Proceedings of a pre-conference workshop of the 27th International Cartographic Conference:
Spatial data infrastructures, standards, open source and open data for geospatial (SDI-Open 2015)
20-21 August 2015, Brazilian Institute of Geography and Statistics (IBGE), Rio de Janeiro, Brazil

A solution for processing large files in the LASer (LAS) format using the message passing interface (MPI) and parallel file systems

Jeffrey Wendel[1], Michael P. Finn[1], John Kosovich[2], Jeff Falgout[2], Yan Liu[3]

[1]U.S. Geological Survey, Center of Excellence for Geospatial Information Science
mfinn@usgs.gov
[2]U.S. Geological Survey, Core Science Analytics, Synthesis, & Libraries
[3]CyberInfrastructure and Geospatial Information Laboratory, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign

Keywords: parallel throughput architecture, lidar, MPI, geospatial data, message passing interface, point clouds, raster datasets

1. Introduction

In recent high-performance computing (HPC) research, a persistent, restrictive case problem arises when designing scalable computational solutions for geospatial data with regard to input/output (I/O) (Behzad *et al.*, 2012; Finn *et al.*, 2015). We inspected high performance I/O for supporting parallel read and write of raster (grid) datasets, and more particularly, very large lidar point clouds that were interpolated to grid datasets. We illustrate a fresh solution for processing large lidar datasets by taking advantage of HPC power through the use of the Message Passing Interface (MPI) and the Lustre Parallel File System (Piernas *et al.*, 2007).

2. Study Area, data, and test design

We acquired lidar point cloud data over areas of the Great Smoky Mountains and the Grand Canyon National Parks in the United States. We used the lasmerge application (Isenburg, 2014) to merge a subset of the Great Smoky Mountains data into one file of approximately 16 gigabytes (GB) with 572,693,051 points over a 40,000 X 20,000 meter area. Also, we used the lasmerge application to merge a subset of the Grand Canyon data into one file of approximately 120 GB with 4,294,967,295 points (maximum for LAS v.1.2) over a 25,000 X 30,000 meter area.

Producing a DEM typically involves filtering and transforming (e.g. reprojecting) LASer (LAS) file format (ASPRS, 2011) data, and using that result to produce a DEM. We named our programs p_las2las and p_points2grid and tested them using the two large test files on the Extreme Science and Engineering Discovery Environment (XSEDE). Initial testing of our compiled parallel implementations in this environment using both the 16 GB and 120 GB point cloud files provided good results, which will be described below.

3. Description, implementation, and results

3.1 *p_las2las*

3.1.1 Description:

The las2las application and supporting LASlib library were extended with the MPI application programming interface (API) to allow the application to be run in parallel on a cluster. Our goal is an application that scales to arbitrarily large input, limited only by the volume of disk space needed to store the input and output files. Figure 1 shows the high level view of the application. The processes across the top are run in parallel, while the vertical flow describes the job flow.

