
Ian Kenney* Oliver Beckstein†

September 14, 2015

Analysis of molecular dynamics (MD) trajectories is becoming more and more challenging with simulation times routinely exceeding microseconds (with millions of frames) and increasing in size (with millions of particles). The MDAnalysis library (mdanalysis.org) is a versatile open source package that provides a foundation for analysing trajectories, with a particular emphasis on biomolecular simulations. In order to improve MDAnalysis for large simulations, we developed a set of non-trivial benchmarking systems of variable sizes. We generated a modular library of lipid vesicles with the coarse-grained Dry Martini force field, ranging in size from 10 nm to 30 nm radius. These vesicles can be combined in larger multi-vesicle systems in order to produce a range of benchmark systems with variable particle numbers. The systems will also allow us to address the question under which conditions vesicles fuse. We generated simulation systems with up to six vesicles and 3.5 million particles and performed initial simulations on local computing resources. We benchmarked the performance of MDAnalysis to load and process large systems. Our trajectory data set and benchmarking results will be useful to guide future development of MDAnalysis and its integration with the SPIDAL library.

1 Introduction

Molecular dynamics (MD) simulations have become an important computational tool to study biomolecular systems, in particular membrane proteins and membrane systems. Analysis of molecular dynamics (MD) trajectories is becoming more and more challenging with simulation times routinely exceeding microseconds (with millions of frames) and increasing in size (with millions of particles). The increase in data volume is driven by (1) improvements in hardware (such as GPUs) and algorithms (2) use of multi-copy enhanced sampling methods, and (3) new efficient representations of the physical interactions such as coarse-grained models, which allows simulations of larger systems and at longer time steps.

We have been leading the development of the open source MDAnalysis library (mdanalysis.org). It provides a Python-based tool kit to access trajectories generated by all major MD packages such as NAMD, Gromacs, CHARMM, Amber, LAMMPS, DL_Poly, Desmond, and
Table 1: Library of coarse-grained vesicles with dipalmitoylphosphatidylcholine (DPPC) lipids and two example benchmark systems with 3 and 6 large 30-nm vesicles.

<table>
<thead>
<tr>
<th>label</th>
<th>vesicles</th>
<th>radius (nm)</th>
<th>atoms</th>
<th>lipids</th>
</tr>
</thead>
<tbody>
<tr>
<td>R10.0</td>
<td>1</td>
<td>10</td>
<td>84192</td>
<td>7016</td>
</tr>
<tr>
<td>R12.5</td>
<td>1</td>
<td>12.5</td>
<td>122208</td>
<td>10184</td>
</tr>
<tr>
<td>R15.0</td>
<td>1</td>
<td>15</td>
<td>167352</td>
<td>13946</td>
</tr>
<tr>
<td>R17.5</td>
<td>1</td>
<td>17.5</td>
<td>219000</td>
<td>18250</td>
</tr>
<tr>
<td>R20.0</td>
<td>1</td>
<td>20</td>
<td>277728</td>
<td>23144</td>
</tr>
<tr>
<td>R22.5</td>
<td>1</td>
<td>22.5</td>
<td>343500</td>
<td>28625</td>
</tr>
<tr>
<td>R25.0</td>
<td>1</td>
<td>25</td>
<td>416208</td>
<td>34684</td>
</tr>
<tr>
<td>R27.5</td>
<td>1</td>
<td>27.5</td>
<td>496044</td>
<td>41337</td>
</tr>
<tr>
<td>R30.0</td>
<td>1</td>
<td>30</td>
<td>582984</td>
<td>48582</td>
</tr>
<tr>
<td>3R30.0</td>
<td>3</td>
<td>30</td>
<td>1748952</td>
<td>145746</td>
</tr>
<tr>
<td>6R30.0</td>
<td>6</td>
<td>30</td>
<td>3497904</td>
<td>291492</td>
</tr>
</tbody>
</table>

HOOMD$^{26}$; in addition, common formats such as the protein databank (PDB) format can be read and written. Although many time-critical parts of MDAnalysis are written in C or Cython, not all parts of the library are yet optimized to handle very large systems. Here we report on the initial development of a set of non-trivial benchmarking systems of variable sizes that are to be used to determine performance bottlenecks in MDAnalysis and pin-point areas on which future improvement efforts should be focused. Instead of generating pure solvent systems that can be easily scaled to any desired particle number we instead looked for less homogeneous and scientifically more interesting systems. We focused on the question how biomolecular membranes (lipid bilayers) fuse and form larger aggregates. In particular we are interested in the process by which multiple vesicles—spheroidal bilayer-enclosed structures (Fig. 1)—aggregate. Understanding the underlying physics is important for biological transport processes in the synapses$^{27}$ and the Golgi apparatus$^{28,29}$ but might also be of interest for the development of drug delivery vehicles$^{30}$.

