A Framework for Flexible and Scalable Replica-Exchange on Production Distributed CI

Brian K. Radak*
radakb@biomaps.rutgers.edu

Melissa Romanus†
melissa@cac.rutgers.edu

Emilio Gallicchio*
emilio@biomaps.rutgers.edu

Tai-Sung Lee*
taisung@biomaps.rutgers.edu

Ole Weidner†
ole.weidner@rutgers.edu

Nan-Jie Deng*
nanjie.deng@gmail.com

Peng He*
hepeng.yanglu@gmail.com

Wei Dai*
daiwei@physics.rutgers.edu

Darrin M. York*
york@biomaps.rutgers.edu

Ronald M. Levy*
ronlevy@lutece.rutgers.edu

Shantenu Jha†
shantenu.jha@rutgers.edu

ABSTRACT

Replica exchange represents a powerful class of algorithms used for enhanced configurational and energetic sampling in a range of physical systems. Computationally it represents a type of application with multiple scales of communication. At a fine-grained level there is often communication with a replica, typically an MPI process. At a coarse-grained level, the replicas communicate with other replicas – both temporally as well as in amount of data exchanged. This paper outlines a novel framework developed to support the flexible execution of large-scale replica exchange. The framework is flexible in the sense that it supports different coupling schemes between replicas and is agnostic to the specific underlying simulation – classical or quantum, serial or parallel simulation. The scalability of the framework is assessed using standard simulation benchmarks. In spite of the increasing communication and coordination requirements as a function of the number of replicas, our framework supports the execution of hundreds replicas without significant overhead. Although there are several specific aspects that will benefit from further optimization, a first working prototype has the ability to fundamentally change the scale of replica exchange simulations possible on production distributed cyberinfrastructure such as XSEDE, as well as support novel usage modes. This paper also represents the release of the framework to the broader biophysical simulation community and provides details on its usage.

*BioMaPS Institute for Quantitative Biology and Department of Chemistry and Chemical Biology, Rutgers University, Piscataway, NJ 08854

†Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854

XSEDE '13, July 22-25, 2013, San Diego, USA.

Categories and Subject Descriptors: H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

General Terms: Experience, Technology

Keywords: HPC, Distributed Computing, IMPACT, AMBER, MD, Large Scale, XSEDE resources

1. INTRODUCTION

Replica exchange (RE)[1, 5, 11, 18] denotes a family of advanced conformational sampling algorithms widely employed in molecular simulations of chemical and biological systems. The key aspect of RE algorithms is that multiple importance sampling simulations (typically molecular dynamics trajectories) are executed in parallel, making use of intermittently exchanged information. It has been shown in many contexts[2, 7, 14, 17, 19, 23, 24] that such exchanges can greatly enhance sampling efficiency relative to non-communicating simulations. For physical applications the information exchanged takes the form of thermodynamic state definitions. Strict microscopic reversibility requirements then ensure that each simulation maintains the correct importance sampling weights in each state. The simulations can therefore be viewed as identical "replicas" in both the target configurational and state spaces. This imposition of uniform sampling across replicas makes it easier to diagnose locally insufficient sampling and assess the accuracy of (ultimately desired) global properties.

The advantages of RE approaches are partly counterbalanced by the additional complexities inherent in instating and maintaining a communication framework amongst replicas. This requirement has historically discouraged large scale deployment of RE on XSEDE. In our view this is not necessarily due to lack of suitable hardware and networking resources, but rather to the lack of suitable software technologies capable of efficiently harnessing this latent computational power in a convenient and practical way.

Although the coupling between replicas is conceptually simple, RE applications represent the general challenge of scaling many loosely-coupled simulations. The difficulty

arises from the fact that, although most communication is internal to the individual replicas (generally large-scale MPI-style simulations), there exists a less frequent and comparably slow communication mode that increases in complexity, importance, and cost as the number of replicas increases. Providing an approach that works across multiple replica size, number, and coupling schemes presents both a software and conceptual challenge. Replica states typically change with high frequency and often with significant dependence on the outcome of an exchange. Furthermore, resource assignment and scheduling is typically required after an exchange. Thus there is a need for finer grained resource management than is typically provided by the batch-queue level resource management.

To address some of these issues, there has been recent progress towards asynchronous RE formulations based on a pilot-job framework that can support dynamic and scalable resource execution and management, thereby providing the basis for a flexible and scalable formulation of RE on XSEDE. These developments are the subject of this report.

Many applications areas, such as the ones illustrated here, benefit greatly from multi-dimensional RE implementations employing hundreds to thousands of replicas. However, current implementations of RE methods in use by the computational chemistry community are severely limited in terms of scalability and control when many replicas are involved. In conventional implementations of RE[12] simulations progress in unison and exchanges occur in a synchronous manner whereby all replicas must reach a pre-determined state (typically the completion of a certain number of sampling steps) before exchanges are performed. This synchronous approach has a number of severe limitations. First, sufficient dedicated computational resources must be secured for all of the replicas before the simulation can begin execution. Second, the computational resources must be statically maintained until the simulation is completed. Third, failure of any replica simulation typically causes the whole calculation to abort. The reliance on a static pool of computational resources and zero fault tolerance prevents the synchronous RE approach from being a feasible solution for large scale RE deployments.

