Parallel Processing of LAS Files

Extending Open Source LAS Processing Software to Run in Parallel on a Compute Cluster

We've extended these popular LAS processing programs to run in parallel with the Message Passing Interface (MPI):

From LAStools:

- **lasmerge → p_lasmerge**
- **las2las → p_las2las**
- **lazip → p_laszip**

From Open Topography and Cold Regions Research & Engineering Lab:

- **points2grid → p_points2grid**

Parallel Processing of LAS Files

Extending Open Source LAS Processing Software to Run in Parallel on a Compute Cluster

LAS Processing Pipeline

Given a set of tiled LAS files:

1) Merge all files into one - ***p_lasmerge***
2) Filter ground points or reproject the points - ***p_las2las***
3) Make Digital Elevation Models - ***p_point2grid***
4) Compress to LAZ format - ***p_laszip***

ASPRS LAS File Format 1.2

PUBLIC HEADER BLOCK:

Item	Format	Size	Required
File Signature (“LASF”)	char[4]	4 bytes	*
File Source ID	unsigned short	2 bytes	*
Global Encoding	unsigned short	2 bytes	*
Project ID - GUID data 1	unsigned long	4 bytes	
Project ID - GUID data 2	unsigned short	2 byte	
Project ID - GUID data 3	unsigned short	2 byte	
Project ID - GUID data 4	unsigned char[8]	8 bytes	
Version Major	unsigned char	1 byte	*
Version Minor	unsigned char	1 byte	*
System Identifier	char[32]	32 bytes	*
Generating Software	char[32]	32 bytes	*
File Creation Day of Year	unsigned short	2 bytes	
File Creation Year	unsigned short	2 bytes	
Header Size	unsigned short	2 bytes	*
Offset to point data	unsigned long	4 bytes	*
Number of Variable Length Records	unsigned long	4 bytes	*
Point Data Format ID (0-99 for spec)	unsigned char	1 byte	*
Point Data Record Length	unsigned short	2 bytes	*
Number of point records	unsigned long	4 bytes	*
Number of points by return	unsigned long[5]	20 bytes	*
X scale factor	double	8 bytes	*
Y scale factor	double	8 bytes	*
Z scale factor	double	8 bytes	*
X offset	double	8 bytes	*
Y offset	double	8 bytes	*
Z offset	double	8 bytes	*
Max X	double	8 bytes	*
Min X	double	8 bytes	*
Max Y	double	8 bytes	*
Min Y	double	8 bytes	*
Max Z	double	8 bytes	*
Min Z	double	8 bytes	*

