Scalable Molecular Dynamics with NAMD on the Summit System

B. Acun, D. J. Hardy, L. V. Kale, K. Li, J. C. Phillips, J. E. Stone

NAMD (NAnoscale Molecular Dynamics) is a parallel molecular dynamics application that has been used to make breakthroughs in understanding the structure and dynamics of large biomolecular complexes, such as viruses like HIV and various types of influenza. State-of-the-art biomolecular simulations often require integration of billions of timesteps, computing all interatomic forces for each femtosecond timestep. Molecular dynamics simulation of large biomolecular systems and long-timescale biological phenomena requires tremendous computing power. NAMD harnesses the power of thousands of heterogeneous processors to meet this demand. In this paper, we present algorithm improvements and performance optimizations that enable NAMD to achieve high performance on the IBM Newell platform (with POWER9 processors and NVIDIA Volta V100 GPUs) which underpins the Oak Ridge National Laboratory's Summit and Lawrence Livermore National Laboratory's Sierra supercomputers. The Top-500 supercomputers June 2018 list shows Summit at the number one spot with 187 Petaflop/s peak performance and Sierra third with 119 Petaflop/s. Optimizations for NAMD on Summit include: data layout changes for GPU acceleration and CPU vectorization, improving GPU offload efficiency, increasing performance with PAMI support in Charm++, improving efficiency of FFT calculations, improving load balancing, enabling better CPU vectorization and cache performance, and providing an alternative thermostat through stochastic velocity rescaling. We also present performance scaling results on early Newell systems.

1. Introduction

Structural biology simulation presents two challenges. The first, driven by the revolution in cryo-electron microscopy and other single-molecule imaging methods, is the increasing scale of the biomolecular aggregates that can be resolved in sufficient atomic detail to enable simulation. For a fixed simulation rate, computational requirements scale with the number of atoms simulated. Petascale machines such as NCSA Blue Waters and Oak Ridge Titan have been used for simulations of up to 200 million atoms [1], including studies of the HIV virus capsid, the photosynthetic chromatophore, and the influenza viral coat. Simulations running on Summit will encompass up to two billion atoms while also achieving faster simulation rates than Titan.

The second challenge is the long timescales of many biomolecular processes, many requiring structural rearrangements or other rare events that would take milliseconds or longer to observe in a single molecular dynamics trajectory. To address this challenge, Summit enables the simulation of thousands of replicas of a smaller system, and NAMD is equipped with various methodologies for exchanging temperatures or a variety of steering biases among replicas to statistically sample very long time scales [2]. Depending on the size of the system, NAMD can make effective use of Summit by simulating one replica per Volta GPU (six replicas per node), two replicas per node (one replica for every three Volta GPUs), or one replica per node (one replica for every six Volta GPUs). Due to the lack of inter-node communication, each replica is able to run as a single NAMD process with a higher simulation rate than large, multi-node simulations. Multi-node replicas are also supported if required.

