Matrix Is All You Need: Rearchitecting Quantum Chemistry to Scale on AI Accelerators

Haozhi Han*
School of Computer Science
Peking University
Beijing, China
haozhi.han@stu.pku.edu.cn

Kun Li†
Microsoft Research
Beijing, China
kunli@microsoft.com

Fusong Ju
Microsoft Research
Beijing, China
fusongju@microsoft.com

Qi Li*
University of Science and Technology of China
Hefei, China
liqi123@mail.ustc.edu.cn

Hong An
University of Science and Technology of China
Hefei, China
han@ustc.edu.cn

Yifeng Chen
School of Computer Science
Peking University
Beijing, China
cyf@pku.edu.cn

Yunquan Zhang
Chinese Academy of Sciences
Beijing, China
zyq@ict.ac.cn

Ting Cao
Tsinghua University
Beijing, China
tingcao@mail.tsinghua.edu.cn

Mao Yang
Microsoft Research
Beijing, China
maoyang@microsoft.com

Abstract

Scientific computing remains fundamentally misaligned with the execution paradigm of modern AI accelerators, which rely on structured, low-precision matrix operations for performance and scalability. Quantum chemistry exemplifies this gap through three core scalability limits: irregular computational patterns, fragmented hardware utilization, and limited scientific reach.

In this work, we present Mako, a quantum chemistry system that rearchitects first-principles electronic structure computations as high-performance matrix-aligned kernels to scale on modern AI accelerators. Mako integrates three co-designed components: KernelMako reformulates ERI evaluation into structured matrix operations and leverages CUTLASS to enable transparent, composable MatMul pipelines; QuantMako introduces physics-informed, stage-aware quantization to exploit low-precision compute potential while preserving scientific fidelity; CompilerMako captures static execution patterns across angular momentum classes and automates kernel fusion and architecture-tuned specialization. Mako achieves up to 20× end-to-end speedup on high-angular-momentum basis sets. It sustains over 90% parallel efficiency on a single node and 70% across 64 GPUs, completing the accurate energy calculation of ubiquitin (1,231 atoms, def2-TZVP) from days to just 58 minutes. By restructuring quantum chemistry to align with the AI software-hardware stack, Mako demonstrates how scientific workloads can inherit deep learning–style scalability—scaling beyond the long-standing limits of irregularity, fragmentation, and complexity.

CCS Concepts

• **Computing methodologies** → **Parallel computing methodologies**; • **Applied computing** → *Chemistry*; • **Computer systems organization** → *Parallel architectures.*

Keywords

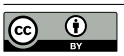
AI Accelerators, Quantum Chemistry, Matrix Multiplication, Density Functional Theory, CUTLASS

ACM Reference Format:
Haozhi Han, Kun Li, Fusong Ju, Qi Li, Hong An, Yifeng Chen, Yunquan Zhang, Ting Cao, and Mao Yang. 2025. Matrix Is All You Need: Rearchitecting Quantum Chemistry to Scale on AI Accelerators. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3712285.3759829

1 Introduction

Scaling scientific computing remains a persistent and fundamental challenge in high-performance computing (HPC) [38, 48, 60, 127]. Despite decades of domain-specific optimizations [52, 85, 96, 130], many critical applications—such as weather forecasting [101, 138], computational fluid dynamics [16, 157], and cosmological simulations [18, 163]—still struggle to scale efficiently on modern architectures. As hardware evolves toward increasingly dense and regular execution paradigms [87, 122, 162], the inherent irregularity of scientific workloads remains poorly aligned with accelerator-driven platforms [28, 56, 123, 159]. This widening gap between theoretical peak performance and practical scalability is becoming a systemic barrier to sustained performance gains [33, 128, 131, 139].

*Work done during an internship at Microsoft Research.
†Corresponding author (likungw@gmail.com).



SC '25, St Louis, MO, USA

ACM ISBN 979-8-4007-1466-5/25/11
https://doi.org/10.1145/3712285.3759829

Quantum chemistry serves as a microcosm of the broader scalability challenges in scientific computing [9, 47, 74]. As a cornerstone of materials science [54, 134], drug discovery [17, 80], and catalysis [45, 66], it relies on solving the Schrödinger equation to calculate first-principles electronic structure [77, 84]. Among its various methods [50, 67, 77, 143], Density Functional Theory (DFT) has emerged as the workhorse, balancing accuracy with computational feasibility [83, 94, 142]. However, despite its widespread adoption, DFT computations remain prohibitively expensive at scale [94, 105, 134], exposing three core limits to scientific computing scalability:

Irregular Computational Patterns. Scientific workloads, such as DFT, often involve irregular computation and memory access patterns, making them inherently resistant to unified optimization. [15, 19, 57, 58, 118]. Consequently, they scale suboptimally on modern architectures that favor regular and highly parallel operations to achieve peak performance [27, 30, 41, 180].

Fragmented Hardware Utilization. The irregularity of scientific computation impedes its adoption on AI accelerators such as NVIDIA Tensor Cores [30, 43, 95, 164]. These units, now powering 8 of the top 10 supercomputers on the TOP500 list, are optimized for low-precision matrix multiplication [24, 132, 169]. While isolated kernels—and in some cases, select application components—have been partially ported to these accelerators, full-scale restructuring of scientific applications remains uncommon, leaving substantial accelerator capacity untapped [23, 56, 89].

Limited Scientific Reach. Scalability bottlenecks across both algorithms and hardware fundamentally constrain scientific discovery. High-angular-momentum wavefunctions, essential for complex systems such as transition metals [61, 104, 155] and strongly correlated electrons [39, 72, 135], incur exponential costs that exceed compute and memory budgets [1, 150]. Existing methods either lack support or accept prohibitive runtimes [12, 37, 91, 100]. Moreover, this limitation extends beyond DFT itself to data generation for downstream AI4Science models, limiting their applicability to more complex quantum systems [64, 165, 166, 178].

These constraints contrast sharply with deep learning, which has redefined scalability through structured, low-precision matrix execution [65, 102, 147, 173]. The convergence toward matrix-aligned models—epitomized by the rise of Transformers [161]—has enabled deep learning to scale efficiently on modern AI accelerators [31, 70, 106, 141]. This disconnect raises a critical question: *Can we reimagine scientific computing to unlock deep learning–level scalability on modern AI accelerators?*

This paper introduces **Mako**[1], a quantum chemistry system that rearchitects first-principles electronic structure computations as high-performance matrix-aligned kernels to scale efficiently on modern AI accelerators.

Mako is grounded in three insights that reimagine scalability across computation, hardware, and scientific discovery:

Scientific Irregularity Masks a Matrix-Aligned Structure. With domain-aware abstraction, many irregular computations—such as those in DFT—can be reformulated into structured matrix multiplications (MatMul), unifying diverse operations under a scalable execution model [89, 107, 109, 140]. This enables scientific computing to leverage the deeply optimized matrix stack developed for deep learning—without relying on architecture-specific tuning [22, 82, 115, 144, 148, 149].

Precision Sensitivity Is More Selective Than It Seems. While scientific computing demands accuracy, extensive studies and commercial adoption [29, 73, 152] have shown that many stages of quantum chemistry are tolerant to reduced precision—much like quantization [44, 137] in deep learning [34, 73]. By coupling domain-validated numerical strategies [73, 90, 136] with low-precision TCUs, we unlock the high throughput of AI accelerators once thought incompatible with first-principles methods.

Structure-to-Compiler Shift Powers Scalable Discovery. Many scientific kernels—especially electron repulsion integrals—exhibit predictable execution patterns, making them ideal for compiler-level optimization [46, 152, 177]. By capturing this structure, we shift tuning from manual efforts to portable, automated compilation—enabling efficient hardware utilization and scaling DFT across architectures to reach more complex scientific frontiers.

Building on these insights, we revisit the entire DFT workflow through the lens of structured matrix computation. Exchange-correlation evaluation and Fock matrix diagonalization are inherently amenable to MatMul-based execution, the former via triple-product projection [21, 171] and the latter through iterative eigensolvers [2, 55, 174]. However, the evaluation of two-electron repulsion integrals (ERI) remains the dominant bottleneck—accounting for up to 80% of total runtime [13, 14, 167]—due to its complex data dependencies and irregular access patterns. To enable a full-stack solution for scalable DFT on AI accelerators, Mako introduces three core components:

KernelMako reformulates ERI evaluation into structured matrix operations and builds atop CUTLASS to enable transparent and composable MatMul pipelines. It introduces: *Implicit Instruction Parallelism*, embedding non-MatMul operations directly within MatMul kernels via instruction-level parallelism to maximize compute utilization; *Lightweight Layout Swizzle*, reshaping shared-memory tensors into MatMul-friendly formats without bank conflicts; *GEMM Coalescing*, merging consecutive GEMMs into single fused kernels to reduce memory traffic and latency. Together, KernelMako rearchitects irregular phases of DFT at the kernel level into a memory-efficient pipeline.

QuantMako leverages physics-informed knowledge to exploit the high throughput of low-precision AI accelerators. It employs: *Fine-Grained Quantization*, selectively downscaling precision across the DFT pipeline and scaling grouped data to enhance low-precision expressiveness; *Dual-Stage Accumulation*, preserving numerical fidelity through high-precision accumulation at both integral and Fock matrix stages; *Convergence-Aware Scheduling*, aligning quantization granularity with validated mixed-precision strategies for balancing efficiency and accuracy across the full workflow.

CompilerMako captures the static structure of ERI computation within a compiler-inspired framework to automate kernel fusion and hardware mapping across angular momentum orders. It features: *Reuse-Guided Planning*, which exploits static intermediates

[1] **Mako** is inspired by the mako shark—the fastest shark in the ocean—symbolizing speed, agility, and the drive to push computational frontiers. It also serves as an acronym for **M**atrix-**A**ligned **K**ernel **O**rchestration, encapsulating the system's core principle of orchestrating quantum chemistry into matrix-aligned kernels that fully exploit the AI software–hardware ecosystem.

and evaluates on-chip memory footprint to drive fusion strategy generation, improving locality and occupancy across diverse ERI variants; and *Architecture-Tuned Compilation*, which compiles MatMul kernels via CUTLASS primitives for portable high performance without manual retuning. Together, CompilerMako abstracts both hardware variability and algorithmic complexity, ensuring consistent scalability across diverse accelerator platforms.

We demonstrate that Mako's ERI kernels achieve up to 2.68× speedup over LibintX [6–8] on the A100 Tensor Core GPU. Mako delivers up to 15.3× speedup over leading GPU baselines while maintaining chemical accuracy within 1 mHartree [71, 90, 99, 156]. On challenging high-angular-momentum basis sets like def2-QZVP and cc-pVQZ, Mako achieves ~20× speedups over GPU4PySCF [86, 172], whereas QUICK lacks support for g-type functions. Mako sustains over 90% parallel efficiency on 8 GPUs and 70% across 64 GPUs. Notably, Mako enables the accurate energy calculation of ubiquitin (1,231 atoms) with the def2-TZVP basis set and reduces end-to-end runtime from days (QUICK [92, 93, 100]) to just 58 minutes, demonstrating both performance and practicality at scale.

To summarize, this work makes the following contributions:

- KernelMako rearchitects DFT into structured MatMul pipelines, enabling scalable execution for scientific applications long considered incompatible with AI accelerators.
- QuantMako delivers AI-inspired quantization into quantum chemistry, unlocking low-precision throughput while preserving scientific fidelity.
- CompilerMako pioneers a compiler-inspired framework that elevates quantum chemistry from handcrafted kernels to systems-level scheduling, scaling across high-angular-momentum regimes and heterogeneous architectures.
- Mako delivers the first system to bring deep learning–style scalability to quantum chemistry—fulfilling long-standing visions of harnessing AI accelerators for scientific computing [35, 36, 60].

2 Background and Challenges

2.1 Quantum Chemistry and Electron Repulsion Integrals

DFT has emerged as the *de facto* standard in quantum chemistry, offering a favorable trade-off between computational efficiency and predictive accuracy [10, 76, 94]. A typical DFT workflow comprises three computational stages: (1) the evaluation of ERI, (2) the application of exchange-correlation functionals, and (3) the diagonalization of the Fock matrix [20, 25, 40, 62, 124].

Among these, ERI evaluation is the primary bottleneck, often accounting for up to 80% of the total runtime in practical computations [13, 14, 167]. This stems from both its computational complexity, which scales as $O(N^4)$ concerning the number of basis functions N, and its irregular execution patterns, which hinder effective parallelization and accelerator utilization [14, 63, 103].

Each ERI quantifies the Coulomb interaction between two electrons and involves four basis functions:

$$(ab \mid cd) = \iint \frac{\phi_a(\mathbf{r}_1)\,\phi_b(\mathbf{r}_1)\,\phi_c(\mathbf{r}_2)\,\phi_d(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|} d\mathbf{r}_1 d\mathbf{r}_2. \quad (1)$$

Each basis function $\phi(\mathbf{r})$ is a contracted Gaussian function (CGF), expressed as a linear combination of multiple primitive Gaussian functions (PGFs):

$$\phi(\mathbf{r}) = \sum_{i=1}^{K} c_i \varphi_i(\mathbf{r}), \quad (2)$$

where K denotes the number of primitive functions in the contraction, c_i are contraction coefficients, and $\varphi_i(\mathbf{r})$ are the PGFs.

