JCTC Journal of Chemical Theory and Computation

Article

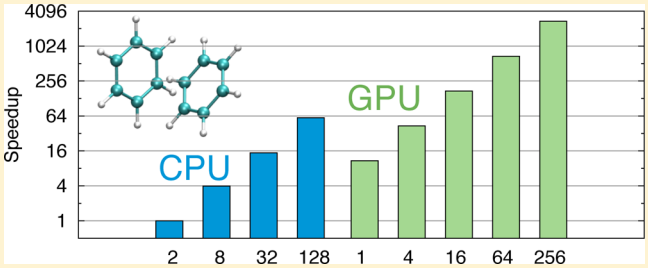
pubs.acs.org/JCTC

Monte Carlo MP2 on Many Graphical Processing Units

Alexander E. Doran and So Hirata*

Department of Chemistry, University of Illinois at Urbana−Champaign, 600 South Mathews Avenue, Urbana, Illinois 61801, United States

ABSTRACT: In the Monte Carlo second-order many-body perturbation (MC-MP2) method, the long sum-of-product matrix expression of the MP2 energy, whose literal evaluation may be poorly scalable, is recast into a single high-dimensional integral of functions of electron pair coordinates, which is evaluated by the scalable method of Monte Carlo integration. The sampling efficiency is further accelerated by the redundant-walker algorithm, which allows a maximal reuse of electron pairs. Here, a multitude of graphical processing units (GPUs) offers a uniquely ideal platform to expose multilevel parallelism: fine-grain data-parallelism for the redundant-walker algorithm in which millions of threads compute and share orbital amplitudes on each GPU; coarse-grain instruction-parallelism for near-independent Monte Carlo integrations on many GPUs with few and infrequent interprocessor communications. While the efficiency boost by the redundant-walker algorithm on central processing units (CPUs) grows linearly with the number of electron pairs and tends to saturate when the latter exceeds the number of orbitals, on a GPU it grows quadratically before it increases linearly and then eventually saturates at a much larger number of pairs. This is because the orbital constructions are nearly perfectly parallelized on a GPU and thus completed in a near-constant time regardless of the number of pairs. In consequence, an MC-MP2/cc-pVDZ calculation of a benzene dimer is 2700 times faster on 256 GPUs (using 2048 electron pairs) than on two CPUs, each with 8 cores (which can use only up to 256 pairs effectively). We also numerically determine that the cost to achieve a given relative statistical uncertainty in an MC-MP2 energy increases as $O(n^3)$ or better with system size n, which may be compared with the $O(n^5)$ scaling of the conventional implementation of deterministic MP2. We thus establish the scalability of MC-MP2 with both system and computer sizes.



1. INTRODUCTION

In 2009, the fastest supercomputer was Cray XT5 Jaguar at Oak Ridge National Laboratory, which consisted of 224 162 AMD Opteron processor cores, according to the TOP500 ranking. In 2012, IBM Blue Gene/Q Sequoia at Lawrence Livermore National Laboratory took the top spot with 98 304 nodes of 16-core PowerPC A2 central processing units (CPUs) for a total of 1 572 864 processor cores. It was already obvious then that the conventional algorithms of computational quantum chemistry based on dense matrix multiplications were not strongly scalable on such machines with hundreds of thousands or even millions of processor cores. This is because interprocessor communications are pervasive in such algorithms and hard to eliminate, making the parallel scalability saturate quickly with the number of CPUs. One may, therefore, have to rethink and redesign quantum-chemistry algorithms for today's supercomputing architectures, which will become tomorrow's commodity hardware.

In late 2012, Jaguar underwent an upgrade and became Cray XK7 Titan with an addition of 18 688 NVIDIA K20X graphical processing units (GPUs), reclaiming the first place in the ranking. Blue Waters at National Center for Supercomputing Applications (NCSA) of University of Illinois, also built by Cray, consists of 26 868 nodes, of which 22 640 XE nodes are each equipped with two 8-core AMD 6276 Interlagos CPUs and the remaining 4228 XK nodes each with a single AMD 6276 Interlagos CPU and an NVIDIA K20X GPU. They herald the modern and future supercomputing architectures containing both many CPUs and many GPUs. This is further confirmed by the fact that today's second fastest supercomputer, China's Tianhe-2, and the announced upgrades of Titan and Sequoia (to be built in 2017) all adopt GPU-based or GPU-like many-core architectures.

A CPU can perform complex instructions and supports overlapping and out-of-order executions as well as branch predictions, whereas a GPU (also known as a coprocessor or accelerator) applies a relatively simple operation on a large amount of data at once, in the so-called SIMD (single-instruction, multiple-data) parallelism. Therefore, any algorithm attempting to harness the full power of *many* GPUs may need to adopt multilevel parallelism, in which a fine-grain data-parallel part is handled by many threads of a GPU, whereas a coarse-grain instruction-parallel part runs on many GPUs or many CPUs. Dense matrix multiplications may now become suitable for the former, but they remain problematic for the latter.

Scalable algorithms on such large and heterogeneous hardware are few, but they include stochastic (or Monte Carlo) algorithms. They achieve scalability by virtue of being a

Received: June 8, 2016
Published: September 7, 2016

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

sparse-integration method, but incur statistical uncertainties. The quantum Monte Carlo (QMC) methods,[1−5] which solve electronic or vibrational Schrödinger equations with such algorithms, boast wide applicability, high scalability (with both system and computer sizes), and near-exact accuracy when convergence is achieved. It is fascinating to observe the efficacy of stochastic algorithms in its applications to the intrinsically stochastic theories of quantum mechanics and thermodynamics. The QMC methods, however, have weaknesses. One is the so-called sign problem, which necessitates the use of approximate nodal structure of a wave function in, for example, diffusion Monte Carlo. Another is the difficulty in computing energy differences in the presence of large statistical uncertainties in total energies. Yet another is the finite-size error.

One of the present authors with coauthors[6] introduced the Monte Carlo second-order many-body perturbation (MC-MP2) method,[7] which weds *ab initio* molecular orbital theory with QMC-like stochastic algorithms,[1−5] thereby achieving the best of the two worlds.[8−18] In it, the usual long sum-of-product expression of an MP2 correlation energy is transformed into a single 13-dimensional integral of functions of two electron pairs. This integral is then evaluated by Monte Carlo (MC) integration using an appropriate weight function for Metropolis sampling[19] of electron-pair distributions. MC-MP2 is rigorously (diagrammatically) size-consistent, is free from any sign problem, and can compute energy differences directly and not as small differences of noisy total energies. The energy differences include correlation energies and correlated ionization and electron-attachment energies[20] as well as quasiparticle energy bands of a crystal,[21] which are evaluated directly by integrating the perturbation corrections to the energies and self-energies. It can be systematically improved by increasing the perturbation order[22] and basis-set size. It does not require two-electron integrals precomputed and stored either in the atomic-orbital or molecular-orbital basis and is, therefore, more scalable with both system and computer sizes. Explicit correlation[23,24] can also be more easily incorporated.

Being embarrassingly parallel by design (used here in a nonderogatory sense), an MC-MP2 program could easily be made to execute efficiently on many CPUs,[25] by simply having all processes perform independent MC integrations and report their intermediate results to the master process occasionally. Willow et al.[25] also introduced a crucial convergence-acceleration measure called the redundant-walker algorithm. In it, more than minimally necessary two electron pairs (say, m pairs) are propagated at a cost increase by a factor of $m/2$. With m electron pairs, there are $m(m-1)/2$ distinct ways of substituting their coordinates in the integrand of the MC-MP2 energy, resulting in a net increase in sampling efficiency by a factor of $(m-1)$. However, it was found that the speedup does not increase indefinitely with m, but saturates at a rather small value of m, because the $m(m-1)/2$ cost of substitution ultimately becomes the hotspot at large m, nullifying the sampling efficiency boost. In short, the redundant-walker MC-MP2 algorithm consists of two equally important parts with disparate data-access patterns: (1) near-independent MC integrations using disjoint sets of electron pairs and (2) a maximal reuse of electron pairs in each independent MC integration.

Here, we show that the two algorithmic parts naturally expose multilevel parallelism, with part (1) ideally handled in coarse-grain instruction-parallelism on many CPUs or many GPUs involving few and infrequent interprocessor communications, while part (2) can greatly benefit from fine-grain data-parallelism on a GPU with data on many electron pairs rapidly computed and made available on shared memory to all processor cores (threads) that need them. Not only does this algorithmic design lead to a highly scalable MC-MP2 method on many GPUs, but it also brings about a qualitative change in the behavior of the redundant-walker algorithm for the better: whereas the efficiency increase from the redundant-walker algorithm grows only linearly with the number of electron pairs (m) on CPUs, it increases quadratically on GPUs and its saturation with m is significantly delayed. This, in turn, is caused by near-perfect parallelization of the redundant-walker algorithm on a GPU until the number of electron pairs is large. An MC-MP2/cc-pVDZ calculation of a benzene dimer (228 basis functions), for instance, is found to be 11 times faster on a single GPU than on two CPUs (using all of their 16 cores) and 2700 times faster on 256 GPUs than on two CPUs (16 cores). These high degrees of speedup are partly due to the fact that whereas the speedup from the redundant-walker algorithm ceases to increase after 256 pairs on CPUs, as many as 2048 pairs can contribute meaningfully on GPUs.

Earlier, we found[6] that the cost of MC-MP2 *per MC step* increases nearly linearly with the number of basis functions for small molecules. In this work, we confirm that this is the case with MC-MP2 with the redundant-walker algorithm for a more extensive set of molecules containing fairly large ones. We also show here numerically that the scaling of the cost of MC-MP2 *to achieve a given relative statistical uncertainty* is $O(n_{\mathrm{ele}}^3)$, where n_{ele} is the number of electrons or a molecule's spatial size, or $O(n_{\mathrm{bas}}^{2.5})$, where n_{bas} is the number of basis functions for a fixed molecular size. These are a vast improvement over the $O(n_{\mathrm{ele}}^5)$ or $O(n_{\mathrm{bas}}^4)$ scaling of the conventional implementation[26] of deterministic MP2.