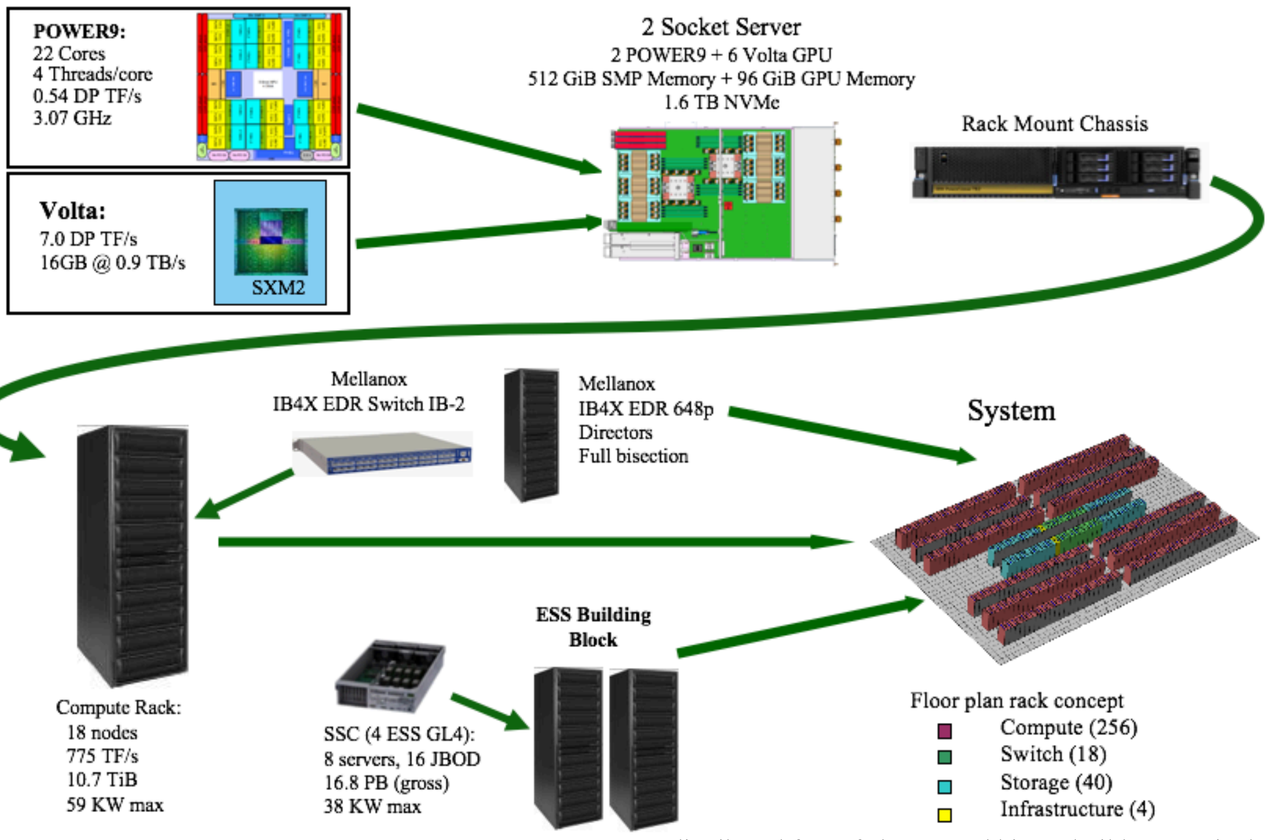
2. Background

Summit Architecture

Summit is a supercomputer with over 200 Petaflops of double precision theoretical performance [3]. Summit is composed of 4,600 powerful IBM AC922 compute nodes, each containing two POWER9 CPUs and six Nvidia Volta V100 GPUs. The POWER9 CPUs have 22 cores running at 3.07 GHz. In operation, Summit reserves one core per CPU socket for operating system use to reduce system noise. The remaining 42 CPU cores on each node are made available for applications. Each CPU core supports up to four hardware threads (SMT4), with pairs of threads sharing L2 and L3 cache regions [4]. The six NVIDIA Tesla V100 GPUs in each node provide a theoretical double-precision arithmetic capability of approximately 40 TeraFLOPS. Dual NVLink 2.0 connections between CPUs and GPUs provides a 25GB/s transfer rate in each direction on each NVLink, yielding an aggregate bidirectional bandwidth of 100GB/s.

Summit has a data network with a non-blocking fat tree topology with a node injection bandwidth of 23 GB/s. Compute nodes have access to high performance storage systems through GPFS, and each node also contains an on-board 1.6TB of NVMe flash storage for use as a so-called burst buffer, to facilitate node-local storage of transient data, e.g., for checkpoint/restart images, and in-situ visualization and analysis.

Figure 1: Summit Architecture.



NAMD

NAMD, recipient of a 2002 Gordon Bell Award and a 2012 Sidney Fernbach Award, is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems [5]. NAMD scales to hundreds of cores for typical simulations and beyond 500,000 cores for the largest simulations. In 2007, NAMD was the first full-featured molecular dynamics application to support GPU acceleration using CUDA [6], [7].

NAMD offers a wide range of traditional molecular dynamics simulation methodologies, including fast methods for long-range electrostatics, with support for the CHARMM, AMBER, and GROMOS force fields. NAMD also provides many advanced features, including support for the Drude polarizable force field in CHARMM, a variety of free energy perturbation methods (with GPU compatibility in all conformational free energy methods and ongoing development for GPU support in alchemical free energy methods), a variety of replica exchange molecular dynamics methodologies, and the Colvars collective variables package. The Tcl scripting interface offered by NAMD enables customization opportunities for extending the underlying methods. The source code and binary builds are distributed free of charge, and binary builds are actively maintained at the NSF supercomputing centers.

Charm++

NAMD is built upon the Charm++ parallel programming framework, which is a C++ based model with data-driven objects [8]. Charm++ provides NAMD performance and portability across many architectures.

The Charm++ adaptive runtime system (RTS) has three main properties: Over-decomposition, asynchronous message driven execution, and migratability. Over-decomposition means the application data is divided into many objects, typically more than the number of processors. In the case of NAMD, the atoms are specially decomposed into patch objects together with compute objects that are used to calculate the forces between the patches. The Charm++ runtime system controls the mapping of these objects to the processors.

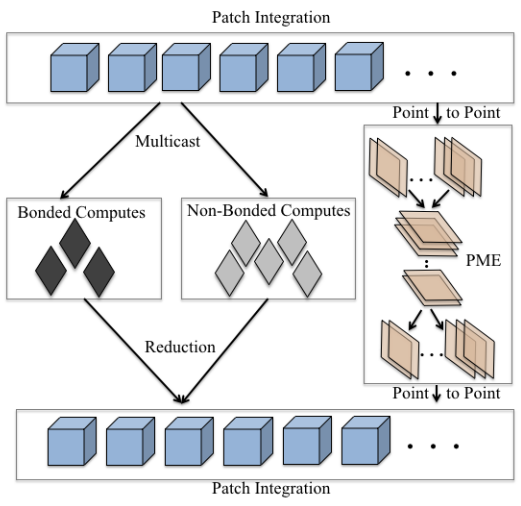
Asynchronous message driven execution means the program execution happens through message calls on objects that may be located on local or remote processors. Asynchronous execution can help reduce the communication overhead that can degrade the performance of applications. The sender and receiver does not block execution after sending a message or when waiting for a message. It can execute other messages available in the queue, therefore overlapping the communication with computation. The messages can also be assigned a priority level based on the application. For NAMD, PME messages that drive the FFT computation are more critical than the non-bonded compute messages coming from the patches. Therefore, PME messages are given higher priority.

