HPC in Computational Chemistry: Bridging Quantum Mechanics, Molecular Dynamics, and Coarse-Grained Models

David Ozog
University of Oregon
ozog@cs.uoregon.edu

Abstract—The past several decades have witnessed tremendous strides in the capabilities of computational chemistry simulations, driven in large part by the extensive parallelism offered by powerful computer clusters and scalable parallel programming methods. However, cluster computing has also seen flattening processor clock frequencies, unsustainable increases in power requirements, and more complicated software that accommodates for the vast diversity of modern heterogeneous computer systems. As scientific methods for modeling and simulating atoms and molecules continue to evolve, software developers struggle to keep up. This position paper describes the primary challenges that face the computational chemistry community, along with recent solutions and techniques that circumvent these difficulties. In particular, I describe in detail the 3 primary models used to simulate atoms and molecules: quantum mechanics (QM), molecular mechanics (MM), and coarse-grained (CG) models. The research literature is rife with examples that utilize high performance computing (HPC) to scale these models to large and relevant chemical problems. However, the grand challenge lies in effectively *bridging* these scales, both spatially and temporally, to study richer chemical models that go beyond single-scale physics. This paper describes the state of the art in multiscale computational chemistry, with an eye toward improving developer productivity with upcoming exascale architectures, in which we require productive software environments, enhanced support for coupled scientific workflows, adaptive and introspective runtime systems, resilience to hardware failures, and extreme scalability.

1 Introduction

Computational chemistry codes comprise some of the most computationally expensive scientific applications of all time, which explains why they are often at the forefront of high performance computing capabilities, both in terms of hardware [1, 2] and software [3, 4]. These codes make use of novel and advanced algorithms, programming models, modern computer architectures, and even data mining and machine learning techniques [5–12]. For example, density function theory codes scale to more than 6,000,000 threads [13], large-scale tensor contractions incorporate advanced communication optimal algorithms [14], Hartree-Fock implementations use several novel programming paradigms [15], and molecular dynamics simulations can cross many orders of magnitude in space and time [16]. These simulations are computationally expensive for many reasons, the primary of which is their considerable algorithmic complexity. The most accurate models of atoms and molecules must incorporate *quantum mechanics (QM)*, and such methods can scale up to $\mathcal{O}(n^8)$ in practice, where n corresponds to the number of basis functions used to represent the system. QM algorithms are some of the most computationally expensive of all practical scientific simulations, and they are intractable for all but the smallest of chemical systems. For example, the largest calculations that include coupled cluster theory simulate only a few dozen atoms. On the other hand, various forms of density functional theory are capable of modeling many thousands of atoms, and classical models can track *billions* of atoms [17]. In fact, many chemical models exist, each achieving a different level of accuracy without incurring the extreme cost of full QM treatment. QM may not be necessary to understand dynamic, thermodynamic, and structural properties of a chemical system, so molecular mechanics (MM) can be used instead. Furthermore, if MM requires tracking too many particles and/or too long of time-scales, then coarse graining (CG) molecular components can still capture relevant physics. A primary goal of this position paper is to describe these different models *individually*, highlighting important optimizations made, so that simulations can effectively apply modern high performance computing (HPC) techniques. However, the grand challenge lies in effectively *bridging* these scales, both spatially and temporally, to study richer chemical models that go beyond single-scale physics.

The development of multiscale and multiresolution modeling software is important for many computational physics and chemistry simulations. For example, when modeling a biological system, certain regions must contain relatively high detail (such as the active binding site of an enzyme), while other regions require much less detail (such as the water solvent that surrounds the enzyme). Of particular interest are methods that bridge the electronic structure of relatively small regions, described by QM, with surrounding parts described by MM. In fact, the scientists who invented these methods recently won the Nobel prize in chemistry in 2013 [18]. QM/MM methods have also inspired adding a third level of *coarse-grained* resolution with QM/MM/CG methods in very recent research [19, 20]. Figure 1 shows an example of a QM/MM/CG system in which the enzyme is modeled with QM, several proteins modeled with MM, and the water solvent modeled with CG. The

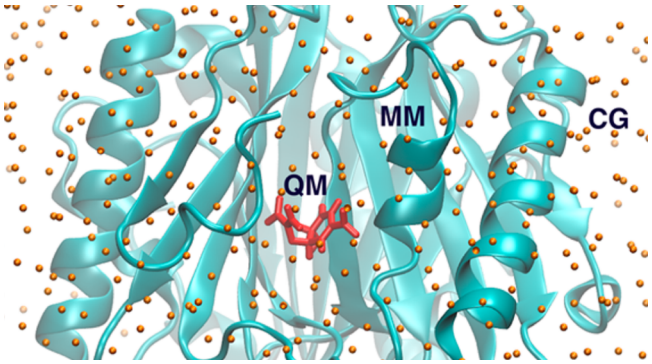


Fig. 1: An example of a QM/MM/CG system in which the enzyme is modeled with QM, several proteins modeled with MM, and the water solvent modeled with CG. This image is from [19].

ultimate challenge here is accomplishing *adaptive resolution*, which allows for different regions of the simulation domain to be modeled with more or less granularity, depending on the desired level of accuracy. Because of its novelty, adaptive resolution computational software generally lacks thorough performance analyses and efficient application of modern optimizations and HPC techniques. This paper describes the state of the art in multiscale computational chemistry, with an eye towards improving developer productivity on upcoming exascale architectures, where we require productive software environments, enhanced support for coupled scientific workflows, adaptive and introspective runtime systems, resilience to hardware failures, and extreme scalability.

The high-level organization of the paper is as follows:

- Section 2: Challenges at extreme scale
- Section 3: Quantum Chemistry models
- Section 4: Molecular Mechanics models
- Section 5: Coarse-Grained models
- Section 6: Multiscale and multiresolution techniques
- Section 7: Relevant applications
- Section 8: Conclusion and future directions.

2 Challenges at Extreme Scale

For several decades, computational scientists have come to rely on the steady increase of CPU clock rates for improving the performance of their application codes. *Moore's Law*, the economic observation that the number of transistors in dense integrated circuit dies roughly doubles every two years, appears to hold into 2015, but there remains another serious problem. *Dennard Scaling*, the observation that power density remains roughly constant as clock rate increases and die size shrinks, has unequivocally ended, which also threatens the end of Moore's Law. Post-petascale machines will struggle to overcome this power wall without drastic adaptations to how computational codes utilize the features of evolving architectures. This will involve efficient load balancing in the face of millions of cores, simultaneous multithreading, vectorizing hardware units, unconventional non-uniform memory hierarchies, and new programming models [21].

Rapidly changing programming environments and the need to scale to millions of cores have lead to a crisis in computational science, wherein codes often scale poorly, software engineering practices lack agility and rigor, and software is difficult to maintain and extend due to increasing complexity [22]. The computational science community and culture requires a radical shift towards improving software productivity to sufficiently adapt to extreme-scale computing platforms [21–25]. Common themes arise in these reports as to what is most important for effectively navigating extreme-scale requirements:

- Component or modular software
- Software productivity metrics
- Software methodology and architecture
- Software development infrastructure
- Scientific workflows (often tightly coupled)
- Verification and validation
- Effective development productivity of teams
- Support for legacy codes
- Multiphysics/multiscale component coupling
- Runtime feedback and control

The sections below consider how these topics are addressed in computational chemistry applications. While these requirements are common to most major computational science software projects, computational chemistry codes are no exception. One goal of this paper is to address how these requirements are satisfied by previous research and recent advancements in chemistry simulation software, along with promising ideas for future improvements. The following sections will cover advances in computational chemistry applications, highlighting when any of the above requirements are well-supported.

3 HPC in Quantum Chemistry

Quantum Chemistry (QC) is the study of atoms and molecules using a quantum mechanical model. In quantum mechanics, energy is quantized, particles obey the Heisenberg uncertainty principle, and wave-particle duality always holds [26]. A thorough introduction to this topic is beyond the scope of this paper, so instead, we concern ourselves with the application of QC to the specific domain described in the next introductory Section 3.1. We then focus on applications that utilize parallel computing to push the limits of QC simulations, in terms of their size and higher-order methods that improve accuracy. Section 3.2 highlights the fact that there are several categories of QC, each with a different fundamental approach, and the subsequent subsections describe these categories in more detail. Section 3.3 describes the dominant parallel programming models used in QC, with a focus on the PGAS model because of its applicability and its wide-spread acceptance in QC codes. Finally, Section 3.4 presents and describes some of the more influential algorithms and methods used in QC codes.

3.1 Introduction and Basics

The goal of QC calculations is to approximately solve the many-body time-independent Schrödinger equation, $\mathbf{H}|\psi\rangle = E|\psi\rangle$, where $\mathbf{H}$ is the Hamiltonian operator, which extracts the sum of all kinetic and potential energies, E,

from the wavefunction, $|\psi\rangle$. Here we make the standard set of assumptions: the Born-Oppenheimer approximation (in which neutrons are fixed but electrons move freely), Slater determinant wavefunctions (that easily satisfy the anti-symmetry principle), and non-relativistic conditions. After these approximations, our focus resides only on the electronic terms of the Hamiltonian and the wavefunction: $\mathbf{H}_{elec}\,|\psi_{elec}\rangle = E\,|\psi_{elec}\rangle$.

The molecular orbitals that express the electronic wavefunction $|\psi_{elec}\rangle$ consist of a sum of *basis functions* from set $\{\phi_j\}$. We desire $\{\phi_j\}$ to be complete, but this is not practical, since it generally requires an infinite number of functions. We therefore truncate to a finite n value large enough to balance the trade-off between accuracy and computational cost:

$$|\psi_i\rangle = \sum_{j=1}^{n} c_{ij}\,|\phi_j\rangle \tag{1}$$

Typically, each basis function is composed of one or more *Gaussian primitive* functions centered at the nucleus location. As we will see in Section 3.2.1, 6-dimensional integrals containing these primitives are the dominant computational component of QC applications. However, because Gaussian integrals have convenient analytical solutions, the complexity of a single integral is in practice $\mathcal{O}(K^4)$, where K is the number of Gaussians used to represent the basis functions [27, 28].

3.2 Overview of the Most Popular QC Methods

Broadly speaking, QC methods fall into two categories: *ab initio*, and everything else. The phrase *ab initio* means "from first principles", which in computational chemistry means that these methods converge to the exact solution of the Schrödinger equation as the collection of basis functions tends towards completeness. The most popular classes of *ab initio* methods are Hartree-Fock, post-Hartree-Fock, and multireference methods. Subsection 3.2.1 covers the Hartree-Fock method, and subsection 3.2.2 covers various post-Hartree-Fock methods; but other sources [29, 30] better describe the multireference methods, which are quite similar to post-Hartree-Fock.

Another QC method that some consider *ab initio* is density functional theory (DFT), discussed in subsection 3.2.3. DFT is the most widely used of the QC methods, and it has many interesting HPC applications, a few of which we consider below. Finally, we briefly consider semi-empirical methods in subsection 3.2.4, and quantum Monte Carlo in subsection 3.2.5.

3.2.1 Hartree-Fock and the Self-Consistent Field Method

The Hartree-Fock (HF) method is the most fundamental of the QC methods, because it is the fundamental starting point for approximately solving the Schrödinger equation. HF attempts to determine the c_{ij} values that best minimize the ground state energy, in accordance with the variational principle. The *ground state* is the configuration of the molecule, along with all its electrons, that exhibits the lowest possible energy. All other states are called excited states. In computational chemistry, the *variational principle* states that the energy of an approximate wave function is always too high; therefore, the best wavefunction is the one that minimizes the ground state energy. This principle is the foundation for all iterative QC methods: The degree of energy minimization determines the relative quality of different QC results.

Algorithm 1 The SCF Procedure

Inputs:
1) A molecule (nuclear coordinates, atomic numbers, and N electrons)
2) A set of basis functions $\{\phi_\mu\}$

Outputs:
Final energy E, Fock matrix **F**, density matrix **D**, coefficient matrix **C**

1: Calculate overlap integrals $S_{\mu\nu}$, core Hamiltonian terms $H_{\mu\nu}^{core}$, and the two-electron integrals $(\mu\nu|\lambda\sigma)$.
2: Diagonalize the overlap matrix $\mathbf{S} = \mathbf{U}\mathbf{s}\mathbf{U}^\dagger$ and obtain $\mathbf{X} = \mathbf{U}\mathbf{s}^{1/2}$.
3: Guess the initial density matrix **D**.
4: **while** E not yet converged **do**
5: Calculate **F** from $H_{\mu\nu}$, **D**, and $(\mu\nu|\lambda\sigma)$.
6: Transform **F** via $\mathbf{F}' = \mathbf{X}^\dagger\mathbf{F}\mathbf{X}$.
7: $E = \sum_{\mu,\nu} D_{\mu\nu}(H_{\mu\nu}^{core} + F_{\mu\nu})$
8: Diagonalize $\mathbf{F}' = \mathbf{C}'\epsilon\mathbf{C}'^\dagger$.
9: $\mathbf{C} = \mathbf{X}\mathbf{C}'$
10: Form **D** from **C** by $D_{\mu\nu} = 2\sum_i^{N/2} C_{\mu i}C_{\nu i}^*$.
11: **end while**

By utilizing a numerical technique for iteratively converging the energy, each subsequent iteration becomes more and more consistent with the field that is imposed by the input molecule and basis set. The method is accordingly called the self-consistent field (SCF) method, and Algorithm 1 shows its *de facto* procedure in pseudocode, with the goal of solving the generalized eigenvalue problem $\mathbf{FC} = \mathbf{SC}\epsilon$, where **F** is the Fock matrix (defined below in Eqn. 2), **C** is the matrix composed of expansion coefficients from Eqn. 1, **S** is the matrix composed of *overlap integrals* taken over all space

$$S_{ij} = \int \phi_i^*(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r} = \langle\phi_i|\phi_j\rangle\,,$$

and ϵ is the energy eigenvalue. Many SCF iterations are required for energy convergence, so steps 1-3 of Algorithm 1 cost much less than steps 5-10. Step 5 normally comprises a majority of the execution time in Hartree-Fock codes, because each element of the Fock matrix requires computing several two-electron integrals:

$$F_{ij} = H_{ij}^{\text{core}} + \sum_{\lambda\sigma}(2(\mu\nu|\lambda\sigma) - (\mu\lambda|\nu\sigma)) \tag{2}$$

The two-electron integrals on the right-hand side are plentiful and expensive to compute [31]. They take this form:

$$(\mu\nu|\lambda\sigma) = \iint \phi_\mu^*(\mathbf{r}_1)\phi_\nu(\mathbf{r}_1)r_{12}^{-1}\phi_\lambda^*(\mathbf{r}_2)\phi_\sigma(\mathbf{r}_2)d\mathbf{r}_1 d\mathbf{r}_2 \tag{3}$$

As mentioned, there are formally $\mathcal{O}(n^4)$ integrals to compute within the basis set. However, from the definition in 3, we see there are only $\sim n^4/8$ unique integrals by permutational symmetry, and the number of non-zero integrals is asymptotically $\mathcal{O}(n^2)$ when Schwartz screening is employed.

While the two-electron integrals comprise the most computation time in SCF, communication and synchronization overheads can very well dominate, particularly at large scale. Inherent communication costs arise from the parallel

decomposition of HF codes, leading to load imbalance and synchronization delays. Parallel HF applications also exhibit highly diverse accesses across distributed process memory spaces. As such, HF is well-suited for a programming model that emphasizes lightweight and one-sided communication within a single global address space. This is the subject of Section 3.3.1.

Nearly all quantum chemistry codes implement HF, and many contain parallel versions. QC codes with parallel versions include NWChem, Q-Chem, GAMMESS, Gaussian, Psi, CFOUR, GPAW, MOLPRO, ACES III, Quantum ESPRESSO, MPQC and many more. Many consider the most scalable code to be NWChem [3, 4], but there remain serious scalability issues due to the inherent load imbalance and the difficulty in exploiting data locality at scale. The load imbalance comes from the fact that we must bundle the 2-electron integrals into what are called *shell quartets* in order to reuse a lot of the intermediate integral values. Formally, a shell quartet is defined as:

$$(MN|PQ) = \{(ij|kl) \text{ s.t. } \begin{array}{l} i \in \text{shell } M, \\ j \in \text{shell } N, \\ k \in \text{shell } P, \\ l \in \text{shell } Q\} \end{array}$$

where "shells" refer to the common notion of electrons that have the same principal quantum number, as one learns in their 1st year of general chemistry. Because this bundling is an *essential* optimization, it introduces more load imbalance to the problem, because some quartets are simply more expensive than others, particularly after screening out quartets that are close to zero. Formally, a two-electron integral is screened if it satisfies this condition:

$$\sqrt{(ij|ij)(kl|kl)} < \tau$$

where τ is the desired screening threshold value.

Optimizing for locality is also very difficult, because tiled accesses across the 4-dimensional space of two-electron integral permutations are wide-spread, the data is sparse, redistribution is expensive, and the locality patterns strongly depend on the input molecular geometry.

3.2.2 Post-Hartree-Fock Methods

Post-Hartree-Fock (PHF) methods improve on the Hartree Fock approximation by including the electron correlation effects that HF mostly ignores (except for parallel-spin electrons [32], the discussion of which is beyond the scope of this paper). In HF, electron correlation is treated in an average way, whereas realistic models must account for the fact that each electron feels instantaneous Coulombic repulsions from each of the other electrons. PFH methods still invoke the Born-Oppenheimer approximation under non-relativistic conditions, but now include more than a single Slater determinant.

The primary PHF methods are configuration interaction (CI), coupled cluster (CC), Møller-Plesset perturbation theory, and multi-reference methods; but there exist many others. Here, we focus on a brief introduction and overview of CI and CC, although it is difficult to appreciate without a full treatment, as in [33].

