pRasterBlaster: Fast, Accurate Raster Reprojection

David Mattli, Michael P. Finn, Michael Stramel

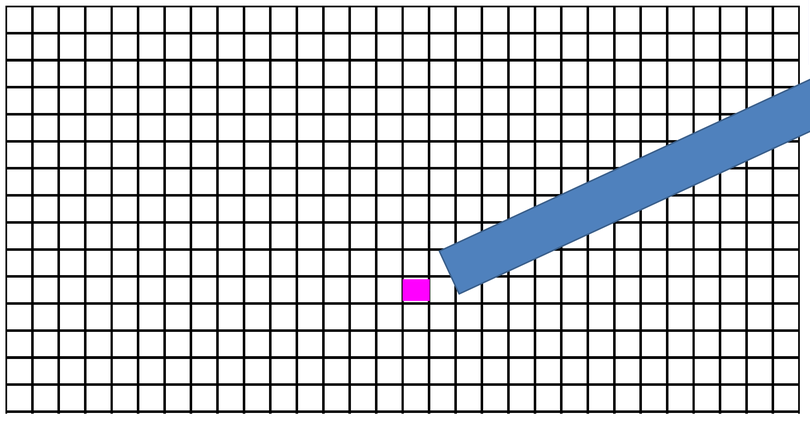
As part of its "Multi Resolution Raster" project the Center of Excellence for Geospatial Information Science (CEGIS) has researched fast and accurate reprojection of raster data for the USGS National Map. An early result of these efforts was the mapIMG software package. The mapIMG package implements reprojection of global raster datasets and solved several problems present in commercial offerings of the time. It also implemented several new categorical resampling techniques.

pRasterBlaster is a new implementation of mapIMG's reprojection techniques designed to run on multicore, parallel computers. It includes the sophisticated resampling techniques along with a new parallel reprojection implementation. pRasterBlaster uses multiple processors to quickly reproject large raster datasets.

http://cegis.usgs.gov

Step 1: Calculate and Partition Output Space

Each map projection represents a distinct coordinate system
- The area of the output raster dataset must be calculated by finding a minbox.
- The edge of the input raster dataset is iterated over translating input coordinates to output
- The smallest box that contains all of the calculated output coordinates is the minbox
- The calculated output minbox is then partitioned into areas to be assigned to processors
- Each output partition is matched with a partition in the input raster dataset
- This partition pair, input and output, is a single task for a processor



Output Raster Coordinate Space

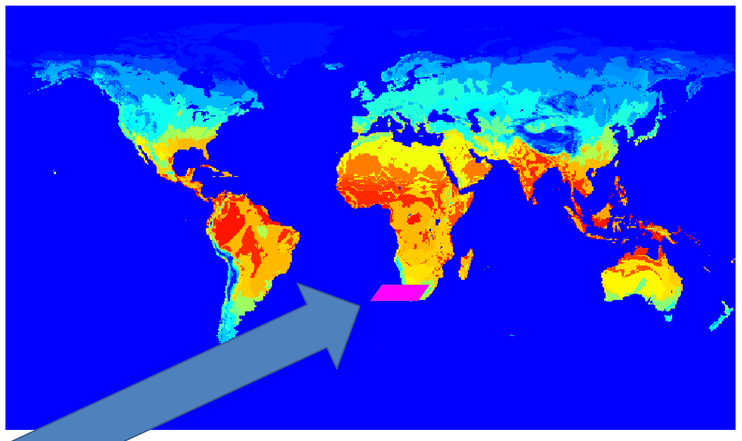
Output Space Partitioning

The picture above depicts an example output coordinate space overlayed with a grid representing the partitions. Early implementations used groups of rows as the partitions but now partitions can be arbitrary rectangular areas.

The magenta rectangle from the output raster coordinate space and the rhombus from the in put raster dataset represent the same area. Each area would be loaded into memory.

Step 2: Read Input and Reproject

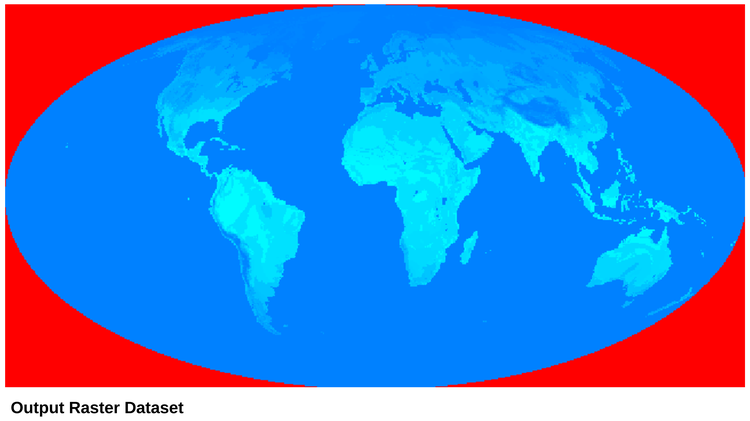
Each processor is assigned a quantity of input/output partition pairs
- Memory is allocated to hold the output and input partitions.
- The input partition is read from the filesystem.
- For each pixel in the output, the equivalent pixels in the input are used to find the resampled value
- Once the resampling is complete, write the output raster to a per-processor temporary file



