ExTASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques

Vivekanandan Balasubramanian*, Iain Bethune†, Ardita Shkurti‡, Elena Breitmoser†
Eugen Hruska¶‖, Cecilia Clementi§‖, Charles Laughton‡ and Shantenu Jha*
*Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854, USA
†EPCC, The University of Edinburgh, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, UK, EH9 3FD
‡ School of Pharmacy and Centre for Biomolecular Sciences, University of Nottingham, University Park, Nottingham NG7 2RD, UK
§ Department of Chemistry, Rice University, Houston, TX 77005, USA
¶ Department of Physics, Rice University, Houston, TX 77005, USA
‖ Center for Theoretical Biological Physics, Rice University, Houston, TX 77005, USA

Abstract—**For many macromolecular systems the accurate sampling of the relevant regions on the potential energy surface cannot be obtained by a single, long Molecular Dynamics (MD) trajectory. New approaches are required to promote more efficient sampling. We present the design and implementation of the Extensible Toolkit for Advanced Sampling and analYsis (Ex-TASY) for building and executing advanced sampling workflows on HPC systems. ExTASY provides Python based "templated scripts" that interface to an interoperable and high-performance pilot-based run time system, which abstracts the complexity of managing multiple simulations. ExTASY supports the use of existing highly-optimised parallel MD code and their coupling to analysis tools based upon collective coordinates which do not require a priori knowledge of the system to bias. We describe two workflows which both couple large "ensembles" of relatively short MD simulations with analysis tools to automatically analyse the generated trajectories and identify molecular conformational structures that will be used on-the-fly as new starting points for further "simulation-analysis" iterations. One of the workflows leverages the Locally Scaled Diffusion Maps technique; the other makes use of Complementary Coordinates techniques to enhance sampling and generate start-points for the next generation of MD simulations. We show that the ExTASY tools have been deployed on a range of HPC systems including ARCHER (Cray CX30), Blue Waters (Cray XE6/XK7), and Stampede (Linux cluster), and that good strong scaling can be obtained up to 1000s of MD simulations, independent of the size of each simulation. We discuss how ExTASY can be easily extended or modified by end-users to build their own workflows, and ongoing work to improve the usability and robustness of ExTASY.**

I. Introduction and Motivation

Approximately 30-40% of compute cycles on US XSEDE [1], [2] is devoted to research on biomolecular systems using Molecular Dynamics (MD) simulations. Much of the computational cost comes from the need for an adequate sampling of the conformational space accessible to these complex and flexible systems in order to answer a particular research question. For example, to calculate free energies one needs an adequate sample from the Boltzmann weighted ensemble of states for the system in order to estimate the thermodynamic quantity of interest. Another example is the study of kinetic processes such as self-assembly or drug-target association, where the integration of data from large numbers of trajectories is required to build a statistically meaningful model of the dynamical process.

The high dimensionality of these macromolecular systems and the complexity of the associated potential energy surfaces (creating multiple metastable regions connected by high free energy barriers) pose significant challenges to adequately sample the relevant regions of the configurational space. In other words, beside the "curse of dimensionality" associated with the large number of degrees of freedom, MD trajectories can easily get "trapped" in a low free energy state and fail to explore other biologically relevant states. The waiting time to escape from a local free energy minimum increases exponentially with the height of the free energy barrier that needs to be crossed to reach another state. Metastable states separated by free energy barriers of several tens of $k_B T$ (where k_B is the Boltzmann constant and T is the physiological temperature) are not uncommon in biologically relevant systems, but can not at present be routinely sampled with standard MD simulations.

In practice, better sampling of the relevant regions of a macromolecule configurational space can be achieved through methodologies able to bias the sampling towards scarcely visited regions, reducing the waiting time inside a metastable state by artificially flattening the energy barriers between states - e.g. Metadynamics [3] or Accelerated Dynamics [4]. Although the results can be usually reweighed to reproduce the correct Boltzmann statistics, kinetics properties are not easily recovered from biased simulations (unless used in combinations with unbiased simulations, see e.g. [5]). In addition, the design of an effective bias usually requires some *a priori* information on the system of interest, for instance on a suitable choice of collective variables to describe slow timescale processes.

An alternative approach to tackle the sampling problem is the development of ensemble or swarm simulation strategies, where data from large numbers of simulations, which may be weakly coupled or not coupled at all, are integrated (e.g. Replica Exchange [6] and Markov State Models (MSMs) [7]).

This last class of methods is of increasing interest for a variety of reasons. Firstly, the hardware roadmap is now

based almost entirely on increasing core counts, rather than clock speeds. On the face of it, these developments favour weak scaling problems (larger and larger molecular systems to be simulated) over strong scaling problems (getting more data faster on a system of fixed size). However, by running ensembles of simulations over these cores and integrating the data using, e.g. MSM approaches, timescales far in excess of those sampled by any individual simulation are effectively accessed. In the last few years several studies have been published [8]–[12] where, using MSM methods, processes such as protein folding or ligand binding have been completely and quantitatively characterized (thermodynamically and kinetically) from simulations orders of magnitude shorter than the process of interest.

It is becoming increasingly clear that the application of ensemble simulation strategies on state-of-the-art computational facilities has an unparalleled potential to permit accurate simulations of the largest, most challenging, and generally most biologically relevant, biomolecular systems. The main challenge in the development of the ensemble approach for faster sampling of complex macromolecular systems is on the design of strategies to adaptively distribute the trajectories over the relevant regions of the systems configurational space, without using any *a priori* information on the system global properties. The definition of smart "adaptive sampling" approaches that can redirect computational resources towards unexplored yet relevant regions is currently of great interest.

In light of the challenge posed by trends in computer architecture, the need to improve sampling, and the range of existing MD codes and analysis tools, we have designed and implemented the Extensible Toolkit for Advanced Sampling and analYsis (ExTASY). ExTASY provides three key features within a single framework to enable the development and applications requiring advanced sampling in a flexible and highly scalable environment. Firstly, as an extensible toolkit, ExTASY allows a wide range of existing software to be integrated, leveraging the significant community investment in highly optimised and well-tested software packages and enabling users to continue to work with tools that they are familiar with. Support for specific MD codes and analysis tools is provided in order to demonstrate how ExTASY may be used, but users can easily add other tools as needed.

Secondly, ExTASY is flexible, providing a programmable interface to link individual software components together and construct sampling workflows. Workflows capture a sequence of execution of individual tools, and the data transfers and dependencies between them. First class support is provided for defining large ensembles of independent simulations. Thus complex calculations may be scripted and then executed without the need for user intervention.

Thirdly, ExTASY workflows may be executed either locally, or on remote High Performance Computing systems. Complexities such as the batch queueing system and data transfer are abstracted, making it easy for users to make use of the most appropriate compute resources they have access to. In addition, this abstraction allows workflows to be executed without exposing each component to queue waiting time, and respecting the dependencies defined in the workflow, for many simulations to be scheduled and executed in parallel.

