Runtime Steering of Molecular Dynamics Simulations Through In Situ Analysis and Annotation of Collective Variables

Silvina Caino-Lores
scaino-lores@acm.org
Department of Electrical Engineering and Computer Science, University of Tennessee-Knoxville
USA

Michel A. Cuendet
mac2109@med.cornell.edu
Department of Physiology and Biophysics, Cornell University
USA

Jack Marquez
jmarquez@utk.edu
Department of Electrical Engineering and Computer Science, University of Tennessee-Knoxville
USA

Ekaterina Kots
edk4002@med.cornell.edu
Department of Physiology and Biophysics, Cornell University
USA

Trilce Estrada
trilce@unm.edu
Department of Computer Science, University of New Mexico
USA

Ewa Deelman
deelman@isi.edu
Information Sciences Institute, University of Southern California
USA

Harel Weinstein
haw2002@med.cornell.edu
Department of Physiology and Biophysics, Cornell University
USA

Michela Taufer
taufer@acm.org
Department of Electrical Engineering and Computer Science, University of Tennessee-Knoxville
USA

ABSTRACT

This paper targets one of the most common simulations on petascale and, very likely, on exascale machines: molecular dynamics (MD) simulations studying the (classical) time evolution of a molecular system at atomic resolution. Specifically, this work addresses the data challenges of MD simulations at exascale through (1) the creation of a data analysis method based on a suite of advanced collective variables (CVs) selected for annotation of structural molecular properties and capturing rare conformational events at runtime, (2) the definition of an *in situ* framework to automatically identify the frames where the rare events occur during an MD simulation and (3) the integration of both method and framework into two MD workflows for the study of early termination or termination and restart of a benchmark molecular system for protein folding —the Fs peptide system (Ace-A_5(AAARA)_3A-NME)— using Summit. The approach achieves faster exploration of the conformational space compared to extensive ensemble simulations. Specifically, our *in situ* framework with early termination alone achieves 99.6% coverage of the reference conformational space for the Fs peptide with just 60% of the MD steps otherwise used for a traditional execution of the MD simulation. Annotation-based restart allows us to cover 94.6% of the conformational space, just running 50% of the overall MD steps.

PASC '23, June 26–28, 2023, Davos, Switzerland

ACM ISBN 979-8-4007-0190-0/23/06.
https://doi.org/10.1145/3592979.3593420

CCS CONCEPTS

• **Software and its engineering** → **Frameworks**; • **Information systems** → **Online analytical processing**; • **Computing methodologies** → **Molecular simulation**; **Massively parallel and high-performance simulations**; • **Applied computing** → **Molecular structural biology**.

KEYWORDS

Molecular dynamics, scientific workflows, simulation ensembles, *in situ analysis*, collective variables.

ACM Reference Format:
Silvina Caino-Lores, Michel A. Cuendet, Jack Marquez, Ekaterina Kots, Trilce Estrada, Ewa Deelman, Harel Weinstein, and Michela Taufer. 2023. Runtime Steering of Molecular Dynamics Simulations Through In Situ Analysis and Annotation of Collective Variables . In *Platform for Advanced Scientific Computing Conference (PASC '23), June 26–28, 2023, Davos, Switzerland*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3592979.3593420

1 INTRODUCTION

MD simulations are widely utilized in chemistry, material sciences, molecular and structural biology, and in drug design. The system sizes and time scales accessible to MD simulations have been steadily increasing [27], making MD simulations the most common runs on petascale machines. For example, a survey of resources used on XSEDE machines over the past six months [2] shows that biomolecular codes (predominantly MD codes such as Amber [10], CHARMM [8], and NAMD [36]) use 25.7% of the ACCESS (formerly XSEDE) resources (i.e., the total amount of XD service units (SUs) used by jobs in the fields of science indicated). At the same time, the work of Luu and co-authors [33] has shown that HPC computing resources can already be up to 75% idle performing

I/O operations, due to poor data handling while running scientific simulations. The transition from petascale to exascale computing brings unprecedented computing capability to MD simulations, enabling high-frequency sampling of fast events. Next-generation high-performance computing (HPC) systems (e.g., ACCESS high-end clusters such as Stampede2 and national laboratory supercomputers such as Aurora and Frontier) will have dramatically larger compute performance than current systems such as the Oak Ridge National Laboratory supercomputer Summit. The increase in computing capability (e.g., Frontier compute performance is 8x that of Summit [49]) translates directly to an ability to execute longer simulations. For MD simulations, this means generating more data in terms of the number and length of MD trajectories. However, the I/O bandwidth and parallel file system capacity of next-generation HPC systems will not grow at the same pace. The steady increase (in petaflops) and the stagnant I/O bandwidth (in TB/s) in next-generation machines such as Frontier and Aurora, compared with current and past HPC systems such as Summit, is well known. For example, Frontier's parallel file system has only 2x the performance and capacity of Summit's I/O subsystem, which currently has 2.5 TB/s peak I/O bandwidth. While many architectural implementations and scheduling aspects of next-generation HPC systems remain a topic of discussion, such as the role of burst buffers (BBs) and the feasibility of smart I/O staging, the research community must revisit the way MD simulations are executed. Hardware and scheduling strategies such as BBs and staging are not the magic bullet: I/O contention will still be a problem if the burst buffer capability is exceeded [24, 32, 43]. Moreover, BBs can improve offloading bandwidth but do not help upload data from storage. The overall coordination of data generation and analysis will not be able to rely on manual, centralized approaches as it does now [4]. While MD I/O is often manageable at runtime with high stride sampling rates, many applications may benefit from high-frequency sampling (e.g. to study the mechanism of fast and rare conformational changes). In addition, I/O is already an issue for a posteriori analysis. Thus, new frameworks are needed for MD simulations in which HPC meets data analytics.

Our work transforms the centralized nature of the MD analysis into a distributed approach that is predominantly performed *in situ*. It supports a broad range of MD codes and can enable on-the-fly tuning of MD workflows (i.e., early termination and restart). Contrary to traditional MD data analytics that uses centralized data analysis (i.e., first generates and saves all the trajectory data to storage and then relies on the post-simulation analysis), we use collective variables (CVs) to analyze data as they are generated and annotate MD outputs to manage increasingly complex MD workflows. Note that we focus on the analysis of MD-generated data (e.g., capturing significant rare events and monitoring convergence of observables based on inherently noisy and high-dimensional MD outputs) rather than on the generation process (e.g., modeling of the atom interactions and parallelization of the single MD jobs), since other efforts tackle the computing challenges of MD simulation code at exascale. By leveraging the standard formats of MD-generated outputs, users can plug in their own *in situ* analyses and apply them to the most used MD codes through our framework. The workflows built with our framework do not require the recompilation of any MD code or redesigning of any MD script. Instead, they capture outputs in memory at runtime as they are generated. Here, we demonstrate the capabilities of our framework in a case of enhanced adaptive sampling for the exploration of the conformational space of the Fs (folded short) peptide, which is often used to benchmark protein folding experiments and simulations. We study the dynamics and MD step throughput of an ensemble of trajectories analyzed with CVs that can be computed *in situ* using our framework running on Summit. Using annotation-based early termination, we obtain 99.6% coverage of the reference conformational space with just 60.3% of the MD steps otherwise used for a traditional execution of the MD simulation (i.e., without any early termination). Annotation-based restart allows us to cover 94.6% of the conformational space, just running 50% of the overall MD steps. The contributions of this paper are threefold:

- We formulate and implement two *in situ* methods to trace conformational changes in MD simulations at runtime by locally reducing knowledge on high-dimensional molecular organization into a set of relevant CVs.
- We design a modular MD framework to accurately and efficiently collect CVs capable of exposing rare events at runtime while minimizing data movement and communication.
- We implement *in situ* workflows that integrate simulation and analytics to study the benefits of annotation-based early termination or termination and restart for the Fs peptide system (Ace-A_5(AAARA)_3A-NME).

