Characterizing In Situ and In Transit Analytics of Molecular Dynamics Simulations for Next-generation Supercomputers

Stephen Thomas*, Michael Wyatt*, Tu Mai Anh Do†, Loïc Pottier†, Rafael Ferreira da Silva†, Harel Weinstein‡ Michel A. Cuendet‡¶‖ Trilce Estrada§ Ewa Deelman†, Michela Taufer*

*The University of Tennessee at Knoxville, Knoxville, TN, USA
†Information Sciences Institute, University of Southern California, Marina Del Rey, CA USA
‡ Weill Cornell Medicine, Cornell University, New York, NY USA
§ University of New Mexico, Albuquerque, NM USA
¶ Swiss Institute of Bioinformatics, Lausanne, Switzerland
¶ Department of Oncology, Lausanne University Hospital, Lausanne, Switzerland
{sthoma99,wyatt12,mtaufer}@utk.edu {tudo,lpottier,rafsilva,deelman}@isi.edu

***Abstract*—Molecular Dynamics (MD) simulations executed on state-of-the-art supercomputers are producing data at rates faster than it can be written out to disk. In situ and in transit analysis of data generated by MD simulations reduce the original volume of information by several orders of magnitude, thereby alleviating the negative impact of I/O bottlenecks. This work focuses on characterizing the impact of in situ and in transit analytics on the overall MD workflow performance, and the capability for capturing rapid, rare events in the simulated molecular system. The MD simulation and analysis processes share data via remote direct memory access (RDMA) using DataSpaces. Our metrics of interest are time spent waiting in I/O by the MD simulation, lost frames of the MD simulation, and idle time of the analysis. We measure these metrics for a diverse set of molecular systems and characterize their trends for in situ and in transit configurations. We then model which frames are dropped and which ones are analyzed for a real use case. The insights gained from this study are generally applicable for in situ and in transit workflows that require optimization of parameters to minimize loss in workflow performance and analytic accuracy.**

***Index Terms*—Scientific workflows, data analytics, performance, workload modeling, remote direct memory access.**

I. Introduction

This paper targets one of the most common simulation types on petascale and, very likely, exascale machines: Molecular Dynamics (MD) simulations studying the time evolution of a molecular system at atomic resolution. The fields of chemistry, material sciences, molecular biology, and drug design widely utilize MD simulations. The system sizes and time-scales accessible to MD simulations have been steadily increasing [1]. Next-generation High-Performance Computing (HPC) systems will have dramatically larger compute performance than do current systems. The increase in computing capability directly translates into the ability to execute an increasing number of longer simulations and thus to expand the range of biomolecular phenomena that can be studied by MD simulation. This also means generating more data than needs to be analyzed in terms of the number and length of MD trajectories. Because of power constraints, however, the I/O bandwidth and parallel file system capacity of next-generation HPC systems is not likely to grow at the same pace.

In this paper, we address the challenges facing MD simulations on next generation supercomputers by transforming the traditionally performed centralized MD analysis to a distributed in situ or in transit analysis. We define a workflow that analyzes data as it is generated. Note that we focus on the analysis of MD-generated data (i.e., capturing rare events and monitoring convergence of observables based on inherently noisy and high-dimensional MD outputs) rather than on the MD process (i.e., efficient computation of molecular interactions, parallelization, GPU acceleration) itself. By leveraging the standard formats of MD-generated outputs, we design our workflow to be compatible with all of the most-used MD codes. Our workflow does not require the recompilation of any single MD code or the redesign of any MD script. Instead, it captures outputs in memory at runtime as they are generated and uploads the data into an in-memory staging area using DataSpaces [2].

As a prototypical example of compute-intensive data analysis, we model a suite of collective variables that describe molecular structures in terms of distance matrices and output values from linear algebra operations. This type of analysis represents a common workload in MD [3], [4]. The suite integrates into our producer/consumer execution pattern workflow (i.e., the simulation produces data, and the analytics consume data). We model the pattern to demonstrate three pertinent scenarios: (a) when simulations are idle waiting in I/O because the analytics are not able to consume MD frames at the same pace as simulation; (b) when the same simulations, rather than waiting in I/O, drop frames subjected to I/O contention causing loss of information; and (c) when the resources used for the analytics are idle because the simulations are not able to produce MD frames at the same pace. While schedulers can mitigate underutilized resources, losing a fraction of frames may result in inaccurate MD solutions, opening the need to model which frames may be analyzed and the impact on the accuracy of the MD. To this end, we develop a 2-step model that predicts which frames are dropped and which ones are analyzed. We apply the model to study the consequences of dropping frames on simulation accuracy for the real use case of a protein exhibiting a rare event conformation change.

The rest of this paper presents background (Sec. II), the setting of our modeling environment (Sec. III), our modeling (Sec. IV), and its application to a 1BDD trajectory (Sec. V).

II. Background

MD simulations complement wet-lab experiments by providing molecular and atomic resolution information not directly accessible by experiment. Specifically, MD simulations computationally replicate the behavior of a physical molecular system by iterating a two-step algorithm. First, the interactions between atoms are calculated by using a model called a force field, which describes the total energy of the system as a mathematical function of atomic positions and a set of parameters calibrated to reproduce the inter-atomic forces acting on each atom in the molecular structure. Second, based on the calculated forces on each atom of the system, their positions are advanced by solving Newtons equations on a small time step. Calculating long-range forces in systems composed of several hundreds of thousands of atoms is by far the most compute-intensive part of the calculation. An MD job reproduces the evolution of a molecular system under a specified set of thermodynamic conditions (e.g., temperature, volume, and pressure) and external forces (if needed) by computing and writing to storage the systems atomic coordinates, and other relevant properties, at regular intervals as the job evolves. The sequence of molecular conformations (i.e., the trajectory) follows the physics of Boltzmann ensembles of particles and is written to disk. A large-scale MD simulation would typically include an ensemble of MD jobs (as many as hundreds or thousands) that can run on different compute nodes and produce independent trajectories (replicas) [5], [6]; each replica simulates the same molecular system starting from different initial conditions (e.g., positions, velocities) but with the same system parameters (i.e., composition and external forces). Different simulations of the same system or similar systems under different conditions (e.g., temperature, protein mutants, and drug variants) are analyzed comparatively from separate ensemble runs of MD simulations. 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 paper, we do not target the force field development, computational efficiency, or parallelization aspects that have been extensively addressed by the scientific community [7]–[9].

Given the rapid fluctuations of a molecular system at room temperature, properties extracted from a single conformation or single event are not relevant if considered in isolation. Instead, measured properties correspond to ensemble averages incorporating all molecular configurations that form local states. MD simulations reproduce the thermal motions of the molecules and produce ensembles of molecular conformations compatible with a set of given thermodynamic conditions. In general, MD simulations have been utilized to (1) classify the dynamics and dynamic properties of molecular systems; (2) reveal and characterize rare events and metastable states of the molecules that have functional significance (multiple realizations are required for statistical significance); and (3) calculate and monitor the convergence of ensemble averages of observables that can be compared with experiments and/or predict new observations. Formally, a rare event is a transition from one metastable region in conformational space to another, such as a folding phase of a protein or conformational changes of protein domains related to functions such as transport, signaling, or catalysis. Rare events can be monitored efficiently with the use of a small set of statistical metrics called collective variables (CVs) that capture relevant molecular motions [10]

III. Setting the Modeling Environment

A. Distance Matrices: Proxies for Structural Changes

At a given time, t, an MD simulation writes a snapshot or frame of the molecular system to memory. The frame contains all atomic coordinates and complete structural information on the m amino acids $(k_1, k_2, ..., k_m)$ comprised in the structure. Trajectory analysis commonly measures the structural changes of a frame with respect to past frames of the same trajectory or frames in other trajectories, without directly comparing the data contained in either frame. We want to capture two types of structural changes: changes within single amino acid segments and changes of two amino acid segments with respect to each other. To this end, we simplify the molecular system made of m amino acids by extracting the positions of the m α-Carbon (Cα) backbone atoms $(x_i\ ,y_i\ ,z_i)$ for $1 \leq i \leq m$ amino acids and using the backbone atoms to build distance matrices that, together with the matrix eigenvalues, are proxies for structural changes in the molecular system itself [11].

