys.* **2005**, *123*, No. 054101.
(61) Lee, Y. S.; Kucharski, S. A.; Bartlett, R. J. A coupled cluster approach with triple excitations. *J. Chem. Phys.* **1984**, *81*, 5906−5912.
(62) Urban, M.; Noga, J.; Cole, S. J.; Bartlett, R. J. Towards a full CCSDT model for electron correlation. *J. Chem. Phys.* **1985**, *83*, 4041−4046.
(63) Koch, H.; Christiansen, O.; Jørgensen, P.; Sanchez de Merás, A. M.; Helgaker, T. The CC3 model: An iterative coupled cluster approach including connected triples. *J. Chem. Phys.* **1997**, *106*, 1808−1818.
(64) Watts, J. D.; Bartlett, R. J. Economical triple excitation equation-of-motion coupled-cluster methods for excitation energies. *Chem. Phys. Lett.* **1995**, *233*, 81−87.
(65) Watts, J. D.; Bartlett, R. J. Iterative and non-iterative triple excitation corrections in coupled-cluster methods for excited electronic states: the EOM-CCSDT-3 and EOM-CCSD($\tilde{T}$) methods. *Chem. Phys. Lett.* **1996**, *258*, 581−588.
(66) Christiansen, O.; Koch, H.; Jørgensen, P. Response functions in the CC3 iterative triple excitation model. *J. Chem. Phys.* **1995**, *103*, 7429−7441.
(67) Christiansen, O.; Koch, H.; Jørgensen, P. Perturbative triple excitation corrections to coupled cluster singles and doubles excitation energies. *J. Chem. Phys.* **1996**, *105*, 1451−1459.
(68) Matthews, D. A.; Stanton, J. F. A new approach to approximate equation-of-motion coupled cluster with triple excitations. *J. Chem. Phys.* **2016**, *145*, 124102.
(69) Kato, T. On the eigenfunctions of many-particle systems in quantum mechanics. *Communications on Pure and Applied Mathematics* **1957**, *10*, 151−177.
(70) Pack, R. T; Brown, W. B. Cusp conditions for molecular wavefunctions. *J. Chem. Phys.* **1966**, *45*, 556−559.
(71) Hylleraas, E. A. Neue Berechnung der Energie des Heliums im Grundzustande, sowie des tiefsten Terms von Ortho-Helium. *Eur. Phys. J. A* **1929**, *54*, 347−366.
(72) Kutzelnigg, W. r_{12}-Dependent terms in the wave function as closed sums of partial wave amplitudes for large l. *Theoretica Chimica Acta* **1985**, *68*, 445−469.
(73) Ten-no, S. Initiation of explicitly correlated Slater-type geminal theory. *Chem. Phys. Lett.* **2004**, *398*, 56−61.
(74) Ten-no, S. Explicitly correlated second order perturbation theory: Introduction of a rational generator and numerical quadratures. *J. Chem. Phys.* **2004**, *121*, 117−129.
(75) Valeev, E. F. Improving on the resolution of the identity in linear R12 ab initio theories. *Chem. Phys. Lett.* **2004**, *395*, 190−195.
(76) Klopper, W.; Samson, C. C. Explicitly correlated second-order Møller−Plesset methods with auxiliary basis sets. *J. Chem. Phys.* **2002**, *116*, 6397−6410.
(77) Kong, L.; Bischoff, F. A.; Valeev, E. F. Explicitly correlated R12/F12 methods for electronic structure. *Chem. Rev.* **2012**, *112*, 75−107.
(78) Hättig, C.; Klopper, W.; Köhn, A.; Tew, D. P. Explicitly correlated electrons in molecules. *Chem. Rev.* **2012**, *112*, 4−74.
(79) Ten-no, S.; Noga, J. Explicitly correlated electronic structure theory from R12/F12 ansätze. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2012**, *2*, 114−125.
(80) Noga, J.; Kutzelnigg, W.; Klopper, W. CC-R12, a correlation cusp corrected coupled-cluster method with a pilot application to the Be2 potential curve. *Chem. Phys. Lett.* **1992**, *199*, 497−504.
(81) Noga, J.; Kutzelnigg, W. Coupled cluster theory that takes care of the correlation cusp by inclusion of linear terms in the interelectronic coordinates. *J. Chem. Phys.* **1994**, *101*, 7738−7762.
(82) Shiozaki, T.; Valeev, E. F.; Hirata, S. Explicitly correlated combined coupled-cluster and perturbation methods. *J. Chem. Phys.* **2009**, *131*, No. 044118.
(83) Shiozaki, T.; Kamiya, M.; Hirata, S.; Valeev, E. F. Higher-order explicitly correlated coupled-cluster methods. *J. Chem. Phys.* **2009**, *130*, No. 054101.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

(84) Shiozaki, T.; Kamiya, M.; Hirata, S.; Valeev, E. F. Explicitly correlated coupled-cluster singles and doubles method based on complete diagrammatic equations. *J. Chem. Phys.* **2008**, *129*, No. 071101.
(85) Köhn, A.; Richings, G. W.; Tew, D. P. Implementation of the full explicitly correlated coupled-cluster singles and doubles model CCSD-F12 with optimally reduced auxiliary basis dependence. *J. Chem. Phys.* **2008**, *129*, 201103.
(86) Fliegl, H.; Klopper, W.; Hättig, C. Coupled-cluster theory with simplified linear-r12 corrections: The CCSD (R12) model. *J. Chem. Phys.* **2005**, *122*, No. 084107.
(87) Adler, T. B.; Knizia, G.; Werner, H.-J. A simple and efficient CCSD(T)-F12 approximation. *J. Chem. Phys.* **2007**, *127*, 221106.
(88) Hättig, C.; Tew, D. P.; Köhn, A. Communications: Accurate and efficient approximations to explicitly correlated coupled-cluster singles and doubles, CCSD-F12. *J. Chem. Phys.* **2010**, *132*, 231102.
(89) Valeev, E. F. Coupled-cluster methods with perturbative inclusion of explicitly correlated terms: a preliminary investigation. *Phys. Chem. Chem. Phys.* **2008**, *10*, 106−113.
(90) Tew, D. P.; Klopper, W.; Neiss, C.; Hättig, C. Quintuple-zeta quality coupled-cluster correlation energies with triple-zeta basis sets. *Phys. Chem. Chem. Phys.* **2007**, *9*, 1921−1930.
(91) Zhang, J.; Valeev, E. F. Prediction of reaction barriers and thermochemical properties with explicitly correlated coupled-cluster methods: A basis set assessment. *J. Chem. Theory Comput.* **2012**, *8*, 3175−86.
(92) Kolda, T. G.; Bader, B. W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455−500.
(93) Harris, F. E.; Rein, R. Integral approximations for molecular orbital theory. *Theoretica Chimica Acta* **1966**, *6*, 73−82.
(94) Whitten, J. L. Coulombic potential energy integrals and approximations. *J. Chem. Phys.* **1973**, *58*, 4496−4501.
(95) Baerends, E. J.; Ellis, D. E.; Ros, P. Self-consistent molecular Hartree—Fock—Slater calculations I. The computational procedure. *Chem. Phys.* **1973**, *2*, 41−51.
(96) Feyereisen, M. W.; Fitzgerald, G.; Komornicki, A. Use of approximate integrals in ab initio theory. An application in MP2 energy calculations. *Chem. Phys. Lett.* **1993**, *208*, 359−363.
(97) Weigend, F.; Häser, M. RI-MP2: first derivatives and global consistency. *Theor. Chem. Acc.* **1997**, *97*, 331−340.
