Benchmarking Parallelized File Aggregation Tools for Large Scale Data Management

Tiffany Li, Craig Steffen, Ryan Chui, Roland Haas, Liudmila S. Mainzer

NCSA Genomics, University of Illinois at Urbana-Champaign

Introduction

Large-scale genomic data analyses has given rise to bottlenecks in data management due to the production of many small files. Existing file-archiving utilities, such as tar, are unable to efficiently package large datasets with upwards of multiple terabytes and hundreds of thousands of files. To create parallelized and multi-threaded alternatives, ParFu (parallel archiving file utility), MPItar, and ptgz (parallel tar gzip) were developed by the Blue Waters team and the NCSA Genomics team as efficient data management tools, with the ability to perform parallel archiving (and eventually extracting). Scalability was tested for each tool as a function of the number of ranks executed and stripe count on a Lustre filesystem. We used two datasets typically seen in genomic analyses to measure the effects of different file-size distributions. These tests suggest the best user parameters and subsequent costs for usage as efficient replacements of data-packaging tools.

ParFu

- MPI utility with Archival and Extraction modes and parallel data writing [4]
- **Ranks**: read the input data and write them into a single container file in parallel; larger number of ranks uses more compute resources
- Produces a tar compatible archive file

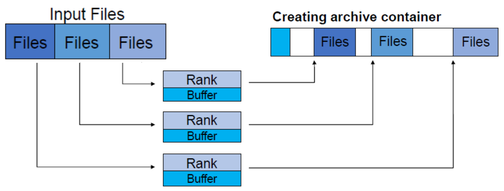

Figure 1. Diagram of ParFu when creating the archive. Different files within the original dataset are written by multiple ranks into the container file with a block size defined by user.

MPItar

- MPI archiving only utility with tar compatible output files and syntax [5]
- **Archival**: uses a master-worker approach where the master scans the filesystem and passes files to workers to reduce the latency of metadata operations
- **Retrieval**: direct access to individual files through a catalog file without having to scan the whole of the tar archive

ptgz

- Multi-threaded file archiving utility with Archival and Extraction modes utilizing gzip compression [6]
- **Compression**: single rank builds file list and assigns roughly equal work to each rank for balanced compression in the archive; uses default compression level; multiple ranks used for parallelized tar gzip and MPItar to build final file.

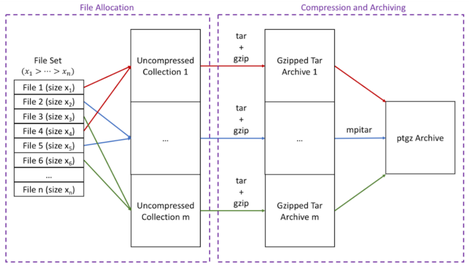

Figure 2. Flow diagram of ptgz compression. Files are separated into groups and are compressed independently into different tar files which are archived into a single tar file.

Methods

- Tested archive creation and extraction on Blue Waters and iForge supercomputers with very different filesystems
  - iForge GPFS with no striping capabilities
  - Blue Waters Lustre filesystem with striping abilities
- ParFu and MPItar were run with 16 ranks per node; ptgz was run with 1 rank per node with 32 threads at default gzip compression
- Datasets were tested against 1, 2, 4, 8, 10, 20, 30, and 40 nodes with stripe counts of 1, 4, 16, 32, and 64

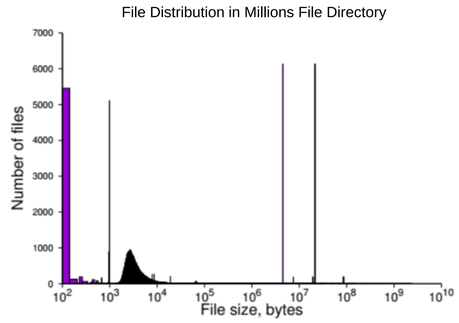

Figure 3. File size distribution of millions dataset with more of a uniform file distribution consisting of many small files and many large files

Input Datasets: included two kinds of file size distributions representative of typical genomic data to measure variations in performance

1. Set of several millions of small files; creates strain on storage due to fragmented distribution of files across tapes in Nearline
2. Results of the best practices GATK [1] variant calling pipeline run on synthetic Illumina sequencing reads generated by NEAT [3], to emulate sequencing at 30X depth, 100 nucleotides long, sequencing error rate of 0.1%, and random mutations at the rate of 0.05%; L-shaped file size distribution.

