SAND2011-2599C

Hybrid algorithms in quantum Monte Carlo

Jeongnim Kim
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL, US
jnkim@illinois.edu

Kenneth P. Esler[*]
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL, US
esler@uiuc.edu

Jeremy McMinis
Department of Physics
University of Illinois at Urbana-Champaign
Urbana, IL, US
mcminis2@illinois.edu

Miguel A. Morales
Lawrence Livermore National Laboratory
Livermore, CA, US
moralessilva2@llnl.gov

Bryan K Clark
The Princeton Center for Theoretical Science
Princeton University
Princeton, NJ, US
bclark@princeton.edu

Luke Shulenburger
Sandia National Laboratories
Albuquerque, NM, US
lshulen@sandia.gov

ABSTRACT

With advances in algorithms and growing computing powers, quantum Monte Carlo (QMC) methods have become a leading contender for high accuracy calculations for the electronic structure of realistic systems. The performance gain on recent HPC systems is largely driven by increasing parallelism: the number of compute cores of a SMP has been going up as well as the number of SMPs, as the Top500 list attests. However, the available memory and memory and communication bandwidth per element has not kept pace with the increasing parallelism. This severely limits the applicability of QMC and the problem size it can handle. OpenMP/MPI hybrid programming provides applications with simple but effective solutions to overcome efficiency and scalability bottlenecks on large-scale clusters based on multi/many-core SMPs. We discuss the design and implementation of hybrid methods in QMCPACK and analyze its performance on current HPC platforms characterized by various memory and communication hierarchies.

Categories and Subject Descriptors

J.2 [**Physical Sciences and Engineering**]: Physics; D.1.3 [**Programming Techniques**]: Concurrent Programming; D.2.11 [**Software Architectures**]: Domain-specific architectures, Patterns

General Terms

Application

[*]Current address: Stone Ridge Technology, Bel Air, MD

Keywords

applications, computational physics, materials

1. INTRODUCTION

Continuum quantum Monte Carlo (QMC) methods employ explicitly correlated wave functions to describe the many-body effects and symmetries in an efficient and compact manner and solve the Schrödinger equation by a stochastic process [1]. QMC is one of the most accurate *ab initio* methods to describe the behavior and properties of quantum many-particle systems. It is general and applicable to a wide range of physical and chemical systems in any dimension, boundary conditions etc. The favorable scaling, 3-4 power of the problem size, and ample opportunities for parallelizations, e.g., over multiple Bloch **k**-vectors, make QMC methods uniquely powerful tools to study the electronic structure of realistic systems on large-scale parallel computers.

The objects in QMC algorithms can be divided into two categories: i) *ensemble data* to represent the physical entities like trial wave functions and ii) *walker data* to represent each $3N$ dimensional configuration and its dynamic state. The *ensemble data* are large but are unmodified during a simulation once the initialization step is completed and are shared by all the walkers. On the other hand, the *walker data* evolve as we sample the configurations using various QMC algorithms. The simplest parallelization strategy is distributing walkers among the tasks, while replicating the *ensemble data* on each task. This fits nicely to the distributed memory model of the Message Passing Interface (MPI) library [2].

Until recently, QMC methods have been able to exploit the steady increase in clock rates and the enhancement of performance enabled by advances in CMOS VLSI technology for faster and bigger simulations without any major changes in this parallel algorithm based on MPI. However, this decade has seen some significant changes in the mode of increase of computational power. The performance gain has been largely driven by increasing parallelism in a shared-memory processor (SMP) unit in the form of multi-core processors and GPUs (Graphics Processing Units). A notable feature of multi/many-core architectures is relatively constant memory per core of a few GB. This can limit the problem size QMC can handle, if the *ensemble data* is replicated on each core for efficiency. Recognizing that the aggregated memory of each SMP node is steadily growing with the number of cores on a processor, one can easily overcome the memory limitation of multi-core processors by using shared-memory programming models such as OpenMP [3].

In this article, we present hybrid QMC algorithms using OpenMP/MPI. Similar to the standard MPI implementation, we replicate the large but *read-only* ensemble data on each MPI task and distribute the walkers among tasks. OpenMP parallel constructs are used to further divide the walkers among threads. The threads and objects are carefully managed to minimize OpenMP overhead and false sharing, while maximizing data locality and cache reuse. The computational cost per core and the overall parallel efficiency are unaffected by the additional parallelization with OpenMP at a modest core count of a few 1000s. As we increase the number of nodes, the hybrid methods become more effective because of the unfavorable scaling of MPI collectives with respect to the task count and much reduced communication needs for the load balance. We expect that the performance advantage of the hybrid methods will expand with increasing parallelism on newer processors.

