SAND2018-5311R

Sandia National Laboratories

Operated for the United States Department of Energy
by National Technology and Engineering Solutions
of Sandia, LLC.

Albuquerque, New Mexico 87185-0101
Livermore, California 94551-0969

date: May 8, 2018

to: Distribution

from: Stan G. Moore (1444), Michael A. Gallis (1513)

subject: Full Trinity run with SPARTA

A heterogeneous run on the full Trinity supercomputer at LANL was performed using SPARTA during March 9-12, 2018. Over 19,000 nodes (9200+ Haswell and 9900+ KNL) and 1.2 million MPI processes were used. The run was successful, with SPARTA running for several hours with good performance (better than the same simulation running on full Sequoia). However, several challenges were encountered, and some unresolved issues remain.

SPARTA is a Direct-Simulation Monte-Carlo (DSMC) code [1-2]. The DSMC method resolves fluid length scales at the particle level and accurately models high-altitude hypersonic re-entry flows. DSMC can be used for non-equilibrium, non-continuum conditions that cannot be simulated with traditional CFD or reproduced experimentally. In addition to re-entry, SPARTA has also been used to model spacecraft, fluid instabilities, turbulence, and porous media.

The turbulent energy evolution for an argon flow at a Reynolds number 500 and a Mach number 0.3 (at the limits of compressibility), was studied during the run. A similar flow has been studied previously by Gallis et al [3] using the Sequoia supercomputer at LLNL. In the incompressible limit, the DSMC simulations were found [3] to agree with corresponding Navier-Stokes Direct Numerical Simulation (DNS) results. The computational power of Trinity offers the possibility of further extending molecular calculations into the compressible regime, where DNS simulations may lose accuracy due to sharp gradients. Thus, molecular simulations will allow new insights into turbulence by directly linking molecular relaxation processes to macroscopic transport processes.

For the full Trinity fun, the same parameters as were run on Sequoia were used to verify correctness and compare performance between the two machines. A 3D grid with 8 billion grid cells (2000 x 2000 x 2000) was used, with approximately 45 particles/cell, giving at total of 360 billion particles. The particular case studied was a very well load-balanced case. The nominal (excluding statistical fluctuations in the number of simulators) workload between cores did not vary more than 3%. Figure 1 presents the energy and energy dissipation as a function of time from the Sequoia and the Trinity runs. Both time and energy have been normalized. We note that the results for this particularly challenging problem (for a DSMC code) are practically identical between Sequoia and Trinity, suggesting that the methods used on Trinity have not affected the quality of the simulation.

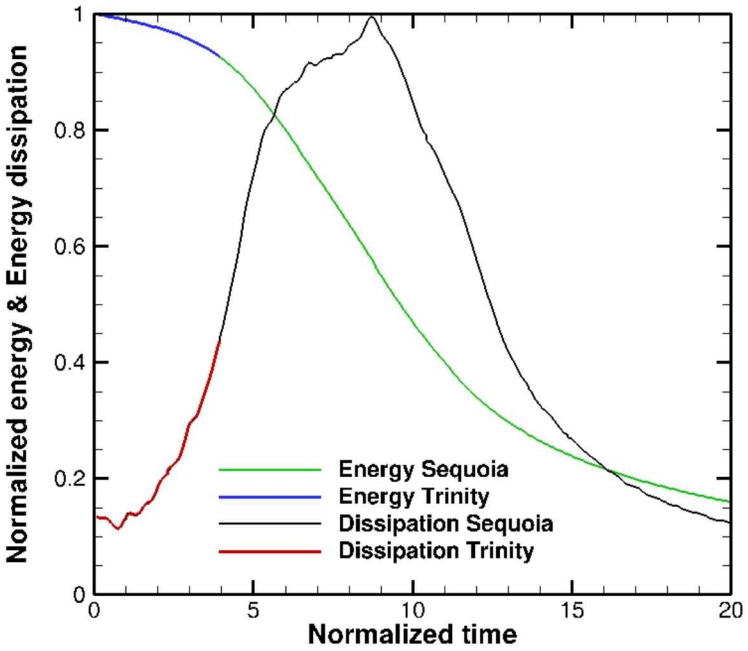

Figure 1. Energy and energy dissipation for a Re=500, Ma=0.3 Taylor-Green Argon flow.

The Trinity machine has two different types of nodes. The more traditional Haswell CPU nodes have 32 physical cores with 2 hardware threads each and support AVX2 vector instructions. The Intel Xeon Phi Knight's Landing (KNL) nodes have 68 physical cores with 4 hardware threads each and support AVX-512 vector instructions. An executable compiled for KNL using AVX-512 cannot run on Haswell. Even though KNL has 68 physical cores, it is often faster to use only 64 of the physical cores per node.