As earlier prototypical implementations of asynchronous RE (aRE) algorithms[9] have illustrated, the RE method itself does not impose the restriction that exchanges occur synchronously across all threads and that all of the replicas run at the same time. The basic idea of aRE is to allow pairs of replicas to perform exchanges independently from the other replicas. This paradigm lends itself naturally to implementations based on the pilot-job framework described below, which, while already extensively employed to automate the asynchronous execution of independent ensembles, can be effectively employed to manage inter-communicating replicas.

In this work we present ASyncRE, an aRE software utility built on the BigJob/SAGA distributed computing environment on XSEDE capable of scaling to arbitrarily large numbers of replicas. Illustrative applications of the software to large-scale multi-dimensional RE problems are presented and analyzed.

2. SCIENTIFIC PROBLEM

The conformational sampling problem in molecular simulations can be described as the problem of efficiently drawing samples x from the ensemble distribution of the chemical system:

$$p(x;\beta,\theta) = \frac{\exp[-U(x;\beta,\theta)]}{Z}\,, \tag{1}$$

where x represents the configuration of the system (atomic coordinates, volume, composition, etc.), Z denotes the configurational partition function, $\beta = 1/k_BT$ is the inverse (absolute) temperature, and $U(x;\beta,\theta) = \beta V(x;\theta)$ is the dimensionless effective potential energy of the system. $U(x;\beta,\theta)$ depends linearly on the inverse temperature and, depending on the ensemble (canonical, isobaric, grand canonical, etc.) also on thermodynamic parameters such as pressure and chemical potential. The effective potential energy also depends on potential energy parameters (*e.g.* the atomic charges or parameters of any biasing potentials), here collectively denoted as θ.

In conventional MD-based sampling implementations, x evolves in time with fixed model parameters. Slow convergence is the main issue of concern with methods of this kind as it is notoriously difficult to achieve equilibration within the time scale afforded by even the fastest supercomputers. Fortunately, great progress has been achieved in recent years with the development of generalized ensemble formulations, which now allow modeling of complex biochemical processes with unprecedented fidelity. This sampling enhancement is achieved via a random walk, not only in conformational space, but also in parameter space.

Amongst generalized ensemble sampling algorithms, replica exchange molecular dynamics (REMD) remains one of the most convenient and effective due to its broad applicability and amenability to nearly all parallel computing architectures. The core concept of REMD is that multiple replicas traveling in conformational space are additionally enabled to move in parameter space by exchanging state parameters amongst each other. The first and most widely employed RE scheme is temperature REMD (T-REMD) in which inverse temperatures β_i are exchanged. T-REMD accelerates interconversions between stable states of the system by letting replicas temporarily visit high temperatures where barrier crossings are more rapid. However, REMD schemes can involve exchanges of any number of state parameters. For example, schemes involving the exchange of more than one parameter are often referred to as *multi-dimensional* REMD schemes.[18]

Any REMD scheme is required to satisfy microscopic reversibility. In the present context this is ensured by structuring exchanges so that permutations of state parameters assigned to replicas are distributed according to the discrete unnormalized probability distribution:[4]

$$p(\{j_M\}) = \exp\left[-\sum_{i=1}^{M} U(x_i;\beta_{j_i},\theta_{j_i})\right] \tag{2}$$

where $\{j_M\}$ denotes one of the $M!$ permutations of a vector of M states, x_i is the atomic configuration of replica i, and β_{j_i} and θ_{j_i} are the inverse temperature and potential parameters assigned to replica i.

In this work we analyze four science application areas that benefit from the application of multi-dimensional REMD protocols. The first is the modeling of binding between a guest molecule and a host to form a supramolecular complex in solution[8]. In this case the dimensionless energy

is[6]:

$$U(x;\beta,\lambda) = \beta\left[V_0(x) + \lambda u(x)\right] \quad (3)$$

where $V_0(x)$ is the potential energy when the host and the guest are separated, $u(x)$ is the interaction energy between the host and the guest, and λ is an alchemical parameter. This system is modeled by multi-dimensional RE with λ and β as exchange parameters. The purpose of the sampling along λ is to enhance mixing of conformations along the alchemical pathway while high temperatures enhance sampling at each alchemical stage.

The second application studied by multi-dimensional RE is the folding of a mini-protein. In this case the dimensionless potential energy function is

$$U(x;\beta,\lambda) = \beta\left[V(x) + w_{\mathrm{Go}}V_{\mathrm{Go}}(x)\right] \quad (4)$$

where $V(x)$ is the unbiased potential energy, V_{Go} is a biasing potential favoring the formation of natively folded conformations[20], and w_{Go} is an energy weight. The exchange parameters are w_{Go} and β so as to enhance the rate of folding and unfolding at multiple temperatures.

The last two applications employ multi-dimensional replica exchange umbrella sampling (RE-US), but in two quite different scenarios, thus demonstrating the broad applicability of the approach. In all RE-US simulations, the dimensionless potential energy can be written as

$$U(x;\beta,\lambda) = \beta\left[V_0(x) + W(x;\lambda)\right] \quad (5)$$

where $V_0(x)$ is the unbiased potential and $W(x;\lambda)$ is a biasing potential applied to one or more coordinates in order to localize sampling to regions that might not otherwise be sampled.

