RepEx: A Flexible Framework for Scalable Replica Exchange Molecular Dynamics Simulations

Antons Treikalis, Andre Merzky
Department of Electrical
and Computer Engineering
Rutgers University
Piscataway, New Jersey, 08854

Haoyuan Chen, Tai-Sung Lee, Darrin M. York
Department of Chemistry
and Chemical Biology
Rutgers University
Piscataway, New Jersey, 08854

Shantenu Jha*
Department of Electrical
and Computer Engineering
Rutgers University
Piscataway, New Jersey, 08854
* shantenu.jha@rutgers.edu

***Abstract*—Replica Exchange (RE) simulations have emerged as an important algorithmic tool for the molecular sciences. Typically RE functionality is integrated into the molecular simulation software package. A primary motivation of the tight integration of RE functionality with simulation codes has been performance. This is limiting at multiple levels. First, advances in the RE methodology are tied to the molecular simulation code for which they were developed. Second, it is difficult to extend or experiment with novel RE algorithms, since expertise in the molecular simulation code is required. We propose the RepEx framework which addresses the aforementioned limitations, while striking the balance between flexibility (any RE scheme) and scalability (several thousand replicas) over a diverse range of HPC platforms. This paper introduces the RepEx framework, the primary contributions of which are: (i) its ability to support different RE schemes independent of molecular simulation codes, (ii) the ability to execute different exchange schemes and replica counts independent of the specific availability of resources, (iii) a runtime system that has first-class support for task-level parallelism, and (iv) provide the required scalability along multiple dimensions.**

I. Introduction

The Replica Exchange (RE) class of methods [1] is a popular technique to enhance sampling in molecular simulations. Although RE methods were introduced for Monte Carlo methods, their use with Molecular Dynamics (MD) has grown rapidly. There are several hundred publications every year using some variant of Replica Exchange Molecular Dynamics (REMD) in a range of scientific disciplines including chemistry, physics, biology and materials science.

REMD simulation consists of two phases: one phase is comprised of MD simulations of N different replicas of the original system, where each replica has different thermodynamic configuration. The other phase involves exchanges of the thermodynamic configurations between replicas using Metropolis-like acceptance criterion. Initially, REMD [2] was used to perform exchanges of temperatures, but has since been extended to perform Hamiltonian Exchange [3], pH Exchange [4] and other exchange types.

Reinforcing the importance of RE, many community MD engines [5], [6], [7], have evolved to support internal RE implementations. These solutions often demonstrate respectable performance but share some significant limitations, many of which stem from the tight integration of RE method with the MD simulation engine. For example, tight integration results in a significant duplication of effort between "competing" MD engines; what is worse, tight integration results in RE algorithmic innovation that is localized to specific MD engines, making it difficult to propagate the innovation across community codes. MD engines [5], [6], [7] are highly optimized and specialized codes, often requiring tens of person-years of development. Domain scientists are typically unprepared for the full complexity of these MD engines, yet they are the ones most capable of algorithmic and methodological innovation. Furthermore, the number of exchange parameters, order and number of dimensions are typically hard-coded; the tight integration introduces a barrier to methodological advances and extensibility.

This points to the role of a framework that decouples the RE methodology from MD engines as well as manages the complex runtime requirements of multiple concurrent simulations. REMD simulations should also be expressed independent so as to be agnostic to the details of mapping of replicas to underlying resources. Furthermore, a REMD framework should ensure that in the presence of a failure of single replica, the entire set of simulations does not have to be stopped or restarted. These and other reasons reiterate the importance of a framework with an effective runtime system as the number of replicas and time-scales increases.

In this paper we present RepEx framework [8] – a user-level software framework with a runtime system designed to support the requirements of scalable REMD simulations over multiple dimensions. Conceptually, RepEx aims to decouple the implementation of RE algorithm from MD simulation engine. RepEx also decouples the resource management from the REMD algorithm and MD simulation engine specific details.

RepEx relies on RADICAL-Pilot (RP) [9] as a runtime system to perform resource allocation, task scheduling and data movement. Another distinctive feature of RepEx, partly arising from the use of RP, is fault tolerance: RepEx can either continue a simulation in the case of replica failure or can relaunch a failed replica.

The functionality and flexibility come at a performance price, especially when compared to highly-customized approaches. We carefully characterize the performance "penalty"

and argue that it is an acceptable trade-off given the functionality and flexibility enhancements proffered by RepEx. In fact, as we will chronicle, the diversity in algorithms, exchange parameters and dimensionality at adequate scales is unprecedented. The design of RepEx facilitates implementation of new RE algorithms with a wide range of MD engines. The implementation of RepEx supports up to three-dimensional REMD simulations with an arbitrary ordering of available exchange types.

This paper is organized as follows: in Section II we outline existing approaches to REMD simulations. In Section III we define the requirements of REMD simulations, provide an overview of the design of RepEx (subsection III-B) and its implementation (subsection III-C). We introduce two concepts – **Replica Exchange Pattern** and **Execution Mode**. In Section IV we present experiments conducted to demonstrate features, capabilities and characterize the performance of RepEx for 1D REMD and REMD simulations in multiple dimensions (M-REMD). We present results using two widely used MD engines – Amber and NAMD, and demonstrate the capability to use multiple CPU nodes for a single replica. Section V concludes with an analysis of RepEx relative to other existing REMD approaches and frameworks.

II. Related Work

In this section, we focus on some novel approaches and frameworks that are designed to implement RE algorithms. We discuss a wider spectrum of software packages, including molecular simulation software packages with integrated RE capability, until the closing section of this paper.

CHARMM: Ref. [10] presents an implementation of a 2D US/H-REMD method, implemented in `REPDSTR` module of the CHARMM [11]. `REPDSTR` uses an MPI level parallel/parallel mode where each replica is assigned multiple MPI processes and dedicated I/O routines.

`REPDSTR` was tested on IBM Blue Gene/P using the binding of calcium ions to the small protein Calbindin D_9k. Obtained results show that 2D US/H-REMD significantly improves the configurational sampling for biological potential of mean force (PMF) calculations, and thus facilitates convergence of the simulation. Authors presented strong scaling performance of 2D US/H-REMD, involving 4096 replicas and utilizing up to 131072 CPUs with nearly linear scaling.

MCA implementation with NAMD: Ref. [12] presented a Charm++ based implementation designed to run MCA was presented. It is tightly bound to the NAMD simulation engine. Charm++ is used to run multiple NAMD instances concurrently, which exchange messages via point-to-point communication functions of Tcl scripting interface. The strong scaling behavior of the swarms-of-trajectories string method implementation using the full-length c-Src kinase system utilizing up to 524288 cores on Blue Gene/Q was presented, as were results of temperature exchange REMD simulations with peptide acetyl-$(AAQAA)_3$-amide [13] in TIP3 solvent on Blue Gene/Q (utilizing up to 32768 cores).