2 COLLECTIVE VARIABLES FOR TRAJECTORY ANNOTATIONS

MD simulations complement wet lab experiments by providing molecular and atomistic resolution information that is either not directly accessible by experiment or challenging to obtain. Specifically, classical MD simulations computationally replicate the behavior of a physical molecular system by iterating a two-step algorithm. First, the interactions between atoms are calculated using a force field model consisting of a mathematical function of atomic positions and a set of pre-calibrated parameters yielding sets of calculated forces between the moving atomistic components of the molecule. Second, based on the calculated forces, the positions of the atoms are advanced by solving Newton's equations on a small time step. Calculating long-range forces in systems of several hundreds of thousands of atoms is by far the most compute-intensive part of the calculation. An MD job reproduces the time-dependent evolution of the structure of any molecular system by computing and writing to store the system's atomic coordinates as they are changed by the Newton equations using the computed forces. Referred to as the molecular *frame*, the molecule's 3D conformation (i.e., structure) determines other relevant properties calculated at regular intervals as the MD evolves in time.The sequence of molecular conformations defined by each component atom's complete set of 3D spatial coordinates is known as the *trajectory*. A trajectory is written to disk at fixed intervals.

The information sought from the trajectories can relate to conformational changes in the molecule's structure, which represent energetically favorable repositionings of collections of atoms from one metastable region in the conformational space of a molecule to another. Because such changes require concerted motion of many

atoms over energy barriers, they are *rare events* in the MD simulation trajectory. Such a rare event can represent a phase transition, the folding of a protein, or protein conformational changes related to the protein's function. In biology, such functions include solute transport across membranes, ligand-triggered signaling, or enzymatic catalysis. When the function-related conformational rearrangements are expected or known, their occurrence in the trajectory can be detected by monitoring a small set of collective variables (CVs) that capture the relevant molecular motions [21]. In practice, a collection of CVs is monitored to calculate ensemble averages over molecular configurations. The traditional approach to MD uses CVs to monitor the evolution of the molecular system a posteriori. The approach presented here allows for active monitoring of CVs to capture the evolution of the molecular systems at runtime and managing ensembles of trajectories to improve the sampling of the conformational space.

A large-scale MD simulation is an ensemble of MD jobs (as many as hundreds of thousands that run on different compute nodes and produce independent trajectories). Each job simulates the same molecular system starting from different initial conditions (e.g., positions, velocities) or under different conditions (e.g., temperature, protein mutants, in complex with various ligands such as drugs). The ensemble-based nature of MD simulations promises computational scalability at exascale for relevant MD applications such as protein structure prediction, protein folding, protein-protein interactions, and protein-ligand interactions. In this work, we focus on the data locally outputted on each node to monitor and control the MD at runtime globally. To this end, we leverage two essential quantities that we compute *in situ* as an MD simulation evolves: the largest eigenvalue of alpha-Carbon (C_α) distance matrices (LEV) capturing molecular states and the effective sample size (ESS) identifying fast transitions of molecular states.

LEVs are based on the eigendecomposition of distance matrices and quantify one or more relevant structures in the molecular system where rare events may occur. Given a frame, we first simplify a targeted secondary structure composed of m amino acids by extracting the positions of its m C_α backbone atoms (x_i, y_i, z_i). To capture the dynamic relationship within the m amino acids, we build the square Euclidean distance matrix D from the positions of the corresponding C_α atoms for each frame. D is a normal matrix (i.e., the matrix is symmetric, the diagonal of D is identically zero, and the off-diagonal elements of D are strictly positive) and have stable eigenvalues. Calculating the proxy distance between two frames is as simple as computing the distance between corresponding eigenvalues [46]. Furthermore, D is constructed from points in 3 dimensions and thus has at most five nonzero eigenvalues. Johnston et al. [26] demonstrated how the five nonzero eigenvalues of D have the property that three are very small (close to zero) and two are high in value and opposite. Thus, we can capture most of the total domain change in eigenvalues in molecular simulations by observing the difference in the largest positive eigenvalue (LEV). Johnston et al. also demonstrated that the distribution of the largest eigenvalue for a given segment is a function of the number of amino acids in the structure and the LEVs of folded secondary structures [26]. We leverage these ranges to assess the status of secondary structures in a trajectory. LEV is computed independently on single frames, which makes it a good candidate for *in situ* annotation. We use LEV to monitor the folding state of secondary structures (e.g., helix or strand), so a time series of LEVs can be used to flag folded or misfolded states for trajectory termination.

ESS captures structural changes over a window of n observations in the trajectory, with n much smaller than the total number of MD steps in a trajectory. If observations are uncorrelated within the window, no significant changes in the sequence of frames occur; otherwise, a rare event may have occurred if autocorrelation is detected. Assume we have an uncorrelated time series $\{x_i\}$, $i = 1, ..., n$. Then the expected variance of the mean is $\sigma^2(\bar{x}) = \frac{\sigma^2}{n}$, where σ^2 is the variance of the $\{x_i\}$. On the other hand, if there is a correlation between consecutive points, the time series can be considered a stationary process with autocorrelation coefficients $\{\rho_k\}$, where k is the lag time between two observations. In this case, $\sigma^2(\bar{x})$ is larger than without autocorrelation because the time series contains less independent information to estimate $\bar{x}$. The variance can be expressed as $\sigma^2(\bar{x}) = \frac{\sigma^2}{n_{\text{eff}}}$, where n_{eff} is called the effective sample size (ESS), which intuitively represents the equivalent number of independent points in the sample, with $1 \leq n_{\text{eff}} \leq n$. A strongly autocorrelated time series will have a low value of n_{eff}. To estimate the value of n_{eff} from the data, we first use the standard (unbiased) estimator r_k for the correlation coefficients ρ_k given by

$$r_k = \frac{\sum_{i=1}^{n-k} (x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}. \tag{1}$$

Then the ESS can be expressed as

$$n_{\text{eff}} \approx \frac{n}{1 + 2\sum_{k=1}^{k_{\max}} \frac{n-k}{n} r_k}. \tag{2}$$

Because the estimation of r_k at large lag times is marred with significant uncertainties, it was shown that the summation could be truncated after term $k_{\max}$, with $k_{\max} = \min\left\{k \mid r_k > 0,\ r_{k+1} < 0\right\}$, i.e., the last value before the autocorrelation function changes sign for the first time [13, 50, 12]. Thus, n_{eff} captures in a single number the amount of correlation in the time series and quantifies the amount of independent information contained in it, hence the name ESS. By calculating n_{eff} on an entire trajectory, Chodera et al. [12] devised a way to measure how much of the initial equilibration time could be discarded. Here, instead of considering the full-time series, we propose to calculate $n_{\text{eff}(t)}$ in a sliding window of n frames and duration τ. We can then expect the following behaviors: i) if the system is transitioning between different macrostates on a time scale comparable to τ, we will observe strong autocorrelation and low $n_{\text{eff}}(t)$; ii) if the system fluctuates around a single macrostate during τ, we will see rapid local fluctuations with little autocorrelation and high $n_{\text{eff}}(t)$. Hence monitoring strong peaks in $n_{\text{eff}}(t)$ can serve as a detector of immobile trajectory behavior. In our case, we use ESS to analyze the sequence of a trajectory's LEVs to get a notion of the mobility trend of the protein.