The rest of the paper is organized as follows: In the next section we discuss the design and implementation of ExTASY. After a brief discussion in Section III of two distinct applications that have been used to design and validate ExTASY, Section IV provides a careful analysis of the performance and scalability of ExTASY. Given the complex interplay between functionality and performance when designing an extensible and production tool, we perform a wide range of experiments aimed to investigate strong and weak scaling properties, inter alia over a set of heterogeneous HPC platforms. We conclude with a discussion of the scientific impact as well as the lessons for sustainable software development.

II. Related Work

The need for better sampling has driven developments in methodology (algorithm), hardware, and software for (bio)molecular simulation.

One of the features of the popular Metadynamics method [3] or Accelerated Dynamics [4] is that a constant analysis of what has been sampled so far is used to bias future sampling into unexplored regions. A range of alternative approaches are now emerging that do likewise, but where alternating segments of data-gathering, and analysis to inform the direction of future sampling, are more coarsely grained. This iterative approach has the advantage over the metadynamics method that the identity of "interesting" directions for enhanced sampling does not need to be defined *a priori*, but can emerge and respond flexibly to the developing ensemble. Another advantage is that the MD-based sampling process and analysis method do not have to be implemented within the same executable, or in two tightly-coupled executables, permitting greater flexibility. Many such methods make use of collective variables (CVs) to define directions in which to promote sampling. A variety of novel, and established, algorithms for the unsupervised and adaptive construction of CVs exist. In addition to the work of Preto and Clementi [13], interleaving cycles of MD simulation with data analysis through Locally Scaled Diffusion Maps [14], related methods include the non-targeted PaCS-MD method of Harada and Kitao [15], variants thereof [16], and the PCA-based method of Peng and Zhang [17].

Better sampling can also come from faster sampling, which has been enabled through hardware developments such as ANTON [18] and MD-GRAPE [19]. These special purpose computers enable much faster calculations of the different contributions to the forces along the trajectories, thus speeding up the clock time required to perform a time integration step in the MD simulation and allowing execution of significantly longer MD trajectories. The cost of, and access to such special purpose computers ensure that in spite of their potential, they will not be as accessible for the wider scientific community as general purpose approaches. Further ANTON requires a customized ecosystem, from bespoke MD engines and ANTON specific data analysis middleware (e.g.,

2

HiMach). Thus ANTON style special-purpose approaches to bio-molecular simulation science cannot take advantage of the rich community driven advances and eco-system.

Methods such as Replica Exchange and Metadynamics require a tight coupling between the simulation and analysis processes, and are thus typically implemented as additional facilities within the core MD code (e.g. replica exchange methods are implemented in AMBER [20], CHARMM [21], GROMACS [22], LAMMPS [23], and NAMD [24], for example), or are provided by a separate package that communicates in a fine-grained manner with the running MD executable, generally though specially-installed "hooks"; an example of this approach is the PLUMED package [25] which provides metadynamics capabilities (amongst others) to AMBER, GROMACS, LAMMPS, NAMD and also Quantum ESPRESSO [26]. In contrast, there is, to our knowledge, so far no established and generally available package to support the types of coarser-grained, adaptive workflows described above.

III. EXTASY: REQUIREMENTS, DESIGN AND IMPLEMENTATION

In this section we first present the requirements that have been considered in the design and implementation of ExTASY, which we then go on to discuss.

A. Requirements

Consistent with the design of many new software systems and tools, we analyze the functionality, performance and usability requirements of ExTASY.

1) Functionality: Specific to sampling, there is a need to couple two very distinct computational stages; each stage can be short-lived when compared to the typical duration of monolithic simulation. Furthermore, the two stages differ in their resource requirements significantly: one stage is characterized by multiple compute intensive MD simulations, the other by a single analysis program that operates on data aggregated from multiple simulations. The "Ex" in ExTASY is a reference to the extensible nature of the framework, and thus any coupling must be between the abstraction of stages and not specific codes for fixed time duration.

Scientists may have access to multiple systems, and wish to submit jobs on each, or may be forced to migrate from using one system to another due to CPU time allocation, or system end-of-life. It is imperative that any software system support **interoperability**, i.e. use of heterogeneous systems with minimal changes. MD simulations may be executed over multiple nodes, depending on the system size, thus any software system should also support the **ability to execute tasks** over multiple nodes.

2) Performance: In order to obtain good sampling, a large number of simulations must be executed. In many cases, the aggregate number of cores required by the set of simulations ("ensemble") is much higher than the total number of cores that are available at a given instance or that can be allocated on a system. The framework must decouple the aggregate (or peak) resource requirement of the workload from the number of cores available or utilized. On the other hand, if access to a large number of cores is available, the framework should be able to use them effectively. In this regard, the **strong and weak scalability** of the framework is to be investigated.

3) Usability: Depending on the application, the user might need to change to using a larger/smaller set of input data, modify simulation parameters or replace any of the simulation or analysis tools. The framework should offer **easy application setup**, minimizing the user's time and effort in the process. The user should only be concerned with decisions on "what" the workflow is and "where" it is being executed. The details as to "how" the deployment and execution occurs should be hidden by the underlying software. Thus the framework should use tools that **abstract complexities** of deployment and execution from the user. Workflow users and developers should be able to concentrate their efforts on the application logic and expect the underlying software to provide a level of transparent and automation of aspects such as data movement and job submission.

B. Design

From these requirements, we identify the following as the primary design objectives of ExTASY.

1) Support range of HPC systems, abstract task execution, data movement from the user.
2) Flexible resource management and coupling capabilities between different stages as well as within a stage.
3) Provide the users with an easy method to specify or change workload parameters without delving into the application itself.

These design objectives put together lead to the following simple software architecture:

1) Middleware for resource and execution management: The ExTASY framework is aimed to provide an easy method to run advanced sampling algorithms on HPC systems. In this process, the ExTASY framework should abstract users from the complexity of application composition, workload execution and resource management. There is need for a middleware that provides such resource and execution management that provides many of the required functionalities and performance. The details of how tasks are mapped or executed on to resources is abstracted from the ExTASY user. The user is only required to provide details of the workload and identify the resource(s) that are accessible. This design choice thus acknowledges the separation of concerns: workload description from its execution on HPC systems.