Asynchronous approaches: Ref. [14], [15] presented ASyncRE package, developed to perform large-scale asynchronous REMD simulations on HPC systems. It implements two REMD algorithms: multi-dimensional RE umbrella sampling with Amber [5] and BEDAM λ RE alchemical binding free energy calculations with IMPACT [16]. AsyncRE uses a similar runtime system as RepEx, is capable of launching more replicas than there are CPUs allocated and has some fault-tolerant capabilities.

Ref. [17] introduced another REMD package targeted at asynchronous RE and optimized for volunteer computing resources. The package can be used on HPC clusters as well. It is customized for IMPACT and supports both 1D and 2D REMD simulations. Distinctive features are fault tolerance, the ability to use a dynamic pool of resources and to use fewer CPU cores than replicas.

Summary: As seen, there are multiple existing packages designed to perform REMD simulations. A significant number of them, however, support only a single MD engine; their design makes it difficult to substitute one simulation engine with another. The tight binding of RE methods to a particular engine presents a barrier for the development of new RE algorithms. Historically, tight integration has prevailed due to the perception that performance trumps all other features.

There are emerging examples of important biomolecular problems, however, that involve multi-state equilibria, and for which the interpretation of experiments requires scanning control variables such as temperature, ionic conditions, and pH in addition to geometrical or Hamiltonian order parameters[18]. These applications have the added challenge that sampling along the space of the order parameters needs to be statistically convergent at all points. Here, the REMD method offers the added advantage that equilibrium between simulations is enhanced through the exchanges. An illustrative example is the "problem space" associated with biocatalysis where the conformational equilibria, metal ion binding and protonation events lead to an active state that can catalyze the chemical steps of the reaction [19]. Thus, these applications require not only the elucidation of the free energy landscape of the chemical reaction itself [20], [21], but also the characterization of the probability of finding the system in the catalytically active state as a function of system variables [22].

To address these novel applications and scenarios, a flexible and efficient multi-dimensional REMD framework that can be used for both system control variables and generalized coordinates is required. Currently, there is no REMD framework capable of providing the necessary flexibility in composing the range of RE methods with MD engines as needed, while providing adequate performance.

III. RepEx: A Framework for Replica Exchange

We outline the requirements of a framework that would support the needs of scientific problems [20], [21], [22] that are not currently met. We then introduce **RepEx** – a framework designed to address these requirements – and discuss its implementation.

A. REMD Requirements

In this sub-section, we motivate the functional, performance (scalability) and usability requirements of a REMD simulation software. We identified the following functional requirements:

Generality is a requirement to maximize a range of replica-exchange methodologies as well as MD engines. A general-purpose framework should support: (i) different types of exchange parameters, (ii) multiple exchange parameters in a single REMD simulation, and (iii) multiple MD engines. A corollary of this requirement is the decoupling of advances in replica-exchange methodology from the MD engines to enable the potential for broader (greater number of REMD applications) and deeper impact (enable new research opportunities).

Execution flexibility arises from the need to decouple the number of CPU cores from the number of replicas. This is equivalent to the ability to set-up a REMD simulation with a number of replicas independent of the number of CPU cores available at a given instance of time. This requirement is motivated by reasons ranging from the job-size dependence of queue waiting times to limitations in the number of available or allocatable CPU cores on a given cluster. In addition, support for both single-core and multi-core replicas should be provided. Currently, **all** REMD frameworks require at least as many CPU cores as replicas, and even more in the case of multi-core replicas; furthermore, the number of replicas that are actively simulated is fixed and statically determined.

Synchronization: To enable a wider range of REMD simulations, support for asynchronous RE is required without loss of generality or execution flexibility.

Interoperability: Most REMD simulations are executed on clusters which vary in scheduling systems, middleware and software environment. To support community production-grade science, a REMD framework should work on multiple high-end machines, as well as small HPC clusters, while retaining functionality and performance.

The above functional requirements have to be balanced with the following performance requirements:

Scalability with the number of replicas: To obtain high sampling quality, REMD simulations should support the ability to run a large number of replicas. Furthermore, given that the number of replicas needed for a REMD simulation scales as $\approx N^d$, where d is the dimensionality (of exchange), the need to support a large number of replicas is greater when applied to multi-dimensional simulations.

Scalability with the number of CPU cores: Whereas the primary performance metric is the scalable execution of a large number of replicas, it is often the case that each replica is multi-node (in Ref. [23], each replica was 768 cores); multi-node replicas are important to simulate large physical systems. Finally, we collapse identified usability requirements into a single one:

Usability: Relative to the simulation phase, the exchange phase is significantly more complex. As a consequence, a framework for REMD should separate the logic of the exchange mechanisms from the simulation mechanisms while concealing the complexity of exchange mechanism. In addition, simulation setup should be automated as much as possible and must be fully specified by intuitive configuration files.

B. Design

To satisfy the requirements outlined in the previous sub-section, we discuss three concepts underpinning the design of RepEx:

- **Replica-Exchange Patterns:** Explicit support for synchronization patterns between simulation and exchange phases.
- **Pilot-Job based Resource Management for Replicas:** Pilot-jobs provide a multi-stage mechanism for workload execution via the use of an initial placeholder job (the "pilot") and thus permit the dynamical allocation of computational resources for replicas.
- **Flexible Execution:** The ability to execute different patterns and numbers of replicas independent of the underlying resources available, i.e., flexible spatial and temporal mapping of replicas to the allocated CPUs.

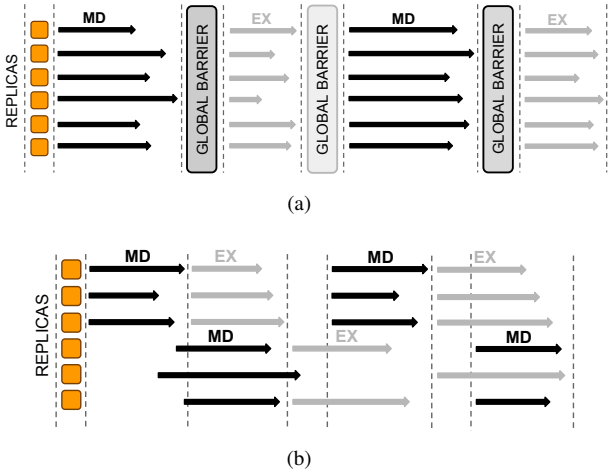


Fig. 1. Schematic representation of Replica Exchange Patterns: (a) Synchronous (b) Asynchronous. x-axis represents time. Yellow squares represent replicas, black arrows MD phase propagation and gray arrows exchange phase propagation. For both phases (MD and exchange), in synchronous pattern exists a global synchronization barrier. In this figure, for synchronous pattern, both MD and exchange are propagated concurrently but this isn't a requirement for this pattern. In asynchronous pattern there is no barrier - MD and exchange can be propagated concurrently, meaning while some replicas run MD other replicas might be running exchange.

