Importing high-resolution datasets into Geogrid

Jonathan D. Beezley*
Department of Mathematical and Statistical Sciences
University of Colorado at Denver

1. Introduction

As computers become more powerful, there is increasing interest in simulating smaller scale phenomena than typically associated with mesoscale weather forecasting. In particular, a typical WRF-Fire simulation occurs at mesh resolutions on the order of 10 m or less. Even the highest resolution surface data provided with WPS is several hundred times coarser. For these fine scale domains, the ability to import custom datasets into WPS is essential for the initialization of a realistic simulation (Mandel et al. 2011; Jordanov et al. 2011).

Resources such as the USGS's seamless data server that provide open access to high quality surface data are generating a large interest in initializing WRF simulations using custom datasets. WPS provides a mechanism for these datasets through a simple binary file format that is described in the WRF technical documentation (Wang et al. 2010); however, there is no standard API or GIS software capable of writing to it. Users who lack sufficient technical knowledge are currently unable to process this data into WRF.

Prior work related to WRF-Fire has lead to small utilities that are able to convert standard GIS GeoTIFF files into Geogrid's binary file format. TopoGrabber is a simple Python application based on this work that is capable of downloading and converting topological data automatically. More recently, modifications to Geogrid have been written allowing it to read GeoTIFF files directly. Current versions of this software are available in the repositories listed at `openwfm.org`. This research was supported by NSF grant AGS-0835579.

*jon.beezley@gmail.com

2. Geogrid binary format

Geogrid is a component of the WRF preprocessor responsible for interpolating surface data onto the simulation's computational grid. It reads surface data from a simple binary format consisting of a single text file specifying metadata and a number of binary files containing a rectangular block of data known as a tile. All of these files must reside in a single directory with the text file named `index` and the binary tiles formatted as `%05i-%05i.%05i-%05i` specifying the index range contained by each file. For example, a tile containing 500 columns and 250 rows could be named `00001-00500.00001-00250`. The tile size is specified globally, so all of the tiles must be exactly the same size. Each tile contains an unformatted array of integers with word size, signedness, and byte order specified by the index. The values are specified row-wise from bottom to top or top to bottom depending on which row ordering is specified. The row ordering controls how the data is ordered in each tile only; tiles are always indexed from bottom to top.

Because Geogrid processes each tile of data individually without special conditions for interpolating grid nodes located near tile bound-

aries, multi-tiled datasets must have a halo region where the tiles overlap each other. The size of the halo region is given in the index as `tile_bdr` and specifies number of extra rows and columns provided on each side of the tile. The extra data in the tiles is provided implicitly without changing the tile indices given by the file names. For example, a tile named `00501-01000.00251-00500` with a border width of 3 would actually contain columns 498–1003 and rows 248–503. Note that all tiles contain the border on all edges, even those tiles lying on the boundaries of the global dataset.

The storage format of the binary files is specifically designed so that a single dataset can be used on any platform regardless of word size or byte order; however it only allows for integer values. To account for non-integer data, the index also contains a global scaling factor by which all elements of the dataset are multiplied. The scaling factor provides the ability to represent fixed point numbers with a global precision. Other parameters specified in the index provide geolocation, field description, units, and a number that represents missing values in the dataset.

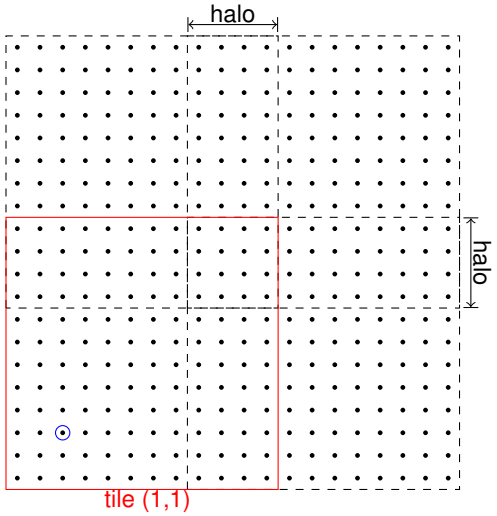

Figure 1: The Geogrid tile format with tile size 8 and halo width 4. Dots represent pixels in the data and rectangles enclose all pixels in a tile. The circled dot is at pixel coordinate (1,1).