(98) Peng, C.; Calvin, J. A.; Pavošević, F.; Zhang, J.; Valeev, E. F. Massively parallel implementation of explicitly correlated coupled-cluster singles and doubles using tiledArray framework. *J. Phys. Chem. A* **2016**, *120*, 10231−10244.
(99) Manby, F. R. Density fitting in second-order linear-r_{12} Møller−Plesset perturbation theory. *J. Chem. Phys.* **2003**, *119*, 4607−4613.
(100) Werner, H.-J.; Manby, F. R.; Knowles, P. J. Fast linear scaling second-order Møller-Plesset perturbation theory (MP2) using local and density fitting approximations. *J. Chem. Phys.* **2003**, *118*, 8149−8160.
(101) Schütz, M.; Manby, F. R. Linear scaling local coupled cluster theory with density fitting. Part I: 4-external integrals. *Phys. Chem. Chem. Phys.* **2003**, *5*, 3349−3358.
(102) Neese, F.; Wennmohs, F.; Hansen, A. Efficient and accurate local approximations to coupled-electron pair approaches: An attempt to revive the pair natural orbital method. *J. Chem. Phys.* **2009**, *130*, 114108.
(103) Neese, F.; Hansen, A.; Liakos, D. G. Efficient and accurate approximations to the local coupled cluster singles doubles method using a truncated pair natural orbital basis. *J. Chem. Phys.* **2009**, *131*, No. 064103.
(104) Pinski, P.; Riplinger, C.; Valeev, E. F.; Neese, F. Sparse maps—A systematic infrastructure for reduced-scaling electronic structure methods. I. An efficient and simple linear scaling local MP2 method that uses an intermediate basis of pair natural orbitals. *J. Chem. Phys.* **2015**, *143*, No. 034108.
(105) Riplinger, C.; Pinski, P.; Becker, U.; Valeev, E. F.; Neese, F. Sparse maps—A systematic infrastructure for reduced-scaling electronic structure methods. II. Linear scaling domain based pair natural orbital coupled cluster theory. *J. Chem. Phys.* **2016**, *144*, No. 024109.
(106) Werner, H.-J.; Knizia, G.; Krause, C.; Schwilk, M.; Dornbach, M. Scalable electron correlation methods I.: PNO-LMP2 with linear scaling in the molecular size and near-inverse-linear scaling in the number of processors. *J. Chem. Theory Comput.* **2015**, *11*, 484−507.
(107) Eichkorn, K.; Treutler, O.; Öhm, H.; Haser, M.; Ahlrichs, R. Auxiliary basis sets to approximate Coulomb potentials. *Chem. Phys. Lett.* **1995**, *240*, 283−290.
(108) Weigend, F.; Köhn, A.; Hättig, C. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *J. Chem. Phys.* **2002**, *116*, 3175−3183.
(109) Beebe, N. H. F.; Linderberg, J. Simplifications in the generation and transformation of two-electron integrals in molecular calculations. *Int. J. Quantum Chem.* **1977**, *12*, 683−705.
(110) Koch, H.; Sánchez de Merás, A.; Pedersen, T. B. Reduced scaling in electronic structure calculations using Cholesky decompositions. *J. Chem. Phys.* **2003**, *118*, 9481−9484.
(111) Folkestad, S. D.; Kjønstad, E. F.; Koch, H. An efficient algorithm for Cholesky decomposition of electron repulsion integrals. *J. Chem. Phys.* **2019**, *150*, 194112.
(112) Dunlap, B. I. Robust and variational fitting. *Phys. Chem. Chem. Phys.* **2000**, *2*, 2113−2116.
(113) Barr, T. L.; Davidson, E. R. Nature of the configuration-interaction method in ab initio calculations. I. Ne ground state. *Phys. Rev. A: At., Mol., Opt. Phys.* **1970**, *1*, 644−658.
(114) Taube, A. G.; Bartlett, R. J. Frozen natural orbitals: Systematic basis set truncation for coupled-cluster theory. *Collect. Czech. Chem. Commun.* **2005**, *70*, 837−850.
(115) Landau, A.; Khistyaev, K.; Dolgikh, S.; Krylov, A. I. Frozen natural orbitals for ionized states within equation-of-motion coupled-cluster formalism. *J. Chem. Phys.* **2010**, *132*, No. 014109.
(116) Kumar, A.; Crawford, T. D. Frozen virtual natural orbitals for coupled-cluster linear-response theory. *J. Phys. Chem. A* **2017**, *121* (3), 708−716.
(117) Yasuda, K. Two-electron integral evaluation on the graphics processor unit. *J. Comput. Chem.* **2008**, *29*, 334−342.
(118) Asadchev, A.; Gordon, M. S. Mixed-precision evaluation of two-electron integrals by Rys quadrature. *Comput. Phys. Commun.* **2012**, *183*, 1563−1567.
(119) Vysotskiy, V. P.; Cederbaum, L. S. Accurate quantum chemistry in single precision arithmetic: Correlation energy. *J. Chem. Theory Comput.* **2011**, *7*, 320.
(120) Knizia, G.; Li, W.; Simon, S.; Werner, H.-J. Determining the numerical stability of quantum chemistry algorithms. *J. Chem. Theory Comput.* **2011**, *7*, 2387−2398.
(121) DePrince, A. E.; Hammond, J. R. Coupled cluster theory on graphics processing units I. The coupled cluster doubles method. *J. Chem. Theory Comput.* **2011**, *7*, 1287−1295.
(122) Pokhilko, P.; Epifanovsky, E.; Krylov, A. I. Double precision is not needed for many-body calculations: Emergent conventional wisdom. *J. Chem. Theory Comput.* **2018**, *14*, 4088−4096.
(123) Beylkin, G.; Mohlenkamp, M. J. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. U. S. A.* **2002**, *99*, 10246−10251.
(124) Benedikt, U.; Auer, A. A.; Espig, M.; Hackbusch, W. Tensor decomposition in post-Hartree−Fock methods. I. Two-electron integrals and MP2. *J. Chem. Phys.* **2011**, *134*, No. 054118.
(125) Peng, B.; Kowalski, K. Highly efficient and scalable compound decomposition of two-electron integral tensor and its application in coupled cluster calculations. *J. Chem. Theory Comput.* **2017**, *13*, 4179−4192.
(126) Lewis, C. A.; Calvin, J. A.; Valeev, E. F. Clustered Low-Rank Tensor Format: Introduction and Application to Fast Construction of Hartree−Fock Exchange. *J. Chem. Theory Comput.* **2016**, *12*, 5868−5880.
(127) Parrish, R. M.; Sherrill, C. D.; Hohenstein, E. G.; Kokkila, S. I. L.; Martínez, T. J. Communication: Acceleration of coupled cluster

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

singles and doubles via orbital-weighted least-squares tensor hypercontraction. *J. Chem. Phys.* **2014**, *140*, 181102.
(128) Hohenstein, E. G.; Parrish, R. M.; Sherrill, C. D.; Martínez, T. J. Communication: Tensor hypercontraction. III. Least-squares tensor hypercontraction for the determination of correlated wavefunctions. *J. Chem. Phys.* **2012**, *137*, 221101.
(129) Hohenstein, E. G.; Parrish, R. M.; Martínez, T. J. Tensor hypercontraction density fitting. I. Quartic scaling second- and third-order Møller-Plesset perturbation theory. *J. Chem. Phys.* **2012**, *137*, No. 044103.
(130) Friesner, R. A. Solution of self-consistent field electronic structure equations by a pseudospectral method. *Chem. Phys. Lett.* **1985**, *116*, 39−43.
(131) Cooley, J. W.; Tukey, J. W. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.* **1965**, *19*, 297−297.
(132) Greengard, L.; Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.* **1987**, *73*, 325−348.