PHF methods often utilize a notation for describing excited determinants with respect to the reference HF wavefunction, $|\Psi_{\text{HF}}\rangle$. Formally, HF assumes that each electron is described by an independent one-electron Hamiltonian:

$$h_i = -\frac{1}{2}\nabla_i^2 - \sum_{k=1}^{M} \frac{Z_k}{r_{ik}}$$

such that $h_i |\psi_i\rangle = \varepsilon_i |\psi_i\rangle$. Because the total Hamiltonian is assumed separable, the many-electron solution becomes $|\Psi_{\text{HF}}\rangle = |\psi_1\psi_2\cdots\psi_N\rangle$. On the other hand, PHF methods strive to discover the *exact* wavefunction:

$$|\Psi\rangle = c_0 |\Psi_{\text{HF}}\rangle + \sum_{i}^{\text{occ.}} \sum_{r}^{\text{vir.}} c_i^r |\Psi_i^r\rangle + \sum_{i<j}^{\text{occ.}} \sum_{r<s}^{\text{vir.}} c_{ij}^{rs} \left|\Psi_{ij}^{rs}\right\rangle + \cdots \quad (4)$$

where the notation $|\Psi_i^r\rangle$ refers to a determinant in which electron i is in excited (or *virtual*) state r and $\left|\Psi_{jk}^{st}\right\rangle$ refers to a doubly-excited determinant in which electron j is in excited state s, and electron k is in excited state t. Notice that if we include all possible configuration states, then we have exactly solved this form of the electronic Schrödinger equation. This accomplishment is referred to as *full CI*, and it is never accomplished in practice. However, given a truncated form of Eqn. 4 (for example, including the first two terms is referred to as CISD), we apply the linear variational method to form the matrix representation of the Hamiltonian, then find the eigenvalues of the matrix. Specifically, we solve the eigenvalue problem:

$$\mathbf{Hc} = E\mathbf{Sc}$$

where $\mathbf{c}$ is now the column-vector of coefficients for our wavefunction in the desired basis set, $\mathbf{S}$ is the overlap matrix, and $\mathbf{H}$ is the CI Hamiltonian.

CC theory, on the other hand, assumes an exponential ansatz operator applied to the reference wavefunction:

$$|\Psi_{\text{CC}}\rangle = e^{\mathbf{T}} |\Psi_{\text{HF}}\rangle$$

where the $\mathbf{T}$ operator definition is:

$$\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_3 + ...$$

$$\mathbf{T}_n = \frac{1}{(n!)^2} \sum_{\substack{a_1 \ldots a_n \\ i_1 \ldots i_n}} t_{i_1 \ldots i_n}^{a_1 \ldots a_n} a_{a_1}^\dagger \ldots a_{a_n}^\dagger a_{i_n} \ldots a_{i_1}$$

Each $t_{i_1 \ldots i_n}^{a_1 \ldots a_n}$ term is a tensor with rank $2n$ that represents the set of amplitudes for all possible excitations of n electrons to excited states in virtual orbitals. The terms a_i and $a_a^\dagger$ are the creation and annihilation operators, respectively, that act on electron states. The creation operator adds a row (electron) and column (orbital) to the Slater determinant, whereas the annihilation operator removes a row and column. By far, the most computationally expensive component of CC codes is the computation of tensor contractions to determine the $\mathbf{T}_n$ terms. This is the object of much advanced research, and we discuss specific accomplishments in Sections 3.4.3, 3.4.4, and, 3.4.5.

3.2.3 Density Functional Theory

Density Functional Theory (DFT) is the most widely used QC method (based on the number of citations throughout the history of *Physical Reviews* [34]). It also happens to be the most prevalent method used for large scale simulations, parallel performance studies, and applications in the chemical industry [35]. DFT has the distinct advantage of being able to ignore the complicated features of the electronic wavefunction, while still accounting for correlation effects that Hartree-Fock ignores. It is strongly based on the notion of the electron density, which is defined as the integral over the spin (s_i) and spatial ($\mathbf{r}_i$) coordinates:

$$\rho(\mathbf{r}) = N \sum_{s_1} \cdots \sum_{s_N} \int d\mathbf{r}_2 \cdots \int d\mathbf{r}_N |\Psi|^2$$

where Ψ is actually a function of $\Psi(\mathbf{r}, s_1, \mathbf{r}_2, s_2, \cdots, \mathbf{r}_N, s_N)$. Unlike Ψ, the electron density ρ is a physical observable that can be measured experimentally, and its integral over all space is conveniently the total number of electrons:

$$N = \int \rho(\mathbf{r}) d\mathbf{r}$$

The goal of most DFT methods is to solve the so-called Kohn-Sham (KS) equations (in similar form to the Schrödinger eigenvalue equations) [30] by representing the total energy of the system as a functional. For example, Thomas and Fermi produced the first DFT model [36] by writing the kinetic energy as:

$$T_{TF}[\rho(\mathbf{r})] = \frac{3}{10}(3\pi^2)^{2/3} \int \rho^{5/3}(\mathbf{r}) d\mathbf{r} \tag{5}$$

This notation $F[\rho(\mathbf{r})]$ implies that F is a **functional**, since it takes ρ, which itself is a function of $\mathbf{r}$, as its argument. In a pivotal result, Hohenburg and Kohn famously proved [37] that some functional of total energy, $E[\rho(\mathbf{r})]$, must exist which represents the many-body ground state both *exactly* and *uniquely*. However, this proof specifies neither the form of the functional, nor how to obtain it. Accordingly, KS-DFT research often consists of exploring different forms of these functionals, and classifying their applicability to certain chemical systems.

Solving the orbital coefficients using KS-DFT is very similar to HF [38], except instead of the F_{ij} elements composing the Fock matrix as in Eqn. 2, we have

$$F_{ij} = H_{ij}^{\text{core}} + G_{ij}^{J} + \alpha G_{ij}^{K} + \beta G_{ij}^{\text{X-DFT}} + \gamma G_{ij}^{\text{C-DFT}}$$

where the first 3 terms on the right hand side are the same as in HF, but the final two more difficult terms consist of functionals of the energy, such as in Eqn. 5. Here, the constants α, β, and γ can enable spanning the limits of HF and DFT and any hybrid mixture of the two. Because of DFT's computational similarity, it shares the same bottlenecks: calculating all the two-electron integrals, forming the Fock and Density matrices,

3.2.4 Semi-Empirical Methods

Semi-empirical methods differ from ab-initio by incorporating empirical observations to account for errors made by assumptions and approximations inherent to solving the Schrödinger equation in practice. For example, including empirical constants in terms of the secular determinant may better approximate resonance integrals [30]. A popular semi-empirical method is the complete neglect of differential overlap (CNDO), which adopts a series of conventions and parameterizations to drastically simplify the application of HF theory. Specifically, CNDO only considers valence electrons and makes the zero differential overlap approximation, which ignores certain integrals [29], reducing the algorithmic complexity from $\mathcal{O}(n^4/8) \sim \mathcal{O}(n^4)$ to $\mathcal{O}(n(n+1)/2) \sim \mathcal{O}(n^2)$.

At this point it is important to note that methods in machine learning [9, 39] often show less error than certain semi-empirical methods. If a semi-empirical does not offer insight from a parameterization, then it may be a better choice to use a machine learning method to improve accuracy.

3.2.5 Quantum Monte Carlo

Quantum Monte Carlo (QMC) is briefly considered here for the sake of completeness, but also because of its extreme scalability [40] and promising results on GPUs [41]. There are many different QMC schemes, but within the limited scope of this paper it suffices to consider those that fall under the variational Monte Carlo category. In short, this QMC method takes the variational principle (defined in 3.2.1) one step further to evaluate the two-electron integrals numerically. Because the energy depends on a given set of variational parameters, QMC utilizes mathematical optimization to determine the ground state energy by selecting the best set of parameters.

3.3 Parallel Programming Models in QC

QC codes have extraordinary demands in term of memory requirements. For instance, consider the well-known hierarchy of CC methods that provides increasing accuracy at increased computational cost [33]:

$$\cdots < CCSD < CCSD(T) < CCSDT < CCSDT(Q) < CCSDTQ < \cdots .$$

Here, S refers to truncation at the singles term, D to the doubles term, T to triples, and Q to quadruples (terms with parentheses refer to *perturbation* terms). The simplest CC method that is generally useful is CCSD, has a computational cost of $\mathcal{O}(n^6)$ and storage cost of $\mathcal{O}(n^4)$, where n is again the number of basis functions. CCSD(T) is a very good approximation to the full CCSDT method, which requires $\mathcal{O}(n^8)$ computation and $\mathcal{O}(n^6)$ storage. The addition of quadruples provides chemical accuracy, albeit at great computational cost. CCSDTQ requires $\mathcal{O}(n^{10})$ computation and $\mathcal{O}(n^8)$ storage. Needless to say, these memory requirements quickly become prohibitive for molecular systems of moderate size. For even just a few water molecules with a reasonable basis set such as cc-pvdz, the application can easily consume dozens of GBs, necessitating some form of distributed memory management. Typically, distributed codes distribute the Fock matrix in some sort of block cyclic or irregular block cyclic fashion.

Furthermore, due to the nature of the entire collection of two-electron integrals, processes that own a particular block of the Fock matrix require access to blocks owned by *other* processes during distributed computation. Treating

such communication algorithms with point-to-point message passing is intractable and requires high synchronization overhead. QM codes therefore require efficient *one-sided* message passing capabilities in which a process can get or put data into the memory of a remote process *without the explicit receiving communication call, or even participation of the CPU on the remote process*. Nowadays, many modern network interconnects such as InfiniBand, PAMI (on Blue Gene/Q), iWARP, Cray Gemini/Aries, support such remote direct memory access (RDMA) capabilities. These operations are necessary, not only for tiled accesses to compute elements of the Fock and density matrices, but also to support *nxtval* like dynamic load balancing, task assignment, and work stealing [42–44].

The extraordinary demands of QC simulations require the support and development of novel and productive parallel programming models and paradigms. Most QC codes run on a single compute node, but the most popular frameworks, such as GAMESS, ACE III, NWChem, and Gaussian have parallel implementations. Most parallel HF codes listed in section 3.2.1 use MPI for distributed message passing, but GAMESS, NWChem, GTFock do not (necessarily). Interestingly, GAMESS uses the distributed data interface (DDI) [45] for message passing, which originated as an interface to support one-sided messaging. At the time of DDI's inception, the MPI-2 specification included one-sided memory window instantiations (`MPI_WIN_CREATE`) and subsequent put and get operations (`MPI_PUT/MPI_GET`) but these functions were apparently not fully offered by any vendor. The DDI programming model emphasizes three types of memory: replicated data (one copy of small matrices one each CPU), distributed data (for large matrices or tensors spread across the cluster), and node-replicated (for data to be stored once per compute node).

3.3.1 PGAS in Quantum Chemistry

The algorithmic characteristics and resource requirements of HF (and post-HF) methods clearly motivate the use of distributed computation. HF tasks are independent and free to be computed by any available processor. Also, simulating a molecule of moderate size has considerable memory requirements that can easily exceed the memory space of a single compute node. However, at a high level of abstraction, *indexing into distributed HF data structures need not be different than indexing shared memory structures*. This programming abstraction is the basis of the partitioned global address space (PGAS) model for distributing and interacting with data. In computational chemistry, this model is advantageous for interacting with arrays and tensors productively and efficiently.

The PGAS model utilizes one-sided communication semantics, allowing a process to access remote data with a single communication routine. Remote memory access (RMA) is particularly advantageous in HF applications for three primary reasons. First, HF exhibits highly irregular access patterns due to the intrinsic structure of molecules and the necessary removal of integrals from the Fock matrix in a procedure called "screening". Second, there is a need for efficient atomic accumulate operations to update tiles of the global matrices without the explicit participation of the target process. Finally, dynamic load balancing in HF is usually controlled by either a single atomic counter [3], or many counters [43, 46], both of which require a fast one-sided fetch-and-add implementation.

The NWChem software toolkit [47] paved the way towards the ubiquity of PGAS in computational chemistry using the Global Arrays (GA) library and the underlying ARMCI messaging infrastructure [3, 4]. GTFock followed suit, also using GA for tiled accesses across the irregularly distributed Fock and density matrices. The GA model is applicable to many applications, including ghost cells and distributed linear algebra, but GA's primary use today is for quantum chemistry. GA is limited to a C and Fortran API, but does have Python and C++ wrapper interfaces. There exist many other PGAS runtimes, including but not limited to: UPC++, Titanium, X10, Chapel, Co-Array Fortran, and UPC.

3.4 Key Algorithmic Improvements

This section highlights some key algorithmic improvements made to QC codes, particularly in the area of parallel computing and HPC. Parallel strong scaling is crucial for QC codes, but the inherent load imbalance and communication bound nature of the calculations require innovative optimization approaches. The following sections describe several such optimizations.

3.4.1 Load Balancing: Static Partitioning + Work Stealing

Section 3.2.1 describes why the primary hindrance to scalability in QC codes is often due to load imbalance, and much research tackles this problem [43, 48, 49]. Although individual two-electron integrals do not possess drastic differences in execution time, the crucial issue is that *bundles* of shell quartets can vary *greatly* in computational cost. It is necessary to designate shell quartets as task units in HF codes because it enables the reuse of intermediate quantities shared by basis functions within a quartet [46, 50]. The goal is to assign these task units to processors with minimal overhead and a schedule that reduces the makespan.

NWChem balances load with a centralized dynamic scheduler, which is controlled by a single global counter control referred to as `nxtval` (for "next value"). Each task has a unique ID, and a call to `nxtval` *fetches* the current ID of an uncomputed task, then atomically *adds* 1 to the counter so the next task gets executed. Once the counter reaches the total number of tasks, all work is done. For this reason, the performance of RMA fetch-and-add operations is very important for the scalability of computational chemistry codes like NWChem and GTFock. This has motivated the implementation and analysis of hardware-supported fetch-and-ops on modern interconnects, such as Cray Aries using the GNI/DMAPP interfaces [42].

The `nxtval` scheme exhibits measurable performance degradation caused by network contention, but it can be alleviated by an informed static partitioning of tasks and the addition of atomic counters to every process or compute node. This strategy is called Inspector/Executor load balancing, which shows substantial speedup in NWChem [43]. The GTFock project takes this notion one step further by following the static partitioning phase with *work stealing* [44, 46]. During the local phase, each process only accesses the local memory counter; however, during the work

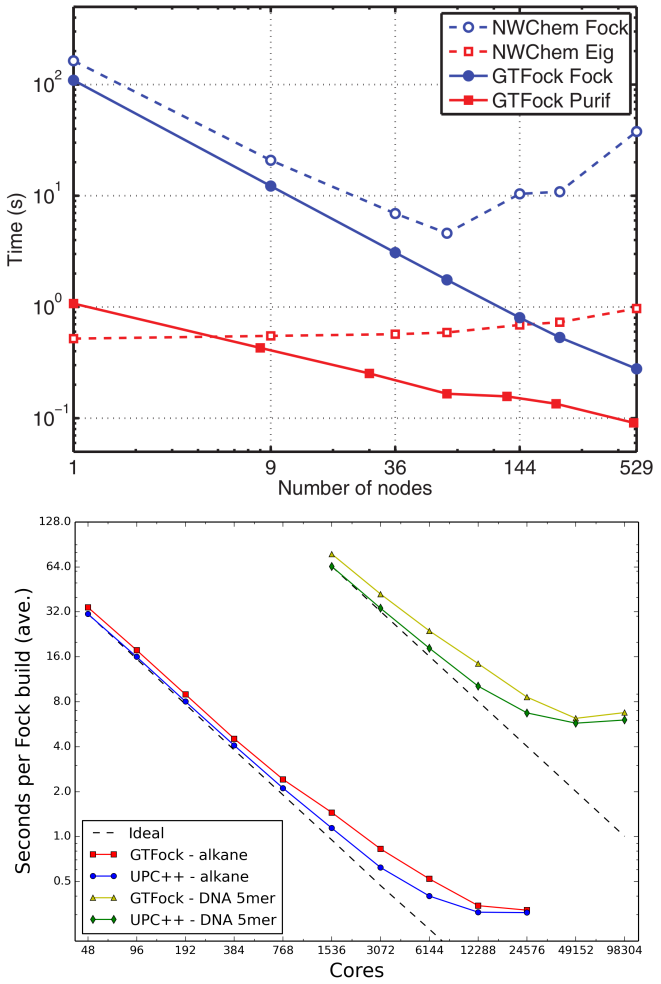


Fig. 2: GTFock improves NWChem's performance of Hartree-Fock calculations. (The top plot is from [44]).

stealing phase, the process accesses other counters remotely. As illustrated in Algorithm 2, each process in GTFock begins an SCF iteration by prefetching the necessary tiles from the global density matrix and storing them into a local buffer. After all local tasks are computed, the global Fock matrix is updated. Then, each process probes the `nxtval` counters of remote processes, looking for work to steal. This algorithm results in many more *local* fetch-and-adds than *remote*, which has important performance implications for how the operations take place. GTFock shows good Xeon Phi performance using OpenMP and vectorized two-electron integral calculation with the OptErd library [44]. Futhermore, the UPC++ PGAS extension for C++ [51] has improved performance by up to 20% using the new DArray library [52], as illustrated in Fig. 3.

3.4.2 Diagonalization vs. Purification

One of the computational bottlenecks of QM codes is the diagonalization of the Fock matrix, which is approximately $\mathcal{O}(n^3)$. Most parallel QM implementations make use of a parallel linear algebra library such as ScaLAPACK, which is a reasonable choice in terms of scalability. However, an alternative "diagonalization-free" technique called *purification* [53] claims to achieve linear scaling by exploiting the fact that elements of the density matrix are short range in coordinate space. This means that matrix elements, $\rho_{ij} \to 0$, as their pairwise distances, $R_{ij} \to \infty$. By truncating elements beyond a certain cutoff distance, this method may achieve linear scaling with system size.