3. GeoTIFF API and storage formats

The GeoTIFF image specification is an extension to the Tagged Image File Format (TIFF) that adds support for georeferencing data. At its core, a TIFF image is just a string of unformatted binary data; however before the data begins in the file, there is a header containing a block of metadata that describes how the bits should be interpreted. The metadata is organized as a series of key/value pairs, which can be accessed through a standard API call to the function `TIFFGetField`. These keys are called "tags" in the TIFF interface and are mapped to two byte unsigned integers by a preprocessing directive contained in `tiff.h`. The baseline TIFF standard specifies the meaning of a number of common tags, which must be supported by all TIFF readers. For example, the tag `TIFFTAG_IMAGEWIDTH` provides the total width of the image in pixels.

The TIFF image specification is very flexible allowing for many different formats including floating point, vector, and complex pixel types. In addition, the data can be organized within the file in a variety of ways allowing efficient access patterns for any number of applications. The most common storage convention is scanline based, where rows are stored from left to right sequentially from top to bottom. GeoTIFF files are often stored in contiguous tiles much like the Geogrid format, except that rows and tiles are ordered from top to bottom. Other formats such as multi-dataset files and multi-resolution pyramid formats are also support, but less often used in practice. The read and write routines in libtiff are optimized for access to individual tiles or scanlines sequentially, but also support random access to these blocks for uncompressed images.

A number of extensions to the baseline standard have been defined; GeoTIFF is one of these extensions adding tags that define geographic coordinate transformations and related enumerated codes (Ritter and Ruth 2000). When linked with the PROJ.4 library, libgeotiff also provides a method to convert pixel coordinates into geographic coordinates directly. This allows a user to find, for example, the latitude and longitude of the bottom left pixel of the image.

4. Modifying Geogrid to read GeoTIFF images

Adding GeoTIFF support to Geogrid without major code refactoring requires creating a layer over the libgeotiff interface to emulate the Geogrid file access routines. First, this emulation requires a metadata inquiry routine that retrieves all available information normally provided by the index. Second, there must be a layer above the actual data retrieval allowing Geogrid to request access to an arbitrary tile of data, which involves reading one or more blocks of data from the file and assembling them together.

The metadata provided by the tags in a GeoTIFF file do not necessarily contain all of the information required by Geogrid. In addition, metadata in a GeoTIFF file may be inaccurate or interpreted incorrectly. To account for this, the implementation uses a standard index file to supplement and optionally override the metadata in the GeoTIFF file. Geogrid reads index files in the subroutine `get_source_params`. If the parameter, `geotiff` is set to a file name in the index, then the file is opened by the GeoTIFF interface, and all that was not specified in the index is filled in by metadata from the GeoTIFF header. The actual metadata used is then output to `geogrid.log` allowing the user to confirm that the parameters are correct. Several parameters are never provided by the GeoTIFF interface and must be set by the user in the index when necessary. These include `description`, `units`, `type`, `category_min`, and `category_max`. The tile size and border are set to reasonable defaults, but can be set to provide more efficient access to the data depending on the internal storage format.

When Geogrid requires a tile of data, it requests a range of indices which is used to construct the file name of the tile. In the GeoTIFF interface, this index range is passed to a subroutine along with a handle to the open GeoTIFF file. The indices are converted from the bottom to top row order used by Geogrid into the storage format specified by the GeoTIFF file. Then one or more tiles or scanlines are read from this file and assembled into a contiguous tile as requested by Geogrid. Any pixels requested by Geogrid that lay outside the file are filled in with a constant associated with the `missing_value` parameter. The GeoTIFF file is kept open throughout the program's execution so that duplicate reads are cached in virtual memory.

The modified source in Geogrid is only compiled in when the user specifies a path to a GeoTIFF installation prefix through the environment variable `GEOGRID`. When the path is not given, WPS will still compile and run just as it would using the original source. This behavior is intended to make the extra features available to those who need it, without adding extra prerequisites for those who do not.

