



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Performance Analysis, Modeling and Scaling of HPC Applications and Tools

A. Bhatele

January 13, 2016

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

PERFORMANCE ANALYSIS, MODELING AND SCALING OF HPC APPLICATIONS AND TOOLS

Principal Investigator: Abhinav Bhatele

Project Summary

Efficient use of supercomputers at DOE centers is vital for maximizing system throughput, minimizing energy costs and enabling science breakthroughs faster. This requires complementary efforts along several directions to optimize the performance of scientific simulation codes and the underlying runtimes and software stacks. This in turn requires providing scalable performance analysis tools and modeling techniques that can provide feedback to physicists and computer scientists developing the simulation codes and runtimes respectively.

The PAMS project is using time allocations on supercomputers at ALCF, NERSC and OLCF to further the goals described above by performing research along the following fronts: 1. Scaling Study of HPC applications; 2. Evaluation of Programming Models; 3. Hardening of Performance Tools; 4. Performance Modeling of Irregular Codes; and 5. Statistical Analysis of Historical Performance Data.

We are a team of computer and computational scientists funded by both DOE/NNSA and DOE/ASCR programs such as ECRP, XStack (Traleika Glacier, PIPER), ExaOSR (ARGO), SDMAV II (MONA) and PSAAP II (XPACC). This allocation will enable us to study big data issues when analyzing performance on leadership computing class systems and to assist the HPC community in making the most effective use of these resources.

1 Project Milestones and Accomplishments

Scaling of OpenAtom We have been doing performance and scaling runs for the OpenAtom code from the University of Illinois at Urbana-Champaign. OpenAtom is a parallel simulation software for studying atomic and molecular systems based on quantum chemical principles. In contrast to classical computational molecular dynamics based on Newtonian mechanics, it uses the Car-Parrinello Ab Initio Molecular Dynamics (CPAIMD) and Born Openheimer Molecular Dynamics (BOMD) approaches. Instead of using an empirical force function, the CPAIMD and BOMD algorithms compute the forces acting on each atom as the summation of multiple terms derived from plane-wave density functional theory. This allows OpenAtom to study complex atomic and electronic physics in semiconductor, metallic, biological and other molecular systems.

We are using our allocation on Titan and Mira to better understand the communication patterns in OpenAtom and to improve its performance. In addition, we are also running OpenAtom with multiple beads. We are analyzing how well OpenAtom scales when we run it with a large number of beads. In the future, we plan to integrate tempering and spins as well in our application which would require even more computational resources.

1. We made significant advances in incorporating multi-instance method support in OpenAtom. There are 4 kinds of multi-instance methods: (1) Path Integrals (or Beads), (2) k-points, (3) Tempering and (4) Spins. We fully implemented and tested the Beads in OpenAtom. We achieved strong scaling results on BG/Q (Mira and Vulcan) and BlueWaters. We have also fully implemented k-points and Tempering and are currently in the process of doing the scaling runs for these. As far as spin is concerned, we are not yet done with its implementation.
2. We have made OpenAtom interoperable with MPI. In this way, OpenAtom can now benefit from MPI libraries. Currently, we are using ScaLAPACK MPI library for computing eigenvalues and eigenvectors of some matrices in OpenAtom.
3. Current OpenAtom code is based on a quantum mechanical theory called DFT (density functional theory). We are now extending OpenAtom to also include another method called Hartree-Fock Self-Consistent Field method. We have implemented the first phase of this method in which the outerproduct of wavefunction is calculated.

In the recent paper that we submitted (which is being reviewed), we scaled single bead of OpenAtom for MOF dataset (with 819 states) upto 32,768 cores. For multiple-bead method, we showed strong scaling upto 262,144 cores on BG/Q.

Charm++ version of ROSS Mira has been used to show concrete performance improvements when running ROSS on top of Charm++ instead of MPI. Weve demonstrated these performance gains by running two models built on top of ROSS: PHOLD and Dragonfly. In doing so we are able to show improvements in a variety of communication patterns and work-loads by taking advantage of Charm++s over decomposition and message-driven scheduling. Now that all of the basic functionality of ROSS has been ported to the Charm++ version we will focus on adding new features such as automatic load balancing and asynchronous GVT support, both of which are enabled by Charm++

Charm++ version of Kripke Kripke is a proxy application for Sn deterministic particle transport. We have ported Kripke to Charm++, and we are analyzing the performance of our parallel sweeps with respect to overdecomposition, load balancing, and mapping decisions to better inform our understanding of different programming models.