In typical simulations, a large fraction (> 50%) of the simulated particles and interactions consists only of the solvent. Here we utilize the recently published implicit solvent coarse-grained Dry Martini force field$^{31}$ to avoid simulation of solvent and focus on the lipids alone. This approach enables us to simulate vesicles of realistic sizes and to include a large number of lipids so that we can test specific lipid analysis algorithms such as LeafletFinder$^{18}$ on large systems. Furthermore, by combining multiple vesicles of varying sizes we can generate inhomogeneous benchmark systems at arbitrary sizes.

2 Methods

2.1 Vesicle library and benchmark systems

We generated a library of lipid vesicles with the coarse-grained Dry Martini force field and the `martini_vesicle_builder.py` script$^{31}$, ranging in size from 10 nm to 30 nm radius (1). Each vesicle contains a realistic number of lipids and is represented with a coarse-grained representation. One half is omitted to show the interior, which is filled with solvent (not shown).
vesicle was energy minimized and simulated in the \( \text{NVT} \) ensemble for 100 ns in order to relax it after model building (see Section 2 for details on the MD simulations).

These vesicles can be combined in larger multi-vesicle systems in order to produce a range of benchmark systems with variable particle numbers and to investigate the question how vesicles fuse. As example systems, we generated simulation systems with 3 and 6 vesicles (each with radius 30 nm), for a total of 1.7 million and almost 3.5 million particles (1).

2.2 MD simulations

MD simulations were performed with Gromacs 5.0.5\(^2\). Simulation settings were chosen as recommended for the Dry Martini force field\(^3\). In particular, dynamics were simulated with Langevin dynamics with a time step of 20 fs and a friction coefficient of 4 ps\(^{-1}\). For runs that utilize GPUs in addition to CPUs, parameters were adjusted to be compatible with the specific requirements of Gromacs (namely, the Verlet neighbor search scheme). Simulations were run on local workstations equipped with 16 cores (2 Intel Xeon E5-2670 2.60 GHz and either a single NVIDIA GTX 980 or a dual GTX 690 GPU).

2.3 Benchmarking and Analysis

Analysis of simulations was carried out with Python scripts and Jupyter notebooks\(^4\) based on MDAnalysis 0.8.1\(^1\)\(^8\) together with the NumPy and SciPy\(^3\)\(^3\) libraries for numerics and matplotlib\(^3\)\(^4\) for plotting. Benchmarking was performed with Python scripts on a local workstation (see above). Simulation performance was measured in "ns/d", i.e., simulated time per wall-clock time, where higher is better. For I/O and frame-based analysis operations we measured the wall-clock time required to perform the same operation in many (typically 40) independent repeats and report mean and standard deviation.

3 Results and Discussion

Our goals were to generate inhomogeneous benchmarking simulations in a modular fashion that will allow us to look at systems of increasing size and to identify bottlenecks in the MDAnalysis library that hinder the processing of large systems. We solved the first problem by developing a vesicle library whose members can be combined into larger multi-vesicle systems. We then used the vesicle library to benchmark MDAnalysis trajectory loading and processing operations.

3.1 Vesicle library

Each single-vesicle system in Table 1 was simulated for 100 ns. Although this is much shorter than the recommended simulation time of 1–10 \( \mu\)s\(^3\), analysis of the radius of gyration of each vesicle indicates that these systems are already rather stable (Fig. 2). These preliminary results indicate that the library of vesicle structures is sufficiently equilibrated to be usable as building blocks for larger simulation systems.

For the construction of two initial multi-vesicle test systems, the largest (radius 30 nm) vesicle was chosen and replicated either three or six times (Fig. 3a) to arrive at systems with either 1.7 million or almost 3.5 million particles (Table 1). The systems were energy minimized and benchmarked on local workstation resources to run at \( \sim 35 \text{ ns/d} \) or \( \sim 15 \text{ ns/d} \) when using two GPUs and 16 cores together with 16 additional hyperthreaded cores (Fig. 3). For such large systems with so many lipids (150,000 and 300,000), the performance is rather high if one compares it to an all-atom bilayer system with explicit water and 256 lipids that runs on the same hardware at \(< 15 \text{ ns/d} \) (data not shown).
Figure 2: Radius of gyration $R_g$ of individual vesicles, collected from the first 50 ns of 100 ns MD simulations.