(133) White, C. A.; Johnson, B. G.; Gill, P. M. W.; Head-Gordon, M. The continuous fast multipole method. *Chem. Phys. Lett.* **1994**, *230*, 8−16.
(134) Saebø, S.; Pulay, P. Local configuration interaction: An efficient approach for larger molecules. *Chem. Phys. Lett.* **1985**, *113*, 13−18.
(135) Pulay, P. Localizability of dynamic electron correlation. *Chem. Phys. Lett.* **1983**, *100*, 151−154.
(136) Hampel, C.; Werner, H.-J. Local treatment of electron correlation in coupled cluster theory. *J. Chem. Phys.* **1996**, *104*, 6286−6297.
(137) Schütz, M.; Hetzer, G.; Werner, H.-J. Linear scaling local electron correlation methods. I. Linear scaling local MP2. *J. Chem. Phys.* **1999**, *111*, 5691−5705.
(138) Hetzer, G.; Schütz, M.; Stoll, H.; Werner, H.-J. Low-order scaling local correlation methods. II: Splitting the Coulomb operator in linear scaling local second-order Møller-Plesset perturbation theory. *J. Chem. Phys.* **2000**, *113*, 9443−9455.
(139) Schütz, M. Low-order scaling local electron correlation methods. III. Linear scaling local perturbative triples correction (T). *J. Chem. Phys.* **2000**, *113*, 9986−10001.
(140) Schütz, M.; Werner, H.-J. Low-order scaling local correlation methods. IV. Linear scaling local coupled-cluster (LCCSD). *J. Chem. Phys.* **2001**, *114*, 661−681.
(141) Schütz, M. Low-order scaling local electron correlation methods. V. Connected triples beyond (T): Linear scaling local CCSDT-1b. *J. Chem. Phys.* **2002**, *116*, 8772−8785.
(142) Almlöf, J. Elimination of energy denominators in Møller-Plesset perturbation theory by a Laplace transform approach. *Chem. Phys. Lett.* **1991**, *181*, 319−320.
(143) Häser, M.; Almlöf, J. Laplace transform techniques in Møller-Plesset perturbation theory. *J. Chem. Phys.* **1992**, *96*, 489−494.
(144) Ayala, P. Y.; Scuseria, G. E. Linear scaling second-order Møller-Plesset theory in the atomic orbital basis for large molecular systems. *J. Chem. Phys.* **1999**, *110*, 3660−3671.
(145) Doser, B.; Lambrecht, D. S.; Ochsenfeld, C. Tighter multipole-based integral estimates and parallel implementation of linear-scaling AO−MP2 theory. *Phys. Chem. Chem. Phys.* **2008**, *10*, 3335−3344.
(146) Maslen, P. E.; Head-Gordon, M. Non-iterative local second order Møller−Plesset theory. *Chem. Phys. Lett.* **1998**, *283*, 102−108.
(147) Wilson, A. K.; Almlöf, J. Møller-Plesset correlation energies in a localized orbital basis using a Laplace transform technique. *Theor. Chim. Acta* **1997**, *95*, 49−62.
(148) Edmiston, C.; Krauss, M. Configuration-interaction calculation of H3 and H2. *J. Chem. Phys.* **1965**, *42*, 1119−1120.
(149) Edmiston, C.; Krauss, M. Pseudonatural orbitals as a basis for the superposition of configurations. I. He2+. *J. Chem. Phys.* **1966**, *45*, 1833−1839.
(150) Meyer, W. Ionization energies of water from PNO-CI calculations. *Int. J. Quantum Chem.* **1971**, *5*, 341−348.
(151) Meyer, W. PNO-CI Studies of electron correlation effects. I. Configuration expansion by means of nonorthogonal orbitals, and application to the ground state and ionized states of methane. *J. Chem. Phys.* **1973**, *58*, 1017−1035.
(152) Meyer, W. PNO-CI and CEPA studies of electron correlation effects. *Theoretica Chimica Acta* **1974**, *35*, 277−292.
(153) Meyer, W.; Rosmus, P. PNO−CI and CEPA studies of electron correlation effects. III. Spectroscopic constants and dipole moment functions for the ground states of the first-row and second-row diatomic hydrides. *J. Chem. Phys.* **1975**, *63*, 2356−2375.
(154) Werner, H.-J.; Meyer, W. PNO-CI and PNO-CEPA studies of electron correlation effects. *Mol. Phys.* **1976**, *31*, 855−872.
(155) Yang, J.; Chan, G. K.-L.; Manby, F. R.; Schütz, M.; Werner, H.-J. The orbital-specific-virtual local coupled cluster singles and doubles method. *J. Chem. Phys.* **2012**, *136*, 144105.
(156) Eriksen, J. J.; Baudin, P.; Ettenhuber, P.; Kristensen, k.; Kjærgaard, T.; Jørgensen, P. Linear-scaling coupled cluster with perturbative triple excitations: The divideexpandconsolidate CCSD-(T) model. *J. Chem. Theory Comput.* **2015**, *11*, 2984−2993.
(157) Høyvik, I.-M.; Jansik, B.; Jørgensen, P. Orbital localization using fourth central moment minimization. *J. Chem. Phys.* **2012**, *137*, 224114.
(158) Bischoff, F. A.; Harrison, R. J.; Valeev, E. F. Computing many-body wave functions with guaranteed precision: the first-order Møller-Plesset wave function for the ground state of helium atom. *J. Chem. Phys.* **2012**, *137*, 104103.
(159) Bischoff, F. A.; Valeev, E. F. Computing molecular correlation energies with guaranteed precision. *J. Chem. Phys.* **2013**, *139*, 114106.
(160) Schäfer, T.; Ramberger, B.; Kresse, G. Quartic scaling MP2 for solids: A highly parallelized algorithm in the plane wave basis. *J. Chem. Phys.* **2017**, *146*, 104101.
(161) Kottmann, J. S.; Bischoff, F. A. Coupled-cluster in real space. 1. CC2 ground state energies using multiresolution analysis. *J. Chem. Theory Comput.* **2017**, *13*, 5945−5955.
(162) Mardirossian, N.; McClain, J. D.; Chan, G. K.-L. Lowering of the complexity of quantum chemistry methods by choice of representation. *J. Chem. Phys.* **2018**, *148*, No. 044106.
(163) 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.
(164) Day, P. N.; Jensen, J. H.; Gordon, M. S.; Webb, S. P.; Stevens, W. J.; Krauss, M.; Garmer, D.; Basch, H.; Cohen, D. An effective fragment method for modeling solvent effects in quantum mechanical calculations. *J. Chem. Phys.* **1996**, *105*, 1968−1986.
(165) Svensson, M.; Humbel, S.; Froese, R. D. J.; Matsubara, T.; Sieber, S.; Morokuma, K. ONIOM: A multilayered integrated MO + MM method for geometry optimizations and single point energy predictions. A test for Diels-Alder reactions and Pt(P(t-Bu)3)2 + H2 oxidative addition. *J. Phys. Chem.* **1996**, *100*, 19357−19363.
(166) Humbel, S.; Sieber, S.; Morokuma, K. The IMOMO method: Integration of different levels of molecular orbital approximations for geometry optimization of large systems: Test for n-butane conformation and SN2 reaction: RCl+Cl-. *J. Chem. Phys.* **1996**, *105*, 1959−1967.
(167) Gao, J. Toward a molecular orbital derived empirical potential for liquid simulations. *J. Phys. Chem. B* **1997**, *101*, 657−663.
(168) Huang, L.; Massa, L.; Karle, J. Kernel energy method illustrated with peptides. *Int. J. Quantum Chem.* **2005**, *103*, 808−817.