3 DESIGN OF OUR *IN SITU* FRAMEWORK

Contrary to traditional MD data analytics that uses centralized data analysis (i.e., first generates and saves all the trajectory data to storage and then conducts post-simulation analysis), we propose an approach that bypasses storage and enables on-the-fly tuning of MD workflows (i.e., early termination and restart of MD jobs). We introduce the concept of *in situ conjunction*, which is the

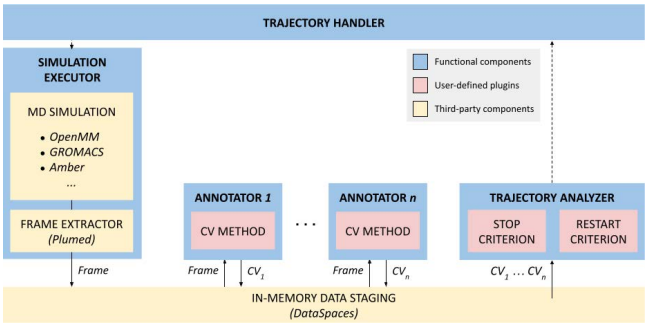


Figure 1: Architecture of our *in situ* framework with its functional components that enable the *in situ* conjunction.

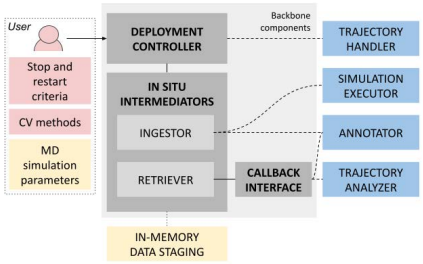


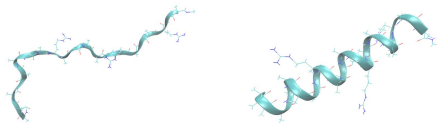
Figure 2: View of our *in situ* framework's backbone and its relationship with the end user and other elements in Fig. 1.

association of functional components, user-defined plugins, and third-party components that work together to analyze data as they are generated and annotate MD frames *in situ*. Figure 1 shows the architecture that supports this functionality in our *in situ* framework. The *functional components* of the *in situ* conjunction are the entities involved in the control of MD simulation workflows. They can be adapted based on the type of scientific discovery targeted by an MD simulation (e.g., protein structure prediction, protein folding, and protein-protein or protein-ligand interactions). Our framework has four functional components:

The *simulation executor* connects two key third-party components: the MD simulation and the frame extractor. Our *in situ* framework is agnostic to the MD code and does not require the recompilation of the used MD code or redesigning its script. Instead, it captures MD frames in memory at runtime as they are generated through Plumed v2.6.0. This plugin can extract frames from running simulations and is available for all the major MD codes [9]. The frames contain raw atom coordinates collected in a standard format of MD-generated outputs. We leverage the format to conduct *in situ* data analytics agnostic to the MD simulation code. The simulator executor wraps the simulation and frame extraction functionality and transfers the extracted frames to the in-memory data staging area, represented by Dataspaces v1.8.1 [17]. The *annotators* are the components that compute the CVs from the frames produced by the simulation executor. We define annotations as a list of CV values per frame constituting the metadata for a time step. The frequency of pulling the data and the number and nature of CVs to be computed are set by the user. A single annotator can be shared for all the CVs, or individual annotators can be created for each CV separately. CV calculation methods are pluggable components in the form of Python scripts provided by the end user. The annotations associated with a frame are transferred to the in-memory staging area for further analysis. Persistence and management of the annotation buffers are delegated to Dataspaces. The *trajectory analyzer* is the component that resorts to user-defined termination and restarts criteria to make decisions about the ongoing simulation based exclusively on the analysis of the CV time series. To this end, the trajectory analyzer obtains the annotations from the in-memory staging area for one or multiple time steps. The number of steps taken into consideration for the decision is referred to as the *window* for the trajectory analysis. This window is configured in the pluggable termination or restarts analysis code provided by the end user as a Python script. Finally, the *trajectory handler* monitors the trajectory analyzer to execute termination and restart decisions on the simulation executor. Currently, the trajectory handler can restart a trajectory from its initial state. We are working on extending this functionality to support workflows in which the restart point is decided dynamically.

The design of these functional components and interconnections is built around our *in situ* framework's *backbone components* which fulfill deployment, data transfer, and callback functions. Figure 2 shows this internal architecture and how the functional components are associated with each backbone component. There are three backbone components: The *deployment controller* is the central orchestrator of the *in situ* conjunction. It spawns and sets up the communication channels between all the components of the *in situ* framework, including the functional components and any additional third-party elements. The deployment controller is the single entry point for the batch job representing the *in situ* conjunction. Users interact with the workflow through the deployment controller by indicating in a Bash script how the simulation is going to run and where these user-defined plugins are located. One or more *in situ intermediators* connect the in-memory staging area with the functional components that produce and consume data (frames). There are two types of *in situ* intermediators: ingestors and retrievers. Ingestors transfer data into the in-memory staging area and are used in the simulation executor and the annotators to store frames and CV values, respectively. Retrievers transfer data from the in-memory staging area and are used in the annotators and trajectory analyzer to obtain frames and CV values, respectively. Note that ingestors and retrievers can be combined to build bidirectional components like the annotators. Our *in situ* framework can be integrated into any workflow management system. Thus, scheduling and resource allocation from the perspective of the computing system are out of the scope of the framework's configuration. We do not enforce the co-location of any components in the same node, since co-locating coupled *in situ* components in the same node does not always result in the best performance [15]. We do allow the user to define a minimum number of cores or GPUs to dedicate to the simulation, the analysis and the data staging server.

Our *in situ* architecture allows scientists to target diverse problems by composing workflows with user-defined CV methods and tailored trajectory analysis techniques. In the following section, we present two MD workflows that leverage the runtime monitoring of the LEV CV and the associated ESS through our *in situ* workflow for effective trajectory analyses of a folding Fs peptide.



(a) (b)

Figure 3: Example of unfolded conformation of the Fs peptide (a) and its folded reference helical conformation (b).

4 GAINS IN *IN SITU* MD WORKFLOWS

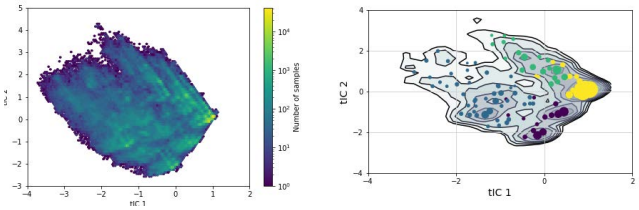
We assess the capabilities of our *in situ* framework with the Fs peptide (Ace-A_5(AAARA)_3A-NME) model system for protein folding to quantify gains with trajectories' early termination (MD workflow I) and termination and restart (MD workflow II).

4.1 Experimental Setup

We analyze the folding of the Fs peptide system into an α-helix. The folding process takes approximately 200ns. Figure 3 shows an example of the protein's unfolded state and its fully folded helical conformation. Studying the dynamics and folding mechanism of systems such as the Fs peptide is key to characterizing the dynamics of larger molecular systems in their early folding events [22].

We generated 40 all-atom molecular dynamics trajectories using GROMACS 2021.4 [1, 31] on the Oak Ridge National Laboratory's Summit supercomputer [45]. Due to the small number of atoms in the Fs peptide, each simulation ran on a single node with one GPU and eight CPU cores allocated. These trajectories are available in Dataverse. The simulations were set up using the Amber03 force field [19] and taking as initial conformation one of 10 random unfolded conformations constructed with the Modeller software [42]. Figure 3a shows a sample from these unfolded conformations. After a standard equilibration phase, we ran production trajectories (i.e., 400ns at 300K and constant pressure with a 0.002ps time step and constraints on bonds involving hydrogen atoms). Since we target scenarios with high-frequency sampling, we recorded the frames at every time step to conduct the comprehensive post hoc analysis presented here. In our *in situ* setup, we are using stride 10 to calculate CVs, which still allows us to study the MD with very high frequency. We measured $261.10 \pm 4.88 ns/day$ simulation performance without Plumed, $250.13 \pm 6.25 ns/day$ with Plumed extracting frames with stride 1, and $248.61 \pm 3.61 ns/day$ with stride ten. We monitored the conformation of the Fs peptide by recording all backbone dihedral angles, the LEV CV, and the associated ESS in windows of 500 frames. The computation of CVs is an asynchronous process in *in situ* analyses with no perturbation of the MD simulations [48]. Our *in situ* framework calculated the selected CVs with stride 10 and resulted in 20K CV values per trajectory. These annotations are available in Dataverse as part of the data artifact associated with this paper.