1) Replica Exchange patterns: The RepEx framework captures the distinction between different synchronization scenarios using two RE patterns and exposes them to end-users, independent of MD simulation engine and the resources available.

Synchronous RE Pattern: The Synchronous RE pattern depicted in Figure 1 (a), corresponds to the scenario where all replicas must finish the simulation phase, before any replicas can transition to the exchange phase and vice versa. There is a global synchronization barrier after each phase, which forces the replicas arriving at the barrier early to wait for the lagging replicas. Once all the replicas arrive at the barrier, all of them

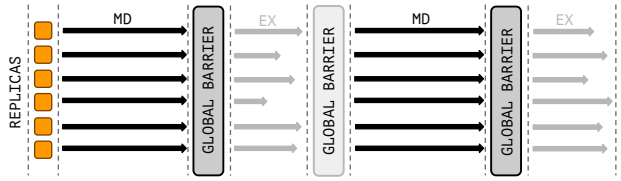


Fig. 2. Schematic representation of Execution Mode I. On the x-axis is time. Yellow squares represent replicas, black arrows MD phase propagation and gray arrows exchange phase propagation. Both MD and exchange phase for all replicas are performed concurrently. After MD and exchange phase is placed a global barrier, ensuring that all replicas enter next phase simultaneously.

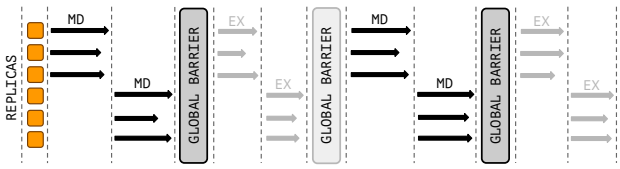


Fig. 3. Schematic representation of Execution Mode II. On the x-axis is time. Yellow squares represent replicas, black arrows MD phase propagation and gray arrows exchange phase propagation. Replicas don't propagate MD and exchange phase concurrently. Batch size for each phase is determined by the number of allocated CPUs. A global synchronization barrier is present after each phase.

transition to the next phase. This pattern is the conventional way of running REMD.

Asynchronous RE Pattern: Asynchronous RE Pattern (Figure 1 (b)), does not impose any synchronization barriers. While some replicas are in the simulation phase, others might be in the exchange phase. Based on some criterion, a subset of replicas transition to the exchange phase, while other replicas continue in the simulation phase.

2) Pilot-Job based Resource Management for Replicas: The Pilot-Job concept was originally introduced to reduce queue waiting times for workloads on distributed clusters. Pilot systems have been generalized [24] to provide a variety of capabilities, but the two most important are: management of dynamically varying resources and execution of dynamic workloads. A Pilot system supports the execution of workloads with multiple, heterogeneous and dependent tasks [9]. Integration of a pilot system in our design enables different RE patterns and number of replicas, independent of the resources available.

3) Flexible Execution:: Decoupling the workload size from the available resources requires: (i) the details of workload be kept separate from the details of resources — type, quantity and availability, and (ii) the ability to execute a workload of a given size (N replicas) independent of the specific resources available. We have seen how Pilot systems enable the former; we now discuss how they support the latter.

Depending upon the relative values of the resources available (R) to the size of simulations S, (where S is equal to the number of replicas $\times$ resource requirement of each replica), REMD simulations are executed differently. Thus, there are two Execution Modes: when $R > S$ (Execution Mode I), and when $R < S$ (Execution Mode II). Each Execution Mode can be used with any of the RE patterns.

Execution Mode I: In Execution Mode I the number of allocated CPU cores satisfies execution requirements of all replicas at a given instant of time. For example, if each replica requires a single CPU core to run, enough cores are allocated to run all replicas concurrently. Figure 2 illustrates capturing Execution Mode I in the context of a Synchronous RE pattern.

Execution Mode II: Execution Mode II supports the scenario when the number of available CPU cores is insufficient to run all replicas concurrently. The ratio of cores to replicas is a user defined variable, but typically is a term of a geometric series, e.g., $\frac{1}{2}$, $\frac{1}{4}$, etc. As a result, only a fraction of replicas can propagate simulation or exchange phase concurrently. A schematic representation of Execution Mode II is illustrated in Figure 3; for simplicity, we depict the synchronous RE Pattern, but Execution Mode II can be used with any of the two available RE Patterns.

While users can select an Execution Mode, exact execution details are determined by execution management module of RepEx. The Execution Mode abstraction hides the details of the execution, which differ based upon the relative values of R and S. The implementation of Execution Modes vary in the spatial and temporal mapping of workload (tasks) to the allocated CPUs. Specifically, they differ in:

- Order and level of concurrency for task execution
- Number of pilots used for a given simulation
- Number of concurrently used target HPC resources

Execution Mode is a subset of execution options, decoupling simulation requirements from the resource availability and enabling flexible usage of allocated HPC resources. A user should be able to switch between available Execution Modes without any refactoring. In addition to providing conceptual simplicity by hiding details of the execution, Execution Modes provides a significant practical functionality: it permits the study of systems not otherwise possible thanks to the ability to launch more replicas then there are allocatable CPU cores on a target cluster. For example, a user can assign as many cores to each replica as needed. More importantly, if a user needs to perform a simulation involving 10,000 replicas they can do so on a cluster of less than 10,000 cores, e.g., a cluster of only 1024 cores.

C. Implementation

At the core of RepEx are three modules (Figure 4): **Remote Application Modules (RAM):** responsible for creation of individual input files for replicas, reading data from simulation output files and performing exchange procedures. Unlike other RepEx modules, which are client-side, these modules execute on HPC cluster.

Application Management Modules (AMM): support *generality* by managing exchange parameters, input parameters, simulation input/output (I/O) files and file movement patterns. AMM is used to instantiate replica objects according to the *simulation input file*. AMM is MD engine specific since I/O files and arguments for each MD engine are different.