To run on both types of nodes simultaneously, separate executables were compiled for each node type. The KNL executable used Sandia's Kokkos performance portability library [4] to support OpenMP multithreading, along with AVX-512 vector instructions. The Haswell executable used the Kokkos Serial backend (no OpenMP support), along with AVX2 vector instructions. A driver program written by Mike Davis (Cray) was then used to determine the node type and launch either the Haswell or KNL executable, based on node type. A total of 64 MPI ranks were used on both node types. For Haswell, MPI ranks were placed on the hyperthreads, while for KNL, 4 OpenMP threads per MPI rank were placed on the hyperthreads to fill the node with tasks. Using 64 MPI ranks on Haswell and 64 MPI x 4 OpenMP threads on KNL has been found to give reasonable performance with SPARTA.

Unfortunately, the version of the Slurm workload manager on Trinity did not support running separate "srun" commands on the different nodes, so the affinity options to "srun" were shared between Haswell and KNL. The options used were "--cpu_bind=core". This may have led to slightly suboptimal affinity for one of the node types but was a necessary compromise. It also wasn't possible to use core specialization (i.e. "-S 4") for KNL because it isn't supported on Haswell when using 64 MPI. These limitations for heterogeneous jobs should be addressed in a future Slurm version.

Prior experience has shown that Trinity is prone to both node failures and SIGBUS errors. For example, when running LAMMPS during the Trinity KNL Open Science period, Stan Moore (org. 1444) encountered a total of 23 nodes failures. Additionally, users frequently see SIGBUS errors on Trinity when running large jobs. Node failures, such as a kernel panic, cause an entire node to go down until it is rebooted by system administrators. SIGBUS errors do not cause the node to go down, but will kill the

"srun" command. We observed 6 hardware failures during the full Trinity run than killed the job: 3 SIGBUS errors and 3 node failures.

SPARTA supports restarting using binary checkpoint files. Checkpoint files were periodically written to disk. For this simulation, the checkpoint file size was more than 20 TB, so to save space, checkpoint files were toggled between two file names. Checkpoint files were written to the Cray DataWarp burst buffers to improve file I/O performance. When the job was terminated, the checkpoint files were automatically staged out to the Lustre parallel file system.

In order to improve job robustness, the "srun" command was put inside a for-loop, and when the job failed, the job was automatically restarted (in the same jobid) using the latest checkpoint file. A few extra nodes were requested for backup (but not used in the "srun" command) in case of node failure. This method worked well for the SIGBUS errors: the job automatically restarted and little time was lost due to these failures. However, node failures unexpectedly killed the entire job, not just the "srun" command. To recover from node failures, multiple jobs were submitted to the queue so that if one job was killed, another job would begin using the latest checkpoint file. It was later determined that using the Slurm "--no-kill" option will allow the job to continue after node failure. However this is currently not a viable option because testing revealed the subsequent "srun" command hung and the DataWarp reservation was prematurely deallocated after node failure. These issues have been reported to the LANL HPC support, and the "srun" hang is expected to be fixed in a future version of Slurm.

Twice the latest checkpoint file was corrupted after trying to restart from a node failure. This file corruption caused a segmentation fault or internal error in the SPARTA code. It was necessary to manually switch to the backup checkpoint file, otherwise the simulation couldn't make progress. Corruption was again reproduced multiple times after the DAT when using DataWarp, but a root cause wasn't identified. It is difficult to detect subtle corruption of checkpoint files outside of SPARTA. Since the DAT, robustness to checkpoint file corruption has been improved by using temporary files to signal file corruption after the file is read in by SPARTA. When corruption is found, the script automatically switches to the backup checkpoint file. This method will be used in future runs.

Initially, SPARTA performance on full Trinity was much slower than expected (but the code was running correctly and making forward progress). Built-in timers showed that most of the time was spent outside of normal computation (such as the particle move, sort, and collide algorithms) and was instead in the "other" category. This suggested a severe load imbalance issue. Using static load balancing showed better performance when most of the work was shifted from the Haswell nodes to the KNL nodes, but prior benchmarking on the Mutrino testbed showed that the KNL and Haswell nodes were less than 2x imbalanced. This suggested a very slow Haswell node. Timers were added to the code to find the MPI process with the largest time spent in particle move. The ID of the slow node was determined, and the node was excluded from the allocation This process was repeated until two slow Haswell nodes were eliminated, allowing the simulation to run over 20x faster and giving the expected performance. If even one node is running slow, the performance of the entire simulation will significantly degrade due to MPI barriers or other synchronization points in the program. A small standalone MPI program has been written to easily determine slow nodes and will be used in future runs to eliminate slow nodes.

