Optimizing Application I/O by Leveraging the Storage Hierarchy Using the Scalable Checkpoint Restart Library with a Monte Carlo Particle Transport Application on the Trinity Advanced Computing System

Michael M. Pozulp, Gregory B. Becker, Patrick S. Brantley, Shawn A. Dawson, Kathryn Mohror, Adam T. Moody, Matthew J. O'Brien
Lawrence Livermore National Laboratory, Livermore, CA, USA

SC16, Salt Lake City, UT
November 14-17, 2016

Abstract

This poster exhibits our experience using the Scalable Checkpoint Restart library (SCR) [1] to achieve I/O speedups during checkpoint and restart. We ran Lawrence Livermore National Laboratory's (LLNL) Monte Carlo particle transport code, Mercury [2], on Trinity [3] at Los Alamos National Laboratory (LANL). We performed a weak scaling study and observed speedups at 16 nodes and above, including a 30x maximum speedup at 4096 nodes. We benchmarked read performance by restarting from the checkpoints we wrote and observed speedups for 11 out of 12 node counts, including a 9x maximum speedup at 2048 nodes. Finally, we ran a user problem in which using SCR reduced median time-to-checkpoint by 20x. Our results show that leveraging the storage hierarchy is necessary for optimizing application I/O.

Introduction

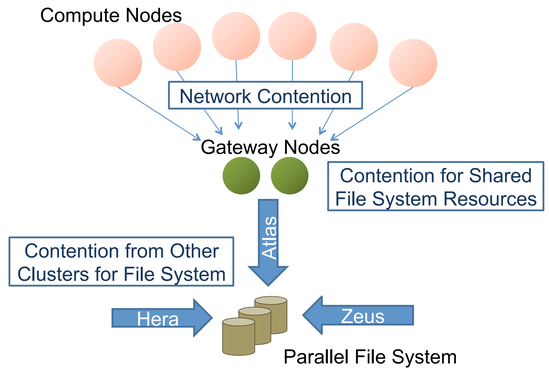

Figure 1. Network diagram showing parallel file system contention.

Writing checkpoints to the parallel file system (PFS) is very expensive due to multiple sources of contention (Fig. 1).

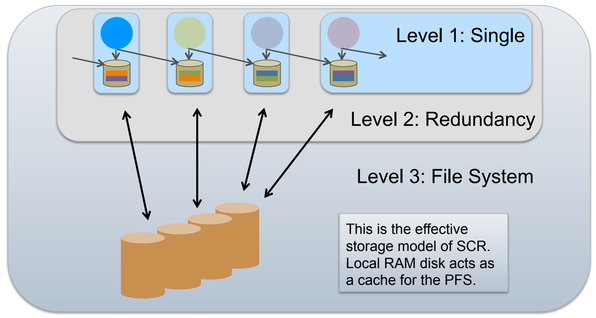

Figure 2. The three-tier storage hierarchy under investigation.

SCR presents a storage hierarchy (Fig. 2) that allows us to compare the cost of writing to the PFS versus RAM disk augmented with SCR's XOR redundancy scheme (Fig. 3).

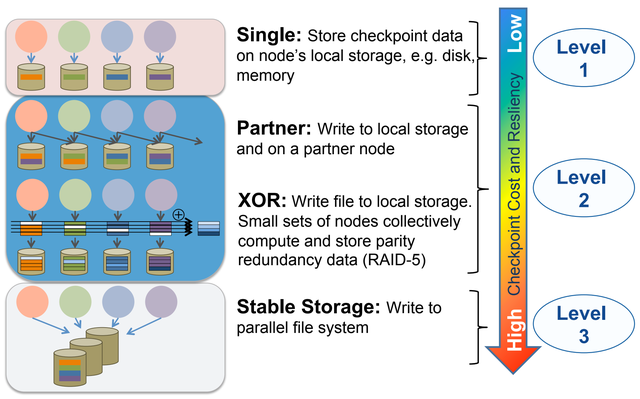

Figure 3. Reliability and cost in the three-tier storage hierarchy.

Introduction (cont.)

SCR research [1] has shown the PFS scaling well until the PFS bandwidth [4] is saturated, while RAM disk scales indefinitely (Fig. 4).