ASPRS LAS File Format 1.2

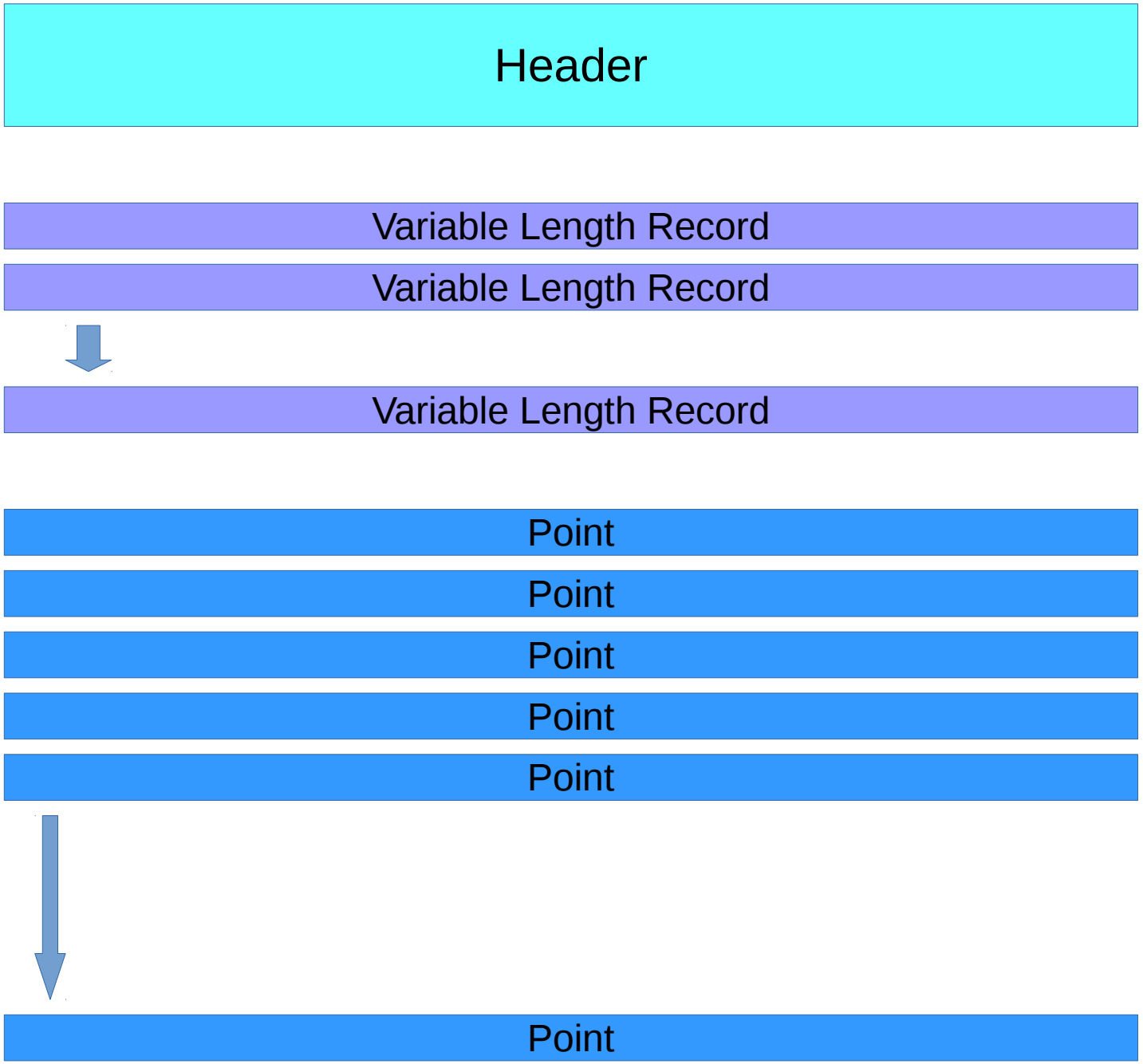
VARIABLE LENGTH RECORD HEADER

Item	Format	Size	Required
Reserved	unsigned short	2 bytes	
User ID	char[16]	16 bytes	*
Record ID	unsigned short	2 bytes	*
Record Length After Header	unsigned short	2 bytes	*
Description	char[32]	32 bytes	

POINT DATA RECORD FORMAT 1:

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0, 1, 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3, 4, 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	unsigned char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*