Input Raster Dataset

Step 3: Combine Temporary Files

Each processor writes its output to a exclusive temporary file
- After all of the processors finish their partitions, they each take turns copying their temporary file contents to the final output file



Output Raster Dataset

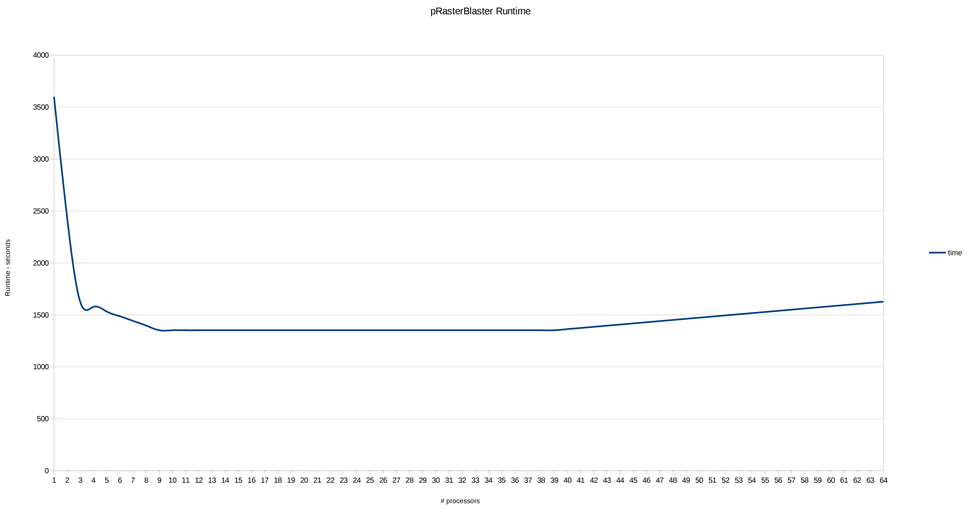
This raster dataset is the same as the above but in Mollweide projection. The red areas are those outside of the projected space. These areas have important performance consequences, see “Load Balancing” under Problems Encountered.

References

Finn, M.P., and Mattli, D.M., 2012, User’s guide for mapIMG 3—Map image re-projection software package: U.S. Geological Survey Open-File Report 2011–1306, 12 p.

Performance

The performance of a parallel program is very dependent on the number of processors and communication overhead. Below is a chart showing performance for a typical raster dataset with the current implementation. Performance levels out mainly because of the increased overhead from copying the temporary output files.



Problems Encountered

- **Parallel File I/O**
  - Raster dataset reprojection is an I/O intensive problem and file system I/O in parallel cluster environments is an area of active research. The current implementation is correct but slow. The use of parallel I/O routines from MPI-IO could potentially provide better performance.
  - The original implementation of pRasterBlaster used a naive parallel I/O strategy. Each process wrote to an exclusive portion of a shared file.
- **Load Balancing**
  - pRasterBlaster detects areas of the raster dataset that are outside of the input coordinate space. These areas are represented in red in the Output Raster Dataset. pRasterBlaster detects these areas and avoids reprojection and resampling. As a result these areas are processed very quickly. With the current partitioning scheme a processor may receive partitions that are almost complete made up of this outside area. Those processors complete very quickly where others may take much longer.
- **Dynamic Partitioning**
  - The algorithm pRasterBlaster uses to reproject raster datasets presents an unusual problem for partitioning. After the output space is partitioned, each partition is matched with its equivalent input area. The size of the output partition is easily controlled but the matching input area is dependent on a number of factors: output projection, projection parameters, and scaling.
  - If the matching input area is too large it may not fit into the memory assigned to the processor. As a result smaller partition must be found.
- **File I/O on Shared Clusters**
  - Because pRasterBlaster intensively uses the filesystem, running multiple instances simultaneously results in lower performance versus running one job at a time. Shared the limited I/O resources of large shared clusters is an area that needs further investigation.

red

2012-08-02