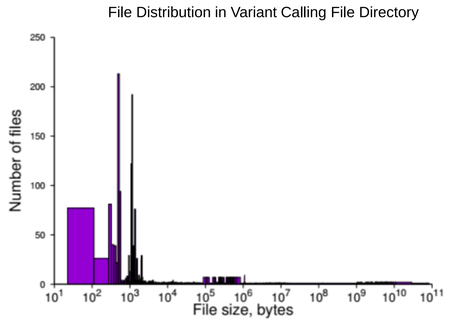

Figure 4. File size distribution of synthetic variant calling data with an L-shaped distribution consisting of many small files and some very large files

Results

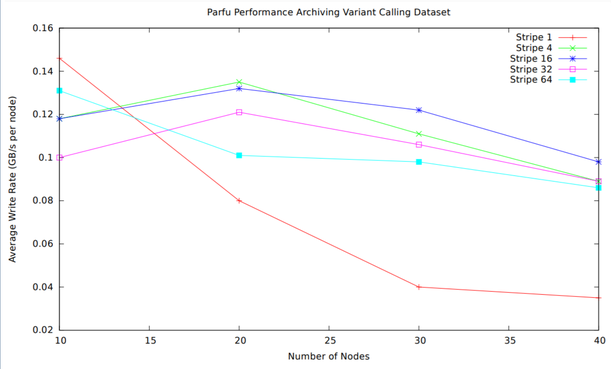

Figure 5. ParFu performance measured as GB/s per node on a dataset with an L-shaped file size distribution on Blue Waters for archival creations across stripe counts 1, 4, 16, 32, 64

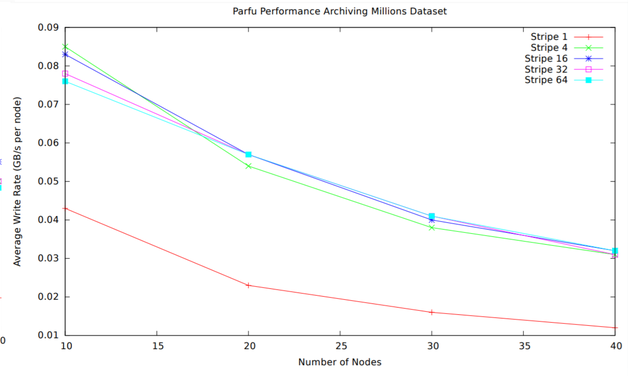

Figure 6. ParFu unstriped performance on a dataset with many small files on Blue Waters for archival creations across stripe counts 1, 4, 16, 32, and 64

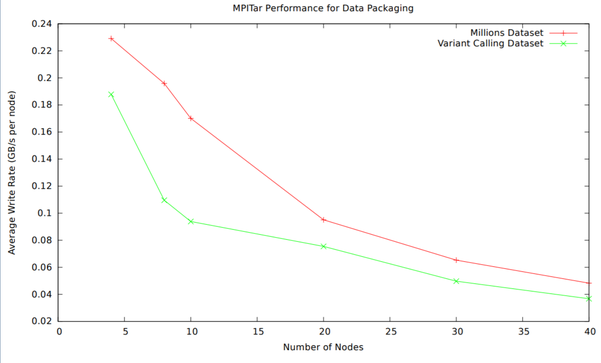

Figure 7. MPItar performance on both types of datasets: many small files and an L-shaped distribution, to compare performance as a function of the number of ranks

