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, and Matthew J. O'Brien
Lawrence Livermore National Laboratory

***Abstract*—The poster accompanying this summary 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 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.**

I. Introduction

Writing checkpoints to the parallel file system (PFS) is very expensive due to multiple sources of contention. Compute nodes sending data to I/O gateway nodes compete with I/O and communication traffic on the network. Gateway nodes sending data to the PFS compete with I/O traffic from gateway nodes that are connected to other clusters. The Scalable Checkpoint Restart library (SCR) [1] presents a storage hierarchy that reduces checkpoint cost by providing alternatives to the PFS. The storage hierarchy on Trinity at Los Alamos National Laboratory (LANL) [3] has four levels of interest to applications writing checkpoints. They are

1) Node-local storage using RAM disk
2) Node-local storage augmented by one of SCR's redundancy schemes [4]
3) Burst buffer storage using a pool of solid-state drives
4) PFS storage using Lustre

Most applications with which we are familiar restrict their I/O to level 4. One reason is because the PFS is reliable. Sophisticated data storage schemes are employed on the PFS to reduce the likelihood of unrecoverable data loss to a minimum. Another reason is programmability. No special coding is necessary for applications to run on a PFS instead of on a serial file system. The cost of these advantages is performance. When the PFS is used by many applications simultaneously, there is contention for I/O bandwidth that can hurt application I/O performance. Since the bandwidth of the PFS is not unlimited, even the performance of a single application with exclusive use of the PFS is subject to I/O scalability issues. Applications that encounter I/O contention or non-scalability must consider alternatives to the PFS if optimal I/O is desired.

SCR research has confirmed this analysis by showing that the PFS scales well until the PFS bandwidth is saturated. A node-local alternative is RAM disk, which was shown to scale indefinitely. SCR also provides redundancy schemes and an abstraction layer for reliability and programmability [1].

We are interested in using SCR to leverage this storage hierarchy to optimize I/O performance of Mercury [2], Lawrence Livermore National Laboratory's (LLNL) Monte Carlo particle transport code. Our test bed has been Trinity, where Mercury will be used for production runs in the near future. A simple way to achieve results with reduced statistical variation in Mercury is to simulate more particles. One cost of simulations involving more particles is increased checkpoint size, because Mercury needs the state of every particle that it was tracking at the time of a checkpoint in order to restart the simulation from that checkpoint.

For this first investigation of I/O performance on Trinity, we sought to compare levels 2 and 4 of the storage hierarchy. That is, we used SCR to compare the cost of writing to the PFS versus RAM disk augmented with SCR's XOR redundancy scheme, which is the default operating mode provided by SCR [4]. Since SCR research has shown the PFS scaling well until the PFS bandwidth is saturated, while RAM disk scales indefinitely, we expect RAM disk to outperform the PFS at sufficient scale.

II. Weak Scaling Study

We investigated SCR by performing a weak scaling study of I/O performance using Mercury on LANL's Trinity machine. We ran Mercury with 32 MPI ranks per node, 1 rank per core, with and without SCR RAM disk checkpoint caching enabled.

Each rank wrote 20 MB using an N-N I/O pattern. 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. 1).

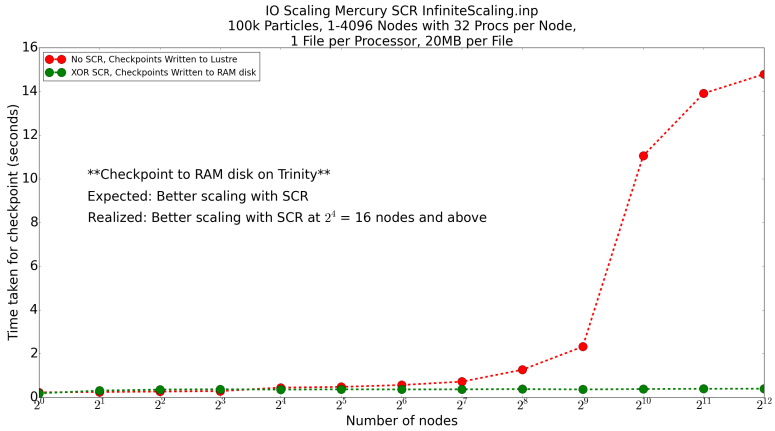

Fig. 1. Time-to-checkpoint versus node count for weak scaling study.