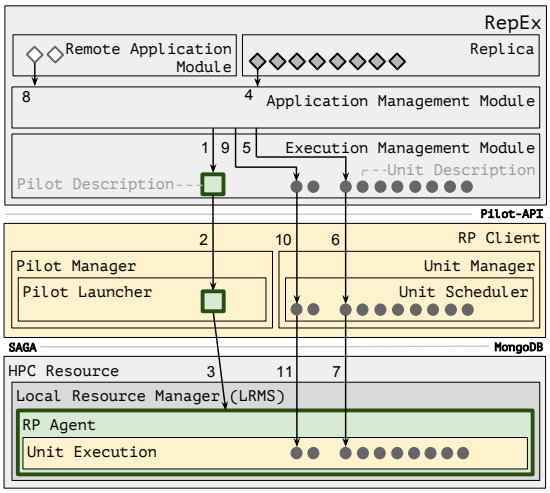


Fig. 4. RepEx framework: control flow diagram. Top gray box represents RepEx package, middle (yellow) square represents RADICAL Pilot and bottom square represents a target resource. The numbered arrows represent control flow. First a Pilot description (step 1) is created, which is submitted to RPs Pilot Manager (step 2). Pilot Manager launches a Pilot instance (RP Agent or simply a task container) on a target resource (step 3). MD phase begins at step 4: replica objects (dark gray rhombi) are instantiated, which are then translated into Unit Descriptions (grey circles) at step 5. Units are then submitted to RPs Unit Manager (step 6), which in turn submits them for execution to RPs Agent (step 7). Exchange phase begins at step 8. Remote Application Modules (white rhombi) are instantiated. Similarly to replica objects, they are transated into Unit Descriptions (step 9). Unit Descriptions are submitted to RPs Unit Manager (step 10), which submits units for execution to RPs Agent (step 11).

Execution Management Modules (EMM): enables a separation of execution details, namely resource management and workload configuration, from the simulation. Most of the resource management calls (RP API) are performed in EMM. Encapsulation of synchronization routines by EMM, enables a developer to specify synchronous or asynchronous RE by a single EMM.

D. Validation

3D-REMD was performed using order parameters of temperature, and umbrella sampling in the ϕ and ψ torsion angles (as shown in Figure 5). In the temperature (T) dimension, six windows were chosen from 273K to 373K by geometrical progression. In both umbrella (U) dimensions, eight windows were selected uniformly between 0° and 360°, where each window corresponds to a harmonic restraint centered on it with a force constant of 0.02 kcal·mol^{-1}·degree^{-2}. The total number of replicas is therefore 6×8×8=384. Each replica was previously equilibrated for $>$ 1 ns. In the production run, we set the exchange attempt interval (cycle) to be 20000 steps (20 ps) and in a 15-hour run with 400 cores (25 nodes) on Stampede [25], the simulation finished 90 cycles (1.8 ns). The acceptance ratios of exchange attempts are approximately 3% for T dimension and 25% for U dimensions. Free energy profiles were then generated from the last 1 ns of production data using the maximum likelihood approach implemented in the vFEP package [26], [27]. Free energy

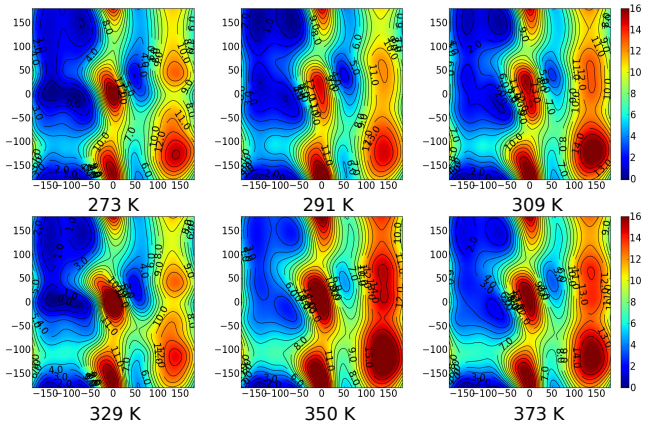


Fig. 5. Free energy profile of alanine dipeptide backbone torsion at six different temperatures. In all six subplots, the x and y-axes correspond to ϕ and ψ torsion angles, respectively. The range of energies is from 0 kcal/mol to 16 kcal/mol while each level in the contour corresponds to a 1 kcal/mol increment.

profile of the alanine dipeptide system along the ϕ-ψ angles has been studied extensively by theoretical and experimental methods. Figure 5 demonstrates identical free energy profiles at 300K, compared with other recent computational studies [28], [29]. The enthalpy contribution of $\alpha R/c7_{eq}$ transition has not been reported in any computational studies. Nevertheless, the temperature dependency of free energy profiles in Figure 5 has been utilized to estimate this enthalpy contribution as 1.2 kcal/mole, comparable to the known experimental reported value 1.1 kcal/mole. [30]

IV. Experiments

Having validated both the design and implementation of RepEx, in this section we discuss experiments performed to demonstrate the unique functional capabilities of RepEx and characterize its performance. For all experiments, we measure and plot the average REMD simulation cycle time, over 4 simulation cycles. Physical models used for the experiments was alanine dipeptide (Ace-Ala-Nme) solvated by water molecules was used with the Synchronous RE pattern. The Extreme Science and Engineering Discovery Environment [25] (XSEDE) allocated systems – Stampede and SuperMIC were used.

Simulation cycle time is defined as:

$$T_c = T_{MD} + T_{EX} + T_{data} + T_{RepEx-over} + T_{RP-over} \quad (1)$$

where:

- T_{MD} - MD simulation time. Time to perform a certain number of simulation time-steps
- T_{EX} - Exchange time. Time for calculations required to perform an exchange phase
- T_{data} - Data time. Time to perform data movement, mostly within remote resource. For example, moving Amber's .mdinfo files to a "staging area" after the simulation phase, to make them accessible by the tasks of the exchange phase. I/O time is not a part of data time.

- $T_{RepEx-over}$ - RepEx overhead. Time to prepare tasks for execution and time to perform local RepEx method calls
- $T_{RP-over}$ - RP overhead. Time required for task launching on a cluster and time for internal RP communication

For M-REMD simulations, T_c is comprised of the 1D cycle time for each dimension, since simulations are performed only in one dimension at any given instant of time.

We calculate weak scaling efficiency as:

$$E_w = \frac{T_1}{T_N} \times 100\% \quad (2)$$

where:

- T_1 - amount of time to complete a single work unit (simulation cycle with 8 replicas) using a single processing unit (8 CPU cores)
- T_N - time to complete N work units (simulation cycle with N replicas) using N processing units (N*8 CPU cores)

We calculate strong scaling efficiency as:

$$E_s = \frac{T_1}{(M/N_{min}) \times T_N} \times 100\% \quad (3)$$

where:

- T_1 - amount of time to complete all work (simulation cycle with N replicas) using a single processing unit (8 CPU cores)
- T_N - amount of time to complete all work (simulation cycle with N replicas) using M processing units (M*8 CPU cores)
- N_{min} - a single processing unit (8 CPU cores)
- M - number of used CPU cores

Results obtained in Section IV can be reproduced by following instructions at [31]. All experiments were performed with RP version 0.35. Though the lastest version of RP is 0.40.1 and is capable of substantially better performance due to various optimizations, difference in versions only alter the RP overhead timings (as well as data timings) presented in this section, and will have minimal impact on the overall performance characterization of RepEx.