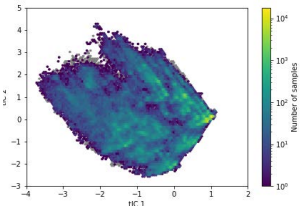
We organized the trajectories in four independent simulation subsets FS^1, FS^2, FS^3, and FS^4, each containing 10 trajectories starting from the conformations mentioned above. Subsets FS^3 and FS^4 were checkpointed at the GROMACS' default setting of 15-minute intervals to enable trajectory restarts for the second MD workflow (Sec. 4.3). We denote the trajectories in the entire Fs dataset as $FS = \bigcup_{i \in [1,4]} FS^i$.



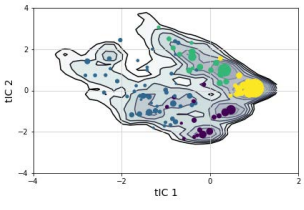
(a) tICA projections. (b) MSM representation.

Figure 4: Time-structure independent component analysis (tICA) projection showing the number of samples in each region of the free energy map (a) and the Markov State Model (MSM) representing the dynamics of the Fs peptide system in the entire *FS* trajectory set (b). MSM microstates are represented on the free energy surface. Colors indicate microstates groupings (i.e., macrostates); circles are proportional to the number of samples in the macrostate.

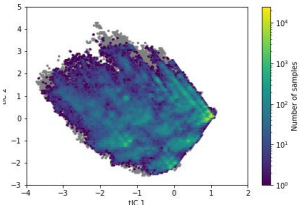
The free energy landscape of a biomolecular system such as the Fs peptide is characterized by metastable states separated by free energy barriers [7]. To describe this conformational space, we first create its 2D representation by applying a time-lagged Independent Component Analysis (tICA) decomposition technique [34] to the *FS* set of MD trajectories. Next, on the 2D tICA space, we build a Markov State Model (MSM) to describe the dynamics and kinetics of the Fs peptide folding in Fig. 4. Evaluation of the tICA projection in Fig. 4a involved (1) extracting backbone dihedral angles from the Cartesian coordinates of the system; (2) performing tICA on this dataset with a time lag of 5,000, and (3) projecting all the MD trajectories of the *FS* set onto the first two tICA components which reflect the two slowest degrees of freedom. In the resulting tICA space, tIC 1 axis captures the Fs peptide's folding stage, with lower values corresponding to more folded conformations. The most populated state on the tICA map is a completely folded alpha helix ($tIC1 \approx 1$), surrounded by two partially folded intermediate states ($0 < tIC1 < 1$). As expected, due to its high conformational flexibility, the unfolded state ($tIC1 < 0$) covers the largest space area on the tICA map. The 2D projection of the FS trajectory set onto the tICA space is further discretized into 100 microstates with the K-Means algorithm to build an MSM of the Fs peptide folding. An MSM is built by estimating a transition probability matrix (TPM), the elements of which represent all the pairwise probabilities of transitions between the 100 microstates. To identify metastable states of the process (i.e., the unfolded and folded states along with the intermediate conformations of the FS peptide folding), we applied Perron Cluster Cluster Analysis (PCCA++) [14] to the evaluated TPM. PCCA++ assembles the microstates into kinetic macrostates based on the estimated transition rates connecting them. Fig. 4b shows a kinetic map of the conformational space with four metastable states explored by the *FS* trajectory set. The metastable states exhibiting slow inter-transition rates are therefore separated by high free energy barriers. We follow this protocol in the following sections to build analogous tICA projections and MSMs for different ensemble operations (i.e., early termination and restart) and analyze how they affect the exploration of the conformational space.



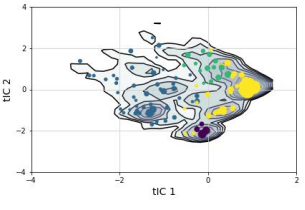
(a) tICA projections for $FS^1 \cup FS^2$ (no checkpoints).



(b) MSM for $FS^1 \cup FS^2$ (no checkpoints).



(c) tICA projections for $FS^3 \cup FS^4$ (with checkpoints).



(d) MSM for $FS^3 \cup FS^4$ (with checkpoints).

Figure 5: tICA projection and MSM representing the dynamics of the Fs peptide system trajectory subsets without checkpoints ($FS^1 \cup FS^2$) (a and b) and with checkpoints ($FS^3 \cup FS^4$) (c and d). The grey area in the tICA projections represents the projection of the entire FS dataset.

For our MD workflows, we divide the entire trajectory set into two subsets: a first containing the trajectories that do not have intermediate checkpoints ($FS^1 \cup FS^2$), and a second containing the trajectories with intermediate checkpoints saved ($FS^3 \cup FS^4$). In Fig. 5 we projected these subsets onto the tICA map generated in Fig. 4. We see that by choosing subsets of trajectories, we obtain slight differences in the coverage of the conformational space (96.50% vs. 98.68%) and the microstate mapping (Figs. 5b and 5d). Although these small variations can lead to a microstate clustered with different macrostates (compare yellow and purple dots), we observe a similar microstate distribution in both subsets.

4.2 MD Workflow I: Trajectory Termination

Our first MD workflow explores the application of our framework to conduct *in situ* analysis of the LEV CV. We use the LEV time series the annotator generates to determine if the trajectory has reached the folded state. When the folded conformation is detected in the trajectory analyzer (i.e., the LEV is within a range associated with the folded state for this 21-residue molecule [26]), we set up an early termination criterion so that the trajectory handler terminates the execution of the specific MD job generating that trajectory, saving computational resources. We analyze the effect of the early termination on exploring the conformational space.

4.2.1 Validation of LEV CVs. We use the LEV CV to determine the folded state of the Fs peptide [26]. Because this protein has 21 residues (21 C_α atoms), the expected LEV range for the folded state is $[4000, 4400]$ according to [26]. For the sake of reproducible and trustworthy results, we validate that this CV accurately captures the folded state in the Fs system by studying the RMSD of the trajectories vs. the reference helical conformation in Fig. 3b, since very low RMSD values under 0.5Å correlate exclusively with the

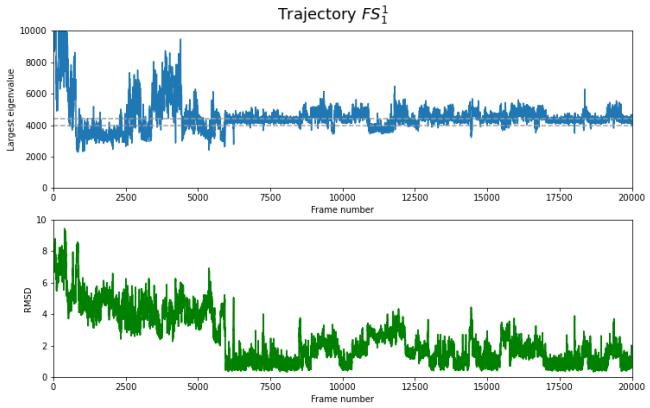
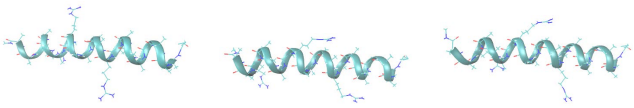


Figure 6: Time series of LEV CV (top) and RMSD of trajectory starting from unfolded conformation vs. helical conformation depicted in Fig. 3b (bottom).



