ANL/ALCF/ESP-13/16

Multiscale Molecular Simulations at the Petascale (Parallelization of Reactive Force Field Model for Blue Gene/Q)

ALCF-2 Early Science Program Technical Report

Argonne Leadership Computing Facility

ANL/ALCF/ESP-13/16

Multiscale Molecular Simulations at the Petascale (Parallelization of Reactive Force Field Model for Blue Gene/Q)

ALCF-2 Early Science Program Technical Report

prepared by
Adrian W. Lang[1], Gard Nelson[2], Christopher Knight[3], and Gregory A. Voth[2]
[1]Argonne Leadership Computing Facility, Argonne National Laboratory
[2]University of Chicago
[3]Computing, Environment, and Life Sciences, Argonne National Laboratory

May 7, 2013

Parallelization of Reactive Force Field Model for Blue Gene/Q

Adrian W. Lange,[1] Gard Nelson,[2] Christopher Knight,[3] and Gregory A. Voth[2,4]

[1]*Leadership Computing Facility, Argonne National Laboratory, Argonne, IL 60439, USA*
[2]*Department of Chemistry, University of Chicago, Chicago, IL 60637, USA*
[3]*Computing, Environment, and Life Sciences, Argonne National Laboratory, Argonne, IL 60439, USA*
[4]*James Franck Institute, Institute for Biophysical Dynamics, and Computation Institute, University of Chicago, Chicago, IL 60637, USA*

(Dated: April 1, 2013)

Our efforts in developing a highly parallel implementation of the reactive force field model, the Multi-state Empirical Valence Bond (MS-EVB) method, are discussed. We have introduced multi-threading, a state decomposition parallelism, and replica exchange capabilities. Example calculations are presented to demonstrate the improved productivity and scalability of the new code on the Blue Gene/Q supercomputer, Mira. We show that we are now able to successfully scale to half of Mira and have the potential to scale even further.

I. INTRODUCTION

Modeling the dynamics of complex biological molecular systems, such as proteins solvated in water, is a crucial facet of understanding how life works at the level of atomic detail and also understanding how we might be able to develop biotechnology for energy production and/or medical purposes. The vast number of atoms present in biological systems, though, prohibits the application of highly accurate quantum mechanical electronic structure methods, because the computational cost of these methods scales exponentially with respect to the system size (*i.e.*, the number of electron basis functions). In order to study the dynamics of such molecular systems with atomic detail, one instead opts to employ a classical molecular mechanics force field whose energy and forces can be computed at orders of magnitude faster than electronic structure. A typical molecular mechanics force field developed specifically for biological molecules approximate chemical bonds as harmonic springs between two atoms. While this approximation works reasonably well for most molecular systems near equilibrium, it is unable to account for the realistic quantum mechanical nature of chemical bonds, which can break and form dynamically.

The Multi-state Empirical Valence Bond (MS-EVB) method[1,2] is one way to incorporate chemical reactivity into an otherwise non-reactive force field, enabling the study of chemical reaction dynamics in complex biological systems without suffering the expense of electronic structure calculations. In short, MS-EVB makes the *ansatz* that a reactive molecular system can be decomposed into a set of diabatic states, each representing one of several possible chemical bonding topologies, akin to the "resonance structures" familiar to most chemists. These diabatic states form a basis set, and the total system is then a linear combination of the states, written as

$$|\Psi\rangle = \sum_I c_I \, |\psi_I\rangle \ , \tag{1.1}$$

where $|\psi_I\rangle$ is a diabatic state with coefficient c_I. MS-EVB then constructs a model Hamiltonian matrix, $\mathbf{H}$. The diagonal elements, H_{II}, are simply the total energy of each state. The off-diagonal elements, H_{IJ}, are a coupling energy between two states that serve as a reactant and a product state (see Ref. 2 for more detail). Finally, the Hamiltonian is diagonalized according to a Schrödinger-like equation,

$$\mathbf{Hc} = E\mathbf{c} \tag{1.2}$$

to yield a set of eigenvalue energies, E, and eigenvectors, $\mathbf{c}$, containing the coefficients c_I. The minimum eigenvalue energy is selected as the final MS-EVB energy, and forces on each atom are computed via the Hellman-Feynman theorem. The MS-EVB approach can thus be viewed as a coarse-graining (or, a multi-scale) approach to quantum chemistry electronic structure, which allows us to access larger molecular systems sizes and longer time scales.