The hybrid QMC algorithms are implemented in QMCPACK, an open-source QMC package designed for large-scale parallel computers [4]. It makes extensive use of object-oriented and generic programming and design patterns [5] for reusability and extensibility. High computational efficiency is achieved through inlined specializations of C++ templates and by using SIMD intrinsics for core kernels. It utilizes standard file formats for input and output, using XML standard and HDF5 [6], to streamline the QMC workflow and facilitate data exchange with other applications.

The rest of the paper is organized as follows. First, we introduce QMC methods and core algorithms. We analyze critical computational components and scaling properties with problems size. In section 4, we present hybrid algorithms using OpenMP/MPI at the walker level and detail our efforts to achieve high computational and parallel efficiency on large-scale clusters. Our on-going works to overcome the limitations of current implementations are discussed in section 5 and the conclusions follow.

2. QMC METHODS

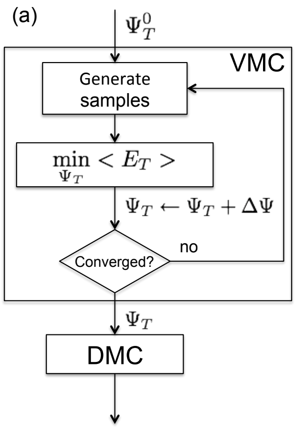
In quantum mechanics, all physically observable quantities for a system containing N particles can be computed from the $3N$-dimensional *wave function*, $\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$. The exact wave function for the system satisfies the time-independent Schrödinger equation,

$$\hat{H}\Psi = E_0\Psi, \tag{1}$$

where E_0 is the ground-state energy for the system and $\hat{H}$ is the Hamiltonian, or total energy operator. For example, the Hamiltonian for an N-electron system is given as

$$\hat{H} = \sum_{i=1}^{N} -\frac{1}{2}\nabla_i^2 + \sum_{i=1}^{N} V_{\text{ext}}(\mathbf{r}_i) + \sum_{i<j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{2}$$

where $V_{\text{ext}}(\mathbf{r}_i)$ is the *external* potential generated by the nuclei, applied electric fields, etc. Since the exact solution Ψ is known only for simple few-body systems, QMC methods employ a *trial* wave function Ψ_T, which takes the general form



(b) DMC
```
for generation = $1 \cdots M$ do
  for walker = $1 \cdots N_w$ do
    let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
    for particle $i = 1 \cdots N$ do
      set $\mathbf{r}_i' = \mathbf{r}_i + \delta$
      let $\mathbf{R}' = \{\mathbf{r}_1 \ldots \mathbf{r}_i' \ldots \mathbf{r}_N\}$
      ratio $\rho = \Psi_T(\mathbf{R}')/\Psi_T(\mathbf{R})$
      if $\mathbf{r} \to \mathbf{r}'$ is accepted then
        update state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update $E_T$
  end for{walker}
end for{generation}
```

Figure 1: (a) QMC workflow and (b) DMC algorithm.

$$\Psi_T(\mathbf{R}) = \prod_k \psi_k(\{\alpha\}, \mathbf{R}). \tag{3}$$

Here, each component ψ_k is given as an analytic function of a number of optimizable parameters $\{\alpha\}$.

For any $\Psi_T(\mathbf{R})$, we can compute an energy as the expected value of $\hat{H}$,

$$E_T = \frac{\int d^{3N}\mathbf{R}\ \Psi_T^*(\mathbf{R})\hat{H}\Psi_T(\mathbf{R})}{\int d^{3N}\mathbf{R}\ |\Psi_T(\mathbf{R})|^2}, \tag{4}$$

where $\mathbf{R}$ is a $3N$-dimensional vector representing the positions of the N particles. A *variational principle* guarantees that $E_T \geq E_0$ for all Ψ_T, which implies that the lower the energy a given Ψ_T has, the closer it is to the exact solution of the Schrödinger equation. In QMC, E_T is estimated by stochastic sampling as

$$E_T \approx \frac{\sum_i^M w(\mathbf{R}_i)E_L(\mathbf{R}_i)}{\sum_i^M w(\mathbf{R}_i)}, \tag{5}$$

where E_L is a local energy, $E_L(\mathbf{R}) = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$ and w weight at $\mathbf{R}$.

