Lawrence Livermore National Laboratory

LLNL Mercury Project Trinity Open Science Final Report

Patrick Brantley, Shawn Dawson, Scott McKinley, Matt O'Brien, Doug Peters, Mike Pozulp, Greg Becker, Kathryn Mohror, Adam Moody

April 20, 2016

LLNL-TR-689799

Disclaimer

High Level Summary of Project Science Exploration

The Mercury Monte Carlo particle transport code [1,2] developed at Lawrence Livermore National Laboratory (LLNL) is used to simulate the transport of radiation through urban environments. These challenging calculations include complicated geometries and require significant computational resources to complete. As a result, a question arises as to the level of convergence of the calculations with Monte Carlo simulation particle count. In the Trinity Open Science calculations, one main focus was to investigate convergence of the relevant simulation quantities with Monte Carlo particle count to assess the current simulation methodology. Both for this application space but also of more general applicability, we also investigated the impact of code algorithms on parallel scaling on the Trinity machine as well as the utilization of the Trinity DataWarp burst buffer technology in Mercury via the LLNL Scalable Checkpoint/Restart (SCR) library [3,4].

We ran multiple suites of urban modeling calculations to assess Monte Carlo particle convergence. Specifically, we ran suites of 1×10^9 particles on 1,152 processors (standard methodology), 2×10^9 particles on 2,304 processors, 4×10^9 particles on 4,608 processors, 8×10^9 particles on 9,216 processors, and 16×10^9 particles on 18,432 processors. All of the simulation suites successfully completed with the exception of one calculation of the largest suite that encountered an internal code error that we were not able to resolve before the end of the Trinity Open Science period. These simulations confirmed that the number of Monte Carlo particles used in standard simulations (1×10^9) agree to within one standard deviation with the simulations with the largest number of particles. The estimated statistical standard deviation of the results decreased with increasing particle count but at a lower rate than anticipated. Further work will be required to fully understand this aspect of the convergence study results.

Our investigation of Mercury parallel scalability on Trinity was very productive. Using the urban modeling user problem, we investigated weak scaling (with number of Monte Carlo simulation particles) for the “AllProcessorTree”, “Blocking”, and “NonBlocking” options for the “test for done” algorithm in the code. The simulation time for the three algorithmic options for different numbers of processors is shown in Fig. 1. The “NonBlocking” algorithm produced nearly flat weak scaling from 1,152 processors to 18,432 processors, exhibiting a maximum of 8% loss of efficiency. These simulations using the “NonBlocking” option did exhibit a significant degradation in weak scaling at 36,864 processors, but the overall run time using this algorithm remained significantly lower than achieved using the other code algorithm settings. The knowledge gained through the Trinity Open Science Period calculations with respect to parallel scaling will likely benefit future calculations.

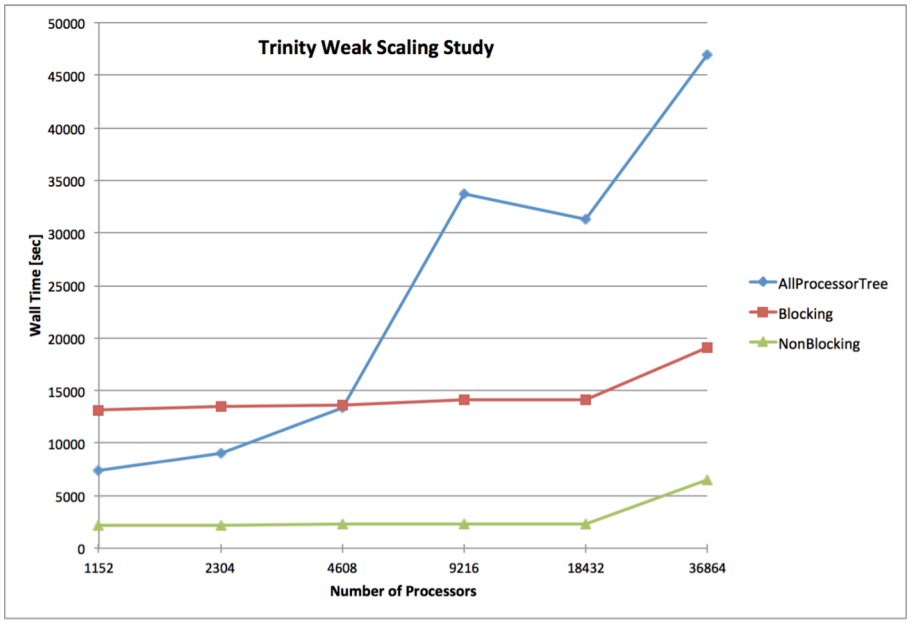

Figure 1 – Trinity urban modeling user problem weak scaling results

The Scalable Checkpoint/Restart (SCR) library [3,4] development team planned to port the SCR library to Trinity and the DataWarp burst buffer capability in stages. The initial step was to port SCR to run on Trinity's RAM disk followed by using progressively more advanced features of DataWarp. We planned to first use DataWarp as a private, scratch "node-local" file system. After that, we planned to add DataWarp C API calls in SCR to transfer files with private, scratch mode. Finally, we planned to use the DataWarp pre/post stage features, which would require shared, scratch mode. The first three stages require minimal changes to SCR, while the shared, scratch mode requires more substantial SCR development effort.

The SCR library port to RAM disk uncovered a number of problems, which have largely been resolved. SCR is now working well with RAM disk on Trinity. We believe we have everything in place to try DataWarp as a "node-local" file system, but this feature of DataWarp was not initially ready. While waiting for that capability, we proceeded to begin to modify SCR to use DataWarp in shared, scratch mode. Those SCR changes are still in progress.