(169) Netzloff, H. M.; Collins, M. A. Ab initio energies of nonconducting crystals by systematic fragmentation. *J. Chem. Phys.* **2007**, *127*, 134113.
(170) Beran, G. J. O. Approximating quantum many-body intermolecular interactions in molecular clusters using classical polarizable force fields. *J. Chem. Phys.* **2009**, *130*, 164115.
(171) Lao, K. U.; Herbert, J. M. Accurate and efficient quantum chemistry calculations for noncovalent interactions in many-body systems: The XSAPT family of methods. *J. Phys. Chem. A* **2015**, *119*, 235−252.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

(172) Yang, W. Direct calculation of electron density in density-functional theory. *Phys. Rev. Lett.* **1991**, *66*, 1438−1441.
(173) Friedrich, J.; Hanrath, M.; Dolg, M. Fully automated implementation of the incremental scheme: Application to CCSD energies for hydrocarbons and transition metal compounds. *J. Chem. Phys.* **2007**, *126*, 154110.
(174) Li, W.; Piecuch, P.; Gour, J. R.; Li, S. Local correlation calculations using standard and renormalized coupled-cluster approaches. *J. Chem. Phys.* **2009**, *131*, 114109.
(175) Kristensen, K.; Ziółkowski, M.; Jansík, B.; Kjærgaard, T.; Jørgensen, P. A locality analysis of the divid−expand−consolidate coupled cluster amplitude equations. *J. Chem. Theory Comput.* **2011**, *7*, 1677−1694.
(176) Kjærgaard, T.; Baudin, P.; Bykov, D.; Kristensen, K.; Jørgensen, P. The divide-expand-consolidate coupled cluster scheme. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2017**, *9*, No. e1319.
(177) Wesolowski, T. A.; Warshel, A. Frozen density functional approach for ab initio calculations of solvated molecules. *J. Phys. Chem.* **1993**, *97*, 8050−8053.
(178) Jacob, C. R.; Neugebauer, J. Subsystem density-functional theory. *WIREs Comput. Mol. Sci.* **2014**, *4*, 325−362.
(179) Zgid, D.; Chan, G. K.-L. Dynamical mean-field theory from a quantum chemical perspective. *J. Chem. Phys.* **2011**, *134*, No. 094115.
(180) Chibani, W.; Ren, X.; Scheffler, M.; Rinke, P. Self-consistent Green's function embedding for advanced electronic structure methods based on a dynamical mean-field concept. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2016**, *93*, 165106.
(181) Georges, A.; Kotliar, G.; Krauth, W.; Rozenberg, M. J. Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions. *Rev. Mod. Phys.* **1996**, *68*, 13−125.
(182) Knizia, G.; Chan, G. K.-L. Density matrix embedding: A simple alternative to dynamical mean-field theory. *Phys. Rev. Lett.* **2012**, *109*, 186404.
(183) White, J. C.; Davidson, E. R. An analysis of the hydrogen bond in ice. *J. Chem. Phys.* **1990**, *93*, 8029−8035.
(184) Stoll, H. Correlation energy of diamond. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1992**, *46*, 6700−6704.
(185) Mayhall, N. J.; Raghavachari, K. Molecules-in-Molecules: An extrapolated fragment-based approach for accurate calculations on large molecules and materials. *J. Chem. Theory Comput.* **2011**, *7*, 1336−1343.
(186) Richard, R. M.; Herbert, J. M. A generalized many-body expansion and a unified view of fragment-based methods in electronic structure theory. *J. Chem. Phys.* **2012**, *137*, No. 064113.
(187) Rolik, Z.; Kallay, M. A general-order local coupled-cluster method based on the cluster-in-molecule approach. *J. Chem. Phys.* **2011**, *135*, 104111.
(188) Rolik, Z.; Szegedy, L.; Ladjánszki, I.; Ladóczki, B.; Kállay, M. An efficient linear-scaling CCSD(T) method based on local natural orbitals. *J. Chem. Phys.* **2013**, *139*, No. 094105.
(189) Nagy, P. R.; Kállay, M. Optimization of the linear-scaling local natural orbital CCSD(T) method: Redundancy-free triples correction using Laplace transform. *J. Chem. Phys.* **2017**, *146*, 214106.
(190) Nagy, P. R.; Samu, G.; Kállay, M. Optimization of the linear-scaling local natural orbital CCSD(T) method: Improved algorithm and benchmark applications. *J. Chem. Theory Comput.* **2018**, *14*, 4193−4215.
(191) Guo, Y.; Becker, U.; Neese, F. Comparison and combination of "direct" and fragment based local correlation methods: Cluster in molecules and domain based local pair natural orbital perturbation and coupled cluster theories. *J. Chem. Phys.* **2018**, *148*, 124117.
(192) Schmitz, G.; Hättig, C.; Tew, D. P. Explicitly correlated PNO-MP2 and PNO-CCSD and their application to the S66 set and large molecular systems. *Phys. Chem. Chem. Phys.* **2014**, *16*, 22167−22178.
(193) Schmitz, G.; Hättig, C. Perturbative triples correction for local pair natural orbital based explicitly correlated CCSD(F12*) using Laplace transformation techniques. *J. Chem. Phys.* **2016**, *145*, 234107.
(194) Riplinger, C.; Neese, F. An efficient and near linear scaling pair natural orbital based local coupled cluster method. *J. Chem. Phys.* **2013**, *138*, No. 034106.
(195) Riplinger, C.; Sandhoefer, B.; Hansen, A.; Neese, F. Natural triple excitations in local coupled cluster calculations with pair natural orbitals. *J. Chem. Phys.* **2013**, *139*, 134101.
(196) Pavošević, F.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E. F. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. IV. Linear-scaling second-order explicitly correlated energy with pair natural orbitals. *J. Chem. Phys.* **2016**, *144*, 144109.
(197) Pavošević, F.; Peng, C.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E. F. SparseMaps—A systematic infrastructure for reduced scaling electronic structure methods. V. Linear scaling explicitly correlated coupled-cluster method with pair natural orbitals. *J. Chem. Phys.* **2017**, *146*, 174108.
(198) 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.
(199) Schwilk, M.; Ma, Q.; Köppl, C.; Werner, H.-J. Scalable electron correlation methods. 3. Efficient and accurate parallel local coupled cluster with pair natural orbitals (PNO-LCCSD). *J. Chem. Theory Comput.* **2017**, *13*, 3650−3675.
(200) Ma, Q.; Schwilk, M.; Köppl, C.; Werner, H.-J. Scalable electron correlation methods. 4. Parallel explicitly correlated local coupled cluster with pair natural orbitals (PNO-LCCSD-F12). *J. Chem. Theory Comput.* **2017**, *13*, 4871−4896.
(201) Ma, Q.; Werner, H.-J. Scalable electron correlation methods. 5. Parallel perturbative triples correction for explicitly correlated local coupled cluster with pair natural orbitals. *J. Chem. Theory Comput.* **2018**, *14*, 198−215.
(202) Demel, O.; Pittner, J.; Neese, F. A local pair natural orbital-based multireference Mukherjee's coupled cluster method. *J. Chem. Theory Comput.* **2015**, *11*, 3104−3114.
(203) Menezes, F.; Kats, D.; Werner, H.-J. Local complete active space second-order perturbation theory using pair natural orbitals (PNO-CASPT2). *J. Chem. Phys.* **2016**, *145*, 124115.
(204) Guo, Y.; Sivalingam, K.; Valeev, E. F.; Neese, F. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. III. Linear-scaling multireference domain-based pair natural orbital N-electron valence perturbation theory. *J. Chem. Phys.* **2016**, *144*, No. 094111.
