Design and Performance Analysis of a Distributed HPC Molecular Dynamics Code on Distributed Resources

Promita Chakraborty, Shantenu Jha
Dept. of Computer Science & CCT, Louisiana State University

Molecular Dynamics (MD) is an important step towards understanding the behavior of biological systems. Even though, the state-of-the-art computational resource and techniques have made it feasible to do large-scale MD simulations, MD remains a very compute-intensive job, especially for large realistic systems. Thus there exists interest in developing and understanding the performance of MD code on multiple distributed HPC resources. To realize this goal, we have developed a preliminary parallel and distributed MD code and benchmarked it on the LONI (Louisiana Optical Network Interface) grid, which uses dedicated light-path networks to connect supercomputers across Louisiana. This work is motivated by an attempt to utilize new infrastructure and to devise new programming strategies for MD simulations, with a focus on distributing the workload across various machines while retaining the advantages of parallelization within a single machine. Current practice is to distribute the job amongst various machines only when the resource requirement is more than the capacity of a single machine. In our work, we divided the job into several workloads, even if a single machine was capable of handling it. We tested our developed code on upto three distributed resources of LONI, namely Bluedawg, Zeke and Ducky. These are IBM P5 clusters with 114 nodes per cluster. Based on performance data, we show that without any serious optimization, the performance degradation as defined by total CPU-hrs on multiple machines is about 10-20% of the performance over a single machine, which has useful consequences when time to finish is critical. Based on this analysis, the users of grid resources can pick their choice between two different strategies: (1) optimize the CPU-hours used, and live with the huge wait time involved, (2) or use an extra 10-20% CPU-hours and optimize the overall job throughput. Our analysis has the potential to be extended to parallel codes other than MD. We believe that this study can lead to a better job distribution and resource allocation to optimize throughput.