RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

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

Brian K. Radak, **Melissa Romanus**, Emilio Gallicchio, Tai-Sung Lee, Ole Weidner, Nan-Jie Deng, Peng He, Wei Dai, Darrin York, Ronald Levy, Shantenu Jha

Overview

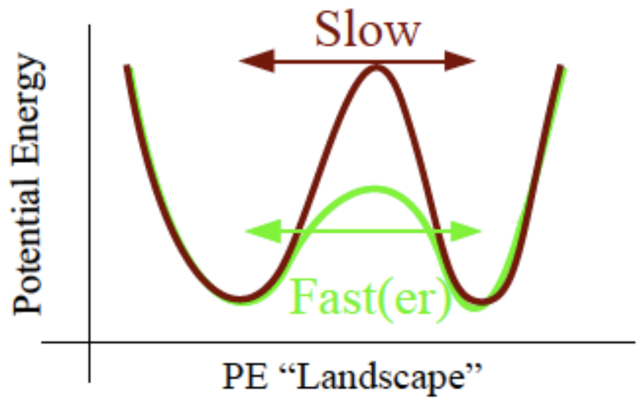
Introduction

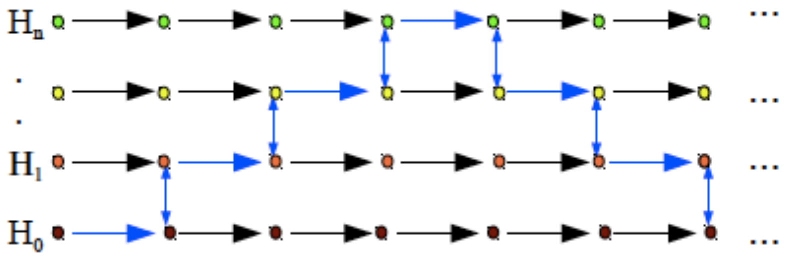
- Replica-exchange molecular dynamics (REMD) is a method for effectively sampling high-dimensional rough energy landscapes
  - Enhanced sampling relative to regular MD simulations
  - Systems with similar potential energies can sample conformations at different temperatures
    - Potential to overcome energy barriers on the potential energy surface
  - Can be used for a range of physical systems from protein-folding to binding energy affinity calculations
- Traditional REMD previously limited to smaller scales
  - Enhanced sampling comes at the cost of coordination overhead
- Goals of this Paper: Release Async RE package to public – show current performance statistics to scientific community
  - *Note: This paper does not aim to provide a computational chemistry analysis of the results.*

"Replica" Exchange

- Multiple importance sampling simulations (i.e. MD trajectories) executed in parallel, using intermittently exchange information
- Information exchanged corresponds to thermodynamic state definitions
  - Microscopic reversibility requirements ensure simulation maintains the correct importance sampling weights in each state
    - Thus, sims can be viewed as identical "replicas" in both target configurational and state spaces
    - Advantage of such uniform sampling: Diagnose locally insufficient sampling, assess the accuracy of global properties

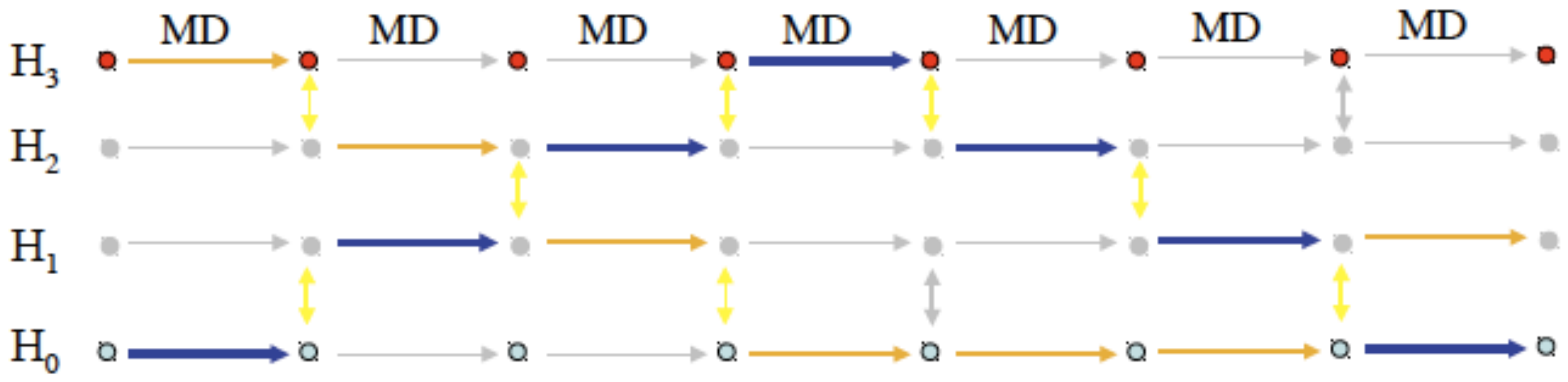
Why Replica Exchange?