Given a frame, we build two types of distance matrices: (1) Euclidean distance matrices from the positions of the corresponding $C\alpha$, $D = [d_{ij}]$ with $i, j = 1, ..., m$ and

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \qquad (1)$$

to capture changes within single amino acid segments and (2) bipartite distance matrices $B = [b_{ij}]$ to capture changes of two amino acid segments S_1 and S_2 with respect to each other. B is of size $m \times m$, with elements defined by

$$b_{ij} = \begin{cases} d_{ij}, & \text{if } i \in S_1 \text{ and } j \in S_2 \\ 0, & \text{otherwise.} \end{cases} \qquad (2)$$

The Euclidean distance matrix D and the bipartite distance matrix B have three fundamental properties: they are symmetric, diagonal elements are zeros, and off-diagonal elements are strictly positive. Johnston and co-authors show how the larger eigenvalues of each one of these matrices can be computed in isolation (i.e., without keeping other frames in memory) and can serve as a CV that, unlike other CVs, can identify structural changes of substructures [11]. By computing these CVs on each frame, we drop the requirement to keep frames in memory as the simulation evolves.

We leverage this work to build a suite of analysis scenarios with different numbers of matrices and different matrix sizes. The number of matrices and matrix sizes are dependent on the system size and non-overlapping segment lengths in which we cut the molecular system into strings of amino acids. The scenarios range from the fine-grained study of as many substructures as possible (with segment length as small as two amino acids) in which we generate many small matrices, to the coarse-grained study of the entire molecular system through one single matrix with a size matching the number of amino acids in the system. Figure 1 shows the number of matrices (dotted line) and matrix size (solid line) when bipartite matrices are generated with different segment lengths. In this figure, we use N_α=1266 as an example, where the minimum segment length applicable to the analysis is 2 and the maximum segment length is $^{N_\alpha}/_2 = 633$. The number of matrices and matrix size impact the computational cost of the analytics (i.e., the larger the number of the matrices, the larger the number of eigenvalues computed) and the memory use (i.e., the larger the matrices, the larger the memory use). The eigenvalues of the matrices generated from a given frame are computed sequentially in this study.

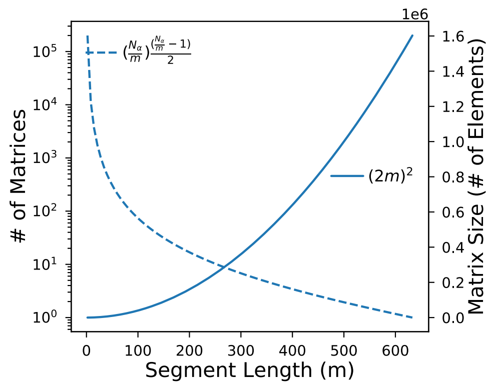


Fig. 1: Number of distance matrices and their size as a function of the non-overlapping segment lengths in which a molecular system with $N_\alpha = 1266$ can be divided.

B. Molecular Systems: From Small to Large Systems

We consider three different molecular systems of increasing size (i.e., number of atoms). Figure 2 depicts the three molecular systems. Table I provides details about the individual molecular systems: the size described in terms of small, medium, and large; the name of the molecular system (MD system); the number of atoms, N; the number of carbon atoms, N_α; and the estimated number of steps that the simulation can perform per second wall clock time (TPS). For the estimations, we run NAMD benchmarks [12] on GPUs of high-end clusters and interpolate the output of the benchmarks for our three molecular systems.

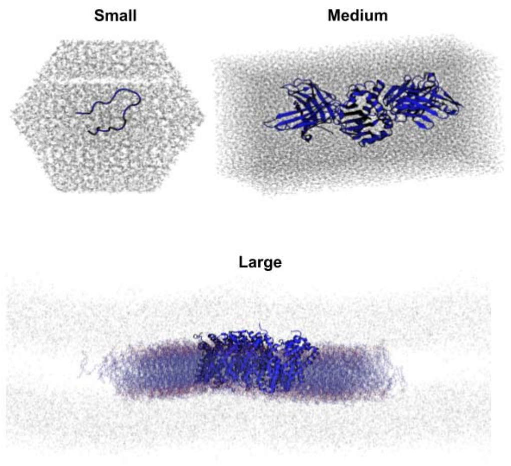


Fig. 2: Visualization of the three molecular systems considered for this study. Table I provides more details about the individual molecular systems.

TABLE I: Our three molecular systems and characteristics.

Size	**MD System**	N	N_α	**TPS**
Small	Trp cage	12,619	20	511
Medium	T cell receptor	81,092	605	460
Large	Gltph	270,088	1,266	318

C. MD Workflow: Example of Producer-consumer Patterns

Our workflow integrates MD simulations with in situ or in transit analytics. The workflow is structured as a producer/consumer pattern with the MD simulation producing snapshots $(x_i(t), y_i(t), z_i(t))$ (i.e., frames) output at a regular interval of steps (i.e., strides), and one or more analytics modules serving as the consumer. Figure 3 illustrates the workflow used in this study. We assume as the system size decreases, the possible structural changes become faster, requiring the stride of the associated MD simulation to become smaller in order to capture all the changes. Thus, the rate at which frames are generated for the analysis depends on the molecular system size. For each molecular system, we select four strides that scale to system size and follow the ratio of $1 : 5 : 10 : 50$. For example, for the large system, the stride values are 100, 500, 1000 and 5000; a frame is output every $100 * \Delta t$, $500 * \Delta t$, $1000 * \Delta t$, and $5000 * \Delta t$, where Δt is the time step size and is computed as the inverse of the TPS. We use elements of Plumed, a plug-in software package compatible with many state-of-the-art CV calculation packages [13], to capture a frame when generated by the MD code. Plumed is implemented as a plug-in and thus no changes to the MD code are needed: we engineer a Plumed function to read a frame