GTFock and the UPC++ HF implementation described in the previous subsection both make use of the purification technique, but more needs to be done to compare its performance relative to standard diagonalization methods. For instance, we know that GTFock SCF iterations take far less time than NWChem iterations, but we do not yet know about convergence properties of the estimated ground state energy between the two approaches. For instance, the performance per iteration of purification may be better than diagonalization, but the convergence may be slow enough to deem these improvements fruitless. Future work should consider such convergence behavior in the context of overall application performance.

3.4.3 Coupled Cluster and Tensor Contractions

For higher level QM methods such as coupled cluster, the tensor contraction is the most important operation related to achieving scalable performance. The tensor contraction itself is essentially a matrix multiplication, except between tensors, which are multidimensional arrays. In fact, tensor contractions can be reduced to matrix multiplication by a series of index reordering operations. Most of the load imbalance in NWChem arises from such a decomposition of multidimensional tensor contractions into a large collection of matrix multiplication operations. Interesting work by Solomonik et al. [54] considers a cyclic distribution of tensor data with the Cyclops Tensor Framework (CTF), which preserves symmetry and drastically reduces inherent load imbalance showing significant speedups over ScaLAPACK and NWChem. Further work proves a communication *op-*

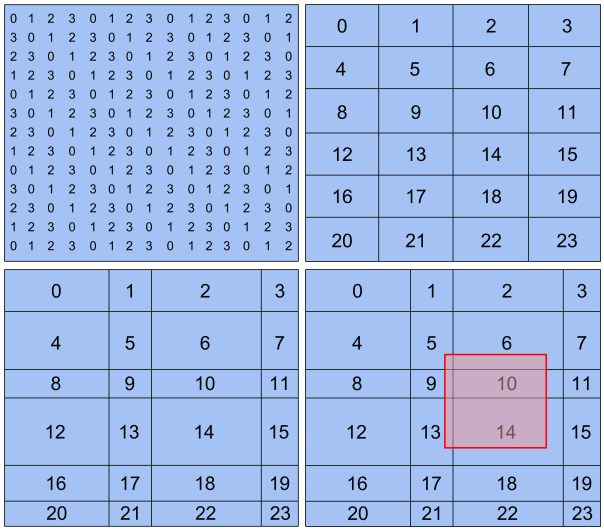


Fig. 3: By default, UPC++ uses a regular round-robin distribution (upper left), while GA uses a regular 2D block-cyclic distribution (upper right). After screening and repartitioning, HF arrays are *irregularly* distributed (bottom left). A tiled access (in red) may span multiple owning processes (bottom right), in this case ranks 5, 6, 9, 10, 13, and 14.

timal algorithm on the 5D torus of BG/Q for dense tensor contractions. The RRR framework (for Reduction, Recursive broadcast, and Rotation) shows significant speedups over CTF for certain tensor contractions, such as:

$$C[i,j,m,n] = A[i,j,m,k] \times B[k,n]$$

but not for others, such as:

$$C[i,j,m,n] = A[i,j,k,l] \times B[k,l,m,n]$$

RRR generates several algorithms based on different iteration space mappings and compatible input distributions [14]. The time to generate these algorithms is not reported in this work, and it would be interesting to see how the chosen algorithm depends on tensor size, network topology, and amount of memory per compute node.

3.4.4 DAG Dependencies/Scheduling

Not only are tasks within a single tensor contraction of coupled cluster calculations independent, but so too are *some* tensor contractions independent of each other. Recent work has developed directed acyclic graphs (DAGs) that represent the dependencies between the tensor contractions in CCSD calculations [14]. Using this DAG as a task graph, the tensor contractions are grouped into several pools (see Fig. 4) and executed with fewer global barrier synchronization costs compared to NWChem and coupled cluster. Subsequent work takes this notion one step further to construct a dataflow-based execution by breaking the CC kernels down into a large collection of fine-grained tasks with explicitly defined dependencies [55]. This work uses the PaRSEC dataflow framework directly on the Tensor Contraction Engine of NWChem.

3.4.5 Accelerator applications

Implementation and performance studies of QM methods on accelerated architecture, such as GPUs and the Intel Xeon Phi Many Integrated Core (MICs) have been quite prevalent the past several years. There are too many to describe in great detail here, so this section mentions only a few with impactful results.

Much work has considered running two-electron integrals on GPUs [57–60] with satisfactory results. Now, many QM frameworks enable two-electrons to be run on GPUs including ABINIT, ACES III, ADF, BigDFT, CP2K, GAMESS, GAMESS-UK, GPAW, LATTE, LSDalton, LSMS, MOLCAS, MOPAC2012, NWChem, OCTOPUS, PEtot, QUICK, Q-Chem, QMCPack, Quantum Espresso/PWscf, QUICK, TeraChem, and VASP [61]. DePrince and Hammond implemented the first coupled clusters code on GPUs [62], showing a 4-5 speedup relative to a multithreaded CPU in 2010. In their implementation, the entire SCF iteration takes place on the GPU, but only CCD was implemented. Now, there is a CCSD(T) GPU implementation in NWChem [63].

On the Xeon Phi, Apra et al. optimize CCSD(T) code for the MIC using offload mode [64]. OptErd (the electron repulsion integral library used by GTFock) also makes use of MICs in offload mode [44]. Shan et al. tune the TEXAS integral module, used by NWChem to run more efficiently on multiple architectures, including the Xeon Phi [31]. They also implement MIC optimizations for CCSD(T) in *native* mode [65], which may be more useful for future Knights Landing and Knights Hill architectures.

Other recent work focuses on enhancing rapid development of QM codes by generating code to run on several novel architectures. For example, Titov et al. use metaprogramming to generate SCF code for the GPU and MIC using either CUDA, OpenCL, or OpenMP based parallelism [66]. Also, the KPPA project by Linford et al. [67] automatically generates C or Fortran90 code to simulate chemical kinetic systems, and the same approach could likely be useful for QM codes as well.

Algorithm 2 Load balance and work stealing

```
1: Determine total number of tasks (after screening).
2: Statically partition tasks across process grid.
3: Prefetch data from DAarrays.
4: while a task remains in local queue    /* fetch_add local integer */
5:     compute task
6: end while
7: update Fock matrix DArray via accumulate
8: for Every process p
9:     while a task remains in p's queue  /* fetch_add remote int */
10:        get remote blocks
11:        compute task
12:    end while
13:    update Fock matrix DArray via accumulate
14: end for
```

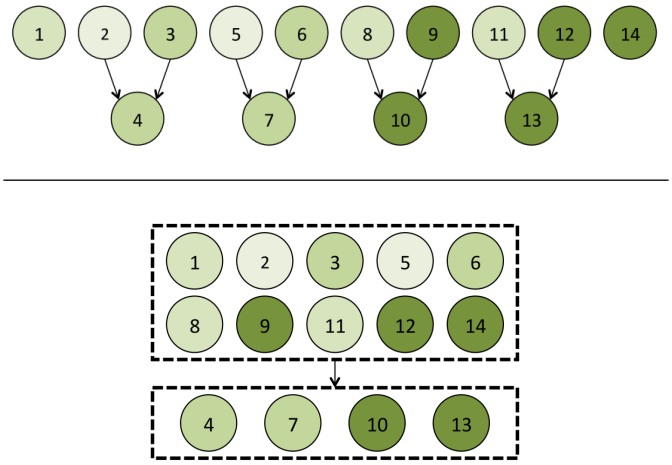


Fig. 4: An example of scheduling DAG dependencies across tensor contractions in coupled cluster doubles (CCD). This image is from [56].

3.4.6 Scientific Workflows in QM

This section briefly highlights work in scientific workflows applied directly to QC. In particular, the MoSGrid project [68–70] has established a science gateway using web-based services for end users to design complex workflows and meta-workflows. MosGrid has considered 3 interesting general workflow use cases in QC. Fig. 5 shows the workflow diagrams for 1) high-throughput analysis of X-ray crystallography files, 2) transition state (TS) analysis and 3) parameter sweeps. While these use cases are extremely interesting in terms of automation, more needs to be done to establish how such workflows and *collections* of workflows should most efficiently be scheduled, and how resources

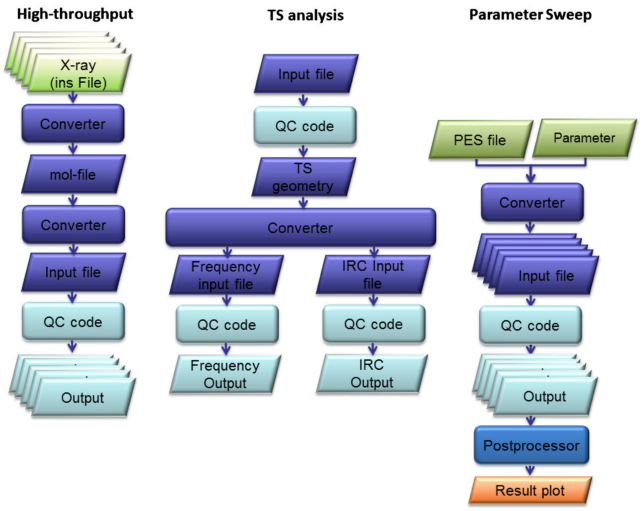


Fig. 5: 3 QM workflow use cases supported by MosGrid. This diagram is from [70].

should be allocated, to accomplish such workflows. Workflow management is clearly becoming more important in modern scientific endeavors, and as we will see in following sections, this is particularly true for multiscale and multiresolution computational chemistry.

4 HPC in Molecular Mechanics/Dynamics

This section highlights the primary aspects of molecular dynamics simulations, and how they have evolved alongside developments in HPC. We consider the most scalable and extendible molecular dynamics software frameworks and how they do and do not address the challenges of extreme scale computing listed in Section 2.

4.1 Introduction and Basics

Molecular Dynamics (MD) simulation is a type of N-body simulation in which the particles of interest are atoms and molecules. MD simulations differ greatly from quantum mechanical simulations: Instead of solving the Schrödinger equation, they solve Newton's equations of motion, which model the classical physics of particle-particle interactions. While MD mostly ignores quantum electronic properties of atoms, it is still able to capture relevant thermodynamics, dipoles, and some reaction pathways/mechanisms. The most important component of accurate MD simulations is the *potential function* (sometimes called the force field or the interatomic potential), which describes how atoms will interact based on their positions. Incidentally, the evaluation of the potential function is also the most expensive computational component of MD simulations. Potentials must account for all the different forms of energy within molecular systems, both bonded (bond stretching, valence angle bending, and dihedral torsions) and non-bonded (electrostatics and van der Waals forces).

The potential energy function of most MD simulations includes the following contributions:

$$U_{\text{total}} = U_{\text{bond}} + U_{\text{angle}} + U_{\text{dihedral}} + U_{\text{VDW}} + U_{\text{Coulomb}} \quad (6)$$

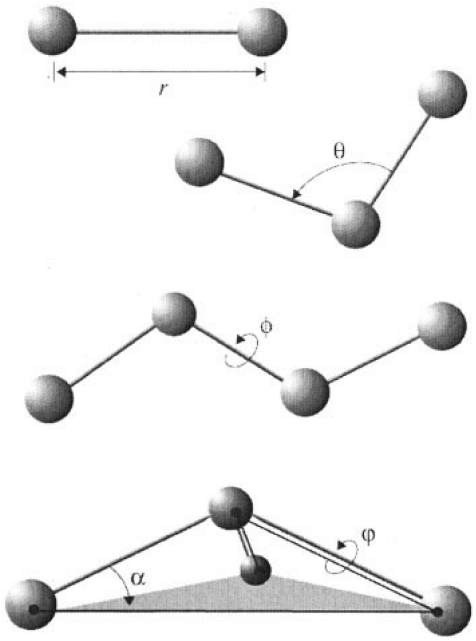


Fig. 6: Diagram showing U components. This diagram is from [71].

U_{bond} corresponds to the bond stretching (as in the top 2-atom molecule in Fig. 6), which in the simplest non-rigid case is modeled as a harmonic spring:

$$U_{\text{bond}} = \sum_{\text{bond } i} k_i^{\text{bond}}(r_i - r_{i(eq)})^2 \quad (7)$$

Similarly, U_{angle} is the bond angle term (as in the 3-atom molecule in Fig. 6), which is also often modeled harmonically:

$$U_{\text{angle}} = \sum_{\text{angle } i} k_i^{\text{angle}}(\theta_i - \theta_{i(eq)})^2 \quad (8)$$

$U_{dihedral}$ corresponds to the torsional interactions between 4 atoms (see angle ϕ in Fig. 6) and is usually more sinusoidal in nature:

$$U_{\text{dihedral}} = \sum_{\text{dihedral } i} \begin{cases} k_i^{\text{dihedral}}[1 + \cos(n_i\phi_i - \gamma_i)], n_i \neq 0 \\ k_i^{\text{dihedral}}(0_i - \gamma_i)^2, n = 0 \end{cases}$$

U_{VDW} is the van der Waals term, which accounts for dipole-related forces, and is usually described by some sort Lennard-Jones potential:

$$U_{\text{VDW}} = \sum_i \sum_{j>i} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \quad (9)$$

Finally, U_{Coulomb} accounts for the more long-range electrostatic interactions, which is almost always described by the standard potential energy between two point charges:

$$U_{\text{Coulomb}} = \sum_i \sum_{j>i} \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} \quad (10)$$

An important concept in MD simulation is the enforcement of periodic boundary conditions (PBCs). Normally, an MD simulation consists of a simulation cube of specified length L (but some software supports more advanced

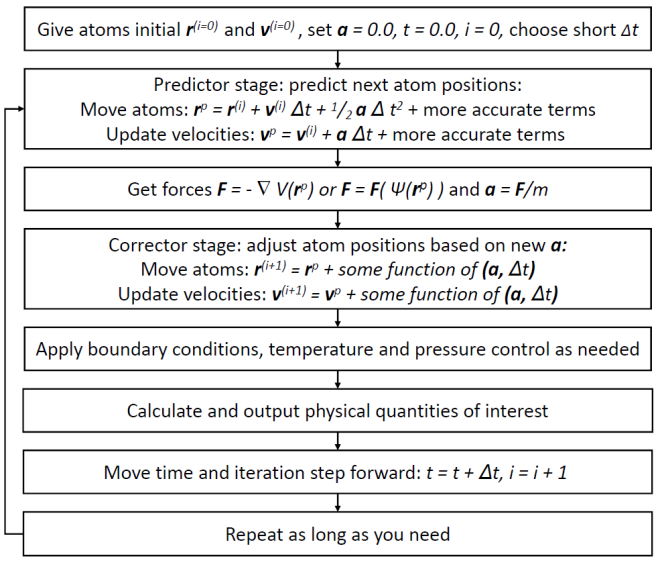
shapes such as parallelepipeds). What happens if a particle leaves the simulation box? PBCs suggest that the particle should re-enter at the opposite face of the cube. In this way, a simulation of a unit cell with PBCs is equivalent to simulating an infinite system with a repeating unit cell, as illustrated in Fig. 7. The PBC technique upholds thermodynamic consistency, and is good for spatial load balancing, which we discuss in Section 4.3.1.

The basic MD algorithm is shown in Algorithm 3. In summary, we first apply an integrator at each time step to determine the movements of atoms. Then we satisfy any necessary boundary or thermodynamic conditions, calculate any desired quantities, and ultimately move on to the next timestep.

In MD software implementations, the naïve approach is to calculate the particle interactions from Eqn. 6 for all unique pairs. This approach is $\mathcal{O}(n^2)$, which is prohibitive for systems of moderate size. However, it is standard practice to exploit the short-ranged nature of certain components of the total potential energy function. For instance, the first 4 terms of Eqn. 6 are for very short-ranged bonded atoms. Furthermore, the Lennard-Jones potential from Eqn. 9 quickly drops to zero as r increases. By imposing a *cutoff* distance outside of which to neglect any short-range particle interactions, we can drastically reduce the complexity of the simulation to $\sim \mathcal{O}(n)$. Unfortunately, the Coulombic component of the potential is long-ranged by nature, because it drops of as $1/r$. However, methods using Ewald summations can exploit the power of the fast Fourier transform (FFT) to very quickly calculate this long-range potential, which we consider in more detail in Section 4.3.2.

Finally, it is standard for each particle to have a *neighbor list*, which is queried during each simulation timestep to efficiently track necessary particle interactions [73] within a given cutoff distance. This has been deemed the most sensitive and important parameter for performance of MD simulations [17]. It is even more important for parallel implementations, because we must update the neighbor list as particles move outside of the relevant cutoff region. This involves frequent communication, particularly when the number of processes is high, the dynamics are fast, and/or the cutoff distance is large. We consider these issues in more detail in Section 4.3.1.

Algorithm 3 The standard MD simulation algorithm [72].



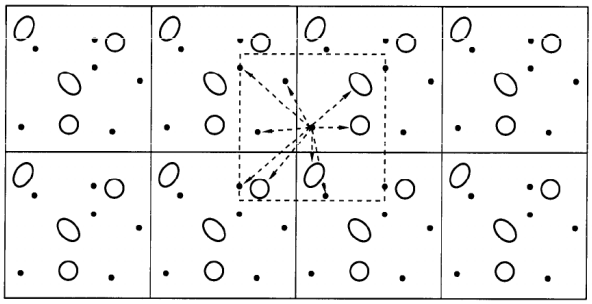


Fig. 7: An example of periodic boundary conditions. This graphic is from [74].

4.2 MD Force Fields

The previous section presented very simple forms for the potential energy components in Eqns. 7, 8, 9, and 10. Together, these simple functions comprise a *force field* for MD simulation, which can also be thought of as a software implementation for evaluating the potential energy function from Eqn. 6. There are *many* such force field implementations, and each is relevant to specific chemical systems of interest. Perhaps the primary challenge of MD simulation studies is that there is currently no general purpose force field - it is necessary to choose the best fit for a given chemical system.

Just as DFT has a very large number of functionals available, so too are there a large number of force fields. The primary categories are classical force fields, polarizable force fields, coarse-grained force fields, and water models. For MD, the more popular options include AMBER, CHARMM, GROMOS, and UFF (the latter being an attempt at a *general* force field) [30].