Various algorithms have been developed to find the ground-state solution, i) either by minimizing the energy by varying parameters in a Ψ_T as in the variational Monte Carlo (VMC) method or ii) by projecting out the ground state by repeatedly applying an imaginary time Green's function or *propagator* $\exp(-\tau\hat{H})$ on Ψ_T as in the diffusion Monte Carlo (DMC) method [1]. In actual calculations, the parameters $\{\alpha\}$ are optimized within VMC to obtain improved, yet still compact wave functions. Once a trial wave function is optimized, it is further improved through a stochastic projection by DMC. This typical QMC workflow is illustrated in Fig. 1(a).

For stochastic sampling methods, the convergence of a simulation is determined by the standard error as

$$\delta = \frac{\sigma}{\sqrt{M}}, \tag{6}$$

for M *uncorrelated* samples. In VMC, all the configurations

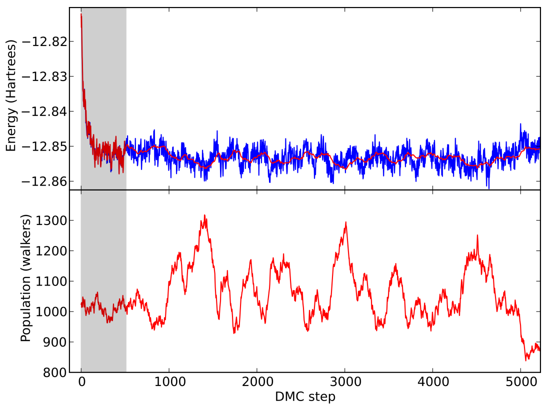


Figure 2: Trace of the DMC energy and walker population during a simulation. The gray shaded region indicates the equilibration period, during which the data should be discarded.

are equally weighted, *i.e.*, $w = 1$, and the energy variance is

$$\begin{aligned}\sigma^2 &= \frac{\sum_i^M E_L(\mathbf{R}_i)^2}{M} - \left(\frac{\sum_i^M E_L(\mathbf{R}_i)}{M}\right)^2 \\ &= < E_T^2 > - < E_T >^2 . \end{aligned} \quad (7)$$

The generalization for $w = \exp^{-\tau(E_L - E_T)}$ in DMC is straightforward [1]. Finally, we can define the efficiency of a QMC simulation as

$$\kappa = \frac{1}{\sigma^2 \tau_{corr} T_{CPU}}, \quad (8)$$

where T_{CPU} is the time to achieve a target error and T_{corr} is the auto-correlation time [7]. An accurate Ψ_T with small σ and τ_{corr} is essential to achieve high quality results with small statistical error. Choosing a compact representation for Ψ_T with a low computational cost is equally important for efficient QMC calculations. The advantage of increasing parallelism for QMC is apparent, as $T_{CPU} \propto T_{MC}/N_p$, where T_{MC} denotes the computational time to generate a MC sample and N_p, the number of parallel processing units.

3. QMC ALGORITHMS

The vast majority of the computational time in an accurate QMC simulation is spent in DMC and therefore minimizing the wall-clock time to obtain a DMC solution within a target error is desired. The optimized computational kernels, physical abstractions and parallelization schemes, however, are also applicable to all other QMC algorithms.

In the DMC algorithm (Fig. 1b), an ensemble of walkers (*population*) is propagated stochastically from generation to generation, where each walker is represented by $\mathbf{R}$. In each propagation step, the walkers are moved through position space by a *drift-diffusion process*. At the end of each step, each walker may reproduce itself, be killed, or remain unchanged (*branching process*), depending on the value of E_L, the local energy at that point in space. The walkers in an ensemble are tightly coupled through a trial energy E_T and a feedback mechanism to keep the average population at the target $10^3 - 10^5$ walkers. This leads to a fluctuating population (Fig. 2b) and necessitates shuffling of walkers among parallel processing units to maintain good load balance in

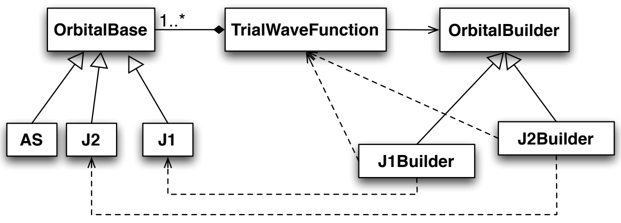


Figure 3: Composite pattern for TrialWaveFunction Ψ_T. It owns a number of `OrbitalBase` objects which are instantiated at runtime as defined in an input file using the builder pattern. The derived classes, denoted as J1, J2, and AS for Ψ_{AS}, implement the virtual functions defined in the `OrbitalBase` class.

an appropriate interval.