(205) Brabec, J.; Lang, J.; Saitow, M.; Pittner, J.; Neese, F.; Demel, O. Domain-based local pair natural orbital version of Mukherjees state-specific coupled cluster method. *J. Chem. Theory Comput.* **2018**, *14*, 1370−1382.
(206) Frank, M. S.; Schmitz, G.; Hättig, C. The PNO−MP2 gradient and its application to molecular geometry optimizations. *Mol. Phys.* **2017**, *115*, 343−356.
(207) Pinski, P.; Neese, F. Communication: Exact analytical derivatives for the domain-based local pair natural orbital MP2 method (DLPNO-MP2). *J. Chem. Phys.* **2018**, *148*, No. 031101.
(208) Pinski, P.; Neese, F. Analytical gradient for the domain-based local pair natural orbital second order Møller-Plesset perturbation theory method (DLPNO-MP2). *J. Chem. Phys.* **2019**, *150*, 164102.
(209) Helmich, B.; Hättig, C. A pair natural orbital implementation of the coupled cluster model CC2 for excitation energies. *J. Chem. Phys.* **2013**, *139*, No. 084114.
(210) Helmich, B.; Hättig, C. A pair natural orbital based implementation of ADC(2)-x: Perspectives and challenges for response methods for singly and doubly excited states in large molecules. *Comput. Theor. Chem.* **2014**, *1040−1041*, 35−44.
(211) Frank, M. S.; Hättig, C. A pair natural orbital based implementation of CCSD excitation energies within the framework of linear response theory. *J. Chem. Phys.* **2018**, *148*, 134102.
(212) Mester, D.; Nagy, P. R.; Kállay, M. Reduced-scaling correlation methods for the excited states of large molecules: Implementation and benchmarks for the second-order algebraic-

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

diagrammatic construction approach. *J. Chem. Theory Comput.* **2019**, *15*, 6111−6126.
(213) Dutta, A. K.; Saitow, M.; Riplinger, C.; Neese, F.; Izsák, R. A near-linear scaling equation of motion coupled cluster method for ionized states. *J. Chem. Phys.* **2018**, *148*, 244101.
(214) Dutta, A. K.; Saitow, M.; Demoulin, B.; Neese, F.; Izsák, R. A domain-based local pair natural orbital implementation of the equation of motion coupled cluster method for electron attached states. *J. Chem. Phys.* **2019**, *150*, 164123.
(215) Haldar, S.; Riplinger, C.; Demoulin, B.; Neese, F.; Izsak, R.; Dutta, A. K. Multilayer approach to the IP-EOM-DLPNO-CCSD method: Theory, implementation, and application. *J. Chem. Theory Comput.* **2019**, *15*, 2265−2277.
(216) Fletcher, G. D.; Fedorov, D. G.; Pruitt, S. R.; Windus, T. L.; Gordon, M. S. Large-scale MP2 calculations on the Blue Gene architecture using the fragment molecular orbital method. *J. Chem. Theory Comput.* **2012**, *8*, 75−79.
(217) Wong, H. C.; Paldus, J. Computer generation of Feynman diagrams for perturbation theory I. General algorithm. *Comput. Phys. Commun.* **1973**, *6*, 1−8.
(218) Arthuis, P.; Duguet, T.; Tichai, A.; Lasseri, R.-D.; Ebran, J.-P. ADG: Automated generation and evaluation of many-body diagrams I. Bogoliubov many-body perturbation theory. *Comput. Phys. Commun.* **2019**, *240*, 202−227.
(219) Xiao, B.; Wang, H.; Zhu, S.-h. A simple algorithm for automatic Feynman diagram generation. *Comput. Phys. Commun.* **2013**, *184*, 1966−1972.
(220) Windus, T. L.; Pople, J. A. Pinnacle: An approach toward object oriented quantum chemistry. *Int. J. Quantum Chem.* **1995**, *56*, 485−495.
(221) Werner, H. J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M. Molpro: A general-purpose quantum chemistry program package. *WIREs Comput. Mol. Sci.* **2012**, *2*, 242−253.
(222) Epifanovsky, E.; Wormit, M.; Kuś, T.; Landau, A.; Zuev, D.; Khistyaev, K.; Manohar, P.; Kaliman, I.; Dreuw, A.; Krylov, A. I. New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations. *J. Comput. Chem.* **2013**, *34*, 2293−2309.
(223) GitHub page of TiledArray library. https://github.com/ValeevGroup/tiledarray (accessed Nov 18, 2020).
(224) GitHub page of itensor library. https://itensor.org/ (accessed Nov 18, 2020).
(225) Solomonik, E.; Matthews, D.; Hammond, J.; Demmel, J. Cyclops tensor framework: Reducing communication and eliminating load imbalance in massively parallel contractions. *2013 IEEE 27th Int. Symp. Parallel Distrib. Process.* 2013; pp 813−824.
(226) Kats, D.; Manby, F. R. Sparse tensor framework for implementation of general local correlation methods. *J. Chem. Phys.* **2013**, *138*, 144101.
(227) Matthews, D. A. High-performance tensor contraction without transposition. *SIAM J. Sci. Comput.* **2018**, *40*, C1−C24.
(228) Sanders, B. A.; Bartlett, R.; Deumens, E.; Lotrich, V.; Ponton, M. A block-oriented language and runtime system for tensor algebra with very large arrays. *2010 ACM/IEEE Int. Conf. High Perform. Comput. Networking, Storage Anal.* 2010; pp 1−11.
(229) Stanton, J. F.; Gauss, J.; Watts, J. D.; Bartlett, R. J. A direct product decomposition approach for symmetry exploitation in many-body methods. I. Energy calculations. *J. Chem. Phys.* **1991**, *94*, 4334−4345.
(230) Hampel, C.; Peterson, K. A.; Werner, H.-J. A comparison of the efficiency and accuracy of the quadratic configuration interaction (QCISD), coupled cluster (CCSD), and Brueckner coupled cluster (BCCD) methods. *Chem. Phys. Lett.* **1992**, *190*, 1−12.
(231) Kállay, M.; Surján, P. R. Higher excitations in coupled-cluster theory. *J. Chem. Phys.* **2001**, *115*, 2945−2954.
(232) 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.
(233) Engels-Putzka, A.; Hanrath, M. A fully simultaneously optimizing genetic approach to the highly excited coupled-cluster factorization problem. *J. Chem. Phys.* **2011**, *134*, 124106.
(234) Lai, P.-W.; Zhang, H.; Rajbhandari, S.; Valeev, E.; Kowalski, K.; Sadayappan, P. Effective utilization of tensor symmetry in operation optimization of tensor contraction expressions. *Chem. Phys. Lett.* **2012**, *9*, 412−421.
(235) Aprà, E.; Rendell, A. P.; Harrison, R. J.; Tipparaju, V.; deJong, W. A.; Xantheas, S. S. Liquid water: Obtaining the right answer for the right reasons. *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. New York, NY, USA, 2009.
(236) Liu, S.; Gotz, T. Hadamard, Khatri-Rao, Kronecker and other matrix products. *International Journal of Information and Systems Sciences* **2008**, *4*, 160−177.
(237) Blackford, L. S.; Demmel, J.; Dongarra, J.; Duff, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A.; Pozo, R.; Remington, K.; Whaley, R. C. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software (TOMS)* **2002**, *28*, 135−151.
(238) Feautrier, P.; Lengauer, C. In *Encyclopedia of Parallel Computing*; Padua, D., Ed.; Springer, 2011; pp 1581−1592.
(239) Bondhugula, U.; Hartono, A.; Ramanujam, J.; Sadayappan, P. A practical automatic polyhedral parallelizer and locality optimizer. *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. New York, NY, USA, 2008; pp 101−113.
