SANDIA REPORT

SAND2012-1087
Unlimited Release
Printed February 2012

Using the Sirocco File System for High-Bandwidth Checkpoints

Matthew L. Curry, Ruth Klundt, H. Lee Ward

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2012-1087
Unlimited Release
Printed February 2012

Using the Sirocco File System for High-Bandwidth Checkpoints

Matthew L. Curry, Ruth Klundt, and H. Lee Ward
Scalable System Software Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1319
{mlcurry, rklundt, lee}@sandia.gov

Abstract

The Sirocco File System, a file system for exascale under active development, is designed to allow the storage software to maximize quality of service through increased flexibility and local decision-making. By allowing the storage system to manage a range of storage targets that have varying speeds and capacities, the system can increase the speed and surety of storage to the application. We instrument CTH to use a group of RAM-based Sirocco storage servers allocated within the job as a high-performance storage tier to accept checkpoints, allowing computation to potentially continue asynchronously of checkpoint migration to slower, more permanent storage. The result is a 10-60x speedup in constructing and moving checkpoint data from the compute nodes.

Introduction and Motivation

The Sirocco file system is a peer-to-peer-inspired file system for exascale under active development. It is part of a multi-year effort among Sandia National Laboratories, the University of Alabama at Birmingham, Clemson University, and Argonne National Laboratory. The motivation behind Sirocco is that, as HPC systems (and their storage systems) grow, increasing component counts will imply a growth in the number of device failures. The storage system will be in some sort of failure mode at all times, requiring it to adapt dynamically to its own health. Peer-to-peer systems are well known for their ability to provide resilience by handling client turnover [7], and provide a ripe area of exploration to improve large scale storage systems.

Resilience and performance concerns influence the design of Sirocco in a number of ways. First, data should be automatically migrated and replicated through the system to maintain data integrity. For example, if data is replicated among three servers, and one fails, data may be replicated immediately without central coordination to another server or set of servers, ensuring that at least three copies remain. Second, data may be relocated or replicated for performance reasons, including to and from a set of compute nodes that hold data in RAM as the fastest tier in a caching model. Finally, clients may write data to any server that provides the client with high quality of service quantified by bandwidth, reliability, I/O operations per second, latency, and so on.

Many DOE applications checkpoint frequently to defend against job failure. This is often a synchronous operation to disk-based storage, requiring all nodes to stop computing until the checkpoint has been placed in its ultimate location. Today's parallel file systems are not able to accept data at the speed compute nodes can produce it, so the application must spend significant time waiting for the checkpoint to complete. To improve performance, the storage system could recruit compute nodes to provide an intermediate fast store that can migrate checkpoints back to disk asynchronously, improving checkpoint performance for the job.

For this report, we demonstrate running Sirocco storage servers on compute nodes as a first tier of fast storage. We deploy the Sirocco nodes alongside a compute job running CTH, with CTH writing checkpoint data to Sirocco. This experiment demonstrates the promise of Sirocco: As development continues, this first tier will transfer the checkpoint data to slower, disk-based storage asynchronously as necessary, allowing bursty write patterns to experience higher performance than the underlying disk-based storage can provide.

Related Work

The tiered storage approach can be seen as a blend of several functionalities offered by other projects. These projects augment the underlying storage system to improve apparent performance or address shortcomings.

IOFSL [1] is an I/O forwarding layer that decreases the number of clients a file system must handle simultaneously. This is done by dedicating nodes to the task of accepting file system requests and performing them on behalf of the compute nodes. IOFSL does not provide asynchronous checkpoint transfer to the parallel file system, but operates synchronously.

SCR [5] is a library that provides a range of checkpoint strategies to applications. This includes having each node within a job checkpoint its data to the memory of a neighboring node, among other arrangements. SCR includes a notion of tiered checkpoints where only some checkpoints are stored to the parallel file system, while others remain resident in other storage tiers.

The PLFS-enabled burst buffer [3] is another piece of software that has similar goals. The burst buffers are special-purpose servers distributed throughout an HPC platform, e.g. one per compute rack. Each burst buffer server is filled with enough flash memory to accept a small number of checkpoints from nearby compute nodes. The data is asynchronously drained from the burst buffers to the parallel file system. PLFS [2] is used to maintain a consistent view of data regardless of its location.