ASPRS LAS File Format 1.2



p_lasmerge

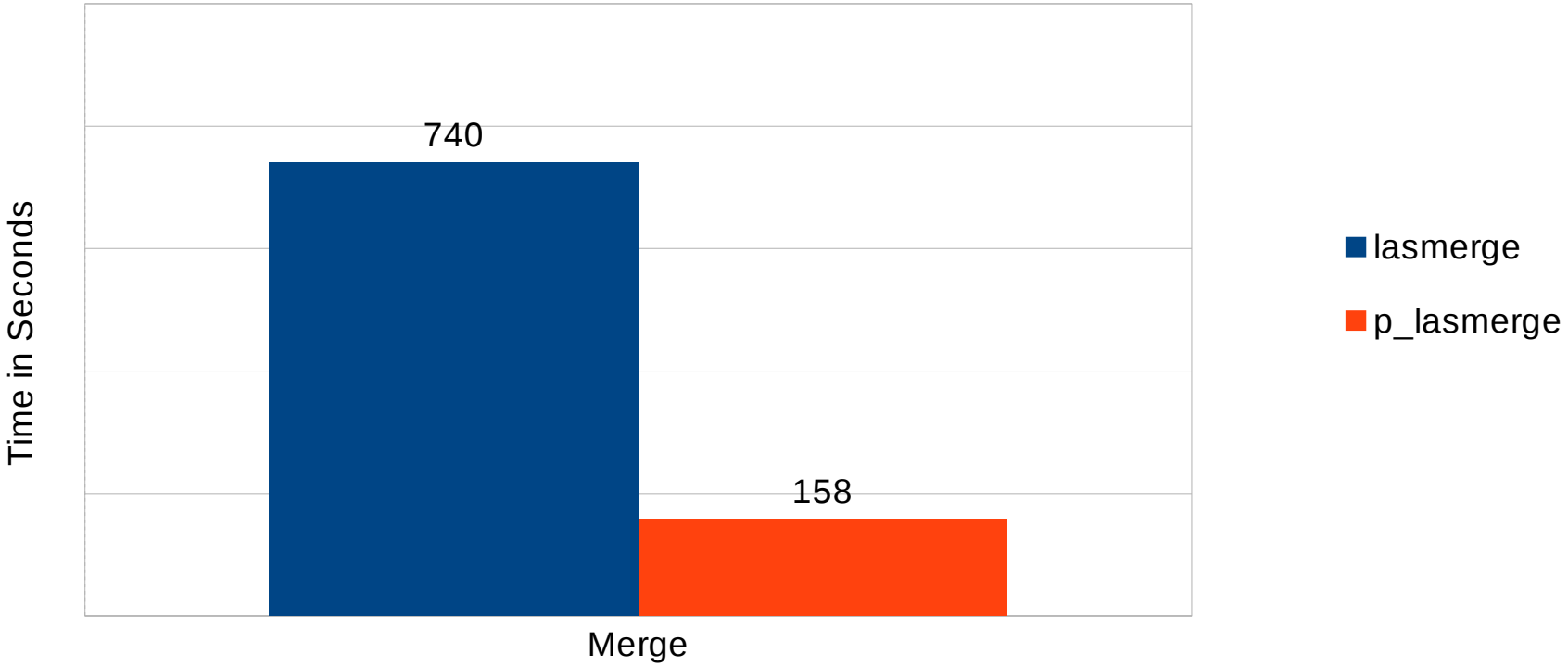
Merges any number of LAS files into one ouput file.
Usage example:
  mpirun -n 100 p_lasmerge -i *.las -o out.las

Approach:

- Process 0 merges input file headers and writes the output file header
- Each process:
  - Determines the set of input files it will process
  - Peeks into the header's of other processes and calculates an output file starting offset
  - Seeks to it's output file offset
  - Reads and writes one point at a time from its input files
- Process 0 gathers and reduces point counts, point counts by return type, minimum and maximum X, Y, and Z values, and updates the output file header.

lasmerge vs. p_lasmerge

785 LAS files merged to one 111GB file



- Native results from runs of lasmerge, version 160124, on Unity head node
- Parallel results from p_lasmerge using 100 cores and five Unity compute nodes, Lustre stripe count of 12 with stripe size of 4MB

laszip

LAZ Compression Overview
LAS → LAZ

- Write header and variable length records (VLRs) uncompressed
- Compress and write points in fixed size chunks, default is 50000 points
  - For each chunk:
    - Write first point uncompressed
    - Compress subsequent points using an entropy encoder (a scheme that predicts and exploits small changes from previous values)
- Write the chunk table (a table of file offsets that point to the first point in each chunk) starting at the end of the compressed point data

