An Efficient RI-MP2 Algorithm for Distributed Many-GPU Architectures

Calum Snowdon[†] and Giuseppe M. J. Barca[*,‡]

†School of Computing, Australian National University, Canberra, Australia
‡School of Computing and Information Systems, University of Melbourne, Melbourne, Australia

E-mail: giuseppe.barca@unimelb.edu.au

Abstract

Second-order Møller-Plesset perturbation theory (MP2) using the Resolution of the Identity approximation (RI-MP2) is a widely used method for computing molecular energies beyond the Hartree-Fock mean-field approximation. However, its high computational cost and lack of efficient algorithms for modern supercomputing architectures limit its applicability to large molecules. In this paper, we present the first distributed-memory many-GPU RI-MP2 algorithm explicitly designed to utilize hundreds of GPU accelerators for every step of the computation. Our novel algorithm achieves near-peak performance on GPU-based supercomputers through the development of a distributed memory algorithm for forming RI-MP2 intermediate tensors with zero inter-node communication, except for a single $\mathcal{O}(N^2)$ asynchronous broadcast, and a distributed memory algorithm for the $\mathcal{O}(N^5)$ energy reduction step, capable of sustaining near-peak performance on clusters with several hundred GPUs. Comparative analysis shows our implementation outperforms state-of-the-art quantum chemistry software by over 3.5 times in speed while achieving an eightfold reduction in computational power consumption. Benchmarking on the Perlmutter supercomputer, our algorithm achieves 11.8 PFLOP/s (83% of peak performance) performing and the RI-MP2 energy calculation on a 314-water cluster with 7,850 primary and 30,144 auxiliary basis functions in 4 minutes on 180 nodes and 720 A100 GPUs. This performance represents a substantial improvement over traditional CPU-based methods, demonstrating significant time-to-solution and power consumption benefits of leveraging modern GPU-accelerated computing environments for quantum chemistry calculations.

1 Introduction

Second-order Möller-Plesset perturbation theory (MP2) has long been a prominent method for calculating molecular energies, incorporating dynamic correlation effects beyond the Hartree-Fock (HF) mean-field approximation.[1] The capabilities and limitations of MP2 are well-characterized. MP2 generally provides accurate equilibrium energies for large-gap, closed-shell systems,[2,3] though it tends to overestimate binding in dispersion-dominated systems.[3–10] Furthermore, MP2 often fails to accurately predict the energetics and geometry of transition metal and open-shell systems compared to more advanced methods.[11–14] Due to its $\mathcal{O}(N^5)$ computational scaling, MP2 becomes quickly computationally impractical with growing molecular sizes.

Despite the advent of less computationally demanding density functionals[14–20] within the Kohn-Sham density functional theory (KS-DFT) framework,[21] which have partly eclipsed the utility of traditional MP2 calcu-

lations, MP2 remains foundational for more accurate quantum chemical methods. Various approaches have been explored to enhance MP2's performance. Orbital Optimized MP2 (OOMP2) yields improved energetics for spin-unrestricted reference wave functions.[22–25] Spin-Component-Scaled MP2 (SCS-MP2) provides major improvements in accuracy and robustness for closed-shell systems.[12,22,26–30] Regularized MP2 methods,[31–35] such as κ-MP2,[24,36] significantly improve accuracy for noncovalent interactions and radical systems. Double-hybrid (DH) density functionals,[19,37–40] combining hybrid meta-GGA exchange and correlation with second-order Görling–Levy perturbation theory, are considered the most accurate functionals in the KS-DFT framework.[41–48]

However, the practical application of these methods to large molecules is hindered by the steep $\mathcal{O}(N^5)$ computational scaling of their underlying MP2 calculations. Consequently, significant research effort has focused designing faster MP2 algorithms.[49–63]

Each of these methods comes with its own approximations and errors compared to the exact, approximation-less application of MP2 theory. Without a doubt, one of the most successful method for accelerating MP2 calculations is using resolution-of-the-identity (RI)[64–66] to yield the RI-MP2 method. Extensive benchmarks with correlation consistent basis sets have demonstrated that the RI approximation introduces errors smaller than the primary basis set incompleteness errors, making them negligible relative to the original MP2 method.[67]

The main advantage of RI-MP2 is that its primary bottlenecks can be reduced to a series of matrix multiplications. Compared to the original semi-direct MP2 algorithm,[68] which is computationally inefficient due to the need for repeated disk- and memory-bound operations, this feature allows a well-implemented RI-MP2 algorithm to run close to the theoretical floating-point peak performance of the underlying computing system, achieving $\mathcal{O}(10\times)$ speeds-ups compared to traditional MP2.

RI-MP2's algorithmic features have made it suitable for high-performance computing (HPC) implementations. The first large-scale parallel RI-MP2 implementation, developed by Bernholdt and Harrison in 1996, used 70 nodes of the Kendall Square Research KSR-2 to study the binding of K+ to 12-crown-4 ether.[69] Since then, the advent of petascale parallel computers has spurred multiple parallel MP2 implementations.[70–79] A notable achievement was Katouda *et al.*'s 2016 algorithm, which computed the RI-MP2/cc-pVTZ energy of a molecule with 240 atoms on the K supercomputer in under 5 minutes.[75] In 2017, Kjærgaard *et al.* developed a parallel RI-MP2 algorithm based on a divide-expand-consolidate scheme, enabling calculations of supramolecular wires with 2,440 atoms and 6,800 electrons on the Titan supercomputer.[80] Using fragmentation methods, the largest RI-MP2 calculations were performed by some of us, modelling the energetics of a peptide with 180,000 electrons and 45,000 atoms[81] and a crystal lattice cut of the ionic liquid 1-ethyl-3-methylimidazolium tetrafluoroborate with 623,016 electrons and 146,592 atoms.[82]

While fragmentation and local-RI-MP2 methods offer fast and efficient approaches for large molecular systems, there remains a critical need for approximation-free RI-MP2 calculations. This is particularly true for generating high-accuracy reference data needed to benchmark the approximations inherent in these lower-scaling MP2 approaches. Moreover, full-system RI-MP2 calculations serve as a robust starting point for more accurate coupled-cluster methods (with or without the RI approximation) and as a key component of the aforementioned double-hybrid density functional approaches. This article addresses the demand for efficient, full-system RI-MP2 calculations by presenting a distributed, many-Graphics Processing Units (GPU) algorithm optimized for such tasks.

The shift from petascale to exascale computing has been primarily driven by the integration of GPUs as the leading source of computational floating-point throughput. To fully capitalize on exascale hardware, quantum chemistry programs must evolve to accommodate both hardware and software advancements. The effort to harness the computational power of GPUs in quantum chemistry began with Yasuda's pi-

oneering work on accelerating two-electron integrals.[84] Since then, significant progress has been made, leading to the development of GPU-accelerated algorithms for Hartree-Fock (HF) and RI-MP2.[85–94]

However, exploiting GPUs fully involves many complexities, as exemplified by Katouda *et al.*'s work,[75] the only known paper describing a distributed many-GPU RI-MP2 routine. They achieved 514 TFLOP/s, 9.3% of peak on 1349 nodes of the TSUBAME 2.5 supercomputer, compared to 42% peak performance for their CPU-only implementation in RI-MP2 energy calculation. This discrepancy is due to GPUs' superior computational density, requiring algorithms with a higher FLOP-to-data transfer ratio to avoid data transfer overheads. Challenges such as Host $\leftrightarrow$ Device data transfer and asynchronous GPU scheduling also complicate GPU utilization.

In contrast to Katouda *et al.*, we present a novel RI-MP2 algorithm designed from the ground up for near-peak performance on GPU-based supercomputers. The novel contributions from this work are:

- A distributed memory algorithm for the formation of the RI-MP2 intermediate tensors with zero inter-node communication besides a single $\mathcal{O}(N^2)$ asynchronous broadcast.
- A distributed memory algorithm for the $\mathcal{O}(N^5)$ energy reduction step of the RI-MP2 algorithm which is capable of sustaining near-peak performance on clusters with several hundred GPUs.

All distributed memory operations are implemented using the Message Passing Interface (MPI).

A noteworthy result of our work is the RI-MP2 computation of a cluster of 314 water molecules (942 atoms, 2512 electrons) at the cc-pVDZ/cc-pVDZ-RIFIT level on 180 nodes of the Perlmutter supercomputer; achieving 11.8 PFLOP/s (83% of peak performance) over a 4-minute computation.

The paper is organized as follows: In Section 2 MP2 theory and the RI approximation will be introduced. The background section finalizes with the challenges of high performance computing oriented to quantum chemistry program development. In Section 3 the overall algorithm for the new RI-MP2 algorithm is presented. Further, in Section 4 the benchmarks and comparisons against state-of-the-art programs are discussed. Section 5 concludes.

2 Background

2.1 MP2 and the RI Approximation

Second-order Møller-Plesset Perturbation Theory (MP2)[1] is a method which provides a correction to the Hartree-Fock (HF) energy of a system by approximating the effects of electron correlation. The closed-shell MP2 energy correction is defined by the following sum:

$$E^{(2)} = \sum_{ij}^{N_o} \sum_{ab}^{N_v} \frac{G_{ia}^{jb}(2G_{ia}^{jb} - G_{ib}^{ja})}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \tag{1}$$

Here i and j are indices for the N_o occupied molecular orbitals (MOs) $\{\psi_i(\boldsymbol{r})\}$, while a and b represent the N_v virtual MOs $\{\psi_a(\boldsymbol{r})\}$; ϵ_i, ϵ_j, ϵ_a, ϵ_b are the MO energies. The fourth-order tensor G_{ia}^{jb} contains two-electron repulsion integrals (ERIs) $G_{ia}^{jb} \equiv (ia|jb)$ over the MOs.

The evaluation of Eq. (1) requires the integrals to either be stored or repeatedly recomputed. The former imposes an undesirable $\mathcal{O}(N^4)$ memory bottleneck,[68] while the latter is computationally inefficient.