Both CGFs and their constituent PGFs are associated with an angular momentum quantum number l, which determines their orbital character: $l = 0$ (s), $l = 1$ (p), $l = 2$ (d), $l = 3$ (f), $l = 4$ (g), etc. Notably, ERIs sharing the same angular momentum pattern can follow shared evaluation routines, enabling partial reuse of computational paths.

To facilitate ERI evaluation, basis functions with identical PGF definitions and angular momentum values are grouped into shells. Each shell contains $2 * l + 1$ spherical CGFs. In practice, ERIs are computed over shell quartets—combinations of four shells denoted as $(AB|CD)$, which serve as the basic unit of ERI computation. This organization supports structured traversal and intermediate reuse, although the computation remains challenging to optimize on modern accelerator architectures.

2.2 AI Accelerators and the Matrix-Aligned Computing Stack

Modern AI accelerators, such as NVIDIA and AMD GPUs, Google TPUs, and custom AI ASICs [24, 69, 70, 132, 133], are increasingly architected around tensor core units designed to optimize matrix multiplication and accumulation (MMA), as expressed in Equation (3):

$$D_{m\times n} = A_{m\times k} \times B_{k\times n} + C_{m\times n}. \quad (3)$$

The proliferation of such hardware has catalyzed a co-evolving software ecosystem aimed at bridging algorithmic expressiveness and hardware efficiency. Frameworks such as CUTLASS, Triton, and TVM provide fine-grained control over kernel scheduling, memory hierarchy management, and precision tuning, thereby enabling high-performance kernel generation across diverse workloads [22, 82, 115, 144, 148, 149].

NVIDIA Tensor Cores are chosen as the baseline accelerator in this work due to their wide adoption and well-established software support. Table 1 summarizes the peak throughput of Tensor Cores versus CUDA cores on an A100 GPU. The performance advantage is especially pronounced in low-precision formats such as FP16 and BF16, where Tensor Cores outperform general-purpose cores by up to 4× [113].

To harness this hardware, **CUTLASS** (CUDA Templates for Linear Algebra Subroutines) provides a high-performance, modular CUDA C++ template library developed by NVIDIA. It decomposes computations into composable MMA-based primitives with support for fine-grained parallelism, flexible layout tuning, and architecture-aware scheduling. CUTLASS also supports GPU generations from Volta to Blackwell, enabling scalable performance across hardware revisions [115].

2.3 Opportunity: Giving Hardware What It Wants

The future of HPC lies not in ever-more intricate algorithmic handcrafting, but in expressing computations in forms that modern hardware can natively and efficiently execute [126]. Contemporary accelerators are fundamentally optimized for dense, regular, and matrix-aligned workloads—an execution paradigm that stands in stark contrast to the irregular, branching-intensive nature of many scientific applications, such as quantum chemistry. This gap stems not only from technical limitations but also from historical mismatches in architecture and computational methodologies. Scientific software has historically been architected without regard for the constraints—or opportunities—of today's accelerator-rich landscape.

This growing disconnect calls for a paradigm shift: rather than retrofitting legacy codes onto incompatible hardware, we ask whether the computations themselves can be structurally reimagined to match the strengths of modern accelerators. In this view, structure, numerical precision, and compiler orchestration are not limitations to be tolerated—they are active design dimensions to be exploited. By reformulating scientific workloads into structured, accelerator-friendly representations, we unlock a new frontier of co-design, where scientific computation no longer struggles against hardware evolution, but advances in tandem with it.

2.4 Challenges: Why Existing Work Can't Quench the Thirst With This Opportunity?

2.4.1 Partial Reformulations Miss the Target. Despite decades of optimization, traditional ERI evaluation methods—such as Head-Gordon–Pople [58], Obara–Saika [118], and Rys quadrature [129]—remain fundamentally misaligned with modern AI accelerators. These recursive and branch-heavy formulations exhibit irregular control flow and memory access, limiting vectorization and thread-level parallelism while incurring high register pressure, especially for high-angular-momentum integrals. Even optimized GPU implementations such as TeraChem [136, 167] and QUICK [92, 93, 100] restrict support to angular momentum ≤ 3 due to these structural inefficiencies.

In response, recent efforts—including LibintbX [6–8], and the SHARK engine [109]—have begun to express select ERI routines via BLAS or GPU MatMul backends, signaling a shift toward more accelerator-friendly execution. However, the resulting matrix operations are typically narrow in scope, fragmented across many small kernels, and disconnected from the broader computational structure. Such isolation undermines opportunities for operator fusion, layout-aware data orchestration, and instruction-level scheduling—key enablers of performance on modern accelerators designed for large, regular, and deeply pipelined matrix workloads.

Table 1: A100 GPU SPECIFICATIONS

Precision	Tensor Core	CUDA Core	Speedup
FP64	19.5 TFLOPS	9.7 TFLOPS	2x
FP32/TF32	156 TFLOPS	19.5 TFLOPS	8x
BF16	312 TFLOPS	78 TFLOPS	4x
FP16	312 TFLOPS	78 TFLOPS	4x

2.4.2 Precision-Agnostic Design Limits Acceleration. While AI accelerators excel at low-precision matrix operations, most scientific applications—DFT included—remain locked in full FP64 [13, 14, 46]. This rigidity stems not only from accuracy concerns but from a deeper architectural limitation: scientific codes are typically precision-agnostic, lacking the infrastructure to express or manage numerical fidelity at runtime. Although early efforts—such as Schwarz-bound–based truncation [136] and adaptive SCF thresholds [90]—have been adopted in some commercial packages, these approaches operate at a coarse algorithmic level, often substituting entire kernels with FP32 variants [73]. Critically, they overlook key enablers of deep learning–level quantization: semantic expressiveness [44, 88] and runtime adaptability [88, 137]. Current systems offer no way to declare tolerance bounds, encode phase sensitivity, or orchestrate precision scheduling—precluding any systematic use of low-precision hardware paths.

2.4.3 Code-Centric Paradigms Block System-Level Scalability. ERI development in quantum chemistry follows three paradigms: manual optimization [100], meta-programming [3, 6–8, 152], and code generation [46, 71, 177]. Each offers trade-offs between expressiveness, maintainability, and portability—but all struggle with scalability with increasing angular momentum. As angular momentum rises, both the number of ERI classes and the intra-class computational diversity grow combinatorially, making handcrafted or statically specialized kernels unsustainable. Meta-programming [3, 32, 152] offers concise abstraction but lacks automated awareness of hardware architecture. Code generators [46, 71, 177] emit optimized kernels but rely on rigid heuristics that struggle to generalize across GPU architectures or support high-angular momentum workloads. As a result, even state-of-the-art libraries either cap angular momentum support or rely on brute-force generation of thousands of brittle, non-reusable kernels—blocking generalizable system-level scalability.

3 Mako Design

3.1 KernelMako

Traditional ERI methods—such as recursive algorithms or quadrature-based schemes—exhibit highly irregular computation and memory access patterns, making them ill-suited for AI accelerators that thrive on regular, high-throughput matrix workloads [58, 118, 129]. To close the performance gap and fully exploit accelerator hardware, we revisit ERI evaluation from a matrix-aligned perspective.

Recent work has demonstrated that ERIs can, in principle, be reformulated as structured matrix operations [6, 109, 145]. One such reformulation leverages the McMurchie-Davidson (MMD) algorithm to express ERIs in terms of Hermite Gaussian integrals, enabling a basis transformation into MatMul [6, 97, 109]. As shown in Algorithm 1, the process begins by recursively computing a set of one-index Hermite integrals, which we refer to as r-integrals

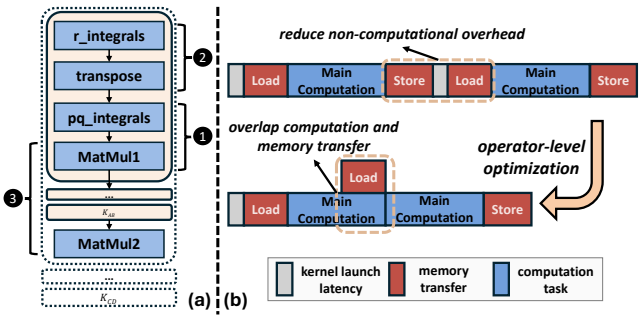


Figure 1: Operator-level Optimizations of KernelMako

$([\tilde{\mathbf{r}}]^{(m)})$, starting from the central quantity $[\mathbf{0}]^{(m)}$, which is associated with the Boys function $F_m(x)$:

$$[\tilde{\mathbf{r}}]^{(m)} \equiv \left(\frac{\partial}{\partial x_R}\right)^{\tilde{r}_x}\left(\frac{\partial}{\partial y_R}\right)^{\tilde{r}_y}\left(\frac{\partial}{\partial z_R}\right)^{\tilde{r}_z}[\mathbf{0}]^{(m)}. \tag{4}$$

The value of $F_m(x)$ is computed using the improved cubic Chebyshev interpolation method developed by Gill et al. [49], with the interpolation coefficients stored in a lookup table. These r-integrals are then constructed via the recursive relation:

$$[\tilde{\mathbf{r}}+\mathbf{1}_i]^{(m)} = \tilde{r}_i\,[\tilde{\mathbf{r}}-\mathbf{1}_i]^{(m+1)} + (P_i - Q_i)\,[\tilde{\mathbf{r}}]^{(m+1)}. \tag{5}$$

Subsequently, the two-index Hermite integrals $[\tilde{\mathbf{p}} \mid \tilde{\mathbf{q}}]$ are evaluated in parallel based on the precomputed one-index terms:

$$[\tilde{\mathbf{p}} \mid \tilde{\mathbf{q}}] \equiv (-1)^{l_q}[\tilde{\mathbf{p}}+\tilde{\mathbf{q}}]^{(0)} \tag{6}$$

Finally, the transformation from the Hermite to the AO basis is executed via matrix multiplication:

$$[\mathbf{ab} \mid \tilde{\mathbf{q}}] = \sum_{\tilde{\mathbf{p}}} E^{\tilde{\mathbf{p}}}_{\mathbf{ab}}[\tilde{\mathbf{p}} \mid \tilde{\mathbf{q}}], [\mathbf{ab} \mid \mathbf{cd}] = \sum_{\tilde{\mathbf{q}}} E^{\tilde{\mathbf{q}}}_{\mathbf{cd}}[\mathbf{ab} \mid \tilde{\mathbf{q}}]. \tag{7}$$

The basis transformation constitutes the predominant portion of the computational cost in this process [6]. While this offers a compelling foundation for matrix-based execution, naive implementation still leaves substantial performance on the table due to kernel launch overhead, uncoalesced memory traffic, and mismatched parallelism between fused operators.

To address these challenges, we introduce KernelMako, a fused kernel execution framework that performs ERI evaluation through fine-grained operator-level optimization. As shown in Figure 1 (a), the design of KernelMako is driven by three key techniques: ❶

Algorithm 1: The Matrix-Aligned Form of the MMD Algorithm

Input: Shell a, b, c, d; E_{AB}; E_{CD}
Output: $(ab|cd)$

```
for i ← 0 to K_CD do
    for j ← 0 to K_AB do
        [r] = compute r integrals() ;
        transpose([r]) ;
        [p|q] = compute pq integrals() ;
        basis transformation: (ab|q] += E_AB ⊗ [p|q] ;
    end
    basis transformation: (ab|cd) += (ab|q] ⊗ E_CD;
end
```

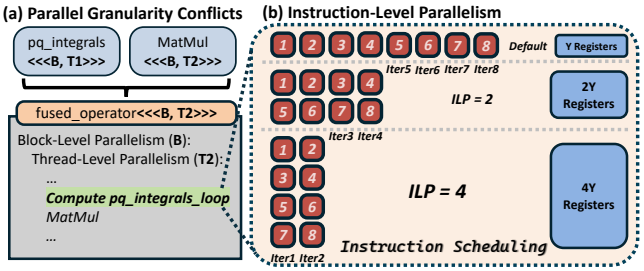


Figure 2: Implicit Instruction Parallelism

Implicit Instruction Parallelism that fuses MatMul and non-MatMul tasks; ❷ *Lightweight Layout Swizzle* that reshape internal structures into MatMul–friendly formats; and ❸ *GEMM Coalescing*, which fuse back-to-back GEMM operations to maximize AI accelerators throughput. Figure 1 (b) illustrates the resulting performance benefits in twofold: (1) it markedly reduces non-computational overhead, including memory traffic to global memory for storing and loading intermediate results and kernel launch latency, thereby minimizing the idle time of AI accelerator; (2) it enables the overlap of computation and memory loading, further boosting overall efficiency.

3.1.1 Implicit Instruction Parallelism. The locally optimal parallel granularity of different operators varies considerably, driven by the distinct computation patterns of each stage in ERI workflow. For instance, MatMul typically demands substantial on-chip resources that limit the number of threads per thread block ($T2$), whereas non-matmul operators (especially memory-intensive operators) often benefit from a higher degree of thread-level parallelism ($T1$). This mismatch gives rise to Parallel Granularity Conflicts ($T1 \neq T2$), which pose a significant challenge when fusing diverse operators into a single kernel.