(a) Frame 5957 $RMSD = 0.426$Å $LEV = 4291.041$

(b) Frame 10021 $RMSD = 0.462$Å $LEV = 4218.265$

(c) Frame 17571 $RMSD = 0.434$Å $LEV = 4270.113$

Figure 7: Fs peptide state in frames 5957 (a), 10021 (b), and 17571 (c) of trajectory FS_1^1. These frames have RMSD $RMSD \leq 0.5$Å and $4,000 \leq LEV \leq 4,400$, which indicates the folded state as visualized in Fig. 6.

folded state of the peptide [7]. Figure 6 compares the LEV values per frame (top) and the RMSD (bottom) of a folding trajectory vs. the reference helical conformation. A sample of three frames meeting the $RMSD \leq 0.5$Å criterion in our MD simulation is shown in Fig. 7. Overall, we empirically observe across all our MD simulations in the entire FS dataset that LEV can capture the folded state of the Fs peptide when it is in the expected range of $[4000, 4400]$. Thus, we use the LEV CV to detect trajectories that reach the folded state. By terminating these folded trajectories, we promote the exploration of the entire conformational space. We analyze the gain of this early termination in the following section.

4.2.2 Gain with In Situ LEV Analysis. Figure 8 shows the frame in which termination is detected *in situ* vs. the first occurrence of the folding event for the entire FS dataset. We observe how ten trajectories never reach the folded state. This is detected by the *in situ* analysis, allowing them to run for the entire 200M steps. We also note how FS^3 and FS^4 include trajectories that are more likely to fold, and when they do, they fold faster. Since there are no differences between the setup of the simulations beyond the addition of checkpoint in FS^3 and FS^4, this may be due to the chaotic nature of MD simulations, which make simulations starting from the same conformation and velocities traverse the conformational space differently [47]. For the trajectories that reach the folded state, the *in situ* analysis allows terminating the trajectory before the fixed, user-defined number of steps is reached. As a result, 70% of the 40 trajectories can be terminated early saving between 90% and 5% of the simulation's steps for each trajectory. As a result, 39.72% of

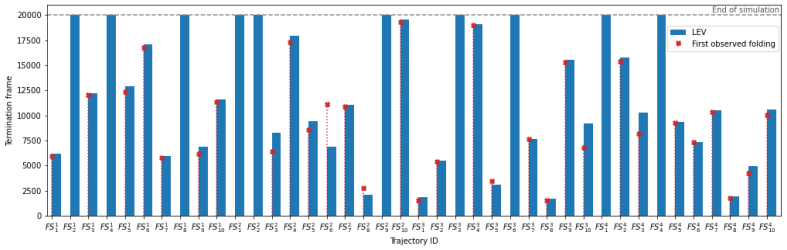


Figure 8: Summary of Fs peptide folded state detection using LEV. The bars indicate the frame in which LEV detects the folded state, while the dots indicate the first actual occurrence of a folding event according to $RMSD \leq 0.5$Å criterion.

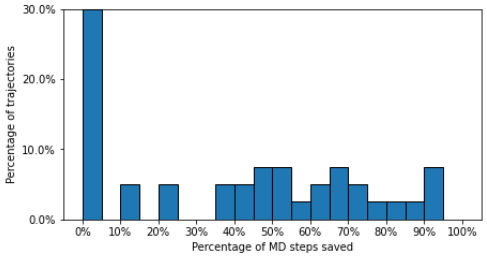
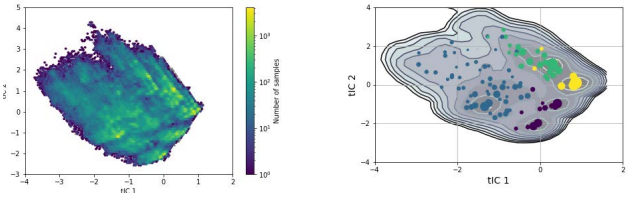


Figure 9: Distribution of trajectories saving MD steps by early termination as suggested by the *in situ* calculated LEV.

Table 1: Gain using LEV-based *in situ* termination for the four trajectory subsets.

Trajectory set	Trajectories stopped	Steps saved	Simulated time saved	Trajectory equivalent
FS^1	70%	33.63%	1,345.42ns	3.36
FS^2	70%	32.37%	1,294.98ns	3.24
FS^3	80%	44.66%	1,786.30ns	4.47
FS^4	80%	48.24%	1929.60ns	4.82



(a) tICA projection, terminated. **(b) MSM, terminated.**

Figure 10: tICA projection (a) and MSM (b) for the complete *FS* trajectory set resulting from terminating the simulations at the LEV criterion. For comparison, refer to Figs. 4a and 4b, respectively.

the total MD steps are saved by *in situ* early termination with LEV in the entire ensemble, representing 6,356 ns simulated time and equivalent to almost 16 400-ns simulations with the configuration of this ensemble. Figure 9 shows the distribution of the percentage of MD steps saved in each trajectory for the entire dataset. Detailed gain for each independent simulation set is presented in Tab. 1.

Figure 10 shows the MSM and tICA projections of the entire *FS* trajectory set when simulations are terminated following the LEV *in situ* termination criterion. We obtain a 99.6% coverage of the tICA space just running 60.3% of the total MD steps for this ensemble, and all four expected macrostates are sampled. Compared with Fig. 4b, which shows the entire MSM for the combined dataset without early termination, we note that there is a more uniform sampling of the macrostates when we build the MSM with the shorter trajectories. Specifically, the sampling of the rightmost macrostate (in yellow) decreases in favor of the other macrostates. This is expected since we are actively reducing the exploration around the folded state by terminating the trajectories that reach this point, to favor the sampling of other states.

We conclude that the most determinant factor for the exploration of the conformational space is the number of trajectories simulated rather than their length because each new trajectory adds to the diversity of the ensemble. Since we can terminate simulations that converge to a specific state such as the folded conformation, we can save computational resources that we can dedicate to run additional simulations widening the diversity of the space explored. This observation is consistent with consensus in the literature [23].

4.3 MD Workflow II: Trajectory Restart

The results in the previous section indicate that terminating trajectories with stable conformations and using the saved resources for new trajectories can improve the sampling of the conformational space. We empirically validate this hypothesis for our Fs system by studying how our framework can assist trajectory termination and restart, and in doing so, can incrementally integrate annotation-based decision-making in the execution of the ensemble. To this end, we add an additional annotator in our framework to capture the ESS of the LEV time series. We analyze the variation in ESS to decide if the trajectory has been in an immobile state for a certain period of time. If this occurs, the trajectory handler is instructed to terminate the trajectory and restart a new trajectory from either its initial conformation or a checkpointed conformation closer to the time step of the terminated trajectory.

4.3.1 Validation of the ESS criterion. We compute the ESS in a window containing the last 500 values of the LEV CV to capture the dynamicity of the spatial changes in the protein. We empirically validate this ESS capability from observations of our MD simulations. Figure 11 shows an example of ESS use. In the top figure, Trajectory FS_4^3 is consistently fluctuating around the 2,000-LEV value for approximately 50ns (between Frames 0 and 15,000). Between Frames 15,000 and 17,500, we can detect the sudden uncorrelated oscillations around a stable conformation by noting the acute increase in the ESS (bottom figure). By making the decision to terminate this trajectory at this point, we can dedicate resources to simulate a different trajectory. Concerning trajectory FS_4^3 in Figure 11, we can

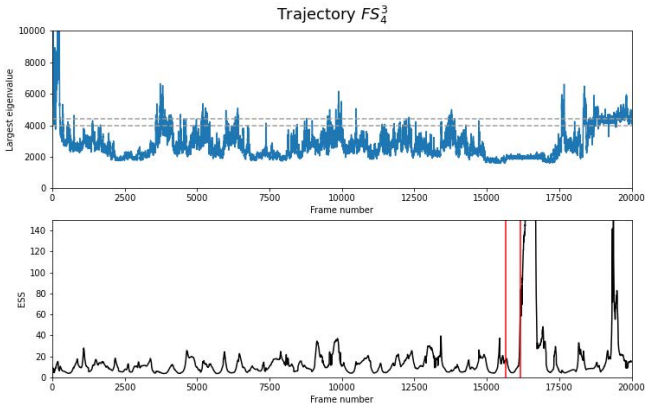
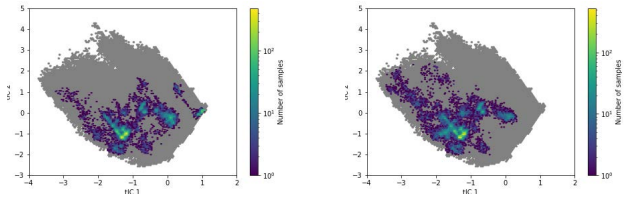


Figure 11: Time series of LEV CV (top) and ESS CV (bottom). The red bars indicate the frame window in which termination conditions were met through ESS analysis.