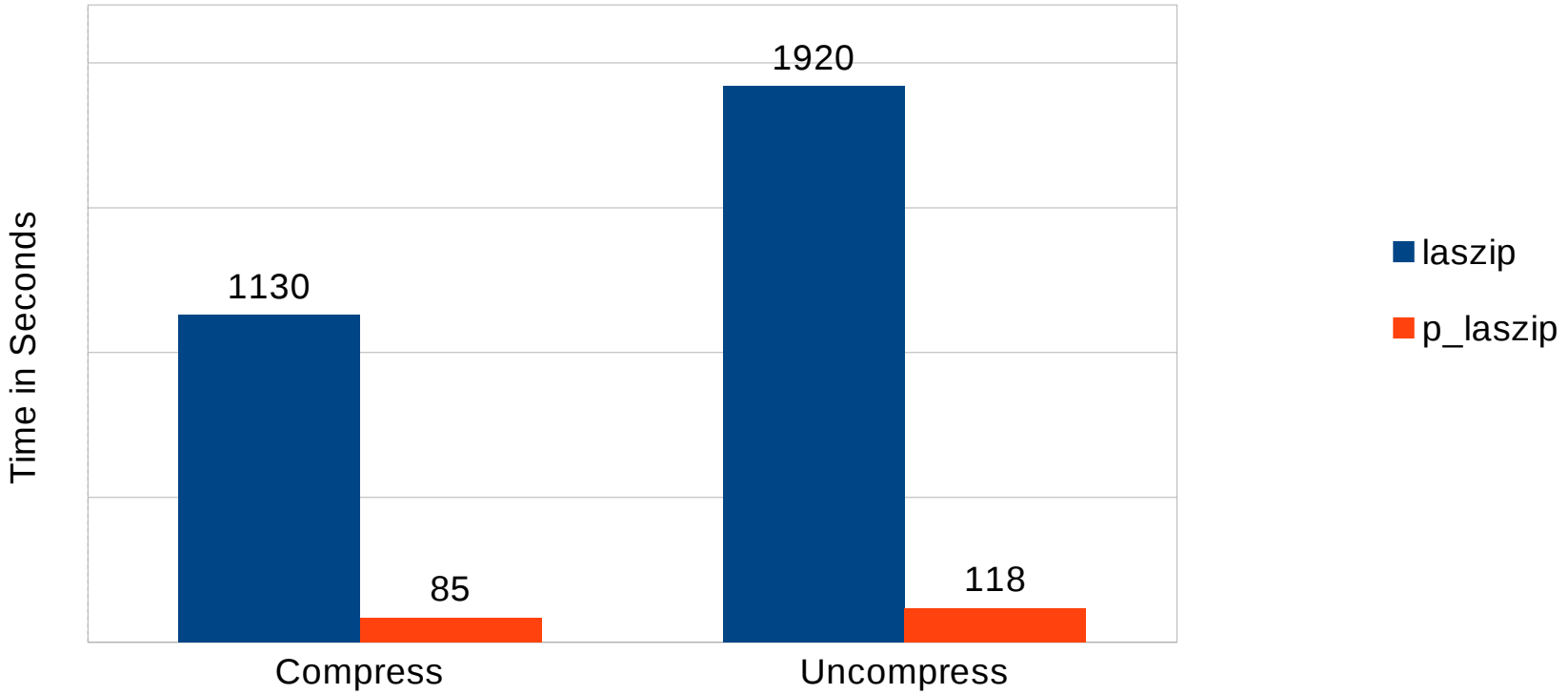
p_laszip

LAS → LAZ

- Each Process:
  - Calculates its point range, on chunk size boundaries, based on its rank and total point count
  - Reads and compresses its point range and sums its total compressed byte count
  - Gathers compressed byte count totals from other processes
  - Based on these totals and its rank, calculates and seeks to its output file offset
  - Reads and compresses its point range again, this time writing the compressed points to the output file
- Process 0 writes the uncompressed header and VLRs
- Process 0 gathers chunk table offsets and compressed chunk sizes from other processes and writes the chunk table starting at the end of the compressed point data