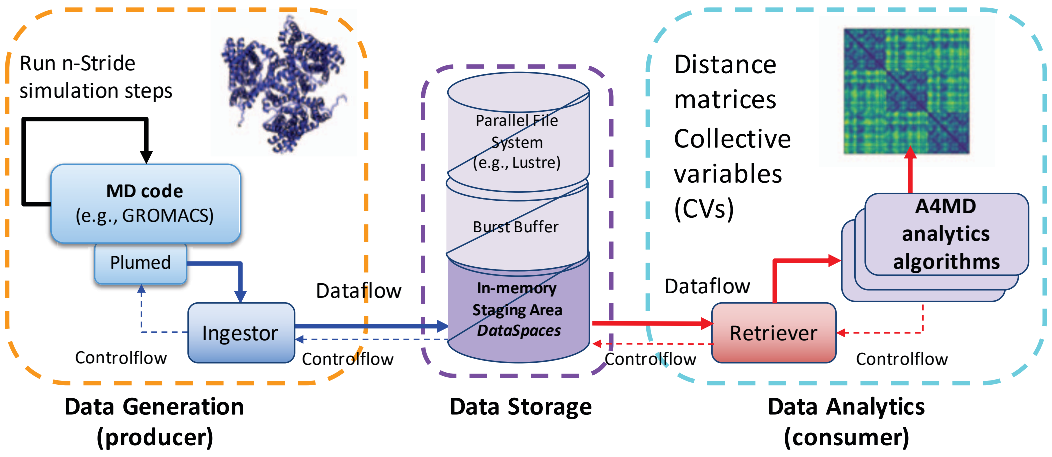

Run n-Stride simulation steps
MD code (e.g., GROMACS)
Plumed
Ingestor
Controlflow
Dataflow
Controlflow
Parallel File System (e.g., Lustre)
Burst Buffer
In-memory Staging Area DataSpaces
Controlflow
Distance matrices
Collective variables (CVs)
A4MD analytics algorithms
Dataflow
Retriever
Controlflow
Data Generation (producer)
Data Storage
Data Analytics (consumer)

Fig. 3: Our workflow integrating in situ or in transit analytics.

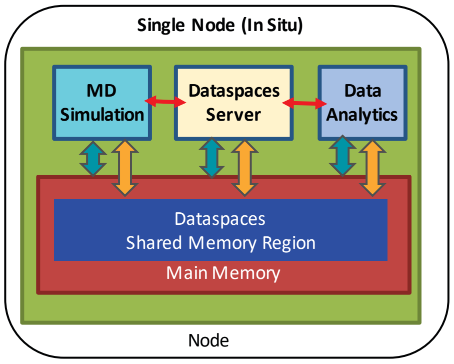
from the MD simulation memory space and transfer it to the DataSpaces shared memory via an ingestor module.

DataSpaces [2], a memory-to-memory framework using remote direct memory access (RDMA), serves as our data transport layer (DTL). It enables efficient and scalable data sharing (i.e., the sharing of trajectory frames) between the MD simulations and the analytics modules. It uses a client/server architecture: the server is a virtual shared memory space that can be concurrently queried by multiple clients. The use of RDMA offers scalable communications between the server and each client. The DataSpaces shared memory is accessed by the ingestor fed by Plumed; a retriever module passes the frame to the analytics modules. The ingestor and retriever use a simple key-value representation to coordinate the data movement, where the key is the time step and the value is the data. The size of the shared memory buffer is fixed. In this work, a DataSpaces buffer size corresponding to the size of a single frame is considered. DataSpaces supports both a default setting, which blocks the producer from writing data from the next time step until the consumer finishes reading the current time step frame and an asynchronous setting, which allows for managing the synchronization between the producer and consumer by the user. We use both mechanisms in this study.

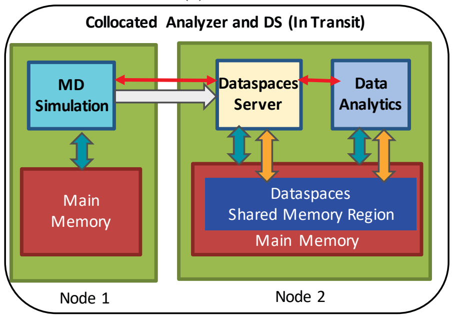
The analytics modules used in this study are `python` modules, but can easily be extended to use modules implemented in other languages. The modules generate one or more matrices per frame; for each matrix, we compute eigenvalues as described in Section III-A.

D. In Situ and In Transit Configurations

We run two workflow configurations that represent an example of in situ and in transit analytics on Haswell nodes of NERSC's Cori. Each Haswell node has two 16-core Intel Xeon processors, 128GB memory, and are connected by a Cray Aries interconnect. The configuration in Figure 4a is representative of the in situ analytics workflow, where data generation and data consumption share the same resources (in this case the same node). Figure 4b, with its collocated analyzer and DataSpaces server (DS) on dedicated resources (in this case Node 2), is representative of an in transit analytic workflows. In this latter configuration, data produced by an MD simulation on one node must be transferred to another node where it is analyzed.



(a) in situ



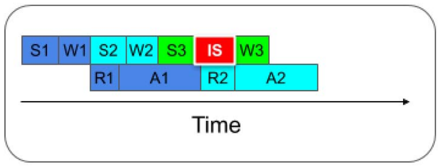
(b) in transit

Fig. 4: In situ (a) and in transit (b) analytic workflows.

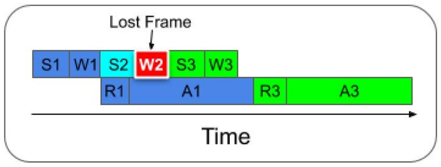
E. Execution Patterns and Metrics

We classify our workflows in terms of their execution patterns into: (a) workflows with fast production of MD frames and slow analytic processing of the frames by the analytic modules; (b) workflows with slow production of MD frames and fast analytic processing of the frames by the analytic modules; and (c) balanced workflows with MD frames that are analyzed at the same rate as they are produced.

In the first type of workflow, because the analysis is not able to consume the frame in a timely manner, the MD simulation either waits in I/O to write new frames to the in-memory staging area of DataSpaces (idle simulation time or IS in Figures 5a) or discards any frame that cannot be ingested into the staging area (lost frame in Figures 5b). Both scenarios negatively impact the scientific throughput delivered by the MD workflow: the first with a lower number of frames over the total simulation time and the second with a loss of MD information (and possible loss of accuracy for the MD simulation).



(a) with simulation idle time



(b) with lost frame

Fig. 5: Two execution patterns associated with fast MD simulations and slow analytics with S1, S2, and S3 referring to MD simulation times; W1, W2, and W3 referring to write times to DataSpaces; R1, R2, and R3 to read times from DataSpaces; A1, A2, and A3 to analytics times; and IS to simulation idle times. Note how in (b) the frame associated to W2 is discarded.

In the second type of workflow, because the MD simulation generates a new frame with large strides (larger than the time needed to analyze the frame), the analytics are waiting in I/O and the associated resources are idle (and thus, may be available for other analyses or for the MD simulation itself). Figure 6 shows this scenario. We characterized this scenario as analysis idle time or IA. While there is not direct negative impact on the science delivered, this scenario can result in under-utilized resources.

