A Benchmark Suite for Evaluating Performance of Map-Reduce Job Implemented with Dask in MDAnalysis Library

Mahzad Khoshlessan Oliver Beckstein

March 13, 2017

MDAnalysis (`http://mdanalysis.org`) is a Python library that provides users with access to raw simulation data that allows structural and temporal analysis of molecular dynamics (MD) trajectories generated by all major MD simulation packages. The size of these trajectories is growing as the simulation times is being extended from micro-seconds to mili-seconds. Therefore, there is a need for high performance computing (HPC) approaches to increase the throughput. MDAnalysis does not yet provide a standard interface for parallel analysis; instead, various existing parallel libraries are currently used to parallelize MDAnalysis-based code. However, a standardized benchmark suite that focuses on helping users evaluate the performance of the MDAnalysis library is not available in the current community. Present study aims to identify possible approaches for bringing HPC into MDAnalysis. In this paper, we present a benchmark suite that can be used to evaluate performance for parallel map-reduce type analysis along with the Dask library for task-graph based distributed computing (`http://dask.pydata.org/`). A range of commonly used MD file formats (CHARMM/NAMD DCD, Gromacs XTC, Amber NetCDF) and different trajectory sizes are benchmarked on different high performance computing (HPC) resources. Different storages like solid state drives (SSDs), hard disk drives (HDDs) and Lustre file system are also tested to examine effect of storage location on the performance. The benchmarks are performed both on a single node and across multiple nodes using multiprocessing and distributed schedulers in Dask library. Overall, our results show strong dependency to hardware and file formats. On a single node we found multiprocessing scheduler to provide slightly better scaling than distributed scheduler on multiple nodes. According to our results, although Map-Reduce job is pleasantly parallel and all processors have the same amount of work to do, some of the processes are much slower than the others in some tests. We hypothesize that

weak performance gains from distributed scheduler is likely due to contention on the network that may slow down individual tasks and lead to overall waits and poor load balancing. All in all, obtaining good parallel performance with a Map-Reduce approach for trajectory analysis is strongly dependent on efficient transfer of trajectory data to memory. Present study provides guidelines for how the choice of trajectory format along with the hardware can lead to good performance. At the end, we also suggest the promising approaches that can be done to get the best possible performance out of MDAnalysis library.