The direct evaluation of the many-dimensional integrals of Eq. (4) by stochastic sampling enables us to employ highly accurate variational wave functions which can capture crucial many-body effects in an efficient manner. The Slater-Jastrow trial wave functions, commonly used in QMC applications to electronic structure, are sums of Slater determinants of single-particle orbitals multiplied by a Jastrow factor:

$$\Psi_T = \exp(J_1)\exp(J_2)\cdots \underbrace{\sum C_i D_i^{\uparrow}(\phi) D_i^{\downarrow}(\phi)}_{\Psi_{AS}}, \quad (9)$$

with $N = N^{\uparrow} + N^{\downarrow}$ for $\uparrow\downarrow$ spins. Here, $\{\phi\}$ denote a set of single-particle orbitals and

$$D_i^{\uparrow} = \det|\mathcal{M}| = \det \begin{vmatrix} \phi_1(\mathbf{r}_1) & \cdots & \phi_1(\mathbf{r}_{N^\uparrow}) \\ \vdots & \vdots & \vdots \\ \phi_{N^\uparrow}(\mathbf{r}_1) & \ldots & \phi_{N^\uparrow}(\mathbf{r}_{N^\uparrow}) \end{vmatrix}, \quad (10)$$

which ensures the antisymmetric property of a Fermionic wave function upon a pair exchange of electrons. The Jastrow factors, which are factorized for computational efficiency, describe the dynamic correlation, whereas static correlation is described by the sum of Slater determinants. The optimization of the many-body trial wave functions is a crucial ingredient for accurate QMC calculations, since accurate trial wave functions reduce statistical and systematic errors. Several new methods for optimization have been developed recently [8, 9]. These new methods enable us to optimize the determinantal coefficients $\{C\}$ in addition to the Jastrow parameters, reducing the dependence of the results on the input trial wave function.

The product form of a trial wave function (Eqs. 3 and 9) is rewritten as

$$\ln \Psi_T(\mathbf{R}) = \sum_k \ln \psi_k(\{\alpha\}, \mathbf{R}), \quad (11)$$

for numerical stability of large-scale QMC calculations. The use of the log representation eliminates the need to normalize the trial wave function, which has to be computed with other properties, and to prevent overflow or underflow in using ill-conditioned values, especially the determinants of large matrices. Each component can be defined in a self-contained class derived from an abstract base class. We express the computations associated with Ψ_T as sums of wave function components, e.g.,

$$\frac{\nabla\Psi_T}{\Psi_T} = \nabla \ln \Psi_T = \sum_k \nabla \ln \psi_k, \tag{12}$$

which is needed for E_L. The actual form of the trial wave function is not known at compile time. We apply the builder pattern to construct a trial wave function component by component according to the input parameters as illustrated in Fig. 3 [5]. The many-body Hamiltonian in Eq. (2) follows the same patterns employed for the trial wave function.

The dominant computation for large systems is set by the antisymmetric part Ψ_{AS} of the trial wave function: i) to evaluate the ratios, T_{ratio}; ii) to update Slater determinants, $T_{\mathcal{M}^{-1}}$; and iii) to fill in the Slater determinants with single-particle orbitals, T_{spo}. Since the particle-by-particle moves we employ change only one column of the $\mathcal{M}$ matrices at a time, we employ a rank-1 update of $\mathcal{M}^{-1}$ using the Sherman-Morrison formula. This allows the inverse to be updated in $\mathcal{O}(N^2)$ time rather than $\mathcal{O}(N^3)$ time for a full inversion and the ratios in $\mathcal{O}(N)$.

The single-particle orbitals are a set of three-dimensional functions defined on the simulation domain. They are typically expressed in a basis set. It is possible to optimize $\{\phi\}$ in a given basis set, but, in practice, we use the solution of an approximate method such as Hartree-Fock or density functional theory (DFT). The most commonly used basis sets are plane-wave basis set for solids in periodic boundary conditions and atomic orbitals for molecular systems. The cost to compute the value of a ϕ scales linearly with the number of basis function evaluations which tends to grow with the system size. This amounts to $\mathcal{O}(N^2)$ cost for each particle move and T_{spo} becomes the bottleneck at large N with these basis sets.