Together, we demonstrate that MC-MP2 is scalable with system size and computer size and thus holds a promise as a future of *ab initio* electronic structure theory.

2. MC-MP2 FORMALISM

The usual long sum-of-product matrix expression of the MP2 correlation energy, $E^{(2)}$, can be converted[6] by Laplace transform[27] into a single 13-dimensional integral. For a closed-shell molecule with n_{ele} electrons whose restricted Hartree−Fock (HF) molecular orbitals (MO) are expanded by n_{bas} atomic-orbital (AO) basis functions, the converted expression is

$$E^{(2)} = \int\int\int\int \mathrm{d}\mathbf{r}_1\,\mathrm{d}\mathbf{r}_2\,\mathrm{d}\mathbf{r}_3\,\mathrm{d}\mathbf{r}_4 \times \int_0^\infty \mathrm{d}\tau\, f(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \tau) \tag{1}$$

with

$$\begin{aligned} f(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \tau) = {} & -2G^-(\mathbf{r}_1, \mathbf{r}_3, -\tau)G^-(\mathbf{r}_2, \mathbf{r}_4, -\tau) \\ & \times G^+(\mathbf{r}_1, \mathbf{r}_3, \tau)G^+(\mathbf{r}_2, \mathbf{r}_4, \tau)r_{12}^{-1}r_{34}^{-1} \\ & + G^-(\mathbf{r}_1, \mathbf{r}_3, -\tau)G^-(\mathbf{r}_2, \mathbf{r}_4, -\tau) \\ & \times G^+(\mathbf{r}_1, \mathbf{r}_4, \tau)G^+(\mathbf{r}_2, \mathbf{r}_3, \tau)r_{12}^{-1}r_{34}^{-1} \end{aligned} \tag{2}$$

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

where $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$ and $G^{\pm}(\mathbf{r}, \mathbf{r}', \tau)$ are the traces of the retarded and advanced HF Green's functions in real space and imaginary time:[22]

$$G^{-}(\mathbf{r}, \mathbf{r}', \tau) = \sum_{i=1}^{n_{\mathrm{occ}}} \varphi_i(\mathbf{r})\varphi_i^*(\mathbf{r}')\mathrm{e}^{-\epsilon_i\tau} \tag{3}$$

$$G^{+}(\mathbf{r}, \mathbf{r}', \tau) = \sum_{a=n_{\mathrm{occ}}+1}^{n_{\mathrm{bas}}} \varphi_a(\mathbf{r})\varphi_a^*(\mathbf{r}')\mathrm{e}^{-\epsilon_a\tau} \tag{4}$$

Here, n_{occ} is the number of occupied orbitals, which is equal to $n_{\mathrm{ele}}/2$, $\varphi_p(\mathbf{r})$ is the pth MO, and ϵ_p is the corresponding orbital energy. Hence, the summation in eq 3 runs over occupied MOs, while that in eq 4 over virtual MOs, whose number is n_{vir} (used later). Each MO is, in turn, a linear combination of AOs:

$$\varphi_p(\mathbf{r}) = \sum_{\mu=1}^{n_{\mathrm{bas}}} C_p^{\mu}\chi_{\mu}(\mathbf{r}) \tag{5}$$

where C_p^{μ} is an MO expansion coefficient determined by the preceding HF procedure and $\chi_{\mu}(\mathbf{r})$ is the μth AO, which is not limited to but can be an atom-centered Gaussian-type orbital (GTO), as in our implementation.

In the MC-MP2 method, we evaluate the one-dimensional integration over τ in eq 1 by quadrature and its 12-dimensional real-space integration over electron coordinates, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, and $\mathbf{r}_4$, by the MC method:

$$E^{(2)} \approx \frac{1}{N}\sum_{n=1}^{N} \frac{\tilde{f}(\mathbf{r}_1^{[n]}, \mathbf{r}_2^{[n]}, \mathbf{r}_3^{[n]}, \mathbf{r}_4^{[n]})}{\tilde{w}(\mathbf{r}_1^{[n]}, \mathbf{r}_2^{[n]})\tilde{w}(\mathbf{r}_3^{[n]}, \mathbf{r}_4^{[n]})} \tag{6}$$

with

$$\tilde{f}(\mathbf{r}_1^{[n]}, \mathbf{r}_2^{[n]}, \mathbf{r}_3^{[n]}, \mathbf{r}_4^{[n]}) = \sum_s w_s f(\mathbf{r}_1^{[n]}, \mathbf{r}_2^{[n]}, \mathbf{r}_3^{[n]}, \mathbf{r}_4^{[n]}, \tau_s) \tag{7}$$

where N is the total number of MC steps, $(\mathbf{r}_1^{[n]},\mathbf{r}_2^{[n]})$ is the random coordinates of a pair of electrons 1 and 2 ("a walker") in the nth MC step, which, in a long MC run, forms a distribution according to the weight-function factor, $\tilde{w}(\mathbf{r}_1^{[n]},\mathbf{r}_2^{[n]})$, and τ_s and w_s are the sth Gauss−Kronrod quadrature[28] point and weight, respectively. The random distribution according to $\tilde{w}(\mathbf{r}_1^{[n]},\mathbf{r}_2^{[n]})$ is achieved by the Metropolis algorithm.[19] An appropriate form[6] of the weight function factor is

$$\tilde{w}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{N_g}\frac{g(\mathbf{r}_1)g(\mathbf{r}_2)}{r_{12}} \tag{8}$$

where $g(\mathbf{r})$ is a sum of s-type GTOs, and the normalization coefficient,

$$N_g = \iint \mathrm{d}\mathbf{r}_1\,\mathrm{d}\mathbf{r}_2 \frac{g(\mathbf{r}_1)g(\mathbf{r}_2)}{r_{12}} \tag{9}$$

is evaluated semianalytically.[29]

The redundant-walker algorithm[25] was introduced as an essential convergence-acceleration measure for MC−MP2. In this algorithm, instead of propagating just two electron pairs (two walkers), $(\mathbf{r}_1^{[n]}, \mathbf{r}_2^{[n]})$ and $(\mathbf{r}_3^{[n]}, \mathbf{r}_4^{[n]})$, which are minimally necessary in eq 6, we propagate m ($m \gg 2$) pairs (m "redundant" walkers), $(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]})$, with $1 \le k \le m$. Having m pairs increases the propagation cost by a factor of $m/2$, but it increases the number of distinct ways to substitute the pair coordinates in eq 6 from 2 to $m(m-1)$, a net $(m-1)$-fold increase in the MC sampling efficiency:

$$E^{(2)} \approx I_N \equiv \frac{1}{N}\sum_{n=1}^{N}\frac{1}{m(m-1)}\sum_{k=1}^{m-1}\sum_{l=k+1}^{m} \times \frac{\tilde{f}(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]}, \mathbf{r}_{1l}^{[n]}, \mathbf{r}_{2l}^{[n]}) + \tilde{f}(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]}, \mathbf{r}_{2l}^{[n]}, \mathbf{r}_{1l}^{[n]})}{\tilde{w}(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]})\tilde{w}(\mathbf{r}_{1l}^{[n]}, \mathbf{r}_{2l}^{[n]})} \tag{10}$$

where we exploit the fact that an interchange of $\mathbf{r}_{1l}^{[n]}$ and $\mathbf{r}_{2l}^{[n]}$ also leads to a distinct summand for a speedup by a factor of 2, which was overlooked in our previous work.[6]

The statistical uncertainty σ is computed with the blocking algorithm of Flyvbjerg and Petersen[30] as

$$\sigma^2 = \frac{N_b^2}{N^2}\sum_{n=1}^{N/N_b}\left[\sum_{n'=1}^{N_b}\frac{1}{N_b}\frac{1}{m(m-1)}\sum_{k=1}^{m-1}\sum_{l=k+1}^{m} \times \left\{\frac{\tilde{f}(\mathbf{r}_{1k}^{[n'']}, \mathbf{r}_{2k}^{[n'']}, \mathbf{r}_{1l}^{[n'']}, \mathbf{r}_{2l}^{[n'']})}{\tilde{w}(\mathbf{r}_{1k}^{[n'']}, \mathbf{r}_{2k}^{[n'']})\tilde{w}(\mathbf{r}_{1l}^{[n'']}, \mathbf{r}_{2l}^{[n'']})} + \frac{\tilde{f}(\mathbf{r}_{1k}^{[n'']}, \mathbf{r}_{2k}^{[n'']}, \mathbf{r}_{2l}^{[n'']}, \mathbf{r}_{1l}^{[n'']})}{\tilde{w}(\mathbf{r}_{1k}^{[n'']}, \mathbf{r}_{2k}^{[n'']})\tilde{w}(\mathbf{r}_{1l}^{[n'']}, \mathbf{r}_{2l}^{[n'']})}\right\} - I_N\right]^2 \tag{11}$$

where N_b is the block size and $n'' = (n-1)N_b + n'$.

3. MULTILEVEL GPU PARALLELISM

Before entering the discussion of our parallel algorithm of MC-MP2, we briefly contrast different operational characteristics between CPUs and GPUs.

A CPU (host) can perform complex instructions efficiently, including overlapping (pipelined) and out-of-order executions. It also makes branch predictions. However, interprocessor communications are relatively slow (both in latency and bandwidth) because two CPUs typically do not share memory but are physically distant from each other. Therefore, an efficient algorithm on many CPUs ought to be coarse-grain embarrassingly parallel with as few and infrequent interprocessor communications as possible. It is exceedingly difficult to remove communications from dense matrix multiplications, which are, therefore, not ideal for this part.