The third property, migratability, is the ability of the runtime system to move objects from processor to processor. The runtime can redistribute the objects dynamically to achieve load balance.

NAMD not only benefits from these attributes of Charm++ but also the portable high-performance communication layer of Charm++. Section 3, gives more details on the PAMI-based communication layer that is used on Summit.

Parallel structure of NAMD: NAMD was one of the early pioneers of the idea of hybrid spatial-and-force decomposition [9]: the atoms are partitioned into cubes, typically of size slightly larger than the cutoff distance. Separately, a set of force computation objects are created, one for each pair of cubes that are within the cutoff distance from each other. These objects are assigned to processors using the Charm++ load-balancing framework. In addition, there are separate sets of objects that implement bonded force calculations, and an FFT-based particle-mesh Ewald (PME) algorithm. This decomposition is depicted in Figure 2. CUDA force calculation is done in single precision. Bonded forces are accumulated as 64-bit fixed point, while nonbonded forces are accumulated in single precision.

Figure 2: Parallel decomposition in NAMD (Figure adapted from [14]).



3. Performance Optimizations

In this section, we describe the performance optimizations implemented on NAMD and Charm++ in order to scale NAMD efficiently on the Summit architecture. These techniques include: data layout changes for GPU acceleration and CPU vectorization, improving GPU offload efficiency, PAMI-based communication layer, optimizing PME/FFT, load balancing, vectorization of NAMD routines on POWER9, and stochastic velocity rescaling. All of the techniques are included in the NAMD 2.13 release.

Data Layout Changes for GPU Acceleration and CPU Vectorization

GPUs and state-of-the-art CPUs such as POWER9 incorporate SIMD architecture concepts and/or vector instructions to improve arithmetic throughput at a low cost in terms of microprocessor logic and die area. Vector instructions enable CPUs to process multiple data items concurrently, performing the same instruction on each data item. GPUs rely very heavily on SIMD architecture, collectively employing thousands of ALUs that are organized into large groups called warps or workgroups, roughly similar to a CPU vector but typically much larger. To exploit SIMD hardware architectures efficiently, data items to be processed must be organized in consecutive memory locations in a so-called structure-of-arrays (SOA) layout. NAMD has been developed in C++ following conventional best practices. This has led to object-oriented data structures and memory layouts that are easy to work with and that lend themselves to customization and extension, however these same data structures and memory layouts inhibit or reduce the efficacy of CPU vector instructions. The prevalent memory layout pattern that has resulted from these C++ objects is what is known as array-of-structures (AOS) layout, which is ill-suited to vectorization because consecutive memory storage locations refer to different object member variables. By redesigning performance-critical NAMD data structures for SOA layout, advanced C++ compilers such as IBM XL C/C++ can often vectorize loops automatically, and host-side data structures are better prepared for replication onto or access from GPU accelerators.

Improving GPU offload efficiency

NAMD was one of the very earliest production HPC codes to adopt GPU computing techniques. Prior to being accelerated by GPUs, a rough breakdown of the amount of computational effort spent in different parts of the calculation is shown in Table 1.

Table 1: NAMD breakdown of the computational effort on CPU

Non-bonded forces, short-range cutoff	90%
Long-range electrostatics, gridded (e.g. PME)	5%
Bonded forces (bonds, angles, etc.)	2%
Correction for excluded interactions	2%
Integration, constraints, thermostat, barostat	1%

The most computation intensive and time consuming parts of NAMD were the first to be accelerated by GPUs. Over the course of the last several years, all force calculations have been implemented using CUDA on GPU-accelerated platforms [6], [7], [10], with bonded force calculations being the most recently implemented, and which appear in NAMD 2.13 for use on Summit.

As the CUDA programming model has evolved to allow greater flexibility and sophistication for collective operations and synchronizations among groupings of threads, the underlying GPU hardware has also evolved, requiring algorithms that use so-called warp-synchronous programming techniques be adapted to new CUDA APIs. To make previously developed NAMD force kernels work properly on the Tesla V100 GPUs on Summit, we have adapted them to a new set of warp-level intrinsic APIs that additionally require the GPU active thread lane mask as an input parameter to ensure the correctness of warp-synchronous calculations.