Here, implicit Instruction-Level Parallelism (ILP), operating at a lower level than Thread-Level Parallelism (TLP) and ThreadBlock-Level Parallelism (BLP), is introduced to bridge the disparities in parallel granularity across different operators in ERI workflow. It enhances the flexibility of BLP and TLP, facilitating efficient fusion of MatMul and non-MatMul operators.

In the ERI computation of Mako, as shown in Figure 2, the *pq_integrals* operator is initially aligned with the coarser-grained parallelism of *MatMul* at both the BLP and TLP levels. However, this alignment reduces the explicit parallelism of *pq_integrals*, degrading its computational efficiency. To compensate for this limitation, Mako further applies implicit ILP within each thread of *pq_integrals* by restructuring its inner loop. As these loops exhibit no data dependencies, instruction scheduling can be leveraged to fuse and reorder multiple iterations for concurrent execution at the instruction level, effectively boosting ILP. Theoretically, even when the BLP_{new} and TLP_{new} of *pq_integrals* are suboptimal, increased ILP can elevate its total effective parallelism to a near-optimal level:

$$BLP_{new} \times TLP_{new} \times ILP \approx BLP_{optimal} \times TLP_{optimal} \tag{8}$$

3.1.2 Lightweight Layout Swizzle. The matrix operators in the ERI computations encounter unexpected challenges due to layout consistency in memory organization.

Blocked Layout (BL): All parameters corresponding to a specific shell quartet are stored contiguously in memory.

Striped Layout (SL): Each parameter of a shell quartet is separated in memory by a logical stride equal to the *quartet_batchsize*.

Efficient MatMul execution requires the pq integrals computed by the *pq_integrals* to be organized in BL. However, before *pq_integrals* are evaluated, the parameters are organized in SL to enable memory coalescing and maximize global memory bandwidth, as each thread is responsible for a single shell quartet. This mismatch creates the layout inconsistency that must be resolved to unlock optimal MatMul performance on AI accelerators. The most direct solution is to perform an explicit transpose of the data, but this approach incurs substantial I/O overhead due to heavy global memory traffic, thus degrading performance.

To address this challenge, we propose a lightweight layout transformation strategy that exploits shared memory (SMEM) bank properties and a swizzling technique to perform an in-place, entirely conflict-free transposition within SMEM. Crucially, this transformation is embedded within the computation of the r integrals—whose total volume is much smaller than that of the pq integrals—thus reducing memory pressure. The swizzling technique rearranges data access patterns to avoid bank conflicts during both load and store, without increasing SMEM usage. Swizzling reorders the relative positions of elements within each SMEM row, effectively optimizing memory access efficiency.

To formalize this mechanism, we define a bijective mapping between logical and physical coordinates:

$$(x_p, y_p) = f\,(x_l, y_l)\,. \tag{9}$$

The mapping must satisfy two key conditions: (1) it must be bijective to preserve one-to-one address correspondence, and (2) it must maintain the original domain ranges of both x and y to ensure indexing validity. Here, we omit the detailed proof, and the mapping relation eq(10) satisfies the aforementioned conditions. $\oplus$ denotes the bitwise exclusive OR (XOR) operation.

$$x_p = x_l \oplus y_l,\ y_p = y_l \tag{10}$$

After swizzling, as illustrated in Figure 3, data is stored in SMEM in a bank-conflict-free manner and subsequently accessed with the same access efficiency. This process effectively realizes a lightweight layout transformation in a memory-efficient way.

3.1.3 GEMM Coalescing. Given that high angular momentum ERIs present a major computational bottleneck and are essential to quantum chemical accuracy, we take them as a case study to demonstrate how Mako achieves extreme optimization of MatMul performance in ERI.

High angular momentum CGFs typically exhibit low contraction degrees K. For example, in the def2-QZVP basis set, the g-orbital CGFs corresponding to most chemical elements have $K = 1$. This physical property enables new optimization opportunities for high angular momentum ERI computations. In particular, for (44|44) integrals with $K_{AB} = K_{CD} = 1$, the basis transformation reduces to two back-to-back MatMul operations:

$$(ab|cd) = E_{AB} \otimes [p|q] \otimes E_{CD}. \tag{11}$$

The sequential structure creates an ideal opportunity for fusion. Executing them within a single fused kernel allows intermediate

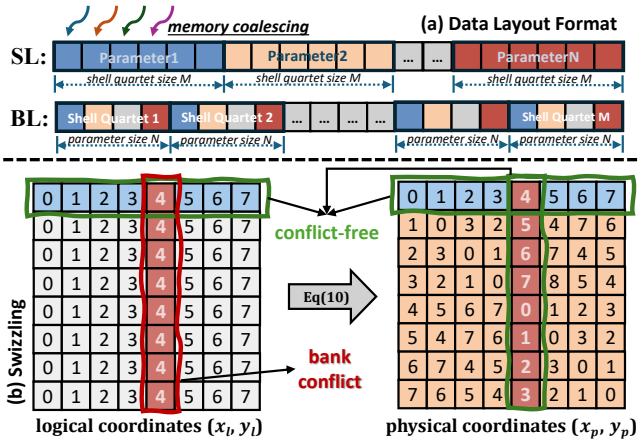


Figure 3: Data Layout and Lightweight Layout Swizzle

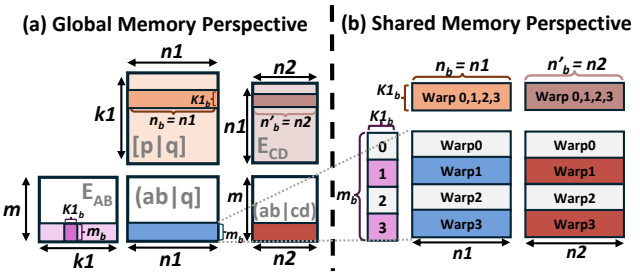


Figure 4: GEMM Coalescing

results to reside entirely in on-chip memory and enables overlapping of matrix I/O with computation, thereby improving overall efficiency.

However, such fusion is nontrivial due to the divergence in parallelization strategies between the two MatMul operations. As depicted in Figure 4, KernelMako overcomes this by adopting a unified N-dimension tiling scheme, wherein each thread block is responsible for a fixed tile in the output matrix along the N axis. On the shared memory level, these tiles are mapped onto warps (`Warp0`–`Warp3`) in a consistent pattern across both MatMul stages. This design ensures that the output of the first MatMul is directly retained in warp-local on-chip memory and immediately consumed by the second MatMul, thereby reducing memory traffic and improving AI accelerators throughput.

KernelMako maximizes AI accelerator throughput by leveraging implicit instruction parallelism, lightweight layout swizzle, and GEMM coalescing to fully exploit operator-level optimizations and translate them into a memory-efficient computational paradigm.

3.2 QuantMako

While KernelMako has enabled a MatMul-centric reformulation of ERI algorithms and achieved up to 2× performance gains even under FP64, this only scratches the surface of AI accelerator potential. As discussed in Section 2.4.2, unlocking the full throughput of low-precision units such as FP16 and TF32 remains a core

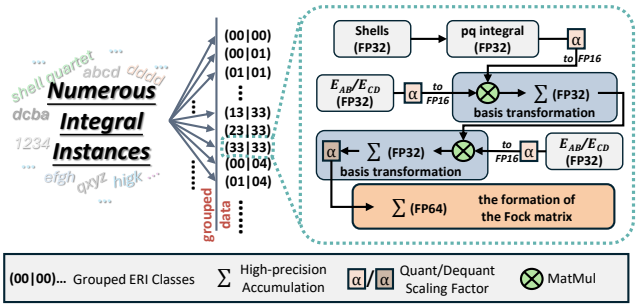


Figure 5: Fine-Grained Quantization and Dual-Stage Accumulation

challenge—one constrained by the precision rigidity of quantum chemistry workloads.

To address this, we introduce QuantMako, a physics-informed quantization framework designed to maximize performance on AI accelerators without compromising scientific fidelity. QuantMako is built on two key insights: (1) *Progressive SCF Tolerance.* During early iterations, the SCF process can tolerate moderate precision loss without affecting the final convergence [90, 136]. (2) *Integral Sparsity.* Within ERI-heavy workloads, a large portion of integrals contribute marginally to the final result—an opportunity revealed by Schwarz screening and magnitude-based filtering [71, 136, 156].

Building on these insights, QuantMako employs a three-pronged strategy to exploit low-precision acceleration while preserving scientific accuracy.

3.2.1 Fine-Grained Quantization. Directly replacing all ERI computations with low-precision arithmetic introduces significant numerical errors and risks overflow or underflow due to the limited dynamic range of FP16. QuantMako mitigates this by applying fine-grained quantization at both the computational workflow and data representation levels, as illustrated in Figure 5.

At the computational level, QuantMako differentiates between precision-sensitive and compute-intensive stages of ERI evaluation. It preserves high-precision arithmetic for the recursive integral generation step—known for its numerical fragility—while introducing FP16 arithmetic into the basis transformation step for the first time. This selective substitution unlocks the full throughput of low-precision AI accelerators, offering up to a theoretical 16× speedup compared to single-precision implementations.

At the data representation level, QuantMako introduces an angular momentum-aware quantization strategy to mitigate the precision degradation caused by naive global scaling. ERI input tensors often exhibit wide dynamic ranges across different angular momentum levels. Uniformly scaling all inputs—e.g., by aligning the global maximum with the FP16 representable bound—amplifies quantization sensitivity to outlier values, leading to severe accuracy loss. Instead, QuantMako groups integral input data by angular momentum level, applying dedicated scaling factors within each angular momentum-specific ERI kernel. This grouping aligns the numerical range of each block with the representable FP16 range, significantly improving quantization robustness.

3.2.2 Dual-Stage Accumulation. To preserve numerical robustness, QuantMako incorporates a dual-stage high-precision accumulation mechanism. *Firstly*, within the ERI computation, the outputs of the FP16-based basis transformation are accumulated using FP32 precision. These intermediate results are then rescaled by the inverse of their quantization scaling factors to perform dequantization, ensuring that numerical fidelity is preserved during the transition from low-precision arithmetic. *Secondly*, during the construction of the Fock matrix, these FP32 integrals are further accumulated into FP64-precision buffers. Although ERIs are computed using fine-grained quantization, the Fock matrix is consistently maintained at full FP64 precision throughout the entire quantum chemistry pipeline.

This design ensures that while the compute-intensive stages benefit from low-precision performance, the final output maintains full double-precision fidelity, satisfying the accuracy demands of quantum chemistry.

3.2.3 Convergence-Aware Scheduling. Although Fine-Grained Quantization and Dual-Stage Accumulation enable the construction of quantized ERI kernels, determining when and where to apply them remains non-trivial. To address this, QuantMako adopts a convergence-aware scheduling strategy grounded in well-established mixed-precision techniques extensively validated in commercial quantum chemistry software [71, 136]. At the integral level (mixed precision), QuantMako leverages density-weighted Schwarz screening and numerical thresholds to classify integrals by importance: critical ones are evaluated with FP64 kernels, moderately important ones with quantized kernels, and negligible ones are pruned. At the iterative level (dynamic precision), it exploits the SCF procedure's inherent tolerance to numerical error by applying relaxed precision thresholds during early iterations—favoring quantized kernels—and gradually tightening them as convergence nears, ensuring FP64-level accuracy.

By coordinating both scheduling dimensions, QuantMako systematically balances performance and accuracy, fusing domain knowledge with hardware characteristics to unlock selective low-precision execution at scale.

3.3 CompilerMako

Although KernelMako and QuantMako successfully leverage AI accelerators for matrix-aligned ERI, a critical challenge remains: how to achieve scalable performance across the full diversity of ERI variants and heterogeneous GPU architectures. The combinatorial explosion of ERI classes with increasing angular momentum (angular momentum), combined with architectural variation across platforms, renders hand-tuned or statically generated kernels brittle, unscalable, and unportable.

To overcome this, we introduce CompilerMako, a compiler-guided framework that transforms ERI kernel generation into a system-level scheduling and tuning problem. At its core, CompilerMako is grounded in the observation that ERI instances—when grouped by angular momentum and contraction degree—follow a finite set of static execution patterns. Analogous to fixed operator graphs in neural networks, these static patterns enable ahead-of-time analysis, fusion, and tuning.

CompilerMako consists of two key strategies: (1) Reuse-Guided Planning captures the static structure of ERI computations and selects fusion strategies that maximize data reuse and on-chip memory efficiency across diverse kernel variants; (2) Architecture-Tuned Compilation generates MatMul-centric kernels using CUTLASS and automatically tunes them for different GPU architectures, achieving near-peak performance without manual retuning.