A GPU (device) consists of several multiprocessors, each of which can execute a large number of concurrent calculations called threads. Several threads are grouped into a "warp" and several warps into a "block." All threads in a block run on a single multiprocessor and have high-speed access to data on its volatile shared memory or on its persistent global memory. A large number of threads should be spawned, so that no cores are idle, while thread load-balancing is automatically ensured by hardware. All threads in a warp are meant to execute a rather simple operation concurrently on many data. Although each thread in a warp has its own instruction pointer and two threads can in principle execute different instructions, if that takes place, all the others in the same warp must remain idle, a situation known as a "thread divergence." In the worst case scenario where all threads in a warp follow different execution paths, the computation is effectively serialized. An important consideration in using GPUs is, therefore, to avoid thread divergence, by exposing fine-grain SIMD parallelism in the algorithm. Dense matrix multiplications may be appropriate for

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

this part, especially with the advent of the CUBLAS library, a GPU-implementation of the BLAS library provided in the CUDA toolkit. Furthermore, data transfer to/from CPU (host) from/to GPU (device) tends to have a high latency. So another important consideration is to minimize such transfers. Finally, conceived originally as a graphics engine, presently a GPU executes single-precision arithmetic more rapidly, and error correcting codes (ECC) are often optional or unavailable.

An ideal algorithm for *many* GPUs may, therefore, implement multilevel parallelism: coarse-grain instruction-parallelism across many CPUs or many GPUs and fine-grain data-parallelism on each GPU with built-in error/fault tolerance or possibly single-precision arithmetic. Data transfers between two CPUs, between a CPU and a GPU, and between two GPUs should be minimized. Here, we show that MC-MP2 can be implemented exactly in this manner and is ideal for many GPUs.

Figure 1 illustrates the multilevel parallel algorithm of MC-MP2. A more detailed outline of the algorithm is given in

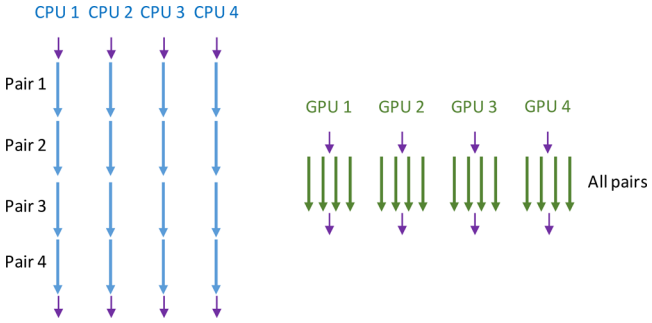


Figure 1. A schematic comparison of the MC-MP2 algorithm on many CPUs and many GPUs. A "pair" refers to an electron pair that undergoes a distorted random walk in six-dimensional real space.

```
C/G  0: Transfer geometry & basis set from CPU to GPU
CPU  1: Loop over MC steps
GPU  2:   Propagate m electron pairs [Eq.(8)]
GPU  3:   Calculate AOs of 2m electrons
GPU  4:   Calculate MOs of 2m electrons [Eq.(5)]
CPU  5:   Loop over τs [Eq.(7)]
GPU  6:     Calculate G± of m(m−1)/2 pairs [Eqs.(3)&(4)]
GPU  7:     Calculate E(2) [Eq.(10)]
CPU  8:   End Loop
C/G  9:   Accumulate & transfer E(2) from GPU to CPU
CPU 10:   Accumulate σ [Eq.(11)]
CPU 11:   If [every 2^n (n = 0, ..., 12) steps] MPI_REDUCE E(2), σ
CPU 12: End loop
```

Figure 2. An outline of the MC-MP2 algorithm.

Figure 2. Each CPU (host) performs its independent MC integration, as indicated by Steps 1 and 12 in Figure 2, not unlike most other QMC implementations. This coarse-grain parallelism can be made scalable to almost any arbitrarily high degree by delaying a synchronized accumulation (MPI_REDUCE) of the energies and statistical uncertainties by all processors. It can also be fault-tolerant (not implemented in this work) and indefinitely restartable (automatically realized) because of the additive nature of the MC integrals and statistical uncertainties.

In our implementation, the energies and statistical uncertainties of all CPUs are accumulated at every 2^n ($7 \le n \le 10$) MC steps, generating a list of statistical uncertainties computed with various block sizes ($N_b = 2^{n_b}$ with $0 \le n_b \le 12$) from a single MC run (the statistical uncertainties with block size $n_b \le n$ are current at every accumulation, while those with block size $n_b > n$ are current at every 2^{n-n_b} accumulations). From this list, one can determine an appropriate block size (which is the smallest value of N_b at which the statistical uncertainty starts to plateau, but before it blows up). In subsequent runs, this optimal value of N_b (typically 128 or 256) can be used as the interval for synchronization for higher scalability.

The redundant-walker algorithm propagates m electron pairs at an $m/2$-fold increased cost, but generates a $m(m-1)/2$-times greater number of MC samplings. We parallelized all computing tasks associated with the redundant-walker algorithm on a GPU (device), that is, steps 2, 3, 4, 6, 7, and 9 of Figure 2, or as shown in Figure 1. Table 1 lists the asymptotic

Table 1. Algorithmic Steps and Their Costs

step[a]	task	cost[b]
2	propagate pairs	$O(mn)$
3	calculate AOs	$O(mn)$
4	calculate MOs	$O(mn^2)$
6	calculate $G^{\pm}$	$O(m^2n)$
7 + 9	calculate E^2	$O(m^2)$

[a]See the corresponding line in Figure 2. [b]m is the number of electron pairs and n is the problem size ($n = n_{bas} \propto n_{ele}$).

cost function of each step. The cost is measured as a function of the number of electron pairs ($m \ge 2$) (which is a parameter set by the user and affects only the performance, but not the converged limits of MC-MP2) and of the problem size (n) (which defines the system size and is equal to n_{bas} and proportional to n_{ele} or the number of atoms). In what follows, we discuss our GPU-parallelization strategy of each step.

Step 2 performs one cycle of the Metropolis algorithm in which m electron pairs make a distorted random walk. The cost of this step is dominated by the evaluation of the value of the weight function (eq 8) at new pair coordinates. It therefore involves $O(mn)$ exponential function calls. This step executes efficiently on either CPU (host) or GPU (device), but performing it on GPU is preferable because it eliminates data transfer needs. Hence, in our implementation, the transfer occurs just once outside all loops (Step 0). One thread propagates one electron pair.

Step 3 calculates the amplitudes of all (n) AOs at m electron pair coordinates at an $O(mn)$ cost. Although the scaling is lower than some other steps, this step can be the hotspot unless m or n is large, because the prefactor on the cost function is the greatest as it involves expensive evaluations of exponential functions and spherical harmonics. To avoid thread divergence, all threads in a block (thus all threads in each warp in the block) are tasked to calculate AOs that share the same spherical harmonics (e.g., d-type GTOs only), but at different electron coordinates concurrently, so that they perform an identical series of instructions and have the same memory-access pattern. The converse algorithm, in which all threads in a block calculate different types of GTOs concurrently at one electron's coordinates, causes a severe thread divergence. The AO amplitudes are stored in $n_{bas} \times m$ arrays $\mathbf{X}_i$ (i = 1 or 2), which persist in the GPU's global memory:

$$(\mathbf{X}_i)_{\mu k} = \chi_\mu(\mathbf{r}_{ik}^{[n]}) \tag{12}$$

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

where i labels electron 1 or 2 in the kth electron pair ($1 \le k \le m$), μ designates an AO, and n counts the MC cycle.

Step 4 generates the whole set of the occupied MO amplitudes in $n_{\mathrm{occ}} \times m$ arrays $\mathbf{\Phi}_i^-$ (i = 1, 2) and the virtual MO amplitudes in $n_{\mathrm{vir}} \times m$ arrays $\mathbf{\Phi}_i^+$ (i = 1, 2):

$$(\mathbf{\Phi}_i^-)_{pk} = \varphi_p(\mathbf{r}_{ik}^{[n]}), \quad 1 \le p \le n_{\mathrm{occ}} \tag{13}$$

$$(\mathbf{\Phi}_i^+)_{pk} = \varphi_{n_{\mathrm{occ}}+p}(\mathbf{r}_{ik}^{[n]}), \quad 1 \le p \le n_{\mathrm{vir}} \tag{14}$$

They are related to the AO amplitudes by

$$\mathbf{\Phi}_i^- = \mathbf{C}^-\mathbf{X}_i \tag{15}$$

$$\mathbf{\Phi}_i^+ = \mathbf{C}^+\mathbf{X}_i \tag{16}$$

where $\mathbf{C}^-$ and $\mathbf{C}^+$ are the $n_{\mathrm{occ}} \times n_{\mathrm{bas}}$ and $n_{\mathrm{vir}} \times n_{\mathrm{bas}}$ arrays of occupied and virtual MO expansion coefficients, respectively; that is,

$$(\mathbf{C}^-)_{p\mu} = C_p^\mu \tag{17}$$

$$(\mathbf{C}^+)_{p\mu} = C_{n_{\mathrm{occ}}+p}^\mu \tag{18}$$

This AO-to-MO transformation is, therefore, $O(mn^2)$ dense matrix multiplications. They are carried out by CUBLASDGEMM (a parallel matrix multiplication subroutine) in the CUBLAS library. For large n, this step becomes the hotspot.