The total combinatorial space of atoms, molecules, chemical systems, simulation techniques, and force fields is dauntingly large. Interesting work considers high-throughput computational materials design to alleviate the burden of exploring the entire space [5, 9, 11, 12]. These projects combine MD and QM methods with data mining and large-scale database management to generate and analyse enormous amounts of simulation data towards the discovery and development of novel materials. However, little work considers machine learning techniques for choosing appropriate force fields given an input chemical system to minimize error, despite the clear similarity to other applications [75]. This may be promising future work.

4.3 High Performance MD Software Frameworks

The diversity of MD simulation software frameworks is dauntingly diverse, however, they can generally be grouped into two different camps. Many MD simulation frameworks focus primarily on atomistic modeling, such as NAMD [71], AMBER [76], DL_POLY [76], CHARMM [77], and Simpatico [78]. Such frameworks require subtle improvements

to force field parameters, and high-level features such as workflow support, but in general the algorithmic developments are well-established [79]. On the other hand, mesoscopic simulation frameworks that model soft matter are relatively less established, because their parameters are more dependent on the chemical system of interest, which may require more complex parameter tuning, and many different modeling methods exist, some of which are more applicable to a particular system setup. The latter type of framework includes LAMMPS, GROMACS, and ESPResSo^{++}. These simulation frameworks are the subject of the following 3 sections.

4.3.1 LAMMPS and Parallel Decomposition Schemes

LAMMPS is a classical molecular dynamics (MD) code which emphasizes parallel performance and scalability. It deploys distributed-memory message-passing parallelism with MPI using a spatial decomposition of the simulation domain. It has GPU (CUDA and OpenCL) and OpenMP capabilities, and also runs on the Intel Xeon Phi. LAMMPS runs from an input script, which has advantages, such as a low barrier to entry, but also disadvantages, because the input script language is unique and difficult to extend. LAMMPS supports modeling atoms, coarse-grained particles, united-atom polymers or organic molecules, all-atom polymers, organic molecules, proteins, DNA, metals, granular materials, coarse-grained mesoscale models, point dipole particles, and more (including hybrid combinations).

Pivotal work in comparing spatial-decomposition, force-decomposition, and atom-decomposition performance [80].

Atom decomposition: With N total atoms and P total processors, the *atom decomposition* simply assigns N/P atoms to each processor. Throughout the simulation, each processor calculates forces and updates the positions only of the atoms which they are assigned. This decomposition implies that a processor will compute all interactions associated with a given atom. This is generally considered the simplest decomposition scheme, but it may require extra communication because each process requires the coordinates of atoms from many other processors. However, if the all-to-all operation is well-implemented on a given network, then this operation is reasonably efficient, particularly when accounting for Newton's 3$^{\text{rd}}$ law to reduce computation in half. Load balance in atom decomposition is good if the row blocks in F have the same number of zeros. This is the case if atomic density is relatively uniform throughout the simulation box. If not, then it helps to randomly permute the number schemes to be less influenced by geometric arrangement.

Force decomposition: In the force matrix F, each element F_{ij} represents the force that atom i feels due to atom j. F is quite sparse because of the fast $1/r^d$ dropoff (d is a positive integer ≥ 2) of short-range potentials combined with the practical use of a pairwise cutoff. Also, it is easy to see by Newton's 3$^{\text{rd}}$ law that F is a symmetric matrix. With the atom decomposition described above, each processor is assigned N/P rows of the F matrix (or equivalently N/P columns). On the other hand, the *force decomposition* technique distributes F into smaller blocks, each $N/\sqrt{P} \times N/\sqrt{P}$. This has the advantage of improving communication costs to $\mathcal{O}(N/\sqrt{P})$ compared to $\mathcal{O}(N)$ in atom decomposition [80]. However, load balance may be worse in force decomposition because the blocks must be uniformly sparse in order for processors to do equal amounts of work. Even if atom density is uniform, geometrically ordered atom identifiers create diagonal bands in F. As in atom decomposition, a random permutation of the numbering scheme helps to reduce load imbalance.

Spatial decomposition: Spatial decomposition involves subdividing the simulation domain into separate 3D boxes, and assigning atoms within a box to a particular processor. As atoms move through the simulation domain, they are reassigned to the appropriate processor based on the spatial decomposition. While it may seem that spatial decomposition may suffer from poor load balance for simulations with spatially-varying density distributions, there are now several options in LAMMPS to improve the load balance. For instance, the `balance` command applies a recursive multisection algorithm to adjust the subdomain sizes in each dimension of the simulation box. Furthermore, when using multi-threading in LAMMPS, an atom decomposition is used for on-node parallelism, whereas spatial decomposition is used for off-node parallelism.

In summary, there are 3 practical ways to decompose MD simulations, and LAMMPS developers were among the first to study and analyze these alternatives; furthermore, their findings influenced many other future implementations. Figure 8 shows their results comparing the 3 decomposition schemes using MPI on a Cray machine [80]. We see that the best choice of decomposition scheme depends on both the scale and the chosen cutoff. This phenomenon was relevant on the outdated Cray T3D, and is certainly relevant today with the diversification of on-node architectures/memory hierarchies, network topologies, and programming models. Many recent exploratory MD applications (some on the latest many-core architectures [81]) deploy strict domain decomposition [82] based on such pivotal work as this. However, it is important to keep in mind that the other alternatives (or hybrid schemes) may be better suited to extreme scale systems where concepts such as minimizing communication and exploiting vectorization are paramount for scalable performance.

4.3.2 GROMACS and Ewald Summation Techniques

This section highlights the unique aspects of the GROMACS MD framework that improve its performance and scalability. In order to understand, a brief introduction to Ewald summations and particle-mesh Ewald (PME) is necessary (a more thorough introduction is found in [74]). The long-range component of the potential energy function is shown in Eqn. 10. The goal for this electrostatic interaction is to solve Poisson's equation given the function for U_{Coulomb}. Ewald summation involves adding a Gaussian function to each point charge in the system, as shown in Fig. 9. In short, we do this because the Gaussian representation has an *exact* solution to Poisson's equation in reciprocal space.

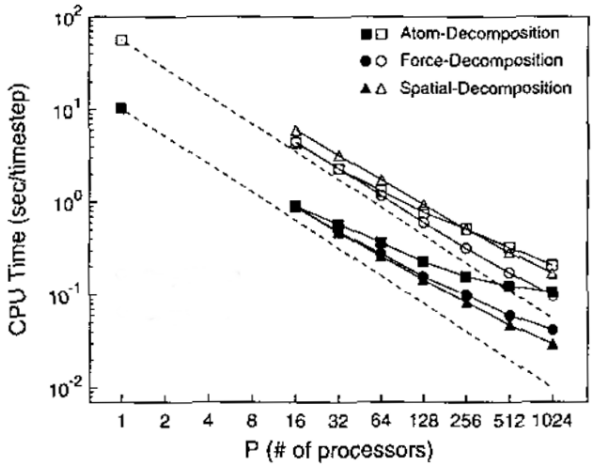


Fig. 8: Comparison of the 3 MD decompositions for two different cutoff lengths. Although these results are dated, the concepts are important for understanding the consequences of domain decomposition techniques when porting MD codes to modern hardware systems using productive programming models. This figure is from [80].

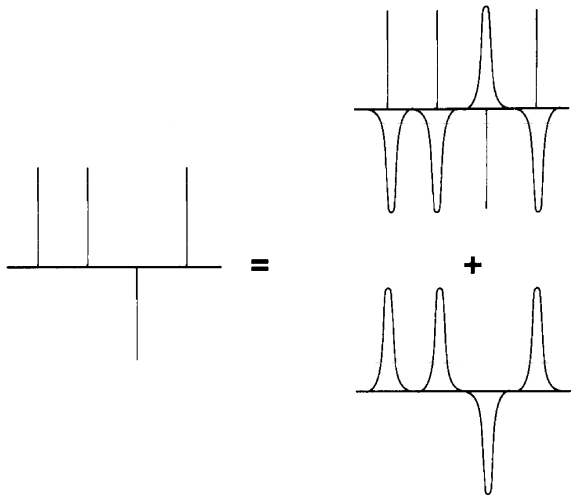


Fig. 9: Ewald summations work by adding Gaussians to the point charges [84]. This enables an exact solution of Poisson's equation, and a very fast FFT evaluation of the long range Coulombic component of the potential energy function. This plot is from [74].

This means that if we transform the spatial coordinates into frequency space using Fourier techniques, then we can trivially solve Poisson's equation across the entire simulation domain. The FFT accomplishes this feat by imposing a domain that is a power of 2 units long in each spatial dimension, then enforcing the point charges to be centers on the nearest grid point.

Unfortunately, FFT scalability does not keep up with the scalability of the short-range component of MD simulations using a given number of processes. However, GROMACS 4.5 was the first framework to solve this problem, and now NWChem utilizes the same technique [83]. This method uses a "pencil" decomposition of reciprocal space, as displayed in Fig. 10. Here, a subset of the total MPI application processes (implemented via MPI communicators) participate in the FFT calculation. At the beginning of each timestep, the direct-space nodes (top of Fig. 10) send coordinate and charge data to the FFT nodes (bottom of Fig. 10). Some number of direct-space nodes (usually 3-4) map onto a single reciprocal-space node. Limiting the computation of the FFT to a smaller number of nodes significantly improves parallel scaling [84], likely due to the communication boundedness of 3D FFTs.

Modern GROMACS (version 5.1) utilizes hierarchical levels of parallelism, as shown in Fig. 11. First, many MD simulations depend on collections of similar instances, which together make up a statistically significant *ensemble*. GROMACS supports tools for easily constructing such ensembles and deploying them on a computer cluster. Second, each simulation within an ensemble is spatially domain decomposed and dynamically load balances over MPI. Third, non-bonded interactions are handled on GPUs because of the short-ranged calculations compatibility with simultaneous multithreading (SIMT). Finally, CPU cores apply SIMD operations to parallelize cluster interactions kernels or bonded interactions, which now makes heavy use of OpenMP pragmas. This grouping of different components onto the best suited device within the heterogeneous architecture is what makes GROMACS a front-runner in MD simulation frameworks, especially in light of the push towards exascale capability.

4.3.3 NAMD

Here, we briefly consider the NAnaoscale Molecular Dynamics (NAMD) framework, which has historically prioritized the strong scaling capabilities of MD simulations more than other frameworks [71]. It's capabilities are similar to that of GROMACS and LAMMPS, but it runs over a more interesting parallel runtime, Charm++. In Charm++, C++ objects called *chares* represent migratable tasks that execute asynchronously and in a one-sided manner. Users are encouraged to "over-decompose" their tasks into chares, because a finer granularity leads to better load balance and adaptability. The runtime of Charm++ manages chares with a system of queues on each compute node with the goal of hiding latency and promoting asynchronous execution.

NAMD deploys a spatial decomposition into so-called "patches" that fill the simulation box. Unlike other common approaches, the number of patches in NAMD is unrelated to the total number of processes. Charm++ encourages this separation of problem decomposition from hardware resources, because it allows for adaptability in the execution. Furthermore, the assignement of patches to processors can be changed between iterations based on measurements made by the Charm++ runtime. The occasional redistribution of tasks at runtime gives NAMD a key scalability advantage in terms of load balance.

We finally briefly mention some of NAMD's auxiliary accomplishments. First, NAMD researchers explored the implications of multi-lingual programming at scale [85], which led to a rich and flexible software architecture, shown pictorially in Fig. 12 (though this architecture is now out of date). Second, FPGA researchers compiled NAMD with

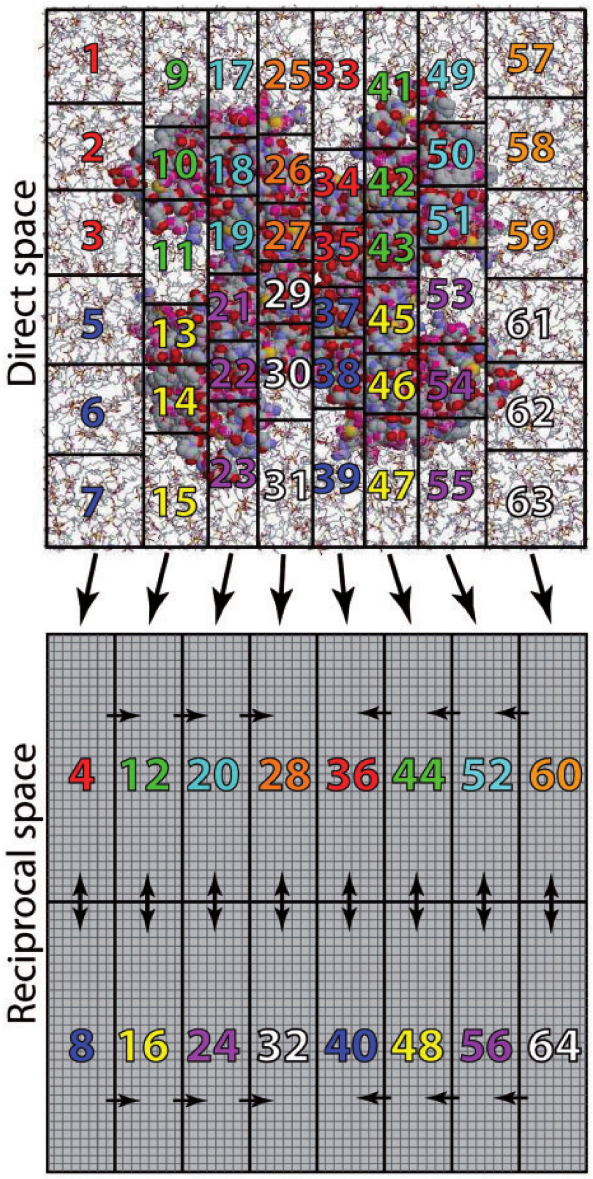


Fig. 10: GROMACS 3D spatial decomposition combined with 2D decomposition in reciprocal space. This diagram is from [84].

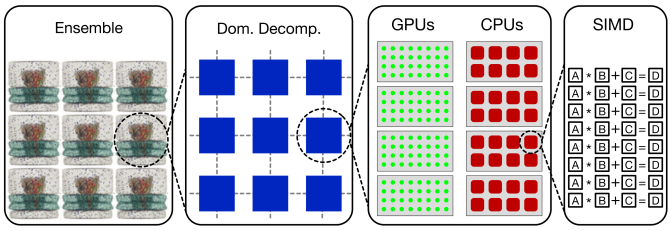


Fig. 11: The hierarchical levels of parallelism in GROMACS. This diagram is from [83].

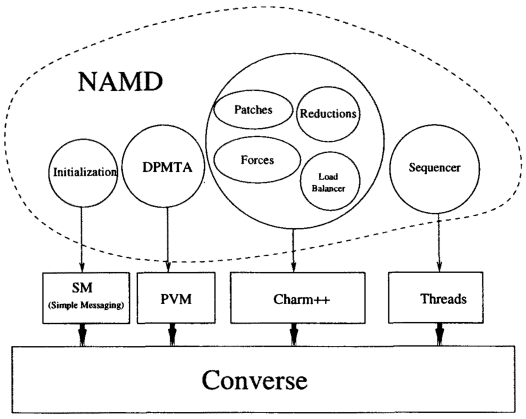


Fig. 12: NAMD's language architecture. This diagram is from [85].

ROCCC, which resulted in an impressive speed-up of the critical region that computes non-bonded interactions [86]. They focused on generating hardware that maximizes parallelism in the FPGA circuit while minimizing the number of off-chip memory accesses.

4.3.4 *ESPResSo^{++}*

ESPResSo^{++} is a re-write of the popular ESPResSo MD software toolkit, which primarily supports simulation of soft matter physics and physical chemistry. Unlike alternatives such as LAMMPS and GROMACS, ESPResSo^{++} boldly prioritizes software *extensibility* over all other concerns during development of MD software, which often (but not always) includes performance [79].

ESPResSo^{++} supports the extensibility of MD simulations by enhancing readability, maintainability, expandability, and verifiability by means of a well-defined coupling between two software interfaces. First, ESPRessSo^{++} consists of a high-level Python-based interface used for 1) importing relevant modules and packages, 2) setting up the chemical system and its interactions, 3) simulation of the system dynamics, and 4) analysis. ESPResSo^{++} also consists of a lower-level C^{++} interface to the simulation engines and various computational tools. Development of new modules must follow a specific protocol for exposing necessary C++ classes to the Python user interface using the Boost.Python library.

This design is quite useful for writing adaptive simulations with in-situ analysis capabilities. For example, to visualize a plot of the pressure every 100 timesteps using a Python library is trivial:

```
integrator = \
        espresso.integrator.VelocityVerlet(system)
...
for i in range(10):
   integrator.run(100)
   P = espresso.analysis.Pressure(system).compute()
   matplotlib.plot(P, len(P))
```

While this is certainly possible with LAMMPS using the following lines in an input script:

```
thermo_style custom pressure
thermo 100
run 10000
```

it is far more difficult to do something as simple as display a plot every 100 timesteps without resorting to custom interprocess communication (which is not clearly possible without changing LAMMPS source code). Not only is this trivial for ESPResSo^{++}, one can go a step further and

visualize simulation progress on-the-fly by sending data over a socket to VMD server:

```
# VMD initialization
sock = espresso.tools.vmd.connect(system)
...
# Send current positions to VMD
espresso.tools.vmd.imd_positions(system, sock)
```

While this is certainly possible with LAMMPS using the following lines in an input script:

```
thermo_style custom pressure
thermo 100
run 10000
```

it is far more difficult to do something in LAMMPS as simple as display a plot every 100 timesteps. This workflow-oriented capability is inherently included in the ESPResSo^{++} software design. The parallel programming model for ESPResSo^{++} is unique in that all communication is done through MPI, but user programs *do not apply an SPMD model*. This unexpected architecture is shown in Figure 13, where a program using the Python scripting interface invokes a controller that launches MPI4Py parallel programs [87]. Furthermore, processes can either communicate through the Python layer of abstraction, or through the C++ layer, as shown in Figure 13. using the Boost.MPI library, ESPResSo^{++} has been shown to be scalable up to 1024 cores, as shown in Figure 14. While the performance is not as good as LAMMPS at scale, its prioritization of extensibility have allowed for more advanced features such as adaptive resolution, and support for various course grained potentials. Profiles suggest that the performance differences between ESPResSo^{++} and LAMMPS are due to optimizations made within the neighbor list construction and the communication of particle data [79]. In this author's opinion, more needs to be done to quantify the communication overheads. For instance, there are may be overheads associated with using Boost.MPI to send serialized object data instead of sending MPI derived types, as LAMMPS does.