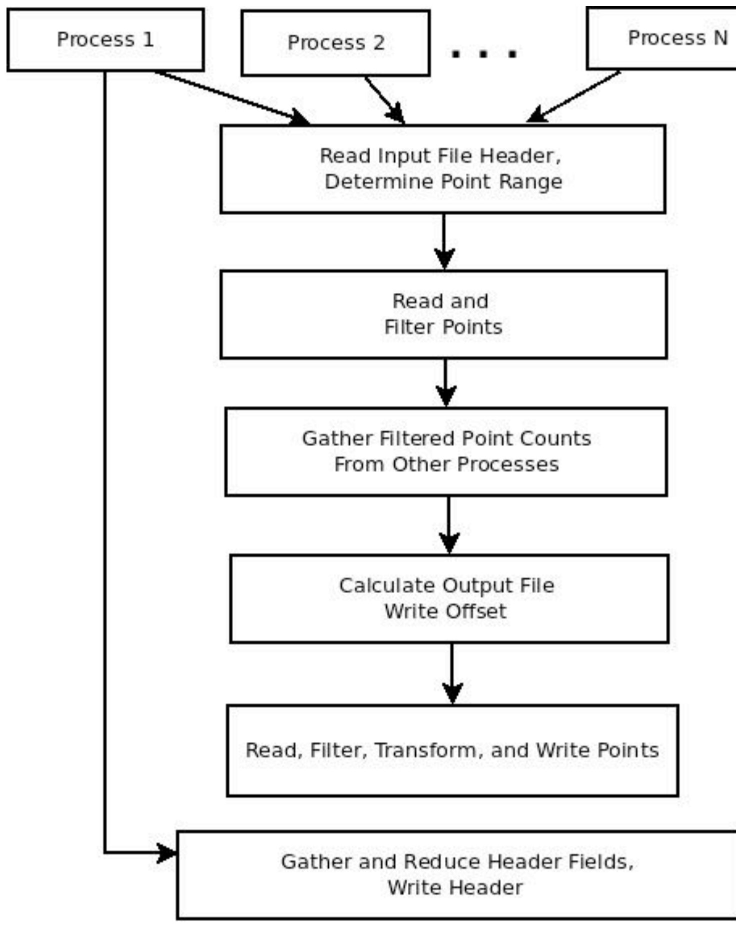

Figure 1. The high level view of the application. The vertical flow describes the job flow while the processes across the top are run in parallel on the flow.

3.1.2 Results

The test of *p_las2las* on the XSEDE Stampede cluster used a Lustre parallel file system with 64 Object Storage Targets (OSTs – Factor *et al*., 2005) available. We striped our 16 GB Smoky Mountains and 120 GB Grand Canyon test files over all 64 OSTs and specified a 4 megabyte (MB) stripe size; a common stripe size for large files managed by Lustre. The output directory was configured similarly. Table 1 shows the results of running the *p_las2las* program on the Stampede supercomputer using the 16 GB Smoky Mountains dataset using various numbers of processes. The asterisk in the table refers to execution runs with native, "unmodified" *las2las* source code from *LAStools* compiled on Stampede with the Intel C++ compiler. Table 2 shows the same data for the 120 GB Grand Canyon dataset. For both sets of results, the tables describe the difference in elapsed time between the various test cases, as a function of number of processors.

Table 1. Smoky Mountains 16 GB File Results on Stampede.

Number of Processes	Filter / Transformation	Output Size	Elapsed Time (seconds)
Native*	None	16 GB	138
Native*	Keep Class 2	2 GB	73
Native*	Reproject	16 GB	502
64	None	16 GB	20
64	Keep Class 2	2 GB	6
64	Reproject	16 GB	26
256	None	16 GB	8
256	Keep Class 2	2 GB	4
256	Reproject	16 GB	9
1024	None	16 GB	8
1024	Keep Class 2	2 GB	5
1024	Reproject	16 GB	8

* Native unmodified *las2las* source code from *LAStools* compiled on Stampede with the Intel C++ compiler.

Table 2. Grand Canyon 120 GB File Results on Stampede

Number of Processes	Filter / Transformation	Output Size	Elapsed Time (seconds)
Native*	None	120 GB	1211
Native *	Keep Class 2	25 GB	623
Native*	Reproject	120 GB	6969
64	None	120 GB	128
64	Keep Class 2	25 GB	59
64	Reproject	120 GB	150
256	None	120 GB	33
256	Keep Class 2	25 GB	18
256	Reproject	120 GB	42
1024	None	120 GB	18
1024	Keep Class 2	25 GB	9
1024	Reproject	120 GB	24

* Native unmodified *las2las* source code from *LAStools* compiled on Stampede with the Intel C++ compiler.

3.2 *p_points2grid*

3.2.1 Description

Our goal is an application that scales to an arbitrarily large input, limited only by the amount of disk space needed to store the input and output files. When run on a cluster, the number of processes used by *p_points2grid* is determined as a parameter to the scheduler. Figure 2 shows the high level view of the application. The job flow is described by the boxes on the right side while the processes along the left are the internal processes of the flow functions.