MS-EVB is a straightforward framework, yet it is a challenge to implement effectively in a code capable of taking advantage of modern supercomputers with thousands of core processors. In the remainder of this work, we discuss how we have tackled this challenge with several advances in our MS-EVB code, Rapid Approach to Proton Transport and Other Reactions (RAPTOR).[3] Specifically, we discuss the parallelization of RAPTOR tailored for running on the IBM Blue Gene/Q (BGQ) supercomputer, Mira, housed at the Argonne Leadership Computing Facility.

Throughout this work, we focus primarily on an example application of MS-EVB for proton transport through a transmembrane protein, Cytochrome *c* Oxidase (CcO). Cco serves as a representative system of interest, although the RAPTOR code is generalizable to a variety of other reactions and molecular systems as well. Our model CcO system consists of 159,519 atoms treated with the CHARMM22 force field for protein molecules and the CHARMM36 force field for the lipid molecules. Water molecules are treated with the SPC/Fw force field, in accord with the MS-EVB3 model,[2] which is used to model the proton transfer reactivity. We use a cutoff

of 10 Å (smoothly attenuated to zero with a switching function beginning at 8 Å) for van der Waals and short-range Coulomb pairwise interactions. Molecular dynamics (MD) simulations are carried out at constant volume and temperature with a Nose-Hoover thermostat. MD is propagated with Velocity-Verlet integration with 1 femtosecond time step intervals.

All calculations presented below are performed on the BGQ architecture supercomputers—each compute node containing 16 core processors and each core containing a quad floating point unit (FPU)—at the Argonne Leadership Computing Facility. Codes have been compiled with the IBM XL compilers with level -O3 optimizations. Performance of the codes is measured as productivity in terms of MD time steps per wall second.

II. RAPTOR IMPLEMENTATION

A. Interface to LAMMPS

The RAPTOR code is an optionally installed user package interface to the LAMMPS,[4] a widely used and freely available, open-source parallel molecular dynamics code written in C++. LAMMPS is parallelized mainly through a domain decomposition scheme, wherein a given unit cell, simulated with periodic boundary conditions (PBC), is divided in Cartesian space into many smaller domains, each of which is treated separately on an individual rank with Message Passing Interface (MPI). Intra-domain interactions are thereby computed entirely in parallel, and inter-domain interactions are handled with minimal MPI communications with neighboring domains.

The domain decomposition scheme in LAMMPS is very effective and scalable for the bonded interactions (*e.g.*, bonds, angles, dihedrals) and short-range non-bonded interactions (*e.g.*, van der Waals interactions) of a force field. Because the magnitude of short-range non-bonded interactions approaches zero rapidly, an interactions cutoff radius can be used. However, for a fixed size system, one cannot divide the unit cell into domains of size smaller than the cutoff radius without errors and/or loss of parallel efficiency, placing a limit on how many MPI ranks can be used with domain decomposition scheme alone. In addition, pairwise Coulomb interactions usually cannot be approximated with a cutoff because $1/r$ does not approach zero fast enough to ignore without possibly introducing severe errors.

The Particle-Particle Particle-Mesh[5] (PPPM) approach (available in LAMMPS) is therefore used in our calculations to divide the work into a short-range part, handled in conjunction with the pairwise van der Waals interactions via domain decomposition, and a long-range part, handled with a Fast Fourier Transform (FFT) performed as parallel calls to the FFTW library. The long-range electrostatics computation (*i.e.*, the k-space computation), despite being $O(NlogN)$ scaling, is unfortunately not nearly as efficiently parallel as the domain decomposition (for the relatively small three dimensional grids used in our calculations) because it involves a substantial amount of MPI communication in order to broadcast the electrostatics from the PPPM mesh globally to all processors, an all-to-all style communication. Moreover, each PPPM evaluation requires 4 3D-FFTs, one forward and three reverse. For MS-EVB calculations, this is compounded further by the fact that the MS-EVB model requires electrostatic energies in the off-diagonal coupling matrix elements, adding to the amount of k-space work. Indeed, at large counts of MPI ranks, the k-space computation can dominate the wall time in MS-EVB calculations with RAPTOR.[3,6] For this reason, previous work in our research group[3] had been devoted to developing a partitioning scheme in which the real-space bonded and non-bonded short-range interactions are performed on one partition of MPI ranks and the k-space computation is performed concurrently on another partition. Thereby, one could assign fewer MPI ranks to the k-space work to hide some of its poorly scaling communication, resulting in improved parallel efficiency and a modest overall speedup at a large number of processors.