- Based upon the Hamiltonian-hopping idea
  - Can move in PE space more effectively, "faster"
  - *Consqueunce:* Thermodynamic equilibrium between Hamiltonian states must be maintained
- Serial Implementations (serial tempering, etc.):
  – free energy weights are adjusted to regulate time spent at each state
- Parallel Implementations (replica exchange):
  – runs as many MD replicas as states; Hamiltonian swaps between replicas
  – overall each state is equally sampled automatically
  – suitable for parallel environments

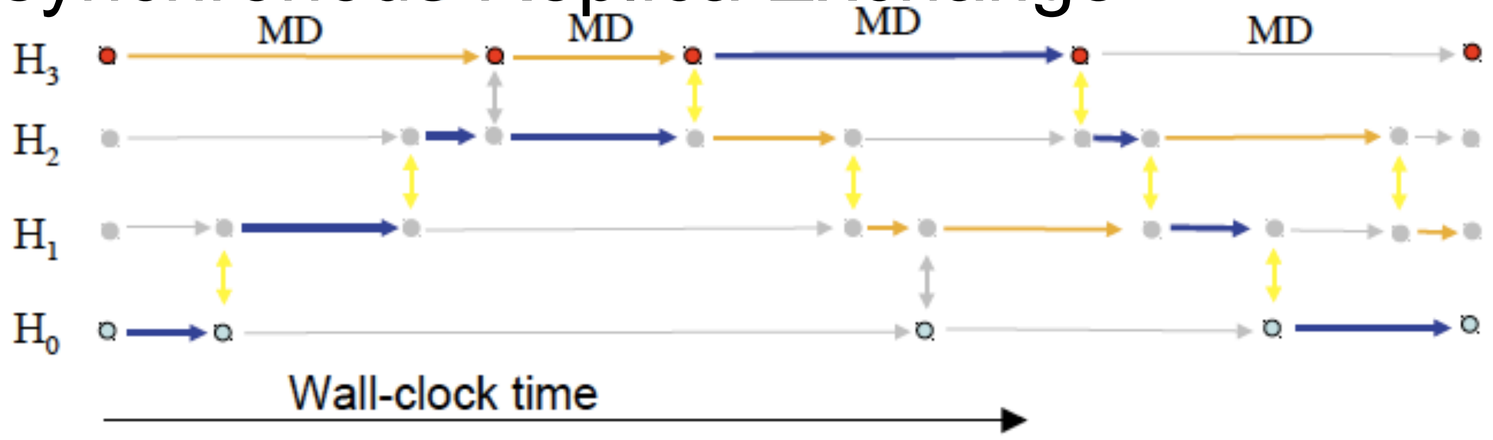
Synchronous Replica Exchange



PROBLEM: NOT scalable to 1000's of replicas/states

- After a fixed number of MD steps all replicas stop computing and exchanges occur simultaneously.
- All replicas need to run at the same time
- Termination of one replica causes termination of the whole RE simulation.
- Not suitable for large RE calculations on limited resources
- Not suitable for pools of delocalized, heterogeneous, dynamic, unreliable, possibly at times isolated processors.
- Multi-dimensional RE schemes are difficult to implement within existing MD engines codebases.
- Overall speed is the speed of the slowest processor. CPU resources – often divided among a minority

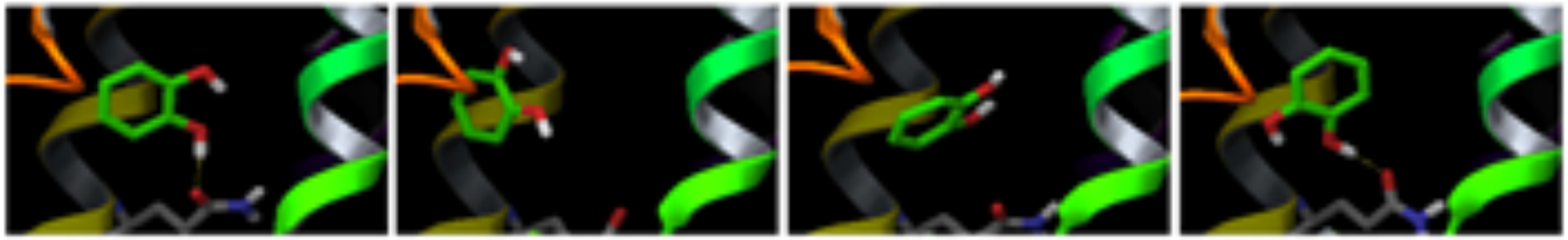
Asynchronous Replica Exchange



- Exchanges are performed at random times between random pairs of available replicas.
- Replicas run independently in between exchanges.
- Termination of one replica does not affect the others.
- Not all replicas need to be running at the same time.
- The number of running replicas can vary depending on available resources.
- Satisfies microscopic reversibility.
- Does not rely on centralized master gather/scatter mechanism.
- Each replica can complete a different number of MD steps in between swaps.
- Suitable for large RE applications on dynamic and heterogeneous computational resources.
- Fast and slow processors automatically operate at near 100% utilization. Overall speed scales as global processing capacity.

