# Parallel Log Structured File System (PLFS)

**Gary Grider, HPC-DO;**
**John Bent, EMC Corporation;**
**Chuck Cranor, Carnegie Mellon University;**
**Jun He, New Mexico Consortium;**
**Aaron Torres, HPC-3;**
**Meghan McClelland, HPC-5;**
**Brett Kettering, HPC-5**

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 storage stack. It transforms a concurrently written single shared file into non-shared component pieces. This reorganized I/O has made write size a non-issue and improved checkpoint performance by orders of magnitude, meeting the project's L2 milestone to show increased performance for checkpointing with LANL codes. LANL is working together with EMC under an umbrella Cooperative Research and Development Agreement (CRADA) to further enhance, design, build, test, and deploy PLFS. PLFS has been integrated with multiple types of storage systems, including cloud storage, and has shown improvements in file storage sizes and metadata rates.

A single shared file written concurrently by a large number of processes and non-optimal write sizes is a challenging workload for the current generation of parallel file systems. Despite severe bandwidth limitations, single shared file checkpointing continues to be used heavily in the high-performance computing (HPC) community because it is convenient from a user standpoint. PLFS transparently rearranges these challenging user I/O patterns into patterns optimized for the parallel files system. 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 150× with improvements in write, read, and metadata performance of our I/O workloads.

As part of the LANL umbrella CRADA with EMC Corporation, LANL and EMC are working to design, enhance, build, test, and deploy PLFS. This collaboration is focused on support for the DOE's exascale initiative and other data-intensive programs and is aimed at boosting HPC capability to ensure efficient resource utilization on the largest supercomputers in the world.

Using PLFS for data placement, LANL and EMC have prototyped a "burst buffer" storage stack. 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. Within this intermediate layer, applications may conduct in-transit analysis and other data-mining operations, allowing for efficiencies in results collection and file archiving. LANL and EMC have shown overall time to problem solution reduced by 23%, with I/O to the burst buffer 3.5–4× faster than directly to Lustre, an open-source parallel file system. A major modification has been the addition of an I/O Forwarding Scalability Layer (I/OFSL), a transport mechanism for a scalable I/O layer that scalability models suggest will be required at exascale.

An increasing number of clusters are being configured for data analytics using the Apache Hadoop open-source version of the Google Internet services tool suite (Google File System, BigTable, etc.) [2,3]. Because HPC and Internet services analytics can both be big data and big compute applications families, it is desirable to be able to mix and match applications on cluster infrastructure. Prior research has demonstrated that Hadoop applications can run efficiently on HPC cluster infrastructure [4]. However, cloud storage systems–such as the Hadoop Distributed Filesystem (HDFS)[5]–are not POSIX-based and do not support multiple concurrent writers to a file. In order to enable the convergence of HPC and cloud computing on the same platform, we adapted the PLFS to enable HPC applications to be able to read and write data from the HDFS cloud storage subsystem. Our enhanced version of PLFS provides HPC applications with the ability to concurrently write from multiple compute nodes into a single file stored in HDFS, thus allowing HPC applications to checkpoint. This work paved the way for PLFS to be able to store to multiple types of backends, increasing its overall versatility and usefulness. Research into data and data log compression capabilities in storage backends is showing very promising results and is planned for a future PLFS release.

Current approaches for metadata handling cannot scale to exaflop supercomputers due to the large overhead of creating and reassembling the metadata. We have developed and evaluated algorithms by which patterns in the PLFS metadata can be discovered and then be used to replace the current metadata [6]. Our evaluation shows that these

**For more information contact Brett Kettering at brettk@lanl.gov.**

**Funding Acknowledgments**
National Science Foundation; DOE, NNSA, Advanced Simulation and Computing Program; LANL National Security Education Center

patterns reduce the size of the metadata by several orders of magnitude, the performance of writes by up to 40%, and the performance of reads by up to 480%. These improvements can allow current checkpointing models to survive the transition from petascale to exascale. Another feature improvement for PLFS is write buffering–the capability of PLFS to buffer writes, reducing the overhead of making many system calls and performing small writes. Flat file mode reduces the number of underlying files and directories created for N-N workloads, decreasing the overall metadata workload and thereby increasing performance.

PLFS met the project's L2 milestones, demonstrating up to 76% increased performance over I/O directly to the file system with LANL codes. PLFS has been shown to work on real problems for real production simulation applications at LANL that use message passing interface (MPI)/IO for N-1 I/O workloads. We have shown that for a benchmark application configured to produce a maximum write bandwidth workload, PLFS does not adversely affect N-N performance. PLFS shows substantial performance improvement over MPI/IO for N-1 workloads for these production simulation codes. Furthermore, PLFS is a general solution, like MPI/IO, that can be used by any application. PLFS is capable of further improving I/O performance by aggregating multiple file systems into a larger virtual file system when the application I/O load justifies the larger file system capabilities. PLFS is integrated into the LANL production system parallel tools installation process and is available for use on LANL HPC platforms.

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

**Special Thanks**
Project contributions by the HPC-5 I/O team, the HPC-3 HPC Tools team, Garth Gibson, EMC Corporation Engineering Team, David Knaak, William Tucker, and Steve Oyanagi.

[1] Bent, J. et al., "PLFS: a Checkpoint Filesystem for Parallel Applications," *Proceedings Conference on High Performance Computing, Networking, Storage and Analysis,* 2009.

[2] Ghemawat, S. et al., *Operating Systems Review* **37**(5), 29 (2003).

[3] Chang, F. et al., "A Distributed Storage System for Structured Data," *USENIX OSDI 2006* (2006).

[4] Ananthanarayanan, R. et al., "Cloud Analytics: Do We Really Need to Reinvent the Storage Stack?" *Proceedings 1st USENIX Workshop on Hot Topics in Cloud Computing (HOTCLOUD '2009)* (2009).

[5] Shvachko, K. et al., "The Hadoop Distributed File System," *MSST 2010* (2010).

[6] He, J. et al., "Discovering Structure in Unstructured Data," *Proceedings 7th Workshop on Parallel Data Storage* (PDSW '12) (2012).