2) Configuration files: Composing the advanced sampling algorithms discussed in the previous sections using the components provided by a particular middleware stack, requires specific knowledge of the middleware itself. The ExTASY framework bypasses this complexity by adding one more level of abstraction. It provides ready-to-use scripts in accordance with advanced sampling methods discussed in this paper that are configurable via configuration files. In these configuration files, the user is exposed to only application-level, meaningful parameters that can be modified as necessary.
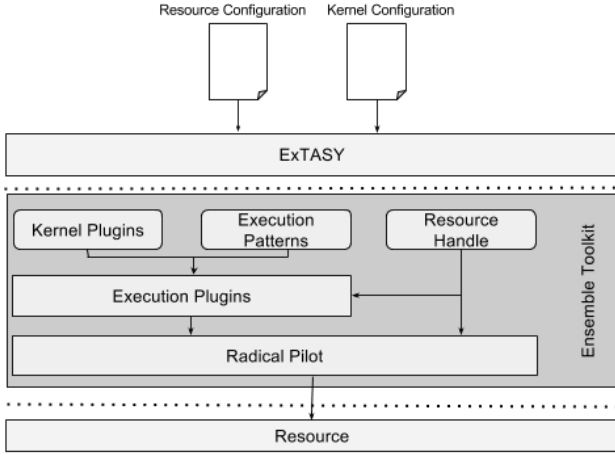
Fig. 1. Design of the ExTASY framework using Ensemble Toolkit as middleware. The ExTASY framework provides ready-to-use scripts created using components provided by Ensemble Toolkit. Parameters related to the resource and workload are exposed via configuration files, which alone are the files that users interact with. Within Ensemble Toolkit, the workload is converted into executable units by the execution plugins and submitted to the resource using RADICAL-Pilot.

C. Implementation

1) Ensemble Toolkit: As mentioned previously, ExTASY requires a middleware for resource and execution management. We chose to use Ensemble Toolkit [27] as the middleware component as it provides several relevant features such as the ability to support MPI tasks, dynamic resource management – one type of which is to be able to execute more tasks than the resources available, support for heterogeneous HPC systems and strong and weak scalability guarantees. Ensemble Toolkit has been tested upto O(1,000) tasks with short and long term plans to support O(10,000) and O(100,000) tasks [27]. Ensemble Toolkit is in turn based upon the pilot abstraction (and the RADICAL-Pilot [28] implementation of the pilot abstraction) to provide much of the flexible and scalable resource management capabilities.

Ensemble Toolkit exposes three components to the user that can be used to express many applications: Kernel Plugins, Execution Patterns, and Resource Handle. Scripts that are part of ExTASY framework use these components to describe the application logic.

2) Configuration files: The application logic is expressed via components of Ensemble Toolkit. The resource and the workload specifications are exposed via configuration files. The ExTASY framework has two types of configuration files: (i) resource configuration, which consist of details of the resource where the application will to be executed such as the resource name, the runtime, and the username and account details used to access the resource, and (ii) kernel configuration, which defines the workload parameters such as the location of input files for the Molecular Dynamcis simulation and analysis tools, parameters for the tools, and workflow parameters such as the number of simulations.
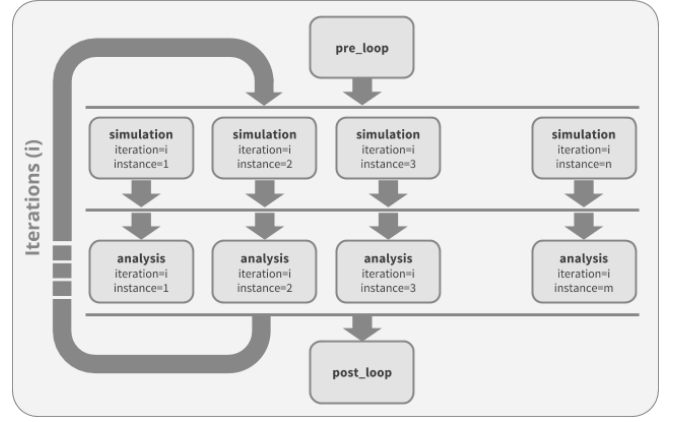

Fig. 2. The SAL pattern common to both sampling algorithms. The crux of the pattern is an iteration over 2 stages: simulation and analysis, where the number of simulation and analysis instances can be different. The pattern may also contain pre- and post- processing stages.

IV. APPLICATIONS

We illustrate the capabilities of the ExTASY approach via two exemplar applications. The two different advanced sampling algorithms implemented with ExTASY are the Diffusion Map-directed-MD (DM-d-MD) and CoCo-MD techniques. Both these algorithms have a common execution pattern: an ensemble of simulation tasks followed by an analysis stage, performed for multiple iterations following the pattern shown in Figure 2.

In the case of the DM-d-MD algorithm, the simulation stage consists of Gromacs and LSDMap in the analysis stage. Whereas, in the CoCo algorithm, the simulation stage consists Gromacs runs and trajectory conversions and analysis consists of CoCo. The individual simulation or analysis tools might differ depending on the algorithm chosen but the overall pattern is observed to be the same.

A. Diffusion Map-directed-MD

The Diffusion Map-directed-MD (DM-d-MD) technique [13] improves the efficiency of computational resources by choosing which replicas of the protein are used to run MD. When replicas are too close to each other, the MD trajectories will be similar. The information gain from simulating MD with close replicas is small. Part of the replicas which are too close to each other are deleted. To hold the total number of replicas constant, replicas which are too far apart from each other are duplicated. In DM-d-MD, a non-linear dimensionality reduction technique, the locally scaled diffusion map (LSDMap) [14] is used to calculate the distance between different replicas. The deletion or duplication of replicas would destroy the correct sampling of the protein. By changing the weights of individual replicas in the reweighting step, the correct sampling of the protein is obtained.

The DM-d-MD technique requires only the protein starting structure. No additional information about the protein is necessary. The user can fine tune the sampling mainly by varying the total number of replicas and the way how the local scale

in LSDMap is calculated. At the begin on the method, the replicas are generated from the protein starting structure. After the MD step, the LSDMap is calculated. LSDMap requires only the final structure for each replica from the MD step. Based on the LSDMap results new replicas for the next iteration of DM-d-MD are chosen from the current replicas. The reweighting ensures that the

It was shown that DM-d-MD technique is, at least, one order of magnitude faster compared to plain MD [13]. This comparison was done for alanine dipeptide and a 12-aminoacid model system, Ala12.

B. The CoCo-MD workflow

The CoCo (**Co**mplementary **Co**ordinates) technique [29] was designed originally as a method to enhance the diversity of ensembles of molecular structures of the type produced by NMR structure determination. The method involves the use of PCA [30]–[32] in Cartesian space to map the distribution of the ensemble in a low (typically 2-4 dimensional) space, and then the identification of un-sampled regions. CoCo generates new conformations for the molecule that would correspond to these un-sampled regions. The number of new structures generated is under the users control the algorithm divides the space into bins at a chosen resolution, marks bins as sampled or not, first returns a structure corresponding to the centre of the un-sampled bin furthest from any sampled one, marks this bin as now sampled, and iterates as many times as desired.

In the CoCo-MD workflow, an ensemble of structures from MD simulations are analysed using the CoCo method; new conformations become the start points for a new round of MD simulations. The latest MD data is added to the previous set, and CoCo repeated. The method is agglomerative - all MD data generated so far is used for each analysis; but also adaptive - a fresh PCA is performed each time. Applied to simulations of the alanine pentapeptide, the CoCo-MD workflow is able to reduce mean first passage times from the extended state to other local minimum states by factors of ten or greater compared to conventional simulations [33].