RE in Computational Terms

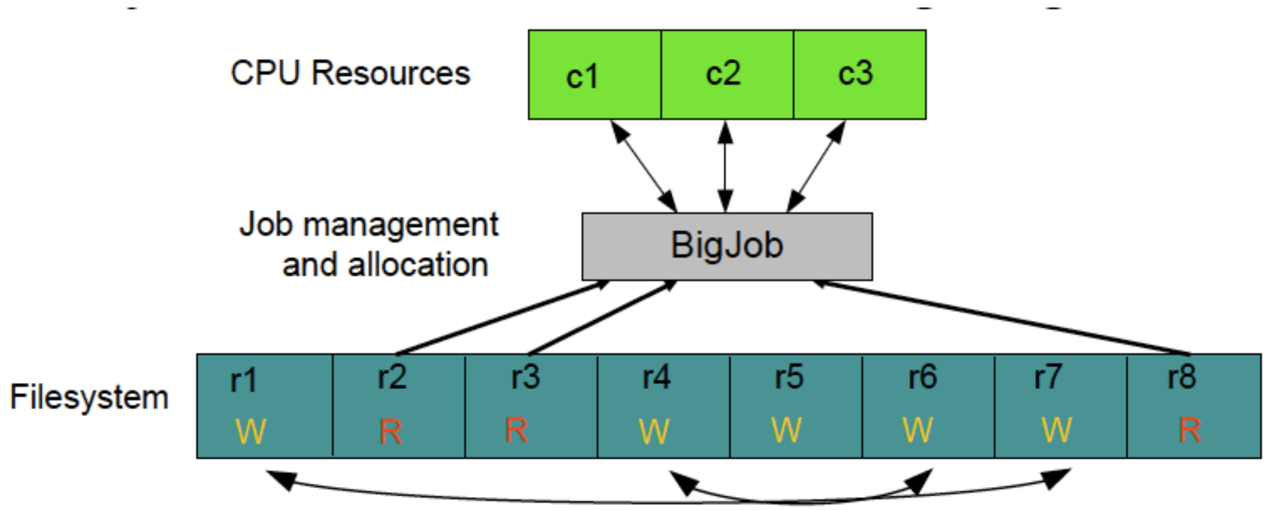
- Molecular Dynamics Simulations
  - Multiple molecular dynamics threads (replicas) are each assigned to a different thermodynamic or potential energy state of the system
  - Executed in parallel
  - Replicas travel in configurationally space as well as in state space by communication and exchange of parameters (state assignments)
- RE Computationally
  - Scaling many loosely-coupled MPI-style MD simulations
    - Use of pre-existing MD simulators such as IMPACT, AMBER, etc.
  - Each replica represents an individual execution model with different input parameters (many jobs submitted to batch queue)



CDI Project Goal: Understand important aspects of the physics of protein-ligand recognition by multidimensional replica exchange (RE) computer simulations

Async Replica-Exchange Package

- Python package built to perform file-based asynchronous parallel replica exchange



- Exchanges are performed for pairs of replicas currently not running
- Plus: Does not require changes to MD code
- Minus: Exchanges only possible when replicas in "Waiting" state = low exchange frequency

Job and Data Management using Pilots

- Job launching and monitoring occurs through the use of BigJob (http://saga-project.github.io/BigJob/)
  - Flexible, extensible Pilot-Job system
  - Pilot-Jobs allow you to submit a “container” job to a batch queue; i.e. 100 tasks does not require 100 separate jobs to the queue – submit 1 job with space for 100 tasks and allow BigJob to manage
  - BigJob is built upon SAGA (Simple API for Grid Applications) – an “access layer” for Distributed Computing Infrastructure
    - SAGA is an OGF standard
    - Particular implementation of SAGA by RADICAL group: saga-python (http://saga-project.github.io/saga-python/)
      - Write code in a well-defined API and be able to access different middleware and operating systems
        - » Use same job description to submit to PBS or SLURM or Clouds
- BigJob is an open-source python module
  - Can be used as an underlying “framework” to develop more advanced applications

AsyncRE: Python Utility for File-Based Asynchronous RE

- Main architecture:
  - Replicas are prepared in individual directories (r1, …, rM)) starting from template input files, or similar.
  - A subset of replicas is launched via BigJob. They enter running “R” state.
  - When a replica completes a run it enters a "W" (wait) state.
  - Exchanges of thermodynamic parameters are conducted by Gibbs sampling among waiting replicas
  - Cycle is repeated.