The RI approach is a commonly-utilized solution to this problem. The RI method approximates the four-centre integrals $(ia|jb)$ through linear combinations of three-centre integrals $(ia|P)$ between the MOs and the functions of an auxiliary basis set. The three-centre integrals can be stored in only $\mathcal{O}(N^3)$ space, and the assembly of the four-centre integrals from the three-centre integrals can be expressed as matrix multiplications which can operate at near-peak performance on modern hardware. Furthermore, the usage of adequately optimized auxiliary basis sets allows RI-MP2 to achieve a

level of accuracy comparable to plain MP2.[95,96]

In detail, the RI method approximates the four-centre integrals by the sum

$$G_{ia}^{jb} \approx \sum_{PQ}^{N_x} D_{ia}^{P} J_{PQ}^{-1} D_{jb}^{Q} \tag{2}$$

where the third-order tensor D_{ia}^{P} contains the three-center two-electron integrals

$$D_{ia}^{P} = \iint \psi_i(\boldsymbol{r}_1)\psi_a(\boldsymbol{r}_1)\frac{1}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|}\phi_P(\boldsymbol{r}_2)\, d\boldsymbol{r}_1\boldsymbol{r}_2 \tag{3}$$

between the MOs and the N_x auxiliary basis set functions $\{\phi_P(\boldsymbol{r})\}$. The matrix J_{PQ}^{-1} is the inverse of the matrix whose elements are the following two-center two-electron integrals

$$J_{PQ} = \iint \phi_P(\boldsymbol{r}_1)\frac{1}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|}\phi_Q(\boldsymbol{r}_2)\, d\boldsymbol{r}_1\boldsymbol{r}_2 \tag{4}$$

The MOs are represented as a linear combination of N atomic orbital (AO) basis functions $\{\phi_\mu\}$ within a primary basis set

$$\psi_m(\boldsymbol{r}) = \sum_{\mu}^{N} C_{m\mu}\phi_\mu(\boldsymbol{r}) \tag{5}$$

where m is a generic index for either occupied or virtual MOs, and the $C_{m\mu}$ coefficients are obtained by solving the HF equations. The three-centre MO integrals D_{ia}^{P} are evaluated via the corresponding linear combination of atomic orbital integrals,

$$D_{ia}^{P} = \sum_{\mu\nu}^{N} C_{i\mu}C_{a\nu}d_{\mu\nu}^{P} \tag{6}$$

where the $d_{\mu\nu}^{P}$ values are the three-centre integrals of atomic orbitals:

$$d_{\mu\nu}^{P} = \iint \phi_\mu(\boldsymbol{r}_1)\phi_\nu(\boldsymbol{r}_1)\frac{1}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|}\phi_P(\boldsymbol{r}_2)\, d\boldsymbol{r}_1\boldsymbol{r}_2 \tag{7}$$

2.2 Two- and Three-Center Integrals on GPU

As per Eq. (2), the RI-MP2 algorithm requires the computation of two- and three-centre two-electron repulsion integrals.

In this work we evaluate the two- and three-centre integrals via a set of recursion relations based on the Head-Gordon-Pople (HGP) algorithm.[97] The HGP algorithm is based on the Obara-Saika recursion relations.[98] These recurrence relations are implemented in a set of efficient GPU kernels analogous to the four-centre integral kernels presented by Barca *et al.*[99,100] Due the similarity to already-published work, these kernels will not be discussed in detail in this paper.

2.3 Algorithmic Features

Of the $\mathcal{O}(N^5)$ total computational complexity of RI-MP2, there is only $\mathcal{O}(N^3)$ computational work which cannot be performed through efficient level-3 BLAS routines.[101]

Using vendor-optimized libraries, these BLAS routines can potentially run at near-peak performance on GPU if the associated data transfers are efficiently hidden. The steps that cannot efficiently be expressed as BLAS routines are the inversion of the Coulomb matrix J, the final accumulation of the energy per Eq. (1), and the formation of the electronic integrals per Eqs. (3),(4).

Rather than inverting J explicitly, Eq. (2) is evaluated by first taking the Cholesky decomposition $J = LL^T$, where L is lower-triangular. Eq. (2) then becomes

$$G_{ia}^{jb} \approx \sum_{PQK}^{N_x} D_{ia}^{P} L_{PK}^{-T} L_{KQ}^{-1} D_{jb}^{Q}. \tag{8}$$

This expression can be evaluated in $\mathcal{O}(N^5)$ time with level-3 BLAS routines. The Cholesky decomposition itself is only $\mathcal{O}(N^3)$ work, and can be performed efficiently through a vendor-optimized LAPACK routine. However, this operation scales poorly across many GPUs.

Combining the elements of G_{ia}^{jb} with the orbital energies as in Eq.(1) requires $\mathcal{O}(N^4)$ work

with a small pre-factor and is easily implemented using GPU parallelism through a simple custom GPU kernel. The calculation of the electronic integrals, as discussed previously, is performed in negligible time due to our use of highly-optimized GPU kernels for integral evaluation.[100,102,103]

In summary, the most computationally significant components of the algorithm can be expressed as efficient BLAS calls. As such, the primary goal of our design is to optimize the workload scheduling such that all GPUs in the system can be running BLAS calls for as great a proportion of the run-time as possible.

With the main computational traits of the algorithm introduced, the following section details how the relevant challenges have been addressed in the implementation of the algorithm in order to achieve near-peak overall performance.

1 Compute J_{PQ} on GPU $\mathcal{O}(N^2)$
2 Cholesky factor J_{PQ} on GPU
// Form D_{ia}^{P} tensor, batched over auxiliary orbitals
3 **foreach** *batch* $\mathcal{P}$ **do**
4 Compute $d_{\mu\nu}^{\mathcal{P}}$ for all μ, ν directly on GPU
5 Compute $D_{\mu a}^{\mathcal{P}} = \sum_{\nu} d_{\mu\nu}^{\mathcal{P}} C_{\nu a}$ by matrix multiplication $\mathcal{O}(N^4)$
6 Compute $D_{ia}^{\mathcal{P}} = \sum_{\nu} D_{\mu a}^{\mathcal{P}} C_{\mu i}$ by matrix multiplication $\mathcal{O}(N^4)$
7 Transfer $D_{ia}^{\mathcal{P}}$ GPU $\rightarrow$ host
8 **end**
// Form E_{ia}^{P} tensor, batched over occupied and virtual orbitals
9 **foreach** *batch* $\mathcal{I}, \mathcal{A}$ **do**
10 Transfer $D_{\mathcal{I}\mathcal{A}}^{P}$ slice host $\rightarrow$ GPU
11 Compute $E_{\mathcal{I}\mathcal{A}}^{P} = \sum_{Q} J_{PQ}^{-1} D_{\mathcal{I}\mathcal{A}}^{Q}$ by triangular solve with Cholesky-decomposed J_{PQ} $\mathcal{O}(N^4)$
12 Transfer $E_{\mathcal{I}\mathcal{A}}^{P}$ slice GPU $\rightarrow$ host
13 **end**

Algorithm 1: Formation of the D_{ia}^{P} and E_{ia}^{P} intermediates for the RI-MP2 calculation. Tensors on the host are stored in shared memory. This entire algorithm scales as $\mathcal{O}(N^4)$ due to the tensor operations on lines 5, 6, and 11. When run on multiple GPUs, iterations of the "**foreach**" loops are distributed across available GPUs.

3 Algorithm

One more detail must be introduced before we detail the algorithm. The ERI assembly can be simplified as per Eq. (2) into a single tensor contraction by contracting the J^{-1} matrix with the MO integrals beforehand via a Triangular Solve routine. In our work we use the tensor E_{ia}^{P}, defined as follows

$$E_{ia}^{P} = \sum_{Q} J_{PQ}^{-1} D_{ia}^{Q}. \tag{9}$$

The ERI evaluation is then simplified to the following expression

$$G_{ia}^{jb} = \sum_{P} D_{ia}^{P} E_{ia}^{P}. \tag{10}$$

An alternative approach, common in the literature,[75,104,105] is to calculate the inverse square-root of the J matrix to obtain a symmetric decomposition, as follows.

$$B_{ia}^{P} = \sum_{Q} J_{PQ}^{-1/2} D_{ia}^{Q} \tag{11}$$

$$G_{ia}^{jb} = \sum_{P} B_{ia}^{P} B_{jb}^{P} \tag{12}$$

This approach has the benefit of only requiring the storage of one three-index tensor rather than two, but the disadvantage of requiring a matrix inverse square-root computation.

As detailed in this Section, our implementation requires additional memory to store two extra tensors for MPI communication. These additional buffers can be used to store the E_{ia}^{P} tensor, meaning the symmetric decomposition offered by the B_{ia}^{P} approach would not offer an improved memory footprint. Thus the E_{ia}^{P} approach comes with no downsides unless the algorithm is only being run on only one node (rendering the MPI buffers redundant).

3.1 Single-Node RI-MP2 Algorithm

There are four key steps to the RI-MP2 algorithm:

1. Formation and Cholesky decomposition of the two-centre integrals J_{PQ} between the auxiliary basis functions.
2. Formation of three-centre atomic orbital (AO) integrals $d_{\mu\nu}^{P}$ and transformation

```
   // Compute E^(2), batched over occupied orbital
      pairs
1  E := 0
2  foreach batch-pair A, B do
3  |   Transfer D^P_iA, E^P_jB slices Host → GPU
4  |   Compute G^jB_iA = Σ_P D^P_iA E^P_jB                    O(N^5)
5  |   forall a ∈ A, b ∈ B, i, j do
6  |   |   E += (G^jb_ia (2G^jb_ia − G^ib_ja)) / (ε_i + ε_j − ε_a − ε_b)   O(N^4)
7  |   end
8  end
```

Algorithm 2: Final MP2 energy reduction algorithm. The loop over batch-pairs is distributed across available GPUs, and also accounts for symmetry between occupied orbital batches. The tensor contraction on line 4 constitutes the $\mathcal{O}(N^5)$ computational bottleneck. Each individual GEMM operation in this contraction is $\mathcal{O}(k^2 N_o^2 N_{aux})$, where k is the virtual batch size.

into the molecular orbital (MO) basis, D_{ia}^P.

3. Formation of the transformed RI integrals E_{ia}^P from D_{ia}^P and the Cholesky-decomposed J_{PQ} matrix.
4. Final energy accumulation using D_{ia}^P and E_{ia}^P.

The first three steps comprise the intermediate tensor formation, which is described by Algorithm 1. The final energy reduction step is described by Algorithm 2. The remainder of this section provides an overview of each of these steps.

Throughout the algorithm, MPI shared memory is used to facilitate multi-GPU parallelism. In the final energy reduction, every slice of D_{ia}^P and E_{ia}^P is used $\mathcal{O}(N)$ times. Storing these tensors in a MPI shared memory window gives each GPU uniform access to them, avoiding communication and load-balancing overheads. MPI shared memory also facilitates parallelism of the intermediate tensor formation, as tensors can be formed in independent slices which are written conflict-free to shared memory.

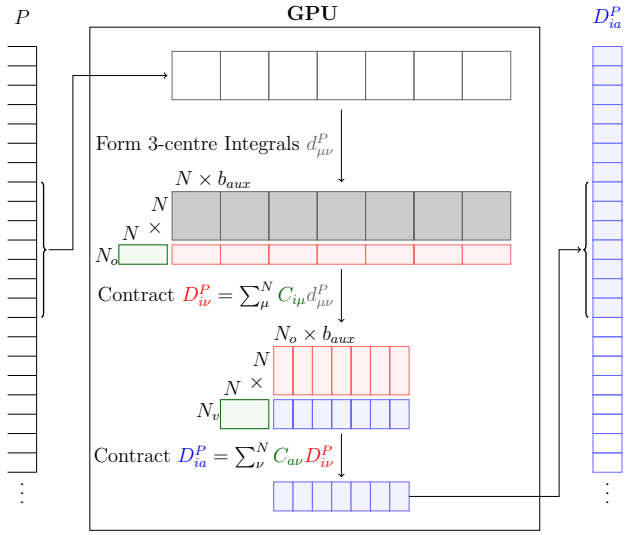


Figure 1: Formation of the D_{ia}^P intermediate, the three-centre integrals in the MO basis. Given a batch of auxiliary indices of dimension b_{aux}, the formation proceeds as follows: 1) Form the three-centre integrals $d_{\mu\nu}^P$ corresponding to the auxiliary batch through custom GPU integral kernels. 2) Contract the integrals with the occupied MO coefficients to form $D_{i\nu}^P$. 3) Contract again with the virtual MO coefficients to form D_{ia}^P. 4) Transfer D_{ia}^P slice into Host shared memory.