(240) Mochizuki, Y.; Yamashita, K.; Murase, T.; Nakano, T.; Fukuzawa, K.; Takematsu, K.; Watanabe, H.; Tanaka, S. Large scale FMO-MP2 calculations on a massively parallel-vector computer. *Chem. Phys. Lett.* **2008**, *457*, 396−403.
(241) Sterling, T.; Anderson, M.; Brodowicz, M. In *High performance computing*; Sterling, T., Anderson, M., Brodowicz, M., Eds.; Morgan Kaufmann: Boston, 2018; pp 83−114.
(242) Newriver compute cluster at Virginia Tech. https://www.arc.vt.edu/computing/newriver (accessed Nov 18, 2020).
(243) Cascades compute cluster at Virginia Tech. https://www.arc.vt.edu/computing/cascades (accessed Nov 18, 2020).
(244) List of top 500 supercomputers in the world in 2020. https://www.top500.org/lists/top500/2020/06/ (accessed Nov 18, 2020).
(245) Dongarra, J.; Sterling, T.; Simon, H.; Strohmaier, E. High-performance computing: Clusters, constellations, MPPs, and future directions. *Comput. Sci. Eng.* **2005**, *7*, 51−59.
(246) NVIDIA. Accelerated computing and the democratization of supercomputing. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/sc18-tesla-democratization-tech-overview-r4-web.pdf, 2018; (accessed Nov 18, 2020).
(247) List of top 500 power efficient supercomputers in the world in 2020. https://www.top500.org/lists/green500/list/2020/06/ (accessed Nov 18, 2020).
(248) Summit supercomputer at Oakridge national lab. https://www.olcf.ornl.gov/summit/ (accessed Nov 18, 2020).
(249) Mira supercomputer at Argonne national lab. https://www.alcf.anl.gov/alcf-resources/mira (accessed Nov 18, 2020).
(250) Aurora supercomputer at Argonne national lab. https://aurora.alcf.anl.gov/ (accessed Nov 18, 2020).
(251) Frontier supercomputer to be installed at Oakridge national lab. https://www.olcf.ornl.gov/frontier/ (accessed Nov 18, 2020).
(252) China's exascale compute system. https://www.nextplatform.com/2019/05/02/china-fleshes-out-exascale-design-for-tianhe-3/amp (accessed Nov 18, 2020).
(253) Fugaku supercomputer at RIKEN Center for Computational Science, Japan. https://www.r-ccs.riken.jp/en/postk/project (accessed Nov 18, 2020).
(254) Tensor processing unit of google. https://cloud.google.com/tpu/ (accessed Nov 18, 2020).
(255) Tensor cores in NVIDIA. https://www.nvidia.com/en-us/data-center/tensorcore/ (accessed Nov 18, 2020).

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

(256) Roe, R. FPGAs and the road to reprogrammable HPC. *Inside HPC* **2019**, *7*.
(257) Kumar, V.; Grama, A.; Gupta, A.; Karypis, G. *Introduction to parallel computing: design and analysis of algorithms*; Benjamin-Cummings Publishing Co., Inc., 1994.
(258) Janssen, C. L.; Nielsen, I. M. B. *Parallel computing in quantum chemistry*, 1st ed.; CRC Press, 2008.
(259) Bernholdt, D. E.; Harrison, R. J. Large-scale correlated electronic structure calculations: the RI-MP2 method on parallel computers. *Chem. Phys. Lett.* **1996**, *250*, 477−484.
(260) Bernholdt, D. Scalability of correlated electronic structure calculations on parallel computers: A case study of the RI-MP2 method. *Parallel Computing* **2000**, *26*, 945−963.
(261) Katouda, M.; Nagase, S. Efficient parallel algorithm of second-order Møller−Plesset perturbation theory with resolution-of-identity approximation (RI-MP2). *Int. J. Quantum Chem.* **2009**, *109*, 2121−2130.
(262) 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.
(263) Dachsel, H.; Lischka, H.; Shepard, R.; nieplocha, J.; Harrison, R. J. A massively parallel multireference configuration interaction program: The parallel COLUMBUS program. *J. Comput. Chem.* **1997**, *18*, 430−448.
(264) Lischka, H.; Müller, T.; Szalay, P. G.; Shavitt, I.; Pitzer, R. M.; Shepard, R. Columbus—a program system for advanced multi-reference theory calculations. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 191−199.
(265) Vogiatzis, K. D.; Ma, D.; Olsen, J.; Gagliardi, L.; De Jong, W. A. Pushing configuration-interaction to the limit: Towards massively parallel MCSCF calculations. *J. Chem. Phys.* **2017**, *147*, 184111.
(266) Umeda, H.; Koseki, S.; Nagashima, U.; Schmidt, M. W. Parallelization of multireference perturbation calculations with GAMESS. *J. Comput. Chem.* **2001**, *22*, 1243−1251.
(267) Umeda, H.; Koseki, S.; Nagashima, U. Improvement of parallelization performance of GAMESS: Global sum and (semi-)direct integral calculation in multireference perturbation calculation. *J. Comput. Chem.* **2004**, *25*, 1175−1183.
(268) Yanai, T.; Saitow, M.; Xiong, X.-G.; Chalupský, J.; Kurashige, Y.; Guo, S.; Sharma, S. Multistate complete-active-space second-order perturbation theory based on density matrix renormalization group reference states. *J. Chem. Theory Comput.* **2017**, *13*, 4829−4840.
(269) Park, J. W.; Shiozaki, T. On-the-fly CASPT2 surface-hopping dynamics. *J. Chem. Theory Comput.* **2017**, *13*, 3676−3683.
(270) Watts, J. D. Parallel algorithms for coupled-cluster methods. *Parallel Computing* **2000**, *26*, 857−867.
(271) Rendell, A. P.; Lee, T. J.; Lindh, R. Quantum chemistry on parallel computer architectures: Coupled-cluster theory applied to the bending potential of fulminic acid. *Chem. Phys. Lett.* **1992**, *194*, 84−94.
(272) Saebø, S.; Pulay, P. Fourth-order Møller−Plessett perturbation theory in the local correlation treatment. I. Method. *J. Chem. Phys.* **1987**, *86*, 914.
(273) Rendell, A. P.; Guest, M. F.; Kendall, R. A. Distributed data parallel coupled-cluster algorithm: Application to the 2-hydroxypyridine/2-pyridone tautomerism. *J. Comput. Chem.* **1993**, *14*, 1429−1439.
(274) Kobayashi, R.; Rendell, A. P. A direct coupled cluster algorithm for massively parallel computers. *Chem. Phys. Lett.* **1997**, *265*, 1−11.
(275) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H.J. J.; Wang, D.; Nieplocha, J.; Aprà, E.; Windus, T. L.; de Jong, W. A. NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477−1489.
(276) Nieplocha, J.; Harrison, R. J.; Littlefield, R. J. Global Arrays: A portable "shared-memory" programming model for distributed memory computers. *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing* **1994**, 340−349.
(277) Nieplocha, J.; Harrison, R. J.; Littlefield, R. Global arrays: A nonuniform memory access programming model for high-performance computers. *J. Supercomput.* **1996**, *10*, 169−189.
(278) Anisimov, V. M.; Bauer, G. H.; Chadalavada, K.; Olson, R. M.; Glenski, J. W.; Kramer, W. T. C.; Aprà, E.; Kowalski, K. Optimization of the coupled cluster implementation in NWChem on petascale parallel architectures. *J. Chem. Theory Comput.* **2014**, *10*, 4307−4316.