A. Characterization of Overheads

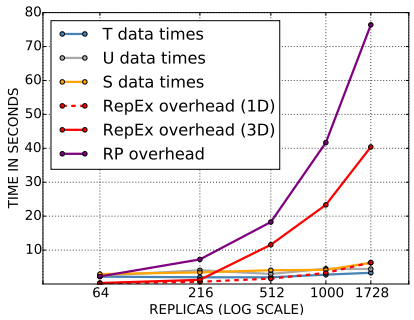


Fig. 6. Characterization of overheads: Data times, RepEx overhead and RP overhead.

As a result of design decisions we have made there are three factors contributing to the simulation overhead: data time, RepEx overhead and RP overhead. In this subsection, we summarize how these factors influence T_c. Figure 6 presents the values of data times, RepEx overheads and RP overheads for simulation runs involving 64, 216, 512, 1000 and 1728 replicas on SuperMIC. For all runs, we use Execution Mode I with a single CPU core per replica.

Values of data times depend on the exchange type. As depicted in Figure 6, data times for temperature exchange are shorter than for umbrella exchange and salt concentration. For all replica counts, data times are relatively small ($\leq$6.3 seconds. This is because the majority of the transfers occur within the cluster. Consequently, data times change as a function of the target system, since the largest contributing factor is the performance of the parallel file system.

RepEx overhead depends on the total number of replicas and on the exchnage type. For all 1D simulations, the overhead of RepEx is nearly identical, since the number of operations to perform task preparation is very similar. RepEx overhead times for 3D simulations are longer, since there is more data associated with each replica and more computations are performed during task preparation.

RP overhead depends only on the number of replicas (tasks) launched concurrently. As we can see in Figure 6, RP overhead is proportional to the number of replicas.

B. Performance Characterization of 1D-REMD

In this subsection, we characterize performance of 1D REMD simulations with RepEx. For each of the three available 1D-REMD simulations, temperature exchange (T-REMD), umbrella exchange (U-REMD) and salt concentration exchange (S-REMD), we measure average cycle times. We perform simulation runs involving 64, 216, 512, 1000 and 1728 replicas in Execution Mode I. All runs are conducted with a single CPU core per replica and **sander** as the Amber executable. We use alanine dipeptide solvated by water molecules comprising a total of 2881 atoms and perform 6000 simulation time-steps between exchanges. We perform all runs on SuperMIC supercomputer [25]. Results of these experiments are presented in Figure 7.

As we can see, for all three exchange types, the time to perform 6000 time-steps is nearly identical, as evidenced by the almost similar average heights of dark green bars in Figure 7 (139.6 seconds).

Next we discuss exchange timings for different exchange parameters, as seen in the lower panel of Figure 7. Timings for temperature and umbrella exchange are similar and have a nearly linear growth rate. For both exchange types, we use a single MPI task to perform an exchange. In case of U-REMD we have implemented a single point energy calculation internally. Despite the fact that U-REMD exchange is more involved, we do not see a significant difference in exchange timings between U-REMD and T-REMD.

Due to the mathematical complexity, the single point energy calculation for S-REMD is calculated using Amber for each

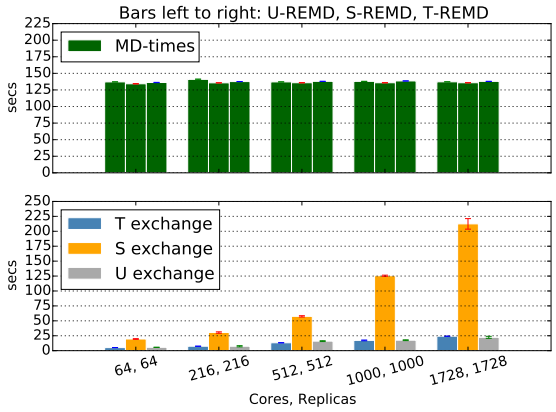


Fig. 7. 1D REMD experiments with RepEx: weak scaling. Decomposition of average simulation cycle times T_c (in seconds) into MD simulation time and exchange time for umbrella sampling, salt concentration and temperature exchange. For all simulation runs the number of replicas is equal to the number of CPU cores (e.g., 1 core per replica) and both vary from 64 to 1728. All simulation runs are performed on SuperMIC supercomputer.

replica in each state. This implies that for each replica, an additional task is required. Since we are using Amber's group files, this task requires at least as many CPU cores as there are potential exchange partners for each replica. Consequently, the exchange times for S-REMD are substantially longer, but nonetheless have a nearly linear growth rate.

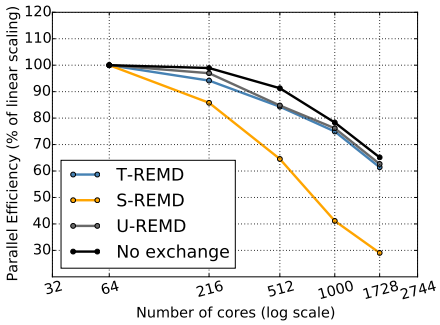


Fig. 8. Parallel Efficiency (% of linear scaling) for Temperature Exchange REMD (1D), Salt Concentration REMD (1D) and Umbrella Sampling REMD (1D) using Amber MD engine on SuperMIC supercomputer.

The parallel efficiency results for the 1D-REMD simulations are presented in Figure 8. We calculate parallel efficiency for the weak scaling scenario and use average cycle time for simulations with 64 cores as starting point, e.g., 100% efficiency. We also present efficiency results for simulations without an exchange phase (black line). This quantifies the influence of exchanges on the efficiency of 1D simulations. Since all tasks have overheads associated with them, we observe a decrease in efficiency even if there is no exchange. Efficiency values for T-REMD and U-REMD are similar and demonstrate linear behavior; efficiency values for S-REMD is lower. This is caused by specifics of the exchange phase, discussed earlier.

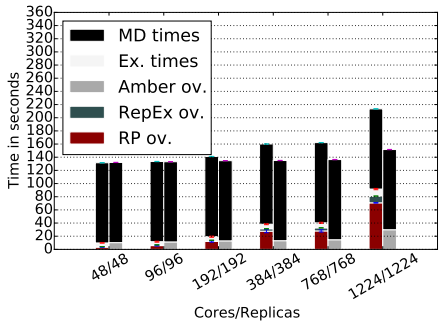


Fig. 9. Decomposition of average simulation cycle times T_c for Temperature Exchange REMD using RepEx and stand-alone Amber.