3.1.1 Formation of the D_{ia}^P and J_{PQ} tensors

The D_{ia}^P tensor is formed following the scheme visualized in Fig. 1. Given an auxiliary index range, a GPU forms the corresponding slice of D_{ia}^P (encompassing all i, a indices) and transfers the slice back to Host shared memory. Device → Host data transfers are hidden by overlapping the formation of each slice with the transfer of the previous slice to the Host. Auxiliary index batches are distributed dynamically across all GPUs to achieve multi-GPU parallelism. The AO integrals $d_{\mu\nu}^P$ are evaluated through custom GPU integral kernels,[100,102] while the subsequent tensor contractions are performed through vendor-optimized BLAS matrix multiplication (DGEMM) operations.[101] Notably, by forming the $\mathcal{O}(N^2)$ data on the GPU and directly performing the MO transformation, the required Host ↔ Device data transfer per batch is only $\mathcal{O}(N_o N_v)$, which

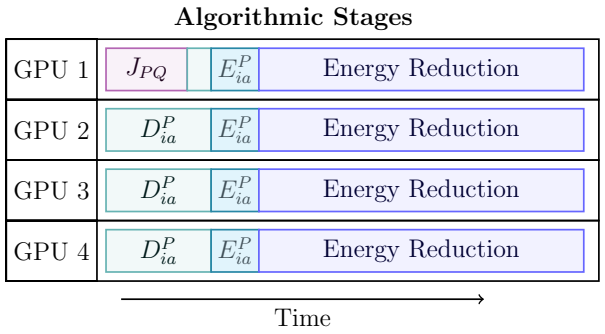


Figure 2: Timeline of the computational stages of the algorithm, as described in section 3.1.

in practice is typically 5-20 times smaller than N^2, as visualized in Fig. 1, where blue squares represent $\mathcal{O}(N_o N_v)$ data and grey square represent $\mathcal{O}(N^2)$ data.

The J_{PQ} matrix is formed through custom GPU integral kernels and is then Cholesky factorized using a vendor-optimized LAPACK routine, DPOTRF. The DPOTRF routine does not scale well over multiple GPUs, presenting an obstacle to parallel efficiency. This problem is addressed by performing the Cholesky decomposition on a single GPU, and transferring it to the other GPUs through Host shared memory. While this single GPU is working on the J_{PQ} tensor, all other GPUs begin parallel computation of D_{ia}^{P}. Once the Cholesky decomposition is complete, the responsible GPU joins in on the computation of D_{ia}^{P}. A dynamic workload distribution is used for the computation of D_{ia}^{P}, allowing this process to occur seamlessly with no loss of parallel efficiency. The effect of this scheme can be seen in Fig. 2, which provides a timeline of the computational effort of the algorithm.

3.1.2 Formation of E_{ia}^{P}

This step is a triangular solve of D_{ia}^{P} with the Cholesky factored J_{PQ}. Batches of the fused ia index are distributed dynamically among the available GPUs. The corresponding blocks of D_{ia}^{P} are transferred to the GPU, which solves the linear system $J_{PQ}E_{ia}^{P} = D_{ia}^{P}$ through vendor-optimized DTRSM calls to produce a block of E_{ia}^{P} which is then transferred back to shared memory. The computation of a slice is overlapped with the transfer of the previous slice back to the Host, along with the transfer of the next D_{ia}^{P} slice to the GPU.

In the present implementation, this is the asymptotic GPU-memory-limiting stage of the algorithm due to the $\mathcal{O}(N_{aux}^2)$ requirement in addition to double-buffered slices of E_{ia}^{P} and D_{ia}^{P}. This bottleneck could be alleviated by working on the J matrix panel-by-panel, but this was found to be unnecessary for the tests in this paper.

3.1.3 Energy Reduction Step

This step is the $\mathcal{O}(N^5)$ computational bottleneck of the algorithm. Pairs D_{ia}^{P} and E_{jb}^{Q} of slices, batched over the virtual orbital indices a and b, are dynamically distributed across GPUs, where they are contracted to form a slice of the two-centre four-electron integrals G_{ia}^{jb}. We choose to batch over virtual orbitals as it provides a larger space of index pairs to batch over in comparison to occupied orbitals, allowing better distribution across GPUs. If the batch size is k, each individual matrix multiplication is $\mathcal{O}(k^2 N_o^2 N_{aux})$, and there are $\mathcal{O}(N_v^2/k^2)$ multiplications in total. The contracted slices are reduced according to Eq. (1) via a custom kernel to form the energy contribution of a slice-pair. Finally, the energy contributions of all slice-pairs are reduced across all GPUs to form the RI-MP2 energy. From the perspectives of computational complexity and concurrent orchestration, this is the most complex step of the algorithm.

3.2 Distributed RI-MP2

The single-node algorithm is extended to a distributed algorithm through a series of mild augmentations. The primary concern in the design of a distributed variant is that the three-index tensors will be too large to store on a single node for large inputs, so the formation and storage of these tensors must be distributed. Each node is assigned a range $\mathcal{A}$ of virtual orbitals, for which it computes the slice $D_{i\mathcal{A}}^{P}$. That is,

each node performs the contraction

$$D_{ia}^{P} = \sum_{\mu\nu} C_{i\mu} C_{a\nu} d_{\mu\nu}^{P}$$

for the virtual orbitals $a \in \mathcal{A}$.

The J_{PQ} matrix is handled in a similar manner to the single-node algorithm. A single GPU on a single node is tasked with its computation. Upon completion, the Cholesky-decomposed matrix is broadcast into shared-memory on every node.

It should be noted that in the work of Katouda *et al.*,[75] the J_{PQ} matrix is ultimately the cause of declining parallel efficiency at scale. Their approach instead explicitly computes $J_{PQ}^{-1/2}$ through matrix inversion following the Cholesky decomposition. This process is performed redundantly on each node. Our approach has two advantages which mitigate the scalability issue. First, the matrix inversion is significantly more expensive than the Cholesky decomposition, so by avoiding this step we reduce the required serial workload. Second, we overlap the Cholesky decomposition with other distributed computations, preventing compute time from being wasted on redundant work.

There are three obstacles to the theoretical scaling of this distribution scheme.

1. Every node must compute the entire $d_{\mu\nu}^{P}$ tensor. This is ultimately insignificant due to the screening techniques and efficient kernels used for integral computation, and presents no real barrier to parallel efficiency.

2. As the number of nodes increases, the dimension of the involved matrix multiplications decreases (as the virtual orbital batch size decreases). Performance of matrix multiplication routines typically degenerates as matrix dimensions decrease, and this degeneration impacts the scalability of the algorithm. This could be mitigated by moving to a two-dimensional data distribution which batches over both occupied and virtual orbitals, but in this work we found this to be unnecessary.

3. The J_{PQ} Cholesky decomposition is not distributed across nodes, making it a strong-scaling bottleneck. However, the $\mathcal{O}(N_{aux}^{3})$ Cholesky decomposition is dwarfed by the $\mathcal{O}(N^{2} N_{o} N_{v})$ formation of the D_{ia}^{P} tensor, meaning the bottleneck will only manifest in the extreme limit of parallel scaling where the distributed D_{ia}^{P} formation becomes faster than the Cholesky decomposition.

These problems were accepted as trade-offs in exchange for minimal inter-node communication for this stage of the algorithm.

3.3 Intermediate Formation

The Intermediate Formation stage of the algorithm concerns the formation of the D_{ia}^{P} and E_{ia}^{P} tensors in a distributed fashion. The computation is distributed along the virtual orbital axis a, with each node assigned a block of virtual orbitals. All of the three-centre AO integrals $d_{\mu\nu}^{P}$ are calculated independently on each node, while each node only performs the section of the MO contraction corresponding to its assigned virtual orbital range. Each node then contracts its D_{ia}^{P} slice with the Cholesky-factored Coulomb matrix J_{PQ} to form the corresponding E_{ia}^{P} slice. The only inter-node communication required in this stage of the algorithm is the broadcast of the Cholesky-factored Coulomb matrix.

The redundant computation of the AO integrals across all nodes can be eliminated by instead distributing the work along the auxiliary orbital axis P. However, the formation of E_{ia}^{P} requires contraction along the P axis, so this approach would necessitate an expensive $\mathcal{O}(N^{3})$ distributed tensor permutation which cannot be efficiently overlapped with computation. In comparison, the computation of the AO integrals is computationally light given the use of screening techniques and efficient GPU-based integral kernels, so their redundant computation on every node does not present a barrier to scalability.

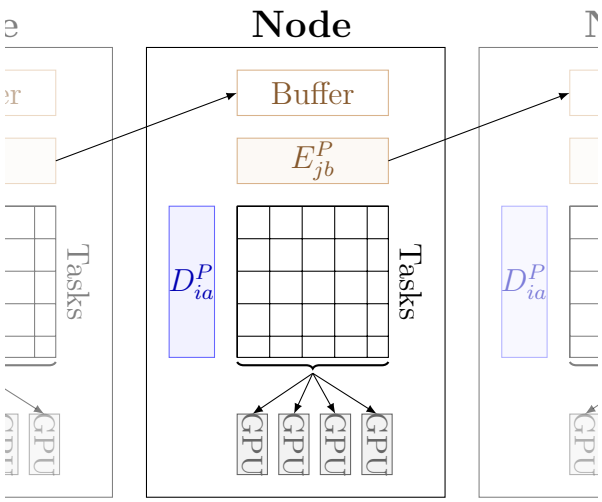


Figure 3: Visualization of the final energy reduction workflow. The tasks corresponding to the resident D^P_{ia} and E^P_{jb} slices are dynamically distributed across available GPUs. Concurrently, the resident E^P_{jb} slice is sent to the next node in the ring while the subsequent E^P_{jb} slice is received from the previous node into a local buffer.

3.4 Energy Reduction Step

For the $\mathcal{O}(N^5)$ energy reduction step, we have implemented a sophisticated scheme leveraging several layers of asynchronism which allows all of the $\mathcal{O}(N^3)$ inter-node communication and Host $\rightarrow$ Device data transfers to be completely hidden by the $\mathcal{O}(N^5)$ computation, given a sufficient problem size for the available computational resources.