Step 6 is repeated for all grid points of τ in a loop executed by a host CPU, but all data needed in this step are persistent in the global memory of the device GPU. In this step, $m \times m$ arrays of retarded and advanced Green's-function traces are constructed:

$$\{\mathbf{G}_{ij}^-(\tau)\}_{kl} = G^-(\mathbf{r}_{ik}^{[n]}, \mathbf{r}_{jl}^{[n]}, \tau) \tag{19}$$

$$\{\mathbf{G}_{ij}^+(\tau)\}_{kl} = G^+(\mathbf{r}_{ik}^{[n]}, \mathbf{r}_{jl}^{[n]}, \tau) \tag{20}$$

where k and l designate an electron pair ($1 \le k, l \le m$), while i and j take the value of 1 or 2, corresponding to electron 1 or 2, respectively, in each pair. They are defined as

$$\mathbf{G}_{ij}^-(\tau) = (\mathbf{\Phi}_i^-)^\dagger \mathbf{T}^-(\tau)\mathbf{\Phi}_j^- \tag{21}$$

$$\mathbf{G}_{ij}^+(\tau) = (\mathbf{\Phi}_i^+)^\dagger \mathbf{T}^+(\tau)\mathbf{\Phi}_j^+ \tag{22}$$

with

$$\{\mathbf{T}^-(\tau)\}_{pq} = \delta_{pq} e^{-\epsilon_p \tau}, \quad 1 \le p, q \le n_{\mathrm{occ}} \tag{23}$$

$$\{\mathbf{T}^+(\tau)\}_{pq} = \delta_{pq} e^{-\epsilon_{n_{\mathrm{occ}}+p} \tau}, \quad 1 \le p, q \le n_{\mathrm{vir}} \tag{24}$$

These Green's-function arrays have the following symmetry:

$$\mathbf{G}_{ij}^-(\tau) = \{\mathbf{G}_{ji}^-(\tau)\}^\dagger \tag{25}$$

$$\mathbf{G}_{ij}^+(\tau) = \{\mathbf{G}_{ji}^+(\tau)\}^\dagger \tag{26}$$

Therefore, $\mathbf{G}_{11}^\pm(\tau)$ and $\mathbf{G}_{22}^\pm(\tau)$ are constructed by CUBLASDSYRKX (a matrix multiplication subroutine for a symmetric matrix) and $\mathbf{G}_{12}^\pm(\tau)$ by CUBLASDGEMM in the CUBLAS library, whereas $\mathbf{G}_{21}^\pm(\tau)$ is obtained as the transpose of the latter (assuming real orbitals). The cost of these multiplications are $O(m^2n)$, and hence this step can also be a hotspot when m is large.

Step 7 is also within the τ-loop and evaluates the sth grid-point contribution to $\tilde{f}(\mathbf{r}_{1k}^{[n]},\mathbf{r}_{2k}^{[n]},\mathbf{r}_{1l}^{[n]},\mathbf{r}_{2l}^{[n]})$ in eq 10:

$$\begin{aligned}\tilde{f}(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]}, \mathbf{r}_{1l}^{[n]}, \mathbf{r}_{2l}^{[n]}) &= -2\sum_s w_s \{\mathbf{G}_{11}^-(-\tau_s)\}_{kl}\{\mathbf{G}_{22}^-(-\tau_s)\}_{kl} \\ &\quad \times \{\mathbf{G}_{11}^+(\tau_s)\}_{kl}\{\mathbf{G}_{22}^+(\tau_s)\}_{kl} \\ &\quad + \sum_s w_s \{\mathbf{G}_{11}^-(-\tau_s)\}_{kl}\{\mathbf{G}_{22}^-(-\tau_s)\}_{kl} \\ &\quad \times \{\mathbf{G}_{12}^+(\tau_s)\}_{kl}\{\mathbf{G}_{21}^+(\tau_s)\}_{kl}\end{aligned} \tag{27}$$

$$\begin{aligned}\tilde{f}(\mathbf{r}_{1k}^{[n]}, \mathbf{r}_{2k}^{[n]}, \mathbf{r}_{2l}^{[n]}, \mathbf{r}_{1l}^{[n]}) &= -2\sum_s w_s \{\mathbf{G}_{12}^-(-\tau_s)\}_{kl}\{\mathbf{G}_{21}^-(-\tau_s)\}_{kl} \\ &\quad \times \{\mathbf{G}_{12}^+(\tau_s)\}_{kl}\{\mathbf{G}_{21}^+(\tau_s)\}_{kl} \\ &\quad + \sum_s w_s \{\mathbf{G}_{12}^-(-\tau_s)\}_{kl}\{\mathbf{G}_{21}^-(-\tau_s)\}_{kl} \\ &\quad \times \{\mathbf{G}_{11}^+(\tau_s)\}_{kl}\{\mathbf{G}_{22}^+(\tau_s)\}_{kl}\end{aligned} \tag{28}$$

Since $1 \le k < l \le m$ (see eq 10), there are $m(m-1)/2$ multiplications in every summation, and each of these multiplications is carried out by a thread. The cost of this step is always minuscule as compared with step 6 as it is independent of the system size (n).

Step 9 transfers the intermediate results (the energy increments) from each device GPU to its host CPU efficiently by using the REDUCE function provided by the THRUST library of the CUDA toolkit. The cost of this step is negligible because only one floating-point number is transferred from the memory of the GPU to that of the CPU.

4. PERFORMANCE OF THE REDUNDANT-WALKER ALGORITHM

All calculations were performed on Blue Waters at NCSA of University of Illinois, which consists of XE CPU nodes and XK GPU nodes. An XE node is equipped with two AMD 6276 Interlagos processors (each containing 8 cores) and 64 GB of RAM with a peak performance of 0.314 TFLOPS. An XK node is configured with a single AMD 6276 Interlagos processor, 32 GB of RAM, and a NVIDIA K20X GPU. The GPU has a peak performance of 1.31 TFLOPS for double-precision arithmetic. This suggests that a GPU (a XK node) is intrinsically roughly 4.2 ($\approx$ 1.31/0.314) times faster than two CPUs (a XE node).

In this article, the number of CPUs refers to the number of AMD 6276 Interlagos processors, each containing eight cores, on an XE node. The number of GPUs is equal to the number of the XK nodes used in a calculation; one CPU core on an XK node is given a minuscule amount of work. The unit of speed is taken as that of a 16-way parallel calculation on two CPUs (using all of their 16 cores) on an XE node. The speed is measured in terms of the number of samples in a unit wall-clock time. A sample is, in turn, a unique $\tilde{f}$ summand in eq 10, whose number is $n_{\mathrm{GPU}}m(m-1)\,N$ for a calculation with N MC steps running on n_{GPU} GPUs or CPUs. Not all samples are of equal quality because of correlation, but we ignore this aspect and count each sample equally.

Figure 3 compares the efficiency of the redundant-walker algorithm on a GPU and on 8 CPUs (64 cores) as a function of the number of electron pairs (m) as measured by the wall time spent in each case to generate 1024 samples per CPU or GPU. Given the excellent scalability across many CPUs or many

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

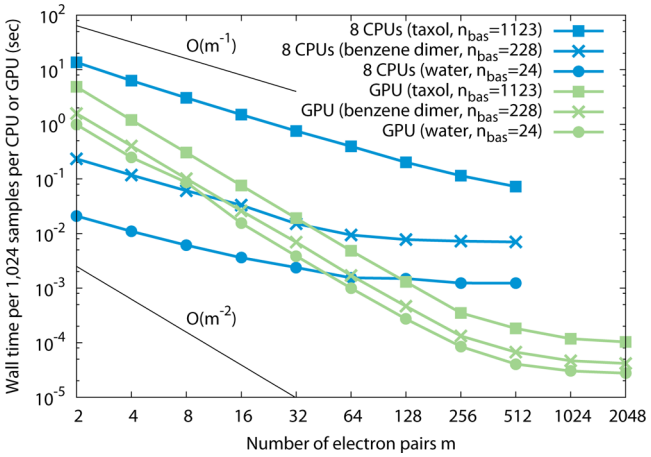


Figure 3. Efficiency of the redundant-walker algorithm on eight CPUs (64 cores) or a single GPU as a function of the number of electron pairs (m) as measured by the wall time (in seconds) spent by each CPU or GPU to sum 1024 samples in the MC-MP2/cc-pVDZ calculations of water (n_{bas} = 24), benzene dimer (n_{bas} = 228), or taxol (n_{bas} = 1123). Functions proportional to m^{-1} and m^{-2} are also shown.

GPUs (see below), this figure will hardly change with the number of CPUs or GPUs.

Focusing on the blue (CPU) curves, we find that the wall time decreases (the speedup increases) linearly with the number of pairs (m). For small problems (water and benzene dimer with n_{bas} = 24 and 228, respectively), however, the speedup slows down at m = 64 or so, whereas it continues to grow for taxol (n_{bas} = 1123) up to m = 512 or more. These are consistent with the algorithm outlined in the previous section. On CPUs, step 4 (MO construction) is the hotspot with an $O(mn^2)$ operation cost for large n (such as taxol), while the number of samplings is $O(m^2)$, leading to a net $O(m)$ increase in the sampling efficiency.[25] When n is small (such as water and benzene dimer), step 6 (Green's function construction) with an $O(m^2n)$ cost surpasses Step 4 and becomes the hotspot as m increases, whereupon both the cost and benefit of the redundant-walker algorithm grow quadratically with m, giving no net increase in sampling efficiency. This explains why the speedup stalls as m approaches or exceeds n. Nevertheless, even for water, the redundant-walker algorithm with m = 64 gives an order-of-magnitude speedup as compared with m = 2.

Shifting our attention to the green (GPU) curves, we notice that the wall time decreases (speedup increases) *quadratically* up to $m \approx 256$, then *linearly* for $256 \leq m \leq 2048$, and the saturation of speedup is delayed until $m \approx 2048$ for all cases considered (water, benzene dimer, and taxol). This occurs because all steps in Table 1 are nearly perfectly parallelized on a GPU for $m \leq 256$ and completed in near-constant wall time because as many threads (up to tens of thousands) as there are arithmetic operations are spawned and they execute these operations concurrently in an automatically load-balanced manner. This makes the wall time spent by these steps independent of m, that is, $O(1)$, while generating $O(m^2)$ samples, yielding a net $O(m^2)$ sampling efficiency increase. Only when the number of pairs (m) approaches 256 and the number of threads necessary to cover all the arithmetic tasks exceeds the number that can be run concurrently, does the wall time start to increase as $O(m)$. Thereupon, the speedup increases only linearly with m (as on CPUs) until it stalls near $m \approx 2048$, which is much greater than the saturation point on CPUs ($m \approx 64$ for water and benzene dimer). Therefore, a GPU brings about a qualitative change in its algorithmic behavior for the better. *The GPU programming paradigm of spawning a number of threads in proportion to the problem size and of load-balancing them makes the efficiency of the redundant-walker algorithm qualitatively better on a GPU.* With 2048 electron pairs, the GPU-parallel redundant-walker algorithm achieves more than 4 orders of magnitude speedup relative to the minimal two pairs even for water and benzene dimer. On CPUs, such a large number of electron pairs cannot be used meaningfully for small molecules.