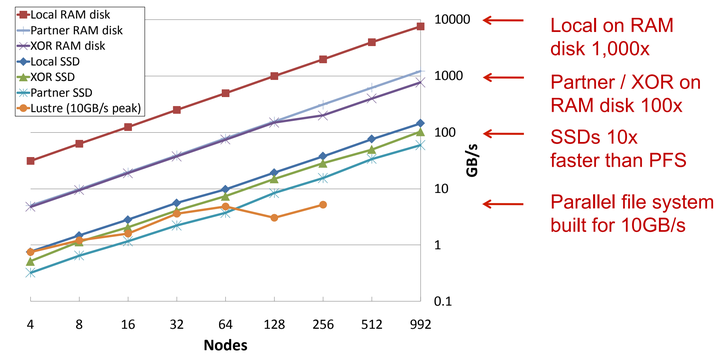

Figure 4. Checkpoint bandwidth vs. node count, SCR scaling study.

Weak Scaling Study

We ran Mercury using 32 MPI ranks per node, 1 rank per core, with and without SCR RAM disk checkpoint caching enabled. Each rank checkpoints approximately 20 MB using an N-N I/O pattern.

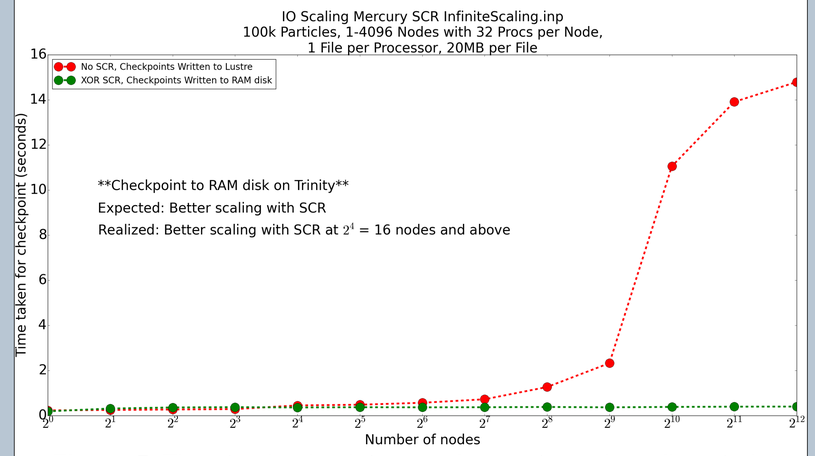

Figure 5. Time-to-checkpoint vs. node count, weak scaling study.

Writing checkpoints to RAM disk proved faster than writing to the PFS at 16 nodes and above, including a 30x maximum speedup at 4096 nodes (Fig. 5). Restarting from RAM disk produced speedups when compared with the PFS for 11 out of 12 node counts, including a 9x maximum speedup at 2048 nodes (Fig. 6).

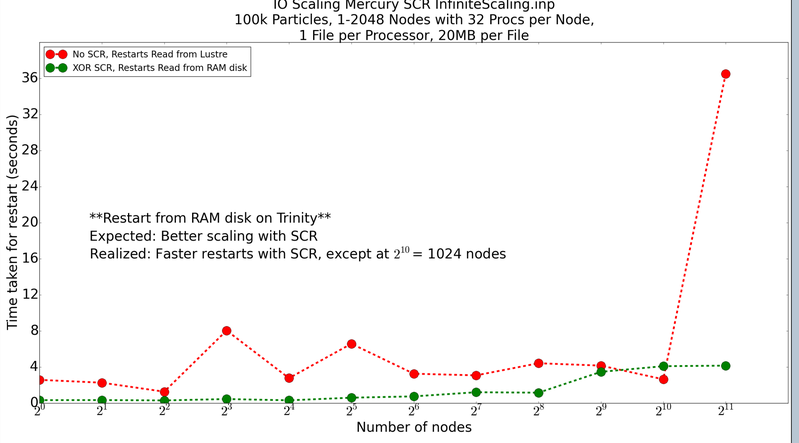

Figure 6. Time-to-restart vs. node count, weak scaling study.

"XOR SCR" in the legend of Fig. 5 and Fig. 6 indicates that SCR was used with the "XOR" redundancy scheme [5]. The PFS and RAM disk scaling that we observed is consistent with SCR research [1].