For this reason, it is more efficient to use a localized basis with compact support. In particular, 3D tricubic B-splines provide a basis in which only 64 elements are nonzero at any given point in space [10, 11]. The one-dimensional cubic B-spline is given by,

$$f(x) = \sum_{i'=i-1}^{i+2} b^{i',3}(x)\ p_{i'}, \tag{13}$$

where $b^i(x)$ are the piecewise cubic polynomial basis functions shown in Figure 4, and $i = \text{floor}(\Delta^{-1}x)$ is the index of the first grid point $\leq x$. Constructing a tensor product in each Cartesian direction, we can represent a 3D orbital as

$$\phi_n(x,y,z) = \sum_{i'=i-1}^{i+2} b_x^{i',3}(x) \sum_{j'=j-1}^{j+2} b_y^{j',3}(y) \sum_{k'=k-1}^{k+2} b_z^{k',3}(z)\ p_{i',j',k',n}. \tag{14}$$

This allows the rapid evaluation of each orbital in constant time. Furthermore, they are systematically improvable with a single spacing parameter, so that accuracy is not compromised.

The use of 3D tricubic B-splines greatly improves the computational efficiency. Even for a modest problem size of 32 electrons, the speed up of B-spline interpolations is more than sixfold over an equivalent PW basis set. The gain in computation time of real-space methods becomes increasingly large as the system size grows. On the downside, this computational efficiency comes at the expense of increased memory use.

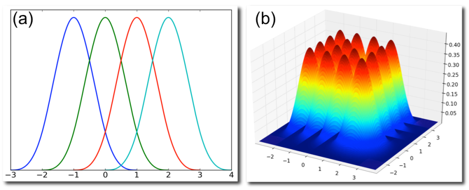


Figure 4: Cubic B-spline basis functions in (a) 1 D and (b) 2 D.

4. HYBRID QMC IMPLEMENTATION

Among the many ways to parallelize the DMC algorithm, the multiplicity of the walkers in an ensemble provides the most natural units for data and task parallelizations. The internal state of each walker, its configuration $\mathbf{R}$ and other data to reduce recomputation, is encapsulated in the `Walker` class. The operations to propagate each walker during the drift-diffusion process are expressed as a parallel loop over the `Walkers`. Once a generation has evolved, the properties of all the walkers in an ensemble are collected to determine E_T and N_w, the number of walkers of the next generation, which employs global reduction operations among MPI tasks. The redistribution of `Walkers` during the load-balance step can be done efficiently by exchanging a serialized `Walker` object as a large message between paired MPI tasks.

The communication overhead in QMC calculations is quite negligible compared to T_{MC}, the computation time. For the reduction operations to perform averages and record results, MPI collectives are found to be the most efficient choice on many platforms. The message size and number of messages for the collectives are independent of N_p, the number of tasks, and the problem size. But, the communication time for the collectives can become a sizable fraction of the total run time and decrease the parallel efficiency at large N_p, as the collectives scale as $\mathcal{O}(\log_2 N_p) - \mathcal{O}(N_p^2)$ depending on the network topology and specific algorithms in use.

There are several different and independent ways to parallelize a QMC calculation beyond the walker level, e.g., over parameter and configuration spaces and over Bloch $\mathbf{k}$-vectors. Such high-level parallelism can be easily managed by mapping a DMC ensemble on a MPI group.

At present, power and thermal considerations constrain the growth in computational power to come almost exclusively through parallelism in the form of multi-core processors. As we scale out the problem size and use finer grids to achieve higher accuracy, the memory required to store the trial wave function including a large table for B-spline interpolations will become larger than that available on a single core, necessitating the distribution of a single trial wave function data over the cores and adding communications to access the data on remote MPI tasks. This limitation, however, can be easily overcome using shared memory programming models that provide direct access to the entire memory of a node rather than that of a core.

Among various parallel programming models, hybrid programming using OpenMP and MPI provides QMC methods with simple but very effective solutions. We can utilize the large shared-memory address space on a SMP node for the data that can be shared. The loop over the `walkers`

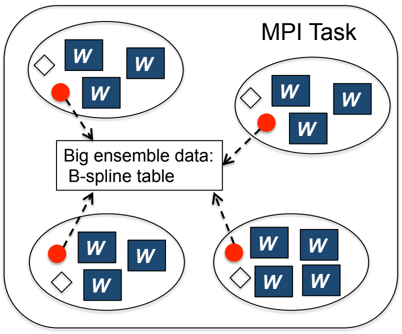


Figure 5: Schematic view of object distribution per MPI task. The ovals denote the OpenMP threads which own `Walker` objects (boxes), and other objects, e.g., Ψ_T (circle) and $\hat{H}$ (diamond).