TraceR simulation framework Using the TraceR simulation framework, we have conducted studies to compare different networks on Titan. These simulations have been performed for three commonly used networks - torus, dragonfly, and fat-tree, each with 46K nodes and 3 million MPI ranks in the prototype systems. Five mini-applications and communication kernels have been used to test various mappings and find the best performing configuration for each of the network. Results from these experiments have been presented as part of a PhD thesis defense and will be used in a manuscript currently being prepared.

Development and testing of TAU Our usage of the ALCC time has primarily focused on development and testing of the TAU performance analysis system. An area of particular effort has been testing and improving our support for analysis of GPU accelerated code on Titan. Full support of the software stack and underlying hardware resources by TAU will facilitate our future investigation of the performance properties of various software projects on these platforms as well as hardware benchmarking. Continuing effort in these areas has yielded improvements to TAU with respect to both the hardware and software systems in use under these allocations, enabling more effective performance analytics.

Modeling the performance of linear solvers We have initiated our study for modeling the performance of scientific applications based on sparse linear algebra. Optimizing resource usage requires knowledge of how applications will behave on the given input. This is challenging for real applications and important for performance. To enable the optimal execution of applications that rely on linear system abstractions or that are built on top of modules relying on such abstractions, we develop a model to predict, given the input data, the performance of a linear solver together with a preconditioner, which are common components in many scientific applications. There exist multiple choices available for the components. We plan to predict the best choice for handling the given input. Application modules can benefit from this in choosing the optimal components for the input passed from other modules.

Modeling application performance We are working on using program paths as a method for application modeling. It is both time-consuming and difficult to develop models that account for communication-computation overlap, imbalance, and irregularity. Critical path analysis is a promising technique for developing such models because paths capture the computational sequences, excluding as much communication as possible, that determine execution time. To make critical path analysis practical, it is necessary to make paths easier to collect (unlike a profile, it is necessary to retain edges) and digest (because they are very long). We are developing a tool that forms path-based models by collecting near critical paths, where paths are formed from program tasks, and where key tasks are modeled.

Inter-job Interference Predictable performance is important for understanding and alleviating application performance issues; quantifying the effects of source code, compiler, or system software changes; estimating the time required for batch jobs; and determining the allocation requests for proposals. Our experiments show that on a Cray XE system, the execution time of a communication-heavy parallel application ranges from 28% faster to 41% slower than the average observed performance. Blue Gene systems, on the other hand, demonstrate no noticeable run-to-run variability. In this project, we focus on Cray machines and investigate potential causes for performance variability such as OS jitter, shape of the allocated partition, and interference from other jobs sharing the same network links [1]. Reducing such variability could improve overall throughput at a computer center and save energy costs.

Scaling of pF3D Scaling studies on Edison and Titan will look at the impact of irregular placement of MPI processes on the interconnect. Our expectation is that the dragonfly interconnect on Edison will perform better than the shared 3D torus on Titan. We may also run the scaling study on Sequoia if a system software issue on Mira continues to “gobble up” so much memory that we are unable to run with more than one MPI process per core.

This work will measure checkpoint rates at full system scale on Mira, Titan, Edison, and Sequoia. There are no known architectural problems that would prevent pF3D from achieving good I/O rates at these scales. We intend to run a test problem with a fixed number of zones per process on these systems. We will also run with a higher zone count on Titan to reflect the increased work per process we will use when pF3D is able to exploit GPUs. The I/O package has tuning parameters and we will report on the best choices for each system and attempt to correlate the differences between the file systems.

2 Project Productivity

2.1 Publications and Presentations

OpenAtom Update: A paper has been submitted for multi-instance method support in OpenAtom project and is currently under review.

ROSS Update: A paper will be submitted to SIGSIM PADS 2016 describing the improvements from porting ROSS, a highly scalable PDES engine written in MPI, to Charm++, an adaptive message-driven runtime system.