3.3.1 Reuse-Guided Planning. Reuse-Guided Planning leverages the static nature of ERI computations to perform architecture-aware fusion that maximizes intermediate reuse within the constraints of on-chip memory. The ERI pipeline consists of sequential stages—recursive integral evaluation (r), parallelizable pairwise contractions (pq), and MatMul-based basis transformation—each producing deterministic intermediate tensors for a given angular momentum (angular momentum) class. These fixed reuse patterns, combined with the limited shared memory per streaming multiprocessor (SM), allow for precise compile-time analysis of fusion feasibility.

Given a candidate fusion strategy $\mathcal{F}$, we estimate the total shared memory usage $S(\mathcal{F})$ by summing the sizes of all live intermediate tensors within a thread block:

$$S(\mathcal{F}) = \sum_{T \in \text{Live}(\mathcal{F})} \text{Size}(T) \tag{12}$$

To maintain high occupancy and enable latency hiding via warp scheduling, we enforce the architectural constraint:

$$S(\mathcal{F}) \leq \frac{1}{2} \cdot \text{SMEM}_{\max} \tag{13}$$

This constraint ensures that at least two thread blocks can be resident on each SM. Additionally, the thread block size is set to a multiple of the warp size to facilitate coalesced memory access and efficient warp-synchronous execution.

We then explore the space of fusion strategies, parameterized by fusion granularity (e.g., fusing r and pq stages or treating them separately), and select the optimal fusion strategy $\mathcal{F}^*$ that maximizes reuse, minimizes global memory traffic, and respects architectural limits. Since reuse patterns and memory footprints are fully analyzable at compile time for each ERI class, the resulting fusion strategy is generated once offline and reused across all integrals within the class—ensuring robust, architecture-aware performance portability.

3.3.2 Architecture-Tuned Compilation. Architecture-Tuned Compilation builds upon CUTLASS's tunable primitives to deliver near-peak MatMul performance across diverse GPU architectures, without requiring manual retuning. For each ERI class, CompilerMako generates two precision-specific kernel variants—quantized and FP64—each requiring distinct performance configurations due to their differing arithmetic intensity and resource usage.

To tune these kernels, CompilerMako adopts a MatMul-centric optimization strategy, systematically exploring a rich configuration space including tiling sizes, MMA shapes, swizzling policies, cluster dimensions, threadblock sizes, and data types. While most parameters directly impact MatMul performance, certain dimensions—especially threadblock shape—also influence shared memory usage and fusion boundaries, creating a coupling between architecture tuning and fusion strategy selection.

Algorithm 2: Architecture-Tuned MatMul-Centric Compilation

Input: ERI class C, precision mode $P \in \{\texttt{FP64}, \texttt{FP32}, \texttt{Quant}\}$
Output: Best-tuned kernel configuration S^*

```
Kernel ← GenerateKernels(C, P) ; // Generate base MatMul kernel for given ERI class and precision
BestTime ← ∞; S* ← NULL;
foreach MatMul parameter configuration M in ExploreMatMulParams(CUTLASS primitives) do
    Apply M to Kernel;
    // Threadblock params may affect fusion strategy
    FusedKernel ← ReuseGuidedPlanning(Kernel, M);
    // Use implicit instruction parallelism
    foreach ILP factor I ∈ {1, 2, ..., 32} do
        TunedKernel ← TuneILPSettings(FusedKernel, I);
        t ← ProfileAndMeasure(TunedKernel);
        if t < BestTime then
            BestTime ← t;
            S* ← (TunedKernel);
        end
    end
end
return S*
```

Once threadblock-level configurations are fixed, CompilerMako invokes Reuse-Guided Planning to derive the optimal fusion strategy compatible with the hardware constraints. This fused execution plan is then compiled into architecture-specialized code. Finally, CompilerMako performs a lightweight tuning pass to select the optimal ILP settings, ensuring sufficient warp occupancy and mitigating any thread-level inefficiency introduced during fusion.

Through this integrated workflow, CompilerMako automatically produces high-performance ERI kernels tailored to specific architectures and workloads. By abstracting both hardware variability and algorithmic complexity, it enables portable, scalable performance across heterogeneous accelerator platforms—eliminating the need for labor-intensive, architecture-specific tuning traditionally required in quantum chemistry systems.

4 Implementation

To efficiently realize operator-level optimizations, harness low-precision computation benefits, and automate the achievement of near-peak performance, the Mako system leverages tunable low-level CUTLASS primitives [115]. These primitives allow fine-grained control over thread-level execution and precision-specific data handling within GPU kernels, effectively overcoming limitations posed by fixed and non-customizable device-side APIs such as cuBLAS [114].

KernelMako. We implement implicit instruction parallelism by introducing an instruction scheduling factor into the *pq_integrals* loop as a template parameter. This factor is jointly optimized with CUTLASS primitives' parameters to achieve optimal ILP efficiency (Section 3.1.1). To enable efficient and conflict-free SMEM transposition, we utilize the Swizzling Functor `cutlass.cute.Swizzle` [117] for layout transformations across

FP64, FP32, and FP16 precisions (Section3.1.2). Additionally, inspired by the CUTLASS example of back-to-back convolution fusion [110], KernelMako adopts GEMM coalescing to overlap MatMul computations with memory access, significantly improving AI accelerator throughput (Section 3.1.3).

QuantMako. We achieve flexible and fine-grained precision management within quantized ERI kernels by utilizing specific CUTLASS primitives, such as `cute::transform` for precision conversion and `cute::MMA_Atom` for precision accumulation [111] (Section 3.2.1 & 3.2.2). This targeted approach effectively exploits the performance advantages of low-precision arithmetic without indiscriminately applying it to all computations, thus ensuring scientific accuracy.

CompilerMako. We employ a Reuse-Guided Planning algorithm to systematically generate and evaluate optimal kernel fusion candidates for ERI computations across various precision levels and angular momentum scenarios (Section 3.3.1). By leveraging tunable CUTLASS template parameters, CompilerMako facilitates adaptive kernel configuration tailored to specific MatMul shapes and diverse hardware architectures. Furthermore, we integrate CUTLASS Profiler [112] to automatically benchmark candidate codes, ensuring selection of the most performant kernel variant fo r the target GPU architecture (Section 3.3.2)

5 Evaluation

We present a comprehensive evaluation of Mako, encompassing both fine-grained microbenchmarks—targeting individual kernel-level optimizations—and holistic end-to-end evaluation that assesses the system's efficiency, effectiveness, and scalability in accelerating quantum chemistry workloads on AI accelerators.

5.1 Microbenchmarks

To ensure a fair and accurate performance analysis, we focus on microbenchmarking ERI computations by isolating and evaluating performance across distinct ERI kernel types. Inspired by recent practices in LibintX [6–8], we adopt a similar methodology that excludes auxiliary factors—such as screening heuristics—which may obscure the core performance characteristics. We further advocate for the broader adoption of microbenchmark-based evaluation, as it provides clearer and more reproducible insights into ERI performance.

Metrics. We use a key performance metric that directly reflects computational throughput: *ShellQuartetsperSecond*. This metric measures the number of shell quartets computed per unit time, offering a precise indicator of raw computational efficiency.

State-of-the-arts. We compare the ERI kernels of Mako against LibintX, the latest ERI computation library developed by the Libint team [160]. Among the existing implementations, LibintX is the only one that provides standalone ERI kernel interfaces, making it suitable for fine-grained microbenchmarking. This choice is also driven by the fact that most quantum chemistry software packages employ tightly coupled computation pipelines, which hinder the isolation of individual kernel components for benchmarking purposes. A broader comparison with additional software packages will be presented in the next section.

Machines. All microbenchmark evaluations are conducted on a single NVIDIA A100 Tensor Core GPU (Ampere architecture, Compute Capability 8.0) [113].

5.1.1 State-of-the-art Comparison. We evaluate the double-precision performance of Mako against LibintX (which also supports FP64 only) using the defined microbenchmark metric. As shown in Figure 6, we report results for three representative contraction degrees—{1,1}, {1,5}, and {5,5}—spanning different levels of angular momentum complexity. Across these configurations, Mako achieves average speedups of 2.67×, 2.34×, and 3.11× under the contraction degrees of {1,1}, {1,5}, and {5,5}, respectively. These results demonstrate consistent performance advantages across a range of ERI kernel types, highlighting Mako's effectiveness in accelerating quantum chemistry primitives.

5.1.2 Ablation Study. To evaluate the contribution of key optimization components in Mako to performance improvement, we conduct an ablation study starting from the baseline implementation and incrementally incorporating Mako's core techniques. As shown in Figure 7(a), introducing KernelMako significantly reduces memory transfer overhead, resulting in a notable improvement in ERI kernel computation throughput, as profiled by NVIDIA Nsight Compute [116]. Building upon this optimization, the application of CompilerMako for automated performance tuning further boosts computational throughput, achieving an average 3.98× overall speedup.

We further perform a comprehensive evaluation of QuantMako, focusing on both performance and numerical accuracy. By exploiting the full potential of AI accelerators, QuantMako delivers an average 4.8× speedup over conventional FP64-based kernels as shown in Figure 7(b). Beyond performance, we examine the numerical stability of QuantMako's (AB|CD) ERI kernels. Treating the FP64 output as the reference, Figure 7(c) reports the Root Mean Squared Error (RMSE) [59] of the (AB|CD) kernels computed by QuantMako. As shown in Table 2, QuantMako reduces RMSE by 4.34× compared to FP16, reaching a precision level that closely approximates FP32. To further ensure scientific reliability, we incorporate the Convergence-Aware Scheduling strategy described in Section 3.2.3. This allows Mako to preserve end-to-end accuracy that robustly satisfies the stringent precision requirements of quantum chemistry workloads, as demonstrated in Section 5.2.1.

5.2 End-to-End Evaluation

In this section, we present an end-to-end evaluation of Mako, focusing on both accuracy and performance across a diverse set of representative cases derived from real-world production scenarios.

Datasets. For accuracy validation, we emphasize structural and chemical diversity by constructing a dataset comprising over 200 molecules with varying sizes and compositions. This dataset includes molecules from the tmQM [11] and PubChem [75] databases.

Table 2: Numerical error comparisons with FP32 and FP16

Kernel Version	Baseline FP32	QuantMako	Baseline FP16
RMSE	2.67e-6	3.36e-5	1.46e-4

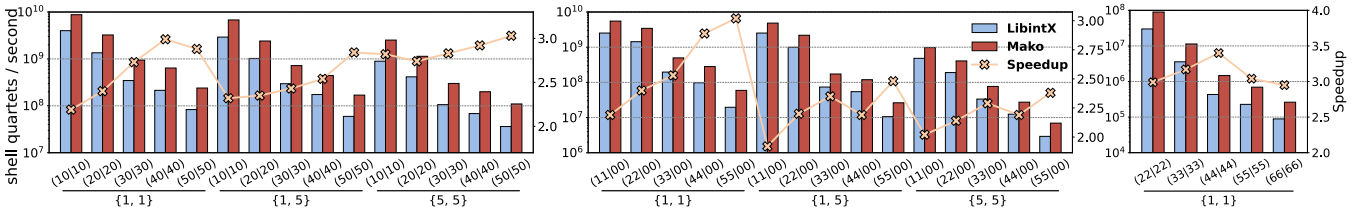


Figure 6: [Microbenchmarks]: FP64 ERI Kernels of Mako Performance Comparison with State-of-the-art on A100

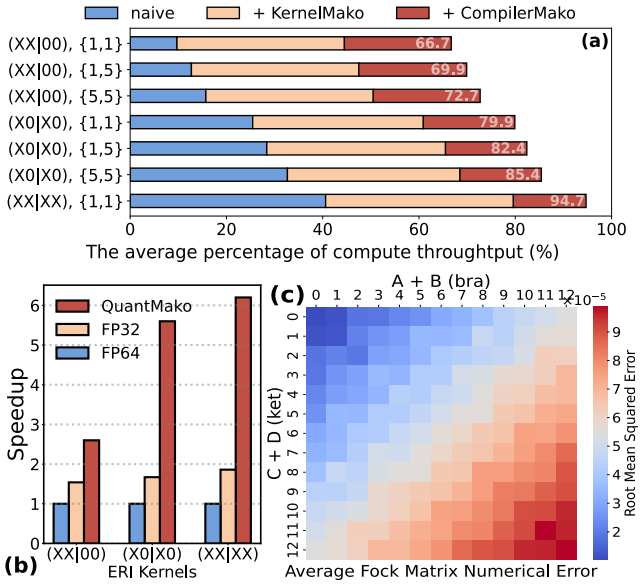


Figure 7: [Microbenchmarks]: Ablation Study

The tmQM dataset provides access to transition-metal-containing molecules with catalytic activity, while the selected subset from PubChem features larger structures with a wide range of chemical compositions. For performance evaluation, we adopt a scalability-oriented approach by testing systems with varying structural characteristics, specifically targeting linear configurations (e.g., polyglycine chains) and compact, globular structures (e.g., water clusters) [86, 120].

Selection of Basis Sets. We selected two widely used families of basis sets: (1) def2-TZVP and def2-QZVP, and (2) cc-pVTZ and cc-pVQZ [42], where T and Q indicate that the respective basis sets include functions up to a maximum angular momentum of 3 (f-type basis function) and 4 (g-type basis function), respectively. This systematic variation in angular momentum allows us to benchmark Mako's capability to handle increasingly complex basis sets and the associated computational demands, thereby enabling more accurate and scalable quantum chemistry simulations to support scientific discovery.