The first RE-US scheme analyzes small-scale conformational changes in an explicitly solvated biopolymer by applying biasing potentials to the backbone degrees of freedom. The second application is a chemical reaction in solution. Biases are applied to the breaking and forming bonds. The chemically reactive solute is treated quantum mechanically, whereas the solute is treated by classical molecular mechanics.

Due to numerous calculations needed in quantum calculations and the unfavorable scaling of these calculations (unlike classical force fields, quantum potentials cannot be described as pair-wise interactions), it is extremely difficult to sample over the conformational space to obtain sufficient and meaningful ensembles. Hence the advantages of a RE-US approach are especially important in QM simulations, such as the example system presented here. RE-US is one of the most promising ways to increase the feasible application of quantum potentials via increased sampling efficiency.

2.1 Computational Requirements

As introduced above and further elaborated below, in order to efficiently perform large scale RE calculations of this kind on XSEDE we adopt an asynchronous formulation of RE, which, unlike conventional synchronous implementations, requires only a fraction of the computing resources nominally required by the application [(number of replicas) times (number of CPU cores per replica)]. Furthermore the loss of computing resources does not cause termination of the application. Conversely, it allows the expansion of the application dynamically as new resources become available.

In the current implementation, communication between pairs of replicas is achieved through a shared filesystem while they are temporarily checkpointed and not actively running. This approach has the advantage of being very general and well supported by BigJob. It also does not require source code modification of legacy MD kernels.

3. SOFTWARE ENVIRONMENT

A Pilot-Job is a mechanism by which a proxy for the actual simulations is submitted on the resource to be utilized; this proxy, in turn, conveys to the application the availability of resources and also influences which tasks are executed.

The P* model [16], a model for Pilot-Abstractions, works to clearly define the computation and data components of a distributed application as 'compute units' and 'data units' in the context of Pilot-Jobs and Pilot-Data. A compute unit describes a self-containing piece of work, e.g. a computational task that potentially operates on a set of input data, while a data unit is a container for a logical group of data that is often accessed together or comprises a larger set of data; e.g. a data file or chunk.

BigJob is a Pilot-Job system implementation which provides a framework for running many types of distributed applications – including but not limited to very-large scale parallel simulations, many small high-throughput simulations, or ensemble-based workflows. Consistent with the P* model, BigJob [21, 15] provides a unified run-time environment for Pilot-Jobs on heterogeneous infrastructures. For this purpose, BigJob provides a higher-level, unifying interface to heterogeneous and/or distributed data and compute resources. The framework is accessed via the Pilot-API, which provides two key abstractions: Pilot-Job and Pilot-Data.

In order for BigJob to work on heterogeneous resources, it requires an interoperability layer which provides access to a variety of middleware. This is achieved through the use of the Simple API for Grid Applications (SAGA) [10, 22].

3.1 AsyncRE

ASyncRE is a Python package for performing file-based aRE simulations. The current implementation is primarily intended for use on computer clusters managed by a queuing system and supported by a shared filesystem. The BigJob distributed computing infrastructure is used for job launching and monitoring.

The ASyncRE package includes a core module that performs common tasks such as job staging through BigJob and exchanging of parameters among replicas. Support for arbitrary MD engines and RE schemes can be introduced via simple user-provided extensions (Figure 1). Modules are currently available for the AMBER and IMPACT MD engines. A similar modular mechanism provides support for arbitrary RE schemes (temperature, Hamiltonian, *etc.*), including arbitrary multidimensional combinations of these (such as 2D RE temperature/Hamiltonian). The software is currently distributed with modules for multi-dimensional RE umbrella sampling with AMBER[3], and BEDAM λ-RE alchemical binding free energy calculations with the IMPACT MD engine[6, 8, 13].

The BigJob-based Asynchronous Replica-Exchange (ASyncRE) framework is available for public download at: `https://github.com/saga-project/asyncre-bigjob`

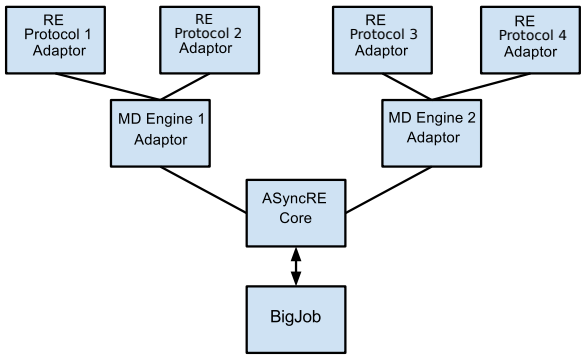

Figure 1: The structure of the ASyncRE library. The ASyncRE Core module implements all of the general-purpose facilities to start and monitor replicas via BigJob, and coordinates replica exchanges. Routines implemented in adaptor modules perform functions specific to particular combinations of MD engines and RE schemes such as AMBER+umbrella sampling or IMPACT+BEDAM.

The algorithm implemented by ASyncRE can be summarized as follows:

1. Job files and executables for the replicas are set up as appropriate for the MD engine and RE scheme as specified by the user-provided module/script. Typically this is accomplished by parsing a set of template input files according to the thermodynamic and potential energy settings that define the thermodynamic states. Each replica lives in a separate sub-directory of the working directory.