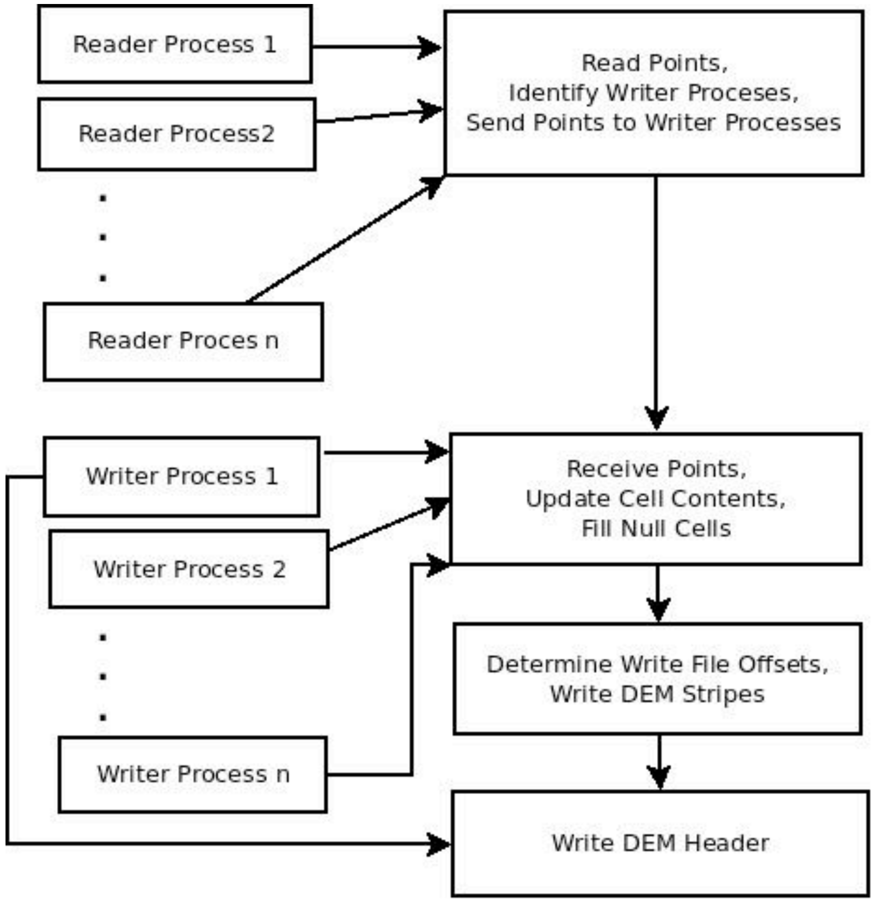

Figure 2. The high level view of the application. The job flow is described by the boxes on the right side while the processes along the left are the internal processes of the flow functions.

3.2.2 Results

The Smoky Mountains and Grand Canyon LAS input files were read from a Lustre File System on the "work" partition of Stampede. The files were striped over 64 OSTs with a 4MB strip size. The DEMs were written to the same directory that held the input files. The directory was configured to write files over 64 OSTs with a 4MB stripe size. Table 3 shows the results for the Smoky Mountains dataset and Table 4 shows the results for the Grand Canyon dataset. These results, in these two tables, show the varying time reading and communicating versus writing as the number of readers or writers are varied at execution time.

Table 3. Smoky Mountains 16 GB Input File Results, (12, 1 meter resolution DEMs totaling 70 GB of output for p_points2grid runs, 12, 6 meter resolution DEMs totaling 2 GB of output for native run. Times are in seconds.)

Number of Processes	Number of Readers	Number of Writers	Time: Reading, Communication	Time: Writing	Elapsed Time (seconds)
Native	1	1	NA	NA	328
128	32	96	33	56	105

128	64	64	26	84	125
512	32	480	20	13	40
512	64	448	10	17	33
512	128	384	8	23	36
512	256	256	7	26	40
512	384	128	11	44	68
1024	64	940	10	11	32
1024	384	640	2	14	29
1024	768	256	6	28	46

Table 4. Grand Canyon 120 GB Input File Results, (12, 1 meter resolution DEMs totaling 71 GB of output for p_points2grid runs, 12, 6 meter resolution DEMs totaling 2 GB of output for native run. Times are in seconds.)