IV. Modeling Data Analytic

A. Modeling Waiting in I/O and Analysis Idle Time

We measure and observe trends for the time spent waiting in I/O for the simulation and idle time for the analytics under

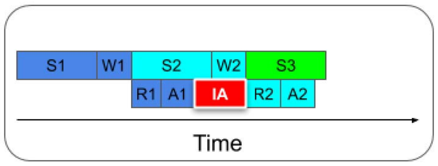
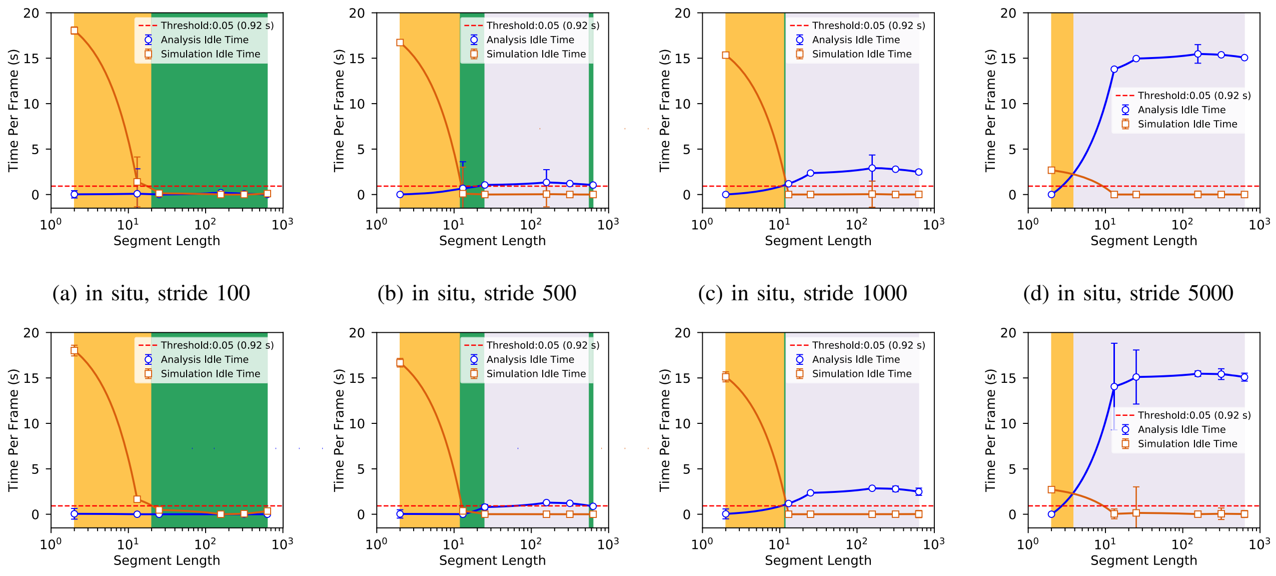


Fig. 6: Execution pattern associated to slow MD simulations and fast analytics with S1, S2, and S3 referring to MD simulation times; W1 and W2 referring to write times to DataSpaces; R1 and R2 to read times from DataSpaces; A1 and A2 to analytics times; and IA to analysis idle times due to waiting for the MD simulation.

different MD workflow settings (i.e., in situ vs. in transit, different strides, and different segment lengths). We cut the molecular systems into equal sized segments; the segment lengths considered are m=[2, 13, 25, $N_\alpha/8, N_\alpha/4$, $N_\alpha/2$] where N_α is the total number of C_α backbone atoms, one per amino acid, in the system. For each segment length, we build bipartite matrices for every $(\frac{N_\alpha}{m})\frac{(\frac{N_\alpha}{m}-1)}{2}$ pairs of segments. We then compute the largest eigenvalues for each of the generated bipartite matrices. Figure 7 shows the measured time spent by the MD simulation waiting in I/O (simulation idle time) and analytics idle time for each stride size of the large molecular system. Similar outcomes are observed for the other medium and small systems. The first row in Figure 7 (7a-7d) shows the measurements obtained with the in situ configuration and the second row of Figures 7 (7e-7h) shows the in transit configuration. In Figure 7, from left to right, the columns indicate stride sizes of 100 (7a,7e), 500 (7b,7f), 1000 (7c,7g), and 5000 (7d,7h) The dots in the figures show the measured times (i.e., squares are the simulation idle times and the circles are the analysis idle times). The error bars show the standard deviation of the measured times for five independent trajectories, each with 1000 frames. To identify the three execution patterns defined in Section III-E, we assume a threshold marked by the dashed horizontal line. The line represents 5% of the largest idle time observed in all the tests. Specifically, when analysis idle time is above the threshold and larger than the simulation idle time, we observe the execution pattern (a) in Section III-E. Execution pattern (b) is observed when simulation idle time is above the threshold and larger than the analysis idle time. Execution pattern (c) is observed when both simulation idle time and analysis idle time are below the threshold. The execution pattern (a), (b), and (c) are shown by the blue, yellow, and green shaded regions in the figure.

We do not observe a significant difference in the measured idle times, for simulation or analysis, between the in situ and in transit configurations. This due to low communication overhead: the two nodes of the in transit configuration are located within close proximity to each other and the size of the frames being transferred are small (7MB).

We observe several trends across Figure 7: At the smallest



(a) in situ, stride 100 (b) in situ, stride 500 (c) in situ, stride 1000 (d) in situ, stride 5000

(e) in transit, stride 100 (f) in transit, stride 500 (g) in transit, stride 1000 (h) in transit, stride 5000

Fig. 7: The observed analysis idle times and simulation idle time with the in situ configurations and in transit configurations.

segment length, simulation idle time is always larger than analysis idle time. As stride size increases (moving left to right from Figures 7a to 7d) the simulation idle time decreases for all segment lengths (e.g., for segment length two we see an 85% reduction from stride 100 to 5000). In contrast, the analysis idle time is 0 for segment length two and all strides, exhibiting the inverse of the simulation idle time trends. As stride increases, analysis idle time increases significantly (e.g., for segment length $N_\alpha/2$ we see a 99.99% increases from stride 100 to 500). We also observe regions in the different parameter settings where the producer/consumer execution pattern is balanced (i.e., small simulation idle time and analysis idle time). Such regions are predominant for small strides and shrink as stride increases.

Rather than allow the MD simulation to wait in I/O, the simulation may instead discard frames that cannot be consumed in a timely fashion by the analytics module. In the next section, we study the workflow when the execution pattern allows discarding frames, as defined in Section III-E.

B. *Modeling Frames Analyzed vs. Frames Lost*

To study the consequences of dropping frames on simulation accuracy, we develop a 2-step model that predicts which frames are dropped and which ones are analyzed. Our model is applicable for both in situ and in transit analytics as no major differences were observed in the patterns outlined in the previous section. Additionally, based on observations in Section IV, we focus the modeling in the region with significant simulation idle times and we use small segment lengths (i.e., 2, 4, 6, 8, 10, 12, and 14).

The first step of the model predicts the fraction of frames analyzed for a given MD simulation, starting from the produced frames. The production of frames is dictated by the stride. The consumption of each frame is dictated by the number of matrices and size of matrices, which are determined by segment length and system size. We define a new term, matrix period, which is the amortized time for production of each matrix and is calculated by dividing the time for a frame to be produced by the number of matrices produced by that frame. We combine the parameters dictating the production and consumption of frames into our model in terms of matrix period and size. To this end, we measure the relationship between the fraction of frames analyzed, f, and the matrix period and matrix size by (a) collecting data from MD simulations with stride length of 100, 500, 1000, and 5000 and the small segment lengths described above, as seen in Figure 8a, and (b) fitting a degree 2 polynomial surface to the data. The surface fitted is defined by:

$$\begin{aligned} f = -0.25X_1 - 0.24X_2 + 0.01X_1^2 + \\ 0.02X_1X_2 + 0.002X_2^2 + 3.48, \end{aligned} \tag{3}$$

where the independent variables X_1 and X_2 are the matrix period and the size of individual matrices being produced. The smaller coefficients for X_2 indicate that the rate of production of matrices (i.e., matrix period) is more important to the fraction of frames analyzed than the size of matrices.