2. As resources become available, a randomly chosen subset of the replicas are submitted to BigJob for execution and enter a running ("R") state. When a replica completes a "cycle" (currently a fixed number of time steps specified in the MD engine input file), it enters a waiting ("W") state, making it eligible for exchange with other replicas as well as initiation of a new cycle.

3. Periodically, exchanges of thermodynamic parameters are attempted between replicas in a waiting state based on the appropriate replica exchange rules. This generally requires the specification of a user-defined module, namely the definition of a procedure for obtaining the (relative) reduced energies defining the Bolzmann sampling weights.

Internally, a dictionary (`status`) is used to track the statuses of the replicas (cycle, thermodynamic state, running status, *etc.*) The status of the application is check-pointed periodically using pickle. When restarting a previously stopped job, the status data structure is restored from this file and the calculation proceeds.

An important feature of ASyncRE is the use of a "run buffer" to hide latencies involved in the management of exchanges. That is, ASyncRE submits subjobs to BigJob in excess of the allocated compute resources such that the submission of a replica for execution does not, in general, imply that it is immediately executed. Rather, the replica is packaged in a BigJob compute unit which is ready to begin execution as soon as sufficient resources are available. In this way BigJob does not have to wait for a replica to be prepared for the next cycle before it can be executed. Instead, replicas are launched from a pool of already prepared replicas.

3.1.1 Installation

Following the installation of BigJob (see above) in the virtual user Python environment, ASyncRE is installed using PyPi as follows:

```
pip install numpy
pip install configobj
pip install async_re-0.1.0.tar.gz
```

`numpy` and `configobj` are currently required dependencies and will be installed automatically after ASyncRE is integrated into the PyPi archive. `async_re-0.1.0.tar.gz` is the current Python distribution package for ASyncRE and is available publicly through `github`.

In addition to installing the package in the virtual environment, it is convenient to maintain a copy of the ASyncRE modules, scripts, and examples in an easy-to-access location such as:

```
cd ~/src
tar zxvf async_re-0.1.0.tar.gz
```

Current application-level class files (such as `amberus_async_re.py` behave as executable scripts when launched directly (see below). Documentation files can be found in the `doc` subdirectory and sample files in the `examples` subdirectory of the package directory.

3.1.2 Execution

A typical sequence of commands to initiate an ASyncRE run on XSEDE is as follows:

```
ssh <cluster_head_node>
source ~/.bigjob/bin/activate
cd <working_directory>
python amber_us.py us.config > LOG 2>& &
```

The second command above activates the virtualenv Python environment (see above). `amber_us.py` is a simple user-provided Python script that loads the appropriate modules and launches the simulation. For example, for AMBER/umbrella sampling it might be:

```
import sys
from amberus_async_re \
  import amberus_async_re_job
rx = amberus_async_re_job(sys.argv[1])
rx.setupJob()
rx.scheduleJobs()
```

A control file (`us.config` above) containing keyword/-value pairs is used for setting ASyncRE runtime parameters. A control file for an AMBER umbrella sampling simulations might look like:

```
# Main settings ------------------------
ENGINE = 'AMBER'
```

```
RE_TYPE = 'AMBERUS'
RE_SETUP = 'yes'
VERBOSE = 'yes'
ENGINE_INPUT_BASENAME = 'foo'
ENGINE_INPUT_EXTFILES = \
   'foo.parm7,foo_0.rst7'
# RE/simulation settings ----------
FORCE_CONSTANTS = \
   '5.0,5.0:5.0,5.0:5.0,5.0:5.0,5.0'
BIAS_POSITIONS = \
   '275.,275.:275.,280.:280.,285.:285.,280.'
# BigJob settings ----------
WALL_TIME = 200
COORDINATION_URL = 'redis://<redis_server>'
RESOURCE_URL = 'pbs://localhost'
QUEUE = 'batch'
BJ_WORKING_DIR = '/home/user/amber_us/agent'
TOTAL_CORES = 16
SUBJOB_CORES = 8
# ----------
```

A detailed description of these settings is provided in the ASyncRE user documentation.

The command above causes, among other things, the submission of a job to the local queuing system named `bliss_job`. Execution terminates after a specified amount of wall-clock time. The internal state of the simulation is check-pointed periodically as well as at the end of execution so that it can be restarted. Failed runs are automatically detected and the relevant replicas are reset and restarted.

The ASyncRE package is user-extensible and users are free to implement RE modalities not natively supported by the current ASyncRE package (see details on writing extension modules in the ASyncRE documentation). Scripts that implement user-provided RE schemes are typically preceded by customized classes/methods (usually overriding one inherited from the main class). A custom application might look like:

```
import sys, math, os, ...
from amber_async_re import pj_amber_job

class myREscheme_async_re_job(pj_amber_job):
    def _checkInput(self):
        ...
    def _buildInpFile(self, replica):
        ...
    def ...
    ...

if __name__ == '__main__':
   rx = myREscheme_async_re_job(sys.argv[1])
   rx.setupJob()
   rx.scheduleJobs()
```

4. LARGE-SCALE REPLICA EXCHANGE ON XSEDE: EXPERIMENTS AND RESULTS

4.1 Performance Model

Like many complex distributed applications, replica exchange consists of multiple individually developed components that are combined and orchestrated to carry out the application workload. In our implementation of the replica exchange workflow, we identify three distinct components: (1) BigJob as the underlying Pilot-Job system, (2) ASyncRE as the replica exchange application framework, and (3) AMBER / IMPACT as the application kernels.

