Benchmarking Parallelized File Aggregation Tools for Large Scale Data Management

Extended Abstract

T. Li
National Center for Supercomputing Applications
Urbana, Illinois
tli54@illinois.edu

C. Steffen
National Center for Supercomputing Applications
Urbana, Illinois
csteffen@ncsa.illinois.edu

R. Chui
National Center for Supercomputing Applications
Urbana, Illinois
rchui2@illinois.edu

R. Haas
National Center for Supercomputing Applications
Urbana, Illinois
rhaas@ncsa.illinois.edu

L.S. Mainzer
National Center for Supercomputing Applications
Urbana, Illinois
lmainzer@illinois.edu

ABSTRACT

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.

CCS CONCEPTS

• **Applied computing** → **Computational genomics**; *Bioinformatics*;

KEYWORDS

Data management, personalized medicine, parallel architecture

ACM Reference format:
T. Li, C. Steffen, R. Chui, R. Haas, and L.S. Mainzer. 2017. Benchmarking Parallelized File Aggregation Tools for Large Scale Data Management. In *Proceedings of Supercomputing Conference, Denver, Colorado USA, November 2017 (SUPERCOMPUTING'17),* 3 pages.
https://doi.org/10.475/123_4

SUPERCOMPUTING'17, November 2017, Denver, Colorado USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Development of genomic analyses has enabled integration of personalized medicine into healthcare. Advanced computing also now permits simultaneous large-scale genomic analyses on hundreds of genomes, making genome sequencing an effective tool for disease diagnosis and prevention [1]. The resulting datasets are upwards of multiple terabytes, causing performance bottlenecks in data management and movement. As the production of data increases with technological advancements in high performance and cloud computing, the need to efficiently manage data becomes increasingly important [1] [6]. The challenges involved includes resource efficiency, validation, and security [1].

Current single-threaded data management tools (i.e. tar) are unable to efficiently package large datasets due to limitations in the speed of read/writes and the movement of data between the disks and tape drives [4]. To create fast, parallelized alternatives, ParFu (parallel archiving file utility)[1], MPItar[2], and ptgz (parallel tar gzip)[3] were developed by NCSA teams at Blue Waters and NCSA Genomics. We performed scalability analyses on the three tools, in relation to the number of ranks executed and stripe count on the filesystem, to measure scaling efficiency of the codes and optimize file system usage.

2 ARCHITECTURE AND IMPLEMENTATION

ParFu is an MPI utility with "creations" and "extractions" modes, currently without an option for compression. It makes a catalog of all the files to be archived, sorted by size, and that list is broadcast to all of the ranks in the ParFu process to minimize the inter-process communication. The MPI ranks read the input data and write them in parallel into a single container file. ParFu produces a tar-compatible archive file while gaining speed due to parallel data writing and efficiently utilizing disk via compact packaging of constituents into the archive.

MPItar uses MPI and a simple master-worker approach to have a single MPI rank scan the files to be included in the tar archive

[1] https://github.com/ncsa/parfu_archive_tool
[2] https://git.ncsa.illinois.edu/rhaas/mpitar
[3] https://github.com/rchui/ptgz

and assign them to a pool of workers. It is designed to limit the number of file metadata operations to, at most, one call to "stat" and one call to "open" per file. MPItar does not attempt to parallelize within a single file, or optimize the order in which files are written. To facilitate extraction of individual files from the archive, MPItar includes a catalog file that makes it possible to extract individual files without having to scan the full archive.

The third tool, ptgz, is a similar utility with compression. It uses multiple nodes to run simultaneous instances of tar-gzip and calls MPItar to package all gzipped blocks into a single archive. In order to load balance the work, threads use an algorithm that alternates between largest-to-smallest and smallest-to-largest assignment of blocks to be gzipped. Extractions use an index file for multi-node, multi-threaded extraction and unpacking of the tar files.

3 METHODS AND RESULTS

ParFu, MPItar, and ptgz were tested on the Blue Waters and iForge, two systems at NCSA with very different filesystems. iForge uses GPFS, which does not give users an ability to do file striping, as is possible with the Blue Waters Lustre filesystem. Scalability was measured as the relationship between the rate of data transfer per node (GB/s/node) and the number of nodes used during the packaging process. On Blue Waters we had the opportunity to test the effects of file system size (i.e. number of object storage targets, OSTs [5]) by placing input data sets on /projects (144 OST) and /scratch (1440 OST), while maintaining consistent stripe count and filesystem destination for the written data. To determine optimal performance, stripe counts of 1, 4, 16, 32, and 64 were used for both archiving and extracting. We used 2, 4, 8, 10, 20, 30, 40 nodes with 16 ranks per node for ParFu and MPItar, and 32 threads per node for ptgz. Comparative scalability tests for input file collections, number of nodes used, and tar compatibility were run for all tools. Performance results were baselined against tar to determine increased benefits for each of our three tools.

Input datasets used to determine ParFu, MPItar, and ptgz scalability included two kinds of file size distributions representative of typical genomic data to measure variations in performance.