TraceR Update: Results collected on Titan have been used in the PhD thesis of Nikhil Jain, which will be publicly available in February. Also a manuscript is being prepared for submission later this month.

2.2 Other

We released a new version of the TAU performance analysis suite. Improvements and bug fixes were informed in part by testing and feedback from our performance tuning efforts using target codes on ALCC systems.

3 Center Feedback

4 Code Description and Characterization

NAMD NAMD is a highly scalable parallel molecular dynamics simulation code used for protein folding and drug discovery at scales ranging from a few thousand to millions of atoms [2]. NAMD is built on top of the Charm++ runtime system and is used widely at NSF and DOE centers. WE plan to evaluate and improve the performance of NAMD for various molecular system sizes on various platforms to support its wide user base.

OpenAtom OpenAtom is implemented on top of Charm++, which is an over-decomposition based parallel programming framework that provides support for message-driven execution of migratable entities empowered by an adaptive runtime system. Charm++ encourages decomposition of parallel computation using units that are natural to the application domain, instead of dividing data into as many pieces as processors. In particular, OpenAtom decomposes the data and the computation across a number of chare objects, whose type and/or number only depend on the AIMD algorithm and the desired grainsize. This allows OpenAtom to exploit the underlying mathematics via a seamless mix of both data and functional decompositions resulting in greater expressed parallelism, and several overlapping phases of computation combined with a longer critical path of dependent computations. The current implementation of OpenAtom in Charm++ is highly scalable, and has exhibited portable performance across three generations of the IBM Blue Gene family and Cray systems, apart from other supercomputing platforms. In OpenAtom, we heavily use the FFTW library. We also use acml.

OpenAtom is now interoperable with MPI and uses the following libraries: BLAS, BLACS, LAPACK and ScaLAPACK. We are also working to incorporate another library called ELPA (Eigenvalue Solvers for Petaflop-Applications) in OpenAtom but it has not yet been completed.

We have been using our allocation on Titan and Mira to do simulations on primarily 2 datasets, water (256 molecules with 70 Ry cutoff) and mof (with 1000 states). For running the simulation for 1 bead, we typically use 128 nodes (on both Titan and Mira). For scientific studies, we need to run OpenAtom with 64 beads, which require 8192 nodes (128*64). Once we integrate tempering

and spins into OpenAtom, we would need to do very large simulations which can easily take up the entire machine.

pF3D pF3D [3, 4, 5] simulates laser-plasma interactions in experiments at the National Ignition Facility (NIF) and other laser facilities. Simulating the full path of the four laser beams in one “quad” in a NIF experiment requires over 50 billion zones and simulating the beams in three quads requires 0.2-1.0 trillion zones. Due to the size of these simulations, scalability is a key concern during the development of pF3D.

pF3D makes a paraxial approximation (all light is assumed to propagate at small angles relative to the laser direction). pF3D uses spectral methods for some physics operators. Due to the paraxial approximation, these operators result 2D FFTs that span the full x- and y-extent of the simulation. The large amount of data moved in the course of these FFTs and their non-local character means that high MPI message rates are required to achieve good performance. pF3D uses custom MPI rank to torus location mappings on Blue Gene/Q systems to improve performance. pF3D performance suffers when there is contention with other processes for link bandwidth. We will run scaling studies on Mira, Titan, and Edison to see how pF3D performs on three very different interconnects and attempt to identify the source of low or variable performance.

Delivering acceptably quick checkpoint restart times is one of the challenges for future large HPC systems. The large zone counts in pF3D simulations mean that achieving high checkpoint I/O rates is required to obtain good efficiency. We plan to measure the checkpoint performance of pF3D on Mira, Titan, Edison, and Sequoia (an NNSA system at LLNL). The runs on Mira and Sequoia will provide a direct comparison of GPFS and Lustre. The tests on Titan will use fewer MPI processes with more zones per process to reflect the I/O challenges of a system with “accelerated nodes”. Edison, in this case, serves as our “traditional x86 cluster” reference point.