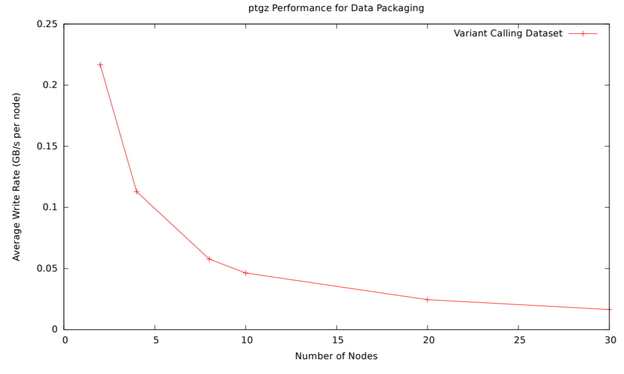

Figure 8. ptgz performance on the Variant Calling dataset with an L-shaped distribution as a function of the numbers of nodes

Discussion

- Preliminary testing showed ParFu is limited by the number of files and the overhead of opening and reading small files in the “millions” dataset
- Overall, both ParFu and ptgz demonstrated increased performance with increasing number of ranks, though not as well as expected
- Increasing the number of ranks will not result in decreased walltime on slower and smaller filesystems, possibly due to saturation
- Striping does not improve performance as much as predicted, possibly due to I/O limitations of the resident file directory [2]
- ParFu displays near-linear scalability on the “variant calling “ dataset, when it is not bound by the latency of reading small files
- Further testing should focus on the effects of changing the block size and number of blocks per group for ParFu, utilizing different compression settings for ptgz, and performance comparisons against popular file aggregation tools.

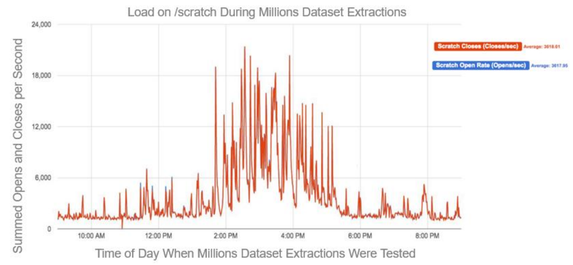

Figure 9. Load on /scratch for the day (4/22/2017) tests for Millions Dataset Extractions with a stripe count of 16 were submitted to Blue Waters. Peaks between 2:00PM and 4:00PM correspond with executed ParFu jobs on /scratch

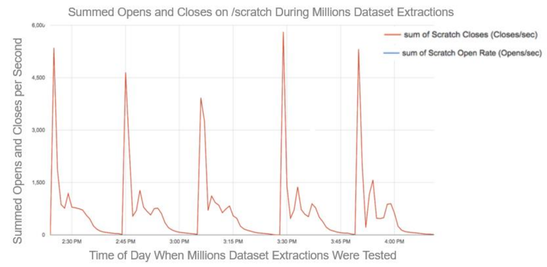

Figure 10. Summed opens and closes on /scratch for the day (4/22/2017) tests for Millions Dataset Extractions with a stripe count of 16 were submitted to Blue Waters. Peaks correspond to each job run and its performed reads and writes.

References

1. DePristo, Mark A., et al. "A framework for variation discovery and genotyping using next-generation DNA sequencing data." Nature genetics 43.5 (2011): 491-498.
2. Feiyi Wang, Veronica G. Vergara Larrea, and Dustin Levermanand Sarp Oral. 2016. FCP: A Fast and Scalable Data Copy Tool for High Performance Parallel File Systems (CUG ‘16). Cray User Group, 8. https://cug.org/proceedings/cug2016_proceedings/includes/files/pap142.pdf
3. https://github.com/zstephens/neat-genreads
4. https://github.com/ncsa/parfu_archive_tool
5. https://git.ncsa.illinois.edu/rhaas/mpitar
6. https://github.com/rchui/ptgz

Acknowledgements

The authors are grateful to the Blue Waters team for their assistance with collecting and interpreting the data for this project. We would like to thank especially Galen Arnold, Jeremy Enos, and Gregory Bauer for their discussions. ParFu has been developed by Roland Haas and Craig Steffen as part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (OCI-0725070 and ACI-1238993) and the state of Illinois. It has been released under the University of Illinois open-source license and is available at https://github.com/ncsa/parfu_archive_tool. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.