We validate the first step of our 2-step model by measuring the error between our surface-predicted and observed values of f. Figure 8b shows the predicted f given a matrix period and matrix size using our fitted surface. The top-left and bottom-right regions shown in white are regions in the parameter space which are excluded in this study. These regions represent unlikely scenarios for MD systems (i.e., parameter spaces that are not seen with typical MD workloads). Figure 8c shows the absolute error between our data and the fitted surface is small. We further validate our work and calculate a R^2 value of 0.88 for the fitted surface.

The second step of our 2-step model takes the output of the first step (i.e., predicted f) and models the distribution of frames analyzed for a given trajectory. We start with the experimentally observed data for analyzed frames of MD simulations in the bottom row of Figure 9 (i.e., 9e,9f,9g and 9h). Due to space constraints, we show only four of the eight MD simulations, each with a different segment length (i.e., 2, 6, 10, and 14). The fraction of analyzed frames are 0.07, 0.41, 0.69, and 0.86 for e, f, g, and h respectively. The histograms depict the discrete distribution for the number of frames lost between analyzed frames in each simulation. For example, e shows that for a segment length of 2, there are either 14 or 15 frames lost between each frame that is analyzed. We note that there at most two values (i.e., period of analyzed frames) that are populated in each histogram and that these values are dependent on the fraction of frames analyzed. We refer to these two periods as k and $k + 1$.

To model the trend in the bottom row of Figure 9, we determine the relationship between f and the discrete distribution of analyzed frame periodicity. We assume that every frame has a period of either k or $k + 1$ for a given value of f, where k is a positive integer. From these definitions we can write

$$\frac{1}{k+1} < f \leq \frac{1}{k} \tag{4}$$

and from that derive

$$\frac{1}{f} - 1 < k \leq \frac{1}{f}. \tag{5}$$

Because $k \in \mathbb{N}$, k is given by:

$$k = \left\lfloor \frac{1}{f} \right\rfloor , \tag{6}$$

where the $\lfloor\ \rfloor$ symbol indicates the "floor" function. Next, we let P and Q be total number of analyzed frames with periods k and $k + 1$. T the total number of analyzed frames. We can write:

$$f = \frac{\text{analyzed frames}}{\text{all frames}} = \frac{P + Q}{P * k + Q * (k + 1)} . \tag{7}$$

We then define $p = {}^{P}\!/_{T}$ and $q = {}^{Q}\!/_{T}$, the proportion of analyzed frames with period k and period $k + 1$. Since every analyzed frames has either a period of k or $k+1$, then $p+q = 1$. Dividing the numerator and denominator for the right side of Equation 7 by T gives us Equation 8.

$$f = \frac{p + q}{pk + q(k + 1)} \tag{8}$$

By applying the relation $p + q = 1$ in Equation 8, we arrive at the simple expression for q,

$$q = \frac{1}{f} - k, \tag{9}$$

where we can use Equation 6 to define q in terms of f:

$$q = \frac{1}{f} - \left\lfloor \frac{1}{f} \right\rfloor . \tag{10}$$

Finally, the probability of analyzing frames with a period of k is given by

$$p = \left\lfloor \frac{1}{f} \right\rfloor - \frac{1}{f} \tag{11}$$

Given a value of f, the proportion p and q of frames analyzed with periods k and $k + 1$, respectively, can be obtained using this model. These values are used to generate the discrete distribution of analyzed frames shown in the top row of Figure 9a - 9d. The similarity between the modeled periods and actual data shown in Figure 9e - 9h qualitatively validates the second step in our 2-step model.

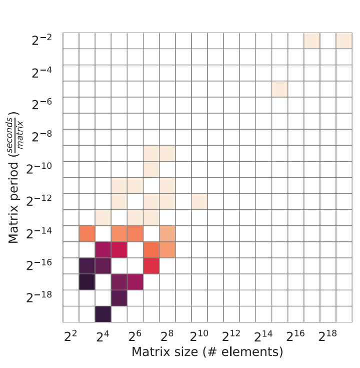
We quantitatively validate the second step of our 2-step model by plotting the observed and modeled probability, p, of period k for all experimentally observed f in Figure 10. Figure 10 shows the experimentally observed p as plotted points and modeled p as a solid line. We see an oscillating pattern for p as f tends toward 0. We observe that all data points fall along the line of expected p values and measure an R^2 of 0.995. This confirms the accuracy of the second step in our model.

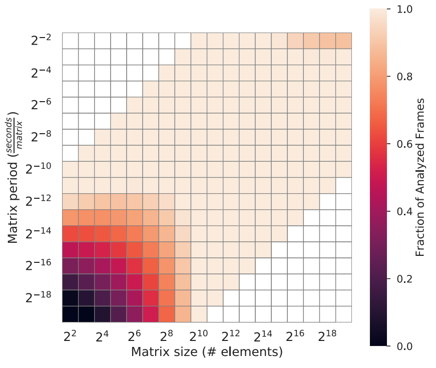
Our 2-step model, while initially defined with data from our molecular systems (from Section III-B), is general to any system. To demonstrate its generality, we apply our 2-step model to a real use case in the next section.

V. Use Case: 1BDD Protein

As an example of the impact of lost frames on the capability of capturing rare events in an actual MD simulation, we measure the relative change between three substructures over time in a trajectory of the B domain of staphylococcal protein A, whose NMR structure is available under entry 1BDD of the protein data bank [14]. The protein has three alpha-helical substructures (i.e., Helix 1, Helix 2, and Helix 3) that form relatively quickly (within the first 1,000 frames) of an MD simulation and exhibit a rare event conformational change immediately after. Initially the protein is compact (Frame 1300), then Helix 3 swings away from the other helices (i.e., Helix 1 and Helix 2), rendering the protein temporarily much less compact (Frame 1330). By Frame 1390, Helix 3 returns and the protein is once again compact, as shown in Figure 11.

The rare event is captured by the larger eigenvalues of the three bipartite matrices for Helices 1-2, Helices 1-3, and Helices 2-3). Figure 12 shows the sudden spike in the eigenvalues in Helices 1-3 and Helices 2-3 but lack of corresponding change in Helices 1-2, indicating Helix 3 moves relative to the other two. The frequency of the collected eigenvalues is at each MD step ($stride = 1$). For small systems such as this one with abrupt changes, such a high sampling frequency is recommended. Consequently, loss of frames, especially during critical phases in the structural evolution of such systems can result in the failure to detect important and rare events, or to compute accurate ensemble averages on trajectories containing short-lived transient states. To observe the effect of loss of frames, we calculate the largest eigenvalue from the Helix 1-2, 1-3 and 2-3 bipartite matrices between time


(a)


(b)

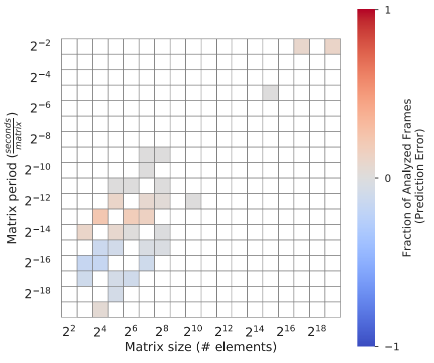
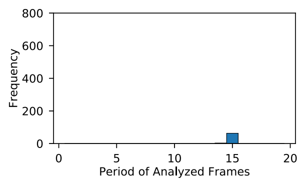
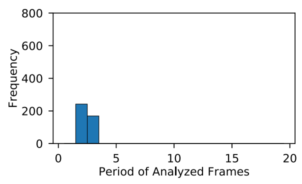
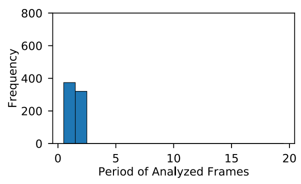
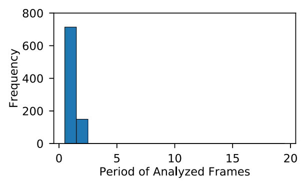

(c)

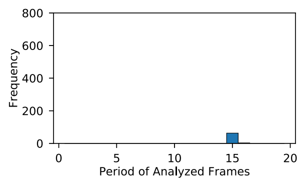
Fig. 8: The polynomial model (Equation 3) for predicting fraction of analyzed frames (b) is obtained by least-square fitting to the dataset shown in (a). The error of the model prediction with respect to the dataset is shown in (c).