When compared with previous HPC systems, we have found that the balance between CPU and GPU computation power has shifted with the new Tesla V100 GPU, and particularly in the dense six-GPU node configuration on Summit. The overall NAMD performance is heavily bounded by CPU performance while the GPU utilization is below 15%. From profiling, we can see that the CPU time between two consecutive non-bonded force GPU kernels are significantly longer than the execution of the force kernel itself. This excess CPU activity is associated with what was originally about 1% of the overall work, responsible for the integration of atomic coordinates, rigid bond constraints, thermostat, and barostat controls. We are in the process of overcoming this CPU calculation bottleneck by offloading these additional work phases to the GPUs.

We have added a new SOA layout for NAMD's patch data, while keeping the original AOS layout for communication between nodes. The new SOA data structure uses a single contiguous buffer to store all atomic coordinates and integration parameters, with arrays padded to 32 elements for coalesced GPU memory access.

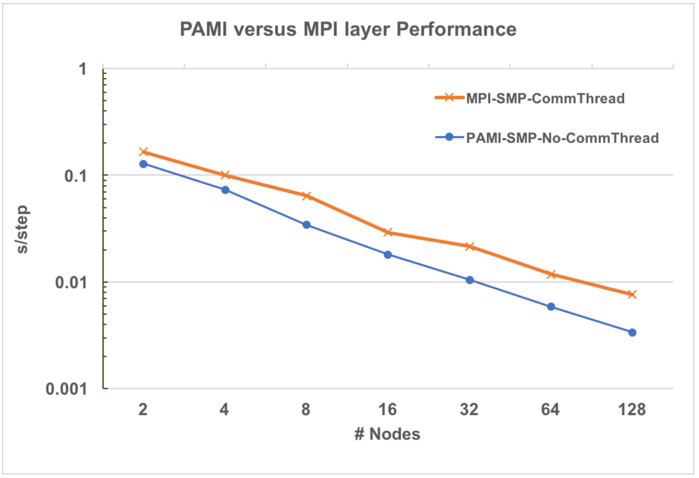
The prototype GPU integrator introduces minimal change to the existing code. During the GPU integration, each patch, running in its own user-level thread, copies its SOA data array to the corresponding GPU global memory before any GPU kernel launch. The SOA data array is copied back to the CPU memory before scheduling the force calculation or starting the load balancing and output phase. During the reduction phase, warp intrinsic operations are used for the reduction of the patch on a GPU before sending out via Charm++ for global reduction.

Currently the drawback for this prototype is that CUDA kernel launch overhead and memory copy overhead is still dominating the integration computation because four memory transfers and two kernel launches are performed at every step for every patch. To further reduce the launch overheads, we are working on a patch aggregation approach which will pack the entire SOA buffer for all patches on a process before copying to GPU and kernel launch.

PAMI

IBM PAMI (Parallel Active Messaging Interface) is a low-level communication library that is provided on Summit [11]**.** Charm++ implements high performance communication using vendor-provided hardware-native communication APIs, unlike many other PGAS languages. These vendor-specific hardware-native communication layers usually outperform the MPI-based communication layer of Charm++. Charm++ initially implemented a PAMI-based communication layer for Blue Gene Systems [12]. The same implementation is still being used on Summit with minor enhancements. The performance plot below shows the performance benefit (reduction in time per step) associated with the use of the Charm++ PAMI communication layer rather than MPI. On Summit, the performance gain provided by the Charm++ PAMI layer is greatest for large node counts.

Figure 3: PAMI-SMP versus MPI-SMP performance.



Optimizing PME/FFT

Long-range electrostatic forces often drive conformational changes in biomolecular systems and so must be accurately represented. The particle-mesh Ewald (PME) method is used by NAMD to efficiently approximate the electrostatic interactions between all pairs of atoms, the number of which grows quadratically with the size of the system, as well as their infinite images in the periodic simulation cell. PME makes use of a fast approximation method important for all but the smallest system sizes. The challenge with PME for parallel scalability is that it requires 3D FFTs be calculated, which entails a many-to-many communication pattern between processors. Although a 3D FFT decomposes into a collection of 1D FFTs across each coordinate axis, the FFT itself is challenging to parallelize effectively, since there is very little calculation required relative to the communication involved, so scales poorly, particularly over multiple GPUs.

NAMD takes two approaches to optimizing the FFT-based PME calculation. For system sizes that are small enough to fit within a single node, the entire PME/FFT calculation is offloaded to a single GPU, making use of the cuFFT library provided by CUDA. This approach is particularly effective when employing replica exchange molecular dynamics for effectively using the power of Summit to run thousands of simultaneous simulations. For larger system sizes that span multiple nodes, NAMD employs PME every three steps and reduces the FFT bandwidth for the gridded part of the calculation by doubling the standard grid spacing, thereby reducing the entire 3D grid size by a factor of 8, while increasing the order of interpolation from 4th order to 8th order to maintain the same accuracy. This re-weighting of the work allows NAMD to benefit from offloading to the GPU the localized grid operations involving the charge spreading from atoms to grid points and the interpolation of grid potentials to atomic forces.