(279) Peng, C.; Calvin, J. A.; Valeev, E. F. Coupled-cluster singles, doubles and perturbative triples with density fitting approximation for massively parallel heterogeneous platforms. *Int. J. Quantum Chem.* **2019**, *119*, No. e25894.
(280) Auer, A. A.; et al. Automatic code generation for many-body electronic structure methods: the tensor contraction engine. *Mol. Phys.* **2006**, *104*, 211−228.
(281) Kowalski, K.; Krishnamoorthy, S.; Olson, R. M.; Tipparaju, V.; Aprà, E. Scalable implementations of accurate excited-state coupled cluster theories: Application of high-level methods to porphyrin-based systems. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 2011; pp 1−10.
(282) Hu, H.-S.; Bhaskaran-Nair, K.; Aprà, E.; Govind, N.; Kowalski, K. Toward enabling large-scale open-shell equation-of-motion coupled cluster calculations: triplet states of β-carotene. *J. Phys. Chem. A* **2014**, *118*, 9087−93.
(283) Janowski, T.; Ford, A. R.; Pulay, P. Parallel calculation of coupled cluster singles and doubles wave functions using array files. *J. Chem. Theory Comput.* **2007**, *3*, 1368−1377.
(284) Janowski, T.; Pulay, P. Efficient parallel implementation of the CCSD external exchange operator and the perturbative triples (T) energy calculation. *J. Chem. Theory Comput.* **2008**, *4*, 1585−1592.
(285) Ford, A. R.; Janowski, T.; Pulay, P. Array files for computational chemistry: MP2 energies. *J. Comput. Chem.* **2007**, *28*, 1215−1220.
(286) Harding, M. E.; Metzroth, T.; Gauss, J.; Auer, A. A. Parallel calculation of CCSD and CCSD(T) analytic first and second derivatives. *J. Chem. Theory Comput.* **2008**, *4*, 64−74.
(287) Wang, M.; May, A. J.; Knowles, P. J. Parallel programming interface for distributed data. *Comput. Phys. Commun.* **2009**, *180*, 2673−2679.
(288) Olson, R. M.; Bentz, J. L.; Kendall, R. A.; Schmidt, M. W.; Gordon, M. S. A novel approach to parallel coupled cluster calculations: Combining distributed and shared memory techniques for modern cluster based systems. *J. Chem. Theory Comput.* **2007**, *3*, 1312−1328.
(289) Piecuch, P.; Kucharski, S. A.; Kowalski, K.; Musiał, M. Efficient computer implementation of the renormalized coupled-cluster methods: The R-CCSD[T], R-CCSD(T), CR-CCSD[T], and CR-CCSD(T) approaches. *Comput. Phys. Commun.* **2002**, *149*, 71−96.
(290) Asadchev, A.; Gordon, M. S. Fast and flexible coupled cluster implementation. *J. Chem. Theory Comput.* **2013**, *9*, 3385−3392.
(291) Deumens, E.; Lotrich, V. F.; Perera, A. S.; Bartlett, R. J.; Jindal, N.; Sanders, B. A. *Annu. Rep. Comput. Chem.*; Elsevier, 2011; Vol. *7*; pp 179−191.
(292) Lotrich, V.; Flocke, N.; Ponton, M.; Yau, A. D.; Perera, A.; Deumens, E.; Bartlett, R. J. Parallel implementation of electronic structure energy, gradient, and hessian calculations. *J. Chem. Phys.* **2008**, *128*, 194104.
(293) Kuś, T.; Lotrich, V. F.; Bartlett, R. J. Parallel implementation of the equation-of-motion coupled-cluster singles and doubles method and application for radical adducts of cytosine. *J. Chem. Phys.* **2009**, *130*, 124122.
(294) Sanders, B. A.; Byrd, J. N.; Jindal, N.; Lotrich, V. F.; Lyakh, D.; Perera, A.; Bartlett, R. J. Aces4: A platform for computational chemistry calculations with extremely large block-sparse arrays. *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2017; pp 555−564.

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

(295) Solomonik, E.; Matthews, D.; Hammond, J. R.; Stanton, J. F.; Demmel, J. A massively parallel tensor contraction framework for coupled-cluster computations. *J. Parallel Distrib. Comput.* **2014**, *74*, 3176−3190.
(296) Ibrahim, K. Z.; Epifanovsky, E.; Williams, S.; Krylov, A. I. Cross-scale efficient tensor contractions for coupled cluster computations through multiple programming model backends. *Journal of Parallel and Distributed Computing* **2017**, *106*, 92−105.
(297) Solomonik, E.; Demmel, J. *Euro-Par 2011 Parallel Processing: 17th International Conference, Euro-Par 2011, Bordeaux, France, August 29 - September 2, 2011, Proceedings, Part II*; Springer: Berlin, Heidelberg, 2011; pp 90−109.
(298) Van De Geijn, R. A.; Watts, J. SUMMA: Scalable universal matrix multiplication algorithm. *Concurr. Pract. Exp.* **1997**, *9*, 255−274.
(299) Timings for TCE implementation of CCSD in NWChem. http://www.nwchem-sw.org/index.php/Benchmarks#Current_developments_for_high_accuracy:_alternative_task_schedulers_.28ATS.29 (accessed Nov 18, 2020).
(300) Titan supercomputer at Oakridge national lab. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/ (accessed Nov 18, 2020).
(301) Peng, C.; Lewis, C. A.; Wang, X.; Clement, M. C.; Pierce, K.; Rishi, V.; Pavoševic, F.; Slattery, S.; Zhang, J.; Teke, N.; Kumar, A.; Masteran, C.; Asadchev, A.; Calvin, J. A.; Valeev, E. F. Massively parallel quantum chemistry: A high-performance research platform for electronic structure. *J. Chem. Phys.* **2020**, *153*, No. 044120.
(302) Neese, F. The ORCA program system. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2012**, *2*, 73−78.
(303) Smith, D. G. A.; et al. PSI4 1.4: Open-source software for high-throughput quantum chemistry. *J. Chem. Phys.* **2020**, *152*, 184108.
(304) Blueridge compute cluster at Virginia Tech. https://www.arc.vt.edu/computing/blueridge (accessed Nov 18, 2020).
(305) Brauer, B.; Kesharwani, M. K.; Kozuch, S.; Martin, J. M. L. The S66 × 8 benchmark for noncovalent interactions revisited: explicitly correlated ab initio methods and density functional theory. *Phys. Chem. Chem. Phys.* **2016**, *18*, 20905−20925.
(306) Shen, T.; Zhu, Z.; Zhang, I. Y.; Scheffler, M. Massive-parallel implementation of the resolution-of-identity coupled-cluster approaches in the numeric atom-centered orbital framework for molecular systems. *J. Chem. Theory Comput.* **2019**, *15*, 4721−4734.
(307) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab initio molecular simulations with numeric atom-centered orbitals. *Comput. Phys. Commun.* **2009**, *180*, 2175−2196.
(308) DePrince, A. E.; Sherrill, C. D. Accuracy and efficiency of coupled-cluster theory using density fitting/Cholesky decomposition, frozen natural orbitals, and a t1-transformed Hamiltonian. *J. Chem. Theory Comput.* **2013**, *9*, 2687−2696.
(309) Gyevi-Nagy, L.; Kappay, M.; Nagy, P. R. Integral-direct and parallel implementation of the CCSD(T) method: Algorithmic developments and large-scale applications. *J. Chem. Theory Comput.* **2020**, *16*, 366−384.
(310) Rendell, A. P.; Lee, T. J.; Komornicki, A.; Wilson, S. Evaluation of the contribution from triply excited intermediates to the fourth-order perturbation theory energy on Intel distributed memory supercomputers. *Theoretica Chimica Acta* **1993**, *84*, 271−287.