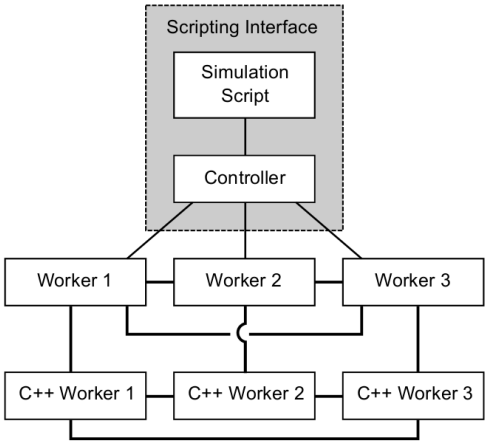


Fig. 13: The parallel execution model of ESPResSo^{++}. This diagram is from [79].

One reason we emphasize ESPresSo^{++} in this paper is because it does a good job addressing most of the extreme-scale requirements listed in Section 2. It emphasizes a modular software infrastructure (sometimes sacrificing absolute best performance), enables scientific workflows as an integral part of the software design, and supports multiphysics/multiscale component coupling in the form of adaptive resolution.

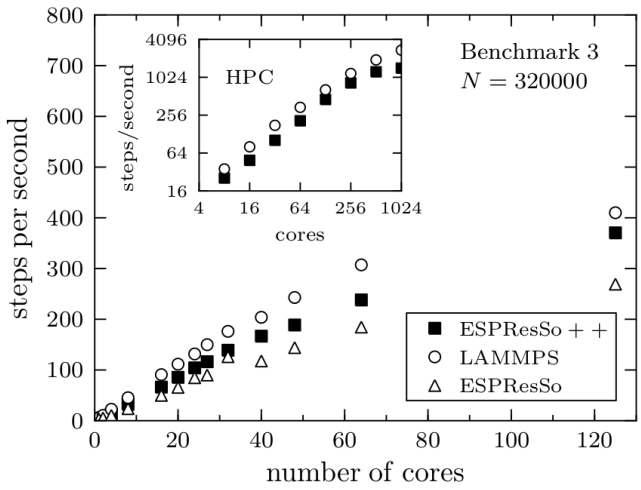


Fig. 14: ESPResSo^{++} strong scaling compared to LAMMPS. These results are from [79].

4.4 Other Noteworthy MD Accomplishments in HPC

This section briefly mentions other noteworthy accomplishments in MD software implementations that utilize HPC. First of all, most key MD frameworks are GPU-accelerated, including ACEMD, AMBER, BAND, CHARMM, DESMOND, ESPResso, Folding@Home, GPUgrid.net, GROMACS, HALMD, HOOMD-Blue, LAMMPS, Lattice Microbes, mdcore, MELD, miniMD, NAMD, OpenMM, PolyFTS, SOP-GPU and more [61]. Other codes, such as ESPResSo^{++} are currently porting and optimizing GPU kernels.

On MICs, miniMD has implemented and published on very effective vectorization techniques that run on the Xeon Phi [81]. LAMMPS, GROMACS, and NAMD also have support for MIC [83], but more optimizations need to be made to compete with the CPU implementation and to balance work between CPU and devices. Incorporating vectorization into full-fledged MD simulation frameworks on *socket-attached* MIC architectures such as Knights Landing and Knights Hill is a very promising area for future work.

Custom alternative architectures also show *remarkable* potential for MD simulation. For instance, researches have ported NAMD benchmarks to an FPGA and report 800x improvement for single-precision performance and 150x improvement for double precision over a conventional x86 architecture [86]. Also, Anton [1, 2], a special purpose supercomputer designed from the ground up to be optimized to MD simulations, shows impressive performance improvements over general purpose machines, with speedups of up to 180x.

5 Coarse-Grained Molecular Simulations

On the largest modern supercomputers, molecular dynamics (MD) simulations of polymer systems contain billions of atoms and span roughly a few nanoseconds of simulation time per week of execution time. Unfortunately, most

macromolecular processes of interest contain many orders of magnitude more particles and often bridge microsecond or even millisecond timescales or longer. These include phenomena like microphase separation transition in block-copolymers [88], phase separation in polymer blends and composite materials [89], polymer crystallization, and glass formation and aging [90] to mention just a few. Despite our pervasive access to massive computing power, full *united-atom (UA)* simulations do not come close to representing real-world polymer systems (see Figure 17), because they are too computationally expensive and slow. Simply put, we require new approximation methods that capture the relevant physics and chemistry while requiring fewer computational resources. The most promising approach is the *coarse-graining (CG)* method, in which groups of atoms are represented as one collective unit. CG has proven to be valuable for eliminating unnecessary degrees of freedom and tackling the scaling complexity of larger problems [91]. The key issue is how to simultaneously maintain solution accuracy and high performance.

The following sections describe some popular approaches for conducting accurate CG simulations. They fall into two broad categories: numerically based and theoretically based. Section 5.1 considers a new software framework called VOTCA that provides an interface to several numerical CG approaches. Section 5.2 considers a more theoretically driven CG technique called integral equation coarse-graining that relies on principles in statistical mechanics to derive an analytical potential that describes CG models of polymer melts.

5.1 VOTCA

There exist several different techniques for applying CG models to molecular systems, such as iterative Boltzmann inversion, force-matching, and inverse Monte Carlo. We begin our discussion of CG techniques by considering the Versatile Object-Oriented Toolkit for Coarse-Graining Applications (VOTCA), because it provides a unified framework that implements many different CG methods and allows their direct comparison [92]. The momentous publication that describes VOTCA [92] provides a nice comparison of these numerically-driven CG techniques, and they are the subjects of the following 3 sections.

5.1.1 Boltzmann Inversion

Boltzmann inversion (BI) is considered to be the simplest method for deriving a CG potential, because it simply involves inverting the distribution functions of a coarse-grained system. Specifically, the bond length, bond angle, and torsion angle distributions are sampled from a simulation trajectory, then inverted to derive the desired potential functions. For example, sampled bond lengths may be fitted to a Gaussian:

$$p(r) = \frac{A}{\omega\sqrt{\pi/2}} \exp\left[\frac{-2(r - r_{eq})^2}{\omega^2}\right]$$

Now, exploiting the fact that a canonical ensemble obeys the Boltzmann distribution between independent degrees of freedom:

$$p(r) = Z^{-1} \exp[-\beta U(r)]$$

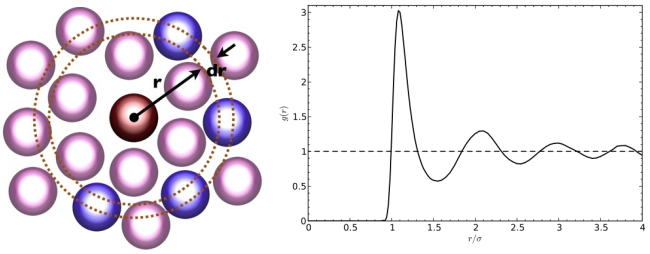


Fig. 15: An illustration of the radial distribution function, $g(r)$. The left image shows a liquid with "solvation shells" surrounding a central atom. The $g(r)$ function on the right shows the statistical likelihood of a solvation shell at a given distance r, which decreases to zero at large distances.

(where Z is the standard partition function), this can be inverted to derive the desired harmonic potential:

$$U(r) = -k\beta T \ln[p(r)] = K_r(r - r_{eq})^2.$$

Here, Z becomes an irrelevant additive constant to the CG potential [93].

This simple approach has disadvantages. First, assumes independent degrees of freedom:

$$\begin{aligned} P(r,\theta,\phi) &= \exp[-\beta U(r,\theta,\phi)] \\ P(r,\theta,\phi) &= P_r(r)P_\theta(\theta)P_\phi(\phi) \end{aligned}$$

which may not be true for some systems; however, it is *is* true, then BI is an *exact* CG representation of the potential. Another problem is that we require smoothing $U(q)$ to provide a continuous force, which can be accomplished with extrapolation. Finally, we require an atomistic reference system to accomplish BI, be we would prefer a CG model that is independent of any full atomistic simulation.

5.1.2 Iterative Boltzmann Inversion

Iterative Boltzmann Inversion (IBI) is very similar to Boltzmann Inversion from the previous section except that the non-bonded CG potential is iteratively updated until it matches the corresponding radial distribution function (RDF) of the atomistic representation:

$$U_{i+1}(r) = U_i(r) - \alpha k_B T \ln\left[\frac{g_i(r)}{g_t(r)}\right]$$

where α is a scaling factor (to reduce large deviations between iterations), $g_i(r)$ is the RDF of the i^{th} iteration and $g_t(r)$ is the target RDF. The RDF is an extremely important quantity in MD simulation studies, because if it is known, it can be used to derive thermodynamic quantities. Fig. 15 shows a graphical representation of an RDF for a liquid chemical system.

5.1.3 Inverse Monte Carlo and Force Matching

Two other popular numerical methods for deriving CG potentials are Inverse Monte Carlo (IMC) and Force Matching (FM). An in depth description of these techniques is unnecessary for this discussion, but it suffices to say that IMC is quite similar to IBI, except that it is based on more rigorous thermodynamic arguments. IMC has advantages over IBI in that it shows better and faster convergence, but also is more expensive computationally.

The FM approach is quite different from IMC and IBI. It is non-iterative like BI, but instead of generating distribution functions from reference simulations, FM generates *force functions* with the goal of matching CG units to atomistic units. It is non-iterative, and therefore less computationally expensive.

5.1.4 Iterative Workflow

VOTCA enables the direct comparison of the above CG methods (and potentially many more) by providing a modular interface to a workflow template for generating CG potentials. Fig. 18 shows this iterative workflow. Such capabilities are important for computational chemists because they allow for controlled verification, validation, and testing of new techniques across several different development teams of MD software. Sections below will present a large variety of multiscale and multiresolution simulation methods that would also benefit from a VOTCA-like simulation framework for comparing workflows. One current drawback of VOTCA, however, is that it is currently focused on numerically-driven techniques for deriving CG potential and force functions, despite the fact that more consistent techniques exist in analytically-driven techniques for deriving CG potentials. This approach is the subject of Section 5.2 below.

5.2 Integral Equation Coarse Graining Theory

The *Integral Equation Coarse-Grained (IE-CG)* model by Guenza and coworkers [96–101] adopts an analytically-derived potential and dramatically improves spatial and temporal scaling of polymer simulations, while accurately preserving thermodynamic quantities and bulk properties [102–104]. Several numerical techniques and force fields exist for performing coarse-grained simulations [105–107]. However, these methods generally preserve either structure or fully preserve thermodynamics, but not both. As a result, only a small level of coarse-graining is typically adopted to limit the errors in the simulated structure and thermodynamics. In contrast, the IECG model adopts the analytical approach offered by statistical mechanics theory, because it recovers crucial structural and thermodynamic quantities such as the equation of state, excess free energy, and pressure, while enabling a much higher level of coarse-graining and the corresponding gains in computational performance.

Although CG polymer physics is a mature field, little has been done to analyze the performance benefits of CG versus UA representations. While it is clear that CG will exhibit computational gains, does it strong scale to as many processors as the corresponding UA simulation? Likely not, because CG tracks far fewer overall particles, sometimes by orders of magnitude. Also, the scalability of CG simulations likely depends on the granularity factor, e.g, the number of UA coordinates a CG unit represents. Despite the purpose of CG research to improve computational efficiency of MD simulations, the relevant literature lacks measurements that quantify expected computational performance of various CG techniques, and how they scale across different supercomputer architectures. Furthermore, CG related parameters may be chosen to give the absolute performance, but do they also give the best accuracy? Two primary goals

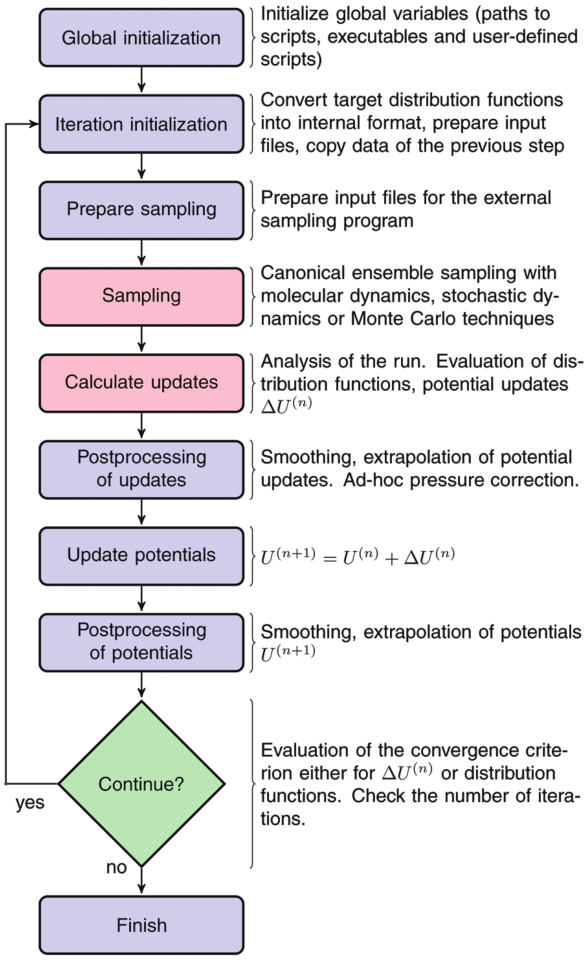


Fig. 16: VOTCA's iterative workflow. The diagram is from [92].

of this paper are to quantify expected performance of IE-CG simulations at large scale, and to evaluate the trade-off between granularity, scalability, and accuracy.

When considering homopolymers, the IECG model represents polymer chains as collections of monomer *blocks* that consist of many monomers and interact via soft long-range potentials. If each chain has N monomers, we say that there are n_b blocks per chain with N_b monomers per block. The most important quantity used to derive the CG potential is the *time correlation function*, which is essentially a measure of the pairwise influence between particles as a function of

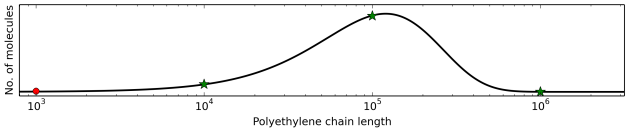


Fig. 17: A representation of the average polyethylene chain length determined by chromatography experiments [94]. Most studies are limited to very short chain lengths (≤ 1000) due to the prohibitive cost of UA simulations, but recent work freely explores the realistic systems with 10^4 to 10^6 monomers per chain [95].

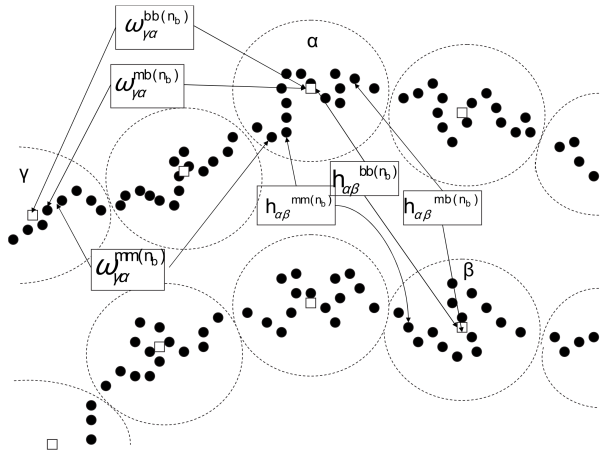


Fig. 18: A representation of the different block averaged components of the IECG equations. The diagram is from [97].

distance. They are related by the radial distribution function, $g(r)$ which is defined as:

$$g(r) = \frac{1}{\rho}\left\langle \frac{1}{N}\sum_{i}^{n}\sum_{j\neq i}^{n}\delta(\vec{r}-\vec{r}_{ij})\right\rangle$$

the total correlation function relates to $g(r)$ by:

$$h(r) = g(r) - 1.$$

The Polymer Reference Inter Site Model (PRISM) site-averaged Ornstein-Zernike equation relates the relevant time correlation functions in Fourier Space:

$$\hat{h}^{mm}(k) = \hat{\omega}^{mm}(k)\hat{c}^{mm}(k)[\hat{\omega}^{mm}(k) + \rho\hat{h}^{mm}(k)]$$

from which we derive the CG potential. These equations contain a large number of terms, so they are neglected here, but can be found in the relevant literature [96–101]. Using this solution for the potential, CG simulations in LAMMPS have been shown to capture the relevant thermodynamics while drastically reducing the number of degrees of freedom [102–104]. Furthermore, using the analytical approach, there are far fewer tunable parameters than the numerical approaches such as Boltzmann inversion. Finally, numerical approaches often require a atomistic reference system, which to some degree defeats the purpose of gaining efficiency with CG. Assuming an informed value of the radius of gyration and the direct correlation function limit as $k \to 0$, called c_0 [103], the IECG potential only requires such reference systems for validation studies.