The Sirocco Storage Server and Client

The Sirocco storage server provides an object-based storage API. Each object has a 128-bit identifier and a 64-bit address space. The storage server supports sparse objects and conflict resolution of objects via update identifiers. The API is RDMA-based, as the network layer (Single-Sided Messaging, or SSM [4]) is heavily influenced by the Portals 4.0 specification [6]. For this experiment, SSM used an MPI-based transport layer, but others are under active development.

No generally useful POSIX-like client exists for Sirocco. While a FUSE-based client has been demonstrated as a proof-of-concept, it is not meant for use by applications. Instead, a simple custom client was crafted for CTH that stores checkpoint data with a file per object, with a one megabyte local buffer to ensure that larger messages are sent to the storage server as needed. Because CTH's checkpoints have predictable names based on the iteration and rank of the process, a job ID and rank can be used to derive a unique object ID. A more friendly POSIX client for HPC is under development.

Instrumenting CTH

To give CTH access to Sirocco servers in the Cielo application environment, we launched extra processes within the CTH allocation specifically dedicated to Sirocco. Where CTH calls `MPI_Init`, we inserted a call to our own initialization routine that created a global communicator, splitting the nodes between a CTH group and a Sirocco group. The Sirocco processes were instructed to enter an event loop to satisfy storage requests, while the CTH nodes were released to participate in the compute job normally. A number of operations CTH performed over and within `MPI_COMM_WORLD` had to be changed to use the new communicator.

Other changes were less invasive. To have CTH use Sirocco's storage protocol, we leveraged syncio, an alternative I/O infrastructure already present in CTH. Syncio optionally coordinates I/O from nodes so that storage servers in certain previous platforms would not be overloaded from many simultaneous I/O requests from CTH. Syncio's API includes read, write, open, and close, creating a straightforward mapping of common I/O tasks. As Cielo does not require syncio functionality, we created simple replacements of those routines that would interact with the Sirocco storage servers directly. A small number of further changes had to be made to routines that were not written to use syncio.

We evenly distributed nodes running Sirocco processes throughout the CTH job. Specifically, we treat the nodes in the job in groups of five, where each group k contains the process IDs between $k \times 80$ and $(k+1) \times 80 - 1$, inclusive. The lowest 16 ranks, which are located in the first node of the group, run Sirocco server instances. The highest 64 ranks, which comprise the other four nodes of the group, run CTH processes.

The CTH processes are directly mapped to the Sirocco processes in the first node of its group. For example, if a Sirocco process in node n has rank $n \times 16 + i$, it will service the CTH processes found in ranks $(n+1) \times 16 + i$, $(n+2) \times 16 + i$, $(n+3) \times 16 + i$, and $(n+4) \times 16 + i$.

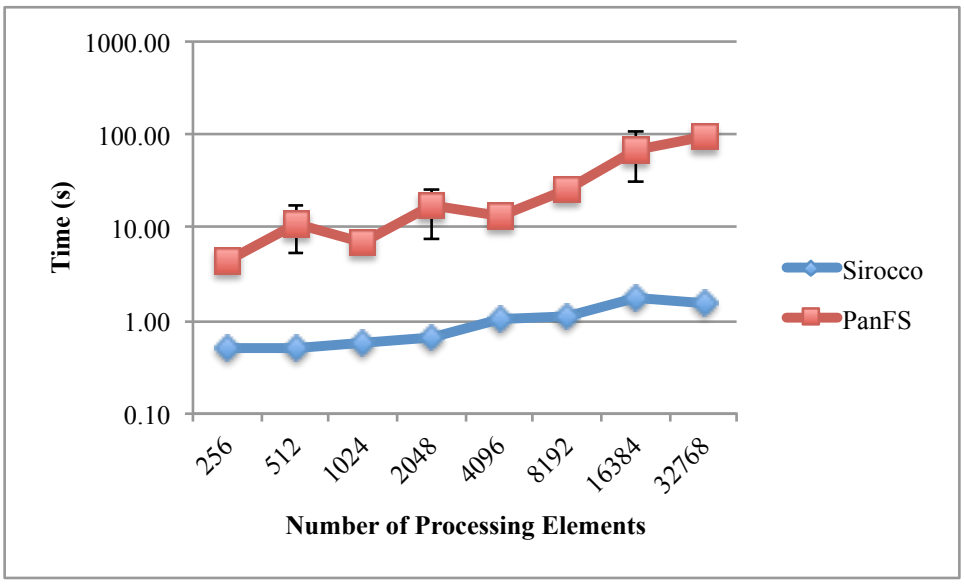

Figure 1. Average checkpoint time for Sirocco versus PanFS. All points include error bars indicating a 99% confidence interval, but may not be visible for data points that had little variance. Both axes use log values.