V. PERFORMANCE EVALUATION

A. Experiment setup

1) Physical system: The 39-residue mixed α/β protein NTL9(1-39) (pdb code 2HBA, 14,100 atoms including water) is chosen as the physical system for our experiments. NTL9 has an experimentally measured folding time of around 1.5 ms [34], and its folding process has been extensively studied by experiment and all-atom MD simulations, both by means of the Folding@Home distributed computing platform coupled with MSM analysis [9], and by Anton supercomputer [35].

The relatively small size of NTL9, and the existence of previous MD simulation results over long timescales, make this protein an ideal candidate for testing and benchmarking our approach. Albeit small, NTL9 is much larger than a simple peptide, and exhibits a folding process with two competing routes [35], thus presenting a non-trivial test for adaptive sampling.

2) HPC systems used: One of the requirements of ExTASY as that it should be interoperable, so we have used several different HPC systems for our experiments, and characterised the performance of ExTASY on each.

Stampede is a Dell Linux cluster located at the Texas Advanced Computing Center, and is part of the Extreme Science and Engineering Discovery Environment (XSEDE). It consists of 6400 compute nodes, each with 2 Intel Xeon 'Sandy Bridge' processors, for a total of 16 CPU cores per node, as well as an Intel Xeon Phi co-processor (not used in our experiments). Stampede uses the SLURM batch scheduler for job submission.

ARCHER is a Cray XC30 supercomputer hosted by EPCC, and operated on behalf of the Engineering and Physical Sciences Research Council (EPSRC) and the Natural Enviroment Research Council (NERC). It has 4920 compute nodes, each with 2 Intel Xeon 'Ivy Bridge' processors, giving 24 cores per node. ARCHER uses the Portable Batch System (PBS) for job submission.

Blue Waters is a Cray XE6/XK7 operated by the National Centre for Supercomputing Applications on behalf of the National Science Foundation and the University of Illinois. The XE6 partition used in this work consists of 22640 compute nodes with 2 AMD 'Interlagos' processors, giving 32 cores per node. Blue Waters uses the TORQUE/Moab workload manager for job submission.

B. Evaluation of individual components

Since the performance of the entire workflow depends on the performance of each of the component parts, we investigate the scaling of both the simulation code (Gromacs) and the analysis tools in isolation on each of the three target platforms, using the NTL9 system used for the full sampling workflows.

1) Simulation tools: The parallel efficiency of Gromacs with respect to a single core on each machine is shown in Figure 3. While efficincies of 69% (ARCHER, 24 cores), 78% (Stampede, 16 cores) and 46% (Blue Waters, 32 cores) suggest that while the scaling for such a relatively small simulation is not ideal, using a single node per simulation is a good use of the available hardware. Beyond a single node, the efficiency drops off so although multiple node simulation tasks are supported by Ensemble Toolkit they are not useful for this benchmark case.

2) Analysis tools: Due to the nature of the two workflows, there are many parallel simulation tasks, but only a single analysis task. Therefore, the analysis task may be configured to run on as many cores as are available to the simulations. Both CoCo and LSDMap are parallelised using MPI, and consist of parts which are independent, e.g., reading of trajectory files in CoCo, and involve communication e.g. diagonalisation of the covariance matrix in CoCo and the diffusion matrix in LSDMap, so the parallel scaling is expected to be sub-linear. The performance of CoCo is also strongly dependent on I/O since it reads the entire trajectory file rather than just the final configurations like LSDMap.
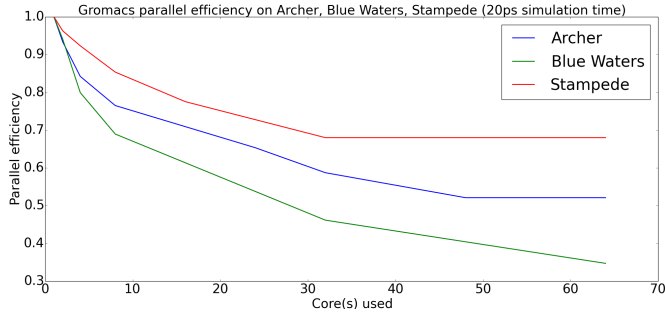
Fig. 3. Gromacs parallel efficiency on ARCHER, Blue Waters and Stampede. A single 20ps gromacs simulation of the NTL9 system is performed using various core counts on the three machines and the execution time is measured.
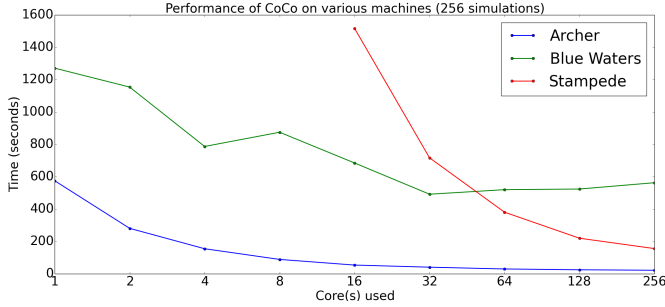

Fig. 4. Strong scaling of CoCo analysis tool on ARCHER, Blue Waters and Stampede. A total of 256 simulations are analyzed using various core counts on the three machines and execution time is measured.
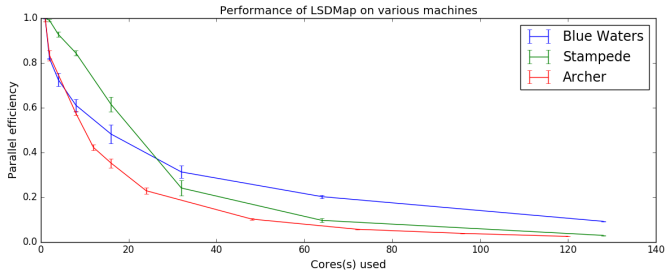

Fig. 5. Parallel efficiency of LSDMap on ARCHER, Blue Waters and Stampede. A total of 2449 structures are analyzed using various core counts on the three machines and execution time is measured.

Figures 4 show the strong scaling of CoCo for a fixed input of 256 simulations. We see that CoCo is able to scale to at least 256 cores on ARCHER and Blue Waters, and to around 32 cores on Stampede, thus for our following experiments we configure the workflow to run CoCo with as many cores as there are input trajectories. LSDMap (Figure 5) however, does not scale efficiently much beyond a single node on each machine, even with over 2000 input structures. Nevertheless, we run LSDMap on as many cores as are available, even though it cannot use them fully. Due to the structure of the workflow, those cores would otherwise sit idle during the analysis phase.
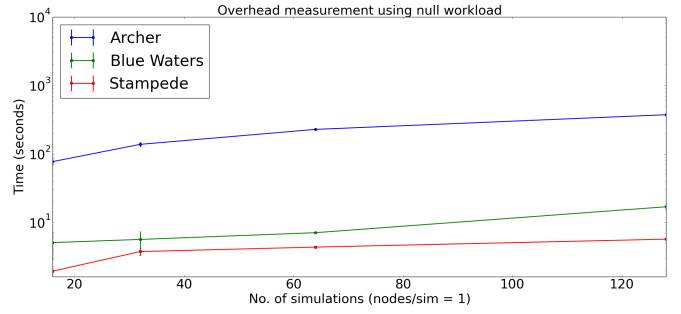
C. Evaluation of ExTASY

Fig. 6. Measurement of overhead from ExTASY on ARCHER, Blue Waters and Stampede. Using only the simulation stage of the SAL pattern, overhead from ExTASY is measured on the three machines at various core counts.