6 Multiscale / Multiresolution Chemistry

One of the grand challenges of modern molecular simulation is bridging length and time scales in physically meaningful and consistent ways [108]. Consider a biomolecular system such as a collection of eukaryotic cells. From a low-level chemical point of view, each cell consists of many different biopolymers such as DNA, proteins, sugars, enzymes, and upwards of 90% liquid water. From a higher-level biological point of view, intercellular communication (between cells) plays an important role in synaptic transmission, hormone secretion, and other signaling events, even at long distances. Currently, MD simulations are capable of modeling a few biopolymers in a water bath, and accurate QM simulations can reasonably simulate regions with a few dozen atoms. CG methods are capable of simulating much larger regions, but methods normally consist of repeating units of the same atomic or molecular type. Continuum mechanics goes one step further, and models materials as a continuous mass rather than a system of discrete particles. For example, the water within a cell can potentially be modeled with methods in hydrodynamics and fluid mechanics. Furthermore, it is known that biological processes occur over extremely long timescales, whereas MD simulations can reasonably only reach the order of microseconds.

In order to effectively and accurately model and simulate such complex systems, we require tractable and robust methods for bridging the QM, MM, CG, and continuum scales, both spatially and temporally, while still exhibiting the correct behavior at the edges of multiscale regions. This research area is expansive, consists of many different facets, and even contains cross-discipline similarities and analogies between topics in chemistry and physics [109, 110]. However, the following sections limit the scope of mutliscale and multiresolution methods to particle-based computational chemistry, with a particular focus on available software frameworks and methodologies, which are currently somewhat lacking possibly due to the difficulty of the problem.

6.1 QM/MM

QM/MM simulations consist of a relatively small region accurately modeled by QM methods, with the remaining regions modeled more efficiently by MM methods. Fig. 19 shows a simple example of a QM/MM hybrid simulation region. In this example, we presume the inner region is interesting because it contains some chemical reaction, which QM models well. The outer region, on the other hand, may contain no chemical reactions, but we still desire consistent thermodynamics and inclusion of long-range effects on the QM region. There are three types of interactions in this hybrid system: interactions between atoms in the QM region, interactions between atoms in the MM region, and interactions between QM and MM atoms. Sections 3 and 4 of this paper described the QM and MM interactions (respectively) in detail, and their evaluation within QM/MM simulations is no different. However, the interactions between the QM and MM regions are more complicated, and researchers have proposed several approaches for handling them.

One simple approach is called *subtractive QM/MM coupling*. The idea is fairly simple, and is represented by this equation:

$$V_{\mathrm{QM/MM}} = V_{\mathrm{MM}}(\mathrm{MM}+\mathrm{QM}) + V_{\mathrm{QM}}(\mathrm{QM}) - V_{\mathrm{MM}}(\mathrm{QM}) \quad (11)$$

Here, we claim that the total potential energy of the QM/MM system, $V_{\mathrm{QM/MM}}$, involves adding two terms and subtracting another term. Fig. 20 shows a representative diagram of the subtractive QM/MM coupling method. The first term on the right hand side of Eqn. 11 corresponds to the sum of the QM and the MM regions evaluated at the MM level, as shown in the third box from the left of

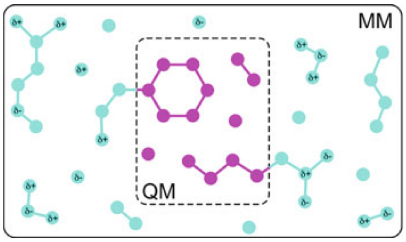


Fig. 19: QM/MM concept. Diagram is from [114].

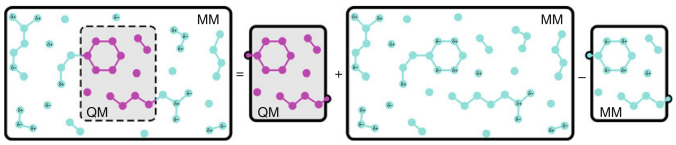


Fig. 20: QM/MM subtractive coupling. Diagram is from [114].

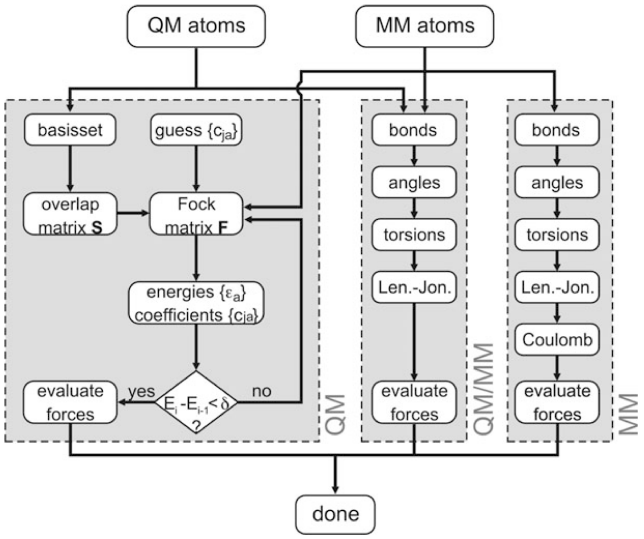


Fig. 21: QM/MM flow scheme. Diagram is from [114].

Fig. 20. The second term on the right hand side corresponds to the QM region evaluated at the QM level, as shown in the second box from the left of Fig. 20. Finally, the subtracted term corresponds to the QM region evaluated at the MM level, as shown in the rightmost box of Fig. 20. The most popular implementation of this approach is the ONIOM method [111], which is available in NWChem and Gaussian. Besides the subtractive QM/MM coupling scheme, there is also additive QM/MM coupling:

$$V_{\mathrm{QM/MM}} = V_{\mathrm{MM}}(\mathrm{MM}) + V_{\mathrm{QM}}(\mathrm{QM}) + V_{\mathrm{QM-MM}}(\mathrm{QM}+\mathrm{MM}) \tag{12}$$

in which the third term on the right hand side considers the interactions between QM and MM atoms explicitly. Another popular, but more sophisticated, approach is capping bonds at the QM/MM boundary. This approach cuts bonds crossing the QM/MM region boundary, and replaces them with something like link atoms or localized orbitals [112]. The specifics of this approach bring about complications that are beyond the scope of this paper, but are covered elsewhere [109, 110, 112–114].

Simulation frameworks that support QM/MM through various packages include NWChem, GROMACS, QuanPol, GAMESS (both US and UK version), and AMBER. The method has become so important for computational chemists that the inventors of QM/MM, Warshel and Levitt, won 2013 Nobel Prize in Chemistry [115]. Despite its importance, there is a dearth of research that considers the parallel performance of these workflows. This is puzzling given the fact that it usually relies on the tight coupling of different simulation modules, even between different frameworks [116]. For instance, Fig. 21 shows the overall control flow of a QM/MM simulation. At a glance, it is clear that this directed acyclic graph (DAG) of dependencies offers opportunity to exploit parallelism, but popular QM/MM frameworks do not support such dynamic execution. Future dissertation work could consider the performance of these methods in the context of scientific workflow management and how to efficiently share computational resources in the face of tight application coupling.

6.2 MM/CG

The MM/CG model is analogous to the QM/MM model, except that MM represents a smaller region more accurately, and CG models the remaining regions more efficiently. Such methods are particular important for multiscale simulations in which there are large regions with well-verified CG models, such as the solvent regions around active proteins. Interestingly, the first CG model of a globular protein was introduced in a 1975 paper by Warshel and Levitt [117], the recipients of the 2013 Nobel Prize in Chemistry. Even more impressive, this was the first multiscale MM/CG simulation, because side-chains of the protein were treated with atom-level detail. Since then, MM/CG has been explored in much more sophisticated scenarios [16, 79, 118–129]. Most notably, methods in *Adaptive Resolution Schemes (AdResS)* consider ways to specify regions having different granularities. AdResS is the subject of the following subsection 6.2.1.

In terms of software implementations, MM/CG is less ubiquitous than QM/MM, but is currently available in GROMACS (however, documentation is limited and support is constrained to certain systems such as those in which only the water solvent is coarse-grained). Other tools, such as the ChemShell environment [116] do apparently allow for flexible MM/CG setups [19], but again, thorough documentation is currently limited to QM/MM procedures. Finally, recent work by Ozog et al. enables switching between MM and CG representations in an automated way [130], which is a considerable step towards AdResS and and MM/CG in the LAMMPS framework.

6.2.1 *Adaptive Resolution (AdResS)*

Many QM/MM studies do not consider dynamical systems in which atoms freely move between the QM and MM regions. For MM/CG simulations however, this may be a more important requirement, because we ofter coarse grain in order to bridge longer timescales. The adaptive resolution, or *AdResS*, technique addresses the following requirements:

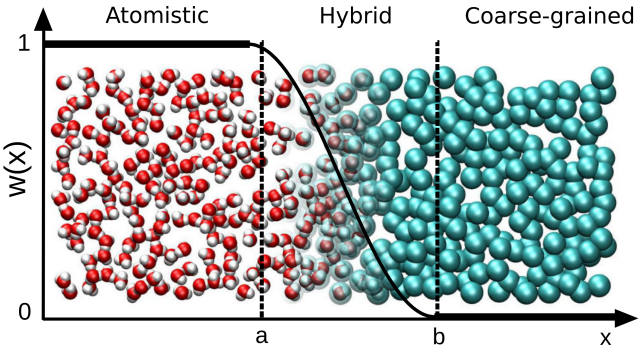


Fig. 22: A simple AdResS simulation box with the switching function, $w(x)$, overlaid.

1) Specification of MM and CG regions, analogous to QM/MM from Fig. 19; however, simpler AdResS experiments consider the scenario in Fig. 22.
2) Free exchange of atoms and molecules from the MM region to the CG region and vice versa.
3) Dynamics should occur under thermodynamic equilibrium, i.e., at the same temperature, density, and pressure throughout the simulation box.

L.D. Site and others have led several explorations [16, 108, 118–120, 123, 126, 128, 129, 131] of an AdResS technique with a hybrid region between MM and CG domains where the derived forces couple in the following manner:

$$\mathbf{F}_{\alpha\beta} = w(R_\alpha)w(R_\beta)\mathbf{F}^{atom}_{\alpha\beta} + [1 - w(R_\alpha)w(R_\beta)]\mathbf{F}^{cm}_{\alpha\beta} \quad (13)$$

This equation describes the force between two molecules as a function of their positions (in the x dimension). Here, α and β refer to two molecules, and the positions of their center of mass are X_α and X_β, respectively. $\mathbf{F}^{atom}_{\alpha\beta}$ is the force derived from the atomistic (MM) potential and $\mathbf{F}^{cm}_{\alpha\beta}$ is the force derived from the CG potential. The function $w(x)$ is a smooth switching function that is zero in the CG region and 1 in the MM region. With this force description, atoms moving in the hybrid region slowly lose (or gain) degrees of freedom. ESPResSo^{++}, from section 4.3.4 and GROMACS+MARTINI use this technique for doing AdResS simulations.

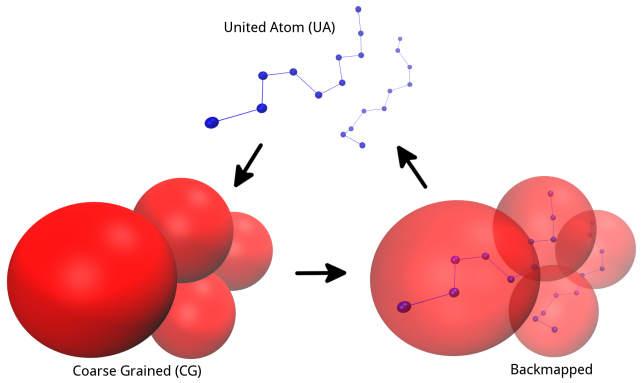


Fig. 23: The high-level progression of a UA↔CG workflow. The CG representation is calculated from UA coordinates, and the UA representation is recovered by solving a backmapping problem. CG spheres appear hard, but are soft with long range effects.

The model from Eqn. 13 is quite simple, and it clearly satisfies requirements 1 and 2 from above. However, to satisfy requirement 3, the authors must make non-trivial adjustments. For instance, an external source of heat must be introduced to each degree of freedom to assure that the temperature is constant through the hybrid region [108]. Also, an external force is added to assure thermodynamic equilibrium [132]. Interestingly, the IECG work discussed in Section 5.2 does not seem to require such adjustments for different levels of granularity. While Eqn. 13 may not be compatible with IECG in its form above, future work should consider whether something like a gradual (or sudden) change in *granularity* (for example, within a polymer melt) can be used for AdResS. Not only might this trivially satisfy requirement 3 without external forces/thermostats, but it is possible computation will be more efficient when using the analytically derived IECG potential.

6.2.2 Backmapping

Defining a new representation of the polymer as a chain of soft colloidal particles greatly reduces the amount of information to be collected and controlled, which speeds up the simulation. It is well known that modeling fewer colloidal particles with an appropriate potential decreases the degrees of freedom and computational requirements by an amount proportional to the granularity [91]. The representation of a polymer as a chain of soft blobs also allows the chains to more easily cross each other, decreasing the required time for the simulation to find the equilibrium structure. However, the information on the molecular local scale needs to be restored at the end of the simulation to account for properties on the UA scale. In a nutshell, it is important to alternate between the CG representation (which speeds up the simulation) and the UA representation (which conserves the local scale information). By quickly switching back and forth from UA to CG, we open doors to new studies of polymeric systems while maintaining simulation accuracy and efficiency. While optimizing the computational performance of CG codes is important, without a way to incorporate UA-level detail, CG efficiency has relatively less value. The goal is to develop an integrated approach for conducting simulations that exploit both CG efficiency and UA accuracy.

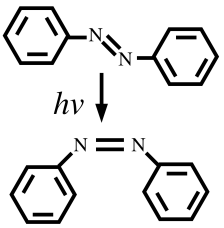
In homopolymer systems, transforming from the UA representation to the CG representation is straightforward: for each subchain having N_b monomers, the new soft sphere coordinate is simply the center of mass of the subchain. On the other hand, the *reverse* procedure of mapping from the CG representation to the UA representation is not generally well-defined. This transformation of a CG model into a UA model is a popular research topic, commonly referred to as the *backmapping* problem (See Fig. 23). For our homopolymer system, the backmapping problem is simply stated as follows: given a collection of CG soft sphere chains coordinates, insert monomer chains in such a way that we would recover the original CG configuration if we were to coarse-grain the system again.

It is easy to see that solutions to backmapping problems are not unique, because there are many different UA config-

urations that could map back to a given CG configuration. Much backmapping work focuses on biomolecules [133], but relatively little work has explored homopolymers. However, efficient backmapping procedures in polymer simulations are imperative for developing a full-fledged adaptively resolved simulations [134]. Furthermore, backmapping procedures contain clear opportunities for exploiting parallelism, which have yet to be explored.

6.3 QM/MM/CG

The availability of QM/MM and MM/CG methods begs the question of whether it is possible and informative to study QM/MM/CG models. It is easy to conjure such a triple-scale workflow that may be of interest. Consider the chemical system shown in Fig. 24, which describes the backmapping loop of a collection of liquid crystals containing azobenzene compounds, which may have potential applications in photo-switching [135]. Upon illumination, the azobenzene molecule can isomerize from a *trans* to a *cis* state:



then possibly decays back to the *trans* state, depending on the immediate neighborhood of other atoms. The first step shown in Fig. 24 involves constructing a CG simulation that represents a collection of azobenzene molecules. Then, similar to the fast equilibration work by Ozog et al. [130] mentioned in Section 6.2, we simulate the CG model long enough to reach equilibration. After equilibrating, a small region is chosen to be modeled with QM, and the atomic coordinates are backmapped using techniques described in Section 6.2.2. A QM simulation determines whether the isomerization takes place, then is equilibrated using an MM model, then reinserted into the larger CG domain. This entire process is then repeated if necessary.

The workflow described in the previous paragraph has no implementation, the paper only presents the scenario as a multiscale problem of interest [108]. However, other recent publications suggests that an implementation of a triple-scale model is possible and accurate [19, 20] using the ChemShell interactive environment [116]. In fact, this is the same work used in the visualization from Fig. 1. The capabilities of ChemShell are quite interesting, because they exploit the simplicity of most hybrid methods (such as described in Section 6.1) to support a sizeable collection of QM, MM, and CG supporting simulation tools. These currently include NWChem, GAMESS-UK, DALTON, Molpro, Gaussian, Q-Chem, DL_POLY, CHARMM, GROMOS, and several more. Fig. 25 shows ChemShell's software architecture, which enables modularity of the components at each scale. One disadvantage (to many) is that ChemShell relies on a custom TCL interactive shell. While this does enable the construction of intricate QM/MM/CG workflows, a more modern programming language such as Python may be a better choice because of its available scientific programming libraries, better extendibility, superior error and exception handling, and support for HPC computing.

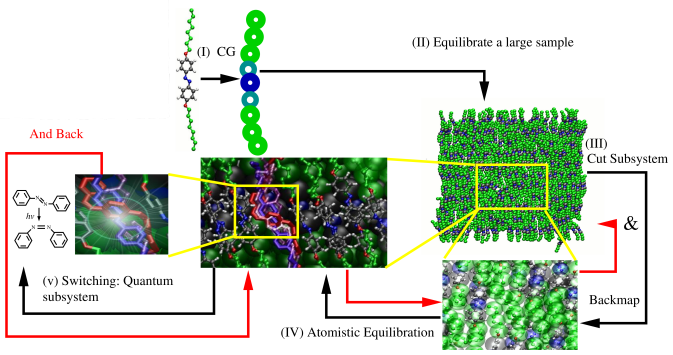


Fig. 24: An example of a triple-scale workflow. The diagram is from [108].

