Parallel Log Structured File System Update

Meghan (Wingate) McClelland, HPC-5;
Gary A. Grider, HPC-DO; Adam Manzanares, HPC-5;
John Bent, EMC Corporation

To improve the checkpoint bandwidth of critical applications at LANL we developed the Parallel Log Structured File System (PLFS)[1]. PLFS is a transformative I/O middleware layer placed within our software stack that transparently rearranges a challenging workload, a concurrently written single shared file, into an optimized workload, non concurrently written non-shared component pieces. This reorganized I/O has made write size a non-issue and improved checkpoint performance by orders of magnitude measured to be as much as 150x with improvements in write, read, and meta data performance of our I/O workloads. Under a CRADA with EMC Corporation, LANL and EMC have successfully demonstrated a prototype "burst buffer" and are working together to further enhance, design, build, test, and deploy PLFS. Future work for PLFS includes integration with the Scalable Checkpoint/Restart (SCR) Library, further improving metadata rates, and capturing semantic information about the data streams.

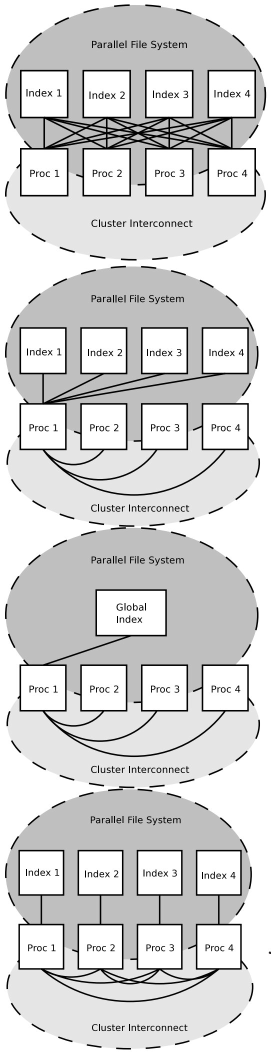


Fig. 1. Index Aggregation Techniques. The figures below represent the workloads generated on the parallel file system and cluster interconnect by PLFS using the original design and our three solutions to improve the read bandwidth of PLFS.

The original architecture of the Parallel Log Structured File System (PLFS) was heavily write optimized and limited read performance. We quickly noticed that with increasing process counts our effective read bandwidth was suffering due to read open times so we developed several techniques to address this problem (Fig 1), which are to aggregate the global index on

- read open with one process and broadcast this result to every other process (Index Broadcast);
- write close, and on read open broadcast the results of the aggregation from one process to all processes (Index Flatten); and
- read open leveraging all processes (Parallel Index Read).

All three share the common property of reducing the number of input/output (I/O) operations that must be conducted by the parallel file system. During a checkpoint read the compute and high speed interconnect resources of cluster machines are largely idle. Our solutions leverage both of these resources and to some degree reduce the amount of concurrent access to files as compared to the original design of PLFS. Bandwidth improvements and faster open and close times are realized with all three methods (Fig. 2).

Metadata management is a difficult challenge for many current parallel file systems. As we move into exascale-class computing, compute systems will have increased component counts which will increase metadata workloads, exacerbating the metadata challenges that we currently face. The flexibility granted by the PLFS layer allows us to alleviate metadata challenges by spreading the I/O workload over multiple physical locations (hashing), turning independent metadata servers into a federated system of servers which improces metadata performance (Fig. 3), and spreading workloads over multiple directories solving the problem of massive-scale file creation in single directories. This increased metadata performance has been demonstrated across two classes of workloads that together represent the checkpoint point I/O patterns of a large class of applications.

The promising effectiveness and flexibility of PLFS has made it necessary to develop PLFS into production software; a regression test suite and quality assurance (QA) procedures have led to improvements in robustness, performance (including archiving and restarting with varying number of processors), and new features. PLFS is in the process of being deployed on current production systems and our plans for exascale I/O include the use of PLFS.

Recently, LANL announced the signing of a new umbrella CRADA with EMC Corporation. As part of this CRADA, LANL and EMC are working together to enhance, design, build, test, and deploy PLFS. This collaboration is focused on support for the Department of Energy's exascale initiative and other data-intensive programs, and is aimed at boosting high-performance computing capability to ensure efficient