Our system utilizes a ring communication algorithm in which each node stores its own D^P_{ia} slice locally, while the E^P_{ia} slices are passed around the ring. This system is visualized in Fig. 3. In practice the E^P_{ia} slices are large, typically at least several gigabytes, owing to their $\mathcal{O}(N^3)$ space complexity. However, the computational effort involved with each slice scales as $\mathcal{O}(N^5)$, meaning a sufficiently large N allows any inter-node communication overhead to be hidden by computation. The same is true for Host $\rightarrow$ Device data transfers.

The computational workflow is implemented through an MPI scheme in which each node hosts three types of MPI ranks:

1. One Worker rank per GPU, each of which issues commands to the corresponding GPU.

2. One Communicator rank, which handles communication of E^P_{ia} slices with other nodes.

3. One Coordinator rank, which dynamically distributes tasks to the workers and coordinates node-local synchronization when necessary.

The remainder of this section details the roles of each of these rank types.

3.4.1 Workers

The core of the energy reduction algorithm is the GPU-based reduction routine which takes virtual-orbital slices of D^P_{ia} and E^P_{jb} and computes the energy contribution of the slice pair. This process has three steps:

1. Copy the D^P_{ia} and E^P_{jb} slices to the GPU.

2. Form $G^{jb}_{ia} = \sum_P D^P_{ia} E^P_{jb}$ via vendor-optimized matrix multiplication routine.

3. Calculate energy contribution per Eq. (1) via custom GPU kernel.

This process is implemented using GPU Streams and Events, along with double-buffering to allow the entire computation to be launched asynchronously. Specifically, a two-Stream system is employed in which all data transfers are issued in one Stream and all computations are issued in the second Stream. Events are used for cross-stream synchronization, allowing the computation Stream to wait for transfers to complete before beginning the corresponding computation, and allowing the memory Stream to delay transfers until the corresponding buffers are free.

In the distributed energy reduction scheme, workers behave as simple asynchronous devices under control of the local coordinator. Workers repeatedly request tasks from the local coordinator, asynchronously launching given tasks on the GPU. If a worker ever has more than two

tasks in-flight at once, it will first wait for one task to complete before requesting further work. This behaviour enables dynamic load-balancing between the workers on a node.

3.4.2 Communicator

The Communicator is the rank which implements the inter-node ring communication scheme, as in Fig. 3. Each node stores its own D_{ia}^P slice locally for the duration of the algorithm, while the E_{ia}^P slices are passed around the ring to be paired with each D_{ia}^P slice. At each step of the communication algorithm, every node sends its locally stored E slice to the following node while receiving a new E slice from the previous node. Due to symmetry of the virtual orbital indices, E slices need only be passed half way around the ring for the algorithm to complete.

An MPI double-buffering system is employed to allow overlap between computation and transfer of E slices. The communication rank is told by the coordinator when it is safe to begin overwriting the contents of a buffer.

3.4.3 Coordinator

The Coordinator is the rank responsible for scheduling inter-node data transfers and distributing work among local Workers. Given slices of D_{ia}^P and E_{ia}^P, the coordinator partitions the available work into tasks and dynamically distributes tasks between available Workers.

Once all tasks corresponding to the current E slice are completed, the Coordinator notifies the Communicator rank that the buffer is now free for the next asynchronous inter-node data transfer.

4 Results

In this Section we characterize the performance of our implementation through wall-times and FLOP-rate calculations.

Section 4.1 begins with a description of the supercomputer systems used in the test runs. Section 4.2 establishes a baseline by comparing the performance of our implementation with three state-of-the-art quantum chemistry codes, GAMESS,[106] QChem[107] and Orca.[108]

The remaining sections consist of an independent evaluation of the performance of our implementation. Section 4.3 presents strong scaling data across multiple GPUs within a node, while Sections 4.4.1 and 4.4.2 present, respectively, weak and strong scaling data on multiple nodes. Water clusters are used for all scaling tests. Only $\mathcal{O}(N^3)$ work in the RI-MP2 algorithm depends on the details of the input beyond the number of basis functions, so the scaling behaviour of water clusters is indicative of the scaling for practical inputs. For completeness, we also present performance benchmarks for some chemically interesting systems in Section 4.5.

Throughout this section, FLOP rates for our software are determined by code instrumentation to count the FLOPs of BLAS calls. All non-BLAS GPU operations are ignored, producing an under-estimate of the true FLOP count. Each $m \times k$ by $k \times n$ GEMM is counted as $2mkn$ FLOPs, while each $n \times n$ by $n \times k$ TRSM is counted as n^2k FLOPs.

Unless otherwise specified, RI-MP2 calculations are performed using the cc-pVDZ and cc-pVDZ-RIFIT as the primary and auxiliary basis set, respectively.

All RI-MP2 calculations do not adopt the frozen core approximation, thereby using all occupied and virtual molecular orbitals.

4.1 The Perlmutter and Gadi Supercomputers

The multi-node scaling data presented in this paper was gathered on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center (NERSC), while the single-node data was gathered on the Gadi supercomputer at the National Computation Infrastructure (NCI) in Australia.

The standard GPU nodes of Gadi each house two 24-core Intel Xeon Platinum 8268 CPUs for a total of 48 CPU cores, in addition to four NVIDIA V100 GPUs with 32GB of High Bandwidth Memory (HBM). Each CPU socket has a peak performance of roughly 2.1 TFLOP/s

and a Thermal Design Power (TDP) of 205 W, while each GPU has a peak performance of 7.8 TFLOP/s and a TDP of 300 W.

Gadi also features nodes with A100 GPUs, which each house two 64-core AMD EPYC 7742 CPUs for a total of 128 CPU cores, along with 8 NVIDIA A100 GPUs with 80GB of HBM each, and a peak performance of 19.5 TFLOP/s and a TDP of 400 W.

Perlmutter consists of 1792 nodes each with a 64-core AMD EPYC 7763 CPU and four A100 GPUs, where 256 nodes have A100's with 80GB of HBM while the remaining nodes have 40GB.

4.2 Comparison with Other Software

The performance of our implementation is compared with that of GAMESS, QChem, and ORCA. For these three software packages, we use a 48-core Gadi Cascade Lake node, with a peak performance of 4.2 TFLOP/s. We measure our software with a single NVIDIA V100 GPU with a peak performance of 7.8 TFLOP/s. Considering the peak performance ratios, the expected speedup of a V100 against the CPU node is 1.85.

In these tests, EXESS, ORCA and QChem are all configured to use Cartesian basis functions, while GAMESS only supports Spherical Harmonics. This reduces the workload for GAMESS by 21% for DZ and 36% for TZ basis sets.

The algorithms are compared using water clusters of varying sizes. All inputs are evaluated with both the cc-pVDZ/cc-pVDZ-RIFIT and cc-pVTZ/cc-pVTZ-RIFIT basis sets, except for the 75 molecule cluster, where the available memory is insufficient to run TZ calculations. The results are presented in Figure 4.

In the DZ tests, the least favourable comparison for EXESS is found in the case of 75 water molecules, where EXESS achieves a speedup of 3.5 over the next best package, GAMESS. In all other cases, the speedup attained by EXESS exceeds 3.5, with typical speedups in the range of 3.5-4.5 times. This exceeds the expected speedup of 1.85 based on the theoretical peak FLOP rates, indicating that our software makes better use of the underlying hardware than the other packages. In the TZ tests, speedups of 2.8-3.8 are achieved over ORCA and QChem, while speedups of 1.7-1.9 are achieved over GAMESS. The favourable performance of GAMESS in comparison to ORCA and QChem is primarily due to the use of spherical harmonics.

Another noteworthy point of comparison is the time allocation cost. On Gadi, a normal 48-core node costs 96 Service Units (SU) per hour, while a single V100 GPU costs only 36 SU per hour. Accounting for the >2.8 times speedup achieved by EXESS and the >2.6 times cheaper hardware, running the calculations with EXESS is over 7 times cheaper in comparison with the other packages (less so when comparing against spherical harmonics). The TDP of each alternative is proportional to the SU cost, so the same factor of 7 applies to savings in power consumption as well.

With a strong baseline established with respect to state-of-the-art packages, we now present scaling tests to demonstrate that this efficiency persists at scale.

4.3 Single-Node Strong Scaling

A strong scaling analysis of the multi-GPU algorithm has been conducted on the Gadi supercomputer, with nodes featuring four NVIDIA V100 GPU cards as well as nodes with eight NVIDIA A100 cards. Performance is evaluated using clusters of 40, 80 and 150 water molecules with the cc-pVDZ/cc-pVDZ-RIFIT basis sets (the 150 water molecule input is too large to run on the V100 nodes). These inputs require, respectively, approximately 5GB, 40GB and 240GB of Host memory to store each of the three-index tensors. Strong scaling results for both V100 and A100 GPUs are shown in Fig. 5. The A100 GPUs with the $(H_2O)_{150}$ input achieve a parallel efficiency of 97% on 8 GPUs while achieving 86% of theoretical peak performance, while the $(H_2O)_{80}$ on V100 GPUs achieves a parallel efficiency of 98% while achieving 85% of theoretical peak performance. These inputs are both large, with wall-times on the scale of minutes to hours. The