To demonstrate this, we compare results of productivity for the Cco model using only the MPI domain decomposition in LAMMPS and also the k-space partitioning scheme (MPI/split) in Figure 1. At 64 nodes (1024 MPI ranks), the domain decomposition has reached the limit where further spatial division competes with the size of the cutoffs, hitting the intrinsic domain limit mentioned above.

As one can infer from Figure 1, RAPTOR does not scale very well with only the domain decomposition or the k-space partitioning scheme, dropping below 50% parallel efficiency at just 16 nodes, a far cry from the 49,152 nodes on Mira. The k-space partitioning improves the scalability slightly, but, again it cannot scale further due to the domain decomposition limit. Therefore, we are compelled to develop new parallelism approaches to improve scalability.

B. Multithreading with OpenMP and QPX SIMD

LAMMPS has recently added the ability (through an optional user package) to take advantage of shared memory multithreading parallelism via the OpenMP API. It does so with a so-called "force decomposition" scheme, where, for example, the pairwise additive N^2 force loop is distributed across threads. Similar loop-level parallelism is implemented for bonded interactions and parts of the PPPM k-space calculation, such as interpolating charges to the mesh. Note that the OpenMP force decomposition is in addition to the usual MPI domain decomposition in LAMMPS.

This multithreaded code, however, was implemented for the general purpose parts of LAMMPS, of which RAPTOR only uses for its diagonal matrix elements in

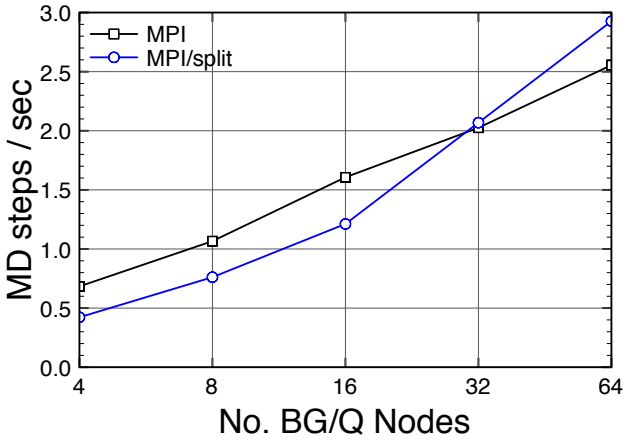

FIG. 1: Strong scaling of RAPTOR code with the CcO model comparing domain decomposition only (MPI) and k-space partitioning (MPI/split) schemes. Runs are carried out in c16 mode (*i.e.* with 16 MPI ranks per node). The Cco system contains 15 MS-EVB states on average.

the model Hamiltonian. So it was our task to merge this existing OpenMP code with RAPTOR and also rewrite a number of portions of RAPTOR that would benefit from such force decomposition multithreading. For instance, this involved multithreading certain pairwise loops for off-diagonal coupling matrix elements as well as rewriting parts of the LAMMPS PPPM interface with RAPTOR. In the process of this re-coding, we also introduced a handful of basic serial optimizations (*e.g.*, loop unrolling or conditional hoisting) that the compilers were unable to recognize in complex loop structures, amounting to a 15% speedup even with running only a single thread. In the PPPM interface of RAPTOR, we were also able to share a few of the FFT calls across thread partitions, providing some additional concurrency. The OpenMP multithreading was further combined with the k-space partitioning scheme, code which has also been introduced to the general release of LAMMPS now.