For more information contact Meghan W. McClelland at meghan@lanl.gov

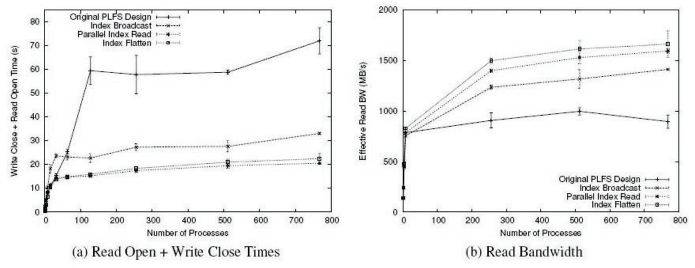
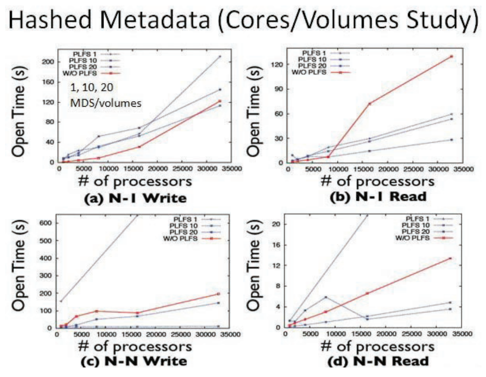


Fig. 2. The graph on the left illustrates the time taken to aggregate the global index of a PLFS file with lower times representing better performance. Note: the write close times are included in the graph because the Index Flatten technique aggregates the global index on the close of a newly written file. The graph on the right illustrates the effective read bandwidth of the original PLFS design and our read optimizations.

Fig. 3. Graphs of results from a hashed metadata study. In most cases, multiple metadata servers (hashing) dramatically improve file open time. Note that smaller open times are better, so lower lines are better performing for these graphs.



resource utilization on the largest supercomputers in the world.

The first deliverable for this CRADA was the demonstration at the 2011 Supercomputing Conference of a prototype “burst-buffer” storage stack (Fig. 4). Economic projections into the exascale era dictate that the storage stack be re-engineered to incorporate an intermediary layer between the compute nodes and the disk-based scratch storage space. This prototype burst-buffer system was a surprisingly effective exploration within this new architectural space. By accomplishing a complicated integration of both new hardware and software elements in a very short timeframe, LANL and EMC demonstrated their ability to work together closely and effectively.

Using a wind-turbine simulation and visualization workload, the demonstration showed much faster data dump times as well as the ability to co-process the visualization with the simulation, as opposed to the current post-processing model. The very encouraging performance results of this demonstration were of great interest to the community and necessitate more investigation of the burst-buffer concept for exascale computing.

The PLFS project is now looking to further improve metadata rates. Large metadata workloads are created both by user N-N workloads and also by PLFS as it transforms user N-1 workloads. For an exascale system, metadata ingest rates will become a horrible bottleneck as N approaches one billion. Current efforts are ongoing to aggregate multiple logical users’ files into a smaller number of physical files to reduce the metadata burden. In addition, the internal PLFS metadata architecture is currently being re-engineered for exascale workloads. One method currently being explored is to use the speed of the compute interconnect workload to shuffle small amounts of data to transform per-process heterogeneous workloads into homogenous ones. This regularity will allow a huge reduction of PLFS’s internal metadata.

As part of a tri-lab effort, PLFS is currently being integrated with Scalable Checkpoint/Restart (SCR) [2], a checkpoint in-memory system for N-N codes developed at LLNL.

Finally, the most ambitious and important PLFS project currently ongoing is to capture semantic information about the data streams. This will allow PLFS to know not just where data is stored but what data is stored where. With this knowledge, analysis and visualization workloads can query high-level semantic data knowledge and avoid costly data filtering which can reduce the amount of data read by orders of magnitude. This outcome is related to active storage projects in the past and can enable scientific workloads to benefit from programming models similar to map-reduce.

PLFS is open source software available from http://sourceforge.net/projects/plfs

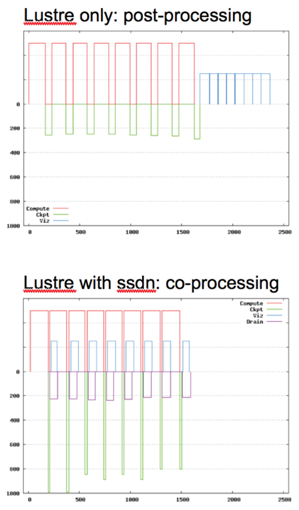


Fig. 4. Graphs of prototype “burst-buffer” demonstration at the 2011 Supercomputing Conference show much faster data dump times as well as the ability to co-process the visualization with the simulation (bottom graph), as opposed to the current post-processing model (top graph).

Special Thanks
Project contributions by the HPC-5 I/O team, the HPC-3 HPC Tools team, and Garth Gibson.

[1] Bent, J. et al., *Proc Conf HPC Networking Storage Anal,* Article 21; DOI=10.1145/1654059.1654081 http://doi.acm.org/10.1145/1654059.16540812009 (2009).
[2] Moody, A. et al., *Proc 2010 ACM/IEEE Int Conf HPC Networking Storage Anal,* **1** (2010).

Funding Acknowledgments
DOE NNSA, Advanced Simulation and Computing Program; National Science Foundation