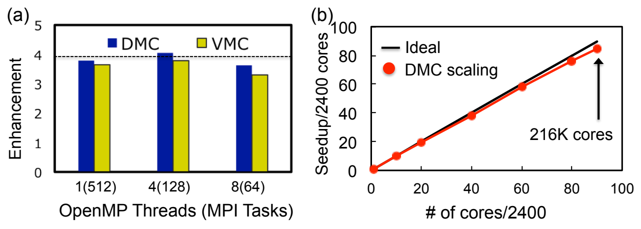


Figure 6: (a) Performance enhancement of OpenMP/MPI hybrid runs on a Cray XT4 when all the cores are utilized. The dotted line denotes the ideal enhancement of 4 on dual quad-core nodes. (b) Strong scaling of DMC on Cray XT5 using 6 OpenMP threads and two MPI tasks per dual hex-core node. All the data are collected on Jaguar at NCCS [12].

can be trivially parallelized using the `parallel-for` construct of OpenMP. However, achieving high performance with OpenMP requires careful data and thread managements.

Our hybrid implementation employs the distributed-memory programming model within OpenMP. Each thread manages a set of `Walkers` using thread-local storage. The objects associated with Ψ_T and $\hat{H}$ are replicated on all the threads except for large, *read-only* objects, such as B-spline tables. They are allocated at the task level and shared among all the threads in a task, significantly reducing the memory footprint of the B-spline representation. Figure 5 illustrates the distribution of `Walkers` and essential objects among OpenMP threads and MPI tasks.

We fully exploit the language features of C++ to manage objects and threads to maximize data locality, eliminate false sharing and minimize the OpenMP overheads. The code modification to implement OpenMP/MPI hybrid algorithms is limited to the high-level QMC drivers. The methods to migrate walker data sets between nodes for load balancing in DMC and accumulating statistical averages are unaffected by introducing the thread-level parallelism.

This hybrid OpenMP/MPI scheme has several advantages over the standard MPI-only scheme.

- Memory optimized: large read-only data to store single-particle orbitals and other shared properties to represent the trial wave function and many-body Hamiltonian can be shared among threads, which reduces the memory footprint of a large-scale problem.

- Data-locality optimized: the data associated with an active `Walker` are in cache during the compute-intensive drift-diffusion process and the operations on an `Walker` are optimized for cache reuse. Thread-local objects are used to ensure the data affinity to a thread.

- Load balanced: `Walkers` in an ensemble are evenly distributed among threads and MPI tasks. The two-level parallelism reduces the population imbalance among MPI tasks and reduces the number of point-to-point communications of large messages (serialized objects) for the `Walker` exchange.

- Communication optimized: the communication overhead, especially for the collective operations necessary to determine E_T and measure the properties of an ensemble, is significantly lowered by using fewer MPI tasks.

Effectiveness of the hybrid scheme is evident in Fig 6(a). The parallel efficiency remains high for any combination of OpenMP threads and MPI tasks even at this modest scale of 512 cores. In fact, DMC scales nearly perfectly with respect to the number of threads: the additional parallelism from multi threading allows more walkers per MPI task and improves the overall parallel efficiency and load balance among SMP nodes. The VMC efficiency reflects the memory bandwidth-limited nature of QMC algorithms with B-splines when all the computing cores are used. The performance enhancement less than the ideal 4 over the runs using only a quarter of the physical cores is expected, since the resources are shared among the multiple cores. This is not unique to OpenMP, as indicated by the pure MPI run using 512 tasks in Fig. 6(a).

For QMC simulations, the added parallelism with more cores on a SMP node is always beneficial, as $T_{cpu} \propto 1/N_{cores}$. The parallel efficiency of hybrid runs depends on the memory architecture and is subject to the quality of compilers and MPI implementations. In general, the best performance for a wide range of problems size is obtained when a MPI task is mapped over a NUMA node as shown in Fig. 6(a) with 4 threads on a quad-core processor.

Hybrid QMC algorithms have allowed us to apply QMC methods to the problems size at unprecedented scales on large-scale clusters of multi-core SMPs, efficiently utilizing all the available computing resources. At present, we can obtain the energy of a defect in a 64-Si crystalline system (256 electrons) to within 1 mHa in an hour using 4800 cores, completing a QMC workflow of VMC optimizations and DMC projection. The B-spline table for the system with a defect, for which no symmetry can be exploited to reduce the table size, requires more than 4 GB and it is necessary to employ 6 OpenMP threads and 2 MPI tasks on the dual hex-core nodes with 16 GB of memory. Figure 6(b) shows the parallel efficiency of DMC of the same system on Cray XT5 [12]. It shows near 95% of the ideal speedup up to 216K cores, which implies that the same *chemical accuracy* can be achieved in a much shorter wall-clock time.