  - Detection of failed replicas: automatically resubmitted.
  - Checkpointing and restart.
  - Resilient execution (i.e. file system errors).
  - Run and “forget”

Async RE Package Current Analysis Methods

- REUS: Replica Exchange Umbrella Sampling

  [Sugita, Kitao, Okamoto, JCP, 113:6042 (2000)]

  $V(x,\lambda) = w(x; d_\lambda)$ $\qquad$ $w(x) = \text{biasing potential}$

  Originally proposed to compute the end-to-end distance PMF of a peptide, in which case the biasing potentials are harmonic restraining potentials of the end-to-end distance d.

- BEDAM: Binding Energy Distribution Analysis Method

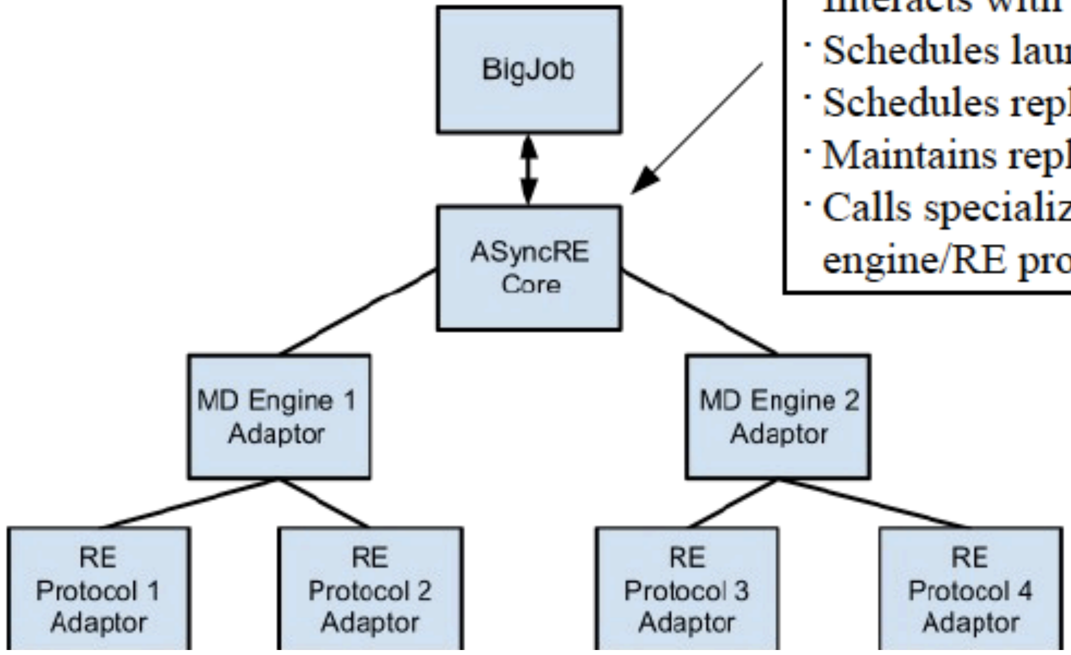
  [Gallicchio, Lapelosa, Levy – unpublished]

  $V(x,\lambda) = \lambda V_{RL}(x)$ $\qquad$ $V_{RL} = \text{ligand - receptor interaction energy}$

  Replicas are distributed from λ=0 (unbound state) to λ=1 (bound state). Replicas at small λ provide good sampling of ligand conformations whereas replicas at larger λ's provide good statistics for binding free energy estimation.

AsyncRE Software Implementation

Modularization to support multiple RE modalities:

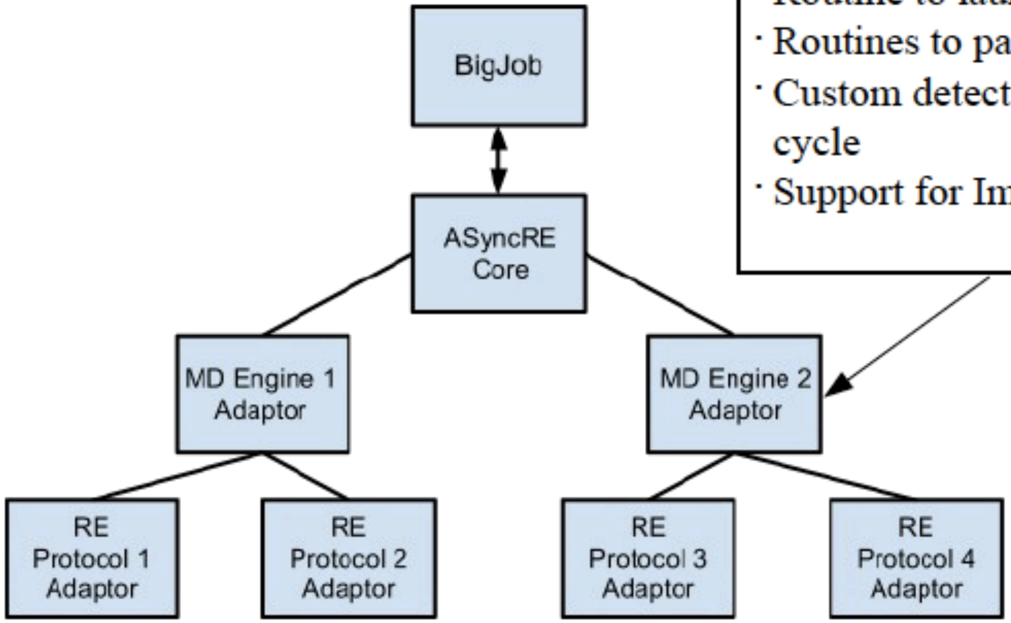


Core module class:
- Implements main logic of AsyncRE scheme
- Interacts with BigJob
- Schedules launching of replicas
- Schedules replica exchanges
- Maintains replica statuses (R, W, cycle #, …)
- Calls specialized routines based on MD engine/RE protocol combinations

https://github.com/saga-project/asyncre-bigjob

AsyncRE Software Implementation

Modularization to support multiple RE modalities:



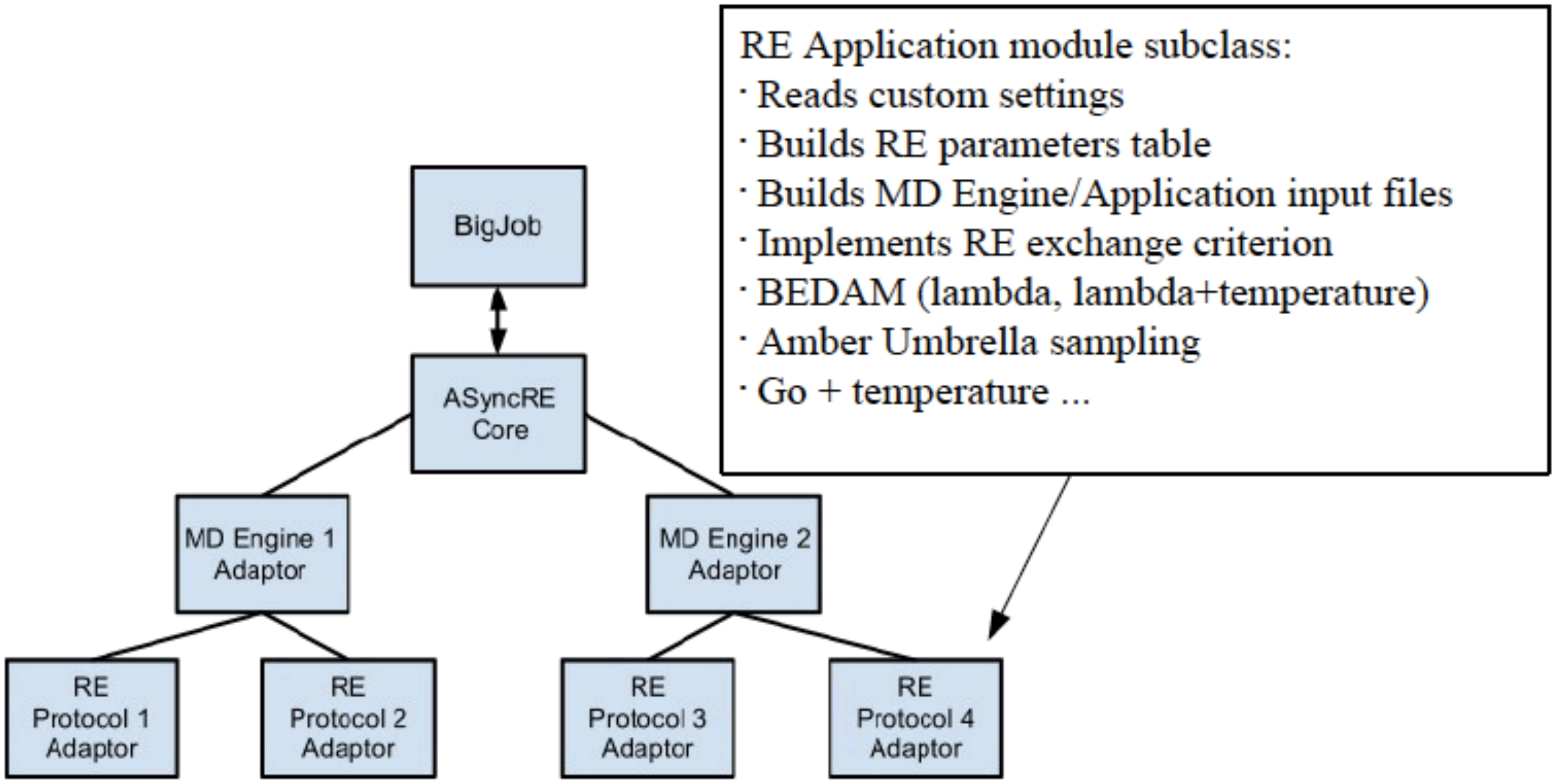
Engine module subclass:

- Routine to launch MD engine
- Routines to parse MD engine input files
- Custom detection of termination of replica cycle
- Support for Impact, Amber

https://github.com/saga-project/asyncre-bigjob

AsyncRE Software Implementation

Modularization to support multiple RE modalities:



RE Application module subclass:
- Reads custom settings
- Builds RE parameters table
- Builds MD Engine/Application input files
- Implements RE exchange criterion
- BEDAM (lambda, lambda+temperature)
- Amber Umbrella sampling
- Go + temperature ...

https://github.com/saga-project/asyncre-bigjob

RUTGERS

AsyncRE Installation

- The AsyncRE package is installed via pip (successor to easy_install). The package is available on PyPi (Python Package Index). A typical installation of the software includes only the following commands:

```
pip install numpy
pip install configobj
pip install async_re-0.1.0.t a r.gz
```

https://github.com/saga-project/asyncre-bigjob

Experiments

- *Performance Evaluation*
  - Software Package has three modes of potential bottlenecks:
    - SAGA/BigJob
    - AsyncRE Python Package
    - AMBER/IMPACT/MD simulation slow down
  - Can introduce application overhead: the fraction of the application runtime that is spent on application management logic, communication, and coordination
- How do we measure ‘quality’ or ‘performance’ in the MD world?
  - In the computational chemistry community, typically the number of nanoseconds of simulation times output per day
  - Must remember “not all simulation time is created equal”
    - Multiple short simulations do not usually contain as much statistical information as a single long simulation of the same length

Systems Investigated (aka SCIENCE!)

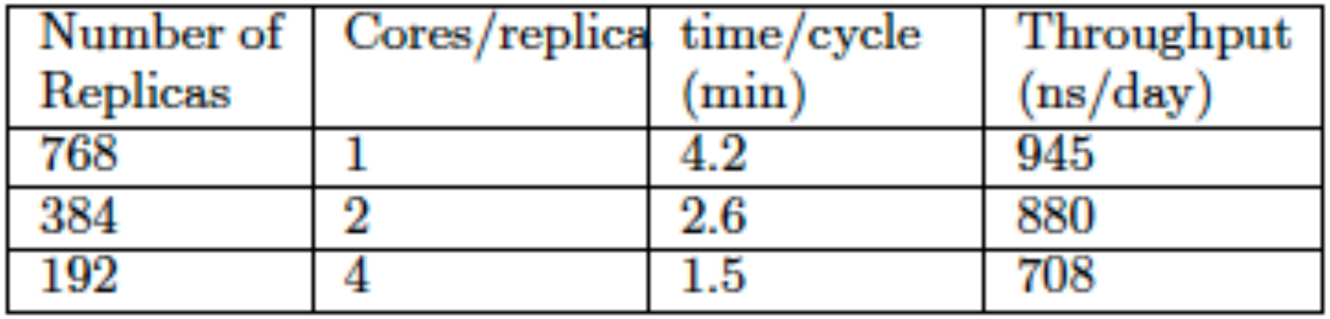
- MD engines
  - IMPACT (implicit solvent)
    - System 1: Host/guest binding of cyclooctanol/β-cyclodextrin. The exchange parameters are all permutations of the system temperatures and an alchemical parameter coupling the host/guest interactions.
    - System 2: Folding of the TrpCage mini-protein. The exchange parameters are all permutations of the system temperature and the coupling weight of a Go-type biasing potential.
  - AMBER (explicit solvent)
    - System 3: Umbrella sampling of the backbone conformational space of alanine dipeptide. The exchange parameters are all permutations of harmonic biasing potentials of each torsion. (i.e. CLASSICAL AMBER RUN)
    - System 4: 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 permutations of harmonic biasing potentials on the breaking and forming bonds (i.e. QM/MM AMBER RUN)