These hardware issues have been reported to the Trinity system admins. The SIGBUS error issue is being investigated by Mike Davis from Cray, with a workaround that successfully reduces the frequency of the errors when the code executable is fully statically linked and the executable is placed in the /tmp folder on the compute nodes. The root causes of the node failures and checkpoint file corruption are still

unknown. Slow nodes continue to plague Trinity: a two more slow Haswell nodes were found using SPARTA later during ATCC5 and reported to system admins. One slow node was found to have faulty hardware.

Besides hardware issues, there were a few code issues exposed in SPARTA. SPARTA hung when reading in checkpoint files at full scale. This was caused by a bit shift that was too large and overflowed a 32 bit integer, leading to undefined behavior. Interestingly, this issue didn't show up on Sequoia at full scale. This issue was fixed during the DAT. A second MPI hang was observed when using the Kokkos version of timer-based dynamic load balancing. The issue didn't show up on Mutrino testbed for small problems. During the full run, we switched to static load balancing, but found for this simulation, using equal work between Haswell and KNL nodes was near optimal. The cause of the second hang (when using dynamic load balancing) was later fixed after the DAT, and this capability will be used in future runs.

Overall the performance of the SPARTA on full Trinity was better than the same simulation running on full Sequoia. SPARTA performance on full Trinity was about 8 timesteps/s, while performance for the same simulation running on one-third of Sequoia was about 2.3 timesteps/s, or projected 6.9 timesteps/s on full Sequoia (neglecting MPI scaling overhead). Without checkpoint file I/O, performance on one-third of Sequoia was about 2.6 timesteps/s, or projected 7.8 timesteps/s on full Sequoia.

This speed-up is a particularly important step for molecular simulations to be applied to turbulent flows. Currently, turbulence is almost exclusively studied at the hydrodynamic (continuum) level. Molecular-level simulations of turbulence have received little, if any, attention to date because the molecular scales and the turbulent scales are considered to be many orders of magnitude apart, and, as a result, molecular turbulence simulations have been heretofore considered to be physically unnecessary and computationally intractable. However, there are cases of practical interest in which the Kolmogorov length and time scales, which are the smallest length and time scales in a turbulent flow, can be within 1-2 orders of magnitude of the mean free path and the mean collision time. Recently, kinetic (molecular-level) effects have been observed in Inertial Confinement Fusion (ICF) and combustion. Thus, taking advantage of ever-increasing computer power, DSMC offers the possibility of generating molecular-level data sets that complement existing computational and experimental data sets. DSMC offers a natural way to study the effects on turbulence of phenomena arising from molecular processes such as compressibility, viscosity, thermal conductivity, and diffusivity. Moreover, the effect of surface roughness on wall-bounded turbulence can be investigated fundamentally because DSMC treats gas-wall interactions at the molecular level. Insights gained could lead to improved turbulence models for continuum codes.

For a next full Trinity run, we propose running a higher Mach number case. The higher Mach number will lead to much larger density variations, requiring the use of dynamic load balancing. Job robustness will be greatly improved by addressing failure modes discovered during the previous full run. For example, slow nodes will be detected and excluded at the beginning of the DAT using a standalone MPI program, and the checkpoint/restart capability will be robust to file corruption by automatically switching to the backup file if necessary. When later Slurm versions are installed on Trinity, performance of the heterogeneous runs may also be improved by using the enhanced options for heterogeneous jobs.

International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] G. A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Oxford University Press, Oxford, UK (1994)
[2] M. A. Gallis, J. R. Torczynski, S. J. Plimpton, D. J. Rader, and T. Koehler, Direct Simulation Monte Carlo: The Quest for Speed, Proceedings of the 29th Rarefied Gas Dynamics (RGD) Symposium, Xi'an, China, July 2014, http://sparta.sandia.gov
[3] M. A. Gallis, T. P. Koehler, J. R. Torczynski, S. J. Plimpton, G. Papadakis, Molecular-Level Simulations of Turbulence and its Decay, Phys. Rev. Lett. **118**, 064501 (2017).
[4] Carter Edwards, H., Trott, C. R. and Sunderland, D. J., Kokkos: Enabling manycore performance portability through polymorphic memory access patterns, Parallel Distr. Com., 74, 12 (2014), 3202-3216.