Load Balancing

During the investigation of NAMD performance, we found that parts of the CPU workload are not well balanced across different PEs. In the bonded force calculation routine, CPU threads copy data from individual buffers, corresponding to each bonded force type, into a single contiguous memory. This ensures that the CPU-to-GPU data transfers achieve higher bandwidth.

In SMP execution mode, the Charm++ CkLoop function is used to dynamically parallelize the copy of different types of bonds using multiple PEs. However, the "exclusion" bonds (which correct for bonded atoms that are intended to be excluded from non-bonded interaction) are always processed by only one PE, due to a serial algorithm for separating modified and non-modified types of van der Waals interaction. A given simulation might have many more exclusion bonds than any other type, causing the PE that is performing the copying of exclusion bonds to become a bottleneck. To overcome this issue, we parallelized the separation of modified and non-modified exclusion bonds and used CkLoop to copy these bonds into a single buffer in parallel. After exclusion bonds are done, the other five types of bonds can be well balanced during the copy. As a result, the overall run time is improved by about 1% after this balancing.

Vectorization of NAMD routines on POWER9

As described above, one of the key requirements to take advantage of CPU vectorization is that performance-critical data structures must be adapted to an SOA layout so that data associated with independent work units to be vectorized are stored contiguously in memory, to facilitate loads and stores to and from CPU vector registers. We expect to adapt all of the remaining performance-critical loops for execution entirely on the GPU, but implementations of new simulation methods, particularly those developed by external researchers, will likely always be implemented on the CPU initially. It is therefore desirable that such methods are able to benefit from CPU vectorization so that they do not negatively impact overall NAMD performance. As a particularly surprising example, early detailed benchmarking of NAMD performance on Summit showed that among several of the algorithms that had not yet been migrated to the GPU, random number generators that are associated with the Langevin thermostat were consuming a significant fraction (15%) of the CPU runtime, which was already limiting the efficacy of GPU acceleration and overall NAMD performance. The Gaussian random number distribution required by the Langevin thermostat algorithm involves costly floating point arithmetic that is aptly suited to vectorization.

Stochastic Velocity Rescaling

Significant performance gains are sometimes available through more efficient algorithms. The Langevin thermostat used for simulating at a desired temperature requires for each time step generating up to 3N Gaussian distributed random numbers (up to three per atom) and an additional calculation of the rigid bond constraints, both of which are substantial fractions of the overall cost of performing integration on the CPU. An alternative approach, the stochastic velocity rescaling thermostat of Bussi, Donadio, and Parrinello [13], gives NAMD as much as a 20% improvement in performance over the Langevin thermostat. Stochastic velocity rescaling is a relatively simple correction to the classic Berendsen thermostat to achieve true sampling of the canonical distribution (as does Langevin), but requires as few as two Gaussian distributed random numbers be calculated every twenty time steps. Furthermore, the rescaling preserves holonomic constraints so that no additional calculation of the rigid bond constraints are required. Unlike the Langevin thermostat, which requires no additional communication, the stochastic velocity rescaling thermostat does require a reduction operation to determine the temperature of the system followed by a broadcast of the new scaling coefficient, the calculation of which requires the two Gaussian distributed random numbers. However, the newly introduced communication is required no more frequently than every twenty time steps and can easily be piggybacked on other communication already being performed by NAMD.

4. Performance Results

In this section, we show preliminary results from benchmarking NAMD on Summit. Note that at the time these results were collected, Summit did not have final versions of its software stack and was still undergoing development and testing.

In order to benchmark NAMD performance for large simulations, we use freely distributable synthetic benchmarks by replicating the 1.06M atom STMV benchmark (satellite tobacco mosaic virus - the first full virus simulation performed with NAMD in 2006). Two benchmarks we show in this section were formed by tiling the 216.832 A cubic cell: a 5x2x2 system of 21 million atoms and a 7x6x5 system of 224 million atoms. The benchmarks use rigid water and bonds to hydrogens to enable a 2 fs time step and use a 12 A cutoff distance with PME every three steps using a 2 A grid and 8th order interpolation.

Pressure control with a relaxed reduction-broadcast barrier is enabled to avoid global synchronization [14].

Figure 4 shows strong scaling of NAMD on the quarter of the Summit system. GPUs on Summit provides about 8x performance compared to CPU only runs. The 21M atom simulation achieves about 128 nanoseconds per day, and the 224M atom simulation achieves close to 32 nanoseconds per day performance. In these early runs on Summit, SMT2 and SMT4 have not shown any performance benefit, therefore we use SMT1. Particular sections of the code that might benefit from SMT need to be studied in detail on why SMT has not provided better performance. Figure 5 shows that stochastic velocity rescaling improves performance about 10% compared to Langevin damping as discussed in the previous section. Average performance is further improved over original results by using a special "bsub" job submission flag to isolate the GPFS file system daemon from NAMD CPU threads by pinning it to a CPU core reserved for use by system software and the operating system itself. Trajectory output is written only at the end of the simulation once, therefore it is not included in the performance results.