(a) FS_4^3, entire. **(b) FS_4^3 restarted with FS_4^4**

Figure 12: tICA projections of the entire FS_4^3 trajectory (a) and trajectory FS_4^3 stopped according to ESS criterion and restarted with FS_4^4 from the initial conformation.

terminate it and restart trajectory FS_4^4 from its initial confirmation. Figure 12 shows the tICA projection of FS_4^3 vs. the tICA projection of FS_4^3 terminated at 323.14ns of simulated time and restarted from the origin as FS_4^4 (we terminate FS_4^4 when a total of 400ns have been simulated for a fair comparison). If we let FS_4^3 run to completion, we get a 35.02% coverage of the tICA space, while we increase this to 40.75% by applying the restart process. Note that by performing the restart the trajectory moves away from the rightmost area of the tICA map, which is heavily sampled in the overall ensemble. This is a desirable behavior that increases the diversity of the sampling and shows that ESS analysis is suitable to define a termination criterion to improve the exploration of the conformational space.

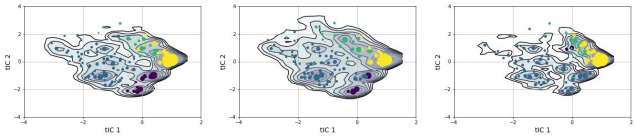
4.3.2 Gain with In Situ ESS Analysis. In this workflow, we are only using subsets FS^3 and FS^4 to test the gain associated to an *in situ* analysis using ESS because we want to be able to compare different restart mechanisms, one of them including a checkpoint component that is only present in these two subsets. We use termination and restart over the subsets FS^3 and FS^4: we terminate a trajectory from FS^3 based on its ESS CV and restart another trajectory from FS^4 on the same node, starting from either its initial conformation or the checkpointed conformation closer to the FS^3 termination time step. For instance, if Trajectory FS_1^3 is terminated at time step t_{ess} based on its ESS, then Trajectory FS_1^4 can be restarted from its original starting conformation at time step t_0 or at the checkpointed configuration at t_c, with t_c equal or below t_{ess}. Table 2 summarizes the full set of experiments and the criteria we use for trajectory termination and restart. Experiment *B* serves as the comparison baseline since it represents the complete executions of subsets FS^3 and FS^4, thus representing the maximum number of steps that can be executed. Experiment *NN* determines the potential gain of terminating the first trajectory from FS^3 at a random point in time, t_{rand}, and restarting the second trajectory from FS^4 from its origin conformation at t_0. In contrast, experiment *AN* builds on the ESS of LEV *in situ* annotations as the termination criterion: it considers the trajectory's mobility to decide to terminate the first trajectory from FS^3 at t_{ESS} and restart the second trajectory from FS^4 at t_0. Finally, experiment *AA* leverages the ESS criterion to decide both termination and restart so that the second trajectory starts from the last simulation checkpoint performed before t_{ESS}, at step time t_c. We compare the MD steps executed in each experiment, tICA coverage, and MSM clustering against the baseline experiment *B* to understand the gain of *in situ* annotations for trajectory analysis. We refer back to the entire *FS* ensemble tICA map and MSM to validate the appropriate exploration of the conformational space.

Due to space constraints, we present results for a limited number of tests. We ran five trials of experiment *NN* to avoid bias in the random selection of the trajectory termination frame. We also ran each trial twice to account for the influence of starting from a trajectory in FS^3 or FS^4 while preserving the same random sequence of termination frames (i.e., trajectories FS_i^3 and FS_i^4 end at the same t_{rand}). To facilitate comparisons, the number of MD steps executed per termination-restart pair is fixed to the entire length of a single trajectory (i.e., 200M MD steps). Restarting a FS_i^3 trajectory with a FS_i^4 trajectory yields 95.09% ± 1.51% tICA coverage while restarting a FS_i^4 trajectory with a FS_i^3 trajectory yields 95.68% ± 0.98% tICA coverage (both compared against the entire tICA space). These observations hold for the omitted figures. We do highlight a sample of three trials in Fig. 13 to show how the random termination affects the MSM clustering, which does not always accurately capture the macrostate distribution observed in Fig. 5d (note that this figure shows the baseline MSM for the $FS^3 \cup FS^4$ subset).

Figure 14 shows the MSM clustering for the four experiments. When comparing random (*NN*) vs. annotation-based (*AN*) termination with ESS with restart from the origin conformations, we note that the tICA coverage is very similar (94.80% and 94.55%, respectively). This indicates that random termination can effectively reduce the number of executed MD steps, but we must accept the variability we observe in Fig. 13, so this is only useful with multiple restarts. We also observe that the MSM clustering in the *AN* experiment includes an undersampled macrostate (in green in Fig. 14c. This occurs because the states not visited in this ensemble are consistently located in the same region of the tICA landscape. By restarting the trajectory in a region near the termination state indicated by ESS, we obtain a macrostate distribution (Fig. 14d) much closer to the overall MSM for the entire *FS* dataset (Fig. 4b). Work in progress investigates the transition probability matrix between the different microstates and macrostate clusterings in the MSMs in Fig. 14 to give further analytical support for these empirical observations and understand the cumulative effect of multiple annotation-based restarts in realistic scenarios.

Table 2: Summary of experiments covering trajectory restart.

Experiment	Termination	Restart	Objective	Representation
B	End	Origin	Establish baseline case	FS_i^3 t_0 t_{N-1} → FS_i^4 t_0 t_{N-1}
NN	Random	Origin	Determine gain with naive termination and naive restart	FS_i^3 t_0 t_{rand} → FS_i^4 t_0 $t_{N\text{-}rand\text{-}1}$
AN	ESS	Origin	Determine gain with annotation-based termination and naive restart	FS_i^3 t_0 t_{ESS} → FS_i^4 t_0 $t_{N\text{-}ESS\text{-}1}$
AA	ESS	Checkpoint	Determine gain with annotation-based termination and restart	FS_i^3 t_0 t_c t_{ESS} → FS_i^4 t_c t_{N-1}



(a) $FS_i^3 \rightarrow FS_i^4$**, trial 1. (b)** $FS_i^3 \rightarrow FS_i^4$**, trial 2. (c)** $FS_i^3 \rightarrow FS_i^4$**, trial 3.**

Figure 13: Impact of random termination in three randomized trials of $FS_i^3 \rightarrow FS_i^4$ **restart.**

5 ASPECTS OF NOVELTY

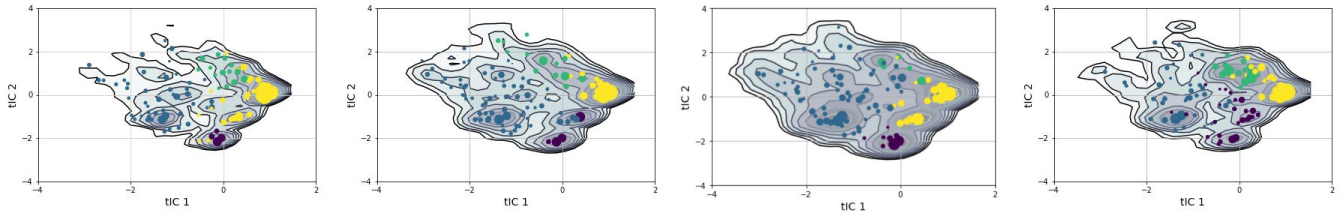
In this paper we introduce a novel framework for large scale ensemble molecular dynamics simulations relying on in situ simulation analysis and workflow management at runtime. A central feature of our workflow is its modular implementation designed to allow seamless addition of trajectory analyzers (CVs), annotators, or trajectory handlers. While this already demonstrated efficiency benefits, these might be improved further by integrating state-of-the-art resampling methods like FAST [51] or WESTPA [52].