1) Characterization of overheads: In addition to the necessity of characterizing performance overhead introduced by a new system approach and implementation, in order to instill confidence in potential users of ExTASY it is important to measure the overheads imposed by ExTASY. The objective is to discern the contributions from the different aspects of the ExTASY framework as opposed to MD and analysis components. The time taken to create, launch and terminate all of the simulations or the instantiate ExTASY framework itself on the resource are examples of the former overhead. All of the experiments use Ensemble Toolkit version 0.3.14.

We ran a single iteration of the workflow with null workloads, i.e., where the task did no work (`/bin/sleep 0`), but otherwise was configured as a "normal" simulation task, launched using MPI and taking a whole node on each of the machines. The number of tasks ranged from 16 to 128, and they were all run concurrently. Figure 6 shows that the overheads on Stampede and Blue Waters are relatively small, growing from <5s to around 15s for 128 tasks. On ARCHER the overheads are much larger (70-350s), which after further investigation is due to the `aprun` job launcher, which performs poorly when multiple concurrent tasks are launched.

2) Strong scaling test: To test the strong scaling of the ExTASY workflows, we fix the number of executing instances (independent MD simulations) to 128 and vary the total number of CPU cores used to execute the workflow. The largest experiments use enough CPU cores that all of the MD simulations execute concurrently. For example, on ARCHER, 128 instances, each executing on a single node (24 cores) gives a maximum of 3072 cores.

Since the MD simulations are independent and may be executed concurrently by ExTASY, we expect that as the number of cores is increased, the simulation time should decrease proportionally. The time spent in the analysis part is expected to be constant since the total amount of MD data is constant, and despite the parallelisation of the analysis tools as can be seen from Figure 5, the time to completion plateaus at fairly low core counts.

Figure 7 (**bottom**) shows the results of this experiment for the DM-d-MD workflow executed on Blue Waters. The
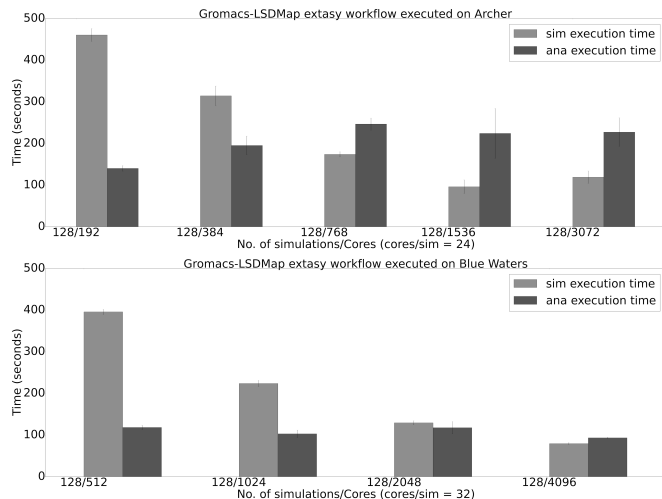
6

Fig. 7. Strong scaling of DM-d-MD workflow on ARCHER (**top**) and Blue Waters (**bottom**). The number of simulations is held constant at 128, number of cores per simulation at 24 on ARCHER and 32 on Bluewaters. The total number of cores used is varied with a constant workload, hence measuring strong scaling performance of the framework.
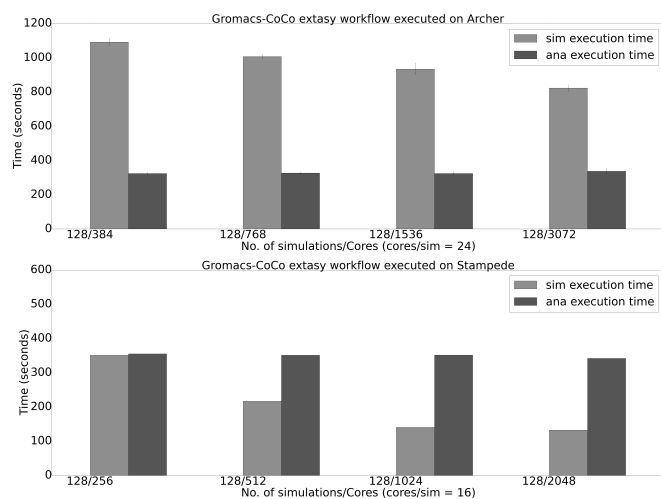

Fig. 8. Strong scaling of CoCo-MD workflow on ARCHER (**top**) and Stampede (**bottom**). The number of simulations is held constant at 128, number of cores per simulation at 24 on ARCHER and 16 on Stampede. The total number of cores used is varied with a constant workload, hence measuring strong scale performance of the framework.

simulation time decreases from 395.7s on 512 cores to 79.52s on 4096 cores, a speedup of 4.97x with 8x as many cores – yielding a scaling efficiency of 62%. The analysis time is essentially constant at around 100s, as expected. The loss of scaling efficiency for the simulation part comes from two sources. Firstly, there is the fixed overhead discussed in Section V-C1 associated with the execution of 128 concurrent tasks, which is approximately 15s. Secondly, the actual computation which occurs within each task take longer when more simulations are run concurrently, due to the fact that they all write to the same shared filesystem. For example, when 16 instances are run concurrently on 512 cores, the MD simulations take an average of 45.6s each. When all 128 instances are run concurrently, each takes 49.0s, or 3.4s slower. If these effects are removed, the effective scaling efficiency on 4096 cores rises to 77%.

Similar results are obtained on ARCHER (Figure 7, **top**), although the scaling of the simulation part tails off on 3072 cores, and the LSDMap analysis takes somewhat longer, with higher variability than Blue Waters. Both of these are due to the fact that both the MD and analysis involve significant I/O, and it is known that opening many small files concurrently is slow as the metadata servers of the parallel Lustre filesystem become a bottleneck [36].

CoCo-MD on Stampede (Figure 8, **bottom**) has similar strong scaling for the simulation part as DM-d-MD. The simulation time decreases from 363s on 256 cores to 83.7s on 2048 cores – a speedup of 4.3x for a 8-fold increase in the number of cores (54% efficiency). However, the analysis time (CoCo) does not scale due to the fact that the parallelisation in CoCo is limited to the number of input trajectories, which is 128 in this case, even if more cores are available.

The CoCo-MD workflow on ARCHER (Figure8, **top**) does not show as good scaling as DM-d-MD, or CoCo-MD on

the other platforms. The reason for this lies in the fact that after the actual molecular dynamics calculation, a 'trajectory conversion' step is required to prepare the data for CoCo. This step only takes a fraction of a second to execute, but there is a very large overhead caused by `aprun`, which allocates resources to and launches each individual task. This does not occur on Blue Waters, which uses the ORTE [37] implementation of RADICAL-Pilot that is not yet default on ARCHER.