5. SCALING WITH COMPUTER SIZE

The details of the speedup measurements in this work are given in the first two paragraphs of the previous section. A discussion of speedup across two disparate hardware architectures is fraught with a danger of losing a valid scientific or even technical meaning. This is particularly true if the two architectures have different degrees of technological maturity, if they mandate different algorithms and computational parameters, or if they produce slightly different numerical results or accuracy, all of which are often the case with any CPU−GPU comparison. In this work, we try to minimize these inevitable disparities by using the CPUs and GPUs on one and the same supercomputer (Blue Waters) installed at once relatively recently. A single computer program was written by the principal author and used for all of the CPU and GPU calculations of MC-MP2, although the GPU algorithm described above only applies to the latter and the number of electron pairs and the statistical uncertainty also differ between the two types of calculations.

Figure 4 compares the speedup in MC-MP2 brought to by many CPUs and by many GPUs. As already demonstrated by our group earlier,[25] MC-MP2 can be easily made scalable on many CPUs by the coarse-grain instruction-parallelism

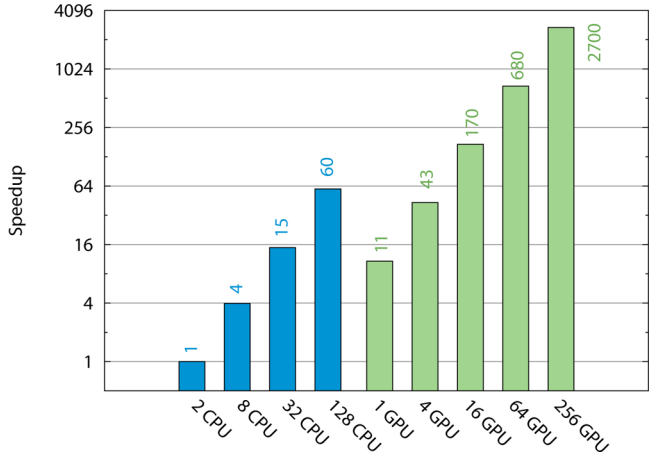


Figure 4. Speedup as a function of the number of CPUs or the number of GPUs in the redundant-walker MC-MP2/cc-pVDZ calculation of a benzene dimer (n_{bas} = 228). The speed was measured as the reciprocal of the wall time needed to sum the same number of samples, and the unit of speed was taken as that of 2 CPUs using all of their 16 cores on an XE node of the Blue Waters supercomputer, whereas one GPU refers to using one GPU and one core of a CPU on an XK node. The CPU calculations used 256 electron pairs (m), while for the the GPU calculations, m = 2048. For both CPU and GPU runs, total correlation energy and statistical uncertainty were accumulated across all processors at every 128 MC steps.

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

Table 2. Electronic Structure Calculations on GPUs

method	molecule	n_{bas}[a]	error[b]	n_{GPU}[c]	n_{CPU}[d]	speedup[e]
MC-MP2/cc-pVDZ[f]	water	24	0.4 (0.2%)	1 (dbl)	16	2.8
MC-MP2/cc-pVDZ[g]	water	24	65 (3%)	64 (sgl)	16	290
MC-MP2/cc-pVDZ[h]	water	24	0.3 (0.2%)	64 (dbl)	16	180
MC-MP2/cc-pVDZ[i]	benzene dimer	228	0.2 (0.01%)	64 (dbl)	16	690
MC-MP2/cc-pVDZ[j]	taxol	1123	9 (0.1%)	64 (dbl)	16	1600
MC-MP2/cc-pVDZ[k]	taxol	1123	...	64 (dbl)	16	2800
HF/3-21G[l]	taxol	647	1.2 (10^{-5}%)	1 (sgl)	1	63
HF/3-21G[l]	valinomycin	882	5 (10^{-5}%)	1 (sgl)	1	93
HF/3-21G[m]	taxol	647	...	4 (mix)	4	28
HF/3-21G[m]	valinomycin	882	...	4 (mix)	4	44
HF/3-21G[m]	olestra	2131	...	4 (mix)	4	182
HF/6-31G(d)[n]	taxol	1032	0	1 (dbl)	1	13.0
HF/6-31G(d)[n]	valinomycin	1350	0	1 (dbl)	1	11.1
HF/6-31G(d)[o]	taxol	1013	0.0013	1 (mix)	1	20
HF/6-31G(d)[o]	valinomycin	1350	0.0088	1 (mix)	1	31
HF/6-31G(d)[o]	olestra	3181	0.0086	1 (mix)	1	114
PW91/6-31G[p]	taxol	660	3.2	1 (sgl)	1	15
PW91/6-31G[p]	taxol	660	0.0005	1 (mix)	1	3.7
PW91/6-31G[p]	valinomycin	882	1.1	1 (sgl)	1	9.2
PW91/6-31G[p]	valinomycin	882	0.0007	1 (mix)	1	2.8
PW91/6-31G[q]	taxol	660	0.005	1 (sgl)	1	40
PW91/6-31G[q]	valinomycin	882	0.006	1 (sgl)	1	39
RI-MP2/cc-pVDZ[r]	*n*-octane	202	0.08	1 (sgl)	1	1.5
RI-MP2/cc-pVDZ[r]	*n*-docosane	538	0.3 (10^{-4}%)	1 (sgl)	1	4.3
RI-MP2/cc-pVDZ[s]	taxol	1123	0	1 (dbl)	1	7.3
RI-MP2/cc-pVDZ[s]	taxol	1123	11	1 (sgl)	1	11
RI-MP2/cc-pVDZ[s]	taxol	1123	0.8	1 (mix)	1	9.3
RI-MP2/cc-pVTZ[s]	taxol	2574	2.9	1 (mix)	1	10
RI-MP2/cc-pVDZ[s]	valinomycin	1542	0	1 (dbl)	1	7.8
RI-MP2/cc-pVDZ[s]	valinomycin	1542	16	1 (sgl)	1	14
RI-MP2/cc- pVDZ[s]	valinomycin	1542	1.9	1 (mix)	1	10
CCD/6-31G[t]	dodecahexaene	124	0.0003	1 (sgl)	8	10
CCD/6-31G[t]	dodecahexaene	124	0	1 (dbl)	8	5.0
CCD/6-31G[t]	octadecanonaene	184	0.006	1 (sgl)	8	8.6
CCD/6-31G[t]	octadecanonaene	184	0	1 (dbl)	8	4.3
CCSD(T) /POL1[u]	"spiro cation"	486	0	64 (dbl)	64	6.2

[a]The number of basis functions. [b]The greater of the absolute single-precision arithmetic error or the statistical uncertainty in the energy (accumulation at every 128 steps with block size of 128) in mE_h (in %). [c]The number of GPUs in the single-precision ("sgl"), double-precision ("dbl"), or mixed-precision ("mix") mode. Some mixed-precision calculations performed double-precision arithmetic on the CPU. [d]The number of CPU cores in the double-precision mode used in a reference calculation to define the unit of speed. [e]The speedup achieved by a calculation on n_{GPU} GPUs relative to the same calculation on n_{CPU} CPU cores. [f]This work (GPU: $m = 2048$, $N = 3.0 \times 10^4$, wall time = 1 h; CPU: $m = 256$). [g]This work (GPU: $m = 2048$, $N = 5.2 \times 10^5$, wall time = 9.5 min; CPU: $m = 256$). [h]This work (GPU: $m = 2048$, $N = 2.4 \times 10^5$, wall time = 7 min; CPU: $m = 256$). [i]This work (GPU: $m = 2048$, $N = 5.4 \times 10^7$, wall time = 40 h; CPU: $m = 512$). [j]This work (GPU: $m = 512$, $N = 4.8 \times 10^8$, wall time = 97 h; CPU: $m = 512$). [k]This work (GPU: $m = 2048$; CPU: $m = 512$). [l]Ufimtsev and Martínez.[31] The speedup is relative to the same calculation using GAMESS. [m]Ufimtsev and Martínez.[32] The speedup is relative to the same calculations using GAMESS. [n]Asadchev and Gordon.[33] [o]Titov et al.[34] The speedup is relative to the same calculations using GAMESS. [p]Yasuda.[35] The Coulomb-energy part of the density-functional calculation with the PW91 functional was accelerated on a GPU. [q]Yasuda.[36] The exchange-correlation-energy part of the density-functional calculation with the PW91 functional was accelerated on a GPU. [r]Vogt et al.[37] RI-MP2 stands for the resolution-of-identity MP2 method. The error for *n*-octane is the average over a series of calculations while rotating the central bond angle. The error for *n*-docosane was not reported; the value of 0.3 mE_h is the average error across the entire alkane series studied by them. [s]Olivares-Amaya et al.[38] [t]DePrince and Hammond.[41] CCD stands for the coupled-cluster doubles method. [u]Ma et al.[42] Only the noniterative triples part of the regularized coupled-cluster singles and doubles with noniterative triples calculation was accelerated by GPUs. "Spiro cation" stands for 5,5′ (4*H*,4*H*′)-spirobi[cyclopenta[c]pyrrole] 2,2′,6,6′-tetrahydro cation.

described above. Relative to the 2-CPU (16-core) execution, the 128-CPU (1024-core) calculation achieves 93% scalability, though the rate goes down to 67% when measured against the serial (one-core) execution (not shown), as the latter involves no interprocessor communication. In all calculations shown in this figure, all processors synchronize and accumulate the energy increment and statistical uncertainty at every 128 MC steps, which is the only hindrance for higher scalability, which can be made arbitrarily less frequent.

The figure also indicates that the 1-GPU calculation is 11 times faster than the 2-CPU (16-core) calculation. This speedup is greater than the intrinsic performance ratio of 1 GPU (a XK node) to 2 CPUs (a XE node) of approximately 4.2 inferred from hardware specification given by NCSA. This is likely due to the fact that on a GPU, as many as 2048 electron

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

pairs can be used meaningfully for the benzene dimer, while CPUs can harness only up to 256 pairs after which the sampling efficiency no longer increases. In other words, the GPU and CPU calculations in this figure differ not only in hardware, but also in calculation parameters.