Each of the three component has its own performance characteristics and can introduce overhead to the overall application. We define overhead from an application perspective as the fraction of the application runtime that is not spent on running the application kernels, but on application management logic, communication and coordination.

From an application point of view, the most important performance metric is throughput, which is defined as the amount of simulation time achieved per hour compute time on a given number of CPU cores. This is equivalent with the time to completion (TTC) for a specific number of steps (s). On the lowest level, TTC is the sum of the queue waiting time and the application runtime:

$$TTC_s = T_Q^O + T_X$$

However, if the overall runtime of the simulation is sufficiently long, T_Q becomes negligible. We split up T_X into application kernel runtime T_R and overhead. Since overhead can occur in all three components, we define T_{BJ}^O as the overhead introduced by BigJob, T_{RE}^O as the overhead introduced by the AsyncRE package, and T_K^O as the overhead introduced in the application kernel itself:

$$TTC_s = T_R + T_{BJ}^O + T_{RE}^O + T_K^O$$

Application kernel overhead occurs for example if the kernel supports thread-level parallelism but doesn't scale linearly with the number of threads. Overhead in ASyncRE occurs when the framework executes management tasks, e.g., finding matching exchange partners for a set of replicas and no computation happens during that time. Lastly, we consider BigJob overhead as the time that is spent by BigJob on the placement and monitoring of subjobs during which no computation occurs. This includes network round-trip time for communication via Redis.

To measure the different aspects of application overhead, instrumentation of all three components is necessary. This is still work in progress and the experiments and results described below only implement probes on a very high level. Nevertheless, the model helps us to reason about the observed differences in application performance. Detailed measurements and instrumentation of BigJob and ASyncRE are planned for the future.

4.2 Systems Investigated

Four physical model systems were investigated, two each with the IMPACT and AMBER MD engines. These systems represent broad classes of problems of chemical interest, namely binding, "large scale" macro-molecular folding, "small scale" conformational changes, and chemical events (*i.e.* bond annihilation/formation). Briefly, the systems are as follows:
IMPACT (implicit solvent)

1. Host/guest binding of cyclooctanol/β-cyclodextrin. The exchange parameters are all combinations (192, 384,

or 768) of the system temperature (8, 16, or 32 values, 300 - 600 K) and an alchemical parameter (24 values, 0 - 1) coupling the host/guest interactions (Eqn. 3).

2. Folding of the TrpCage mini-protein. The exchange parameters are all combinations (84, 336, or 1008) of the system temperature (6, 24, or 72 values, 300 - 600 K) and the coupling weight (14 values, 0 - 0.42) of a Go-type biasing potential (Eqn. 4).

AMBER (explicit solvent)

1. Umbrella sampling of the ϕ/ψ torsions (24 values, 0 - 360 degrees) of alanine dipeptide. The exchange parameters are all combinations (576 total) of the harmonic biasing potentials on each torsion (Eqn. 5).

2. Hybrid quantum mechanical/molecular mechanical umbrella sampling of phosphoryl transfer in 2-hydroxy ethyl ethyl phosphate, a model reaction for base catalyzed RNA cleavage. The exchange parameters are all combinations (192 total) of the harmonic biasing potentials on the breaking (12 values, 1.5 - 4.25Å) and forming bonds (16 values, 1.5 - 5.25Å).

4.3 Experimental Configuration

Multiple RE simulations were performed on each system in one of three different modes intended to probe various scaling aspects of ASyncRE and BigJob. All runs were given a wall time of one hour (with appropriate extrapolations). The three modes were as follows:

I (IMPACT systems 1-2, MM) Fixed Pilot size, fixed Pilot runtime, and varying number of replicas in proportion to the number of cores per replica

II (AMBER system 1, MM-US) Fixed Pilot size and approximately fixed cycle length (in CPU time) while varying the number of concurrent jobs (*i.e.* the number of cores per job) and the simulation time of each cycle.

III (AMBER system 2, QM/MM-US) Fixed replica count while varying the Pilot size.

4.4 Results

The main results for IMPACT system 1 (host/guest binding) are reported in Table 1. In this test the number of replicas was varied from 768 to 192 by increasing the level of parallelism of IMPACT (from 1 core per replica to 4), while the size of the BigJob was kept fixed at 384 cores (half of the CPU cores required to run all of the replicas at once). With 1 core per replica and 768 replicas the measured throughput is 954 ns/day to be compared with the nominal maximal throughput of 1,300 ns/day corresponding to uninterrupted MD at the measured MD CPU speed (4.2 minutes for 10ps per replica). The fraction of the observed throughput relative to the maximum (73%) measures the overhead imposed by replica exchange coordination ($T^O_{BJ} + T^O_{RE}$ above). We see that the throughput (expressed in simulation time per day) is reduced from 945 ns/day with 768 replicas to 708 ns/day with 192 replicas. The reduction is due in large part to parallelization overheads in going from 1 core per replica, as it can be deduced from the time required to complete one MD cycle (10,000 MD steps in this case). The parallel efficiency is approximately 76 and 67% with 2 and 4 cores, respectively, which is reasonable for this very small chemical system. The replica exchange coordination overhead remains approximately constant; for the test with 192 replicas for example the throughput is 76% of the maximum similar to the 73% measured with 768 replicas. This indicates that the ASyncRE/BigJob framework is capable of handling this rather high replicas turn over (up to 4,000 subjob launches per hour) quite efficiently.