Figure 3: Multi-vesicle systems as benchmark simulations. The 6R30.0 system contains six vesicles with radius 30 nm, for a total of about 3.5 million particles. The 3R30.0 system contains three vesicles and about 1.7 million particles. Benchmarks in b and c were performed on a local workstation with 16 Intel Xeon E5-2670 2.60 GHz cores (which provide an additional 16 hyperthreaded cores) and one NVIDIA GTX 690 GPU (providing two GK104 GPU units). Performance (ns/d) is shown against number of OpenMP threads per GPU, for one (blue) and two (green) GPU units.

Benchmarks on larger machines with higher core numbers and more GPU accelerators will need to be performed in the future to reach a performance of about 250 ns/d so that microsecond-length simulations become feasible, which will be required to address scientific questions beyond benchmarking. Nevertheless, these two simulations demonstrate the feasibility of using the vesicle library to construct larger simulation systems and further work will explore other packing arrangements of vesicles of different sizes. Such longer simulations will lead to vesicle fusion that can then be quantitatively studied as a function of thermodynamic parameters, lipid composition, and vesicle sizes.

3.2 MDAnalysis trajectory processing performance

We benchmarked the performance of MDAnalysis to load and process large systems by loading the simulations in the vesicle library. The first step in any MDAnalysis script is to load a topology file (which contains a list of particles and possibly additional static data such as bonds or partial charges) and a trajectory file. The trajectory contains the coordinates, which change for each time
step. In MDAnalysis, the Universe object ties the topology and the trajectory together and part of the process of instantiating Universe(topology, trajectory) is to parse these files. MDAnalysis contains custom parsers for a wide range of topology and trajectory files. The benchmark of loading a Universe with different topology files (and typically a Gromacs XTC trajectory) showed that the load time increases linearly with the input size (Fig. 4a). Loading from a Gromacs binary run input file (TPR) together with the XTC appears to be fastest (<1 s up to 5 s) for all problem sizes up to the investigated maximum of about 600,000 particles. Although not immediately relevant for trajectory processing, loading only from a PDB file (which contains topology information and coordinates) was much worse (5 s up to 30 s) than any other method. This benchmark indicates the PDB reader as a prime candidate for further performance improvements and provides a baseline for increasing the reading speed of the TPR reader further to facilitate rapid reading of larger systems.

As a first simple case for benchmarking analysis we measured the time to calculate the radius of gyration

$$R_g(t) = \sqrt{\frac{1}{M} \sum_{i=1}^{N} m_i (\mathbf{r}_i(t) - \mathbf{R}(t))^2},$$

(1)

where $M$ is the total mass, $\mathbf{R}(t)$ the center of mass, and the sum runs over $N$ particles with mass $m_i$ and position $\mathbf{r}_i(t)$ each at time frame $t$. The calculation of $R_g$ included all particles in the simulation. The average wall-clock time to analyze a single trajectory frame increased linearly with the number of particles, starting with about 10 ms for $\sim 100,000$ particles and increasing to $\sim 93$ ms for 600,000 particles. For long trajectories (order of 1 million frames), analysis times of 100 ms per frame are not acceptable because serial analysis would take about 28 h. Thus we need to improve the per-frame performance. We need to better understand if the performance is limited by loading the trajectory data from disk or by the speed of calculating Eq. 1. We will also need to investigate approaches to parallelize the workload when possible, e.g., by a simple map-reduce algorithm over blocks of the trajectory.
4 Conclusions

Our trajectory data set and benchmarking results will be useful to guide future development of MDAnalysis and its integration with the SPIDAL library.

Acknowledgements Funding was provided by the National Science Foundation for a REU supplement to award ACI-1443054.

References

23. Plimpton S, 1995 Fast parallel algorithms for short-
range molecular dynamics. J. Comp. Phys. 117 1–19.
24. Todorov I T, Smith W, Trachenko K and Dove M T,
2006 DL-POLY_3: new dimensions in molecular dy-
35. Baoukina S and Tieleman D P, 2010 Direct simulation of protein-mediated vesicle fusion: lung surfac-
tant protein b. Biophys J 99 2134–42.