laszip vs. p_laszip

Compress and Uncompress 111 GB LAS File

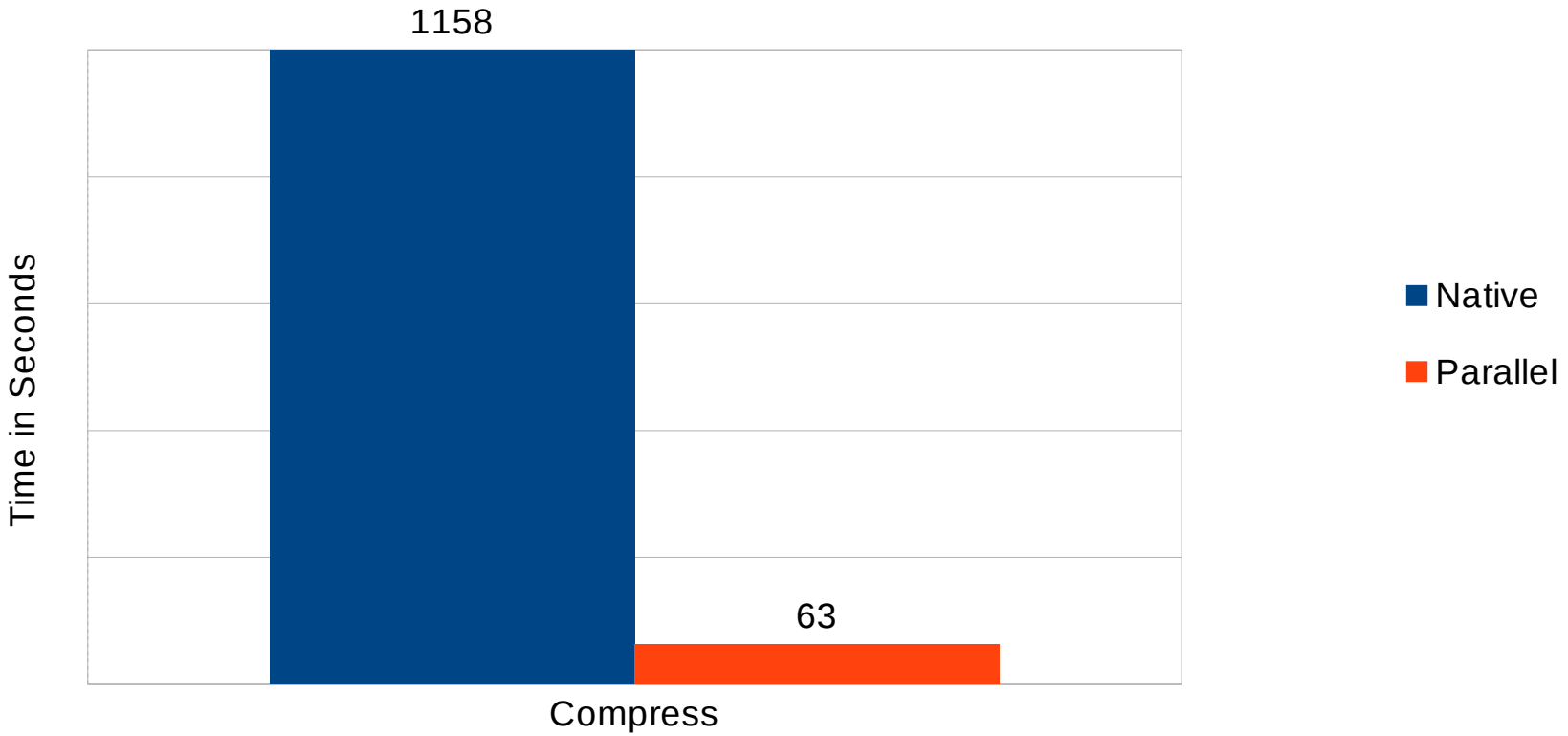


- Native results from runs of laszip, version 160124, on Unity head node
- Parallel results from p_laszip using 100 cores and five Unity compute nodes, Lustre stripe count of 12 with stripe size of 4MB
- Compressed LAZ file size was 12 GB

Embarrassingly Parallel laszip

- Embarrassingly Parallel is a Parallel Computing term synonymous with Perfectly Parallel or Pleasingly Parallel
- Used to describe problems where:
  - ➔ It is easy to decompose the problem into a number of concurrent tasks
  - ➔ Tasks require no communication with one another
  - ➔ Tasks have no dependencies on one another
- An example is using native laszip on a large set of small LAS tiles
- On a cluster the scheduler keeps all cores busy with concurrent laszip runs on individual files from the file set

Embarrassingly Parallel laszip Results
785 LAS Files, ~140 MB each



- Native result with laszip, version 160124, on Unity head node
- Parallel result with native laszip, version 160124, concurrently using 100 cores from 5 Unity compute nodes. Lustre stripe count of 4 and stripe size of 1M
- 785 tiled las files totaling 111GB used as input
- echo 3 > /proc/sys/vm/drop_caches run on all compute nodes and lustre machines prior to runs

Parallel Processing of LAS Files

Extending Open Source LAS Processing Software to Run in Parallel on a Compute Cluster

Github links:
https://github.com/jwend/p_lasmerge
https://github.com/jwend/p_las2las
https://github.com/jwend/p_points2grid
https://github.com/jwend/p_laszip

Questions?