# of replicas	cores/ replica	time/cycle (min)	simulation speed (ns/day)
768	1	4.2	945
384	2	2.6	880
192	4	1.5	708

Table 1: Scaling results for IMPACT system 1 (host/guest binding). Each run consisted of 384 cores running for one hour while varying both the number of replicas and the number of cores allocated to each replica.

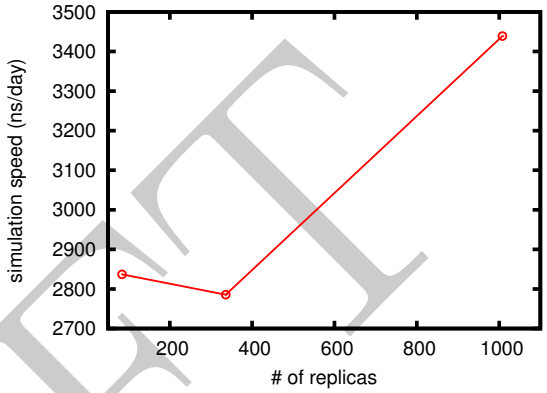

Figure 2: Scaling results for IMPACT system 2 (mini-protein folding). Each run consisted of 504 cores with 12 cores per replica running for one hour while varying the number of replicas. The maximum throughput (3.5 μs/day) is obtained with the largest number of replicas.

The results for IMPACT system 2 (mini-protein binding) (Fig. 2) confirm the general trends observed for the smaller host-guest system above. For this case IMPACT's parallel efficiency is more favorable due to the greater number of atoms. It has been possible to test with the maximum number of cores per replica (12) allowed by the hardware. As expected maximum throughput (3.5μs/day) is obtained with serial replica execution with the largest number of replicas (1008). The maximum throughput in this case is 4.7 μs/day yielding a replica exchange overhead of 75% similar to the host-guest system above. With the largest core count the throughput is 2.8 μs/day, lower than the observed maximum, but yielding a 6-fold speed up in terms of single-replica MD throughput.

Standard REMD simulations employing molecular mechanics (MM) force fields offer parallel scaling in both the number of replicas and the number of processors allocated to each replica. In principle, an optimal scheme should bal-

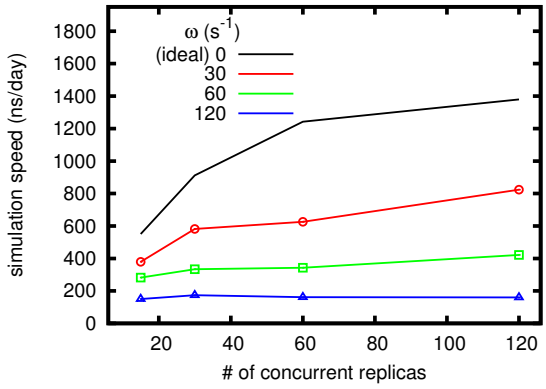

Figure 3: Scaling results for AMBER system 1. Each run consisted of 720 cores divided amongst differing numbers of replicas (the pool of potential replicas was fixed at 576) and coordination frequencies, ω, (*i.e.* the frequency with which simulations are started, stopped, and exchanged). Ideal performance (black line) would be obtained if there were no overhead in launching or coordinating simulations ($\omega = 0$). Increasing the frequency with which replicas are "launched" results in diminished performance.

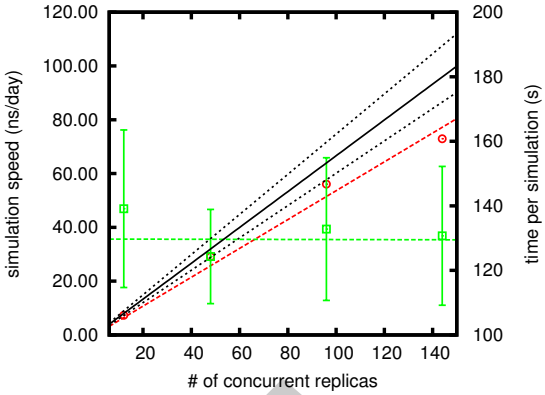

Figure 4: Scaling results for AMBER system 2. The total number of cores allocated was varied while each simulation was always run on a single core. As expected, performance increases linearly (red fit line) with the available resources. The performance of individual simulations is also extremely consistent at different core counts (green fit line), verifying that the linear scaling is in fact representative of an increase in the number of simulations running and not just the increased core count. Ideal behavior is defined as all simulations running with constant efficiency equal to the empirical average (black lines with values and 95% confidence intervals from green linear fit).