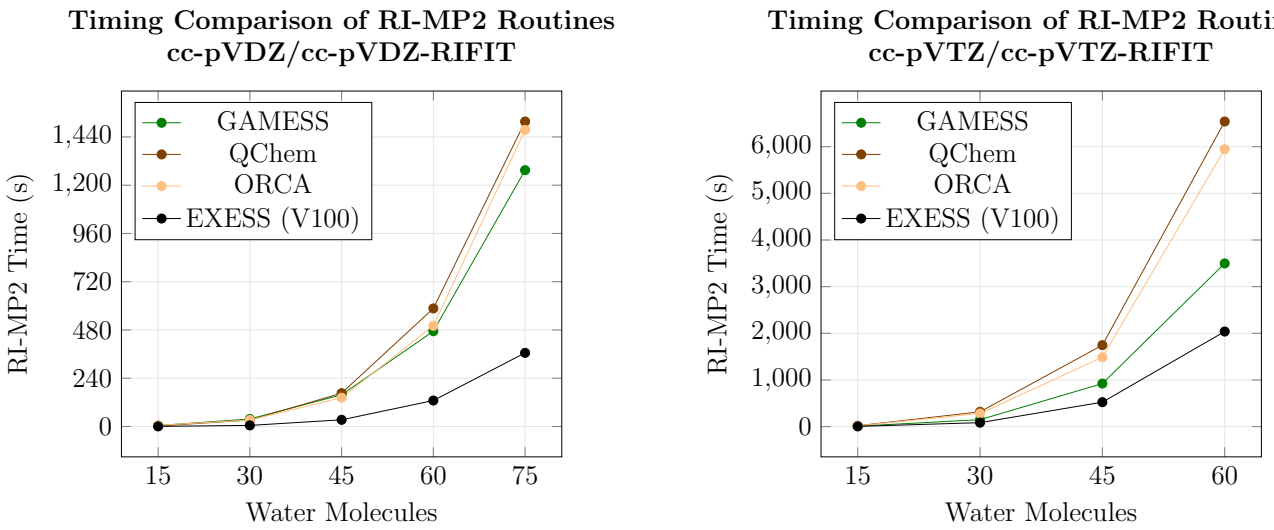
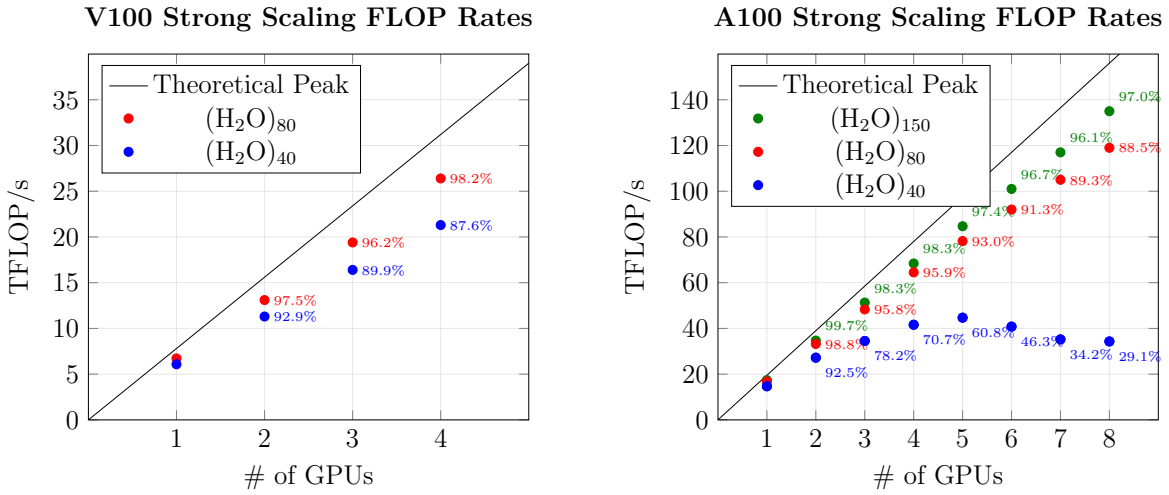


Figure 4: RI-MP2 execution times for three software packages in comparison with EXESS, measured on water clusters of varying sizes with the cc-pVDZ/cc-pVDZ-RIFIT (left) and cc-pVTZ/cc-pVTZ-RIFIT (right) basis sets. The GAMESS, QChem and ORCA packages are run on a 48-core Intel Cascade Lake node, while EXESS is run on a single NVIDIA V100 GPU.



Input	GPU	# of GPUs 1	2	3	4	5	6	7	8
$(H_2O)_{40}$	V100	18.5s	10.0s	6.9s	5.3s				
$(H_2O)_{80}$	V100	496s	255s	172s	126s				
$(H_2O)_{40}$	A100	8.76s	4.70s	3.40s	2.74s	2.51s	2.71s	3.09s	3.17s
$(H_2O)_{80}$	A100	204s	103s	71.0s	53.2s	43.9s	37.3s	32.5s	28.6s
$(H_2O)_{150}$	A100	4310s	2160s	1460s	1100s	887s	741s	640s	559s

Figure 5: Strong scaling data for $(H_2O)_n$/cc-pVDZ/cc-pVDZ-RIFIT inputs, each with $5n$ occupied, $20n$ virtual and $96n$ auxiliary orbitals. Percentages at each data point express the parallel efficiency compared with a single GPU. The table contains wall-times for all test runs. The $(H_2O)_{150}$ input does not fit in the memory of a V100 node and is thus excluded.

smaller $(H_2O)_{40}$ input achieves less desirable scaling on the A100 node due to the dearth of work, namely as the tasks assigned to each GPU become so small that they cannot be executed efficiently by A100 GPUs.

4.4 Distributed Scaling

This section details weak- and strong-scaling results for multi-node computations on the Perlmutter system. Results are presented for both 40GB and 80GB models of NVIDIA A100 GPUs. All scaling tests are carried out with inputs consisting of water clusters using the cc-pVDZ/cc-pVDZ-RIFIT basis sets. Results are shown in Fig. 6, the test cases are described and discussed in the following subsections.

4.4.1 Weak Scaling

Weak scaling tests were carried out with water clusters of varying size to maintain a constant workload per node. The workload associated with a water cluster is approximated by summing the theoretical FLOP costs of each of the main tensor contractions. Inputs were chosen so that the estimated wall time (obtained by dividing FLOP count by theoretical peak performance) was no greater than 4 minutes per input. The chosen input sizes are tabulated in Table 1.

On the smallest node count, 4 nodes with an input of 149 water molecules, our code runs at 89% of peak performance on 40GB GPUs and 93% of peak on 80GB GPUs. The largest input, 160 nodes with 314 water molecules, operates at 71% of peak (8.9 PFLOP/s) on 40GB GPUs and 86% of peak (10.8 PFLOP/s) on 80GB GPUs.

The algorithm as presented has imperfect theoretical weak scaling, as the amount of data stored on each node shrinks as the node count increases (due to $\mathcal{O}(N^3)$ memory compared to $\mathcal{O}(N^5)$ computation). This results in a worse ratio of computation to communication as the node counts grows even though the amount of work per node stays constant. This could be solved by introducing a data replication system to keep the data per node constant along with the computation per node, but this is deferred to future work. Even with this imperfection, we still achieve good weak scaling up to several hundred GPUs.

4.4.2 Strong Scaling

Strong scaling tests were performed with a cluster of 314 water molecules, totalling 1570 occupied, 6280 virtual and 30144 auxiliary basis functions. The total memory footprint of this input, including communication buffers, is 6.64TB.

The node counts included in the tests are multiples of 20, starting at 60. Fewer than 60 nodes would not provide enough memory to handle the input. Parallel efficiencies are measured relative to 60 nodes.

On the 80GB A100 nodes, the code achieves outstanding parallel efficiency with respect to the theoretical peak performance. The $(H_2O)_{314}$ system achieves a 95% parallel efficiency using 640 GPUs, attaining 83% of theoretical peak performance. Scalability in the 40GB nodes reaches a maximum parallel efficiency of 96% at 400 GPUs and degrades to 70% for larger node counts.

The greater amount of HBM in the 80GB GPUs allows each GPU to perform larger matrix multiplications, leading to a better ratio of Host $\rightarrow$ Device data transfer to computation. This is why the 40GB nodes plateau in scaling before the 80GB nodes.

4.5 Timings and Floating-Point Performance

The new distributed RI-MP2 algorithm is applied to two datasets of spherical sections of crystal lattice structures with either 20 neutral molecules, namely benzene and paracetamol, or with 20 ion pairs of ionic liquids, namely 1-ethyl-3-methylimidazolium tetrafluoroborate ($[C_2mim]^+[BF_4]^-$), ethylammonium nitrate ($[EtNH_3]^+[NO_3]^-$), guanidinium tetrafluoroborate ($[Gdm]^+[BF_4]^-$), guanidinium chloride ($[Gdm]^+Cl^-$) and guanidinium ethanesulfonate ($[Gdm]^+[EtSO_3]^-$).

Each input was run on both 16 and 32 nodes of Perlmutter, with four 40GB A100 GPUs each. Results are tabulated in Table 2. Our implementation operates at 50-85% of theoretical peak performance for almost all inputs, demonstrating the applicability of our solution

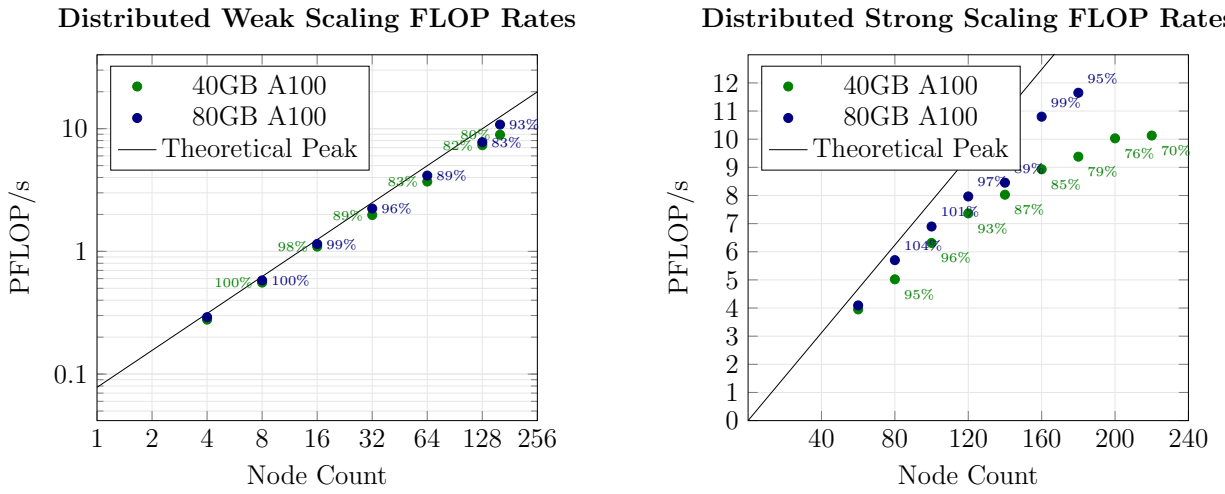


Figure 6: Strong and weak scaling data for our implementation on Perlmutter, with both 40GB and 80GB A100 GPUs. The percentages attached to each point present the respective weak/strong parallel efficiency at the corresponding point relative to the smallest node count (4 for weak, 60 for strong), where a value of 100% implies perfect linear scaling. The inputs used for weak scaling are described in Section 4.4.1, while strong scaling is done on a system of 314 water molecules (1570 occupied, 6280 virtual, 30144 auxiliary basis functions). All inputs use the cc-pVDZ/cc-pVDZ-RIFIT basis sets.

Table 1: Water Cluster Sizes and Estimated Completion Times for the weak scaling inputs.

# Nodes	# GPUs	Water Cluster Size	Est. Completion Time (s)
4	16	149	228s
8	32	172	234s
16	64	198	236s
32	128	227	233s
64	256	261	234s
128	512	300	235s
160	640	314	236s

to practical problems. Of particular note is the 32-node calculation for $[Gdm]^+Cl^-$, in which our implementation sustains just over 52% of peak performance (1.3 PFLOP/s) with a wall-time of only 5.267 seconds, demonstrating that our implementation does not require large inputs to achieve good hardware utilization.