We compare average simulation cycle times T_c for 1D T-REMD using RepEx and Amber. As shown in Figure 9 Amber cycle times are decomposed into two components - MD (simulation) time and Amber overhead. We calculate Amber overhead as a difference of total cycle time and MD time. Cycle times for RepEx are decomposed into four components - MD time, Exchange time, RepEx overhead and RP overhead.

For up to 192 replicas, RepEx demonstrates timings comparable with Amber; for runs involving more replicas, RepEx cycle times are increasingly higher. For simulations with 1224 replicas, the difference between RepEx and Amber cycle times is nearly 60 seconds. As we can see from Figure 9, the largest contributing factor for this difference is the RP overhead, which is nearly 70 seconds.

C. T-REMD with NAMD engine

To demonstrate RepEx's ability to use different MD engines we perform weak scaling experiments using T-REMD with NAMD. On SuperMIC, we use NAMD-2.10 and perform a total of 4000 time-steps between exchanges. We perform runs with 64, 216, 512, 1000 and 1728 replicas. For each replica we use a single CPU core and allocate enough cores to run all replicas concurrently (Execution Mode I). The results of these experiments are provided in Figure 10.

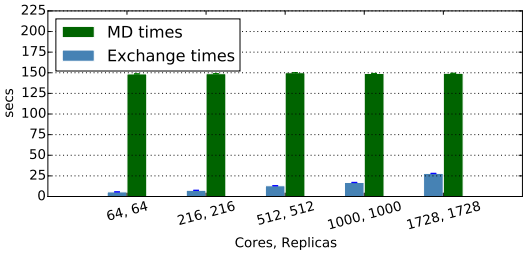


Fig. 10. Experiments with NAMD engine. Decomposition of average simulation cycle times T_c (in seconds) into MD simulation time and Exchange time for weak scaling scenario. Experiments are performed on SuperMIC supercomputer, using T-REMD and 1 core per replica.

As expected, MD times for all cores/replicas pairs are nearly equal. The growth rate for exchange times is linear.

D. M-REMD performance characterization

To characterize M-REMD performance, we perform weak and strong scaling experiments with TSU-REMD on Stampede. In both cases we perform 6000 simulation time-steps between exchanges.

Weak Scaling: For weak scaling experiments we use Execution mode I. The number of replicas in each dimension is kept equal; thus, as the number of replicas in one dimension varies from 4, 6, 8, 10 and 12, the total number of replicas equal to 64, 216, 512, 1000 and 1728, respectively. We use Amber 12.0, and **sander** as Amber executable. For each replica we use 1 CPU core. The results are presented in Figure 11.

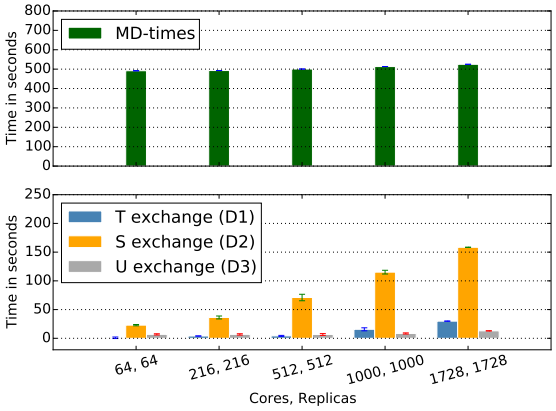


Fig. 11. M-REMD experiments with RepEx - weak scaling. TSU-REMD on Stampede using Amber MD engine. For all runs the number of replicas is equal to the number of CPU cores (both vary from 64 to 1728) and are used single-core replicas. In figure is shown decomposition of average simulation cycle times T_c (in seconds) into MD and exchange times.

In all cases MD times are nearly identical: ~ 495.0 seconds. This is as expected, since variation in the number of replicas should not affect MD time.

We observe a nearly linear scaling for exchange timings in all three dimensions. While temperature and umbrella exchange timings are very similar, salt concentration exchange takes substantially more time. As mentioned in Subsection IV-B, for this exchange type we use Amber to perform a single point energy calculations, which results in doubling of tasks and higher computational requirements.

Parallel Efficiency results are presented in Figure 13(a). We observe a rapid decrease in efficiency with increase in the number of cores. This can be explained by the influence of performance for salt concentration. Despite that, for all core counts, efficiency is above 50%.

Strong Scaling: We fix the number of replicas at 1728 with 12 replicas in each dimension and vary the number of cores from 112 to 1728. Experiments are performed using Execution Mode II and for each replica is used a single CPU core. Results of these experiments are provided in Figure 12.

As illustrated in Figure 12, decrease in MD time is proportional to the number of cores: the doubling of the number of cores, results in decrease of MD time by nearly a half.

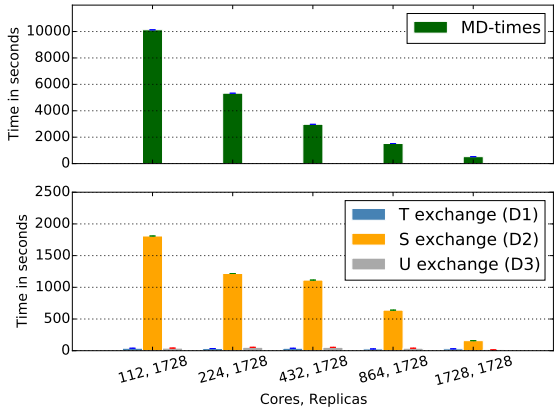
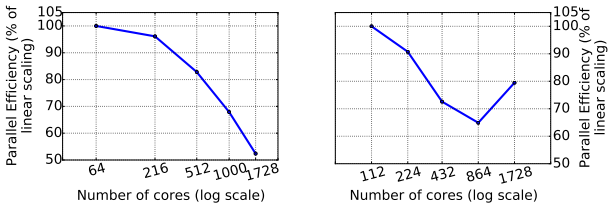


Fig. 12. M-REMD experiments with RepEx - strong scaling. TSU-REMD on Stampede using Amber. Number of replicas is fixed at 1728 (1 core per replica), the number of cores is increased from 112 to 1728. In figure is shown decomposition of average simulation cycle times T_c (in seconds) into MD and exchange times.

Exchange times in temperature exchange and umbrella exchange dimensions are nearly equal for all runs. This highlights the fact, that implementation of temperature exchange and umbrella exchange are similar. Due to task launching delay and grouping of replicas by parameter values in each dimension, the exchanges largely overlap with MD. As a result, tasks which have finished simulation phase sooner can perform certain exchange procedures before an exchange is finalized. In comparison, salt concentration exchange times are significantly higher: at 112 cores, salt exchange time takes nearly 1800 seconds.



(a) (b)

Fig. 13. Parallel Efficiency (% of linear scaling) for TSU-REMD on Stampede using Amber MD engine - (a) weak scaling, (b) strong scaling.