The Mercury code used the Scalable Checkpoint/Restart (SCR) library [3,4] to leverage the storage hierarchy on Trinity for I/O speedup during checkpoint and restart. For a weak scaling test problem, we compared writing checkpoints and reading restarts for the RAM disk and Lustre parallel file system, scaling up from 1 node (32 processors) to 4,096 nodes (131,072 processors) in powers of two. Checkpointing to RAM disk instead of Lustre produced speedups at 16 nodes (512 processors) and above, including a 30X maximum speedup. Reading restarts from RAM disk instead of Lustre produced speedups for all node counts but one, including a 9X maximum speedup. For the urban modeling user problem, writing checkpoints to RAM disk instead of Lustre reduced median time-to-checkpoint by 20X. While exercising the RAM disks on Trinity, we discovered that some RAM disks were left in an inconsistent state that caused SCR to abort

on a `mkdir` failure. We reported the issue to LANL, who discovered that the scripts necessary for wiping the RAM disk between jobs that were utilized on Trinitite had not been installed on Trinity. Finally, there was insufficient time between the release of the first functional version of the DataWarp software/hardware and the end of the Trinity Open Science period for the SCR team to finish porting SCR to Trinity's Burst Buffers. Thus, we were not able to collect the Burst Buffer performance data that we had hoped to collect with the use of SCR in Mercury.

Preliminary results from the Mercury/SCR library studies will be presented at the DOE Centers of Excellence Performance Portability Meeting on April 19-21, 2016.

Technical Problems Encountered

The Mercury code development team encountered some code issues that we were unable to debug as a result of the executable installed on Trinity during the Open Science period not having debug symbols (a LLNL policy for that type of network). We were mostly able to work around these code issues.

The SCR library development team encountered some issues on Trinity:

- SCR build process initially did not work on Cray - improved portability of SCR's build system
- Missing Date::Manip perl module – Trinity admin team installed
- Missing pdsh - installed a private build; would be helpful to add this as a system tool, as it could also be useful to others
- Missing Lustre on initial test bed – SCR uses flock which is not supported in NFS, NFS caching confused us for a while, but we found a way to work around it
- Trinity was the first SCR port that required cross compilation – extended SCR build system for cross-compilation
- Missing script to clean RAM disk between jobs – Trinity admin team installed
- MOAB limitation prevented DataWarp private, scratch – most natural fit for SCR, diverted our plans to jump to final stage
- MOAB is missing an API routine to query the remaining time in the job – this is critical for SCR to function properly, for now we have a work around

From a machine administration and bring-up perspective, perhaps the most interesting lesson for the SCR development team was the amount of time required to move from the test beds (especially Trinitite, Gadget) to the eventual system (Trinity). Many delays were incurred in testing on new test beds or on Trinity for features that were working on Trinitite. We frequently had to debug each time when working on a system incrementally more similar to Trinity. This applied both to the operating system and to the state of the DataWarp. We did not have sufficient time to fully complete and test our port to DataWarp.

On the positive side, the SCR performance data that we were able to glean reinforce our confidence that hierarchical checkpointing, combined with automatic restart, could be an approach to significant performance gains for production codes on the Trinity system. We will continue our efforts to provide the most robust set of hierarchical checkpointing options on the

Trinity system to take advantage of the new technologies, particularly DataWarp, available in that domain.

Total Time/Jobs and Composition of Jobs

The total number of jobs run during the Trinity Open Science period for our calculations was 385, and the total CPU time used for our calculations was 20,613,037.2 hours.

The number and size of jobs run during the Open Science period consisted of three types of calculations.

1. Particle convergence simulation suites of twelve simulations each using 1,152, 2,304, 4,608, 9,216, and 18,432 processors.
2. Parallel scalability investigation suites ranging from 1,152 processors to 36,864 processors in factors of two.
3. Input/output scalability investigations suites ranging from 32 to 131,072 processors.

The weak scaling for our urban modeling user problem was excellent from 1,152 to 18,432 processors, with a maximum of 8% loss of efficiency. We observed a significant ~3X slowdown going from 18,432 to 36,864 processors but did not have time to explore this slowdown in detail.

Size of Data Generated

The Mercury particle convergence simulation suites generated approximately 1-2 TB each of data (output files, plot files, restart and graphic files, etc.), for a total of approximately 14 TB. The Mercury weak scaling studies each produced from approximately 100 GB to 185 GB, for a total of approximately 640 GB. The Mercury SCR library scalability simulations generated restart file sets ranging from approximately 23 GB up to approximately 2.6 TB.

In total, the Mercury Trinity Open Science work produced approximately 17-20 TB of data.

References

[1] P. S. Brantley, R. Bleile, S. A. Dawson, M. S. McKinley, M. J. O'Brien, M. Pozulp, R. J. Procassini, D. Richards, S. M. Sepke, D. E. Stevens, "Mercury User Guide: Version 5.2," LLNL-SM-560687 (Modification #10), Lawrence Livermore National Laboratory Report (2016).

[2] "Mercury Web Site," http://mercury.llnl.gov (2016).

[3] A. Moody, G. Bronevetsky, K. Mohror, B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System," LLNL-CONF-427742, Supercomputing 2010, New Orleans, LA, November 2010.

[4] "SCR Web Site," http://computation.llnl.gov/projects/scalable-checkpoint-restart-for-mpi (2016).