State-of-the-arts. Given that many existing methods are either closed-source or exhibit substantial differences in algorithmic designs, reproducing prior results remains a significant challenge [26, 108, 136]. Nevertheless, we have endeavored to collect and evaluate several widely recognized state-of-the-art quantum chemistry packages, including QUICK [92, 93, 100], and GPU4PySCF [86, 172], to enable a comprehensive end-to-end performance comparison.

Aligned Parameter Settings. To ensure a fair and consistent evaluation of Mako's performance, we adopted a unified set of parameter settings across all end-to-end experiments. Specifically, we applied tight convergence thresholds: the SCF convergence criterion to 10^{-7}. Importantly, all two-electron Coulomb (J) and exchange (K) integrals were computed using analytical expressions, without employing density fitting or other numerical approximations. For the exchange-correlation treatment in DFT calculations, we consistently used the B3LYP functional—a widely used hybrid exchange-correlation functional in DFT [151]. These carefully aligned parameter choices establish a rigorous and controlled computational environment for evaluating Mako's accuracy and efficiency.

Metrics. We use both the average SCF iteration time and the total execution time to evaluate the overall performance of Mako. The average SCF iteration time is calculated over 10 iterations, excluding the first iteration to eliminate initialization overhead [86, 120].

Machines. We conduct single-GPU performance tests on a system featuring an AMD EPYC 7V13 CPU and an NVIDIA A100 Tensor Core GPU (Ampere, Compute Capability 8.0) [113]. Multi-GPU Evaluation. Multi-GPU experiments are run on Azure ND A100 v4 virtual machines. Each node includes 8 NVIDIA A100 Tensor Core GPUs (40 GB each) and 96 AMD EPYC Rome CPU cores. Nodes are interconnected with 200 Gb/s NVIDIA Mellanox HDR InfiniBand. We assign one MPI process per GPU to ensure efficient parallel execution.

5.2.1 Accuracy Validation. We evaluate the numerical accuracy of Mako by computing the Mean Absolute Error (MAE) against several widely adopted quantum chemistry packages. The comparison includes CPU-based software (Psi4 [121] and PySCF [146]) and GPU-accelerated software (QUICK [92, 93, 100] and GPU4PySCF [86, 172]). Detailed results are presented in Table 3. We consider results consistent if the total energy difference between any two implementations is within 1 mHartree [71, 90, 156], a commonly accepted threshold for chemical accuracy. Under this criterion, Mako achieves strong numerical agreement with existing software, demonstrating its correctness and reliability in practical quantum chemistry workloads. Although the quantized version of the ERI kernel introduces larger numerical errors compared to its FP64 counterpart, the adoption of the mature QuantMako technique ensures that the final computational results remain equally accurate.

5.2.2 Single GPU Performance Comparison. Figure 8 presents an end-to-end performance comparison between Mako and

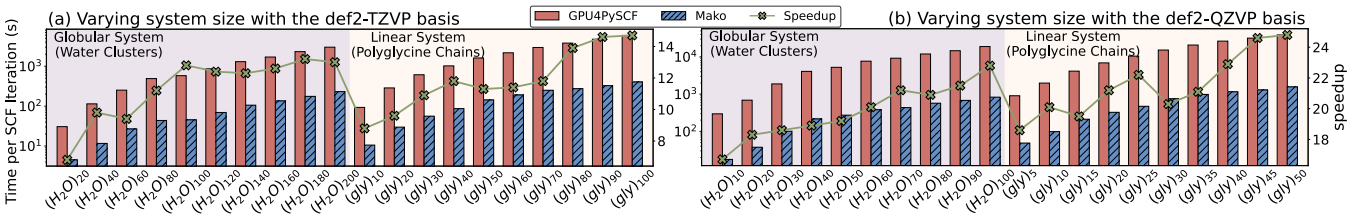


Figure 8: [End-to-End]: Performance Comparison with State-of-the-art on Single A100 GPU

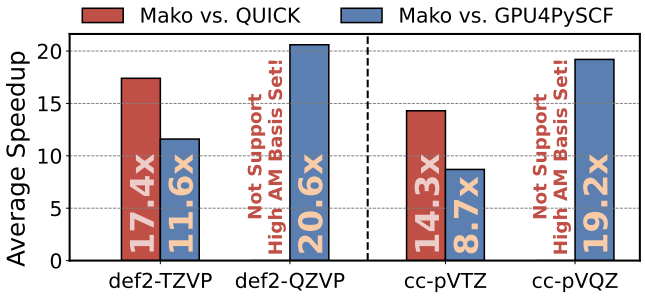


Figure 9: [End-to-End]: Average Speedup across Basis Sets with Progressively Higher Angular Momentum

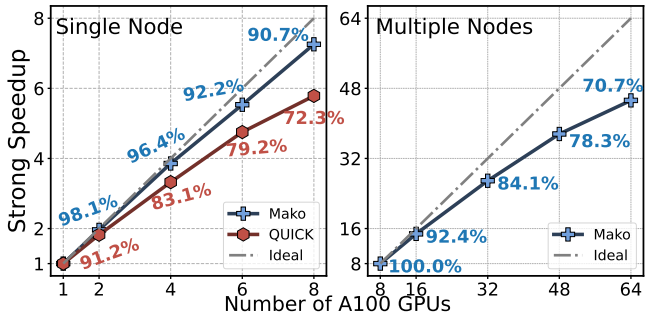


Figure 10: [End-to-End]: Scalability of Mako

Table 3: Mean Absolute Error in the Total Energy of the Converged DFT Calculations Using B3LYP XC Functional

Psi4	PySCF	QUICK	GPU4PySCF
0.023 mHartrees	0.004 mHartrees	0.086 mHartrees	0.004 mHartrees

GPU4PySCF across polyglycine and water cluster systems of increasing size, using def2-TZVP and def2-QZVP that progressively require higher angular momentum components. Figure 9 reports the average speedup of Mako over QUICK and GPU4PySCF across four different basis sets with the datasets. Due to its lack of support for g-type basis functions, QUICK does not support the def2-QZVP and cc-pVQZ basis sets. In contrast, the comparison with GPU4PySCF reveals a clear trend: Mako exhibits increasingly pronounced performance advantages as the angular momentum of the basis sets increases.

The results underscore Mako's robust support for high-angular-momentum basis sets, enabled by its algorithmic co-design with AI accelerators. Since such basis sets are essential for high-accuracy quantum chemical simulations, Mako not only offers superior performance but also demonstrates scalability aligned with the precision frontier of quantum chemistry, thereby realizing the vision of AI accelerators scaling quantum chemistry beyond limits.

5.2.3 Scalability Evaluation. To evaluate the scaling performance of Mako, we studied ubiquitin, a large molecule comprising 1,231 atoms, on 1-8 nodes (1-64 GPUs). As shown in the figure 10, we first compare the parallel efficiency of Mako and QUICK in a single-node environment. Mako demonstrates significantly better parallel efficiency, achieving over 90% efficiency on a single node. Furthermore, we assess Mako's scalability in a multi-node setup. On the platform with 8 nodes and a total of 64 A100 GPUs, Mako maintains over 70% parallel efficiency. It is worth noting that no specific optimization has been applied to the multi-node parallel implementation yet, indicating considerable room for further improvement.

6 Related Work

The rapid evolution of GPU architectures has catalyzed the development of quantum chemistry software [51, 71, 78, 79, 86, 92, 93, 100, 120, 136, 170, 172, 179], which leverage GPU parallelism to accelerate key computational kernels. They have explored a variety of algorithms and implementation strategies for efficient ERI computation [5–8, 13, 99, 152, 158, 175, 176]. In parallel, recent efforts have sought to leverage single arithmetic to improve throughput without sacrificing scientific accuracy. Techniques such as mixed precision [4, 29, 53, 81, 119, 125, 136] have been introduced to accelerate quantum chemistry calculations. Support for high-angular-momentum basis sets [6, 13, 46, 98, 168] has been introduced to improve accuracy in large systems, prompting automated code generation and metaprogramming [6, 8, 46, 68, 152–154] to address the growing complexity of ERI evaluation.

7 Conclusion

We propose Mako, a matrix-aligned quantum chemistry system that restructures DFT computations into matrix-aligned, quantized, and compiler-optimized computations, enabling efficient scaling on modern AI accelerators while delivering significant speedups and maintaining scientific accuracy across diverse hardware and workloads.

References

[1] G Andersson, SE Larsson, G Leander, P Möller, Sven Gösta Nilsson, Ingemar Ragnarsson, Sven Åberg, R Bengtsson, J Dudek, B Nerlo-Pomorska, et al. 1976. Nuclear shell structure at very high angular momentum. *Nuclear Physics A* 268, 2 (1976), 205–256.

[2] Hartwig Anzt, Stanimire Tomov, and Jack J Dongarra. 2015. Accelerating the LOBPCG method on GPUs using a blocked sparse matrix vector product.. In *SpringSim (HPS)*. 75–82.

[3] Gustavo JR Aroeira, Matthew M Davis, Justin M Turney, and Henry F Schaefer III. 2022. Fermi. jl: a modern design for quantum chemistry. *Journal of chemical theory and computation* 18, 2 (2022), 677–686.

[4] Andrey Asadchev and Mark S. Gordon. 2012. Mixed-precision evaluation of two-electron integrals by Rys quadrature. *Computer Physics Communications* 183, 8 (2012), 1563–1567. doi:10.1016/j.cpc.2012.02.020

[5] Andrey Asadchev and Mark S. Gordon. 2012. New Multithreaded Hybrid CPU/GPU Approach to Hartree–Fock. *Journal of Chemical Theory and Computation* 8, 11 (2012), 4166–4176. doi:10.1021/ct300526w arXiv:https://doi.org/10.1021/ct300526w PMID: 26605582.

[6] Andrey Asadchev and Edward F Valeev. 2023. High-performance evaluation of high angular momentum 4-center Gaussian integrals on modern accelerated processors. *The Journal of Physical Chemistry A* 127, 51 (2023), 10889–10895.

[7] Andrey Asadchev and Edward F Valeev. 2023. Memory-efficient recursive evaluation of 3-center Gaussian integrals. *Journal of Chemical Theory and Computation* 19, 6 (2023), 1698–1710.

[8] Andrey Asadchev and Edward F Valeev. 2024. 3-center and 4-center 2-particle Gaussian AO integrals on modern accelerated processors. *The Journal of Chemical Physics* 160, 24 (2024).

[9] Alán Aspuru-Guzik, Roland Lindh, and Markus Reiher. 2018. The matter simulation (r) evolution. *ACS central science* 4, 2 (2018), 144–152.

[10] Evert Jan Baerends and Oleg V Gritsenko. 1997. A quantum chemical view of density functional theory. *The Journal of Physical Chemistry A* 101, 30 (1997), 5383–5403.

[11] David Balcells and Bastian Bjerkem Skjelstad. 2020. tmQM dataset—quantum geometries and properties of 86k transition metal complexes. *Journal of chemical information and modeling* 60, 12 (2020), 6135–6146.

[12] Giuseppe MJ Barca, Colleen Bertoni, Laura Carrington, Dipayan Datta, Nuwan De Silva, J Emiliano Deustua, Dmitri G Fedorov, Jeffrey R Gour, Anastasia O Gunina, Emilie Guidez, et al. 2020. Recent developments in the general atomic and molecular electronic structure system. *The Journal of chemical physics* 152, 15 (2020).

[13] Giuseppe MJ Barca, Jorge L Galvez-Vallejo, David L Poole, Alistair P Rendell, and Mark S Gordon. 2020. High-performance, graphics processing unit-accelerated fock build algorithm. *Journal of Chemical Theory and Computation* 16, 12 (2020), 7232–7238.

[14] Giuseppe MJ Barca, David L Poole, Jorge L Galvez Vallejo, Melisa Alkan, Colleen Bertoni, Alistair P Rendell, and Mark S Gordon. 2020. Scaling the hartree-fock matrix build on summit. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14.

[15] Michael V Berry. 1977. Regular and irregular semiclassical wavefunctions. *Journal of Physics A: Mathematical and General* 10, 12 (1977), 2083.

[16] Muhammad Mubashir Bhatti, M Marin, Ahmed Zeeshan, and Sara I Abdelsalam. 2020. Recent trends in computational fluid dynamics. *Frontiers in Physics* 8 (2020), 593111.

[17] Nick S Blunt, Joan Camps, Ophelia Crawford, Róbert Izsák, Sebastian Leontica, Arjun Mirani, Alexandra E Moylett, Sam A Scivier, Christoph Sunderhauf, Patrick Schopf, et al. 2022. Perspective on the current state-of-the-art of quantum computing for drug discovery applications. *Journal of Chemical Theory and Computation* 18, 12 (2022), 7001–7023.

[18] Stefano Borgani and Andrey Kravtsov. 2011. Cosmological simulations of galaxy clusters. *Advanced Science Letters* 4, 2 (2011), 204–227.