(1) The "Millions" Dataset: several millions of small files (kB), representative of an output from a GWAS analysis[4]. Retrieval of a large number of many small files creates strains on Nearline storage, due to fragmented distribution across tapes.
(2) Variant Calling Dataset: results of the best practices GATK [3] variant calling pipeline run on a set of synthetic Illumina sequencing reads generated by NEAT [5] to emulate reads produced at 30X depth, 100 nucleotides long, at sequencing error rate of 0.1%, with mutations randomly inserted at the rate of 0.05%. The output data for such a workflow consists of many small files (<GB) and small number of large files (tens and hundreds of GB), representative of an L-shaped distribution. Transfer protocols for files are different for small files and large files, resulting in inefficient transfers for L-shaped distributions. Bundling the entire output folder into a single package allows to optimize the network transfer protocol.

[4]http://www.vital-it.ch/software/FastEpistasis
[5]https://github.com/zstephens/neat-genreads

Preliminary results suggest near-linear scalability for ParFu on L-shaped file size distribution, and better scalability for MPItar on flat distributions than L-shaped distributions. For unstriped performances at 40 nodes, ParFu is 145 times faster than tar when packaging the Variant Calling dataset, and is 16 times faster than tar when packaging the Millions dataset. Overall, increasing number of ranks increases performance for all tools. There is possible saturation of ranks by the very large number of small files present in some datasets as well as I/O limitations of the resident file directory that limit scalability for other datasets. ParFu especially is limited by the number of small files, when packaging the Millions dataset, there is a possible oversaturation of resources resulting from the latency of opening and reading a small file. In contrast to the Millions dataset, ParFu displays near-linear scalability while packaging datasets with larger files. MPItar results demonstrate better scalability for the Millions dataset's flat distribution but worse for Variant Calling's L-shaped distribution. ptgz demonstrates similar scalability as MPItar for Variant Calling. Both are limited by the L-shaped file distribution because neither parallelize within files.

4 DISCUSSION

Big-data genomics research utilizes powerful computations [2]. Even with advancements in high performance computing and high throughput technology, there remains a demand for parallelized file aggregation tools in order to manage these datasets. Research and development for parallelized filesystems to scale large data operations must be able to support the I/O demands and may need to explore non-tree based directory architectures [2]. The potential for new filesystems integrating these facets of parallelism, high scalability, and security will yield future advancements in file aggregation/data management in response.

Unlike ParFu, popular file aggregation tools such as ptar[6] and pigz[7] utilize multi-threaded compression, but do not take advantage of the parallel filesystems found in cluster environments. Parallel filesystems can really increase the speed if all processing elements are utilized in parallel. rsync[8] can be useful for file synchronization between systems, using remote-update protocol [8]. However, its delta-transfer algorithm restricts bandwidth, which may strain the filesystem, and running rsync in a parallel file system does not increase the speed of file transfer [8].

A parallel version, parasync, utilizes concurrent rsync runs, but does not perform well with small data sets or slow network connections [7]. fcp [9], developed for fast copies of data between file systems, employs multiple nodes to increase performance. It has good scalability as node count increases, but no significant increase in performance for stripe counts below 32 when the input dataset consists of many small files [9]. ParFu sees similar performance improvements as the number of ranks increases and with stripe counts below four.

Further testing should focus on the effects of changing block size and number of blocks per group for ParFu, utilizing different compression settings for ptgz, and performance comparisons against popular file aggregation tools.

[6]https://github.com/moul/ptar
[7]https://github.com/madler/pigz
[8]https://github.com/AndyA/rsync

ACKNOWLEDGMENTS

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 Craig Steffen with contributions from Roland Haas as part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards 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.

REFERENCES

[1] Akram Alyass, Michelle Turcotte, and David Meyre. 2015. From big data analysis to personalized medicine for all: challenges and opportunities. *BMC Medical Genomics* 8, 1 (27 Jun 2015), 33. https://doi.org/10.1186/s12920-015-0108-y

[2] Steve C. Chiu. 2007. High performance I/O architectures and systems. *The Journal of Supercomputing* 41, 2 (01 Aug 2007), 105–108. https://doi.org/10.1007/s11227-006-0032-1

[3] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, Aaron McKenna, Tim J Fennell, Andrew M Kernytsky, Andrey Y Sivachenko, Kristian Cibulskis, Stacey B Gabriel, David Altshuler, and Mark J Daly. 2011. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat Genet* 43, 5 (05 2011), 491–498. http://dx.doi.org/10.1038/ng.806

[4] Dwayne Edling. 2011. Evaluating Tape Drive Performance. (2011). Retrieved August 4, 2017 from http://www.oracle.com/technetwork/articles/systems-hardware-architecture/evaluating-tape-drive-performance-294728.pdf

[5] M. Factor, K. Meth, D. Naor, O. Rodeh, and J. Satran. 2005. Object storage: the future building block for storage systems. In *2005 IEEE International Symposium on Mass Storage Systems and Technology*. 119–123. https://doi.org/10.1109/LGDI.2005.1612479

[6] Anneke Lucassen and Richard S Houlston. 2014. The Challenges of Genome Analysis in the Health Care Setting. *Genes* 5, 3 (09 2014), 576–585. https://doi.org/10.3390/genes5030576

[7] Harry Mangalam. 2017. parsync - a parallel rsync wrapper for large data transfers. (2017). Retrieved June 14, 2017 from http://moo.nac.uci.edu/~hjm/parsync/

[8] Linux Man Page. "". rsync. (""). Retrieved June 14, 2017 from https://linux.die.net/man/1/rsync

[9] 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