5. LIMITATIONS AND WORKS IN PROGRESS

As previously discussed, QMC computations are dominated by evaluation of Ψ_T and the time to generate a MC sample grows with the problem size N as

$$\begin{aligned} T_{MC} &= N \times (T_{\mathcal{M}^{-1}} + T_{ratios} + T_{spo}) + T_{others} \\ &= \mathcal{O}(N^3) + \mathcal{O}(N^2) + \mathcal{O}(N^2 N_b), \end{aligned} \quad (15)$$

where N_b=1 with B-splines. Other computations involving Jastrow functions scale at most qudratically to N with much smaller prefactors. The number of samples required to reach the same accuracy weakly depends on the problem size, making the overall scaling of $\mathcal{O}(O^3) - \mathcal{O}(N^4)$. This favorable scaling and near-perfect parallel efficiency make QMC methods uniquely powerful tools to study the electronic structure of realistic systems on the current generation of HPC systems. So far, we could reap increasing computing power – faster CPU clock, more cores on a SMP node and more nodes – to improve the efficiency of QMC calculations through `Walker` parallelization. However, significant changes are needed for QMC methods to play expanded roles on emerging architectures which will have an order of magnitude higher concurrency and a deeper memory and communication hierarchy [13].

Our OpenMP/MPI hybrid methods employ many optimizations to improve computational efficiency at the expense of increased memory use. Each walker needs temporary data for update methods, which scales as $\mathcal{O}(N^2)$. This is in addition to the $\mathcal{O}(N^2)$ memory required for the B-spline table shared by all the walkers. As we scale out the problem size, the required memory will eventually become larger than what is available on a SMP node.

Several methods have been under development to overcome the memory limitation. (i) *Single-precision* can be used to store the B-spline table, halving the memory use and time in evaluating the single-particle orbitals. Our extensive tests have shown that results of the double and mostly single-precision GPU implementations agree within the statistical error we have achieved thus far [14]. (ii) We can use a *mixed-basis* approach. The mesh spacing required to accurately represent the orbitals is determined by their smallest feature size. The shortest wavelength features are concentrated around the atomic cores, while in the area between the atoms, the functions are smooth. For this reason, we divide space into spherical regions called muffin tins surrounding the atoms, and an interstitial region between them. Utilizing this dual basis in place of 3D B-splines alone typically allows the same accuracy to be achieved with 5× to 10× less memory, with about the same performance [15]. (iii) We are exploring *distribution* of the B-spline table over a group of SMP nodes using the Global Arrays framework [16]. It provides a partitioned local view of data in the MPI processes and a global-shared-memory abstraction and allows direct access to portions of gloabal arrays that are stored at a remote node's memory.

These improvements will extend the problems size we can study with QMC on large clusters of SMPs, the most common HPC architecture of today and in a near future. However, the scaling of $T_{MC} \sim \mathcal{O}(N^{3-4})$ with the problem size will ultimately place limits on QMC's applicability. Again, the increasing parallelism on newer processors and thread-based parallel programming models on a SMP, e.g., OpenMP 3.0 [3] and Intel TBB [17], provide QMC with solutions.

The forms of Ψ_T and $\hat{H}$ as sums of components, lend themselves to task parallelism using well-encapsulated objects. The critical computational step to evaluate the single-particle orbitals and the determinant updates can be carried out in parallel. Furthermore, we can expose the parallel loops over ions and electrons as in the GPU implementation [14]. Preliminary results show promising performance gains with OpenMP task and nested parallelism by defining large chunks of computational units for the tasks and carefully managing objects. The fine-grained task parallelism removes the constraint on the number of `Walker`s, currently at least one `Walker` per core, and enables practical QMC calculations of larger systems with much reduced time-to-solution.

As we improve the computational efficiency and memory use, other elements of the QMC algorithms that have been ignored will become new barriers to overcome. Our scaling studies expose the performance issues with blocking collectives and parallel I/O at large scales. The *true* parallel efficiency of a complete QMC workflow is also limited by the initialization stage, shown as the gray region of Fig. 2, and therefore reducing the equilibration time is essential to attain high efficiency with increasing parallelism. Solutions are being explored including increased intervals between the synchronizations, asynchronous I/O, use of non-blocking collectives and variable time steps for DMC to speed up the equilibration.