A few methods commonly used in the field attempt to provide a plug-and-go mechanism for identifying the CVs that best describe MD simulations. These are not single observation methods but are derived from the full simulation post hoc [3, 35, 41, 11]. Furthermore, conformational analysis based on principal component analysis (PCA) fails to scale for the same reason, as does an analysis using RMSD, since the all-to-all atomic comparison leads to a critical bottleneck. Using traditional relational databases rather than CVs for trajectory analysis requires a posteriori trajectory upload and data analysis using database functions (e.g., PostgreSQL [28, 20]). Many clustering methods have been applied to molecular structures and MD trajectories to find similarities across datasets. For example, Shao et al. outline how there is not a one-size-fits-all clustering method [44]. Li and Dong describe the effect of clustering algorithms such as Bayesian, k-means, and kinetic clustering on establishing Markov state models for MD simulations [30]. Rodriguez presents a method for fast searches and identification of density peaks in trajectories [40]; the method requires the scientist to pick the number of peaks thought to be correct visually. Such clustering strategies are used with a post-simulation perspective but must be more scalable on large-scale machines.

Software tools are available for comparing metrics representing molecular structures distributedly, including the number and position of ion molecules that permeate a channel. One such method is dynamic tensor analysis [39]. Efforts were undertaken to make the computation as light as possible, but the method still requires that a sequence of distance matrices from the amino acids of the entire protein be stored in memory, resulting in a larger memory footprint than our technique. Centralized algorithms [6, 37] make metrics analyses inefficient when dealing with large proteins and long trajectories. Hybrid approaches have been introduced to handle big data analysis problems in the HPC context [5, 29, 16, 15]. These approaches combine in situ and in transit processing for extreme-scale scientific analysis such as topological analysis, descriptive statistics, and visualization. We note similar efforts to manage an ensemble of trajectories on large distributed infrastructures. Such a framework has been developed for NAMD using the parallel programming system Charm++ [25]. A similar platform has been proposed for GROMACS, based on the distributed high-performance computing platform Copernicus [38]. The high-throughput MD [18] framework developed around the program ACEMD is a Python interface that supervises MD data generation and a posteriori analysis. None of these frameworks is tightly integrated with in situ trajectory analysis and annotation as our framework. We also note that the works cited are specific to a single MD code. In contrast, our approach targets a universal interface completely independent of the underlying MD engine, provided they have Plumed support. In our approach, users do not have conduct format conversions: they receive atom coordinates regardless of the MD engine used. Our approach allows scientists to abstract and reuse their methods as workflow building blocks.

6 CONCLUSIONS

This paper presents an *in situ* framework for annotating structural molecular properties with collective variables (CVs) and managing ensemble workflows at runtime. Our framework is integrated into two MD workflows to study early termination and restart a benchmark molecular system for protein folding — the Fs peptide system (Ace-A_5(AAARA)_3A-NME)— using Summit. The systematic approach we propose relies on a CV (LEV) that does not require a priori knowledge of the conformational changes one seeks to simulate, together with a very general time series analysis method (ESS)



(a) *B*
100% steps, 98.68% coverage.

(b) *NN* (trial 2)
50% steps, 94.80% coverage.

(c) *AN*
50% steps, 95.74% coverage.

(d) *AA*
50% steps, 94.55% coverage.

Figure 14: Comparison of the resulting MSMs for the four experiments defined in Tab. 2 running with $FS_i^3 \rightarrow FS_i^4$ restart. Experiment *NN* is shown for trial 2, which is the closest to the average tICA coverage in all the *NN* trials. Percentages indicate the MD steps executed compared to the baseline experiment *B* and the tICA coverage compared to the baseline tICA map defined with the entire 40-trajectory *FS* dataset.

to detect immobility in trajectories. Evaluation results support our hypothesis that CVs can drive the exploration of the conformational space at runtime while using the available resources more efficiently. These results might be improved further by integrating state-of-the-art resampling methods. Work in progress extends this analysis to larger systems with multiple secondary structures and workflows with checkpoint-restart phases guided by CV heuristics. Our software (i.e., *in situ* framework, user-defined scripts, deployment configuration, and analysis notebooks) is open-source and available in GitHub (https://github.com/Analytics4MD/A4MD). Data artifacts (i.e., MD trajectories and annotations) are available in Harvard Dataverse (https://doi.org/10.7910/DVN/CLP0LY).

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (NSF) under grant numbers 1741057, 1841758, 2138811 and 2223704; the Oak Ridge Leadership Computing Facility under allocation CSC427; the Extreme Science and Engineering Discovery Environment (XSEDE) under allocation TG-CIS200053; and IBM through a Shared University Research Award.