Let’s Talk Pilots

- IMPACT (Systems 1 and 2): Fixed Pilot Size, fixed Pilot runtime, and varying number of replicas in proportion to the number of cores per replica
- AMBER (System 3): Fixed Pilot Size, approx. 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
- AMBER (System 4): Fixed replica count (192) while varying Pilot size

Results – System 1

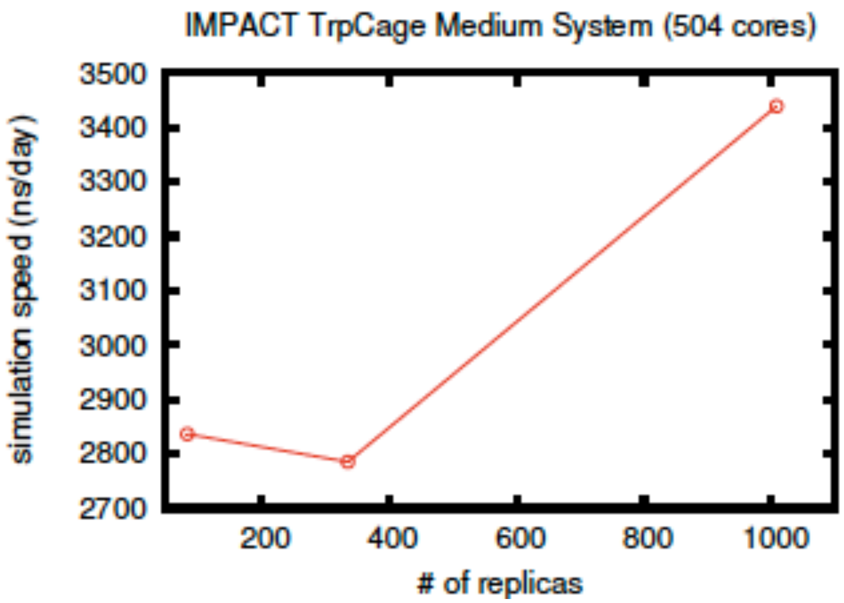
Table 1: IMPACT I Small System, 384 Core BigJob, 60 minute run time



Number of Replicas	Cores/replica	time/cycle (min)	Throughput (ns/day)
768	1	4.2	945
384	2	2.6	880
192	4	1.5	708

- Nominal maximal throughput: 1300 ns/day (uninterrupted MD at the measured MD CPU speed [4.2 minutes for 10ps per replica])
- Observed throughput / maximum = 73%
  - This corresponds to the overhead imposed by replica exchange coordination (i.e. T^{O}_{BJ} + T^{O}_{RE})
- Parallel efficiency (more cores introduces parallel overhead)
  - 76% for 2 cores
  - 67% for 4 cores
- Replica exchange coordination overhead remains approx constant
  - Indicates ASyncRE+BigJob capable of handling ~4000 subjob launches per hour for this particular system

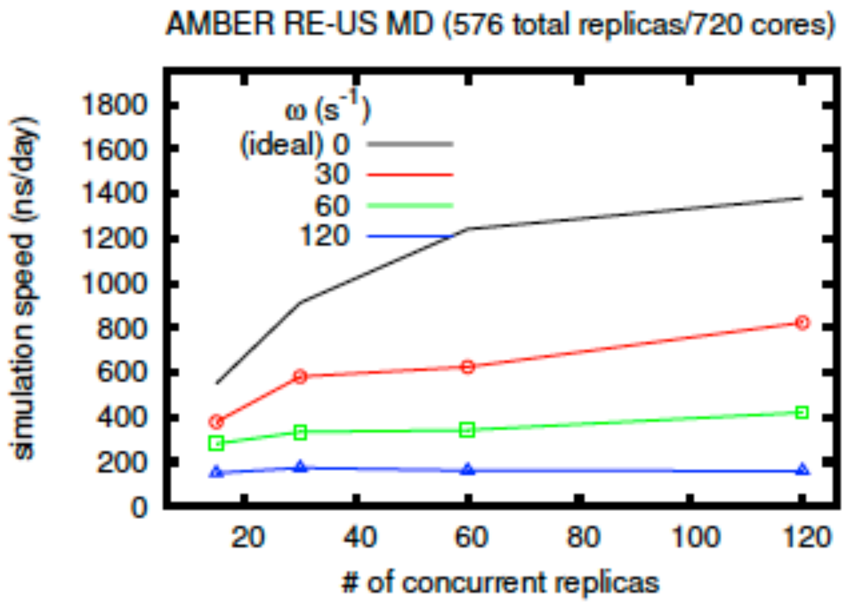
Results – System 2



- Nominal maximal throughput: 4.7 μs/day
- Observed throughput / maximum = 75%
  - This corresponds to the overhead imposed by replica exchange coordination (i.e. $T^{O}_{BJ} + T^{O}_{RE}$)
- With 12-cores we obtain 2.8 μs/day
  - 6-fold speed up in terms of single-replica MD throughput