As we can see from Figure 13(b), the Parallel Efficiency for strong scaling scenario are non-linear. The Parallel Efficiency decreases up to 864 cores, but for 1728 cores we observe an increase in Parallel Efficiency. This is caused by the MPI task scheduling issue of RP, which will be addressed in the next release of RepEx.

E. REMD with Multi-core Replicas

To demonstrate RepEx's capability to execute replicas using multiple CPUs, we use solvated alanine dipeptide with 64366 atoms. We perform a total of 20000 time-steps for simulation phase. We use Stampede and pmemd.MPI as Amber (12.0)

executable for multi-core replicas. Since pmemd.MPI cannot be run on 1 CPU core, for 1 core replicas we use sander.

We perform weak scaling experiments using TUU-REMD with one temperature dimension and two umbrella dimensions. We run simulations with fixed number of replicas and change the number of CPU cores per replica. In all cases, we use 216 replicas and vary the number of cores per replica from 1 to 64. Results of these runs are shown in Figure 14.

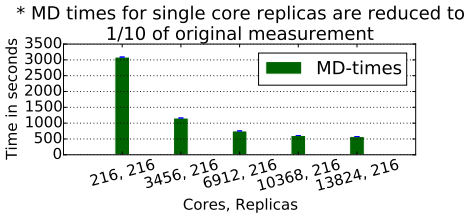


Fig. 14. Weak scaling for multi-core replicas: MD times for TUU-REMD with Amber engine. Experiments are performed on Stampede. Number of replicas is fixed at 216, the number of cores per replica is increased from 1 to 64.

MD times when using a single core per replica are significantly higher than when using multiple cores. We attribute this difference to the use of a highly efficient pmemd.MPI code when running individual replicas over multi-cores/multi-nodes. Increase in the number of cores per replica does not demonstrate a linear behavior. This is not a limitation of the RepEx framework but attributable to the size of the alanine dipeptide, which although relatively larger than the earlier physical system, is small in absolute terms and thus makes it difficult to gain significant performance improvements by using more CPUs.

V. Discussion and Conclusion

RepEx was designed to address the functional, performance and usability requirements outlined in Section III-A. In Section IV, we demonstrated capabilities of RepEx and characterized its performance for 1D and 3D REMD simulations. We saw the range of exchange parameters that it supports and the flexibility in their ordering (e.g., TUU versus TSU). Furthermore we saw RepEx supporting both Amber and NAMD with minimal conceptual or implementation changes. Last but not least, we saw the ability to utilize different RE patterns and different Execution Modes, thus providing a decoupling between exchange parameters (T/U/S), dimensionality and algorithm (sync. vs. async.) with resource management, as well as a decoupling of execution details. As such, it is accurate to say that RepEx satisfies the functional and usability requirements.

The two core design features of RepEx are the separation of MD simulation engine from the implementation of RE algorithm and the use of a pilot-based runtime which separates the algorithm and workload management from the resource management and runtime complexity.

As a consequence of the former, the integration of new MD simulation engines is significantly simplified and facilitates the reuse of RE patterns and Execution Modes. We believe that this also lowers the barrier for development and testing of new REMD algorithms. As a consequence of the second design principle, our implementation decouples execution specifics from the REMD algorithm and enables users to choose from the multiple execution options. Collectively, this allowed us to introduce the concept of Replica Exchange patterns, which can be used interchangeably within RepEx. To the best of our knowledge none of the currently available REMD implementations have this capability.

In Table I, we have summarized the most important features of six existing packages used for REMD, some of which are used by communities of hundreds, if not thousands of users. In this table, we have included three popular MD simulation engines, namely Amber, LAMMPS and Gromacs that have been extended to provide RE capabilities and four REMD packages that have been designed to be external to MD engines. Some of these were reviewed in Section II.

As can be seen from Table I, a majority of the packages are designed to address a subset of features we identified as necessary to be flexible and general purpose. Many packages have eschewed generality for performance. For example, Charm++/NAMD MCA package can utilize O(100,000) cores but doesn't provide flexible resource utilization nor asynchronous exchange capabilities. On the other hand, VCG RE package is one of the few packages, which supports asynchronous RE but it has limited scalability (both in the number of replicas and CPU cores) and is tightly coupled to IMPACT which is not an open source MD engine. Similar to most other existing solutions, both VCG and Charm++/NAMD are limited in the number of exchange parameters as well as in flexibility in the ordering of exchange parameters.

Clearly a balance between performance and functional requirements needs to be maintained. On the evidence of Table I we believe that RepEx provides an optimal balance. As evidenced by the rigorous requirements analysis, design and implementation considerations, RepEx embodies the sound systems engineering principles along with software engineering practices. RepEx has been used for algorithmically innovative molecular science simulations [15]. For example, when simulating a non-enzymatic transesterification reaction using QM/MM, preliminary results with asynchronous RE pattern, demonstrate higher utilization values in comparison with synchronous RE pattern. Utilization is defined as a percentage of ideal resource utilization, when CPUs are used only to perform MD. Importantly, in the case of asynchronous pattern, sampling quality is not compromised due to execution of replicas in groups, in each of the three dimensions.

RepEx can easily be extended to support the use of novel and emerging platforms. For example, support for GPUs is already available on Stampede and will be extended to other machines, such as Blue Waters [32].

There are several planned extensions to the RepEx framework: Single point energy calculations for salt concentration exchange will be implemented. Additional exchange parameters can be added to support other types of multi-dimensional

	Amber	Gromacs	LAMMPS	VCG async	CHARMM	Charm++/NAMD MCA	RepEx
Max replicas	∼2744	∼900	100	240	4096	2048	3584
Max CPU cores	∼5488	∼150000	76800	1920	131072	524288	13824
Fault tolerance	n/a	n/a	n/a	medium	n/a	n/a	medium
MD engines	Amber	Gromacs	LAMMPS	IMPACT	CHARMM	NAMD	Amber, NAMD
RE patterns	sync	sync	sync	sync, async	sync	sync	sync, async
Nr. Exec. Modes	1	1	1	2	1	1	2
Nr. dims	2	2	2	2	2	2	3
Exchange params	3	2	2	2	2	2	3

TABLE I
COMPARISON OF MOLECULAR SIMULATION SOFTWARE PACKAGES WITH INTEGRATED REMD CAPABILITY. WE CHARACTERIZE EACH OF THE SEVEN PACKAGES BASED ON 8 FEATURES. FOR EACH FEATURE WE PROVIDE A NUMERICAL VALUE OR A NAME CORRESPONDING TO THAT FEATURE.

REMD simulations (for example pH exchange). Support for other MD simulation engines will be added.