In both Fig. 1 and Fig. 2, "XOR SCR" in the legend indicates that SCR was used with the "XOR" redundancy scheme [4]. "InfiniteScaling.inp" in the title of Fig. 1 refers to the weak scaling problem run by Mercury. In this problem, P particles per processor are tracked through one of N evenly-sized sections of material. We checkpoint immediately, while each of N processors still has exactly P particles in its section of material. Running $P = 100000$ yielded 20 MB per rank.

The PFS and RAM disk scaling that we observed is consistent with SCR research [1]. Time-to-checkpoint on RAM disk appears to be constant with respect to checkpoint size, a perfect scaling result. Time-to-checkpoint on the PFS appears constant up to a point, after which it increases with checkpoint size, a poor scaling result. The checkpoint size after which the PFS is no longer scalable corresponds to the bandwidth limit of the PFS at the time that we were performing the study.

We studied read performance by restarting from the checkpoints we wrote. RAM disk produced speedups when compared to the PFS for 11 out of 12 node counts, including a 9x maximum speedup at 2048 nodes (see poster).

III. User Problem

We investigated the impact of using SCR to perform multi-level checkpointing in a production setting. The problem we selected is representative of a typical user workload for Mercury on Trinity. In the user problem we employed the same 20 MB per-rank write size and N-N I/O pattern that we used in our weak scaling study. 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 2).

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. Let d_r be the cost of this operation, calculated as the percent difference between the

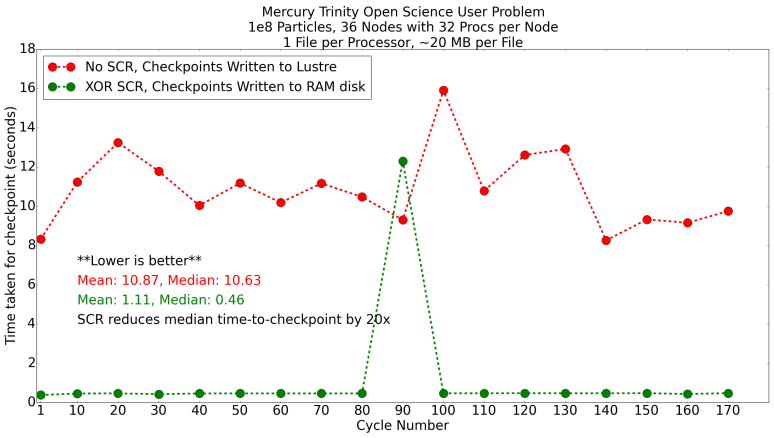

Fig. 2. Time-to-checkpoint versus cycle number for user problem.

average time-to-checkpoint without SCR, t_{avg}, and time-to-checkpoint with SCR at cycle 90, t_{scr90} (1).

$$d_r = \frac{|t_{avg} - t_{scr90}|}{t_{avg}} \cdot 100\% = \frac{|10.3 - 12.3|}{10.3} \cdot 100\% = 19.4\% \quad (1)$$

We expect d_r to disappear as checkpoint size increases because t_{avg} will increase while $t_{avg} - t_{scr90}$ stays constant. This is for the same reason that we used SCR in the first place. Namely, the PFS scales well only until its bandwidth is saturated, while RAM disk scales indefinitely [1].

IV. Conclusion

Exploring alternatives to the PFS 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 it. In a user problem typical of production runs that we expect on Trinity, we reduced median time-to-checkpoint by 20x. SCR provided us with an abstraction layer to leverage the storage hierarchy that will allow us to continue to explore hierarchical I/O at different levels without additional porting effort. Our next step will be to investigate the I/O performance of Mercury on Trinity's burst buffer.

Acknowledgment

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-ABS-697958).

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] SCR Users Guide. Web Page. Accessed Thu Apr 14, 2016. https://computation.llnl.gov/project/scr/files/scr_UsersGuide_v1.1.8.pdf