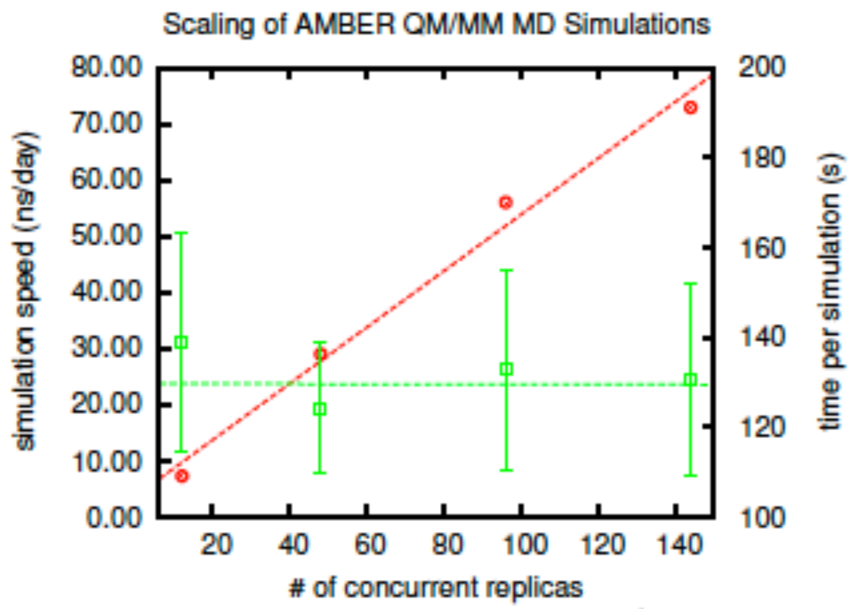
Results – System 3

Umbrella sampling of the backbone conformational space of alanine dipeptide. The exchange parameters are all permutations of harmonic biasing potentials on each torsion.



- Length of each simulation cycle (i.e. the frequency with which simulations are coordinated) was fixed in real time by varying the simulation time per cycle
- Ideal performance considered to be zero coupling in this case
- Diminished results due to coordination overheads
- Launch frequency seems to be fairly uniform at higher freq. (see blue line)
- Conclusion: Generally, the number of replicas can be increased with only a minor performance hit

Results – System 4



- QM/MM is not generally considered parallelizable
    - RE provides efficient sampling irrespective
- Repex FW: BigJob-based Repex
    - Framework to handle O(100)---O(1000) concurrent simulations
    - Amongst the earliest QM/MM
    - Framework similar to classical QM/MM
- Performance increases linearly with core count – no apparent coordination cost for additional simulations
    - QM/MM simulations are serial

Conclusion

- 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
- New AsyncRE python package proves to be able to handle more large-scale replica exchange than was previously possible
  - Modular code design allows for the development of new adaptors or RE methods (development on NAMD adaptor being investigated)
- SAGA/BigJob allow the AsyncRE framework to submit to various platforms and XSEDE machines
- The framework is applied in a consistent manner across simulation engines and execution modes
  - Performance of the simulation engines are inline with expected behavior in the absence of coordination from ASyncRE/BigJob

Future Work

- Develop new MD engine adaptors
  - NAMD
- Stress-test components more individually
  - Clear separation to identify bottlenecks and how to mitigate them
- Utilize BigJob's data-movement capabilities in order to run across multiple XSEDE machines
- Test the capabilities of running order 10^3 replicas
- Data deluge, transfer, analysis
- Adaptive RE: add/remove replicas during run
- Load on front-end host (exchanges can be somewhat expensive)
- Grow user base and user documentation

Acknowledgements

- "The Rutgers Chemists"
  - Brian Radak, Darrin York, Ronald Levy, Emilio Gallicchio, Tai-Sung Lee, Nan-Jie Deng, Peng He, Wei Dai
- Rutgers RADICAL Team
  - http://radical.rutgers.edu
  - Especially: Shantenu Jha, Pradeep Kumar Mantha (graduated), Ole Weidner, Andre Luckow, Andre Merzky, Ashley Zebrowski
- TACC ECSS Assistance: Yaakoub El-Khamra
- National Science Foundation

References

- Rutgers BioMaps: http://biomaps.rutgers.edu
- Rutgers RADICAL: http://radical.rutgers.edu
- AsyncRE Software Package: https://github.com/saga-project/asyncre-bigjob/
- AsyncRE Mailing List: async-replica-exchange@googlegroups.com
- saga-python: http://saga-project.github.io/saga-python/
- BigJob: http://saga-project.github.io/BigJob/

RUTGERS

Replica-Exchange

the end