6. CONCLUSIONS

Many levels of parallelism are afforded by QMC algorithms, including, but not limited to the natural parallelism inherent in Monte Carlo methods, making QMC intrinsically scalable and ideally suited to take advantage of the growth in computational power. We presented OpenMP/MPI hybrid implementations that reduce the memory footprint by selectively replicating data for computational efficiency, while improving the overall parallel efficiency on large-scale clusters of SMPs based on multi-core processors. The walker-parallelization allows QMC methods to handle 1000s of electrons without incurring extra communication overhead and enables high accuracy QMC calculations of realistic systems. We have identified the barriers to overcome, as we further scale out problem size and improve the accuracy of QMC calculations, and offer several solutions to extend QMC's reach on future architectures. With this progress, perhaps in a few years time, QMC simulations will be as ubiquitous as DFT calculations are at present and will become the tool of choice for materials design by simulations.

7. ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy (DOE) under Contract No. DOE-DE-FG05-08OR23336 and by the National Science Foundation under No. 0904572. This research used resources of the National Center for Computational Sciences and the Center for Nanophase Materials Sciences, which are sponsored by the respective facilities divisions of the offices of Advanced Scientific Computing Research and Basic Energy Sciences of DOE under Contract No. DE-AC05-00OR22725. This work was performed in part under the auspices of the US DOE by LLNL under Contract DE-AC52-07NA27344. Sandia National Laboratories is a multiprogram laboratory managed and operated by

Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract No. DE-AC04-94AL85000. This research was supported by an allocation of advanced computing resources provided by the National Science Foundation, under TG-MCA93S030 and TG-MCA07S016, and utilized the Abe and Lincoln cluster at the National Center for Supercomputing Applications, as well as Kraken at the National Institute for Computational Sciences.

8. ADDITIONAL AUTHORS

David M Ceperley, National Center for Supercomputing Applications & Department of Physics, University of Illinois, email: `ceperley@illinois.edu`.

9. REFERENCES

[1] Foulkes W M C, Mitas L, Needs R J and Rajagopal G 2001 *Rev. Mod. Phys.* **73** 33–83

[2] Message passing library http://www.mpi-forum.org/

[3] Openmp application program interface http://www.openmp.org

[4] Jeongnim Kim, K Esler, J McMinis, B Clark, J Gergely, S Chiesa, K Delaney, J Vincent and D Ceperley QMCPACK simulation suite URL `http://qmcpack.cmscc.org`

[5] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides 1994 *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley) chap Methods for Coupled Electronic-Ionic Monte Carlo

[6] Hdf (hierarchical data format) URL `http://hdf.ncsa.uiuc,edu/HDF5`

[7] Box G E P and Jenkins G 1976 *Time Series Analysis: Forecasting and Control* (Holden-Day)

[8] Umrigar C J, Toulouse J, Filippi C, Sorella S and Hennig R G 2007 *Phys. Rev. Lett.* **98** 110201

[9] Esler K P, Kim J, Ceperley D M, Purwanto W, Walter E J, Krakauer H, Zhang S, Kent P R C, Hennig R G, Umrigar C, Bajdich M, Kolorenc J, Mitas L and Srinivasan A 2008 *Journal of Physics: Conference Series* **125** 012057 (15pp)

[10] D Alfè and MJ Gillan 2004 *Phys. Rev. B.* **70** 1661101(R)

[11] K Esler Einspline B-spline library, http://einspline.sf.net URL `http://einspline.sf.net`

[12] Jaguar (Cray XT), ORNL Leadership Computing Facility (OLCF)

[13] Dongarra J, Beckman P, Aerts P, Cappello F, Lippert T, Matsuoka S, Messina P, Moore T, Stevens R, Trefethen A and Valero M 2009 *Int. J. High Perform. Comput. Appl.* **23** 309–322 ISSN 1094-3420

[14] Esler K P, Kim J, Shulenburger L and Ceperley D M 2010 *Computing in Science and Engineering* **99** preprint

[15] Esler K P, Cohen R E, Militzer B, Kim J, Needs R J and Towler M D 2010 *Phys. Rev. Lett.* **104** 185702

[16] Tirukkovalur S, Dinan J, Niu Q, Sadayappan P, Kim J, Srinivasan A, Kumar S and Hammond J 2011 A global address space approach to automated data management for parallel quantum monte carlo applications

[17] James Reinders 2007 *Intel Thread Building Blocks* (O'Reilly Media)