7 Relevant Applications

This section itemizes several example applications that utilize the concepts discussed in this paper. In my opinion, the most encouraging publication comes from the BASF, which is the largest chemical producer in the world [35]. This paper presents applications of quantum chemistry in industry, and highlights the importance of QM/MM methods in designing new materials (CG methods are mentioned as being imporant as well, but beyond the scope of the paper). In addition, the following publications are relevant to multiscale and multiresolution computational chemistry simulations:

- HF and DFT simulations of Lithium/air batteries using a new 3D FFT implementation [13]
- Divide-and-conquer quantum MD with DFT for hydrogen on-demand [34]
- Multiscale crack modeling (w/ LAMMPS and other tools) using the PERMIX computational library [82]
- Light-induced phase transitions in liquid crystal containing azobenzene photoswitch (QM/MM/CG) [131]
- Cloud environment for materials simulations [136]
- Screening quantum chemistry databases for organic photovoltaics [11]

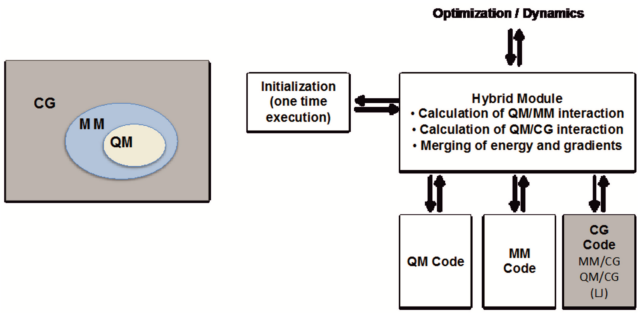


Fig. 25: (A) Schematic of the QM/MM/CG partitioning. (B) Example of the components of a QM/MM/CG workflow (from the ChemShell code). The schematic is from [19].

- Data mining to aid material design [12].

8 Conclusion and Future Directions

Computational chemistry is naturally a multiscale problem: interesting length and time scales span many orders of magnitude, and bridging these scales remains to be an important unsolved problem. Chemical models of vastly different scale require vastly different theories, software implementations, parallel decompositions, algorithms, and parallel programming methods. This paper discussed the challenges within the most prevalent scales of modern molecular simulations: QM, MM, and CG. Software implementations and their utilization of HPC differ greatly across these scale domains, and new research is required to effectively bridge these scales in the face of extreme challenges as the community pursues the exascale computing objective.

In our consideration of QM codes, we learned that QM algorithms often suffer from extremely diverse communication requirements that worsen as we add more processes to the application. More work needs to be done to quantify the potential benefit of optimizing for locality, and how best to reduce communication overhead at scale. Furthermore, Relatively little work has been done in QM on new computer architectures such as Intel's Knights Corner device. However, exploring the necessary vectorization optimizations is remarkably important in light of the large acquisition of copious Xeon Phi resources (Cori, Aurora, and Trinity), which will all contain Knights Landing and Knights Hill architectures.

In our sections highlighting the characteristics of MD codes, we learned that many software frameworks are relatively mature, but may benefit from an updated perspective on their load balance strategies, their tuning of important parameters such as the cutoff distance, and possibly utilizing data mining and machine learning techniques to alleviate the combinatorial explosion of simulation parameters and chemical system knowledge.

In the field of CG theory, we learned that there are several approaches to reducing the number of degrees of freedom from MD simulations. At the highest level, there are numerical and theoretical approaches. Broadly speaking, future work in numerical approaches should improve the accuracy of CG models while considering that introducing more parameters requires their tuning and sufficient training data. For theoretical methods, work remains to extend the models to more diverse chemical systems, such as block copolymers, and models that incorporate adaptive resolution.

In some sense, QM, MM, and CG codes have *individually* reached an appreciable state of maturity, but hybrid codes such as QM/MM/CG are relatively less supported, and the barrier of entry is too high for many computational chemists. Part of the reason for this is the novelty and diversity of many different multiresolution methods in chemistry. Much work remains to improve and eventually verify and validate these existing models, which requires supportive tools (such as VOTCA for CG) that allow for direct comparison of hybrid methods. The community would greatly benefit from more modular, extendible, and scalable software interfaces that enable exploration of different multiscale methods, and offer guidance in choosing the best fit for the chemistry problem at hand. Workflow systems management plays a crucial role in this regard, yet there is a general lack of support, adaptivity, performance metrics, and modularity in this field. The hardware infrastructure is available, we only require the committed development of new and extendible computational chemistry tools for multiscale simulations.

References

[1] D.P. Scarpazza, D.J. Ierardi, A.K. Lerer, K.M. Mackenzie, A.C. Pan, J.A. Bank, E. Chow, R.O. Dror, J.P. Grossman, D. Killebrew, M.A. Moraes, C. Predescu, J.K. Salmon, and D.E. Shaw. Extending the generality of molecular dynamics simulations on a special-purpose machine. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 933–945, May 2013.

[2] D.E. Shaw, J.P. Grossman, J.A. Bank, B. Batson, J.A. Butts, J.C. Chao, M.M. Deneroff, R.O. Dror, A. Even, C.H. Fenton, A. Forte, J. Gagliardo, G. Gill, B. Greskamp, C.R. Ho, D.J. Ierardi, L. Iserovich, J.S. Kuskin, R.H. Larson, T. Layman, Li-Siang Lee, A.K. Lerer, C. Li, D. Killebrew, K.M. Mackenzie, S.Y.-H. Mok, M.A. Moraes, R. Mueller, L.J. Nociolo, J.L. Peticolas, T. Quan, D. Ramot, J.K. Salmon, D.P. Scarpazza, U.B. Schafer, N. Siddique, C.W. Snyder, J. Spengler, P.T.P. Tang, M. Theobald, H. Toma, B. Towles, B. Vitale, S.C. Wang, and C. Young. Anton 2: Raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 41–53, Nov 2014.

[3] Ian T. Foster, Jeffrey L. Tilson, Albert F. Wagner, Ron L. Shepard, Robert J. Harrison, Rick A. Kendall, and Rik J. Littlefield. Toward high-performance computational chemistry: I. scalable fock matrix construction algorithms. *Journal of Computational Chemistry*, 17(1):109–123, 1996.

[4] Robert J. Harrison, Martyn F. Guest, Rick A. Kendall, David E. Bernholdt, Adrian T. Wong, Mark Stave, James L. Anchell, Anthony C. Hess, Rik J. Littlefield, George L. Fann, Jaroslaw Nieplocha, Greg S. Thomas, David Elwood, Jeffrey L. Tilson, Ron L. Shepard, Albert F. Wagner, Ian T. Foster, Ewing Lusk, and Rick Stevens. Toward high-performance computational chemistry: II. A scalable self-consistent field program. *Journal of Computational Chemistry*, 17(1):124–132, 1996.

[5] Tim Meyer, Marco D'Abramo, Adam Hospital, Manuel Rueda, Carles Ferrer-Costa, Alberto Prez, Oliver Carrillo, Jordi Camps, Carles Fenollosa, Dmitry Repchevsky, Josep Lluis Gelp, and Modesto Orozco. MoDEL (Molecular Dynamics Extended Library): ADatabase of Atomistic Molecular Dynamics Trajectories. *Structure*, 18(11):1399 – 1409, 2010.

[6] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies

with machine learning. *Phys. Rev. Lett.*, 108:058301, Jan 2012.

[7] Chi Chen, Dengjie Chen, and Francesco Ciucci. A molecular dynamics study of oxygen ion diffusion in a-site ordered perovskite PrBaCo2O5.5: data mining the oxygen trajectories. *Phys. Chem. Chem. Phys.*, 17:7831–7837, 2015.

[8] Stefano Curtarolo, Dane Morgan, Kristin Persson, John Rodgers, and Gerbrand Ceder. Predicting crystal structures with data mining of quantum calculations. *Phys. Rev. Lett.*, 91:135503, Sep 2003.

[9] J Knap, C E Spear, O Borodin, and K W Leiter. Advancing a distributed multi-scale computing framework for large-scale high-throughput discovery in materials science. *Nanotechnology*, 26(43):434004, 2015.

[10] Jianyin Shao et al. Clustering molecular dynamics trajectories: 1. characterizing the performance of different clustering algorithms. *Journal of Chemical Theory and Computation*, 3(6):2312–2334, 2007.

[11] Johannes Hachmann, Roberto Olivares-Amaya, Adrian Jinich, Anthony L. Appleton, Martin A. Blood-Forsythe, Laszlo R. Seress, Carolina Roman-Salgado, Kai Trepte, Sule Atahan-Evrenk, Suleyman Er, Supriya Shrestha, Rajib Mondal, Anatoliy Sokolov, Zhenan Bao, and Alan Aspuru-Guzik. Lead candidates for high-performance organic photovoltaics from high-throughput quantum chemistry - the Harvard Clean Energy Project. *Energy Environ. Sci.*, 7:698–704, 2014.

[12] Stefano Curtarolo, Gus LW Hart, Marco Buongiorno Nardelli, Natalio Mingo, Stefano Sanvito, and Ohad Levy. The high-throughput highway to computational materials design. *Nature materials*, 12(3):191–201, 2013.

[13] V. Weber, C. Bekas, T. Laino, A. Curioni, A. Bertsch, and S. Futral. Shedding Light on Lithium/Air Batteries Using Millions of Threads on the BG/Q Supercomputer. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 735–744, May 2014.

[14] Samyam Rajbhandari, Akshay Nikam, Pai-Wei Lai, Kevin Stock, Sriram Krishnamoorthy, and P. Sadayappan. A communication-optimal framework for contracting distributed tensors. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 375–386, Piscataway, NJ, USA, 2014. IEEE Press.

[15] A.G. Shet, W.R. Elwasif, R.J. Harrison, and D.E. Bernholdt. Programmability of the HPCS Languages: A case study with a quantum chemistry kernel. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[16] Carsten Hartmann and Luigi Delle Site. Scale bridging in molecular simulation. *The European Physical Journal Special Topics*, 224(12):2173–2176, 2015.

[17] Billion-atom Lennard-Jones benchmarks, 2015. http://lammps.sandia.gov/bench.html#billion.

[18] Nobel Media AB 2014. Web. *The Nobel Prize in Chemistry 2013 - Press Release*. 2015.

[19] Pandian Sokkar, Eliot Boulanger, Walter Thiel, and Elsa Sanchez-Garcia. Hybrid quantum mechanics/molecular mechanics/coarse grained modeling: A triple-resolution approach for biomolecular systems. *Journal of Chemical Theory and Computation*, 11(4):1809–1818, 2015.

[20] Soroosh Pezeshki and Hai Lin. Recent Progress in Adaptive-Partitioning QM/MM Methods for Born-Oppenheimer Molecular Dynamics. *Quantum Modeling of Complex Molecular Systems*, 21:93, 2015.

[21] Hans Johansen, Lois Curfman McInnes, David E. Bernholdt, Jeffrey Carver, Michael Heroux, Richard Hornung, Phil Jones, Bob Lucas, Andrew Siegel, and Thomas Ndousse-Fetter. Software productivity for extreme-scale science. 2014.

[22] Hans Johansen, David E Bernholdt, Bill Collins, Michael Heroux SNL, Robert Jacob ANL, Phil Jones, Lois Curfman McInnes ANL, and J David Moulton. Extreme-scale scientific application software productivity. 2013.

[23] Jack Dongarra et al. The international exascale software project roadmap. *International Journal of High Performance Computing Applications*, 2011.

[24] Saman Amarasinghe, Mary Hall, Richard Lethin, Keshav Pingali, Dan Quinlan, Vivek Sarkar, John Shalf, Robert Lucas, Katherine Yelick, Pavan Balaji, et al. Exascale programming challenges. In *Report of the 2011 Workshop on Exascale Programming Challenges, Marina del Rey*, 2011.

[25] Steve Ashby, P Beckman, J Chen, P Colella, B Collins, D Crawford, J Dongarra, D Kothe, R Lusk, P Messina, et al. The opportunities and challenges of exascale computing. *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pages 1–77, 2010.

[26] Linus Pauling and Edgar Bright Wilson. *Introduction to quantum mechanics: with applications to chemistry*. Courier Corporation, 1985.

[27] Michel Dupuis and Antonio Marquez. The Rys quadrature revisited: A novel formulation for the efficient computation of electron repulsion integrals over Gaussian functions. *The Journal of Chemical Physics*, 114(5):2067–2078, 2001.

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

[29] Frank Jensen. *Introduction to computational chemistry*. John Wiley & Sons, 2013.

[30] Christopher J Cramer. *Essentials of computational chemistry: theories and models*. John Wiley & Sons, 2013.

[31] Hongzhang Shan, Brian Austin, Wibe De Jong, Leonid Oliker, N.J. Wright, and Edoardo Apra. Performance Tuning of Fock Matrix and Two-Electron Integral Calculations for Nwchem on Leading HPC Platforms. In Stephen A. Jarvis, Steven A. Wright, and Simon D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, Lecture Notes in Computer Science, pages 261–280. Springer International Publishing, 2014.

[32] Attila Szabo and Neil S Ostlund. *Modern quantum chemistry: introduction to advanced electronic structure theory*. Courier Corporation, 2012.

[33] Isaiah Shavitt and Rodney J Bartlett. *Many-body methods in chemistry and physics: MBPT and coupled-cluster*

theory. Cambridge university press, 2009.

[34] Ken-ichi Nomura, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta, Kohei Shimamura, Fuyuki Shimojo, Manaschai Kunaseth, Paul C. Messina, and Nichols A. Romero. Metascalable quantum molecular dynamics simulations of hydrogen-on-demand. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 661–673, Piscataway, NJ, USA, 2014. IEEE Press.

[35] Peter Deglmann, Ansgar Schfer, and Christian Lennartz. Application of quantum calculations in the chemical industry - An overview. *International Journal of Quantum Chemistry*, 115(3):107–136, 2015.

[36] L. H. Thomas. The calculation of atomic fields. *Proceedings of the Cambridge Philosophical Society*, 23:542, 1927.

[37] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

[38] Wibe A. de Jong, Eric Bylaska, Niranjan Govind, Curtis L. Janssen, Karol Kowalski, Thomas Muller, Ida M. B. Nielsen, Hubertus J. J. van Dam, Valera Veryazov, and Roland Lindh. Utilizing high performance computing for chemistry: parallel computational chemistry. *Phys. Chem. Chem. Phys.*, 12:6896–6920, 2010.

[39] Andreas Bender, Angela Poschlad, Stefan Bozic, and Ivan Kondov. A service-oriented framework for integration of domain-specific data models in scientific workflows. *Procedia Computer Science*, 18:1087 – 1096, 2013. 2013 International Conference on Computational Science.

[40] Kenneth P Esler, Jeongnim Kim, David M Ceperley, Wirawan Purwanto, Eric J Walter, Henry Krakauer, Shiwei Zhang, Paul RC Kent, Richard G Hennig, Cyrus Umrigar, et al. Quantum Monte Carlo algorithms for electronic structure at the petascale; the endstation project. In *Journal of Physics: Conference Series*, volume 125, page 012057. IOP Publishing, 2008.

[41] Amos Anderson, William A. Goddard, III, and Peter Schrder. Quantum Monte Carlo on Graphical Processing Units, 2007.

[42] A. Vishnu, J. Daily, and B. Palmer. Designing Scalable PGAS Communication Subsystems on Cray Gemini Interconnect. In *High Performance Computing (HiPC), 2012 19th International Conference on*, pages 1–10, Dec 2012.

[43] D. Ozog, J.R. Hammond, J. Dinan, P. Balaji, S. Shende, and A. Malony. Inspector-executor load balancing algorithms for block-sparse tensor contractions. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 30–39, Oct 2013.

[44] Edmond Chow, Xing Liu, Sanchit Misra, Marat Dukhan, Mikhail Smelyanskiy, Jeff R. Hammond, Yunfei Du, Xiang-Ke Liao, and Pradeep Dubey. Scaling up HartreeFock calculations on Tianhe-2. *International Journal of High Performance Computing Applications*, 2015.

[45] Graham D. Fletcher, Michael W. Schmidt, Brett M. Bode, and Mark S. Gordon. The Distributed Data Interface in GAMESS. *Computer Physics Communications*, 128(12):190 – 200, 2000.

[46] Xing Liu, A. Patel, and E. Chow. A new scalable parallel algorithm for fock matrix construction. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 902–914, May 2014.

[47] Marat Valiev, Eric J Bylaska, Niranjan Govind, Karol Kowalski, Tjerk P Straatsma, Hubertus JJ Van Dam, Dunyou Wang, Jarek Nieplocha, Edoardo Apra, Theresa L Windus, et al. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9):1477–1489, 2010.

[48] Y. Alexeev, A. Mahajan, S. Leyffer, G. Fletcher, and D.G. Fedorov. Heuristic static load-balancing algorithm applied to the fragment molecular orbital method. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–13, Nov 2012.

[49] Pai-Wei Lai, K. Stock, S. Rajbhandari, S. Krishnamoorthy, and P. Sadayappan. A framework for load balancing of tensor contraction expressions via dynamic task partitioning. In *SC 2013*, pages 1–10, Nov 2013.

[50] Curtis L Janssen and Ida MB Nielsen. *Parallel computing in quantum chemistry*. CRC Press, 2008.

[51] Yili Zheng, A. Kamil, M.B. Driscoll, Hongzhang Shan, and K. Yelick. UPC++: A PGAS Extension for C++. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 1105–1114, May 2014.

[52] David Ozog, Amir Kamil, Yili Zheng, Paul Hargrove, Jeff R. Hammond, Allen Malony, Wibe de Jong, and Kathy Yelick. A hartree-fock application using upc++ and the new darray library. *(to appear)*, 2015.

[53] Adam H. R. Palser and David E. Manolopoulos. Canonical purification of the density matrix in electronic-structure theory. *Phys. Rev. B*, 58:12704–12711, Nov 1998.

[54] E. Solomonik, D. Matthews, J.R. Hammond, and J. Demmel. Cyclops tensor framework: Reducing communication and eliminating load imbalance in massively parallel contractions. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 813–824, May 2013.

[55] H. McCraw, A. Danalis, T. Herault, G. Bosilca, J. Dongarra, K. Kowalski, and T.L. Windus. Utilizing dataflow-based execution for coupled cluster methods. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, pages 296–297, Sept 2014.