In addition to OpenMP multithreading, we explored the special BGQ-specific QPX vector intrinsics API to take advantage of the possibility of SIMD parallelism on BGQ. For our purposes, we were introduced QPX into the short-range non-bonded pairwise loops as well as parts of the PPPM code specific to RAPTOR. Since these loops have a number of branching conditional statements, it was not straightforward to take full advantage of QPX. We ultimately found that a "buffer/flush" approach was most successful. In this approach, we use a set of `vector4double` buffers to temporarily store information needed to complete a pairwise interaction (or other quantity). The bulk of the floating point operations in the loop are delayed until we have filled the buffers, and then the buffers are flushed by computing four pair interactions simultaneously with QPX vector intrinsic functions. We note that the usual LAMMPS pairwise loop uses a lookup table for the short-range Coulomb interaction, which otherwise involves a somewhat expensive polynomial expansion to approximate the error function. The lookup table is typically faster than the explicit evaluation of the polynomial, but a lookup table is not amenable to QPX vectorization. Thus, we applied QPX to the polynomial expansion and omitted the lookup table. Compared to no QPX vector intrinsics and without the lookup table, our buffer/flush code exhibits a $\sim$50% speedup for the non-bonded force kernel. Compared to no QPX and with the lookup table, though, we observe a $\sim$30% speedup for the non-bonded force kernel. While this is clearly shy of the theoretical speedup of QPX, we nonetheless have found it a welcome enhancement.

Putting all of the above together, we show in Figure 2 the improvements to scaling that are realized with OpenMP and QPX. The BGQ is capable of hyperthreading the quad FPU with OpenMP threads, but this may not always prove beneficial since it may prevent the use of QPX SIMD or possibly increase cache misses due to less memory per MPI rank. We examine this in Figure 2 with two different modes, c4o16 being 4 MPI ranks per node with 16 threads per rank (hyperthreaded mode), and c1o16 being 1 MPI rank per node with 16 thread per rank. It is quite clear that multithreading provides a significant speedup compared to the MPI domain decomposition or k-space partitioning for the same number of nodes. It also pushes the domain decomposition limit to greater node counts since threads reduce the number of MPI ranks per node. Notice that the c1o16 mode (the non-hyperthreaded mode) exhibits lower productivity than c4o16 mode (the hyperthreaded mode) at low node counts, but because c1o16 mode has better parallel efficiency, c1o16 scales further and eventually is more productive as node count increases. This appears to be the result of a combination of causes. One is the ability to use QPX in c1o16 mode both in the FFTs (automatically vectorized by compiler) and our special non-bonded force kernels. Another appears to be that the MPI communication is appreciably faster in c1o16 mode (as compared to c4o16 mode), likely because there is no intra-node communication in c1o16, making the MPI ranks closer in the communication network.

C. State Decomposition

Multithreading has certainly improved RAPTOR's parallelism, yet there has still been one aspect of RAPTOR that has been treated serially up to this point: the matrix elements in the model Hamiltonian are evaluated one at a time. In our model Cco example, there are on average 15 states in the MS-EVB Hamiltonian, and each state's energy and forces can be computed independently of the others. This glaringly obvious source of parallelism had gone untapped until now when we introduced our new "state decomposition" parallel scheme.