The MC-MP2 method is highly scalable in the coarse-grain parallel manner on *many* GPUs. The scalability of the 256-GPU calculation is 99% relative to the 1-GPU execution. As a consequence, the former (256-GPU) achieves a 46-fold speedup relative to the 128-CPU (1024-core) calculation or a 2700-fold speedup from the 2-CPU (16-core) calculation. Recall that on CPUs, the MC integrations are coarse-grain parallelized, but the redundant-walker algorithm (steps 2, 3, 4, 6, 7, and 9 in Figure 2) is still serial; on GPUs, both are parallel. The redundant-walker MC-MP2 method is, therefore, particularly well-suited for an efficient multilevel-parallel execution on many GPUs.

Table 2 compares some of the previous *ab initio* electronic structure calculations and our MC-MP2 calculations on GPUs. In a 2008 summary[31] of the use of GPU in computational quantum chemistry, Ufimtsev and Martínez quoted their HF/3-21G calculations of taxol (647 basis functions) and valinomycin (882 basis functions) on one GPU in the single-precision mode. They achieved remarkable 63- and 93-fold speedups, respectively, relative to the corresponding serial calculations using GAMESS. The same authors[32] extended these calculations to multiple GPUs with mixed single/double-precision arithmetic and reported a 182-fold speedup in the HF/3-21G calculation of olestra (2131 basis functions) going from 4 CPU cores to 4 GPUs. Asadchev and Gordon[33] implemented a GPU-parallel Rys-quadrature algorithm for two-electron integrals and achieved a one-order-of-magnitude speedup in the HF calculations of taxol and valinomycin with a range of basis sets relative to an implementation of a similar highly optimized algorithm for a CPU. Titov et al.[34] also performed a GPU-based HF calculation with a larger basis set, achieving a 114-fold speedup in the 6-31G(d) calculation of olestra (3181 basis functions) from one CPU core to one GPU.

Yasuda[35,36] accelerated the Coulomb and exchange-correlation energy evaluation in density-functional calculations with one GPU (single or mixed precision). He reported a 15-fold speedup in the former for taxol with the 6-31G basis set (660 basis functions) and a 40-fold speedup in the latter for the same system.

In 2008, Vogt et al.[37] recorded a speedup by a factor of 4.3 in their resolution-of-identity (RI) MP2 calculations on one GPU (single precision) for *n*-docosane (538 basis functions), whereas the performance for a smaller *n*-octane was less good. In 2010, the same group led by Olivares-Amaya[38] improved the speedup by up to a factor of 10 for taxol with the cc-pVTZ basis set (2574 basis functions) or a factor of 14 for valinomycin with the cc-pVDZ basis set (1542 basis functions), both on one GPU in the single-, double-, or mixed-precision mode. In 2016, Tomlinson et al.[39] reported a modest GPU speedup of an otherwise extremely efficient implementation of the conventional MP2 method. In the same year, Song and Martínez[40] reported a cubic-scaling approximation of MP2 (atomic-orbital-based scaled-opposite-spin MP2 using tensor hypercontraction) that can handle a large molecule with 1600 basis functions. However, the information about the GPU speedup was not given.

More recently, DePrince and Hammond[41] made a thorough performance comparison of various implementations of the coupled-cluster doubles (CCD) method running on up to 2 CPUs (8 cores) or one GPU. Their GPU code achieved the impressive speedup by a factor of 4−5 in the double-precision mode and 8−10 in the single-precision mode as compared with double-precision multithreaded execution using 8 CPU cores. Ma et al.[42] accelerated the noniterative triples part of their regularized coupled-cluster singles, doubles, and noniterative triples method on as many as 64 GPUs (double precision). For a molecule with 486 basis functions, the time for noniterative triples calculation was compressed by a factor of 6.2 relative to that on 64 CPU cores, which translates to at least an impressive 390-fold speedup from one CPU core to 64 GPUs.

Our MC-MP2 method, with its stochastic algorithm specifically designed for scalability, is measured to achieve a 180-fold speedup for water (24 basis functions), a 690-fold speedup for benzene dimer (228 basis functions), and a 2800-fold speedup for taxol (1123 basis functions), all going from 2 CPUs (16 cores) with 256 or 512 electron pairs to 64 GPUs (double precision) with 2048 electron pairs. The speedup increases with the number of basis functions, which is consistent with the well-known performance increase of the CUBLAS library with problem size.[43] These massive speedups are the result of a mutual enhancement of GPU-parallelism and the redundant-walker convergence acceleration as well as of the use of many GPUs as opposed to just one. When the same number (512) of electron pairs is used, the speedup for taxol is 1600 as opposed to 2800. The speedup for water from 2 CPUs (16 cores) to one GPU is measured to be 2.8. The corresponding value from a serial (one-core) execution to one GPU is 29 (not shown) and much less than $16 \times 2.8 = 45$ because parallel scaling on CPUs is less than perfect.

The speedup obtained with the single-precision mode of GPUs (a factor of 290 from 2 CPUs to 64 GPUs for water) is roughly 50% greater than the speedup with the double-precision mode. However, the MC-MP2 energy from the former calculation suffers from a large absolute error of 65 mE_h from the deterministic MP2 energy, which is also 80 times greater than the statistical uncertainty of 0.8 mE_h. The statistical uncertainty itself is much greater in the single-precision mode than in the double-precision mode (0.3 mE_h) despite twice as many MC steps in the former calculation as in the latter. Both of these are clearly caused by a rapid accumulation of round-off errors during long summations, which are inevitable in MC-MP2. We, therefore, conclude that the small speedup by the single-precision arithmetic does not outweigh the excessive precision error[44] caused by it in MC-MP2 on GPUs. In deterministic calculations, the single-precision arithmetic error is reported to be typically 10^{-4} to 10^{-5}% of the energy.

6. SCALING WITH SYSTEM SIZE

In our previous study[20] we showed that the cost of MC-MP2 *per MC step* (wall time T_0) grows linearly with system's spatial size (which may be measured by $n_{ele} \propto n_{bas}$) for a few small molecules:

$$T_0 \propto n_{ele} \propto n_{bas} \tag{29}$$

In Figure 5, we confirm that the cost per MC step of MC-MP2 *with the redundant-walker algorithm* is also asymptotically proportional to the system size for a wider range of molecules including one with 1123 basis functions. This can be understood by noting that step 6 (Green's function calculation) of Table 1 with an $O(m^2n)$ operation cost is the hotspot of each MC step if a sufficiently large number (m) of electron pairs can

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

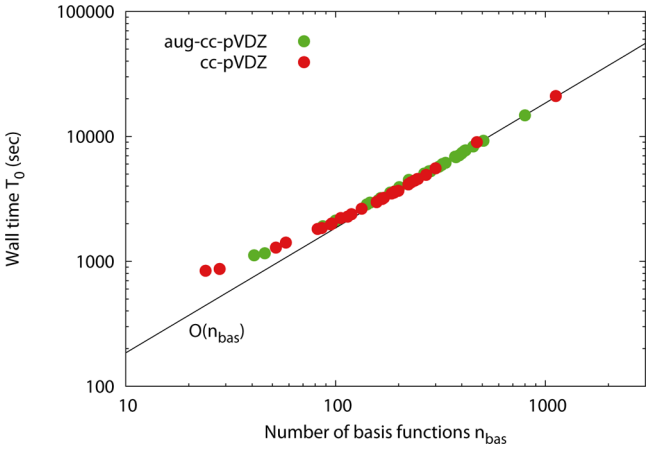


Figure 5. Wall time T_0 (in seconds) required for 16 384 MC steps of the MC-MP2/cc-pVDZ (MC-MP2/aug-cc-pVDZ) calculations of 34 (33) molecules of different sizes as a function of the number of basis functions (n_{bas}) on a single GPU. The largest molecules for the cc-pVDZ and aug-cc-pVDZ calculations are taxol (n_{bas} = 1123) and tetrahydrocannabinol (n_{bas} = 799), respectively, while the smallest is water (n_{bas} = 24 or 41). The number of electron pairs (m) was 512. A function proportional to n_{bas} is plotted as a solid line. The wall time is also asymptotically linear with the number of electrons (not shown).

be used. It is, however, wrong to call MC-MP2 a "linear-scaling" method on this basis, because the number of MC steps (N_{fin}) to reach a given accuracy may also depend on the system size. In fact, it has been our experience that the number indeed grows with system size.

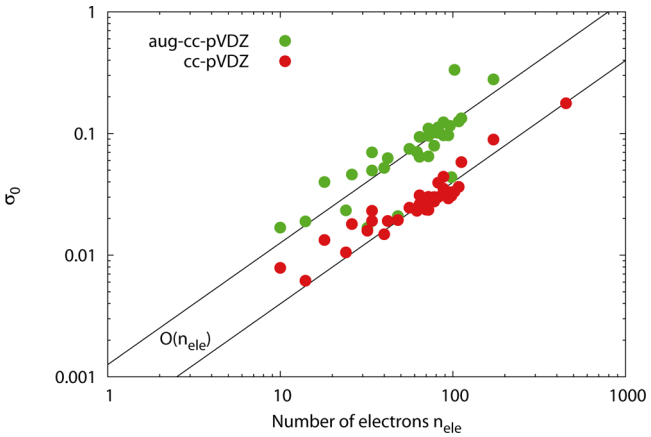


Figure 6. Relative statistical uncertainties (σ_0) in the MC-MP2/cc-pVDZ (MC-MP2/aug-cc-pVDZ) correlation energies of 34 (33) molecules of different sizes as a function of the number of electrons (n_{ele}). The largest molecules for the cc-pVDZ and aug-cc-pVDZ calculations are taxol (n_{ele} = 452) and tetrahydrocannabinol (n_{ele} = 172), respectively, while the smallest is water (n_{ele} = 10). The number of MC steps (N) was 16 384, and the number of electron pairs (m) was 512. The statistical uncertainties were computed with a block size (N_b) of 128. Functions proportional to n_{ele} are plotted as solid lines.

