mpiFileUtils: A Parallel and Distributed Toolset for Managing Large Datasets

Danielle Sikich, Giuseppe Di Natale, Matthew LeGendre, Adam Moody
Lawrence Livermore National Laboratory
Livermore, California
sikich1,dinatale2,legendre1,moody20@llnl.gov

ABSTRACT

Modern HPC applications can generate massive datasets—on the order of hundreds of gigabytes or terabytes per run. Ensemble suites can contain thousands of runs, resulting in petabytes of information. It can take days to perform simple data-management operations on these datasets, such as copying data or changing file permissions. It is also logistically difficult to organize those data-management operations when they exceed the job time limits or uptime of a machine.

The core problem is that the applications that produce and consume data are highly-parallel and frequently optimized for HPC file systems such as Lustre. But the UNIX file management tools, like cp, rm, chmod, and cmp, are single-process solutions that are impractical on HPC datasets. These tools can bottleneck when dealing with millions of files and metadata operations or when handling the petabytes of data backing those files.

mpiFileUtils is an MPI-based suite of file management tools built on a common base library for handling large HPC datasets. It is the result of a multi-institutional collaboration between Los Alamos National Laboratory, Lawrence Livermore National Laboratory, Oak Ridge National Laboratory, DataDirect Networks, Red Hat, and Australian National University who have all contributed code and tools [1]. It contains distributed versions of the traditional cp, rm, cmp, chmod, and ls tools, as well as HPC-specific tools for managing striping and broadcasting files. These tools are similar to, and in many ways inspired by, the UNIX command-line tools, but they are implemented to use MPI-based parallelism and optimized to work with HPC file systems.

The mpiFileUtils project has four main objectives. First, there should be a library containing common functionality to simplify the development of new tools. We call this library `libmfu`. Common code and data structures that may benefit multiple tools should reside in the common library. As more capability accumulates over time, this library makes it easier and faster to create and extend tools.

Second, the tools should be scalable, such that a user can complete their task faster by running more processes, provided that there is sufficient parallelism inherent in the task at hand and provided that the backing file system performance permits it. In other words, the tools should not be the bottleneck in scaling.

Third, the tools should support parallel file systems commonly found in HPC centers. To start, we have focused on Lustre, GPFS, and NFS. Others can be added as necessary over time. The tools will detect and optimize for the underlying file systems where possible. This ensures that the tools will function effectively at many HPC sites so that users can rely on the tools being widely available.

Fourth, user interfaces across tools in the suite should be similar, and the tools should interoperate where possible. We aim to provide a consistent set of options and parameters for the tools. Additionally, the tools read and write a common file format to transfer state from one to another.

mpiFileUtils can provide speedups of up to nearly 60x compared to their single-process counterparts. A dataset was generated to test the performance of some of the most popular tools in the mpiFileUtils tool suite. For our experiments, data was both read and written to the same file system.

Table 1: The dataset was generated by fdtree [2] and it consists of one million files, each approximately 4 kilobytes in size, in a directory tree with a depth of four. The serial jobs for dcp and cp were submitted on a cluster with a twelve hour time limit. The speedups are all relative to the time taken for the corresponding Linux tool.

Tool	1 Proc	64 Procs	256 Procs
cp	12 hours (1x)		
dcp	12 hours (1x)	19 min. (37x)	14 min. (51x)
rsync	6 hours (1x)		
dcmp	12 hours (1x)	7 min. (51x)	6 min. (60x)
rm	15 min. (1x)		
drm	28min.	3 min. (5x)	5 min. (3x)

Find mpiFileUtils on github at:
`https://github.com/hpc/mpifileutils`

PDSW'17, November 2017, Denver, Colorado USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.475/123_4

ACM Reference Format:
Danielle Sikich, Giuseppe Di Natale, Matthew LeGendre, Adam Moody. 2017. mpiFileUtils: A Parallel and Distributed Toolset for Managing Large Datasets. In *Proceedings of Parallel Data Storage & Data Intensive Scalable Computing Systems, Denver, Colorado USA, November 2017 (PDSW'17),* 2 pages.
https://doi.org/10.475/123_4

1 ACKNOWLEDGEMENTS

This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory, Contract DE-AC52-07NA27344. LLNL-PROC-740981.

REFERENCES

[1] Jharrod LaFon, Satyajayant Misra, and John Bringhurst. 2012. On Distributed File Tree Walk of Parallel File Systems. In *SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM Press, Article No. 87.

[2] Mark Seager and Bill Loewe. 2016. fdtree. https://github.com/LLNL/fdtree. (2016).