ance the efficiency gains of both types of scaling in order to produce the most simulation time in a given period of real time. In order to assess the efficiency of ASyncRE/BigJob in combination with the AMBER MD engine, a fixed size pilot job of 720 cores was allocated (AMBER system 1 under Experimental Configuration) with various numbers of cores allotted to each simulation. In this scheme, the number of simulations being coordinated in REMD varies, as well as the simulation rate (in ns/day) attained by each replica. In order to probe the efficiency with which ASyncRE/BigJob coordinates simulations, the length of each simulation cycle (alternatively the frequency with which simulations are coordinated) was fixed in real time by varying the simulation time per cycle (Figure 3). Ideal performance would be obtained if there were no overhead in coordinating the simulations. However, some cost must be incurred in starting, stopping, and restarting simulation cycles as well as in coordinating exchanges amongst stopped replicas and the cost of this overhead is obviously expected to increase as it is incurred more frequently. As seen in Figure 3, this does indeed result in diminished output (as measured in ns/day) as the coordination frequency increases. However, for a fixed coordination interval, performance appears to quickly plateau with respect to the number of simulations being coordinated, especially in comparison to ideal, un-coordinated behavior.

The efficiency of many advanced simulation models, including quantum mechanically based methods, is fundamentally difficult or impossible to improve by parallelization. However, RE-US provides a general tool for increasing the efficiency of such simulations anyway, by increasing the statistical power of the short trajectories that can still be obtained in a reasonable time frame. In the present work, scaling in this way would rely exclusively on the ability of ASyncRE/BigJob to handle hundreds or thousands of concurrent simulations. As a test of this, the performance benefit of increasing the pilot size was checked in conjunction with the AMBER MD engine running a serial quantum mechanical/molecular mechanical potential with RE-US (AMBER system 2 under Experimental Configuration). As seen in Figure 4 performance increases linearly with the core count and tracks very closely with an estimated ideality, especially up to about 100 replicas. It should be noted that ideal performance assumes no starting/stopping overhead as well as perfect and consistent hardware performance. Therefore, this behavior is likely not even exactly attainable with uncoupled, embarrassingly parallel execution.

5. DISCUSSION

The results presented here demonstrate ASyncRE/BigJob as a single general purpose framework for enabling replica exchange protocols with arbitary simulation engines. This is accomplished by suitable abstraction of replica exchange as a particular variant of the pilot job abstraction. That is, a protocol is added to communicate information and regulate exchanges. Because this coordination is only dependent on the statistical laws underlying importance sampling methods (*i.e.* detailed balance), it is applicable to *any* simulation engine or combination of engines. This versatility is utilized in the development of simulation engine adaptors (Figure 1) and provides access to essentially any existing implementation of a simulation methodology (*e.g.* MD) or model (*e.g.* MM or QM/MM). Information exchange between adaptors and the Pilot are further mediated by specialized exchange adaptors, thereby enabling arbitrary exchange protocols (*e.g.* temperature exchange or umbrella sampling).

The tests reported here demonstrate that the above framework is applied in a consistent manner across simulation engines and execution modes. That is, the performance of

the simulation engines are inline with expected behavior in the absence of coordination from ASyncRE/BigJob. This can been seen for the host/guest system with implicit solvation in Table 1, where near linear decreases in time per cycle are obtained with linear increase the processor count (perfect linear scaling is unlikely for such a small system). Similar behavior is seen for the system in Figure 3 with AMBER. Figure 4 shows that behavior is also consistent as the amount of coordinate resources increase. That is, there is little to no penalty for running additional concurrent jobs on additional resources (note that the ideal behavior here only an esimate).

Lastly, stress tests of the coordination overhead of ASyncRE/BigJob show that some efficiency is lost when the exchange intervals are very short (Figure 3). Fortunately, there appears to be some mitigation of this effect at lower coordination frequencies and it may be possible to optimize performance by tuning the coordination frequency to the number of replicas. That is, while it is desirable to avoid diminished simulation speed, this may not always be an issue in practice since one can, to a degree, manually control the coordination frequency. Additionally, inefficient or poorly scaling applications (such as QM/MM methods) may never even be able to obtain short enough cycles to cause concern. Regardless, this phenomenon will be the subject of future investigation and development.

Naïvely, an optimal simulation produces as much simulation time as possible in a given period of real time. Therefore, it is tempting to evaluate the quality of a simulation protocol based on its simulation speed. However, not all simulation time is equal – e.g., multiple short simulations do not usually contain as much statistical information as a single long simulation of the same length, although the real time speed of the former generally exceeds the latter. The basic aim of REMD is to increase the statistical power of multiple simulations by facilitating the exchange of information between them in a concerted fashion. This should be kept in mind when analyzing performance graphs, as one might be tempted to run as many small simulations as possible in an attempt maximize simulation speed. For real applications and performance optimizations, other more physically and statistically relevant metrics must be employed, however, these are beyond the scope of the present work.

Acknowledgement

This work is funded by an NSF Cyber-enabled Discovery and Innovation Award (CHE-1125332) and NSF-ExTENCI (OCI-1007115). This work has also been made possible by computer resources provided by TeraGrid TRAC award TG-MCB090174 and the BioMaPS High Performance Computing Center. BKR acknowledges additional computational resources from a Peter Kollman Graduate Award in Supercomputing through the ACS Division of Computers in Chemistry and NICS (Kraken). This work builds upon the critical contributions of the BigJob development team, in particular Andre Luckow and Andre Merzky. We are grateful to Yaakoub El-Khamra and Tommy Minyard (TACC) for help with debugging and performance tuning of BigJob on Lonestar, Ranger, and Stampede.