(311) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. Acceleration of streamed tensor contraction expressions on GPGPU-based clusters. *2010 IEEE International Conference on Cluster Computing*. 2010; pp 207−216.
(312) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. GPU-based implementations of the noniterative regularized-CCSD(T) corrections: Applications to strongly correlated systems. *J. Chem. Theory Comput.* **2011**, *7*, 1316−1327.
(313) Aprà, E.; Klemm, M.; Kowalski, K. Efficient implementation of many-body quantum chemical methods on the Intel Xeon PhiTM coprocessor. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2014; p 674−684.
(314) Bentz, J. L.; Olson, R. M.; Gordon, M. S.; Schmidt, M. W.; Kendall, R. A. Coupled cluster algorithms for networks of shared memory parallel processors. *Comput. Phys. Commun.* **2007**, *176*, 589−600.
(315) Rendell, A. P.; Lee, T. J.; Komornicki, A. A parallel vectorized implementation of triple excitations in CCSD (T): Application to the binding energies of the AlH3, AlH2F, AlHF2 and AlF3 dimers. *Chem. Phys. Lett.* **1991**, *178*, 462−470.
(316) Kowalski, K.; Krishnamoorthy, S.; Villa, O.; Hammond, J. R.; Govind, N. Active-space completely-renormalized equation-of-motion coupled-cluster formalism: Excited-state studies of green fluorescent protein, free-base porphyrin, and oligoporphyrin dimer. *J. Chem. Phys.* **2010**, *132*, 154103.
(317) Fan, P.-D.; Valiev, M.; Kowalski, K. Large-scale parallel calculations with combined coupled cluster and molecular mechanics formalism: Excitation energies of zincporphyrin in aqueous solution. *Chem. Phys. Lett.* **2008**, *458*, 205−209.
(318) Kowalski, K.; Hammond, J. R.; De Jong, W. A.; Fan, P. D.; Valiev, M.; Wang, D.; Govind, N. *Computational methods for large systems: Electronic structure approaches for biotechnology and nanotechnology*; John Wiley and Sons: Hoboken, NJ, USA, 2011; pp 167−200.
(319) Straatsma, T.; Bylaska, E.; van Dam, H.; Govind, N.; de Jong, W.; Kowalski, K.; Valiev, M. In *Annual Reports in Computational Chemistry*; Wheeler, R. A., Ed.; Annual Reports in Computational Chemistry; Elsevier, 2011; Vol. 7; pp 151−177.
(320) Verma, P.; Perera, A.; Morales, J. A. New massively parallel linear-response coupled-cluster module in ACES III: Application to static polarisabilities of closed-shell molecules and oligomers and of open-shell radicals. *Mol. Phys.* **2015**, *11*, 1−15.
(321) Peng, C.; Clement, M. C.; Valeev, E. F. State-averaged pair natural orbitals for excited states: A route toward efficient equation of motion coupled-cluster. *J. Chem. Theory Comput.* **2018**, *14*, 5597−5607.
(322) Teke, N. K.; Pavošević, F.; Peng, C.; Valeev, E. F. Explicitly correlated renormalized second-order Greens function for accurate ionization potentials of closed-shell molecules. *J. Chem. Phys.* **2019**, *150*, 214103.
(323) Rishi, V.; Valeev, E. F. Can the distinguishable cluster approximation be improved systematically by including connected triples? *J. Chem. Phys.* **2019**, *151*, No. 064102.
(324) Ma, Q.; Werner, H.-J. Scalable electron correlation methods. 2. Parallel PNO-LMP2-F12 with near linear scaling in the molecular size. *J. Chem. Theory Comput.* **2015**, *11*, 5291−5304.
(325) DePrince, A. E., III; Hammond, J. R.; Sherrill, C. D. *Electronic structure calculations on graphics processing units*; John Wiley & Sons, Ltd, 2016; Chapter 13, pp 279−300.
(326) DePrince, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. Density-fitted singles and doubles coupled cluster on graphics processing units. *Mol. Phys.* **2014**, *112*, 844−852.
(327) Eriksen, J. J. Efficient and portable acceleration of quantum chemical many-body methods in mixed floating point precision using OpenACC compiler directives. *Mol. Phys.* **2017**, *115*, 2086−2101.
(328) Kaliman, I. A.; Krylov, A. I. New algorithm for tensor contractions on multi-core CPUs, GPUs, and accelerators enables CCSD and EOM-CCSD calculations with over 1000 basis functions on a single compute node. *J. Comput. Chem.* **2017**, *38*, 842−853.
(329) Lyakh, D. I. An efficient tensor transpose algorithm for multicore CPU, Intel Xeon Phi, and NVidia Tesla GPU. *Comput. Phys. Commun.* **2015**, *189*, 84−91.
(330) Hynninen, A.-P.; Lyakh, D. I. cuTT: A high-performance tensor transpose library for CUDA compatible GPUs. *arXiv* **2017**.
(331) NVIDIA's cuTensor library. https://docs.nvidia.com/cuda/cutensor/index.html (accessed Nov 18, 2020).
(332) Kim, J.; Sukumaran-Rajam, A.; Hong, C.; Panyala, A.; Srivastava, R. K.; Krishnamoorthy, S.; Sadayappan, P. Optimizing

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX

tensor contractions in CCSD(T) for efficient execution on GPUs. *ICS '18*. New York, New York, USA, 2018; pp 96−106.
(333) Kim, J.; Sukumaran-Rajam, A.; Thumma, V.; Krishnamoorthy, S.; Panyala, A.; Pouchet, L.-N.; Rountev, A.; Sadayappan, P. A code generator for high-performance tensor contractions on GPUs. *EEE/ACM International Symposium on Code Generation and Optimization*. 2019; pp 85−95.
(334) Springer, P.; Bientinesi, P. Design of a high-performance GEMM-like tensor−tensor multiplication. *ACM Trans. Math. Softw.* **2018**, *44*, 1−29.
(335) Di Napoli, E.; Fabregat-Traver, D.; Quintana-Ortí, G.; Bientinesi, P. Towards an efficient use of the BLAS library for multilinear tensor contractions. *Applied Mathematics and Computation* **2014**, *235*, 454−468.
(336) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K.; Agrawal, G. Optimizing tensor contraction expressions for hybrid CPU-GPU execution. *Cluster Comput* **2013**, *16*, 131−155.
(337) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. GPU-based implementations of the noniterative regularized-CCSD(T) corrections: Applications to strongly correlated systems. *J. Chem. Theory Comput.* **2011**, *7*, 1316−1327.
(338) Bhaskaran-Nair, K.; Ma, W.; Krishnamoorthy, S.; Villa, O.; van Dam, H. J. J.; Aprà, E.; Kowalski, K. Noniterative multireference coupled cluster methods on heterogeneous CPU−GPU systems. *J. Chem. Theory Comput.* **2013**, *9*, 1949−1957.
(339) Ma, W.; Bhaskaran-Nair, K.; Villa, O.; Aprà, E.; Tumeo, A.; Krishnamoorthy, S.; Kowalski, K. *Electronic structure calculations on graphics processing units*; John Wiley & Sons, Ltd: Chichester, UK, 2016; pp 301−326.
(340) GitHub page of a custom fork of cuTT library. https://github.com/ValeevGroup/cutt/ (accessed Nov 18, 2020).
(341) Nvidia, C. Cublas library. *NVIDIA Corporation, Santa Clara, California* **2008**, *15*, 31.
(342) GitHub page of Umpire library. https://github.com/LLNL/Umpire (accessed Nov 18, 2020).

https://dx.doi.org/10.1021/acs.chemrev.0c00006
Chem. Rev. XXXX, XXX, XXX−XXX