We accomplish the state decomposition in a manner

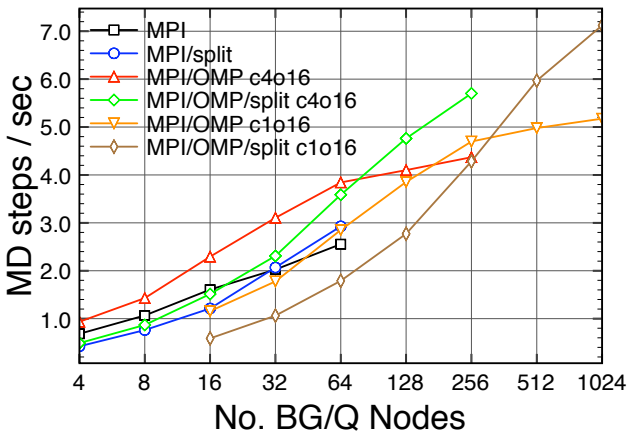

FIG. 2: Strong scaling of RAPTOR code with the CcO model with OpenMP multithreading and QPX (data labeled with OMP). Results from Figure 1 are included for comparison. The mode is listed for each OMP run. c4o16 = 4 MPI ranks per node and 16 threads per MPI rank (hyperthreaded mode). c1o16 = 1 MPI rank per node and 16 threads per MPI rank. MPI/OMP/split combines the OpenMP threading with the k-space partitioning scheme. Each line of data is extended to the domain decomposition limit.

analogous to the k-space partitioning scheme, but instead of dividing by real-space and k-space work for all MS-EVB states, we divide the work by individual MS-EVB state. This allows us to have not just two partitions (as in k-space partitioning) but as many partitions as we have MS-EVB states, distributing the work of both real-space and k-space across more processors. In practice, though, because the number of MS-EVB states is dynamic during the course of an MD simulation, we find that it is most efficient to use slightly fewer state partitions than there are MS-EVB states in order to avoid workless partitions.

The state decomposition scheme in RAPTOR is carried out by running LAMMPS in partitioned mode, which simply uses `MPI_Split` to create a new MPI subcommunicator for each partition. All partitions share the same coordinates and velocities so that they all end up with the same set of MS-EVB states after completing the MS-EVB state search algorithm.[2] Once the states have been determined, each partition decides which state(s) to work on with a simple static load balance and then computes the work. Thus, each partition stores in memory only a subset of the entire set of MS-EVB state energies and forces. With the energies and couplings in hand, the partitions perform an `MPI_Allreduce` of the MS-EVB Hamiltonian [a modest communication of data size $O(N_{\text{states}}^2)$ with N_{states} usually less than 20], which is subsequently diagonalized by each partition to yield the minimum eigenvalue energy (*i.e.*, the MS-EVB energy) and its corresponding eigenvector coefficients. Each partition computes the MS-EVB force on atoms according to the Hellman-Feynman theorem:

$$F_i = -\sum_{I,J}^{\text{states}} c_I c_J \frac{\partial H_{IJ}}{\partial \mathbf{x}_i} \,, \tag{2.1}$$

where F_i is the force on the i-th atom with respect to the $\mathbf{x}_i$ Cartesian coordinate, and c_I is the I-th coefficient from the minimum energy eigenvector. Since each partition has only stored in memory those matrix elements ($\partial H_{IJ}/\partial \mathbf{x}_i$) that it has worked on, another MPI all-reduce communication is necessary, which, in order to take advantage of MPI collective communications, is performed within an MPI communication group containing the same spatial domains of different partitions. The time step can then be propagated as usual on each partition, updating coordinates and velocities for the next iteration.

We present the productivity of the state decomposition scheme in Figure 3. Note that the state decomposition is a layer of parallelism on top of the MPI domain decomposition and OpenMP multithreading. We observe an appreciable speedup as we add more state partitions, with 8 partitions scaling best for the CcO model, which has on average 15 MS-EVB states in the MD run. At low node counts, however, state partitioning is not as productive as the other schemes. This is not especially surprising since adding more state partitions reduces the number of ranks in the domain decomposition, a tradeoff in computational speed. Nonetheless, the state decomposition exhibits better parallel efficiency and eventually is more productive as the node count is increased.

The new state decomposition and multithreading has certainly improved the scalability and productivity of RAPTOR by a great deal. Before, RAPTOR could hardly scale to 32 BGQ nodes, but now we are capable of scaling to multiple BGQ racks (1 rack = 1,024 nodes) with acceptable parallel efficiency.

III. REPLICA EXCHANGE UMBRELLA SAMPLING

In addition to the advances made in parallelizing RAPTOR for single trajectory MD simulations, we also have implemented a novel ensemble-run (replica exchange umbrella sampling) interface to perform enhanced sampling ensemble MD simulations.