[19] Peter Brezany, Michael Gerndt, Viera Sipkova, and Hans P Zima. 1992. SUPERB support for irregular scientific computations. In *Proceedings of the Scalable High Performance Computing Conference*. 314–321.

[20] Kieron Burke. 2012. Perspective on density functional theory. *The Journal of chemical physics* 136, 15 (2012).

[21] Asbjorn M Burow and Marek Sierka. 2011. Linear scaling hierarchical integration scheme for the exchange-correlation term in molecular and periodic systems. *Journal of Chemical Theory and Computation* 7, 10 (2011), 3097–3104.

[22] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. 2018. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 578–594.

[23] Yuetao Chen, Kun Li, Yuhao Wang, Donglin Bai, Lei Wang, Lingxiao Ma, Liang Yuan, Yunquan Zhang, Ting Cao, and Mao Yang. 2024. Convstencil: Transform stencil computation to matrix multiplication on tensor cores. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. 333–347.

[24] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. 2021. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro* 41, 2 (2021), 29–35.

[25] Aron J Cohen, Paula Mori-Sánchez, and Weitao Yang. 2012. Challenges for density functional theory. *Chemical reviews* 112, 1 (2012), 289–320.

[26] Larry A Curtiss, Paul C Redfern, and Krishnan Raghavachari. 2007. Gaussian-4 theory. *The Journal of chemical physics* 126, 8 (2007).

[27] Abdul Dakkak, Cheng Li, Jinjun Xiong, Isaac Gelado, and Wen-mei Hwu. 2019. Accelerating reduction and scan using tensor core units. In *Proceedings of the ACM International Conference on Supercomputing*. 46–57.

[28] Emanuele Danovaro, Andrea Clematis, Antonella Galizia, Giuseppe Ripepi, Alfonso Quarati, and Daniele D'Agostino. 2014. Heterogeneous architectures for computational intensive applications: a cost-effectiveness analysis. *J. Comput. Appl. Math.* 270 (2014), 63–77.

[29] Sambit Das, Phani Motamarri, Vikram Gavini, Bruno Turcksin, Ying Wai Li, and Brent Leback. 2019. Fast, scalable and accurate finite-element based ab initio calculations using mixed precision computing: 46 PFLOPS simulation of a metallic dislocation system. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 1–11.

[30] Shail Dave, Riyadh Baghdadi, Tony Nowatzki, Sasikanth Avancha, Aviral Shrivastava, and Baoxin Li. 2021. Hardware acceleration of sparse and irregular tensor computations of ml models: A survey and insights. *Proc. IEEE* 109, 10 (2021), 1706–1752.

[31] Jeff Dean. 2021. Introducing pathways: A next-generation ai architecture. *Google Blog* 366 (2021).

[32] Rosa Di Felice, Maricris L Mayes, Ryan M Richard, David B Williams-Young, Garnet Kin-Lic Chan, Wibe A de Jong, Niranjan Govind, Martin Head-Gordon, Matthew R Hermes, Karol Kowalski, et al. 2023. A perspective on sustainable computational chemistry software development and integration. *Journal of chemical theory and computation* 19, 20 (2023), 7056–7076.

[33] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, et al. 2011. The international exascale software project roadmap. *The international journal of high performance computing applications* 25, 1 (2011), 3–60.

[34] Jack Dongarra, John Gunnels, Harun Bayraktar, Azzam Haidar, and Dan Ernst. 2024. Hardware Trends Impacting Floating-Point Computations In Scientific Applications. *arXiv preprint arXiv:2411.12090* (2024).

[35] Jack Dongarra, John Gunnels, Harun Bayraktar, Azzam Haidar, and Dan Ernst. 2024. Hardware Trends Impacting Floating-Point Computations In Scientific Applications. arXiv:2411.12090 [math.NA] https://arxiv.org/abs/2411.12090

[36] Jack Dongarra and David Keyes. 2024. The co-evolution of computational physics and high-performance computing. *Nature Reviews Physics* 6, 10 (2024), 621–627.

[37] Brett I Dunlap. 2005. Angular momentum in molecular quantum mechanical integral evaluation. *Computer physics communications* 165, 1 (2005), 18–36.

[38] Thom H Dunning Jr, Robert J Harrison, David Feller, and Sotiris S Xantheas. 2002. Promise and challenge of high-performance computing, with examples from molecular modelling. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360, 1795 (2002), 1079–1105.

[39] M Eminyan, KB MacAdam, J Slevin, and H Kleinpoppen. 1974. Electron-photon angular correlations in electron-helium collisions: measurements of complex excitation amplitudes, atomic orientation and alignment. *Journal of Physics B: Atomic and Molecular Physics* 7, 12 (1974), 1519.

[40] Eberhard Engel. 2011. *Density functional theory*. Springer.

[41] Mattan Erez, Jung Ho Ahn, Jayanth Gummaraju, Mendel Rosenblum, and William J Dally. 2007. Executing irregular scientific applications on stream architectures. In *Proceedings of the 21st annual international conference on Supercomputing*. 93–104.

[42] Basis Set Exchange. 2025. Basis Set. https://www.basissetexchange.org/.

[43] Massimiliano Fasi, Nicholas J Higham, Mantas Mikaitis, and Srikara Pranesh. 2021. Numerical behavior of NVIDIA tensor cores. *PeerJ Computer Science* 7 (2021), e330.

[44] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[45] Richard A Friesner and Victor Guallar. 2005. Ab initio quantum chemical and mixed quantum mechanics/molecular mechanics (QM/MM) methods for studying enzymatic catalysis. *Annu. Rev. Phys. Chem.* 56, 1 (2005), 389–427.

[46] Jorge Luis Galvez Vallejo, Giuseppe MJ Barca, and Mark S Gordon. 2023. High-performance GPU-accelerated evaluation of electron repulsion integrals. *Molecular Physics* 121, 9-10 (2023), e2112987.

[47] Jorge L Galvez Vallejo, Calum Snowdon, Ryan Stocks, Fazeleh Kazemian, Fiona Chuo Yan Yu, Christopher Seidl, Zoe Seeger, Melisa Alkan, David Poole, Bryce M Westheimer, et al. 2023. Toward an extreme-scale electronic structure system. *The Journal of Chemical Physics* 159, 4 (2023).

[48] Al Geist and Daniel A Reed. 2017. A survey of high-performance computing scaling challenges. *The International Journal of High Performance Computing Applications* 31, 1 (2017), 104–113.

[49] Peter MW Gill, Benny G Johnson, and John A Pople. 1991. Two-electron repulsion integrals over Gaussian s functions. *International journal of quantum chemistry* 40, 6 (1991), 745–752.

[50] Stefan Grimme. 2003. Improved second-order Møller–Plesset perturbation theory by separate scaling of parallel-and antiparallel-spin pair correlation energies. *The Journal of chemical physics* 118, 20 (2003), 9095–9102.

[51] Zhen Guo, Zigeng Huang, Qiaorui Chen, Jiang Shao, Guangcheng Liu, Hung Pham, Changsu Cao, Ji Chen, and Dingshun Lv. 2025. ByteQC: GPU-Accelerated Quantum Chemistry Package for Large-Scale Systems. *arXiv preprint arXiv:2502.17963* (2025).

[52] Tobias Gysi, Carlos Osuna, Oliver Fuhrer, Mauro Bianco, and Thomas C Schulthess. 2015. STELLA: A domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 1–12.

[53] Adela Habib, Joshua Finkelstein, and Anders MN Niklasson. 2024. Efficient mixed-precision matrix factorization of the inverse overlap matrix in electronic structure calculations with AI-hardware and GPUs. *Journal of Chemical Theory and Computation* 20, 16 (2024), 7102–7112.

[54] Jürgen Hafner, Christopher Wolverton, and Gerbrand Ceder. 2006. Toward computational materials design: the impact of density functional theory on materials research. *MRS bulletin* 31, 9 (2006), 659–668.

[55] Azzam Haidar, Stanimire Tomov, Jack Dongarra, and Nicholas J Higham. 2018. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 603–613.

[56] Haozhi Han, Kun Li, Wei Cui, Donglin Bai, Yiwei Zhang, Liang Yuan, Yifeng Chen, Yunquan Zhang, Ting Cao, and Mao Yang. 2025. FlashFFTStencil: Bridging Fast Fourier Transforms to Memory-Efficient Stencil Computations on Tensor Core Units. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. 355–368.

[57] Hwansoo Han and Chau-Wen Tseng. 2006. Exploiting locality for irregular scientific codes. *IEEE Transactions on Parallel and Distributed Systems* 17, 7 (2006), 606–618.

[58] Martin Head-Gordon and John A Pople. 1988. A method for two-electron Gaussian integral and integral derivative evaluation using recurrence relations. *The Journal of chemical physics* 89, 9 (1988), 5777–5786.

[59] Timothy O Hodson. 2022. Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development Discussions* 2022 (2022), 1–10.

[60] Torsten Hoefler, Marcin Copik, Pete Beckman, Andrew Jones, Ian Foster, Manish Parashar, Daniel Reed, Matthias Troyer, Thomas Schulthess, Daniel Ernst, et al. 2024. Xaas: Acceleration as a service to enable productive high-performance cloud computing. *Computing in Science & Engineering* 26, 3 (2024), 40–51.

[61] JJ Hopfield. 1969. Angular momentum and transition-metal superconductivity. *Physical Review* 186, 2 (1969), 443.

[62] Ben Hourahine, Bálint Aradi, Volker Blum, Frank Bonafe, Alex Buccheri, Cristopher Camacho, Caterina Cevallos, MY Deshaye, T Dumitrică, A Dominguez, et al. 2020. DFTB+, a software package for efficient approximate density functional theory based atomistic simulations. *The Journal of chemical physics* 152, 12 (2020).

[63] Hua Huang and Edmond Chow. 2018. Accelerating quantum chemistry with vectorized and batched integrals. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 529–542.

[64] Weile Jia, Han Wang, Mohan Chen, Denghui Lu, Lin Lin, Roberto Car, E Weinan, and Linfeng Zhang. 2020. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In *SC20: International conference for high performance computing, networking, storage and analysis*. IEEE, 1–14.

[65] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. 2018. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205* (2018).

[66] Garima Jindal, Hemanta K Kisan, and Raghavan B Sunoj. 2015. Mechanistic insights on cooperative catalysis through computational quantum chemical methods. *ACS Catalysis* 5, 2 (2015), 480–503.

[67] Erin R Johnson and Axel D Becke. 2005. A post-Hartree–Fock model of intermolecular interactions. *The Journal of chemical physics* 123, 2 (2005).

[68] K. Grace Johnson, Seema Mirchandaney, Ellis Hoag, Alan Heirich, Alex Aiken, and Todd J. Martínez. 2022. Multinode Multi-GPU Two-Electron Integrals: Code Generation Using the Regent Language. *Journal of Chemical Theory and Computation* 18, 11 (2022), 6522–6536. doi:10.1021/acs.jctc.2c00414 arXiv:https://doi.org/10.1021/acs.jctc.2c00414 PMID: 36200649.

[69] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th annual international symposium on computer architecture*. 1–14.

[70] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*. 1–12.

[71] Fusong Ju, Xinran Wei, Lin Huang, Andrew J Jenkins, Leo Xia, Jia Zhang, Jianwei Zhu, Han Yang, Bin Shao, Peggy Dai, et al. 2024. Acceleration without disruption: DFT software as a service. *Journal of Chemical Theory and Computation* 20, 24 (2024), 10838–10851.

[72] LB Ju, CT Zhou, TW Huang, K Jiang, H Zhang, SZ Wu, B Qiao, and SC Ruan. 2017. Production of high-angular-momentum electron beams in laser-plasma interactions. *Physical Review E* 95, 5 (2017), 053205.

[73] Aditya Kashi, Hao Lu, Wesley Brewer, David Rogers, Michael Matheson, Mallikarjun Shankar, and Feiyi Wang. 2024. Mixed-precision numerics in scientific applications: survey and perspectives. *arXiv preprint arXiv:2412.19322* (2024).

[74] Ricky A Kendall, Edoardo Aprà, David E Bernholdt, Eric J Bylaska, Michel Dupuis, George I Fann, Robert J Harrison, Jialin Ju, Jeffrey A Nichols, Jarek Nieplocha, et al. 2000. High performance computational chemistry: An overview of NWChem a distributed parallel application. *Computer Physics Communications* 128, 1-2 (2000), 260–283.

[75] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. 2023. PubChem 2023 update. *Nucleic acids research* 51, D1 (2023), D1373–D1380.

[76] Wolfram Koch and Max C Holthausen. 2015. *A chemist's guide to density functional theory*. John Wiley & Sons.

[77] Walter Kohn, Axel D Becke, and Robert G Parr. 1996. Density functional theory of electronic structure. *The journal of physical chemistry* 100, 31 (1996), 12974–12980.

[78] Karol Kowalski, Raymond Bair, Nicholas P Bauman, Jeffery S Boschen, Eric J Bylaska, Jeff Daily, Wibe A De Jong, Thom Dunning Jr, Niranjan Govind, Robert J Harrison, et al. 2021. From NWChem to NWChemEx: Evolving with the computational chemistry landscape. *Chemical reviews* 121, 8 (2021), 4962–4998.