3) Weak scaling test: To investigate the weak scaling properties of ExTASY, we fix the ratio of number of instances to CPU cores, and vary the number of instances with the constraint that all simulations can execute concurrently. For example, on ARCHER 16 instances are executed on one node each (24 cores) giving a total of 384 cores, and the number of instances is increased to 128, i.e., the number of cores is 3072.

Since all simulations run concurrently, and the length of each simulation does not change, we expect the simulation time to be a constant. However, the analysis part will increase since the performance of the analysis tools is a function both of the input data size (depending on the number of instances), as well as the number of cores available, and even though the number of cores is proportional to the data size, the amount of work grows faster than linearly with the data size.

For the DM-d-MD workflow on Blue Waters (Figure 9, **top**) we observe a small increase of 21.8s in the simulation part as we scale from 512 to 4096 cores. Similar to the strong scaling results, this is combination of the overhead due to the increased number of tasks with a slowdown of the individual tasks themselves. The analysis is found to increase sub-linearly. As discussed in section V-B2 the LSDMap computation consists of parts which are both linear and quadratic in the size of the input data. Combined with an increasing number of cores
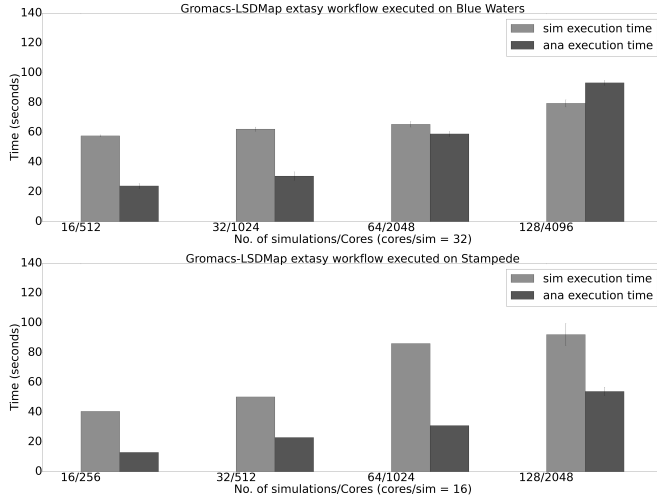
Fig. 9. Weak scaling of DM-d-MD workflow on Blue Waters (**top**) and Stampede (**bottom**). The number of cores per simulation is held constant at 32 on Blue Waters and 16 on Stampede. The total number of simulations is varied from 16-128 and the cores used are increased proportionally. By keeping the ratio of the workload to the number of resources constant, we observe the weak scaling performance of the framework.
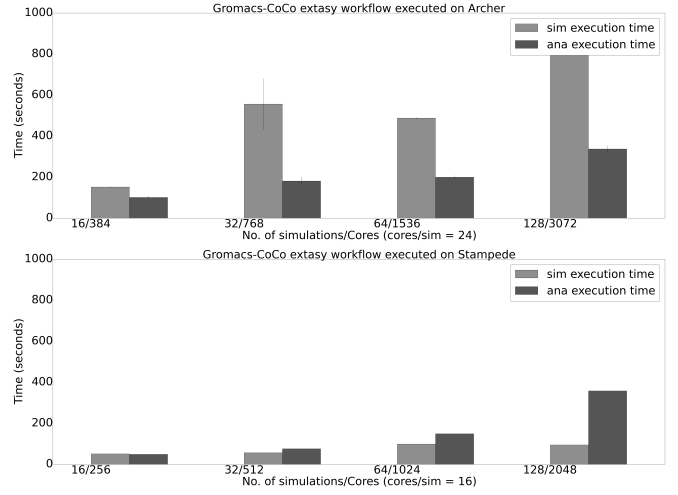

Fig. 10. Weak scaling of CoCo-MD workflow on ARCHER (**top**) and Stampede (**bottom**). The number of cores per simulation is held constant at 24 on ARCHER and 16 on Stampede. The total number of instances is varied from 16-128 and the cores used are increased proportionally. By keeping the ratio of the workload to the number of resources constant, we observe the weak scaling performance of the framework.
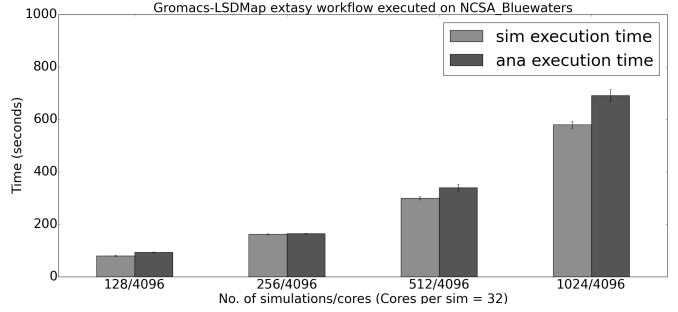

Fig. 11. DM-d-MD workflow on Stampede. Workload is increased from 128 instances to 1024, keeping the number of cores constant at 4096. Within the overheads, the increase in execution time is in proportion with increase in the workload.

available for the tool, sub-linear scaling is the result. Similar behaviour is observed on Stampede (Figure 9, **bottom**), although with a different weighting of the simulation and analysis part, reflecting the fact that the performance of each kernel depends on how well optimised the application binary is on the execution platform.

The weak-scaling of CoCo-MD on ARCHER (Figure 10, **top**) shows very clearly the aprun bottleneck discussion in Section V-C2, and the effect increases as the number of concurrent tasks grows. However, the analysis part scales better than linearly, which is to be expected since CoCo consists of parts which weak scale ideally (independent operations per trajectory file) and parts such as the construction and diagonalisation of the covariance matrix which grow as the data size squared or more.

On Stampede, the weak scaling of the simulation part of the CoCo-MD workflow (Figure 10, **bottom**) is much better than ARCHER. The simulation time grows only by around 50s compared to over 700s on ARCHER over the range of cores that we tested. CoCo scales almost identically to ARCHER.