Figure 4: NAMD strong scaling performance on Summit using up to 1K nodes. The upper and lower lines of each line type indicate results for the 21M atom and 224M atom simulations, respectively.

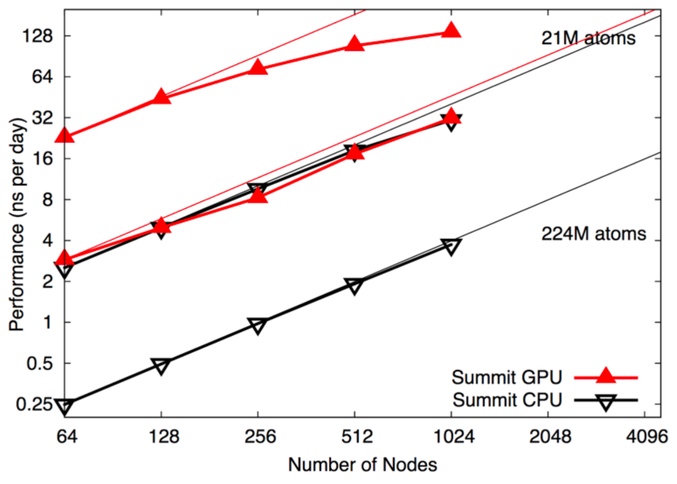
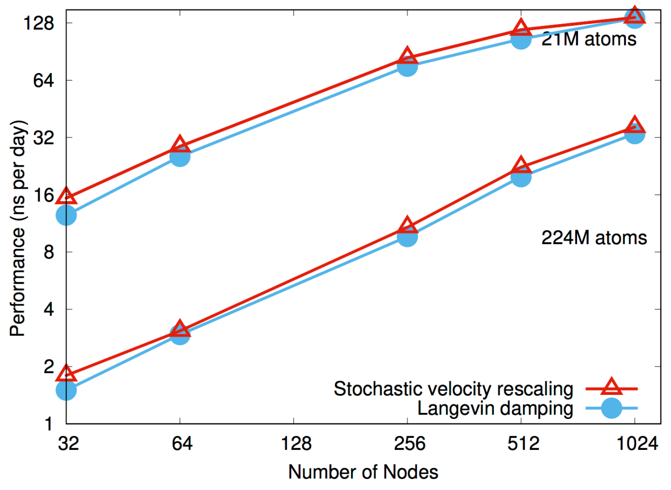


Figure 5: NAMD strong scaling results comparing stochastic velocity rescaling to Langevin damping.



5. Challenges and Ongoing Work

The single most important performance enhancement is to offload the integrator and related routines to the GPU. The Volta generation of GPU is now so fast that NAMD is held back by any computational work remaining on the CPU that scales with the number of atoms, effectively a modern-day version of Amdahl's Law. Depending on the simulation parameters, the floating point operations required for the CPU-based integrator can be less than 1% of the total computational work. Furthermore, NAMD patches were not implemented to be migratable units of work and the force calculation performed on GPUs requires little enough time as to render the measurement-based Charm++ load balancing ineffective. Offloading the integrator will help to restore the expected balance of computational work. Adoption of the fundamental NAMD atomic data structures to SOA form will need to proceed in pace with the integrator GPU offloading.

Although much of the integrator work is exceedingly data parallel, there are other parts of the integrator algorithms that are more difficult to implement on the GPU, in particular the rigid bond constraint calculation and the pressure reduction accumulations over so-called hydrogen groups. The rigid bond constraint implementation in NAMD, using successive bond relaxation, does not expose sufficient data parallelism desired for GPU implementation. Alternative algorithms that provide better data parallelism, such as matrix-SHAKE [14] and P-LINCS [15], are currently under investigation.

More broadly, NAMD, and biomolecular simulation in general, faces substantial challenges arising from the direction supercomputer architectures are going and strong-scaling needs of biophysicists. Each simulation time step is one or two femtoseconds, and so billions of steps are needed to simulate trajectories of microseconds duration. So, it is clear that we need to simulate hundreds of nanoseconds per day. Assuming a femtosecond timestep, 100 ns/day requires a timestep to be completed in less than a millisecond. With the planned advances in NAMD, this will be achieved for single-node runs easily. But for larger molecular systems, one has to use multiple nodes. Typical simulations of interest range from 100,000 to 10-million atom systems, with a few heroic simulations over hundred million atoms. Indeed, in the past, NAMD has demonstrated that it can complete a timestep in less than 250 microseconds on the BlueGene/Q machine simulating 92,000 atoms using 16,000 cores. This machine had slow low-power cores coupled with relatively fast network. However, as individual compute nodes get fatter, often without corresponding increase in across-node bandwidth, with relatively high kernel startup overheads, it will be challenging to meet and exceed these strong scaling targets. In terms of Summit itself, one can restate the challenge concretely: to obtain good scaling for one million atoms simulated on a few hundred nodes. One of the key techniques for achieving the strong scaling in the past was adaptive communication-computation overlap, which depended on firing small force calculation computations, taking tens of microseconds, as soon as their data was available. It will require a combination of provisioning of higher bandwidth, low-overhead communication of short messages, and ability to fire short kernels on subsets of GPGPU cores under program control and with low overhead, along with significant software experimentation and modification, to break through these strong scaling challenges in the future.