Figure 6 quantifies this dependence. Let σ_0 be the relative statistical uncertainty after a certain number of MC steps (N_0, which is equal to 16 384 in this figure) defined by

$$\sigma_0 = \frac{\sigma}{E^{(2)}} \tag{30}$$

where σ is evaluated by eq 11 and appropriately blocked so as not to be underestimated. This statistical uncertainty σ, in turn, falls off accurately in proportion to $N^{-1/2}$ in MC-MP2, as in most other QMC methods.

Figure 6 testifies that the relative statistical uncertainty, σ_0, is proportional to the system's spatial size or n_{ele}:

$$\sigma_0 \propto n_{ele} \tag{31}$$

The number of MC steps (N_{fin}) required for the relative statistical uncertainty to reach a given tolerance σ_{fin} bears the relation,

$$\sigma_{fin} = \sigma_0 \left(\frac{N_0}{N_{fin}} \right)^{1/2} \tag{32}$$

The total cost (total wall time T_{fin}) is

$$T_{fin} = N_{fin} T_0 \tag{33}$$

Combining eqs(29, 31, 32, and 33, we conclude

$$T_{fin} = N_{fin} T_0 \propto \sigma_0^2 T_0 \propto n_{ele}^3 \tag{34}$$

Hence, the cost of MC-MP2 to reach a given relative statistical uncertainty increases as $O(n_{ele}^3)$ or cubically with the system's spatial size. Figure 7 is a direct observation of this cubic scaling

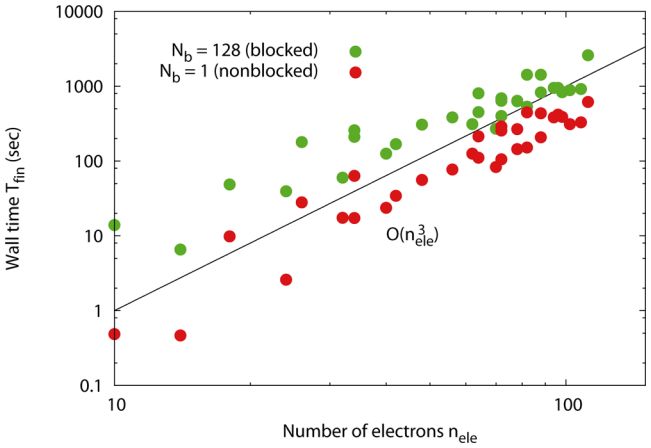


Figure 7. Wall time T_{fin} (in seconds) required for the MC-MP2/cc-pVDZ calculations to reach the relative statistical uncertainty (σ_0) of 0.1 as a function of the number of electrons (n_{ele}) on a single GPU. The statistical uncertainty was computed without or with blocking using a block size (N_b) of 128. The number of electron pairs (m) was 512 in all cases. A function proportional to n_{ele}^3 is plotted as a solid line.

in the case of the cc-pVDZ-basis-set calculations. These calculations were carried out on a single GPU with the same number (512) of electron pairs, and hence the observed scaling is intrinsic to MC-MP2 and unaffected by these other acceleration measures (although a proportional increase in the number of electron pairs with system size may also be a sensible protocol, possibly leading to an even more favorable cost scaling). This is a large improvement over the $O(n_{ele}^5)$ scaling of the conventional, deterministic MP2; MC-MP2 is expected to become more efficient than the latter for a sufficiently large molecule.

Next, we consider the scaling of the MC-MP2 cost with n_{bas} for a molecule of a fixed size (in our test, the benzene

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

molecule). Figure 8 shows that the relative statistical uncertainty σ_0 grows more slowly with a basis-set extension

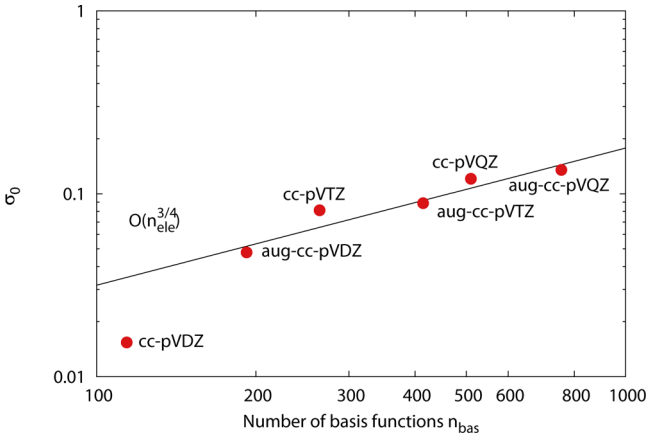


Figure 8. Relative statistical uncertainties (σ_0) in the MC-MP2 correlation energies of benzene as a function of the number of basis functions (n_{bas}). The number of MC steps (N) was 16 384, and the number of electron pairs (m) was 512. The statistical uncertainties were computed with a block size (N_b) of 128. A function proportional to $n_{ele}^{3/4}$ is plotted as a solid line.

of the same molecule than with a simultaneous increase in both spatial and basis-set sizes considered above. The σ_0 curve falls on $n_{bas}^{0.75}$, that is,

$$\sigma_0 \propto n_{bas}^{0.75} \tag{35}$$

for a fixed n_{ele}. This in conjunction with eqs 29, 32, and 33; we find

$$T_{fin} = N_{fin} T_0 \propto \sigma_0^2 T_0 \propto n_{bas}^{2.5} \tag{36}$$

Therefore, the cost of MC-MP2 to reach a target relative statistical uncertainty is observed to increase as $O(n_{bas}^{2.5})$ for a basis-set extension for a fixed spatial size. This may be compared with $O(n_{bas}^4)$ scaling of the deterministic MP2 for a basis-set extension (this quartic scaling arises from the most expensive quarter AO-to-MO integral transformation).[26]

We note that the prefactor multiplying the cost function of MC-MP2 is extremely large, making the crossover point at which (sub)cubic-scaling MC-MP2 becomes actually faster (for an acceptable statistical uncertainty) than the conventional quintic-scaling MP2 far into large problems. For instance, a MC-MP2 calculation of the water molecule takes as long as 1 h to achieve a statistical uncertainty of 0.4 mE_h or 0.2% of the correlation energy on one GPU (see Table 2) . For taxol (1123 basis functions), the correlation energy with a statistical uncertainty of 9 mE_h (0.1%) is obtained in 97 h on 64 GPUs. This may be compared with a domain-based local pair natural orbital (DLPNO) MP2 calculation[45] of sildenafil (1209 basis functions), which completes in 1.2 h on one CPU core. It might, however, be possible to combine the two (local-correlation and stochastic) algorithms for mutual enhancement.

7. CONCLUSION

Earlier, we introduced a stochastic algorithm of MP2, called MC-MP2.[6,7] It can compute energy differences directly,[20] does not suffer from the sign problem or require a fixed-node approximation, and is rigorously (diagrammatically) size-consistent.[21] It can also be systematically improved toward exactness by raising the perturbation order[22] and increasing the basis-set size. It does not require two-electron (or any other) integrals either in the AO or MO basis, can incorporate a correlation factor easily to become explicitly correlated,[23] and is linear scaling in cost *per MC step* with system size.[20] It parallelizes easily and efficiently on many CPUs.[25]

In this work, we demonstrated that MC-MP2 is scalable with system size on the *per accuracy* basis. The total cost to achieve a given relative statistical uncertainty increases only as $O(n_{ele}^3)$, where n_{ele} is the number of electrons and is proportional to the system's spatial size (and the number of basis functions, n_{bas}). It is also shown to increase even more favorably as $O(n_{bas}^{2.5})$, where n_{bas} is the number of basis functions in a given molecule of a fixed size. These are a vast improvement over the $O(n_{ele}^5)$ or $O(n_{bas}^4)$ cost increase of the conventional implementation[26] of deterministic MP2. However, the prefactor of the cost function of MC-MP2 is extremely large as compared with that of deterministic MP2.

Here, we also showed numerically that MC-MP2 is scalable not only on many CPUs by virtue of its embarrassingly parallel algorithm by design, but also on many GPUs as it lends itself to multilevel parallelism. Fine-grain parallelism of the redundant-walker convergence-acceleration algorithm is shown to be particularly effective on a GPU, as it automatically load-balances computational tasks associated with many electron pairs on proportionally many threads spawned also automatically. As a result, on a GPU, the redundant-walker algorithm improves the sampling efficiency by a factor of $O(m^2)$, where m is the number of electron pairs, whereas on a CPU, only by a factor of $O(m)$. A MC-MP2 calculation of a benzene dimer executes 11 times faster on a GPU than on 2 CPUs (using all of their 16 cores) or 2700 times faster on 256 GPUs than on 2 CPUs (16 cores), partly owing to the fact that the redundant-walker algorithm can use 2048 electron pairs on a GPU, while its performance boost stalls after 256 pairs or so on CPUs.

In this work, therefore, we have established parallel scalability of MC-MP2 on thousands of CPU cores and hundreds of GPUs as well as its $O(n_{ele}^3)$ and $O(n_{bas}^{2.5})$ cost scaling with problem size. The latter is thanks to the sparse integration achieved by the Monte Carlo method with a weight function containing a Coulomb singularity,[6] which more heavily samples electron pairs with shorter interelectronic distances. Since such close electron pairs occur anywhere in space relative to fixed nuclear positions, their biased sampling is not possible with simple product grids, but is straightforward with the Metropolis algorithm, which is the essence of MC-MP2. A trade-off for the favorable cost scaling is the statistical uncertainty, although there are no systematic errors. Therefore, the unique features of MC-MP2 (such as reduced cost scaling) are hardly imitable by other deterministic algorithms of MP2, barring ones using a sparse grid for Coulomb integrals (we are not aware of any). Also, MC-MP2 samples in real space and is, therefore, rather different from the methods[8−18] that stochastically sample in a Hilbert space. We believe that it is most closely related to variational Monte Carlo, but with no variational parameter because it is perturbative.