4) Effect of larger ensembles: To distinguish the effects caused by strong scaling (increasing parallelism with a fixed amount of work) and weak scaling (increasing parallelism proportionally to the amount of work), we also measured the effect of increasing the amount of work with a fixed number of compute cores available. Figure 11 shows the results for the DM-d-MD workflow running on Blue Waters as we vary the number of MD instances from 128 to 1024, keeping the total number of cores available at 4096. Since each task runs on a single node (32 cores per instance), only 128 simulation tasks can run concurrently. Ideally, we would expect that the simulation time should increase linearly with the number of instances. In practice, we see that the time taken grows by

only 7.4x as the number of instances increases from 128 and 1024 i.e., a factor of 8. This is due to the fact that some of the overheads related to managing the tasks that occur before or after execution in the 128 task case are one-time overheads, i.e., those overheads are hidden as they are done concurrently (in the RADICAL-Pilot Agent) with the execution of the remaining tasks when the number of instances is greater than 128. The scaling of the analysis part is consistent with that discussed in Section V-B2, that there is a close to linear scaling since the larger the ensemble size, the more parallelism is available in LSDMap

5) Dynamic simulations: An important characteristic of the LSDMap and CoCo based workflows is that the number of instances typically changes after each simulation-analysis iteration. Thanks to the pilot-abstraction, the ExTASY framework supports flexible mapping between the number of concurrent instances and the total number of cores, while being agnostic of the number of cores per instance. This functionality is used by the DM-d-MD workflow, where, depending on the progress
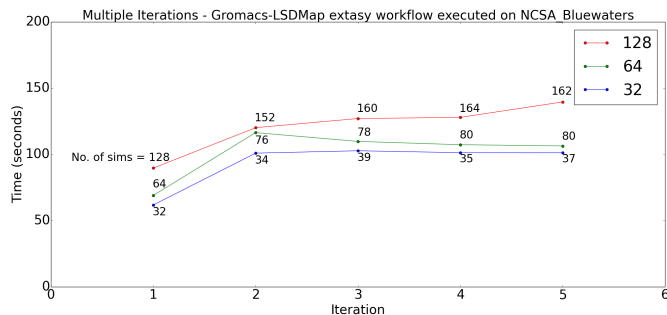
Fig. 12. Support for dynamic workload in ExTASY: The DM-d-MD algorithm dictates the number of instances at every iteration. The number of instances in each iteration (for a total of 5 iterations) when starting with 32, 64, 128 instances is presented.

through the conformation space of the system being explored, LSDMap may decide to spawn more (or less) trajectories for the next iteration of sampling. Figure 12 illustrates this capability. We ran the DM-d-MD workflow on Blue Waters for three configurations with 32, 64 and 128 initial instances. We can see that after an initial growth phase the number of instances seems to stablise for the remaining iterations, although the difference from the starting configuration and the number of iterations taken to stabilise is not algorithmically or systematically predictable. The flexible resource utilization capabilities that ExTASY is built upon prove critical.

D. Summary of Experiments

We have shown illustrative performance data for two different applications – CoCo-MD and DM-d-MD – based on different analysis methods, on three distinct HPC platforms - ARCHER, Blue Waters and Stampede. The overall scaling to O(1000) simulations is clearly demonstated, and we have analysed the scaling behaviour of the ExTASY framework itself (overheads) and the individual simulation and analysis programs which constitute the workflow.

VI. DISCUSSION AND CONCLUSION

State-of the-art computational biophysics approaches aim to balance three different requirements: force-fields accuracy, advanced sampling capabilities, and rigorous and fast data analysis. These points are strongly interconnected. In particular, it is becoming clear that advanced sampling and data analysis need to be tightly coupled to work efficiently and accurately, as the configurational space that has already been sampled needs to be analyzed on-the-fly to inform on how to proceed with further sampling. Furthermore, many advanced sampling algorithms for biomolecular simulations require flexible and scalable support for multiple simulations. As no final solution yet exists on the best strategy for adaptive sampling (and different physical systems may require a combination of strategies), there is a need to allow combination of different MD engines with different analysis tools, and can be easily extended or modified by end-users to build their own workflows, both for development of new strategies, and for applications to the study of complex biomolecular systems.

ExTASY is designed and implemented to provide a significant step in this direction. ExTASY allows to simulate many parallel MD trajectories (by means of standard MD engines), to extract long timescale information from the trajectories (by means of different dimensionality reduction methods), and to use the information extracted from the data for adaptively improving sampling. ExTASY has been utilized by different MD engines and analysis algorithms, with only pre-defined and localized changes.

In Section III, we formally identified the functional, performance and usability requirements to support the coupling of MD simulations with advanced sampling algorithms. We then presented the design and implementation of ExTASY, an *extensible*, *portable* and *scalable* Python framework for building advanced sampling workflow applications to achieve these requirements. After establishing accurate estimates of the overhead of using ExTASY, Section V consisted of experiments designed to validate the design objectives; we performed experiments that characterized ExTASY along traditional scaling metrics, but also investigated ExTASY beyond single weak and strong scaling performance. With the exception of some machine specific reasons,ExTASY displayed linear scaling for both strong and weak scaling tests on various machines up to O(1000) simulation instances on up to O(1000) nodes for both DM-d-MD and CoCo-MD workflows.

In order to keep the footprint of new software small, ExTASY builds upon well-defined and understood abstractions and their efficient and interoperable implementation (Ensemble Toolkit, RADICAL-Pilot). This provides double duty: The core functionality of ExTASY can be provided by simple higher level extensions of complex system software, while allowing it to build upon the performance and optimization of the underlying system software layers. This also allows ExTASY to employ good systems engineering practice: well-defined and good base performance, while being amenable to platform-specific optimizations (e.g. using ORTE on Blue Waters [37]).

The design of ExTASY to reuse existing capabilities, for extensibility to different MD codes and sampling algorithms while providing well defined functionality and performance are essential features to ensure the sustainability of ExTASY. Compared to existing software tools and libraries for advanced sampling, ExTASY provides a much more flexible approach that is agnostic of individual tools and compute platforms, is architected to enable efficient and scalable performance and has a simple but general user interface. The ExTASY toolkit is freely available from http://www.extasy-project.org.

The ExTASY toolkit has been used to deliver two hands-on computational science training exercises and tutorials to the bio-molecular simulations community with a focus on advanced sampling. Participants were given the opportunity to utilize HPC systems in real time for advanced sampling problems of their own. Details of both events can be found at http://extasy-project.org/events.html#epccmay2016. A link to the lessons and experience from the first workshop can be found at: https://goo.gl/nMSd27.

VII. ACKNOWLEDGMENTS

REFERENCES

[1] "NSF XSEDE Annual Report (2012)," page 382, Figure 32 https://www.xsede.org/documents/10157/169907/2012+Q2+Annual+Report.pdf.

[2] Using XDMoD to facilitate XSEDE operations, planning and analysis XSEDE '13 Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, doi 10.1145/2484762.2484763.

[3] A. Barducci, M. Bonomi, and M. Parrinello, "Metadynamics," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1, no. 5, pp. 826–843, 2011. [Online]. Available: http://dx.doi.org/10.1002/wcms.31

[4] L. C. Pierce, R. Salomon-Ferrer, C. A. F. de Oliveira, J. A. McCammon, and R. C. Walker, "Routine access to millisecond time scale events with accelerated molecular dynamics," *Journal of Chemical Theory and Computation*, vol. 8, no. 9, pp. 2997–3002, 2012, pMID: 22984356. [Online]. Available: http://dx.doi.org/10.1021/ct300284c

[5] H. Wu, F. Paul, C. Wehmeyer, and F. Noé, "Multi-ensemble Markov models of molecular thermodynamics and kinetics," *Proc. Natl. Acad. Sci. USA*, vol. in press, 2016.