Fault tolerance is another challenge in large-scale systems. Charm++ provides various fault tolerance strategies to handle hard and soft errors: checkpoint/restart, message logging, proactive evacuation and targeted protection for silent data corruption [14]. Nevertheless NAMD uses its own periodic disk-based checkpointing of simulation state, which allows a simulation to be continued after faults. Particularly when the job schedulers start allowing applications to recover from failures and continue running, NAMD would benefit from Charm++'s fault tolerance strategies.

6. Conclusion

NAMD is one of the most scalable and impactful molecular dynamics applications. NAMD is supported in many different architectures and major supercomputers in the U.S., including very recently on Summit. As one of the ORNL's Center for Accelerated Application Readiness (CAAR) projects, performance optimization of NAMD on Summit's CPU-GPU architecture continues. In this paper, we have shown some of the optimizations implemented and preliminary results from scaling NAMD on a quarter of the full Summit system, and we have also identified immediate challenges.

7. Acknowledgment

First and foremost, we acknowledge the recent passing of two members of the NAMD team: Prof. Klaus Schulten established the Theoretical and Computational Biophysics Group (TCBG) and led the collaborative NIH Center for Macromolecular Modeling and Bioinformatics, the home of NAMD, from 1992 until his passing in late 2016; and Antti-Pekka Hynninen contributed to GPU kernel development in NAMD through the IBM/NVIDIA Center of Excellence program at Oak Ridge National Laboratory and then at NVIDIA until his passing in 2017. We thank Sameer Kumar for his help in optimizing NAMD on IBM systems. We thank Brian Radak of the TCBG for his implementation of stochastic velocity rescaling. Over the years, many other students and staff of the Parallel Programming Laboratory and the TCBG at the University of Illinois have contributed to NAMD software to make this project a success.

This work was supported by the National Institutes of Health grant NIH P41-GM104601 "Center for Macromolecular Modeling and Bioinformatics." This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We thank Tjerk Straatsma for his assistance as the point of contact for our Center for Accelerated Application Readiness project.

8. References

1. 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.," *SC ... conference proceedings. SC (Conference: Supercomputing)*, vol. 2014, pp. 81–91, 2012.
2. W. Jiang *et al.*, "Generalized Scalable Multiple Copy Algorithms for Molecular Dynamics Simulations in NAMD.," *Computer physics communications*, vol. 185, no. 3, pp. 908–916.
3. "Summit," *https: //www.olcf.ornl.gov/olcf-resources/compute-systems/summit/*. 2016.
4. S. K. Sadasivam, B. W. Thompto, R. Kalla, and W. J. Starke, "IBM Power9 Processor Architecture," *IEEE Micro*, vol. 37, no. 2.
5. J. C. Phillips *et al.*, "Scalable molecular dynamics with NAMD.," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1781–1802.
6. J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten, "Accelerating molecular modeling applications with graphics processors," *J. Comput. Chem.*, vol. 28, no. 16, pp. 2618–2640.
7. J. C. Phillips, J. E. Stone, and K. Schulten, "Adapting a Message-Driven Parallel Application to GPU-Accelerated Clusters," *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. Austin, Texas, 01-Nov-2008.
8. B. Acun *et al.*, "Parallel Programming with Migratable Objects: Charm++ in Practice," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014.
9. L. Kalé *et al.*, "NAMD2: Greater Scalability for Parallel Molecular Dynamics," *Journal of Computational Physics*, vol. 151, pp. 283–312, 1996.
10. J. E. Stone, A.-P. Hynninen, J. C. Phillips, and K. Schulten, "Early

Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms.," *High performance computing : 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings. ISC High Performance (Conference) (31st : 2016 : Frankfurt, Germany)*, vol. 9945, pp. 188–206, Jun. 2016.

11. S. Kumar *et al.*, "PAMI: A parallel active message interface for the BlueGene/Q supercomputer," *Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Shanghai, China, 01-May-2012.
12. S. Kumar, Y. Sun, and L. V. Kale, "Acceleration of an Asynchronous Message Driven Programming Paradigm on IBM Blue Gene/Q," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, 2013.
13. G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of Chemical Physics*, vol. 126, no. 1, Jan. 2007.
14. B. Acun, R. Buch, L. V. Kale, J. Phillips, "NAMD: Scalable Molecular Dynamics Based on the Charm++ Parallel Runtime System," *Exascale Scientific Applications: Scalability and Performance Portability*. Chapman and Hall/CRC, 2017.
15. R. Elber, A. P. Ruymgaart, and B. Hess, "SHAKE parallelization.," *The European physical journal. Special topics*, vol. 200, no. 1, pp. 211–223, Nov. 2011.
16. B. Hess, "P-LINCS: A parallel linear constraint solver for molecular simulation," *Journal of Chemical Theory and Computation*, vol. 4, no. 1, pp. 116–122.