A. Background

Umbrella sampling is a commonly used technique to compute the free energy along a chosen path. The path is usually referred to as a "collective variable" (CV) because it often involves a combination of atom coordinates, such as the distance or angles between groups of atoms. The CV is then divided into multiple "windows", and each window is given an artificial restraint (*i.e.*, a bias) to

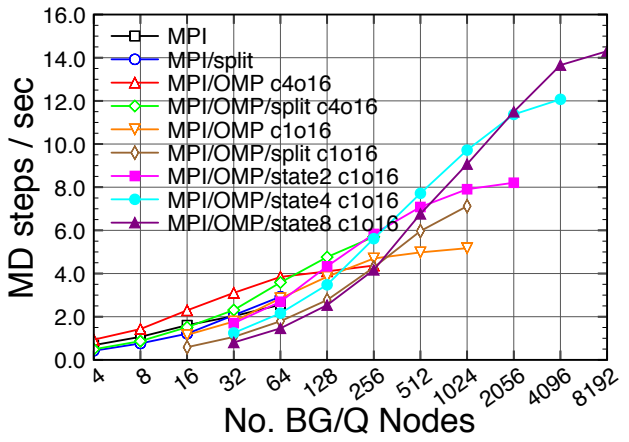

FIG. 3: Strong scaling of RAPTOR code with the CcO model with state decomposition. Results from Figure 1 and Figure 2 are included for comparison. The mode is listed for each OMP run. c4o16 = 4 MPI ranks per node and 16 threads per MPI rank (hyperthreaded mode). c1o16 = 1 MPI rank per node and 16 threads per MPI rank. The number of state partitions is listed, where "state2" is 2 state partitions, for example. Note that the data for MPI/OMP c1o16 is the equivalent of using one state partition. Each line of data is extended to the domain decomposition limit.

be applied to the molecular system of interest in order to keep the system near that window in the CV space. By doing so, one ensures good sampling of each window along the CV, which might otherwise not be the case if parts of the CV are energetically unfavorable. After sampling the windows, usually with MD runs, one uses statistics of the biased energy to compute a relative free energy value at each window, amounting to a free energy surface along the CV.

The umbrella sampling approach described above can be performed by running each window independently in separate calculations. However, it is now well-known that independent runs like this can produce artifacts and/or poorly converged free energy surfaces if the MD runs are not long enough and/or neighboring windows do not have substantial overlap in CV space. One solution to overcome this issue is to perform a loosely-coupled ensemble simulation, where all windows are run simultaneously in one big single calculation. At certain points in time, the ensemble simulation is halted, and one attempts to swap umbrella sampling restraints between neighboring windows. Swaps are accepted or rejected according to the canonical Metropolis Monte Carlo criteria to satisfy detailed balance. This approach is known as replica exchange umbrella sampling (REUS), and it has been shown to reduce artifacts and speed convergence in umbrella sampling.[7] The swapping of windows increases the sampling overlap between neighboring windows, and it helps to prevent the system from becoming "stuck" in metastable energy wells. REUS requires more computational power to run all the windows simultaneously, but it actually reduces the time to solution of a converged free energy surface as compared to completely uncoupled umbrella sampling windows, thereby actually requiring less overall CPU time.

B. LAMMPS Ensembles Implementation

In our work, we are interested in computing free energy surfaces of proton transport, and we want to use REUS enhanced sampling to speed the process. In the CcO model, we have identified a CV for proton uptake along a certain channel of the protein that leads to a catalytic complex buried within the interior of the protein. Details of this proton uptake channel will not concern us here and can be found elsewhere in the literature. For the purpose of the current work, we instead want to focus on the computational aspect.

Prior to this report, we were only able to perform the conventional uncoupled umbrella sampling with RAPTOR. We have now developed an implementation of REUS for RAPTOR, which is part of a development code by the name of LAMMPS Ensembles (LE). LE works by using `MPI_Split` to create a set of MPI subcommunicators in which each subcommunicator creates its own instance of the LAMMPS program class. Each LAMMPS instance runs MD of its own replica of the molecular system within its subcommunicator. At a specified time interval, the ensemble of replicas must synchronize in order to attempt swaps according to the REUS algorithm.