Application Impact

All tests were conducted on Cielo, a Cray XE6 that is used as a capability platform at Sandia National Laboratories and Los Alamos National Laboratory. The job was run with the shaped charge example problem at scales varying from 256 to 32,768 cores. Baseline runs were completed with CTH writing checkpoints directly to the 10PB Panasas storage, which uses PanFS as the file system. The Sirocco runs require extra cores for storage servers, so each Sirocco run included 25% more nodes than the CTH job required. For example, the 256 core job had 320 cores requested in the allocation, with 64 cores used exclusively as Sirocco storage servers. All tests were run through at least four checkpoints, and we recorded the average of the first four checkpoints. The time interval between checkpoints was specified to be small, but varied between thirty seconds to two minutes among different batches of runs.

Figure 1 gives the results of this experiment. Checkpoint performance was increased by at least 10x, with larger jobs demonstrating up to a 60x performance increase. One significant feature of Figure 1 is the variance experienced by each storage platform. These benchmarks were run while the machine was in general use, so it is likely that there was occasional significant contention for PanFS I/O services. The Sirocco partition, because it is dedicated to a single job, displayed almost no variance.

While these jobs used a rather large 25% increase in allocation of nodes per job, this was a decision made to ensure that each node had enough space available for checkpoint storage. CTH

compresses its checkpoints, causing some processes to write much more data than others. With a good mechanism for load balancing, and implementation of data migration capability, it may be possible to exploit a much smaller fraction of servers with good performance.

Conclusion

This demonstration of early Sirocco functionality shows a significant benefit for a real I/O workload, checkpointing, in a real application, CTH. By running Sirocco storage servers within a job as RAM-only stores, CTH was able to store checkpoints 10-60x faster than storing to PanFS, allowing the job to continue computing sooner. While this prototype did not include automatic data migration, the checkpoint was available to be pushed or pulled to disk-based storage as needed after the compute nodes continued computing. Future developments include the ability to dynamically spawn Sirocco nodes to absorb checkpoints, expanding this mechanism to other fast tiers of storage like flash memory, and sharing of dynamic Sirocco nodes between multiple jobs as needed.

References

[1] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. Scalable I/O forwarding framework for high-performance computing systems. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1 –10, September 2009.

[2] John Bent, Garth Gibson, Gary Grider, Ben McClelland, Paul Nowoczynski, James Nunez, Milo Polte, and Meghan Wingate. PLFS: A checkpoint filesystem for parallel applications. In *SC '09 Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 21:1–21:12, New York, NY, USA, November 2009. ACM.

[3] John Bent and Gary Grider. U.S. Department of Energy best practices workshop on file systems & archives: Usability at Los Alamos National Lab. In *Proceedings of the 5th DOE Workshop on HPC Best Practices: File Systems and Archives*, September 2011.

[4] Matthew S. Farmer, Anthony Skjellum, Matthew L. Curry, and H. Lee Ward. The design and implementation of SSM: A single-sided message passing library based on Portals 4.0. Technical Report UABCIS-TR-2012-01242012, Department of Computer and Information Sciences, University of Alabama at Birmingham, 115A Campbell Hall, 1300 University Blvd, Birmingham, Alabama 35294-1170, to appear.

[5] Adam Moody, Greg Bronevetsky, Kathryn Mohror, and Bronis R. de Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[6] Rolf Riesen, Ron Brightwell, Kevin Pedretti, Brian Barrett, Keith Underwood, Arthur B. Maccabe, and Trammell Hudson. The Portals 4.0 message passing interface. Technical Report SAND2008-2639, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, April 2008.

[7] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31:149–160, August 2001.

DISTRIBUTION:

1 MS 0899 RIM-Reports Management, 9532 (electronic copy)

Sandia National Laboratories