[6] Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics method for protein folding," *Chemical Physics Letters*, vol. 314, no. 12, pp. 141 – 151, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009261499011239

[7] J. D. Chodera and F. Noé, "Markov state models of biomolecular conformational dynamics," *Current Opinion in Structural Biology*, vol. 25, pp. 135 – 144, 2014, theory and simulation / Macromolecular machines. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959440X14000426

[8] F. Noé, C. Schtte, E. Vanden-Eijnden, L. Reich, and T. R. Weikl, "Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 19 011–19 016, 2009. [Online]. Available: http://www.pnas.org/content/106/45/19011.abstract

[9] V. A. Voelz, G. R. Bowman, K. Beauchamp, and V. S. Pande, "Molecular simulation of ab initio protein folding for a millisecond folder NTL9(1-39)," *J. Amer. Chem. Soc.*, vol. 132, no. 5, pp. 1526–1528, 2010.

[10] I. Buch, T. Giorgino, and G. De Fabritiis, "Complete reconstruction of an enzyme-inhibitor binding process by molecular dynamics simulations," *Proc. Natl. Acad. Sci. USA*, vol. 108, no. 25, pp. 10 184–10 189, 2011. [Online]. Available: http://www.pnas.org/content/108/25/10184.abstract

[11] S. Gu, D.-A. Silva, L. Meng, A. Yue, and X. Huang, "Quantitatively characterizing the ligand binding mechanisms of choline binding protein using markov state model analysis," *PLoS Comput. Biol.*, vol. 10, no. 8, pp. 1–11, 08 2014. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pcbi.1003767

[12] N. Plattner and F. Noé, "Protein conformational plasticity and complex ligand-binding kinetics explored by atomistic simulations and markov models," *Nat. Commun.*, vol. 6, 2015.

[13] J. Preto and C. Clementi, "Fast recovery of free energy landscapes via diffusion-map-directed molecular dynamics," *Phys. Chem. Chem. Phys.*, vol. 16, no. 36, p. 19181, 2014.

[14] M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, "Determination of reaction coordinates via locally scaled diffusion map," *J. Chem. Phys.*, vol. 134, March 2011.

[15] X. D. Guo et al., "Computational studies on self-assembled paclitaxel structures: Templates for hierarchical block copolymer assemblies and sustained drug release," *Biomaterials*, vol. 30, no. 33, pp. 6556 – 6563, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142961209008503

[16] R. Harada, Y. Takano, T. Baba, and Y. Shigeta, "Simple, yet powerful methodologies for conformational sampling of proteins," *Phys. Chem. Chem. Phys.*, vol. 17, pp. 6155–6173, 2015. [Online]. Available: http://dx.doi.org/10.1039/C4CP05262E

[17] J. Peng and Z. Zhang, "Simulating large-scale conformational changes of proteins by accelerating collective motions obtained from principal component analysis," *Journal of Chemical Theory and Computation*, vol. 10, no. 8, pp. 3449–3458, 2014, pMID: 26588312. [Online]. Available: http://dx.doi.org/10.1021/ct5000988

[18] D. E. Shaw et al., "Anton, a special-purpose machine for molecular dynamics simulation," *Commun. ACM*, vol. 51, no. 7, pp. 91–97, Jul. 2008.

[19] I. Ohmura et al., "MDGRAPE-4: a special-purpose computer system for molecular dynamics simulations," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2021, 2014. [Online]. Available: http://rsta.royalsocietypublishing.org/content/372/2021/20130387

[20] R. Salomon-Ferrer et al., "An overview of the Amber biomolecular simulation package," *WIREs Comput. Mol. Sci.*, vol. 3, no. 2, pp. 198–210, 2013.

[21] B. R. Brooks et al., "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations," *J. Comput. Chem.*, vol. 4, no. 2, pp. 187–217, 1983.

[22] M. J. Abraham et al., "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 12, pp. 19 – 25, 2015.

[23] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *J. Comp. Phys.*, vol. 117, no. 1, pp. 1–19, 1995.

[24] J. C. Phillips et al., "Scalable molecular dynamics with NAMD," *J. Comput. Chem.*, vol. 26, no. 16, pp. 1781–1802, 2005.

[25] G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, "PLUMED 2: New feathers for an old bird," *Comp. Phys. Comm.*, vol. 185, no. 2, pp. 604–613, 2014.

[26] P. Giannozzi et al., "Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials," *J. Phys. Condens. Matter*, vol. 21, no. 39, p. 395502, 2009.

[27] V. Balasubramanian, A. Treikalis, O. Weidner, and S. Jha, "Ensemble Toolkit: Scalable and Flexible Execution of Ensembles of Tasks," 2016, (accepted ICPP 2016) http://arxiv.org/abs/1602.00678.

[28] A. Merzky et al., "Executing Dynamic and Heterogeneous Workloads on Super Computers," 2016, http://arxiv.org/abs/1512.08194.

[29] C. A. Laughton, M. Orozco, and W. Vranken, "COCO: A simple tool to enrich the representation of conformational variability in NMR structures," *Proteins*, vol. 75, no. 1, pp. 206–216, April 2009.

[30] I. T. Jolliffe, *Principal component analysis*. New York: Springer Verlag, 2002.

[31] E. C. Sherer, S. A. Harris, R. Soliva, M. Orozco, and C. A. Laughton, "Molecular dynamics studies of DNA A-tract structure and flexibility." *J Am Chem Soc*, vol. 121, pp. 5981–5991, 1999.

[32] S. T. Wlodek, T. W. Clark, L. R. Scott, and J. A. McCammon, "Molecular dynamics of acetylcholinesterase dimer complexed with tacrine." *J Am Chem Soc*, vol. 119, pp. 9513–9522, 1997.

[33] A. Shkurti, E. Rosta, V. Balasubramanian, S. Jha, and C. Laughton, "CoCo-MD: A Simple and Effective Method for the Enhanced Sampling of Conformational Space," 2016, unpublished, manuscript in preparation.

[34] J.-C. Horng, V. Moroz, and D. P. Raleigh, "Rapid cooperative two-state folding of a miniature $\alpha\beta$ protein and design of a thermostable variant," *J. Mol. Biol.*, vol. 326, no. 4, pp. 1261–1270, 2003.

[35] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, "How fast-folding proteins fold," *Science*, vol. 334, no. 6055, pp. 517–520, 2011.

[36] D. Henty, A. Jackson, C. Moulinec, and V. Szeremi, "Performance of Parallel IO on ARCHER," 2015. [Online]. Available: http://archer.ac.uk/documentation/white-papers/parallelIO/ARCHER_wp_parallelIO.pdf

[37] M. Santcroos et al., "Executing dynamic heterogeneous workloads on Blue Waters with RADICAL-Pilot," *Proceedings of the Cray User Group (CUG) 2016*, 2016. [Online]. Available: http://www2.epcc.ed.ac.uk/~ibethune/files/RP-ORTE-CUG2016.pdf