We have modified LE to perform REUS with RAPTOR. LE is yet another layer of parallelism on top of the previously described domain decomposition, state decomposition, and multithreading. A challenge unique to MS-EVB in REUS, though, is that there can exist a substantial load imbalance between replicas. Every replica contains the same molecular system but with different coordinates, and since the number of MS-EVB states depends on the spatial configuration of water molecules in our simulations, every replica has a different amount of work to compute. Fewer states means less work and will complete a fixed number of time steps in less wall time than a replica with more states. The issue is that REUS must synchronize the ensemble runs to attempt swapping, and this must be done after completing a predetermined number of MD steps. Those replicas with few states will, therefore, wait in an `MPI_Barrier` until the replica with the most number of states completes, an obvious waste of time.

To address this the load imbalance issue, we have introduced a dynamic load balance algorithm to REUS. In this scheme, we assign one of the LE replica subcommunicators to act as a listener. After completing the predetermined number of MD steps, each replica sends a message to the listener to inform the listener of being finished. The listener continues to receive messages until all replicas have reported in, at which time the listener broadcasts a signal to all other replicas that swapping

may proceed. In the meantime, while the non-listener replicas are waiting for the signal from the listener, the replicas asynchronously continue to take more MD steps, checking back for the signal after every few MD steps. The listener replica also continues to run similarly during this time. The result is that the ensemble continues to produce MD sampling while concurrently waiting to synchronize for the swapping. Ultimately, the replicas with fewer states produce several more MD steps than those with more states. But, because of the swapping in REUS, the "fast" replicas traverse the windows such that no individual window is over-sampled compared to others, which could be the case in conventional umbrella sampling.

In Figure 4, we present a weak scaling plot of using the REUS code with RAPTOR. The molecular system is still the usual CcO model from before, and we compare running REUS with different numbers of replicas using the usual synchronous static load balance versus using our asynchronous dynamic load balance, described above. Each replica in these data runs the CcO model system on 128 nodes in c1o16 mode using 2 states in the state decomposition scheme. Swaps are attempted after 200 MD steps (i.e., 200 femtoseconds). In the asynchronous dynamic load balance scheme, this is the number of steps a replica takes before sending a message to the listener replica. The full REUS run involves reported here involve 20,000 MD steps, but because replicas continue to run while waiting in the asynchronous scheme, the total number of MD steps in the end may differ. So, as the measure of performance, we present the mean productivity of the replicas in the ensemble,

$$\text{Mean productivity} = \frac{1}{tN_{\text{rep}}} \sum_{i}^{\text{replicas}} N_{i,\text{step}} \ . \qquad (3.1)$$

where $N_{i,\text{steps}}$ is the number of MD steps taken by replica i, t is the wall time, and N_{rep} is the number of replicas in the ensemble. Also, the standard deviation in the total number of MD steps taken is reported for the asynchronous scheme.

Figure 4 shows the REUS algorithm has fairly good weak scaling properties, having not dropped below half of parallel efficiency after increasing the number of replicas by 16 times both in the synchronous and asynchronous algorithms. However, the asynchronous algorithm is more productive by nearly as much as 3 MD steps/second on average at 64 replicas, a result of the improved load balance. Furthermore, the asynchronous algorithm does not make certain replicas over-sample any window too much, as the standard deviation of MD steps taken suggests.

Finally, we present strong scaling data for an REUS run encompassing the full CV of our CcO system in Table I. The CV is divided in total into 96 windows (*i.e.*, 96 replicas) running in c1o16 mode with 2 states in the state decomposition scheme and using the asynchronous dynamic load balance algorithm for REUS. Note that 24576 nodes represents half of Mira. We observe that

