The 36th International Symposium on Remote Sensing of Environment,
11 – 15 May 2015, Berlin, Germany, ISRSE36-593-1

Cloud Optimized Image Format and Compression

P. Becker, L. Plesea, T. Maurer

Esri, 380 New York St, Redlands, CA, 92373, USA
PBecker@esri.com, LPlesea@esri.com, TMaurer@esri.com

Commission VI, WG VI/4

KEY WORDS: Raster Format, Image Format, Compression, Cloud Storage, MRF, LERC

ABSTRACT:

Cloud based image storage and processing requires revaluation of formats and processing methods. For the true value of the massive volumes of earth observation data to be realized, the image data needs to be accessible from the cloud. Traditional file formats such as TIF and NITF were developed in the hay day of the desktop and assumed fast low latency file access. Other formats such as JPEG2000 provide for streaming protocols for pixel data, but still require a server to have file access. These concepts no longer truly hold in cloud based elastic storage and computation environments.
This paper will provide details of a newly evolving image storage format (MRF) and compression that is optimized for cloud environments. Although the cost of storage continues to fall for large data volumes, there is still significant value in compression. For imagery data to be used in analysis and exploit the extended dynamic range of the new sensors, lossless or controlled lossy compression is of high value. Compression decreases the data volumes stored and reduces the data transferred, but the reduced data size must be balanced with the CPU required to decompress. The paper also outlines a new compression algorithm (LERC) for imagery and elevation data that optimizes this balance. Advantages of the compression include its simple to implement algorithm that enables it to be efficiently accessed using JavaScript. Combing this new cloud based image storage format and compression will help resolve some of the challenges of big image data on the internet.

1. MANUSCRIPT

1.1 Overview

Image processing and analysis has changed significantly over the last 15 years. Traditional desktop image processing packages were designed to process one image at a time. Similarly much image processing was done in a sequential mode one image at a time. The massive increase in computation performance, storage and processing technology has changed image processing and analysis. Cloud infrastructures now enable massive volumes of imagery to be stored and accessed with many processes running in parallel. For example processes to compute segments or apply feature identification from images require multiple servers to quickly access large volumes of such image data. Esri's ArcGIS Image Server technology is an example of technology that enables large collections of imagery to be quickly accessed with the server applying a wide range of on-the-fly functions to transform the source pixels into valuable information products. These functions apply both geometric and radiometric transformations to the pixels, but require the servers to very quickly access near random sets of pixels from a large collection of images or rasters. A typical query that may need such access is the creation of a temporal NDVI (Normalized Difference Vegetation Index) profile for an area of interest. The databases on which the Dynamic Image Services are based provide instantaneously the list of scenes required as well as pixel georeferencing, but the pixel data must then be read from the mass storage and processed.

Such Dynamic Image Server based access enables a single copy of the source data to be stored while providing the client application with a near infinite range of products without the need to store the intermediate products. In this way the volumes of data stored are significantly reduced. Users can define their own processing functions to be applied on the servers. Such processes can be applied using the REST based image service or geoprocessing requests. Image Service requests provide synchronous access and typically process a screen worth's of pixels at a time, returning the results directly to the client application. Geoprocessing requests are typically asynchronous and can involve the server performing large number of individual processes on raster and vector datasets with the product typically being a map that is then accessed.

All these applications require fast access to the pixel values. Although ArcGIS can access imagery in any standardized format, performance of any processing system is affected by the storage location of the imagery, as well as the format and compression of the imagery. This has led Esri to carefully review how imagery is stored and compressed. Most imagery is currently stored in formats such as geoTIF, JPEG2000 or NITF. There are also a wide range of other formats such as netCDF, HDF or GRIB used primarily for scientific data. Each of these formats has developed a range of flavours and many such as geoTIF were adapted from more generic formats. For example geoTIF can be quite well optimized for access, if it is tiled and includes required overviews or reduced resolution datasets, but a large proportion of geoTIF files exists as non-tiled and so is not optimized for more random access.

1.2 Cloud Raster Considerations

These traditional image formats were all designed more than 10 years ago, when 'cloud computing' had not been conceived. The design criteria for the formats was primarily to handle the traditional desktop type access. Esri has been reviewing the requirements for storing and accessing imagery to identify how best to optimize access for cloud based image access and processing. Moving data to the cloud provides an inherent opportunity to change the storage format. Once in the cloud the mode of data access changes to using APIs to extract the required pixels or using protocols such as GeoREST, WMS or WCS. The format in which the data is stored needs to interoperable with multiple programs, but need not be in its original format, so long as the original data can be retrieved as part of a download process. The focus changes to ensuring that cloud based processing tools get fast access to the pixels as well as associated metadata.