Software and Data: RepEx is available under MIT open-source license. RepEx source code is available at [8]. RepEx documentation is provided at [33]. Results obtained in Section IV can be reproduced by following instructions at [31].

ACKNOWLEDGMENT

This work is supported by NSF CHE-1265788. We acknowledge allocation TG-MCB090174 for computing time on XSEDE allocated resources. We acknowledge NSF ACI 1516469 and ACI 1515572 (Cheatham) for time on Blue Waters machine.

REFERENCES

[1] R. H. Swendsen and J.-S. Wang, "Replica monte carlo simulation of spin-glasses," *Physical Review Letters*, vol. 57, no. 21, p. 2607, 1986.
[2] Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics method for protein folding," *Chemical physics letters*, vol. 314, pp. 141–151, 1999.
[3] H. Fukunishi *et al.*, "On the hamiltonian replica exchange method for efficient sampling of biomolecular systems: application to protein structure prediction," *The Journal of chemical physics*, vol. 116, pp. 9058–9067, 2002.
[4] Y. Meng and A. E. Roitberg, "Constant ph replica exchange molecular dynamics in biomolecules using a discrete protonation model," *J. Chem. Theory Comput.*, vol. 6, pp. 1401–1412, 2010.
[5] R. Salomon-Ferrer *et al.*, "An overview of the amber biomolecular simulation package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 3, pp. 198–210, 2013. [Online]. Available: http://dx.doi.org/10.1002/wcms.1121
[6] J. C. Phillips *et al.*, "Scalable molecular dynamics with namd," *J. Comput. Chem.*, vol. 26, pp. 1781–1802, 2005.
[7] S. Pronk *et al.*, "Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics*, vol. 29, pp. 845–854, 2013.
[8] "Repex on github," https://github.com/radical-cybertools/radical.repex, Nov 2015.
[9] A. Merzky *et al.*, "RADICAL-Pilot: Scalable Execution of Heterogeneous and Dynamic Workloads on Supercomputers," 2015, (under review) http://arxiv.org/abs/1512.08194.
[10] W. Jiang *et al.*, "Calculation of free energy landscape in multi-dimensions with Hamiltonian-exchange umbrella sampling on petascale supercomputer," *J. Chem. Theory Comput.*, vol. 8, pp. 4672–4680, 2012.
[11] B. R. Brooks *et al.*, "Charmm: the biomolecular simulation program," *J. Comput. Chem.*, vol. 30, pp. 1545–1614, 2009.
[12] W. Jiang *et al.*, "Generalized scalable multiple copy algorithms for molecular dynamics simulations in namd," *Computer physics communications*, vol. 185, pp. 908–916, 2014.
[13] W. Shalongo *et al.*, "Distribution of helicity within the model peptide acetyl (aaqaa) 3amide," *Journal of the American Chemical Society*, vol. 116, pp. 8288–8293, 1994.
[14] B. K. Radak *et al.*, "A framework for flexible and scalable replica-exchange on production distributed ci," ser. XSEDE '13, 2013, pp. 26:1–26:8.
[15] B. K. Radak *et al.*, "Characterization of the Three-Dimensional Free Energy Manifold for the Uracil Ribonucleoside from Asynchronous Replica Exchange Simulations," *Journal of Chemical Theory and Computation*, vol. 11, pp. 373–377, 2015, http://dx.doi.org/10.1021/ct500776j. [Online]. Available: http://dx.doi.org/10.1021/ct500776j
[16] J. L. Banks *et al.*, "Integrated modeling program, applied chemical theory (impact)," *J. Comput. Chem.*, vol. 26, pp. 1752–1780, 2005.
[17] J. Xia *et al.*, "Large-scale asynchronous and distributed multidimensional replica exchange molecular simulations and efficiency analysis," *J. Comput. Chem.*, vol. 36, pp. 1772–1785, 2015.
[18] C. Bergonzo *et al.*, "Multidimensional replica exchange molecular dynamics yields a converged ensemble of an RNA tetranucleotide," *J. Chem. Theory Comput.*, vol. 10, pp. 492–499, 2014.
[19] M. T. Panteva *et al.*, *Multiscale Methods for Computational RNA Enzymology*. Elsevier, 2015, ch. 14.
[20] B. Ensing *et al.*, "Metadynamics as a tool for exploring free energy landscapes of chemical reactions," *Acc. Chem. Res.*, vol. 39, pp. 73–81, 2006.
[21] E. Vanden-Eijnden, "Some recent techniques for free energy calculations," *J. Comput. Chem.*, vol. 30, pp. 1737–1747, 2009. [Online]. Available: http://dx.doi.org/10.1002/jcc.21332
[22] T. Dissanayake *et al.*, "Interpretation of pH-Activity Profiles for Acid-Base Catalysis from Molecular Simulations," *Biochemistry*, vol. 54, pp. 1307–1313, 2015.
[23] J. B. Swadling *et al.*, "Structure, dynamics, and function of the hammerhead ribozyme in bulk water and at a clay mineral surface from replica exchange molecular dynamics," *Langmuir*, vol. 31, pp. 2493–2501, 2015.
[24] M. Turilli *et al.*, "A Comprehensive Perspective on Pilot-Jobs," 2015, http://arxiv.org/abs/1508.04180.
[25] "Extreme science and engineering discovery environment," https://www.xsede.org/resources/overview, Nov 2015.
[26] T.-S. Lee *et al.*, "A new maximum likelihood approach for free energy profile construction from molecular simulations," *J. Chem. Theory Comput.*, vol. 9, pp. 153–164, 2013.
[27] T.-S. Lee *et al.*, "Roadmaps through free energy landscapes calculated using the multidimensional vFEP approach," *J. Chem. Theory Comput.*, vol. 10, pp. 24–34, 2014.
[28] W. Sinko *et al.*, "Population Based Reweighting of Scaled Molecular Dynamics," *J. Phys. Chem. B*, vol. 117, pp. 12 759–12 768, 2013.
[29] X. Peng *et al.*, "Free energy simulations with the AMoeba polarizable force field and metadynamics on GPU platform." *J. Comput. Chem.*, vol. 37, pp. 614–622, Mar. 2016.
[30] T. Takekiyo *et al.*, "Temperature and PressureEffects on Conformational Equilibria of Alanine Dipeptide in Aqueous Solution," *Biopolymers*, vol. 73, pp. 283–290, 2004.
[31] "Repex experiments," https://github.com/radical-cybertools/radical.repex/blob/master/EXPERIMENTS.md, Nov 2015.
[32] "Blue waters supercomputer at the national center for supercomputing applications," https://bluewaters.ncsa.illinois.edu/blue-waters, Nov 2015.
[33] "Repex documentation," http://repex.readthedocs.org/en/master/, Nov 2015.