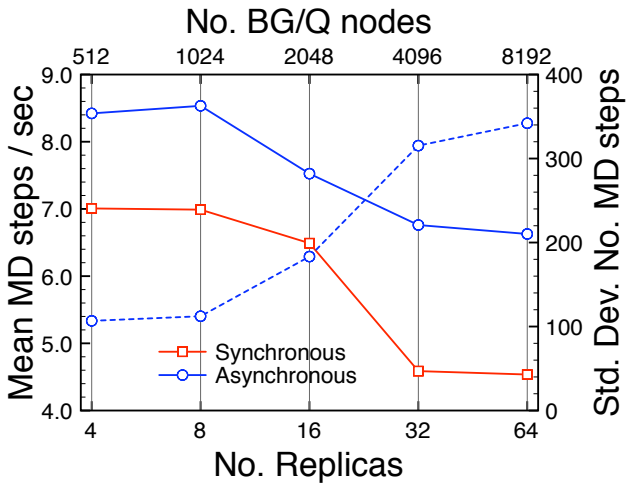

FIG. 4: Weak scaling of REUS with RAPTOR comparing the usual synchronous static load balance and the asynchronous dynamic load balance. Replicas are run on 128 nodes in c1o16 mode using 2 states in the state decomposition scheme.

TABLE I: Strong scaling of REUS with RAPTOR. 96 replicas run in c1o16 mode with 2 states in state decomposition scheme using asynchronous dynamic load balance algorithm.

No. BGQ nodes	Replica mean MD steps/sec[a]	Parallel efficiency[b]	Ensemble MD steps/sec[c]
3072	2.49	1.00	239.0
6144	4.33	0.85	415.7
12288	7.08	0.67	679.7
24576	10.43	0.50	1001.3

[a] Eq. (3.1).
[b] Relative to 3072 nodes.
[c] Eq. (3.2).

our code performs quite well in strong scaling, remaining above 50% parallel efficiency in all cases. In addition to the mean productivity, we present the ensemble productivity,

$$\text{Ensemble productivity} = \frac{1}{t} \sum_{i}^{\text{replicas}} N_{i,\text{step}} \ , \qquad (3.2)$$

which provides a measure of how much useful sampling data is being produced per second with the single calculation. That is, at 24,576 nodes (or 393,216 core processors), our REUS run is producing, on average, just over 1 picosecond total of reactive MD every wall second, a truly astounding achievement possible only on a leadership scale supercomputer like Mira.

IV. CONCLUSION

We have provided an overview of our efforts in transforming the originally poorly scaling RAPTOR code into one which can harness up to half of Mira (24 racks)

with appreciable parallel efficiency. This has been accomplished by introducing multithreading via OpenMP, developing a new state decomposition parallel algorithm, and creating a novel implementation of replica exchange umbrella sampling with the LAMMPS Ensembles code. Our new highly scalable RAPTOR code is currently being used on Mira to compute free energy surfaces of proton transport in complex biological systems at a unprecedented rate. We expect to report the results of these calculations in future work.

V. ACKNOWLEDGEMENTS

This work has been supported through the Early Science Project at the Argonne National Laboratory Leadership Computing Facility. A. W. Lange would like to extend a special thanks to Jeff Hammond at the Argonne Leadership Computing Facility as well as to Luke Westby at the University of Notre Dame for providing the development code for LAMMPS Ensembles, which has served as the foundation of our REUS code.

[1] U. W. Schmitt and G. A. Voth. The computer simulation of proton transport in water. *J. Chem. Phys.*, 111(9361), 1999.

[2] Y. Wu, H. Chen, F. Wang, F. Paesani, and G. A. Voth. An improved multistate empirical valence bond model for aqueous proton solvation and transport. *J. Phys. Chem. B*, 112:467–482, 2008.

[3] Y. Peng, C. Knight, P. Blood, L. Crosby, and G. A. Voth. Extending parallel scalability of lammps and multiscale reactive molecular dynamics. Technical report, 2012.

[4] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comp. Phys.*, 117:1–19, 1995.

[5] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Taylor and Francis Group, New York, 1988.

[6] T. Yamashita, Y. Peng, C. Knight, and G. A. Voth. Computationally efficient multiconfigurational reactive molecular dynamics. *J. Chem. Theory and Comput.*, 8:4863–4875, 2012.

[7] W. Jiang, Y. Luo, L. Maragliano, and B. Roux. Calculation of free energy landscape in multi-dimensions with hamiltonian-exchange umbrella sampling on petascale supercomputer. *J. Chem. Theory and Comput.*, 8:4672–4680, 2012.

Argonne Leadership Computing Facility
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439

www.anl.gov