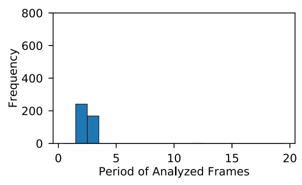

(a) predicted, stride 2

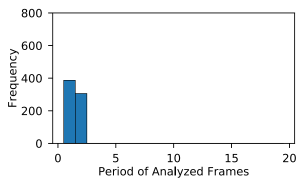

(b) predicted, stride 6


(c) predicted, stride 10


(d) predicted, stride 12


(e) observed, stride 2


(f) observed, stride 6


(g) observed, stride 10

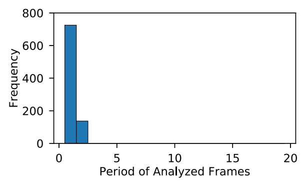

(h) observed, stride 12

Fig. 9: Predicted (a)-(d) and observed (e)-(h) periods at which frames are analyzed.

steps 1000 and 2000 for all fractions of frames analyzed. We use the 2-step model described in Section IV-B to generate the periodic indices of the analyzed frames. To model the random occurrence of the abrupt transition in the 1BDD protein conformations, we also randomize the first value of the analyzed frame index selected by the 2-step model, using a uniform distribution in the interval $[1, k]$. This introduces a randomized shift in the phase of the periodic pattern without changing the periodic pattern itself. In order to quantify the expected loss of information due to loss of periodic frames in time series data, we introduce the concept of effective number of frames, n_{eff}, in a correlated time series, defined as [15]

$$n_{\text{eff}} \doteq \frac{n}{1 + 2\sum_{k=1}^{n-1} \frac{n-k}{n} \rho_k}. \tag{12}$$

Here, the total number of samples (frames captured by analysis) is denoted by n and ρ_k represents the k-th point of the empirical autocorrelation function. In practice, since the accuracy of ρ_k decreases for long autocorrelation times, we truncate the sum at the first passage through zero [16]. For uncorrelated time series data, n_{eff} is equal to n. For autocorrelated time series data, decreasing the number of samples down to $n_{\text{eff}} = n$ is not expected to increase the standard expected error on the mean, $s_a(\overline{x})$. The unbiased sample estimator [17] for $s_a(\overline{x})$ is:

$$s_a(\overline{x}) = \sqrt{\frac{(n-1)}{n(n_{\text{eff}}-1)}}\, s. \tag{13}$$

Alternatively, we can quantify the empirical error on the mean, s_e, taking as reference the full-sample mean value, $\overline{x}(f = 1)$, i.e. the mean value of the observable (largest eigenvalue) when 100% of the frames are analyzed,

$$s_e(\overline{x}, f) = \frac{1}{M}\sqrt{\sum_{m=1}^{M} (\overline{x_m}(f) - \overline{x}(f=1))^2}. \tag{14}$$

Here, $\overline{x_m}(f)$ is the empirical error on mean in trial k when a fraction f of the frames are analyzed, and M is the number of trials.

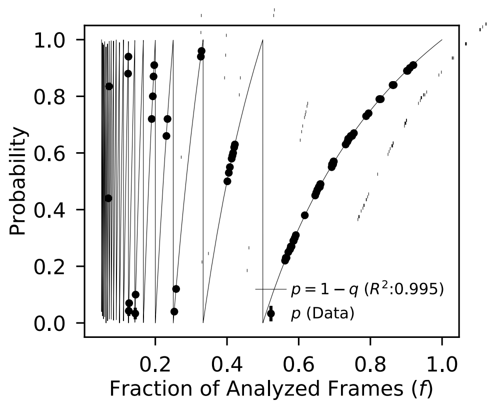
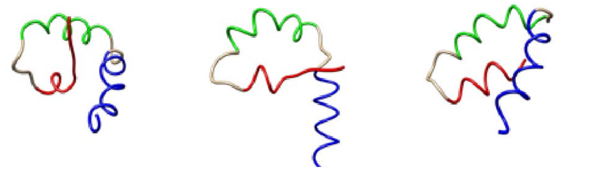


Fig. 10: The dots shows observed probability p for analyzed frame period k given different fraction of frames analyzed, f, in the large system. The solid line shows the modeled trend from our 2-step model, Equation 11, describing this behavior. The excellent quality of fit validates the model.



(a) Frame 1330 (b) Frame 1360 (c) Frame 1390

Fig. 11: Three helices of the 1BDD protein referred as Helix 1, Helix2, and Helix 3 are shown here in blue green and red.

This is observed for the largest eigenvalues data when low fractions of frames are analyzed. The fractions at which $n_{\text{eff}} = n$ is observed for Helices 1-2, 1-3 and 2-3 are 0.09, 0.06 and 0.03. Figure 13 shows the n_{eff} values estimated from the largest eigenvalue data as a function of the fraction of frames analyzed. For all three helix pairs, the n_{eff} is observed not to increase after 20% of the frames are analyzed. For Helix 1-2, 1-3 and 2-3 n_{eff} is observed to be roughly 66, 30 and 15, respectively. Based on the observation in Figure 14, both s_a and s_e increase substantially when large fractions of frames are lost, thereby increasing the uncertainty in the observables.

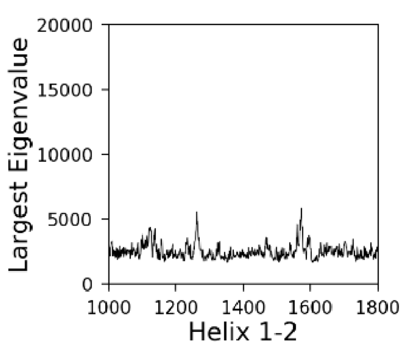
It is evident from the n_{eff} data in Figure 13 and the s_a and s_e data in Figure 14 that two regions exist where the fraction of frames analyzed has different implications. Above a threshold fraction of frames analyzed, f_t, whose value we expect to be close to n_{eff}/n, loss of frames does not affect the analytics. Below f_t, the analytics enter a regime where uncertainty increases. To further understand the effect of lost frames on the ability to detect abrupt structural transformations in the protein system, we closely examine at the region below f_t ($f = 0.01$) for the Helix 1-3 largest eigenvalue data. The error bars on the data point near the abrupt transition around time step 1390 indicate that the rare event can be lost at low fractions ($f < f_t$).