A key question concerning the long-term viability of MC-MP2 then has to do with its overall, absolute cost in comparison with other reduced-scaling MP2 methods. MP2 (or any other electronic structure methods) is well-known[46−49] to be made linear scaling in cost[45,50−57] for spatially extended insulating systems by exploiting locality of correlation therein. There are other mechanisms by which the cost scaling can be

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

reduced such as in the scaled-opposite-spin MP2 of Jung et al.,[58] the tensor hypercontraction MP2 of Martínez and coworkers,[59,60] and their combination.[40] The latter methods, however, tend to have systematic errors and/or have nominally one-rank higher scaling (quartic) than cubic- or subcubic-scaling MC-MP2. Therefore, it seems more important to ask if MC-MP2 can have a unique domain of applicability in the presence of linear-scaling local MP2.[45,50−57] Because the linear-scaling local-correlation and the (sub)cubic-scaling stochastic algorithms can be combined with each other for mutual enhancement, the question is whether the latter (MC-MP2) with a huge prefactor on its (sub)cubic cost function can ever be faster than the former (MP2) with a minuscule prefactor on its quintic cost function for a local small correlated domain or for a larger problem that is indivisible into localized subproblems.

We believe that the comparison should be made to the whole classes of methods or their underlying physics by measurements of performance of their best implementations. The verdict should depend on the properties computed (since the acceptable statistical uncertainty and thus the prefactor of the cost function differ greatly, depending on whether one is computing total energy or ionization/electron-attachment energies), the rank of theory (how the comparative performance changes with the perturbation order, basis set, explicit correlation, etc.), the system characteristics (systems with long-range correlation or charge transfer, and heavy elements as well as large basis-set calculations may favor MC-MP2), the computers used (a large number of CPUs and complex, heterogeneous hardware architectures favor MC-MP2), and even the future trend of the computers and human factors (such as the cost of porting and tuning codes from an older parallel computer to a new one). This work serves as a step toward such rigorous comparisons.

AUTHOR INFORMATION

Corresponding Author

*E-mail: sohirata@illinois.edu.

Funding

This work has been supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences under Award No. DE-FG02-11ER16211. It is also part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (Awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana−Champaign and its National Center for Supercomputing Applications. S.H. has also been supported by CREST, Japan Science and Technology Agency

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank Dr. Soohaeng Y. Willow for providing the program reported in ref 25.

REFERENCES

(1) Ceperley, D. M.; Alder, B. J. *Phys. Rev. Lett.* **1980**, *45*, 566.
(2) Hammond, B. L.; Lester, W. A.; Reynolds, P. J. *Monte Carlo Methods in Ab Initio Quantum Chemistry*; World Scientific: Singapore, 1994.
(3) Lüchow, A.; Anderson, J. B. *Annu. Rev. Phys. Chem.* **2000**, *51*, 501.
(4) Foulkes, W. M. C.; Mitas, L.; Needs, R. J.; Rajagopal, G. *Rev. Mod. Phys.* **2001**, *73*, 33.
(5) Kolorenč, J.; Mitas, L. *Rep. Prog. Phys.* **2011**, *74*, 026502.
(6) Willow, S. Y.; Kim, K. S.; Hirata, S. *J. Chem. Phys.* **2012**, *137*, 204122.
(7) Hirata, S.; He, X.; Hermes, M. R.; Willow, S. Y. *J. Phys. Chem. A* **2014**, *118*, 655.
(8) Zhang, S.; Krakauer, H. *Phys. Rev. Lett.* **2003**, *90*, 136401.
(9) Thom, A.; Alavi, A. *Phys. Rev. Lett.* **2007**, *99*, 143001.
(10) Ohtsuka, Y.; Nagase, S. *Chem. Phys. Lett.* **2008**, *463*, 431.
(11) Booth, G. H.; Thom, A. J. W.; Alavi, A. *J. Chem. Phys.* **2009**, *131*, 054106.
(12) Kozik, E.; Van Houcke, K.; Gull, E.; Pollet, L.; Prokof'ev, N.; Svistunov, B.; Troyer, M. *Europhys. Lett.* **2010**, *90*, 10004.
(13) Cleland, D.; Booth, G. H.; Alavi, A. *J. Chem. Phys.* **2010**, *132*, 041103.
(14) Petruzielo, F. R.; Holmes, A. A.; Changlani, H. J.; Nightingale, M. P.; Umrigar, C. J. *Phys. Rev. Lett.* **2012**, *109*, 230201.
(15) Shepherd, J. J.; Booth, G. H.; Alavi, A. *J. Chem. Phys.* **2012**, *136*, 244101.
(16) Booth, G. H.; Chan, G. K. L. *J. Chem. Phys.* **2012**, *137*, 191102.
(17) Neuhauser, D.; Rabani, E.; Baer, R. *J. Chem. Theory Comput.* **2013**, *9*, 24.
(18) Ten-no, S. *J. Chem. Phys.* **2013**, *138*, 164126.
(19) Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. *J. Chem. Phys.* **1953**, *21*, 1087.
(20) Willow, S. Y.; Kim, K. S.; Hirata, S. *J. Chem. Phys.* **2013**, *138*, 164111.
(21) Willow, S. Y.; Kim, K. S.; Hirata, S. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *90*, 201110.
(22) Willow, S. Y.; Hirata, S. *J. Chem. Phys.* **2014**, *140*, 024111.
(23) Willow, S. Y.; Zhang, J. M.; Valeev, E. F.; Hirata, S. *J. Chem. Phys.* **2014**, *140*, 031101.
(24) Johnson, C. M.; Doran, A. E.; Zhang, J. M.; Valeev, E. F.; Hirata, S. 2016, unpublished.
(25) Willow, S. Y.; Hermes, M. R.; Kim, K. S.; Hirata, S. *J. Chem. Theory Comput.* **2013**, *9*, 4396.
(26) Head-Gordon, M.; Pople, J. A.; Frisch, M. J. *Chem. Phys. Lett.* **1988**, *153*, 503.
(27) Almlöf, J. *Chem. Phys. Lett.* **1991**, *181*, 319.
(28) Kronrod, A. *Nodes and weights of quadrature formulas: Sixteen-place tables*; Consultants Bureau, 1965.
(29) Obara, S.; Saika, A. *J. Chem. Phys.* **1986**, *84*, 3963.
(30) Flyvbjerg, H.; Petersen, H. G. *J. Chem. Phys.* **1989**, *91*, 461.
(31) Ufimtsev, I. S.; Martínez, T. *Comput. Sci. Eng.* **2008**, *10*, 26.
(32) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 2619.
(33) Asadchev, A.; Gordon, M. S. *J. Chem. Theory Comput.* **2012**, *8*, 4166.
(34) Titov, A. V.; Ufimtsev, I. S.; Luehr, N.; Martinez, T. J. *J. Chem. Theory Comput.* **2013**, *9*, 213.
(35) Yasuda, K. *J. Comput. Chem.* **2008**, *29*, 334.
(36) Yasuda, K. *J. Chem. Theory Comput.* **2008**, *4*, 1230.
(37) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. *J. Phys. Chem. A* **2008**, *112*, 2049.
(38) Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L.; Shao, Y.; Aspuru-Guzik, A. *J. Chem. Theory Comput.* **2010**, *6*, 135.
(39) Tomlinson, D. G.; Asadchev, A.; Gordon, M. S. *J. Comput. Chem.* **2016**, *37*, 1274.
(40) Song, C.; Martínez, T. J. *J. Chem. Phys.* **2016**, *144*, 174111.
(41) DePrince, A. E., III; Hammond, J. R. *J. Chem. Theory Comput.* **2011**, *7*, 1287.
(42) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. *J. Chem. Theory Comput.* **2011**, *7*, 1316.
(43) Leang, S. S.; Rendell, A. P.; Gordon, M. S. *J. Chem. Theory Comput.* **2014**, *10*, 908.
(44) Ceperley, D. M.; Bernu, B. *J. Chem. Phys.* **1988**, *89*, 6316.
(45) Pinski, P.; Riplinger, C.; Valeev, E. F.; Neese, F. *J. Chem. Phys.* **2015**, *143*, 034108.

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832

(46) Sæbø, S.; Pulay, P. *Chem. Phys. Lett.* **1985**, *113*, 13.
(47) Yang, W. *Phys. Rev. Lett.* **1991**, *66*, 1438.
(48) Saebo, S.; Pulay, P. *Annu. Rev. Phys. Chem.* **1993**, *44*, 213.
(49) Kohn, W. *Int. J. Quantum Chem.* **1995**, *56*, 229.
(50) Pulay, P.; Saebø, S. *Theor. Chim. Acta* **1986**, *69*, 357.
(51) Hetzer, G.; Pulay, P.; Werner, H. J. *Chem. Phys. Lett.* **1998**, *290*, 143.
(52) Ayala, P. Y.; Scuseria, G. E. *J. Chem. Phys.* **1999**, *110*, 3660.
(53) Schütz, M.; Hetzer, G.; Werner, H. J. *J. Chem. Phys.* **1999**, *111*, 5691.
(54) Lee, M. S.; Maslen, P. E.; Head-Gordon, M. *J. Chem. Phys.* **2000**, *112*, 3592.
(55) Saebø, S.; Pulay, P. *J. Chem. Phys.* **2001**, *115*, 3975.
(56) Werner, H. J.; Manby, F. R.; Knowles, P. J. *J. Chem. Phys.* **2003**, *118*, 8149.
(57) Baudin, P.; Ettenhuber, P.; Reine, S.; Kristensen, K.; Kjærgaard, T. *J. Chem. Phys.* **2016**, *144*, 054102.
(58) Jung, Y. S.; Lochan, R. C.; Dutoi, A. D.; Head-Gordon, M. *J. Chem. Phys.* **2004**, *121*, 9793.
(59) Hohenstein, E. G.; Parrish, R. M.; Martínez, T. J. *J. Chem. Phys.* **2012**, *137*, 044103.
(60) Schumacher, S. I. L. K.; Hohenstein, E. G.; Parrish, R. M.; Wang, L. P.; Martínez, T. J. *J. Chem. Theory Comput.* **2015**, *11*, 3042.

NOTE ADDED AFTER ASAP PUBLICATION

This paper was published ASAP on September 21, 2016, with errors in Table 2 and Figure 8. The corrected version was reposted on September 26, 2016.

DOI: 10.1021/acs.jctc.6b00588
J. Chem. Theory Comput. 2016, 12, 4821−4832