Bilge Acun *IBM Research Division, IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598, USA (Bilge.Acun2@ibm.com)*. Dr. Acun received a B.S. degree in 2012 from the Department of Computer Science at Bilkent University in Ankara, Turkey and a Ph.D. in 2017 from the Department of Computer Science at University of Illinois at Urbana-Champaign in Illinois, USA. Her dissertation focused on improving the energy efficiency of the large scale systems. She joined IBM as a Research Staff Member after completing her Ph.D. Her research interests include improving the performance of massively parallel applications, parallel runtimes, energy efficiency and variability in large scale systems.

David J. Hardy *Theoretical and Computational Biophysics Group, Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA (dhardy@ks.uiuc.edu).* Dr. Hardy received a B.S. in Mathematics and Computer Science in 1994 from Truman State University, an M.S. in Computer Science in 1997 from the Missouri University of Science and Technology, and a Ph.D. in Computer Science in 2006 from the University of Illinois at Urbana-Champaign. He develops the NAMD parallel molecular dynamics application. His research interests include fast methods for calculating electrostatics, numerical methods for time integration, and GPU computing.

Laxmikant V. Kale *Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA (kale@illinois.edu).* Prof. Kale received the B.Tech degree in Electronics Engineering from Benares Hindu University, Varanasi, India in 1977, and a M.E. degree in Computer Science from Indian Institute of Science in Bangalore, India, in 1979. He received a Ph.D. in computer science in from State University of New York, Stony Brook, in 1985. He joined the faculty of the University of Illinois at Urbana-Champaign in 1985, where he is now the Paul and Cynthia Saylor Professor of Computer Science and the director of the Parallel Programming Laboratory. Prof. Kale has been working on various aspects of parallel computing, with a focus on enhancing performance and productivity via adaptive runtime systems, and with the belief that only interdisciplinary research involving multiple CSE and other applications can bring back well-honed abstractions into Computer Science that will have a long-term impact on the state-of-art. His collaborations include the widely used Gordon-Bell award winning (SC 2002) biomolecular simulation program NAMD, computational cosmology (ChaNGa), and quantum chemistry (OpenAtom). He takes pride in his group's success in distributing and supporting software embodying the research ideas, including Charm++, and Adaptive MPI. Prof. Kale is a fellow of the ACM and IEEE, and a co-winner of the 2012 IEEE Sidney Fernbach award.

Ke Li *Nvidia Corporation, Santa Clara, CA 95051, USA (kel@nvidia.com).* Dr. Li received a B.S. degree in Electrical Engineering in 2011 from Fudan University, China and a Ph.D. degree in Electrical Engineering in 2017 from Washington University in St. Louis, USA. His dissertation focused on enabling novel PET (Positron Emission Tomography) systems by using GPU parallel computing to achieve near real time reconstruction. He joined Nvidia in 2017 as a senior developer technology engineer. He is interested in high performance computing applications for molecular dynamics, medical imaging with deep learning focus.

James C. Phillips *NCSA Blue Waters Project Office, University of Illinois at Urbana-Champaign, Urbana IL 61801, USA (jcphill@illinois.edu).* Dr. Phillips received a B.S. in Physics and Mathematics in 1993 from Marquette University in Milwaukee, WI, and an M.S. in 1994 and a Ph.D. in 2002, both in Physics, from the University of Illinois at Urbana-Champaign. From 1999-2017 he was the lead NAMD developer at the Theoretical and Computational Biophysics Group at the Beckman Institute, University of Illinois at Urbana-Champaign. During this time Dr. Phillips received a 2002 Gordon Bell Award and NAMD PIs Klaus Schulten and Laxmikant Kale shared the 2012 IEEE Sidney Fernbach Award recognizing the contributions of NAMD to high performance computing. Dr. Phillips is now the quality assurance lead in the Blue Waters Project Office of the National Center for Supercomputing Applications, where he is working to ensure the scientific productivity of the next several generations of supercomputers.

John E. Stone *Theoretical and Computational Biophysics Group, Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA (johns@ks.uiuc.edu).* Mr. Stone received B.S. and M.S. degrees in Computer Science from the Missouri University of Science and Technology in 1994, and 1998. Mr. Stone is the lead developer of VMD, a high performance molecular visualization tool used by researchers all over the world. His research interests include molecular visualization, GPU computing, parallel computing, ray tracing, haptics, virtual environments, and immersive visualization. Mr. Stone was inducted as an NVIDIA CUDA Fellow in 2010. In 2015 Mr. Stone joined the Khronos Group Advisory Panel for the Vulkan Graphics API. In 2017 and 2018 Mr. Stone was awarded as an IBM Champion for Power, for innovative thought leadership in the technical community. Mr. Stone is a member of ACM and IEEE.