Trilinos We employ Trilinos for our case study. Trilinos is a platform to develop complex multi-physics engineering and scientific applications. In our case study of modeling the performance of sparse linear algebra with finite element method (FEM), we generate the sparse matrix input data using the finite element mesh discretization library, MFEM. We use these matrices for representing four arbitrarily chosen types of physics problems using 31 mesh geometries at various discretization order and mesh refinement level. We employ 9 linear solvers and 13 preconditioner choices from Trilinos. Then, we executes each combination of preconditioner, solver, matrix, and problem to produce data for empirical modeling.

DEVE DEVE (discrete event viral evolution) simulates the evolution of quasispecies of positive-sense single strand RNA viruses, especially, Flaviviruses such as Dengue. We use allocation mostly on modeling and development of the application. This application exhibits behaviors that are very different from other scientific applications based on linear algebra. The model relies on the optimistic parallel discrete event simulation paradigm. We have implemented hybrid rollbacks based on application-level reverse computation in addition to a small amount of state savings. We leverage the Rensselaer’s Optimistic Simulation System (ROSS), which is based on the Time Warp mechanism. The model code is implemented in C++ employing many of boost libraries.

DEVE simultaneously simulate the random activity of RNA viruses inside of a cell as well as the diffusion of virions within the cell population. We model cells in a culture medium divided into pixels, each containing zero or more cells. Each pixel, represented as a logical process (LP), independently simulates two major types of activities in the location, which are extracellular and intracellular. Simulation produces the quasispecies evolved from a given initial viral and cell populations as a

result of genotype-specific translation, replication, and mutation rates as well as receptor binding fitnesses. The results produced are written into files using MPI-IO. A typical production run will require $10^5 - 10^7$ processors for an hour.

Kripke Kripke is a proxy app developed at LLNL for Sn deterministic particle transport codes. Kripke solves for the flux of all particles in a volume of interest at a certain time given some initial knowledge of the materials, boundary conditions, and any particle-generating sources inside the domain. It does so by performing parallel sweeps over the domain for every energy group and direction of interest. We have ported it to run on Charm++ and also Adaptive MPI.

We are using the allocation on Titan to better understand the performance of sweeps at scale, comparing Charm++ support for asynchrony with the existing MPI reference implementation as well as that same implementation running on Adaptive MPI, an implementation of MPI on top of Charm++.

Code Name	Dense Linear Algebra	Sparse Linear Algebra	Monte Carlo	FFTs	Particles	Structured Grids	Unstructured Grids	AMR
Kripke						■		
NAMD				■	■			
OpenAtom	■			■	■			
pF3D						■		
Trilinos		■						

Table 1: Algorithmic motifs in each of our major production codes

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-TR-680812).

References

- [1] Abhinav Bhatele, Kathryn Mohror, Steven H. Langer, and Katherine E. Isaacs. There goes the neighborhood: performance degradation due to nearby jobs. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '13. IEEE Computer Society, November 2013. LLNL-CONF-635776.
- [2] Abhinav Bhatele, Laxmikant V. Kalé, and Sameer Kumar. Dynamic topology aware load balancing algorithms for molecular dynamics applications. In *23rd ACM International Conference on Supercomputing*, 2009.
- [3] Steven Langer, Abhinav Bhatele, Todd Gamblin, Bert Still, Denise Hinkel, Mike Kumbara, Bruce Langdon, and Ed Williams. Simulating laser-plasma interaction in experiments at the national ignition facility on a Cray XE6. In *Cray User Group Meeting*, CUG '12, April 2012. LLNL-PROC-547711.
- [4] C. H. Still, R. L. Berger, A. B. Langdon, D. E. Hinkel, L. J. Suter, and E. A. Williams.

Filamentation and forward brillouin scatter of entire smoothed and aberrated laser beams. *Physics of Plasmas*, 7(5):2023–2032, 2000.

- [5] R. L. Berger, B. F. Lasinski, A. B. Langdon, T. B. Kaiser, B. B. Afeyan, B. I. Cohen, C. H. Still, and E. A. Williams. Influence of spatial and temporal laser beam smoothing on stimulated brillouin scattering in filamentary laser light. *Phys. Rev. Lett.*, 75(6):1078–1081, Aug 1995.