User Problem

We ran an urban modeling problem (Fig. 8) in Mercury to study the impact of I/O speedups in a production setting. We ran on 36 nodes using 32 MPI ranks per node, 1 rank per core. We wrote 1 checkpoint every 10 simulation cycles, yielding 18 total checkpoints (Fig. 7).

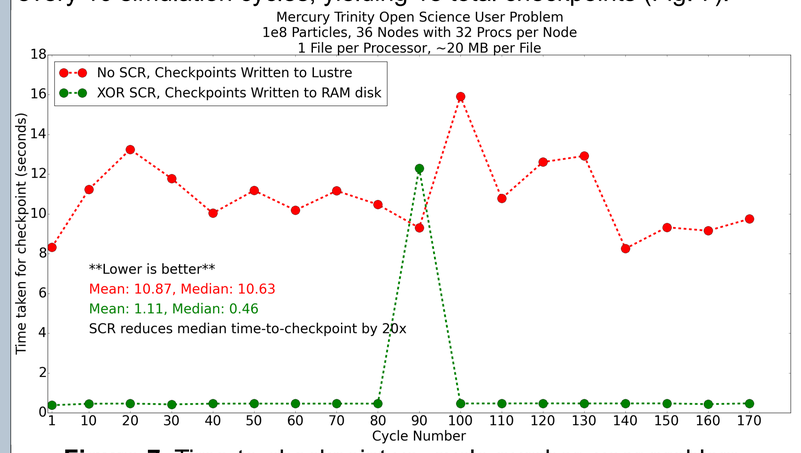

Figure 7. Time-to-checkpoint vs. cycle number, user problem.

When writing checkpoints to RAM disk instead of the PFS, we observed a median time-to-checkpoint speedup of 20x. For resiliency, SCR wrote to RAM disk and also flushed the 10th checkpoint to the PFS at cycle 90. This operation did not substantially exceed the average PFS checkpoint time without SCR.

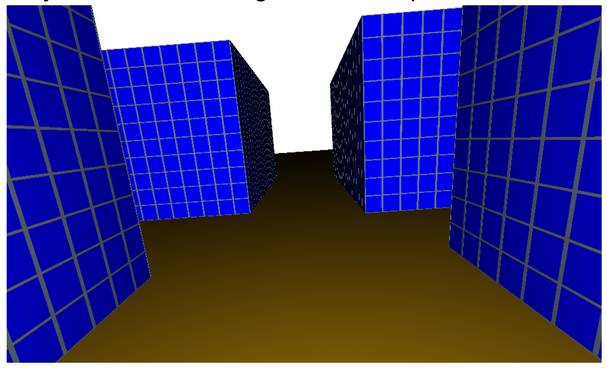

Figure 8. Example urban model buildings [6] viewed with Mercury.

Conclusion

Exploring alternatives to the parallel file system is a necessary path towards optimal application I/O. We realized significant speedups on RAM disk, peaking at 30x for writing a checkpoint and at 9x for restarting from a checkpoint. In the user problem, we reduced median time-to-checkpoint by 20x and reduced time-to-solution in a production Mercury run. SCR provided an abstraction layer that will allow us to continue to explore hierarchical checkpointing without additional porting effort. Our next step will be to investigate the I/O performance of Mercury on Trinity's burst buffers.

References

[1] Adam Moody, Greg Bronevetsky, Kathryn Mohror, Bronis R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System," LLNL-CONF-427742, Supercomputing 2010, New Orleans, LA, November 2010.
[2] Brantley, P. S., R. C. Bleile, S. A. Dawson, M. S. McKinley, M. J. O'Brien, M. Pozulp, R. J. Procassini, D. Richards, S. M. Sepke, and D. E. Stevens, "Mercury User Guide: Version 5.2," LLNL-SM-560687 (Modification #10), Lawrence Livermore National Laboratory Report (2016).
[3] Trinity Advanced Technology System. Web Page. Accessed Thu Apr 14, 2016. http://www.lanl.gov/projects/trinity/
[4] About the Lustre File System. Web Page. Accessed Tue Jul 5, 2016. http://lustre.org/about/
[5] SCR Users Guide. Web Page. Accessed Thu Apr 14, 2016. https://computation.llnl.gov/project/scr/files/scr UsersGuide v1.1.8.pdf
[6] Kevin Kramer, Applied Research Associates, Inc., personal communication (2013)

LLNL-POST-698037

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