5. Conclusion

The implemented changes to Geogrid allows a user to import a USGS GeoTIFF dataset directly into the WRF workflow without the (often) difficult and error prone procedure of converting the data. The GeoTIFF specification is commonly used and can be provided as output from a wide range of GIS applications. The interface is compiled in as an optional component and is useful for both experienced GIS users with complicated workflows and those who just want to

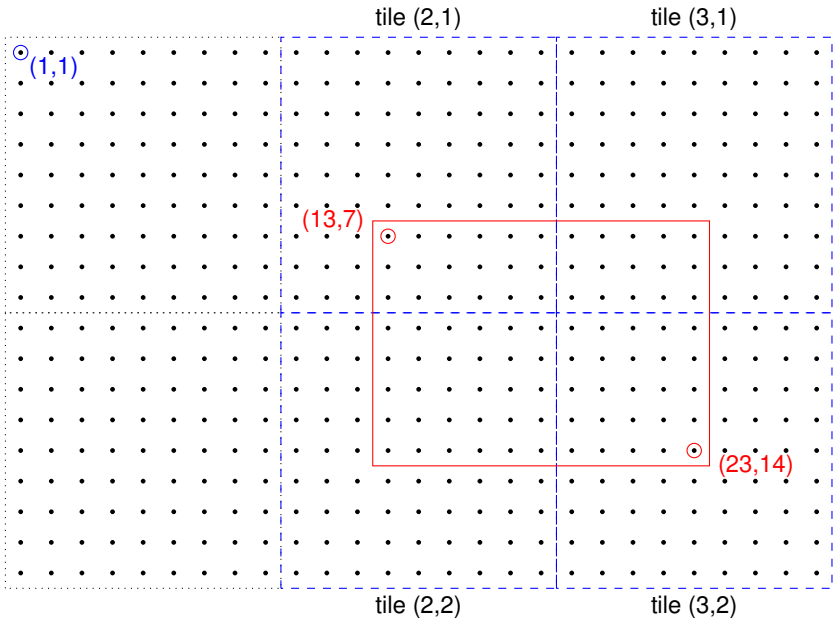

Figure 2: For GeoTIFF tile size 9 ordered from bottom to top and global pixel coordinates 13–23×7–14 requested, the output is constructed from the tiles highlighted in blue dashed lines.

replace a single data source. The implementation allows overriding erroneous metadata without modification to the source images and efficient access to the data. In addition, reading GeoTIFF files directly eliminates many of the limitations inherent in the Geogrid file format. Floating point data is read at full precision, and very large BigTIFF files (more than 2 GB) can be used even if they exceed $100,000$ pixels in a single axis. The current version of this software can be accessed from the repositories listed at `openwfm.org`.

References

Jordanov, G., J. D. Beezley, N. Dobrinkova, A. K. Kochanski, and J. Mandel, 2011: Simulation of the 2009 Harmanli fire (Bulgaria). 8th International Conference on Large-Scale Scientific Computations, Sozopol, Bulgaria, June 6-10, 2011, lecture notes in Computer Science, Springer, to appear.

Mandel, J., J. D. Beezley, and A. K. Kochanski, 2011: Coupled atmosphere-wildland fire modeling with WRF-Fire version 3.3. *Geoscientific Model Development Discussions*, **4**, 497–545, doi:10.5194/gmdd-4-497-2011.

Ritter, N. and M. Ruth, 2000: GeoTIFF format specification revision 1.0. `http://www.remotesensing.org/geotiff/spec/geotiffhome.html`.

Wang, W., C. Bruyère, M. Duda, J. Dudhia, D. Gill, H.-C. Lin, J. Michalakes, S. Rizvi, X. Zhang, J. D. Beezley, J. L. Coen, and J. Mandel, 2010: ARW version 3 modeling system user's guide. Mesoscale & Miscroscale Meteorology Division, National Center for Atmospheric Research, `http://www.mmm.ucar.edu/wrf/users/docs/user_guide_V3/ARWUsersGuideV3.pdf`.