REFERENCES

[1] Mark James Abraham et al. 2015. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1, 19–25.
[2] [n. d.] ACCESS: Advanced CI Coordination Ecosystem: Services Support. https://access-ci.org/. Online; accessed 23 December 2022. ().
[3] Andrea Amadei, Antonius BM Linssen, and Herman JC Berendsen. 1993. Essential dynamics of proteins. *Proteins: Structure, Function, and Bioinformatics*, 17, 4, 412–425.
[4] Vivek Balasubramanian, Travis Jensen, Matteo Turilli, Peter Kasson, Michael Shirts, and Shantenu Jha. 2020. Adaptive ensemble biomolecular applications at scale. *SN Computer Science*, 1, 2, 104.
[5] Janine C Bennett et al. 2012. Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.
[6] Christoph Best and H-C Hege. 2002. Visualizing and identifying conformational ensembles in molecular dynamics trajectories. *IEEE Computing in Science & Engineering (CiSE)*, 4, 3, 68–75.
[7] Gregory R Bowman, Xuhui Huang, and Vijay S Pande. 2009. Using generalized ensemble simulations and Markov state models to identify conformational states. *Methods*, 49, 2, 197–201.
[8] B. R. Brooks et al. 2009. CHARMM: The biomolecular simulation program. *J. of Computational Chemistry*, 30, 10, 1545–1614.
[9] Giovanni Bussi and Gareth A Tribello. 2019. Analyzing and biasing simulations with PLUMED. In *Biomolecular Simulations*. Springer, 529–578.
[10] David A. Case et al. 2005. The Amber biomolecular simulation programs. *J. of Computational Chemistry*, 26, 16, 1668–1688.
[11] Michele Ceriotti, Gareth A Tribello, and Michele Parrinello. 2011. Simplifying the representation of complex free-energy landscapes using sketch-map. *Proc. of the National Academy of Sciences*, 108, 32, 13023–13028.
[12] John D Chodera. 2016. A Simple Method for Automated Equilibration Detection in Molecular Simulations. *J. of Chemical Theory and Computation*, 12, 4, (Apr. 2016), 1799–1805.
[13] John D Chodera, William C Swope, Jed W Pitera, Chaok Seok, and Ken A Dill. 2007. Use of the Weighted Histogram Analysis Method for the Analysis of Simulated and Parallel Tempering Simulations. *J. of Chemical Theory and Computation*, 3, 1, (Jan. 2007), 26–41.
[14] Peter Deuflhard and Marcus Weber. 2005. Robust perron cluster analysis in conformation dynamics. *Linear algebra and its applications*, 398, 161–184.
[15] Tu Mai Anh Do, Loıc Pottier, Rafael Ferreira da Silva, Silvina Caıno-Lores, Michela Taufer, and Ewa Deelman. 2022. Performance assessment of ensembles of in situ workflows under resource constraints. *Concurrency and Computation: Practice and Experience*, e7111.
[16] Tu Mai Anh Do et al. 2022. Co-scheduling Ensembles of In Situ Workflows, 43–51.
[17] Ciprian Docan, Manish Parashar, and Scott Klasky. 2010. Dataspaces: an interaction and coordination framework for coupled simulation workflows. In *Proc. of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC)*, 25–36.
[18] S Doerr, MJ Harvey, Frank Noé, and G De Fabritiis. 2016. HTMD: high-throughput molecular dynamics for molecular discovery. *J. of Chemical Theory and Computation*, 12, 4, 1845–1852.
[19] Y Duan et al. 2003. A Point-Charge Force Field for Molecular Mechanics Simulations of Proteins Based on Condensed-Phase Quantum Mechanical Calculations. *J. of Computational Chemistry*, 24, 16.
[20] Michael Feig, Matin Abdullah, Lennart Johnsson, and B Montgomery Pettitt. 1999. Large scale distributed data repository: design of a molecular dynamics trajectory database. *Future Generation Computer Systems (FGCS)*, 16, 1, 101–110.
[21] Giacomo Fiorin, Michael L. Klein, and Jérôme Hénin. 2013. Using collective variables to drive molecular dynamics simulations. *Molecular Physics*, 111, 22-23, 3345–3362.
[22] S Gnanakaran, Hugh Nymeyer, John Portman, Kevin Y Sanbonmatsu, and Angel E Garcıa. 2003. Peptide folding simulations. *Current Opinion in Structural Biology*, 13, 2, 168–174.
[23] Jérôme Hénin, Tony Lelièvre, Michael R Shirts, Omar Valsson, and Lucie Delemotte. 2022. Enhanced sampling methods for molecular dynamics simulations. *arXiv preprint arXiv:2202.04164*.
[24] S Herbein et al. 2016. Scalable I/O-Aware Job Scheduling for Burst Buffer Enabled HPC Clusters. In *Proc. of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 69–80.
[25] Wei Jiang et al. 2014. Generalized scalable multiple copy algorithms for molecular dynamics simulations in NAMD. *Computer Physics Communications*, 185, 3, 908–916.
[26] Travis Johnston, Boyu Zhang, Adam Liwo, Silvia Crivelli, and Michela Taufer. 2017. In situ data analytics and indexing of protein trajectories. *J. of Computational Chemistry*, 38, 16, 1419–1430.
[27] Perilla JR et al. 2015. Molecular dynamics simulations of large macromolecular complexes. *Curr Opin Struct Biol.*, 31, 64–74.
[28] Anand Kumar et al. 2015. DCMS: A data analytics and management system for molecular simulation. *J. of Big Data*, 2, 1, 1–22.
[29] Sriram Lakshminarasimhan et al. 2013. Scalable in situ scientific data encoding for analytical query processing. In *Proc. of the 22nd International Symposium on High-performance Parallel and Distributed Computing (ICPP)*, 1–12.

[30] Yan Li and Zigang Dong. 2016. Effect of clustering algorithm on establishing Markov state model for molecular dynamics simulations. *J. of Chemical Information and Modeling*, 56, 6, 1205–1215.

[31] Lindahl, Abraham, Hess, and Van Der Spoel. 2021. GROMACS 2021.4 Source code. (Nov. 2021). Retrieved Dec. 4, 2022 from https://zenodo.org/record/5636567.

[32] Ning Liu et al. 2012. On the Role of Burst Buffers in Leadership-class Storage Systems. In *Proc. of the 28th IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, 1–11.

[33] Huong Luu et al. 2015. A multiplatform study of i/o behavior on petascale supercomputers. In *Proc. of the 24th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 33–44.

[34] Yusuke Naritomi and Sotaro Fuchigami. 2011. Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: The case of domain motions. *J. of Chemical Physics*, 134, 6, (Feb. 2011), 065101. Retrieved Dec. 22, 2022 from.

[35] Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. 2013. Identification of slow molecular order parameters for Markov model construction. *J. of Chemical Physics*, 139, 1, 07B604_1.

[36] J. C. Phillips et al. 2005. Scalable molecular dynamics with NAMD. *J. of Computational Chemistry*, 26, 16, 1781–1802.

[37] Joshua L Phillips, Michael E Colvin, and Shawn Newsam. 2011. Validating clustering of molecular dynamics simulations using polymer models. *BMC bioinformatics*, 12, 1, 1–23.

[38] Sander Pronk et al. 2015. Molecular simulation workflows as parallel algorithms: the execution engine of Copernicus, a distributed high-performance computing platform. *J. of Chemical Theory and Computation*, 11, 6, 2600–2608.

[39] Arvind Ramanathan, Ji Oh Yoo, and Christopher J Langmead. 2011. On-the-fly identification of conformational substates from molecular dynamics simulations. *J. of Chemical Theory and Computation*, 7, 3, 778–789.

[40] Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science*, 344, 6191, 1492–1496.

[41] Mary A Rohrdanz, Wenwei Zheng, Mauro Maggioni, and Cecilia Clementi. 2011. Determination of reaction coordinates via locally scaled diffusion map. *J. of Chemical Physics*, 134, 12, 03B624.

[42] Andrej Sali, Tom L Blundell, et al. 1993. Comparative protein modelling by satisfaction of spatial restraints. *J. of Molecular Biology*, 234, 3, 779–815.

[43] Kento Sato et al. 2014. A User-Level Infiniband-Based File System and Checkpoint Strategy for Burst Buffers. In *Proc. of the 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid)*, 21–30.

[44] Jianyin Shao, Stephen W Tanner, Nephi Thompson, and Thomas E Cheatham. 2007. Clustering molecular dynamics trajectories: 1. Characterizing the performance of different clustering algorithms. *J. of Chemical Theory and Computation*, 3, 6, 2312–2334.

[45] [n. d.] Summit. (). Retrieved Dec. 4, 2022 from https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[46] 2009. *Poincaré's Legacies, Part II, pages from year two of a mathematical blog*. American Mathematical Society. Chap. 1.5 When are eigenvalues stable?

[47] Michela Taufer, David P. Anderson, Pietro Cicotti, and Charles L. Brooks III. 2005. Homogeneous Redundancy: a Technique to Ensure Integrity of Molecular Simulation Results Using Public Computing. In *Proc. of the 19th International Parallel and Distributed Processing Symposium (IPDPS*.

[48] Michela Taufer et al. 2019. Characterizing in situ and in transit analytics of molecular dynamics simulations for next-generation supercomputers. In *Proc. of the 15th IEEE International Conference on eScience (eScience)*, 188–198.

[49] Zacaria T. et al. [n. d.] Frontier supercomputer debuts as world's fastest, breaking exascale barrier. https://www.ornl.gov/news/frontier-supercomputer-debuts-worlds-fastest-breaking-exascale-barrier. Online; accessed 23 December 2022. ().

[50] Andrzej Zieba and Piotr Ramza. 2011. Standard Deviation of the Mean of Autocorrelated Observations Estimated with the Use of the Autocorrelation Function Estimated From the Data. *Metrology and Measurement Systems*, 18, 4, 529–542.

[51] Maxwell I Zimmerman, Justin R Porter, Xianqiang Sun, Roseane R Silva, and Gregory R Bowman. 2018. Choice of adaptive sampling strategy impacts state discovery, transition probabilities, and the apparent mechanism of conformational changes. *Journal of chemical theory and computation*, 14, 11, 5459–5475.

[52] Matthew C Zwier et al. 2015. Westpa: an interoperable, highly scalable software package for weighted ensemble simulation and analysis. *Journal of chemical theory and computation*, 11, 2, 800–809.