[56] Pai-Wei Lai, Kevin Stock, Samyam Rajbhandari, Sriram Krishnamoorthy, and P. Sadayappan. A framework for load balancing of tensor contraction expressions via dynamic task partitioning. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 13:1–13:10, New York, NY, USA, 2013. ACM.

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

[58] Ivan S Ufimtsev and Todd J Martinez. Quantum chemistry on graphical processing units. 2. direct self-consistent-field implementation. *Journal of Chemical Theory and Computation*, 5(4):1004–1015, 2009.

[59] Ivan S Ufimtsev and Todd J Martinez. Quantum chemistry on graphical processing units. 3. analytical energy gradients, geometry optimization, and first principles molecular dynamics. *Journal of Chemical Theory and Computation*, 5(10):2619–2628, 2009.

[60] Andreas W Götz, Thorsten Wölfle, and Ross C Walker. Quantum chemistry on graphics processing units. *Annual Reports in Computational Chemistry*, 6:21–35, 2010.

[61] Mark Berger. *NVIDIA computational chemistry and biology*. November 2015.

[62] III A. Eugene DePrince and Jeff R. Hammond. Coupled cluster theory on graphics processing units i. the coupled cluster doubles method. *Journal of Chemical Theory and Computation*, 7(5):1287–1295, 2011. PMID: 26610123.

[63] Wenjing Ma, Sriram Krishnamoorthy, Oreste Villa, Karol Kowalski, and Gagan Agrawal. Optimizing tensor contraction expressions for hybrid cpu-gpu execution. *Cluster Computing*, 16(1):131–155, 2013.

[64] Edoardo Aprà, Michael Klemm, and Karol Kowalski. Efficient Implementation of Many-body Quantum Chemical Methods on the Intel®Xeon Phi®Coprocessor. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 674–684, Piscataway, NJ, USA, 2014. IEEE Press.

[65] Hongzhang Shan, Samuel Williams, Wibe de Jong, and Leonid Oliker. Thread-level Parallelization and Optimization of Nwchem for the Intel MIC Architecture. In *Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores*, PMAM '15, pages 58–67, New York, NY, USA, 2015. ACM.

[66] Alexey V. Titov, Ivan S. Ufimtsev, Nathan Luehr, and Todd J. Martinez. Generating efficient quantum chemistry codes for novel architectures. *Journal of Chemical Theory and Computation*, 9(1):213–221, 2013.

[67] John C. Linford, John Michalakes, Manish Vachharajani, and Adrian Sandu. Multi-core acceleration of chemical kinetics for simulation and prediction. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 7:1–7:11, New York, NY, USA, 2009. ACM.

[68] S. Herres-Pawlis, A. Hoffmann, R. Grunzke, W.E. Nagel, L. De La Garza, J. Kruger, G. Terstyansky, N. Weingarten, and S. Gesing. Meta-Metaworkflows for Combining Quantum Chemistry and Molecular Dynamics in the MoSGrid Science Gateway. In *Science Gateways (IWSG), 2014 6th International Workshop on*, pages 73–78, June 2014.

[69] S. Herres-Pawlis, A. Hoffmann, L. De La Garza, J. Kruger, S. Gesing, and R. Grunzke. User-friendly metaworkflows in quantum chemistry. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–3, Sept 2013.

[70] Sonja Herres-Pawlis, Alexander Hoffmann, kos Balask, Peter Kacsuk, Georg Birkenheuer, Andr Brinkmann, Luis de la Garza, Jens Krger, Sandra Gesing, Richard Grunzke, Gabor Terstyansky, and Noam Weingarten. Quantum chemical metaworkflows in mosgrid. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2014.

[71] James C Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D Skeel, Laxmikant Kale, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of computational chemistry*, 26(16):1781–1802, 2005.

[72] Knordlun. *Molecular dynamics algorithm*. 2015.

[73] Mike P Allen and Dominic J Tildesley. *Computer simulation of liquids*. Oxford university press, 1989.

[74] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Academic press, 2001.

[75] Tom M Mitchell. *Machine learning. WCB*. McGraw-Hill Boston, MA:, 1997.

[76] David A Pearlman, David A Case, James W Caldwell, Wilson S Ross, Thomas E Cheatham, Steve DeBolt, David Ferguson, George Seibel, and Peter Kollman. AMBER: A package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Computer Physics Communications*, 91(1):1–41, 1995.

[77] Bernard R Brooks, Charles L Brooks, Alexander D MacKerell, Lennart Nilsson, Robert J Petrella, Benoît Roux, Youngdo Won, Georgios Archontis, Christian Bartels, Stefan Boresch, et al. CHARMM: the biomolecular simulation program. *Journal of computational chemistry*, 30(10):1545–1614, 2009.

[78] Simpatico: Simulation package for polymer and molecular liquids (2015), 2015. http://research.cems.umn.edu/morse/code/simpatico/home.php.

[79] Jonathan D. Halverson, Thomas Brandes, Olaf Lenz, Axel Arnold, Sta Bevc, Vitaliy Starchenko, Kurt Kremer, Torsten Stuehn, and Dirk Reith. Espresso++: A modern multiscale simulation package for soft matter systems. *Computer Physics Communications*, 184(4):1129 – 1149, 2013.

[80] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995.

[81] Simon J. Pennycook, Chris J. Hughes, M. Smelyanskiy, and S. A. Jarvis. Exploring SIMD for Molecular Dynamics, Using Intel®Xeon®Processors and Intel®Xeon Phi®Coprocessors. In *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, IPDPS '13, pages 1085–1097, Washington, DC, USA, 2013. IEEE Computer Society.

[82] Hossein Talebi, Mohammad Silani, Stphane P.A. Bordas, Pierre Kerfriden, and Timon Rabczuk. A computational library for multiscale modeling of material failure. *Computational Mechanics*, 53(5):1047–1071, 2014.

[83] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilrd Pll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 12:19 – 25, 2015.

[84] Sander Pronk, Szilrd Pll, Roland Schulz, Per Larsson, Pr Bjelkmar, Rossen Apostolov, Michael R. Shirts,

Jeremy C. Smith, Peter M. Kasson, David van der Spoel, Berk Hess, and Erik Lindahl. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 2013.

[85] L.V. Kal, M. Bhandarkar, R. Brunner, N. Krawetz, J. Phillips, and A. Shinozaki. NAMD: A case study in multilingual parallel programming. In Zhiyuan Li, Pen-Chung Yew, Siddharta Chatterjee, Chua-Huang Huang, P. Sadayappan, and David Sehr, editors, *Languages and Compilers for Parallel Computing*, volume 1366 of *Lecture Notes in Computer Science*, pages 367–381. Springer Berlin Heidelberg, 1998.

[86] J Villareal, John Cortes, and W Najjar. Compiled code acceleration of NAMD on FPGAs. *Proceedings of the Reconfigurable Systems Summer Institute*, 2007.

[87] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124 – 1139, 2011. New Computational Methods and Software Tools.

[88] Frank S Bates. Block copolymers near the microphase separation transition. 2. linear dynamic mechanical properties. *Macromolecules*, 17(12):2607–2613, 1984.

[89] Sanat K. Kumar, Nicolas Jouault, Brian Benicewicz, and Tony Neely. Nanocomposites with Polymer Grafted Nanoparticles. *Macromolecules*, 46(9):3199–3214, 2013.

[90] Anton Smessaert and Jörg Rottler. Recovery of polymer glasses from mechanical perturbation. *Macromolecules*, 45(6):2928–2935, 2012.

[91] J. Baschnagel, K. Binder, P. Doruker, A. Gusev, O. Hahn, K. Kremer, W. Mattice, F. Müller-Plathe, M. Murat, W. Paul, S. Santos, U. Suter, and V. Tries. Bridging the Gap Between Atomistic and Coarse-Grained Models of Polymers: Status and Perspectives. In *Viscoelasticity, Atomistic Models, Statistical Chemistry*, volume 152 of *Advances in Polymer Science*, pages 41–156. 2000.

[92] Victor Rhle, Christoph Junghans, Alexander Lukyanov, Kurt Kremer, and Denis Andrienko. Versatile object-oriented toolkit for coarse-graining applications. *Journal of Chemical Theory and Computation*, 5(12):3211–3223, 2009.

[93] Timothy C. Moore, Christopher R. Iacovella, and Clare McCabe. Derivation of coarse-grained potentials via multistate iterative boltzmann inversion. *The Journal of Chemical Physics*, 140(22), 2014.

[94] Andrew Peacock. *Handbook of Polyethylene: Structures, Properties, and Applications*. CRC Press, 2000.

[95] David Ozog, Allen Malony, and Marina Guenza. The UA/CG workflow: High performance molecular dynamics of coarse-grained polymers. *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016.

[96] E. J. Sambriski and M. G. Guenza. Theoretical coarse-graining approach to bridge length scales in diblock copolymer liquids. *Phys. Rev. E*, 76:051801, Nov 2007.

[97] A. J. Clark and M. G. Guenza. Mapping of Polymer Melts onto Liquids of Soft-Colloidal Chains. *The Journal of Chemical Physics*, 132(4):–, 2010.

[98] G. Yatsenko, E. J. Sambriski, M. A. Nemirovskaya, and M. Guenza. Analytical soft-core potentials for macromolecular fluids and mixtures. *Phys. Rev. Lett.*, 93:257803, Dec 2004.

[99] I. Y. Lyubimov and M. G. Guenza. Theoretical Reconstruction of Realistic Dynamics of Highly Coarse-Grained cis-1,4-polybutadiene Melts. *The Journal of Chemical Physics*, 138(12):120000, March 2013.

[100] I. Lyubimov and M. G. Guenza. First-principle approach to rescale the dynamics of simulated coarse-grained macromolecular liquids. *Phys. Rev. E*, 84:031801, Sep 2011.

[101] J. McCarty and M. Guenza. Multiscale modeling of binary polymer mixtures: Scale bridging in the athermal and thermal regime. *The Journal of Chemical Physics*, 133(9):094904, 2010.

[102] A. J. Clark, J. McCarty, and M. G. Guenza. Effective Potentials for Representing Polymers in Melts as Chains of Interacting Soft Particles. *The Journal of Chemical Physics*, 139(12):–, 2013.

[103] J. McCarty, A. J. Clark, J. Copperman, and M. G. Guenza. An analytical coarse-graining method which preserves the free energy, structural correlations, and thermodynamic state of polymer melts from the atomistic to the mesoscale. *The Journal of Chemical Physics*, 140(20):–, 2014.

[104] A. Clark, J. McCarty, I. Lyubimov, and M. Guenza. Thermodynamic Consistency in Variable-Level Coarse Graining of Polymeric Liquids. *Phys. Rev. Lett.*, 109:168301, Oct 2012.

[105] James F. Dama, Anton V. Sinitskiy, Martin McCullagh, Jonathan Weare, Benot Roux, Aaron R. Dinner, and Gregory A. Voth. The theory of ultra-coarse-graining. 1. general principles. *Journal of Chemical Theory and Computation*, 9(5):2466–2480, 2013.

[106] Dirk Reith, Mathias Pütz, and Florian Müller-Plathe. Deriving effective mesoscale potentials from atomistic simulations. *Journal of Computational Chemistry*, 24(13):1624–1636, 2003.

[107] Siewert J. Marrink and D. Peter Tieleman. Perspective on the martini model. *Chem. Soc. Rev.*, 42:6801–6822, 2013.

[108] Luigi Delle Site. What is a multiscale problem in molecular dynamics? *Entropy*, 16(1):23, 2013.

[109] M.F. Horstemeyer. Multiscale modeling: A review. In Jerzy Leszczynski and Manoj K. Shukla, editors, *Practical Aspects of Computational Chemistry*, pages 87–135. Springer Netherlands, 2010.

[110] S.A. Baeurle. Multiscale modeling of polymer materials using field-theoretic methodologies: a survey about recent developments. *Journal of Mathematical Chemistry*, 46(2):363–426, 2009.

[111] Stefan Dapprich, Istvn Komromi, K.Suzie Byun, Keiji Morokuma, and Michael J Frisch. A new ONIOM implementation in Gaussian98. Part i. The calculation of energies, gradients, vibrational frequencies and electric field derivatives1. *Journal of Molecular Structure: Theoretical Chemistry*, 461462:1 – 21, 1999.

[112] Hai Lin and Donald G. Truhlar. QM/MM: what have we learned, where are we, and where do we go from here? *Theoretical Chemistry Accounts*, 117(2):185–199, 2007.

[113] Nandun M. Thellamurege, Dejun Si, Fengchao Cui, Hongbo Zhu, Rui Lai, and Hui Li. Quanpol: A full spectrum and seamless QM/MM program. *Journal of Computational Chemistry*, 34(32):2816–2833, 2013.

[114] Gerrit Groenhof. Introduction to QM/MM simulations. In Luca Monticelli and Emppu Salonen, editors, *Biomolecular Simulations*, volume 924 of *Methods in Molecular Biology*, pages 43–66. Humana Press, 2013.

[115] Arieh Warshel and Michael Levitt. Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *Journal of molecular biology*, 103(2):227–249, 1976.

[116] A ChemShell. *Computational Chemistry Shell*. 2013.

[117] Michael Levitt and Arieh Warshel. Computer simulation of protein folding. *Nature*, 253(5494):694–698, 1975.

[118] Matej Praprotnik, Kurt Kremer, and Luigi Delle Site. Adaptive molecular resolution via a continuous change of the phase space dimensionality. *Phys. Rev. E*, 75:017701, Jan 2007.

[119] Matej Praprotnik, Luigi Delle Site, and Kurt Kremer. Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly. *The Journal of Chemical Physics*, 123(22):–, 2005.

[120] Matej Praprotnik, Luigi Delle Site, and Kurt Kremer. Multiscale simulation of soft matter: From scale bridging to adaptive resolution. *Annual Review of Physical Chemistry*, 59(1):545–571, 2008. PMID: 18062769.

[121] Julija Zavadlav, Manuel N. Melo, Ana V. Cunha, Alex H. de Vries, Siewert J. Marrink, and Matej Praprotnik. Adaptive resolution simulation of MARTINI solvents. *Journal of Chemical Theory and Computation*, 10(6):2591–2598, 2014.

[122] Julija Zavadlav, Manuel Nuno Melo, Siewert J. Marrink, and Matej Praprotnik. Adaptive resolution simulation of an atomistic protein in MARTINI water. *The Journal of Chemical Physics*, 140(5):–, 2014.

[123] L. Delle Site, A. Agarwal, C. Junghans, and H. Wang. Adaptive Resolution Simulation as a Grand Canonical Molecular Dynamics Scheme: Principles, Applications and Perspectives. *ArXiv e-prints*, December 2014.

[124] Alexander B. Kuhn, Srinivasa M. Gopal, and Lars V. Schfer. On using atomistic solvent layers in hybrid all-atom/coarse-grained molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 11(9):4460–4472, 2015.

[125] Sebastian Fritsch, Christoph Junghans, and Kurt Kremer. Structure Formation of Toluene around C60: Implementation of the Adaptive Resolution Scheme (AdResS) into GROMACS. *Journal of Chemical Theory and Computation*, 8(2):398–403, 2012.

[126] Cameron Abrams, Luigi Delle Site, and Kurt Kremer. Multiscale computer simulations for polymeric materials in bulk and near surfaces. In Peter Nielaba, Michel Mareschal, and Giovanni Ciccotti, editors, *Bridging Time Scales: Molecular Simulations for the Next Decade*, volume 605 of *Lecture Notes in Physics*, pages 143–164. Springer Berlin Heidelberg, 2002.

[127] H. Wang and A. Agarwal. Adaptive resolution simulation in equilibrium and beyond. *The European Physical Journal Special Topics*, 224(12):2269–2287, 2015.

[128] S. Fritsch, S. Poblete, C. Junghans, G. Ciccotti, L. Delle Site, and K. Kremer. Adaptive resolution molecular dynamics simulation through coupling to an internal particle reservoir. *Phys. Rev. Lett.*, 108:170602, Apr 2012.

[129] Animesh Agarwal and Luigi Delle Site. Path integral molecular dynamics within the grand canonical-like adaptive resolution technique: Simulation of liquid water. *The Journal of Chemical Physics*, 143(9):–, 2015.

[130] David Ozog, Jay McCarty, Grant Gossett, Allen D. Malony, and Marina Guenza. Fast equilibration of coarse-grained polymeric liquids. *Journal of Computational Science*, 9:33 – 38, 2015. Computational Science at the Gates of Nature.

[131] Marcus Bockmann, Dominik Marx, Christine Peter, Luigi Delle Site, Kurt Kremer, and Nikos L. Doltsinis. Multiscale modelling of mesoscopic phenomena triggered by quantum events: light-driven azo-materials and beyond. *Phys. Chem. Chem. Phys.*, 13:7604–7621, 2011.

[132] Simn Poblete, Matej Praprotnik, Kurt Kremer, and Luigi Delle Site. Coupling different levels of resolution in molecular simulations. *The Journal of Chemical Physics*, 132(11), 2010.

[133] Tsjerk A. Wassenaar, Kristyna Pluhackova, Rainer A. Böckmann, Siewert J. Marrink, and D. Peter Tieleman. Going Backward: A Flexible Geometric Approach to Reverse Transformation from Coarse Grained to Atomistic Models. *Journal of Chemical Theory and Computation*, 10(2):676–690, 2014.

[134] Christine Peter and Kurt Kremer. Multiscale simulation of soft matter systems - from the atomistic to the coarse-grained level and back. *Soft Matter*, 5:4357–4366, 2009.

[135] Zahid Mahimwalla, KevinG. Yager, Jun-ichi Mamiya, Atsushi Shishido, Arri Priimagi, and ChristopherJ. Barrett. Azobenzene photomechanics: prospects and potential applications. *Polymer Bulletin*, 69(8):967–1006, 2012.

[136] K. Jorissen, F.D. Vila, and J.J. Rehr. A high performance scientific cloud computing environment for materials simulations. *Computer Physics Communications*, 183(9):1911 – 1919, 2012.