Number of Processes	**Number of Readers**	**Number of Writers**	**Time: Reading, Communication**	**Time: Writing**	**Elapsed Time (seconds)**
Native	1	1	NA	NA	1548
512	64	448	104	15	135
512	128	384	55	21	90
512	256	256	60	30	110
1024	64	960	80	7	101
1024	128	896	39	11	62
1024	256	768	27	8	51
1024	384	640	24	15	53
1024	512	512	26	19	56
1024	768	256	47	24	90
1024	896	128	89	44	167
4096	256	3840	17	10	63
4096	512	3584	10	11	53
4096	1024	3072	8	8	46
4096	2048	2048	8	18	76

Our test runs of *p_points2grid* specified a grid resolution of 1 meter. No output or cell value types were specified, so each run produced 12 1-meter-resolution DEMs. In the Smoky Mountains test case, each DEM has a dimension of 40,000 columns by 20,000 rows and the total size of all 12 files is approximately 70 GB. In the Grand Canyon test case each DEM has a dimension of 31,000 columns by 26,500 rows and the total size of all 12 files is approximately 71 GB. We also ran the native "unmodified" *points2grid* application against our test datasets. We had to specify a 6 meter grid resolution because the memory requirements for 1 meter resolution were well beyond what the native application supports. These runs produced 12 DEMS totaling about 2 GB, or 36 times smaller than the 1 meter grid resolution DEMS produced by *p_points2grid*.

4. Conclusions

By creating parallel processing algorithms based on the open source *las2las* and *points2grid* code bases, we have shown greatly reduced run times processing extremely large datasets (over 100 GB), both in classifying the points and in generating DEMs. Using these programs, *p_las2las* and *p_points2grid*, we have shown through preliminary testing approximately two or more orders of magnitude reduction in processing time. In addition, we have shown scalability up to 4,096 processes.

Disclaimer

Any use of trade, product, or firm names in this paper is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

ASPRS (American Society for Photogrammetry and Remote Sensing) (2008) *LAS Specification, Version 1.2*. Internet at http://www.asprs.org/a/society/committees/standards/asprs_las_format_v12.pdf. Last accessed 24 November 2014.

Behzad, B., Y. Liu, E.Shook, M. P. Finn, D. M. Mattli, and S. Wang (2012). A Performance Profiling Strategy for High-Performance Map Re-Projection of Coarse-Scale Spatial Raster Data. Abstract presented at the *Auto-Carto 2012, A Cartography and Geographic Information Society Research Symposium,* Columbus, OH.

Factor, M., K. Meth, D. Naor, O. Rodeh, and J. Satra (2005) Object storage: the future building block for storage systems. In *LGDI '05: Proceedings of the 2005 IEEE International Symposium on Mass Storage Systems and Technology*, pages 119–123, Washington, DC, USA. IEEE Computer Society.

Finn, Michael P., Yan Liu, David M. Mattli, Babak Behzad, Kristina H. Yamamoto, Qingfeng (Gene) Guan, Eric Shook, Anand Padmanabhan, Michael Stramel, and Shaowen Wang (2015). High-Performance Small-Scale Raster Map Projection Transformation on Cyberinfrastructure. Paper accepted for publication as a chapter in *CyberGIS: Fostering a New Wave of Geospatial Discovery and Innovation*, Shaowen Wang and Michael F. Goodchild, editors. Springer-Verlag.

Isenburg, Martin (2014) *lasmerge: Merge Multiple LAS Files into a Single File*. Internet at http://www.liblas.org/utilities/lasmerge.html. Last accessed 03 March 2015.

Piernas, J., J. Nieplocha, and E. Felix (2007). Evaluation of active storage strategies for the lustre parallel file system. *Proceedings of the ACM/IEEE Conference on Supercomputing*. ACM, New York.

rapidlasso GmbH (2014) *Lastools*. Internet at http://rapidlasso.com/lastools/. Last accessed 24 November 2014.