VI. Related Work

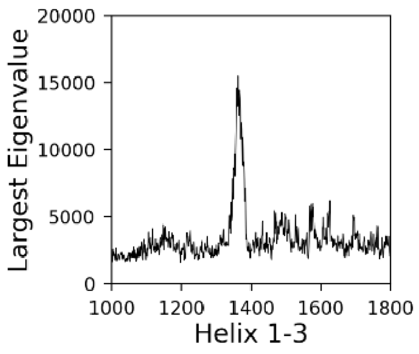
The use of traditional relational databases for trajectory analysis requires a posteriori trajectory upload and analysis of data using database functions (e.g., PostgreSQL [18], [19]). For finding similarities across datasets, many clustering methods have been applied to molecular structures and to MD trajectories in particular. For example, Shao et al. outline how there is not a one-size-fits-all clustering method [20]. Li and Dong describe the effect of clustering algorithms such as Bayesian clustering, k-means clustering, and kinetic clustering on establishing Markov state models for MD simulations [21]. Rodriguez presents a method for fast searches and identification of density peaks in trajectories [22]; the method requires the scientist to pick visually the number of peaks thought to be correct. Such clustering strategies are used with a post-simulation perspective but are not scalable on exascale machines. On the other hand, another class of methods, the data-streaming algorithms, are designed to analyze data on the fly. A framework for clustering evolving data streams [23] and data-streaming methods for high-quality clustering [24] are good resources regarding clustering of time series data. Some aspects of these methods, originally not developed for MD data, will be integrated in our framework. For classifying molecular structures in MD simulations, work has been done based on knowledge of the protein states a priori [25]–[27].

Software tools are available for comparing metrics in a distributed fashion. One such method is dynamic tensor analysis [28]. 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 with our technique. Centralized algorithms [29], [30] make metrics analyses inefficient when dealing with large proteins and long trajectories; assessments rely only on synthetic folding trajectories. Hybrid approaches have been introduced to handle big data analysis problems in the HPC context [31], [32]. These approaches combine in situ and in transit processing for extreme-scale scientific analysis such as topological analysis, descriptive statistics, and visualization.

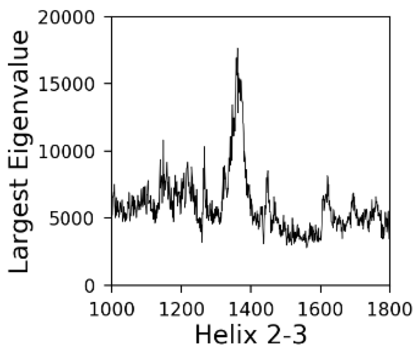
We note some recent efforts to manage ensembles of trajectories on large distributed infrastructures. Such a framework has been developed for NAMD using the parallel programming system Charm++, on which NAMD is built, and implementing some of the enhanced sampling methods outlined above [6]. A similar platform has been proposed for Gromacs, based on the distributed high-performance computing platform Copernicus [33]. The high-throughput MD [34] framework developed around the program ACEMD is a Python interface that supervises MD data generation and a posteriori analysis. At this point, none of these frameworks is tightly integrated with in situ trajectory analysis.



(a) Helices 1-2



(b) Helices 1-3



(c) Helices 2-3

Fig. 12: Largest eigenvalue for Helices 1-2, 1-3, and 2-3.

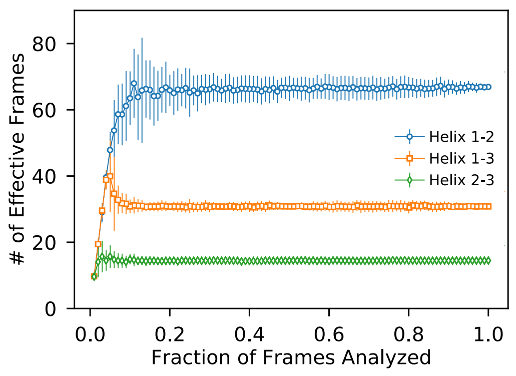


Fig. 13: Effective number of frames for given fractions of frame analyzed.

VII. Conclusions

This paper presents the characterization of in situ and in transit analytics of MD simulations for supercomputers and the modeling of MD workloads, including simulations that are idle waiting in I/O because the analytics are not able to consume MD frames at the same pace that simulations produce frames; or that, rather than waiting in I/O, simulations drop frames subjected to I/O contention causing loss in information. We develop a novel 2-step model to predict the fraction of analyzed frames and the exact frames that are lost in a given MD trajectory. We show that our model fits observed data with R^2 values of 0.88 and 0.99 for Step 1 and Step 2 respectively. We apply our model to a real use case, a 1BDD protein trajectory, to assess the impact of lost frames on capturing the swing of one helix away from the other two helices in the protein. Future work includes the study of our modeling for a diverse set of MD trajectories and the study of the impact of more complex analytics on a diverse set of molecular systems.

Acknowledgments

This work is funded by NSF #1841758, #1741040, #1741057, and #1740990; and partly funded by DOE #DE-SC0012636, DE-AC02-05CH11231.MD, DE-AC05-00OR22725, and DE-AC02-05CH11231.

References

[1] J. R. Perilla, B. C. Goh, C. K. Cassidy, B. Liu, R. C. Bernardi, T. Rudack, H. Yu, Z. Wu, and K. Schulten, "Molecular dynamics simulations of large macromolecular complexes," *Current opinion in structural biology*, vol. 31, pp. 64–74, 2015.

[2] C. Docan, M. Parashar, and S. Klasky, "Dataspaces: an interaction and coordination framework for coupled simulation workflows," *Cluster Computing*, vol. 15, no. 2, pp. 163–181, 2012.

[3] D. L. Theobald, "Rapid calculation of rmsds using a quaternion-based characteristic polynomial," *Acta Crystallographica Section A: Foundations of Crystallography*, vol. 61, no. 4, pp. 478–480, 2005.

[4] V. Calandrini, E. Pellegrini, P. Calligari, K. Hinsen, and G. R. Kneller, "nmoldyn-interfacing spectroscopic experiments, molecular dynamics simulations and models for time correlation functions," *École thématique de la Société Française de la Neutronique*, vol. 12, pp. 201–232, 2011.

[5] B. Zagrovic, C. D. Snow, M. R. Shirts, and V. S. Pande, "Simulation of folding of a small alpha-helical protein in atomistic detail using worldwide-distributed computing," *Journal of molecular biology*, vol. 323, no. 5, pp. 927–937, 2002.

[6] W. Jiang, J. C. Phillips, L. Huang, M. Fajer, Y. Meng, J. C. Gumbart, Y. Luo, K. Schulten, and B. Roux, "Generalized scalable multiple copy algorithms for molecular dynamics simulations in namd," *Computer physics communications*, vol. 185, no. 3, pp. 908–916, 2014.

[7] J. C. Phillips, Y. Sun, N. Jain, E. J. Bohm, and L. V. Kalé, "Mapping to irregular torus topologies and other techniques for petascale biomolecular simulation," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 81–91.

[8] J. Vymetal and J. Vondrasek, "Critical assessment of current force fields. short peptide test case," *Journal of chemical theory and computation*, vol. 9, no. 1, pp. 441–451, 2012.