5 Conclusion

This paper presented a novel distributed memory multi-GPU RI-MP2 algorithm, which accelerates all steps of the algorithm on GPUs and has been carefully optimized to improve parallel scaling and performance at the single-node, and many-node levels. For the single node version of the algorithm, we demonstrate near-optimal parallel scaling and near-peak performance on up to 4 NVIDIA V100 GPUs and up to 8 NVIDIA A100 GPUs on the Gadi supercomputer for large inputs, and slightly worse but still near-optimal performance and scaling for smaller inputs, as well as favourable comparisons to state-of-the-art libraries. As a highlight, over 85% of peak performance (135 TFLOP/s) is achieved on a node with 8 A100 GPUs for the RI-MP2 calculation of a cluster of 150 water molecules.

The distributed memory version, achieves outstanding weak and strong scalability on the Perlmutter supercomputer, achieving 83% of peak performance (11.8 PFLOP/s) with a 95%

Table 2: Performance of the distributed RI-MP2 algorithm on a variety of inputs of practical interest.

Molecule	n_o	n_v	n_{aux}	Nodes	Wall-Time (s)	FLOP rate (TFLOP/s)	% Peak
$(\text{benzene})_{20}$	420	1980	9720	16	9.191	799.815	64%
$(\text{benzene})_{20}$	420	1980	9720	32	5.545	1328.97	53%
$(\text{paracetamol})_{20}$	800	3400	17220	16	126.136	1051.35	84%
$(\text{paracetamol})_{20}$	800	3400	17220	32	79.767	1694.21	68%
$([C_2\text{mim}]^+[BF_4]^-)_{20}$	1020	3980	20460	16	325.736	1066.54	85%
$([C_2\text{mim}]^+[BF_4]^-)_{20}$	1020	3980	20460	32	171.209	2066.55	83%
$([EtNH_3]^+[NO_3]^-)_{20}$	580	2320	11640	16	25.191	886.5	71%
$([EtNH_3]^+[NO_3]^-)_{20}$	580	2320	11640	32	26.221	871.68	35%
$([Gdm]^+[BF_4]^-))_{20}$	740	2560	13680	16	52.849	972.056	78%
$([Gdm]^+[BF_4]^-)_{20}$	740	2560	13680	32	34.502	1488.27	60%
$([Gdm]^+Cl^-)_{20}$	500	1680	8880	16	8.548	791.971	63%
$([Gdm]^+Cl^-)_{20}$	500	1680	8880	32	5.267	1302.14	52%
$([Gdm]^+[EtSO_3]^-)_{20}$	900	3280	16980	16	146.017	1049.3	84%
$([Gdm]^+[EtSO_3]^-)_{20}$	900	3280	16980	32	81.529	1911.56	77%

parallel efficiency using 720 80GB A100 GPUs over 180 nodes. The weak scaling of the code is not ideal due to the reduction of data across nodes compared to the expensive computational step. However, this could be ameliorated with further development.

A limitation in the algorithm is imposed by the steep memory requirement of storing the three-index tensors in Host memory, and the reliance on GPUs. For HPC systems with hundreds of gigabytes of available memory and available GPUs this is not an issue, but this renders our algorithm inaccessible for ordinary workstations.

Overall, the new distributed memory RI-MP2 algorithm presented in this paper shows that GPU oriented design can lead programs to fully exploit the hardware and provide thereby incredible performance and speedups over past implementations.

Acknowledgement GMJB thanks the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility, NERSC award ERCAP0026496 for computing resources on the Perlmutter supercomputer. GMJB thanks the Pawsey Centre for Extreme Scale Readiness (PaCER) for both funding and computing resources. GMBJ thanks the National Computational Merit Allocation Scheme (NCMAS) and the Australian National University Merit Allocation Scheme (ANUMAS) for their respective computational resources grants on the Gadi supercomputer at National Computational Infrastructure and on the Setonix supercomputer at the Pawsey Supercomputing Centre.

References

(1) Møller, C.; Plesset, M. S. Note on an Approximation Treatment for Many-Electron Systems. *Phys. Rev.* **1934**, *46*, 618–622.

(2) Helgaker, T.; Gauss, J.; Jørgensen, P.; Olsen, J. The prediction of molecular equilibrium structures by the standard electronic wave functions. *J. Chem. Phys.* **1997**, *106*, 6430–6440.

(3) Witte, J.; Goldey, M.; Neaton, J. B.; Head-Gordon, M. Beyond Energies: Geometries of Nonbonded Molecular Complexes as Metrics for Assessing Electronic Structure Approaches. *J. Chem. Theory Comput.* **2015**, *11*, 1481–1492, PMID: 26574359.

(4) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. A fifth-order perturbation comparison of elec-

tron correlation theories. *Chem. Phys. Lett.* **1989**, *157*, 479–483.

(5) Sinnokrot, M. O.; Sherrill, C. D. Highly Accurate Coupled Cluster Potential Energy Curves for the Benzene Dimer: Sandwich, T-Shaped, and Parallel-Displaced Configurations. *J. Phys. Chem. A* **2004**, *108*, 10200–10207.

(6) Jurečka, P.; Šponer, J.; Černý, J.; Hobza, P. Benchmark database of accurate (MP2 and CCSD(T) complete basis set limit) interaction energies of small model complexes, DNA base pairs, and amino acid pairs. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1985–1993.

(7) Sherrill, C. D.; Takatani, T.; Hohenstein, E. G. An Assessment of Theoretical Methods for Nonbonded Interactions: Comparison to Complete Basis Set Limit Coupled-Cluster Potential Energy Curves for the Benzene Dimer, the Methane Dimer, Benzene-Methane, and Benzene-H2S. *J. Phys. Chem. A* **2009**, *113*, 10146–10159, PMID: 19689152.

(8) Tkatchenko, A.; DiStasio, J., Robert A.; Head-Gordon, M.; Scheffler, M. Dispersion-corrected Møller–Plesset second-order perturbation theory. *J. Chem. Phys.* **2009**, *131*, 094106.

(9) Tomasz Janowski, A. R. F.; Pulay, P. Accurate correlated calculation of the intermolecular potential surface in the coronene dimer. *Mol. Phys.* **2010**, *108*, 249–257.

(10) Hobza, P. Calculations on Noncovalent Interactions and Databases of Benchmark Interaction Energies. *Acc. Chem. Res.* **2012**, *45*, 663–672, PMID: 22225511.

(11) Byrd, E. F. C.; Sherrill, C. D.; Head-Gordon, M. The Theoretical Prediction of Molecular Radical Species: a Systematic Study of Equilibrium Geometries and Harmonic Vibrational Frequencies. *J. Phys. Chem. A* **2001**, *105*, 9736–9747.

(12) Lochan, R. C.; Shao, Y.; Head-Gordon, M. Quartic-Scaling Analytical Energy Gradient of Scaled Opposite-Spin Second-Order Møller-Plesset Perturbation Theory. *J. Chem. Theory Comput.* **2007**, *3*, 988–1003, PMID: 26627418.

(13) Tentscher, P. R.; Arey, J. S. Geometries and Vibrational Frequencies of Small Radicals: Performance of Coupled Cluster and More Approximate Methods. *J. Chem. Theory Comput.* **2012**, *8*, 2165–2179, PMID: 26593847.

(14) Loipersberger, M.; Bertels, L. W.; Lee, J.; Head-Gordon, M. Exploring the Limits of Second- and Third-Order Møller–Plesset Perturbation Theories for Noncovalent Interactions: Revisiting MP2.5 and Assessing the Importance of Regularization and Reference Orbitals. *J. Chem. Theory Comput.* **2021**, *17*, 5582–5599, PMID: 34382394.

(15) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *J. Chem. Phys.* **2010**, *132*, 154104.

(16) Grimme, S. Density functional theory with London dispersion corrections. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2011**, *1*, 211–228.

(17) Vydrov, O. A.; Van Voorhis, T. Nonlocal van der Waals density functional: The simpler the better. *J. Chem. Phys.* **2010**, *133*, 244103.

(18) Ehrlich, S.; Moellmann, J.; Grimme, S. Dispersion-Corrected Density Functional Theory for Aromatic Interactions in Complex Systems. *Acc. Chem. Res.* **2013**, *46*, 916–926, PMID: 22702344.

(19) Goerigk, L.; Grimme, S. Double-hybrid density functionals. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2014**, *4*, 576–600.

(20) Mardirossian, N.; Head-Gordon, M. Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals. *Mol. Phys.* **2017**, *115*, 2315–2372.

(21) Kohn, W.; Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **1965**, *140*, A1133.

(22) Neese, F.; Schwabe, T.; Kossmann, S.; Schirmer, B.; Grimme, S. Assessment of Orbital-Optimized, Spin-Component Scaled Second-Order Many-Body Perturbation Theory for Thermochemistry and Kinetics. *J. Chem. Theory Comput.* **2009**, *5*, 3060–3073, PMID: 26609985.

(23) Bozkaya, U.; Sherrill, C. D. Orbital-optimized MP2.5 and its analytic gradients: Approaching CCSD(T) quality for noncovalent interactions. *J. Chem. Phys.* **2014**, *141*, 204105.

(24) Lee, J.; Head-Gordon, M. Regularized Orbital-Optimized Second-Order Møller–Plesset Perturbation Theory: A Reliable Fifth-Order-Scaling Electron Correlation Model with Orbital Energy Dependent Regularizers. *J. Chem. Theory Comput.* **2018**, *14*, 5203–5219, PMID: 30130398.

(25) Rettig, A.; Hait, D.; Bertels, L. W.; Head-Gordon, M. Third-Order Møller–Plesset Theory Made More Useful? The Role of Density Functional Theory Orbitals. *J. Chem. Theory Comput.* **2020**, *16*, 7473–7489, PMID: 33161713.

(26) Grimme, S. Improved second-order Møller–Plesset perturbation theory by separate scaling of parallel- and antiparallel-spin pair correlation energies. *J. Chem. Phys.* **2003**, *118*, 9095–9102.

(27) Gerenkamp, M.; Grimme, S. Spin-component scaled second-order Møller–Plesset perturbation theory for the calculation of molecular geometries and harmonic vibrational frequencies. *Chem. Phys. Lett.* **2004**, *392*, 229–235.

(28) Jung, Y.; Lochan, R. C.; Dutoi, A. D.; Head-Gordon, M. Scaled opposite-spin second order Møller–Plesset correlation energy: An economical electronic structure method. *J. Chem. Phys.* **2004**, *121*, 9793–9802.

(29) Lochan, R. C.; Jung, Y.; Head-Gordon, M. Scaled Opposite Spin Second Order Møller-Plesset Theory with Improved Physical Description of Long-Range Dispersion Interactions. *J. Phys. Chem. A* **2005**, *109*, 7598–7605, PMID: 16834130.

(30) Grimme, S.; Goerigk, L.; Fink, R. F. Spin-component-scaled electron correlation methods. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012**, *2*, 886–906.

(31) Assfeld, X.; Almlöf, J. E.; Truhlar, D. G. Degeneracy-corrected perturbation theory for electronic structure calculations. *Chem. Phys. Lett.* **1995**, *241*, 438–444.