[79] Thomas D Kühne, Marcella Iannuzzi, Mauro Del Ben, Vladimir V Rybkin, Patrick Seewald, Frederick Stein, Teodoro Laino, Rustam Z Khaliullin, Ole Schütt, Florian Schiffmann, et al. 2020. CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics* 152, 19 (2020).

[80] Gautam Kumar, Sahil Yadav, Aniruddha Mukherjee, Vikas Hassija, and Mohsen Guizani. 2024. Recent advances in quantum computing for drug discovery and development. *IEEE Access* (2024).

[81] Henryk Laqua, Jörg Kussmann, and Christian Ochsenfeld. 2021. Accelerating seminumerical Fock-exchange calculations using mixed single-and double-precision arithmethic. *The Journal of Chemical Physics* 154, 21 (2021).

[82] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. 2021. MLIR: Scaling compiler infrastructure for domain specific computation. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2–14.

[83] Kurt Lejaeghere, Gustav Bihlmayer, Torbjörn Björkman, Peter Blaha, Stefan Blügel, Volker Blum, Damien Caliste, Ivano E Castelli, Stewart J Clark, Andrea Dal Corso, et al. 2016. Reproducibility in density functional theory calculations of solids. *Science* 351, 6280 (2016), aad3000.

[84] Ira N Levine, Daryle H Busch, and Harrison Shull. 2009. *Quantum chemistry*. Vol. 6. Pearson Prentice Hall Upper Saddle River, NJ.

[85] Kun Li, Honghui Shang, Yunquan Zhang, Shigang Li, Baodong Wu, Dong Wang, Libo Zhang, Fang Li, Dexun Chen, and Zhiqiang Wei. 2019. OpenKMC: a KMC design for hundred-billion-atom simulation using millions of cores on Sunway Taihulight. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–16.

[86] Rui Li, Qiming Sun, Xing Zhang, and Garnet Kin-Lic Chan. 2025. Introducing GPU Acceleration into the Python-Based Simulations of Chemistry Framework. *The Journal of Physical Chemistry A* (2025).

[87] Zhaoshi Li, Leibo Liu, Yangdong Deng, Shouyi Yin, Yao Wang, and Shaojun Wei. 2017. Aggressive pipelining of irregular applications on reconfigurable hardware. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 575–586.

[88] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[89] Yuechen Lu, Lijie Zeng, Tengcheng Wang, Xu Fu, Wenxuan Li, Helin Cheng, Dechuang Yang, Zhou Jin, Marc Casas, and Weifeng Liu. 2024. Amgt: Algebraic multigrid solver on tensor cores. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–16.

[90] Nathan Luehr, Ivan S. Ufimtsev, and Todd J. Martínez. 2011. Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units (GPUs). *Journal of Chemical Theory and Computation* 7, 4 (2011), 949–954. doi:10.1021/

ct100701w arXiv:https://doi.org/10.1021/ct100701w PMID: 26606344.
[91] Marina V Malyshkina and Alexander S Novikov. 2021. Modern software for computer modeling in quantum chemistry and molecular dynamics. *Compounds* 1, 3 (2021), 134–144.
[92] Madushanka Manathunga, Hasan Metin Aktulga, Andreas W Gotz, and Kenneth M Merz Jr. 2023. Quantum mechanics/molecular mechanics simulations on NVIDIA and AMD graphics processing units. *Journal of Chemical Information and Modeling* 63, 3 (2023), 711–717.
[93] Madushanka Manathunga, Chi Jin, Vinícius Wilian D Cruzeiro, Yipu Miao, Dawei Mu, Kamesh Arumugam, Kristopher Keipert, Hasan Metin Aktulga, Kenneth M Merz Jr, and Andreas W Gotz. 2021. Harnessing the power of multi-GPU acceleration into the quantum interaction computational kernel program. *Journal of Chemical Theory and Computation* 17, 7 (2021), 3955–3966.
[94] Narbe Mardirossian and Martin Head-Gordon. 2017. Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals. *Molecular physics* 115, 19 (2017), 2315–2372.
[95] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. 2018. Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 522–531.
[96] Victor A Mateevitsi, Mathis Bode, Nicola Ferrier, Paul Fischer, Jens Henrik Göbbert, Joseph A Insley, Yu-Hsiang Lan, Misun Min, Michael E Papka, Saumil Patel, et al. 2023. Scaling Computational Fluid Dynamics: In Situ Visualization of NekRS using SENSEI. In *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. 862–867.
[97] Larry E McMurchie and Ernest R Davidson. 1978. One-and two-electron integrals over Cartesian Gaussian functions. *J. Comput. Phys.* 26, 2 (1978), 218–231.
[98] Yipu Miao and Kenneth M. Jr. Merz. 2015. Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units. *Journal of Chemical Theory and Computation* 11, 4 (2015), 1449–1462. doi:10.1021/ct500984t arXiv:https://doi.org/10.1021/ct500984t PMID: 26574356.
[99] Yipu Miao and Kenneth M Merz Jr. 2013. Acceleration of electron repulsion integral evaluation on graphics processing units via use of recurrence relations. *Journal of Chemical Theory and Computation* 9, 2 (2013), 965–976.
[100] Yipu Miao and Kenneth M Merz Jr. 2015. Acceleration of high angular momentum electron repulsion integrals and integral derivatives on graphics processing units. *Journal of chemical theory and computation* 11, 4 (2015), 1449–1462.
[101] John Michalakes. 2020. Hpc for weather forecasting. *Parallel Algorithms in Computational Science and Engineering* (2020), 297–323.
[102] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).
[103] Vladimir Mironov, Yuri Alexeev, Kristopher Keipert, Michael D'mello, Alexander Moskovsky, and Mark S Gordon. 2017. An efficient MPI/OpenMP parallelization of the Hartree-Fock method for the second generation of Intel® Xeon Phi™ processor. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 1–12.
[104] Vladimir S Mironov. 2022. Reaching the Maximal Unquenched Orbital Angular Momentum L= 3 in Mononuclear Transition-Metal Complexes: Where, When and How? *Inorganics* 10, 12 (2022), 227.
[105] Stephan Mohr, Laura E Ratcliff, Luigi Genovese, Damien Caliste, Paul Boulanger, Stefan Goedecker, and Thierry Deutsch. 2015. Accurate and efficient linear scaling DFT calculations with universal applicability. *Physical Chemistry Chemical Physics* 17, 47 (2015), 31360–31370.
[106] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 1–15.
[107] Frank Neese. 2003. An improvement of the resolution of the identity approximation for the formation of the Coulomb matrix. *Journal of computational chemistry* 24, 14 (2003), 1740–1747.
[108] Frank Neese. 2012. The ORCA program system. *Wiley Interdisciplinary Reviews: Computational Molecular Science* 2, 1 (2012), 73–78.
[109] Frank Neese. 2023. The SHARK integral generation and digestion system. *Journal of Computational Chemistry* 44, 3 (2023), 381–396.
[110] Nvidia. 2025. Back-to-back Convolution in CUTLASS. https://github.com/NVIDIA/cutlass/tree/main/examples/13_two_tensor_op_fusion.
[111] Nvidia. 2025. CuTe in CUTLASS. https://docs.nvidia.com/cutlass/media/docs/cpp/cute/index.html.
[112] Nvidia. 2025. CUTLASS Profiler. https://docs.nvidia.com/cutlass/media/docs/cpp/profiler.html.
[113] Nvidia. 2025. NVIDIA A100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/a100/.
[114] Nvidia. 2025. NVIDIA cuBLAS Documentation. https://docs.nvidia.com/cuda/cublas/.
[115] Nvidia. 2025. NVIDIA CUTLASS Documentation. https://docs.nvidia.com/cutlass/index.html.
[116] Nvidia. 2025. NVIDIA Nsight Compute. https://developer.nvidia.com/nsight-compute.
[117] Nvidia. 2025. Swizzling Functor in CUTLASS. https://github.com/NVIDIA/cutlass/blob/main/include/cute/swizzle_layout.hpp.
[118] Shigeru Obara and A Saika. 1986. Efficient recursive computation of molecular integrals over Cartesian Gaussian functions. *The Journal of chemical physics* 84, 7 (1986), 3963–3974.
[119] Roberto Olivares-Amaya, Mark A. Watson, Richard G. Edgar, Leslie Vogt, Yihan Shao, and Alán Aspuru-Guzik. 2010. Accelerating Correlated Quantum Chemistry Calculations Using Graphical Processing Units and a Mixed Precision Matrix Multiplication Library. *Journal of Chemical Theory and Computation* 6, 1 (2010), 135–144. doi:10.1021/ct900543q arXiv:https://doi.org/10.1021/ct900543q PMID: 26614326.
[120] Elise Palethorpe, Ryan Stocks, and Giuseppe MJ Barca. 2024. Advanced techniques for high-performance fock matrix construction on gpu clusters. *Journal of Chemical Theory and Computation* 20, 23 (2024), 10424–10442.
[121] Robert M Parrish, Lori A Burns, Daniel GA Smith, Andrew C Simmonett, A Eugene DePrince III, Edward G Hohenstein, Ugur Bozkaya, Alexander Yu Sokolov, Roberto Di Remigio, Ryan M Richard, et al. 2017. Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. *Journal of chemical theory and computation* 13, 7 (2017), 3185–3197.
[122] Biagio Peccerillo, Mirco Mannino, Andrea Mondelli, and Sandro Bartolini. 2022. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. *Journal of Systems Architecture* 129 (2022), 102561.
[123] Louis Pisha and Łukasz Ligowski. 2021. Accelerating non-power-of-2 size Fourier transforms with GPU tensor cores. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 507–516.
[124] Felix Plasser, Anna I Krylov, and Andreas Dreuw. 2022. libwfa: Wavefunction analysis tools for excited and open-shell electronic states. *Wiley Interdisciplinary Reviews: Computational Molecular Science* 12, 4 (2022), e1595.
[125] Pavel Pokhilko, Evgeny Epifanovsky, and Anna I. Krylov. 2018. Double Precision Is Not Needed for Many-Body Calculations: Emergent Conventional Wisdom. *Journal of Chemical Theory and Computation* 14, 8 (2018), 4088–4096. doi:10.1021/acs.jctc.8b00321 arXiv:https://doi.org/10.1021/acs.jctc.8b00321 PMID: 29969560.
[126] Jonathan Ragan-Kelley. 2025. The Future of Fast Code: Giving Hardware What It Wants. https://pldi24.sigplan.org/details/pldi-2024-papers/98/The-Future-of-Fast-Code-Giving-Hardware-What-It-Wants.
[127] Daniel Reed, Dennis Gannon, and Jack Dongarra. 2022. Reinventing high performance computing: challenges and opportunities. *arXiv preprint arXiv:2203.02544* (2022).
[128] Daniel A Reed and Jack Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.
[129] J Rys, M Dupuis, and HF King. 1983. Computation of electron repulsion integrals using the Rys quadrature method. *Journal of Computational Chemistry* 4, 2 (1983), 154–157.
[130] Ryuichi Sai, John Mellor-Crummey, Jinfan Xu, and Mauricio Araya-Polo. 2024. Automated code generation of high-order stencils for a dataflow architecture. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–13.
[131] Vivek Sarkar, William Harrod, and Allan E Snavely. 2009. Software challenges in extreme scale systems. In *Journal of Physics: Conference Series*, Vol. 180. IOP Publishing, 012045.
[132] Gabin Schieffer, Daniel Araújo De Medeiros, Jennifer Faj, Aniruddha Marathe, and Ivy Peng. 2024. On the rise of amd matrix cores: Performance, power efficiency, and programmability. In *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 132–143.
[133] Gabin Schieffer, Daniel Medeiros, Jennifer Faj, Aniruddha Marathe, and Ivy Peng. 2024. *Characterizing the Performance, Power Efficiency, and Programmability of AMD Matrix Cores*. Technical Report. Lawrence Livermore National Laboratory (LLNL), Livermore, CA (United States).
[134] Gabriel R Schleder, Antonio CM Padilha, Carlos Mera Acosta, Marcio Costa, and Adalberto Fazzio. 2019. From DFT to machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials* 2, 3 (2019), 032001.
[135] Charles Schwartz. 1962. Importance of angular correlations between atomic electrons. *Physical Review* 126, 3 (1962), 1015.
[136] Stefan Seritan, Christoph Bannwarth, Bryan S Fales, Edward G Hohenstein, Christine M Isborn, Sara IL Kokkila-Schumacher, Xin Li, Fang Liu, Nathan Luehr, James W Snyder Jr, et al. 2021. TeraChem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science* 11, 2 (2021), e1494.
[137] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony

and low-precision. *Advances in Neural Information Processing Systems* 37 (2024), 68658–68685.

[138] Gilad Shainer, Tong Liu, John Michalakes, Jacob Liberman, Jeff Layton, Onur Celebioglu, Scot A Schultz, Joshua Mora, and David Cownie. 2009. Weather research and forecast (WRF) model performance and profiling analysis on advanced multi-core HPC clusters. *10th LCI ICHPCC* (2009).