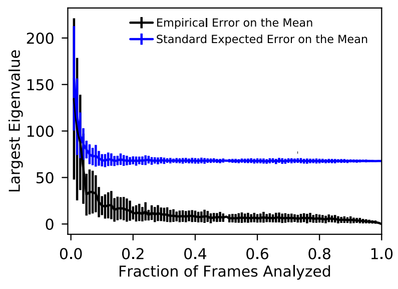
[9] A. T. Tzanov, M. A. Cuendet, and M. E. Tuckerman, "How accurately do current force fields predict experimental peptide conformations? an adiabatic free energy dynamics study," *The Journal of Physical Chemistry B*, vol. 118, no. 24, pp. 6539–6552, 2014.

[10] A. Kitao and N. Go, "Investigating protein dynamics in collective coordinate space," *Current opinion in structural biology*, vol. 9, no. 2, pp. 164–169, 1999.

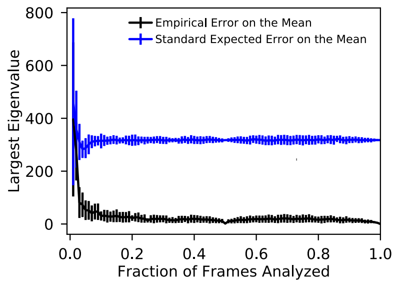
[11] T. Johnston, B. Zhang, A. Liwo, S. Crivelli, and M. Taufer, "In situ data analytics and indexing of protein trajectories," *Journal of computational chemistry*, vol. 38, no. 16, pp. 1419–1430, 2017.

[12] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with namd," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005.

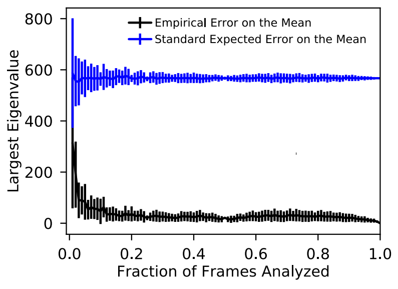
[13] G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, "Plumed 2: New feathers for an old bird," *Computer Physics Communications*, vol. 185, no. 2, pp. 604–613, 2014.



(a) Helices 1-2



(b) Helices 1-3



(c) Helices 2-3

Fig. 14: Trend in standard error of mean indicate that the uncertainty in largest eigenvalue increases when the fraction of frames analyzed is lower.

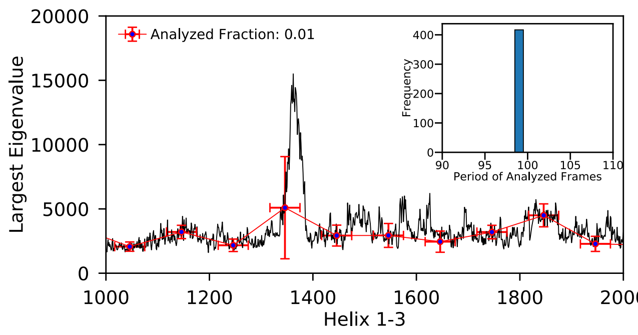


Fig. 15: The black lines show the original helix 1-3 largest eigenvalue data while the red line show the mean index of the 1% analyzed frames generated by the 2-step model over 1000 random trials where the phase of the periodic pattern is shifted. The horizontal error bars show the standard deviation in the analyzed frame index and the vertical error bars indicate the standard deviation in the largest eigenvalue. The inset shows the histogram of the periods between analyzed frames.

[14] H. Gouda, H. Torigoe, A. Saito, M. Sato, Y. Arata, and I. Shimada, "Three-dimensional solution structure of the b domain of staphylococcal protein a: comparisons of the solution and crystal structures," *Biochemistry*, vol. 31, no. 40, pp. 9665–9672, 1992.

[15] A. J. Bartels, *Zur Morphologie geophysikalischer Zeitfunktionen*, 1935.

[16] J. D. Chodera, "A simple method for automated equilibration detection in molecular simulations," *Journal of chemical theory and computation*, vol. 12, no. 4, pp. 1799–1805, 2016.

[17] G. Bayley and J. Hammersley, "The effective number of independent observations in an autocorrelated time series," *Supplement to the Journal of the Royal Statistical Society*, vol. 8, no. 2, pp. 184–197, 1946.

[18] A. Kumar, V. Grupcev, M. Berrada, J. C. Fogarty, Y.-C. Tu, X. Zhu, S. A. Pandit, and Y. Xia, "Dcms: A data analytics and management system for molecular simulation," *Journal of big data*, vol. 2, no. 1, p. 9, 2015.

[19] M. Feig, M. Abdullah, L. Johnsson, and B. M. Pettitt, "Large scale distributed data repository: design of a molecular dynamics trajectory database," *Future Generation Computer Systems*, vol. 16, no. 1, pp. 101–110, 1999.

[20] J. Shao, S. W. Tanner, N. Thompson, and T. E. Cheatham, "Clustering molecular dynamics trajectories: 1. characterizing the performance of different clustering algorithms," *Journal of chemical theory and computation*, vol. 3, no. 6, pp. 2312–2334, 2007.

[21] Y. Li and Z. Dong, "Effect of clustering algorithm on establishing markov state model for molecular dynamics simulations," *Journal of chemical information and modeling*, vol. 56, no. 6, pp. 1205–1215, 2016.

[22] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[23] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[24] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002, pp. 685–694.

[25] V. A. Tatsis, C. Tjortjis, and P. Tzirakis, "Evaluating data mining algorithms using molecular dynamics trajectories," *International journal of data mining and bioinformatics*, vol. 8, no. 2, pp. 169–187, 2013.

[26] F. Zheng, J. Zhang, and G. Grigoryan, "Tertiary structural propensities reveal fundamental sequence/structure relationships," *Structure*, vol. 23, no. 5, pp. 961–971, 2015.

[27] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "Cath–a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1109, 1997.

[28] A. Ramanathan, J. O. Yoo, and C. J. Langmead, "On-the-fly identification of conformational substates from molecular dynamics simulations," *Journal of Chemical Theory and Computation*, vol. 7, no. 3, pp. 778–789, 2011.

[29] C. Best and H.-C. Hege, "Visualizing and identifying conformational ensembles in molecular dynamics trajectories," *Computing in Science & Engineering*, vol. 4, no. 3, p. 68, 2002.

[30] J. L. Phillips, M. E. Colvin, and S. Newsam, "Validating clustering of molecular dynamics simulations using polymer models," *BMC bioinformatics*, vol. 12, no. 1, p. 445, 2011.

[31] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci *et al.*, "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 49.

[32] S. Lakshminarasimhan, D. A. Boyuka, S. V. Pendse, X. Zou, J. Jenkins, V. Vishwanath, M. E. Papka, and N. F. Samatova, "Scalable in situ scientific data encoding for analytical query processing," in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*. ACM, 2013, pp. 1–12.

[33] S. Pronk, I. Pouya, M. Lundborg, G. Rotskoff, B. Wesen, P. M. Kasson, and E. Lindahl, "Molecular simulation workflows as parallel algorithms: The execution engine of copernicus, a distributed high-performance computing platform," *Journal of chemical theory and computation*, vol. 11, no. 6, pp. 2600–2608, 2015.

[34] S. Doerr, M. Harvey, F. Noe, and G. De Fabritiis, "Htmd: high-throughput molecular dynamics for molecular discovery," *Journal of chemical theory and computation*, vol. 12, no. 4, pp. 1845–1852, 2016.