References

[1] M. Andrec, A. K. Felts, E. Gallicchio, and R. M. Levy. Protein folding pathways from replica exchange simulations and a kinetic network model. *Proc Natl Acad Sci U S A*, 102(19):6801–6806, May 2005.

[2] G. Bussi, F. L. Gervasio, A. Laio, and M. Parrinello. Free-energy landscape for β hairpin folding from combined parallel tempering and metadynamics. *Journal of the American Chemical Society*, 128(41):13435–13441, Oct. 2006.

[3] D. A. Case, T. A. Darden, T. E. Cheatham III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. W. Götz, I. Kolossváry, K. F. Wong, F. Paesani, J. Vanicek, R. M. Wolf, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. R. Roe, D. H. Mathews, M. G. Seetin, C. Salomon-Ferrer, R. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. A. Kollman. *AMBER 12*. University of California, San Francisco, San Francisco, CA, 2012.

[4] J. Chodera and M. Shirts. Replica exchange and expanded ensemble simulations as gibbs sampling: Simple improvements for enhanced mixing. *Journal of Chemical Physics*, 135(19):194110, 2011.

[5] A. K. Felts, Y. Harano, E. Gallicchio, and R. M. Levy. Free energy surfaces of beta-hairpin and alpha-helical peptides generated by replica exchange molecular dynamics with the agbnp implicit solvent model. *Proteins: Struct., Funct., Bioinf.*, 56:310–321, 2004.

[6] E. Gallicchio, M. Lapelosa, and R. M. Levy. Binding energy distribution analysis method (BEDAM) for estimation of protein-ligand binding affinities. *J. Chem. Theory Comput.*, 6(9):2961–2977, Sept. 2010.

[7] E. Gallicchio and R. M. Levy. Advances in all atom sampling methods for modeling protein-ligand binding affinities. *Curr. Opin. Struct. Biol.*, 21(2):161–166, Apr. 2011.

[8] E. Gallicchio and R. M. Levy. Prediction of sampl3 host-guest affinities with the binding energy distribution analysis method (BEDAM). *J Comp Aided Mol Design*, Online First, 2012.

[9] E. Gallicchio, R. M. Levy, and M. Parashar. Asynchronous replica exchange for molecular simulations. *J. Comp. Chem.*, 29(5):788–794, 2008.

[10] U. H. Hansmann and Y. Okamoto. New monte carlo algorithms for protein folding. *Current Opinion in Structural Biology*, 9(2):177–183, 1999.

[11] W. Jiang, Y. Luo, L. Maragliano, and B. Roux. Calculation of Free Energy Landscape in Multi-Dimensions with Hamiltonian-Exchange Umbrella Sampling on Petasacle Supercomputer. *J. Chem. Theory Comput.*, 8:4672–4680, 2012.

[12] M. Lapelosa, E. Gallicchio, and R. M. Levy. Conformational transitions and convergence of absolute binding free energy calculations. *J. Chem. Theory Comput.*, 8:47–60, 2012.

[13] P. Liu, X. Huang, R. Zhou, and B. J. Berne. Hydrophobic aided replica exchange: an efficient algorithm for protein folding in explicit solvent. *J Phys Chem B*, 110(38):19018–19022, Sep 2006.

[14] A. Luckow, L. Lacinski, and S. Jha. SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems. In *The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 135–144, 2010.

[15] A. Luckow, M. Santcroos, A. Merzky, O. Weidner, P. Mantha, and S. Jha. P*: A model of pilot-abstractions. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–10, Oct.2012.

[16] Y. Meng and A. E. Roitberg. Constant ph replica exchange molecular dynamics in biomolecules using a discrete protonation model. *J. Chem. Theory Comput.*, 6(4):1401–1412, Apr 2010.

[17] A. Mitsutake, Y. Mori, and Y. Okamoto. Multi-dimensional multicanonical algorithm, simulated tempering, replica-exchange method, and all that. *Physics Procedia*, 4:89–105, 2010.

[18] K. Murata, Y. Sugita, and Y. Okamoto. Free energy calculations for dna base stacking by replica-exchange umbrella sampling. *Chemical Physics Letters*, 385(1-2):1–7, Feb. 2004.

[19] T. V. Pogorelov and Z. Luthey-Schulten. Variations in the fast folding rates of the¡ i¿ λ¡/i¿-repressor: A hybrid molecular dynamics study. *Biophysical journal*, 87(1):207–214, 2004.

[20] BigJob, `http://saga-project.github.io/BigJob/`.

[21] saga-python, `http://saga-project.github.io/saga-python/`.

[22] C. J. Woods, J. W. Essex, and M. A. King. The development of replica-exchange-based free-energy methods. *J. Phys. Chem. B*, 107(49):13703–13710, Dec. 2003.

[23] I.-C. Yeh, M. A. Olson, M. S. Lee, and A. Wallqvist. Free-energy profiles of membrane insertion of the m2 transmembrane peptide from influenza a virus. *Biophysical Journal*, 95(11):5021–5029, Dec. 2008.