(32) Stück, D.; Head-Gordon, M. Regularized orbital-optimized second-order perturbation theory. *J. Chem. Phys.* **2013**, *139*, 244109.

(33) Ohnishi, Y.-y.; Ishimura, K.; Ten-no, S. Interaction Energy of Large Molecules from Restrained Denominator MP2-F12. *J. Chem. Theory Comput.* **2014**, *10*, 4857–4861, PMID: 26584372.

(34) Evangelista, F. A. A driven similarity renormalization group approach to quantum many-body problems. *J. Chem. Phys.* **2014**, *141*, 054109.

(35) Rostam M. Razban, D. S.; Head-Gordon, M. Addressing first derivative discontinuities in orbital-optimised

opposite-spin scaled second-order perturbation theory with regularisation. *Mol. Phys.* **2017**, *115*, 2102–2109.

(36) Shee, J.; Loipersberger, M.; Rettig, A.; Lee, J.; Head-Gordon, M. Regularized Second-Order Møller–Plesset Theory: A More Accurate Alternative to Conventional MP2 for Noncovalent Interactions and Transition Metal Thermochemistry for the Same Computational Cost. *J. Phys. Chem. Lett.* **2021**, *12*, 12084–12097, PMID: 34910484.

(37) Zhao, Y.; Lynch, B. J.; Truhlar, D. G. Doubly Hybrid Meta DFT: New Multi-Coefficient Correlation and Density Functional Methods for Thermochemistry and Thermochemical Kinetics. *J. Phys. Chem. A* **2004**, *108*, 4786–4791.

(38) Grimme, S. Semiempirical hybrid density functional with perturbative second-order correlation. *J. Chem. Phys.* **2006**, *124*, 034108.

(39) Tarnopolsky, A.; Karton, A.; Sertchook, R.; Vuzman, D.; Martin, J. M. L. Double-Hybrid Functionals for Thermochemical Kinetics. *J. Phys. Chem. A* **2008**, *112*, 3–8, PMID: 18081266.

(40) Chai, J.-D.; Head-Gordon, M. Long-range corrected double-hybrid density functionals. *J. Chem. Phys.* **2009**, *131*, 174105.

(41) Karton, A.; Tarnopolsky, A.; Lamère, J.-F.; Schatz, G. C.; Martin, J. M. L. Highly Accurate First-Principles Benchmark Data Sets for the Parametrization and Validation of Density Functional and Other Approximate Methods. Derivation of a Robust, Generally Applicable, Double-Hybrid Functional for Thermochemistry and Thermochemical Kinetics. *J. Phys. Chem. A* **2008**, *112*, 12868–12886, PMID: 18714947.

(42) Zhang, I. Y.; Xu, X. Doubly hybrid density functional for accurate description of thermochemistry, thermochemical kinetics and nonbonded interactions. *Int. Rev. Phys. Chem.* **2011**, *30*, 115–160.

(43) Ji, H.; Shao, Y.; Goddard, W. A.; Jung, Y. Analytic Derivatives of Quartic-Scaling Doubly Hybrid XYGJ-OS Functional: Theory, Implementation, and Benchmark Comparison with M06-2X and MP2 Geometries for Nonbonded Complexes. *J. Chem. Theory Comput.* **2013**, *9*, 1971–1976, PMID: 23671408.

(44) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. A look at the density functional theory zoo with the advanced GMTKN55 database for general main group thermochemistry, kinetics and noncovalent interactions. *Phys. Chem. Chem. Phys.* **2017**, *19*, 32184–32215.

(45) Mardirossian, N.; Head-Gordon, M. Survival of the most transferable at the top of Jacob's ladder: Defining and testing the ωB97M(2) double hybrid density functional. *J. Chem. Phys.* **2018**, *148*, 241736.

(46) Santra, G.; Cho, M.; Martin, J. M. L. Exploring Avenues beyond Revised DSD Functionals: I. Range Separation, with xDSD as a Special Case. *J. Phys. Chem. A* **2021**, *125*, 4614–4627, PMID: 34009986.

(47) Santra, G.; Martin, J. M. L. Do Double-Hybrid Functionals Benefit from Regularization in the PT2 Term? Observations from an Extensive Benchmark. *J. Phys. Chem. Lett.* **2022**, *13*, 3499–3506, PMID: 35417181.

(48) Wappett, D. A.; Goerigk, L. Benchmarking Density Functional Theory Methods for Metalloenzyme Reactions: The Introduction of the MME55 Set. *J. Chem. Theory Comput.* **2023**, *19*, 8365–8383, PMID: 37943578.

(49) Saebo, S.; Pulay, P. Local Treatment of Electron Correlation. *Annu. Rev. Phys. Chem.* **1993**, *44*, 213–236.

(50) Schütz, M.; Hetzer, G.; Werner, H.-J. Low-order scaling local electron correlation methods. I. Linear scaling local MP2. *J. Chem. Phys.* **1999**, *111*, 5691–5705.

(51) Lee, M. S.; Maslen, P. E.; Head-Gordon, M. Closely approximating second-order Møller–Plesset perturbation theory with a local triatomics in molecules model. *J. Chem. Phys.* **2000**, *112*, 3592–3601.

(52) 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.

(53) DiStasio, R. A.; Jung, Y.; Head-Gordon, M. A resolution-of-the-identity implementation of the local triatomics-in-molecules model for second-order Møller- Plesset perturbation theory with application to alanine tetrapeptide conformational energies. *J. Chem. Theory Comput.* **2005**, *1*, 862–876.

(54) Mochizuki, Y.; Koikegami, S.; Nakano, T.; Amari, S.; Kitaura, K. Large scale MP2 calculations with fragment molecular orbital scheme. *Chem. Phys. Lett.* **2004**, *396*, 473–479.

(55) Doser, B.; Lambrecht, D. S.; Kussmann, J.; Ochsenfeld, C. Linear-scaling atomic orbital-based second-order Møller–Plesset perturbation theory by rigorous integral screening criteria. *J. Chem. Phys.* **2009**, *130*, 064107.

(56) 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*, 034108.

(57) Baudin, P.; Ettenhuber, P.; Reine, S.; Kristensen, K.; Kjærgaard, T. Efficient linear-scaling second-order Møller-Plesset perturbation theory: The divide–expand–consolidate RI-MP2 model. *J. Chem. Phys.* **2016**, *144*, 054102.

(58) 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.

(59) Song, C.; Martínez, T. J. Atomic orbital-based SOS-MP2 with tensor hypercontraction. I. GPU-based tensor construction and exploiting sparsity. *J. Chem. Phys.* **2016**, *144*, 174111.

(60) Kjærgaard, T.; Baudin, P.; Bykov, D.; Eriksen, J. J.; Ettenhuber, P.; Kristensen, K.; Larkin, J.; Liakh, D.; Pawlowski, F.; Vose, A.; Wang, Y. M.; Jørgensen, P. Massively parallel and linear-scaling algorithm for second-order Møller–Plesset perturbation theory applied to the study of supramolecular wires. *Comput. Phys. Commun.* **2017**, *212*, 152–160.

(61) Bykov, D.; Kjaergaard, T. The GPU-enabled divide-expand-consolidate RI-MP2 method (DEC-RI-MP2). *J. Comput. Chem.* **2017**, *38*, 228–237.

(62) Barca, G. M.; Galvez Vallejo, J. L.; Poole, D. L.; Alkan, M.; Stocks, R.; Rendell, A. P.; Gordon, M. S. Enabling Large-Scale Correlated Electronic Structure Calculations: Scaling the RI-MP2 Method on Summit. Int. Conf. High Perform. Comput. Netw. Storage Anal. New York, NY, USA, 2021; pp 1–15.

(63) Barca, G. M. J.; Snowdon, C.; Vallejo, J. L. G.; Kazemian, F.; Rendell, A. P.; Gordon, M. S. Scaling Correlated Fragment Molecular Orbital Calculations on Summit. Int. Conf. High Perform. Comput. Netw. Storage Anal. 2022; pp 1–14.

(64) Feyereisen, M.; 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.

(65) Weigend, F.; Häser, M. RI-MP2: First derivatives and global consistency. *Theor. Chem. Acc.* **1997**, *97*, 331–340.

(66) Weigend, F.; Häser, M.; Patzelt, H.; Ahlrichs, R. RI-MP2: optimized auxiliary basis sets and demonstration of efficiency. *Chem. Phys. Lett.* **1998**, *294*, 143–152.

(67) Weigend, F.; Köhn, A.; Hättig, C. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *The Journal of chemical physics* **2002**, *116*, 3175–3183.

(68) Frisch, M. J.; Head-Gordon, M.; Pople, J. A. Semi-direct algorithms for the MP2 energy and gradient. *Chemical physics letters* **1990**, *166*, 281–289.

(69) Bernholdt, D. E.; Harrison, R. J. Large-scale correlated electronic structure calculations: the RI-MP2 method on parallel computers. *Chemical Physics Letters* **1996**, *250*, 477–484.

(70) Katouda, M.; Kobayashi, M.; Nakai, H.; Nagase, S. Two-Level hierarchical parallelization of second-order Møller-Plesset perturbation calculations in divide-and-conquer method. *J. Comput. Chem.* **2011**, *32*, 2756–2764.

(71) Del Ben, M.; Hutter, J.; VandeVondele, J. Second-order Møller-Plesset perturbation theory in the condensed phase: An efficient and massively parallel Gaussian and plane waves approach. *J. Chem. Theory Comput.* **2012**, *8*, 4177–4188.

(72) Kristensen, K.; Kjaergaard, T.; Høyvik, I.-M.; Ettenhuber, P.; Jørgensen, P.; Jansik, B.; Reine, S.; Jakowski, J. The divide-expand-consolidate MP2 scheme goes massively parallel. *Mol. Phys.* **2013**, *111*, 1196–1201.

(73) Katouda, M.; Nakajima, T. MPI/OpenMP hybrid parallel algorithm of resolution of identity second-order Møller-Plesset perturbation calculation for massively parallel multicore supercomputers. *J. Chem. Theory Comput.* **2013**, *9*, 5373–5380.

(74) 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.

(75) 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.

(76) 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.

(77) Martinez-Martinez, L. A.; Amador-Bedolla, C. GPU algorithm for the scaled opposite-spin (SOS) MP2 energy evaluation. *J. Mex. Chem. Soc.* **2017**, *61*, 60–66.

(78) Pham, B. Q.; Gordon, M. S. Hybrid Distributed/Shared Memory Model for the RI-MP2 Method in the Fragment Molecular Orbital Framework. *Journal of Chemical Theory and Computation* **2019**, *15*, 5252–5258, PMID: 31509402.

(79) Barca, G. M. J.; McKenzie, S. C.; Bloomfield, N. J.; Gilbert, A. T. B.; Gill, P. M. W. Q-MP2-OS: Møller–Plesset Correlation Energy by Quadrature. *Journal of Chemical Theory and Computation* **2020**, *16*, 1568–1577, PMID: 31972086.