[139] John M Shalf and Robert Leland. 2015. Computing beyond moore's law. *Computer* 48, 12 (2015), 14–23.

[140] Yihan Shao and Martin Head-Gordon. 2000. An improved J matrix engine for density functional theory calculations. *Chemical Physics Letters* 323, 5-6 (2000), 425–433.

[141] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[142] David S Sholl and Janice A Steckel. 2022. *Density functional theory: a practical introduction*. John Wiley & Sons.

[143] John C Slater. 1951. A simplification of the Hartree-Fock method. *Physical review* 81, 3 (1951), 385.

[144] Benjamin F Spector, Simran Arora, Aaryan Singhal, Daniel Y Fu, and Christopher Ré. 2024. ThunderKittens: Simple, Fast, and Adorable AI Kernels. *arXiv preprint arXiv:2410.20399* (2024).

[145] Ryan Stocks, Jorge L Galvez Vallejo, CY Fiona, Calum Snowdon, Elise Palethorpe, Jakub Kurzak, Dmytro Bykov, and Giuseppe MJ Barca. 2024. Breaking the Million-Electron and 1 EFLOP/s Barriers: Biomolecular-Scale Ab Initio Molecular Dynamics Using MP2 Potentials. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[146] Qiming Sun, Xing Zhang, Samragni Banerjee, Peng Bao, Marc Barbry, Nick S Blunt, Nikolay A Bogdanov, George H Booth, Jia Chen, Zhi-Hao Cui, et al. 2020. Recent developments in the PySCF program package. *The Journal of chemical physics* 153, 2 (2020).

[147] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.

[148] tile ai. 2025. tilelang. https://github.com/tile-ai/tilelang.

[149] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. 10–19.

[150] William W Tipton, Neil D Drummond, and Richard G Hennig. 2014. Importance of high-angular-momentum channels in pseudopotentials for quantum Monte Carlo. *Physical Review B* 90, 12 (2014), 125110.

[151] Julian Tirado-Rives and William L Jorgensen. 2008. Performance of B3LYP density functional methods for a large set of organic molecules. *Journal of chemical theory and computation* 4, 2 (2008), 297–306.

[152] Alexey V Titov, Ivan S Ufimtsev, Nathan Luehr, and Todd J Martinez. 2013. Generating efficient quantum chemistry codes for novel architectures. *Journal of chemical theory and computation* 9, 1 (2013), 213–221.

[153] Alexey V. Titov, Ivan S. Ufimtsev, Nathan Luehr, and Todd J. Martinez. 2013. Generating Efficient Quantum Chemistry Codes for Novel Architectures. *Journal of Chemical Theory and Computation* 9, 1 (2013), 213–221. doi:10.1021/ct300321a arXiv:https://doi.org/10.1021/ct300321a PMID: 26589024.

[154] Gábor János Tornai, István Ladjánszki, Ádám Rák, Gergely Kis, and György Cserey. 2019. Calculation of Quantum Chemical Two-Electron Integrals by Applying Compiler Technology on GPU. *Journal of Chemical Theory and Computation* 15, 10 (2019), 5319–5331. doi:10.1021/acs.jctc.9b00560 arXiv:https://doi.org/10.1021/acs.jctc.9b00560 PMID: 31503475.

[155] W Töws and GM Pastor. 2015. Many-body theory of ultrafast demagnetization and angular momentum transfer in ferromagnetic transition metals. *Physical review letters* 115, 21 (2015), 217204.

[156] Satoki Tsuji, Yasuaki Ito, Haruto Fujii, Nobuya Yokogawa, Kanta Suzuki, Koji Nakano, and Akihiko Kasagi. 2024. Dynamic Screening of Two-Electron Repulsion Integrals in GPU Parallelization. In *2024 Twelfth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 211–217.

[157] Jiyuan Tu, Guan Heng Yeoh, Chaoqun Liu, and Yao Tao. 2023. *Computational fluid dynamics: a practical approach*. Elsevier.

[158] Ivan S Ufimtsev and Todd J Martinez. 2008. Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation. *Journal of Chemical Theory and Computation* 4, 2 (2008), 222–231.

[159] Manuel Ujaldon and Joel Saltz. 2005. The GPU on irregular computing: Performance issues and contributions. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*. IEEE, 7–pp.

[160] Edward F Valeev et al. 2020. Libint: A library for the evaluation of molecular integrals of many-body operators over Gaussian functions. *For the current version, see https://github. com/evaleev/libint/tree/v1* (2020).

[161] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[162] János Végh and Alin Tisan. 2019. The need for modern computing paradigm: Science applied to computing. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 1523–1532.

[163] Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. 2020. Cosmological simulations of galaxy formation. *Nature Reviews Physics* 2, 1 (2020), 42–66.

[164] Qing Wang, Matthias Ihme, Yi-Fan Chen, and John Anderson. 2022. A TensorFlow simulation framework for scientific computing of fluid flows on tensor processing units. *Computer Physics Communications* 274 (2022), 108292.

[165] Tong Wang, Xinheng He, Mingyu Li, Yatao Li, Ran Bi, Yusong Wang, Chaoran Cheng, Xiangzhen Shen, Jiawei Meng, He Zhang, et al. 2024. Ab initio characterization of protein molecular dynamics with AI2BMD. *Nature* (2024), 1–9.

[166] Tong Wang, Xinheng He, Mingyu Li, Bin Shao, and Tie-Yan Liu. 2023. AIMD-Chig: Exploring the conformational space of a 166-atom protein Chignolin with ab initio molecular dynamics. *Scientific Data* 10, 1 (2023), 549.

[167] Yuanheng Wang, Diptarka Hait, K Grace Johnson, O Jonathan Fajen, Juncheng Harry Zhang, Rubén D Guerrero, and Todd J Martínez. 2024. Extending GPU-accelerated Gaussian integrals in the TeraChem software package to f type orbitals: Implementation and applications. *The Journal of Chemical Physics* 161, 17 (2024).

[168] Yuanheng Wang, Diptarka Hait, K. Grace Johnson, O. Jonathan Fajen, Juncheng Harry Zhang, Rubén D. Guerrero, and Todd J. Martínez. 2024. Extending GPU-accelerated Gaussian integrals in the TeraChem software package to f type orbitals: Implementation and applications. *The Journal of Chemical Physics* 161, 17 (11 2024), 174118. doi:10.1063/5.0233523 arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0233523/20237020/174118_1_5.0233523.pdf

[169] Wikipedia. 2024. Top500 Supercomputers. https://en.wikipedia.org/wiki/TOP500.

[170] Karl A Wilkinson, Paul Sherwood, Martyn F Guest, and Kevin J Naidoo. 2011. Acceleration of the GAMESS-UK electronic structure package on graphical processing units. *Journal of computational chemistry* 32, 10 (2011), 2313–2318.

[171] David B Williams-Young, Wibe A De Jong, Hubertus JJ Van Dam, and Chao Yang. 2020. On the efficient evaluation of the exchange correlation potential on graphics processing unit clusters. *Frontiers in chemistry* 8 (2020), 581058.

[172] Xiaojie Wu, Qiming Sun, Zhichen Pu, Tianze Zheng, Wenzhi Ma, Wen Yan, Yu Xia, Zhengxiao Wu, Mian Huo, Xiang Li, et al. 2025. Enhancing GPU-Acceleration in the Python-Based Simulations of Chemistry Frameworks. *Wiley Interdisciplinary Reviews: Computational Molecular Science* 15, 2 (2025), e70008.

[173] Feng Yan, Olatunji Ruwase, Yuxiong He, and Trishul Chilimbi. 2015. Performance modeling and scalability optimization of distributed deep learning systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1355–1364.

[174] Dechuang Yang, Yuxuan Zhao, Yiduo Niu, Weile Jia, En Shao, Weifeng Liu, Guangming Tan, and Zhou Jin. 2024. Mille-feuille: A tile-grained mixed precision single-kernel conjugate gradient solver on gpus. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–16.

[175] Koji Yasuda. 2008. Two-electron integral evaluation on the graphics processor unit. *Journal of Computational Chemistry* 29, 3 (2008), 334–342.

[176] Koji Yasuda and Hironori Maruoka. 2014. Efficient calculation of two-electron integrals for high angular basis functions. *International Journal of Quantum Chemistry* 114, 9 (2014), 543–552.

[177] Jun Zhang. 2018. libreta: Computerized optimization and code synthesis for electron repulsion integral evaluation. *Journal of Chemical Theory and Computation* 14, 2 (2018), 572–587.

[178] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. 2023. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423* (2023).

[179] Weiqing Zhou, Daye Zheng, Qianrui Liu, Denghui Lu, Yu Liu, Peize Lin, Yike Huang, Xingliang Peng, Jie J Bao, Chun Cai, et al. 2025. ABACUS: An Electronic Structure Analysis Package for the AI Era. *arXiv preprint arXiv:2501.08697* (2025).

[180] Maohua Zhu, Tao Zhang, Zhenyu Gu, and Yuan Xie. 2019. Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 359–371.

Appendix: Artifact Description

A Overview of Contributions and Artifacts

A.1 Paper's Main Contributions

C_1 We rearchitect Density Functional Theory (DFT) into structured matrix multiplications (MatMul) pipelines, enabling scalable execution for scientific applications long considered incompatible with AI accelerators.

C_2 We deliver AI-inspired quantization into quantum chemistry, unlocking low-precision throughput while preserving scientific fidelity.

C_3 We pioneer a compiler-inspired framework that elevates quantum chemistry from handcrafted kernels to systems-level scheduling, scaling across high-angular-momentum regimes and heterogeneous architectures.

C_4 We present the benchmark and numerical accuracy of Shark, encompassing the three contributions described above.

C_5 We evaluated the scalability of Shark using up to 64 GPUs by simulating ubiquitin (1,231 atoms) with the def2-TZVP basis set.

A.2 Computational Artifacts

A_1 https://doi.org/10.5281/zenodo.15269220

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_4	Figure 8, 9
	C_5	Figure 10

B Artifact Identification

B.1 Computational Artifact A_1

Relation To Contributions

The computational artifact A_1 is a fully developed quantum chemistry software constructed based on the insights presented in our paper, which leverages AI accelerators to scale quantum chemistry computations beyond traditional limits. Similar to other established quantum chemistry software, A_1 is employed to perform quantum chemistry calculations, such as Density Functional Theory (DFT), to determine the ground-state energies and forces of atomic/molecular systems. The computational artifact A_1 is a concrete implementation of contributions C_1, C_2, and C_3. Moreover, both contributions C_4 and C_5 require A_1 for evaluation.

Expected Results

On a single GPU evaluations, the computational artifact A_1 significantly outperforms GPU4PySCF and QUICK, achieving increasingly greater speedups as the angular momentum requirements of the basis sets grow. Specifically, on an A100 Tensor Core GPU, A_1 attains a 20× speedup compared to GPU4PySCF for challenging high-angular-momentum basis sets such as def2-QZVP. On multi-GPU evaluations, the computational artifact A_1 maintains a parallel efficiency exceeding 90% on 8 GPUs and achieves 70% efficiency across 64 GPUs.

Expected Reproduction Time (in Minutes)

For the water60.xyz case with the def2-QZVP basis set, the expected computation time of this artifact on an A100 Tensor Core GPU is approximately 7 minutes. However, fully reproducing Figure 8 would require several days due to the significantly slower performance of GPU4PySCF.

Artifact Setup (incl. Inputs)

Hardware:

- AMD EPYC 7V13 CPU
- NVIDIA A100 Tensor Core GPU 80GB

Software:

- GCC 11.4.0
- CUDA Toolkit 12.4
- OpenMPI 4.1.2
- CMake 3.22.1

Datasets: The datasets are provided within the computational artifact A_1.

Installation and Deployment:

```
# Install other dependencies
bash scripts/install_third_party.sh
# Configure and build
cmake -S. -Bbuild -DCMAKE_BUILD_TYPE=Release
cmake --build build -j $(nproc)
```

Artifact Execution

The following command serves to execute the compiled Shark system.

```
mpirun -n 1 --cpus-per-proc 6 build/bin/shark --
    mol sample/water60.xyz
```

Here, the `-n` flag specifies the number of GPUs to be utilized. Both `-n` and `-cpus-per-proc` are parameters associated with the Message Passing Interface (MPI) and their usage aligns with standard MPI conventions. The executable file for the compiled Shark quantum chemistry software is located at `build/bin/shark`. The input structure for the calculation is provided in the `sample/water60.xyz` file, which can be substituted with other input systems defined in the XYZ format.

Artifact Analysis (incl. Outputs)

Upon execution, the Shark software will report two key metrics for assessing its computational performance: the total wall-clock time of the calculation and the average time taken for each Self-Consistent Field (SCF) iteration, excluding the initial one. The latter metric corresponds to the statistical data presented in Figure 8. These metrics are crucial for evaluating the computational efficiency of the software.

In addition to performance metrics, Shark will also provide a breakdown of the energy contributions from different components of the investigated atomic or molecular system. For the purpose of verifying the accuracy of the results, the primary focus should be on confirming that the calculated Total Energy is consistent with

the Total Energy values obtained from other quantum chemistry software packages when applied to the identical system using the same set of parameters.