(80) Kjærgaard, T.; Baudin, P.; Bykov, D.; Eriksen, J. J.; Ettenhuber, P.; Kristensen, K.; Larkin, J.; Liakh, D.; Pawlowski, F.; Vose, A.; Wang, Y. M.; Jørgensen, P. Massively parallel and linear-scaling algorithm for second-order Møller–Plesset perturbation theory applied to the study of supramolecular wires. *Computer Physics Communications* **2017**, *212*, 152–160.

(81) Barca, G. M. J.; Vallejo, J. L. G.; Poole, D. L.; Alkan, M.; Stocks, R.; Rendell, A. P.; Gordon, M. S. Enabling large-scale correlated electronic structure calculations: scaling the RI-MP2 method on summit. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York, NY, USA, 2021; pp 1–15.

(82) Barca, G. M. J.; Snowdon, C.; Vallejo, J. L. G.; Kazemian, F.; Rendell, A. P.; Gordon, M. S. Scaling Correlated Fragment Molecular Orbital Calculations on Summit. 2022; pp 72–85, ISSN: 2167-4337.

(83) Collins, M. A.; Bettens, R. P. A. Energy-Based Molecular Fragmentation Methods. *Chemical Reviews* **2015**, *115*, 5607–5642.

(84) Yasuda, K. Two-electron integral evaluation on the graphics processor unit. *Journal of Computational Chemistry* **2008**, *29*, 334–342.

(85) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. Accelerating Resolution-of-the-Identity Second-Order Moller-Plesset Quantum Chemistry Calculations with Graphical Processing Units. *The Journal of Physical Chemistry A* **2008**, *112*, 2049–2057.

(86) Asadchev, A.; Gordon, M. S. Mixed-precision evaluation of two-electron integrals by Rys quadrature. *Computer Physics Communications* **2012**, *183*, 1563–1567.

(87) Asadchev, A.; Gordon, M. S. New Multithreaded Hybrid CPU/GPU Approach to Hartree–Fock. *Journal of Chemical Theory and Computation* **2012**, *8*, 4166–4176.

(88) Asadchev, A.; Allada, V.; Felder, J.; Bode, B. M.; Gordon, M. S.; Windus, T. L. Uncontracted Rys Quadrature Implementation of up to G Functions on Graphical Processing Units. *Journal of Chemical Theory and Computation* **2010**, *6*, 696–704.

(89) Ufimtsev, I. S.; Martínez, T. J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation. *Journal of Chemical Theory and Computation* **2008**, *4*, 222–231.

(90) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *Journal of Chemical Theory and Computation* **2009**, *5*, 1004–1015.

(91) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *Journal of Chemical Theory and Computation* **2009**, *5*, 2619–2628.

(92) Yasuda, K.; Maruoka, H. Efficient calculation of two-electron integrals for high angular basis functions. *International Journal of Quantum Chemistry* **2014**, *114*, 543–552.

(93) Kussmann, J.; Ochsenfeld, C. Hybrid CPU/GPU Integral Engine for Strong-Scaling *Ab Initio* Methods. *Journal of Chemical Theory and Computation* **2017**, *13*, 3153–3159.

(94) Miao, Y.; Merz, K. M. Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units. *Journal of Chemical Theory and Computation* **2015**, *11*, 1449–1462.

(95) Feyereisen, M.; 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.

(96) Jung, Y.; Sodt, A.; Gill, P. M. W.; Head-Gordon, M. Auxiliary basis expansions for large-scale electronic structure calculations. *Proceedings of the National Academy of Sciences* **2005**, *102*, 6692–6697, Publisher: Proceedings of the National Academy of Sciences.

(97) Gill, P. M. W.; Head-Gordon, M.; Pople, J. A. An efficient algorithm for the generation of two-electron repulsion integrals over gaussian basis functions. *International Journal of Quantum Chemistry* **1989**, *36*, 269–280.

(98) Obara, S.; Saika, A. Efficient recursive computation of molecular integrals over Cartesian Gaussian functions. *The Journal of Chemical Physics* **1986**, *84*, 3963–3974.

(99) Barca, G. M. J.; Alkan, M.; Galvez-Vallejo, J. L.; Poole, D. L.; Rendell, A. P.; Gordon, M. S. Faster Self-Consistent Field (SCF) Calculations on GPU Clusters. *Journal of Chemical Theory and Computation* **2021**, *17*, 7486–7503, PMID: 34780186.

(100) Galvez Vallejo, J. L.; Barca, G. M.; Gordon, M. S. High-performance GPU-accelerated evaluation of electron repulsion integrals. *Molecular Physics* **2022**, e2112987.

(101) Dongarra, J.; Croz, J.; Hammarling, S.; Duff, I. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software (TOMS)* **1990**, *16*, 1–17.

(102) Barca, G. M. J.; Vallejo, J. L. G.; Poole, D. L.; Alkan, M.; Stocks, R.; Rendell, A. P.; Gordon, M. S. Enabling large-scale correlated electronic structure calculations: scaling the RI-MP2 method on summit. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York, NY, USA, 2021; pp 1–15.

(103) Barca, G. M. J.; Snowdon, C.; Vallejo, J. L. G.; Kazemian, F.; Rendell, A. P.; Gordon, M. S. Scaling Correlated Fragment Molecular Orbital Calculations on Summit. 2022; pp 72–85, ISSN: 2167-4337.

(104) Kwack, J.; Bertoni, C.; Pham, B.; Larkin, J. Performance of the RI-MP2 Fortran Kernel of GAMESS on GPUs via Directive-Based Offloading with Math Libraries. Accelerator Programming Using Directives: 6th International Workshop, WACCPD 2019, Denver, CO, USA, November 18, 2019, Revised Selected Papers. Berlin, Heidelberg, 2019; p 91–113.

(105) Baudin, P.; Ettenhuber, P.; Reine, S.; Kristensen, K.; Kjærgaard, T. Efficient linear-scaling second-order Møller-Plesset perturbation theory: The divide–expand–consolidate RI-MP2 model. *The Journal of Chemical Physics* **2016**, *144*, 054102.

(106) Barca, G. M. J.; Bertoni, C.; Carrington, L.; Datta, D.; De Silva, N.; Deustua, J. E.; Fedorov, D. G.; Gour, J. R.; Gunina, A. O.; Guidez, E.; Harville, T.; Irle, S.; Ivanic, J.; Kowalski, K.; Leang, S. S.; Li, H.; Li, W.; Lutz, J. J.; Magoulas, I.; Mato, J.;

Mironov, V.; Nakata, H.; Pham, B. Q.; Piecuch, P.; Poole, D.; Pruitt, S. R.; Rendell, A. P.; Roskop, L. B.; Ruedenberg, K.; Sattasathuchana, T.; Schmidt, M. W.; Shen, J.; Slipchenko, L.; Sosonkina, M.; Sundriyal, V.; Tiwari, A.; Galvez Vallejo, J. L.; Westheimer, B.; Włoch, M.; Xu, P.; Zahariev, F.; Gordon, M. S. Recent developments in the general atomic and molecular electronic structure system. *The Journal of Chemical Physics* **2020**, *152*, 154102.

(107) Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T. B.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I.; Khaliullin, R. Z.; Kús, T.; Landau, A.; Liu, J.; Proynov, E. I.; Rhee, Y. M.; Richard, R. M.; Rohrdanz, M. A.; Steele, R. P.; Sundstrom, E. J.; Woodcock III, H. L.; Zimmerman, P. M.; Zuev, D.; Albrecht, B.; Alguire, E.; Austin, B.; Beran, G. J. O.; Bernard, Y. A.; Berquist, E.; Brandhorst, K.; Bravaya, K. B.; Brown, S. T.; Casanova, D.; Chang, C.-M.; Chen, Y.; Chien, S. H.; Closser, K. D.; Crittenden, D. L.; Diedenhofen, M.; DiStasio Jr., R. A.; Dop, H.; Dutoi, A. D.; Edgar, R. G.; Fatehi, S.; Fusti-Molnar, L.; Ghysels, A.; Golubeva-Zadorozhnaya, A.; Gomes, J.; Hanson-Heine, M. W. D.; Harbach, P. H. P.; Hauser, A. W.; Hohenstein, E. G.; Holden, Z. C.; Jagau, T.-C.; Ji, H.; Kaduk, B.; Khistyaev, K.; Kim, J.; Kim, J.; King, R. A.; Klunzinger, P.; Kosenkov, D.; Kowalczyk, T.; Krauter, C. M.; Lao, K. U.; Laurent, A.; Lawler, K. V.; Levchenko, S. V.; Lin, C. Y.; Liu, F.; Livshits, E.; Lochan, R. C.; Luenser, A.; Manohar, P.; Manzer, S. F.; Mao, S.-P.; Mardirossian, N.; Marenich, A. V.; Maurer, S. A.; Mayhall, N. J.; Oana, C. M.; Olivares-Amaya, R.; O'Neill, D. P.; Parkhill, J. A.; Perrine, T. M.; Peverati, R.; Pieniazek, P. A.; Prociuk, A.; Rehn, D. R.; Rosta, E.; Russ, N. J.; Sergueev, N.; Sharada, S. M.; Sharmaa, S.; Small, D. W.; Sodt, A.; Stein, T.; Stück, D.; Su, Y.-C.; Thom, A. J. W.; Tsuchimochi, T.; Vogt, L.; Vydrov, O.; Wang, T.; Watson, M. A.; Wenzel, J.; White, A.; Williams, C. F.; Vanovschi, V.; Yeganeh, S.; Yost, S. R.; You, Z.-Q.; Zhang, I. Y.; Zhang, X.; Zhou, Y.; Brooks, B. R.; Chan, G. K. L.; Chipman, D. M.; Cramer, C. J.; Goddard III, W. A.; Gordon, M. S.; Hehre, W. J.; Klamt, A.; Schaefer III, H. F.; Schmidt, M. W.; Sherrill, C. D.; Truhlar, D. G.; Warshel, A.; Xua, X.; Aspuru-Guzik, A.; Baer, R.; Bell, A. T.; Besley, N. A.; Chai, J.-D.; Dreuw, A.; Dunietz, B. D.; Furlani, T. R.; Gwaltney, S. R.; Hsu, C.-P.; Jung, Y.; Kong, J.; Lambrecht, D. S.; Liang, W.; Ochsenfeld, C.; Rassolov, V. A.; Slipchenko, L. V.; Subotnik, J. E.; Van Voorhis, T.; Herbert, J. M.; Krylov, A. I.; Gill, P. M. W.; Head-Gordon, M. Advances in molecular quantum chemistry contained in the Q-Chem 4 program package. *Mol. Phys.* **2015**, *113*, 184–215.

(108) Neese, F. The ORCA program system. *WIRES Comput. Molec. Sci.* **2012**, *2*, 73–78.

TOC Graphic