There does not exist a 'god' format that will handle all the requirements. We have looked into identifying the optimum format taking into consideration the following primary requirements:
- Be accessible from cloud storage such as S3 as well as enterprise storage systems such as NAS and SAN.
- Handle very large volumes.
- Enable large numbers of scenes/images/rasters.
- Support both georeferenced of non-georeferenced imagery from satellite, aerial or UAS sensors.
- Support 8 to 64 bits/band with potentially large number of bands.
- Enable fast random access in terms of both scale and extent.
- The data can be assumed to be WORM (Write Once Read Many) as such scenes are typically not modified.
- Enable many simultaneous requests.
- Enable direct access and streaming.
- Handle different compression methods.

Unlike traditional file or enterprise storage, cloud storage such as S3 is accessed through HTTP and has relatively higher latency for each individual request, hence access is optimized by minimizing the number of requests that are made to identify and extract a group of pixels.

1.3 Meta Raster Format

Esri has identified the Meta Raster Format (MRF) designed by NASA [*1] as a highly optimal format due to its very simple design that enables cloud optimization and extensibility.

MRF is a very simple format for tiling imagery. Its original purpose was as a high performance web tile service storage format. MRF is optimized for fast reading and splits a raster dataset into 3 separate files:
Metadata file (.MRF) – XML file containing key properties such as the number of rows & columns, data type, tiling, tile packing, projection and location information. This file is purposely kept small.
Data file – File containing tiles of imagery data. Tiles may be fully formed raster images such as PNG, JPEG and TIF, or raw data, possibly compressed using Deflate or other compression algorithms. Esri has also added LERC compression as a tile encoding (see below).
Index (.IDX) – Very simple binary index of tile offsets and sizes within the data file, establishing the geometric organization of the tiles.
The extensions for the files are optional and can be changed if required.
Since MRF is a GDAL format, additional metadata not directly handled by MRF but supported by GDAL can be stored in .aux.xml (as defined by GDAL) or other metadata standards defined by source data products. Typically such metadata gets ingested into a database and is only accessed during processes that crawl for the data. MRF rasters can include reduced resolution overviews with factor 2 or 3, created using nearest or average down sampling.

The splitting of the raster in to three files is an MRF feature that helps to accelerate access to the data tiles, by optimizing file location on different classes of storage. In its simplest implementation copies of the small MRF and IDX files can be stored on low latency storage, while data file remains on slower storage. As a result when access to a tile is required, all the required metadata can be read with only limited requests to read from the slower storage. In the GDAL implementation, access to remote files can be achieved using VSICurl. The multiple file structure of MRF also enables applications to easily cache tiles that may be accessed multiple times thereby reducing repeat requests for the same tiles.

The MRF GDAL driver is open source and is available on Github. Esri has been contributing to its development and are integrating it into ArcGIS 10.3.1. Esri is also developing a JavaScript based reader that enables certain MRF files to be directly read and streamed by browser based applications. This JavaScript implementation also provides additional value for enabling a range of cloud processing algorithms to be implemented directly on MRF files.

MRF provides a way of optimizing access to the millions of scenes from satellite, aerial and UAS sensor. It has a number of advantages over the more complex traditional file formats, as well as key value map raster implementations such as NoSQL which are more optimized for dynamically changing data sets. MRF does have its limitations. It is not highly optimized for storing a very large disparate dataset, such as a single raster to define 1m resolution imagery of the entire globe. It is also not optimized for multi-dimensional datasets or for environments were multiple processors need to write to a single rasters, as may be the case for the output from raster analysis.

1.4 LERC Compression

Traditional image formats include a range of compressions. Common lossless compressions include LZW, Deflate, PNG, and JPEG2000. Common lossy compression include JPEG and JPEG2000. A lot of work has gone into optimizing these compressions for a wide variety of data sources. JPEG2000 has been optimized in many ways, and so now incorporates a wide range of different flavours. JPEG has remained relatively static and has been highly optimized due to its integration into web clients.

Compression of the data is important as it reduces both the storage costs and transfer volume. The reduction in data transfer volume can speed up access, on the condition that the CPU load required to decompress the imagery is low. One of the issues with some existing compression types is that the CPU load to decompress the image becomes a significant factor in the access speed. Where this cost is split over many separate client applications the cost can be negligible, but in applications where servers are processing massive volumes of data, the decompression costs become very significant. Similarly, to enable web clients to directly access the data without plugins, and for some big data processes, the decompression needs to be implementable in JavaScript.

We reviewed what lossy and lossless compression methods are most appropriate for MRF. JPEG is the most common lossy compression and is very efficient. It is primarily useful for 8bit 3band imagery having the advantage of being very fast for typical natural colour imagery. It does not provide as high a compression as some wavelet based compression methods, but has the significant advantage of being directly usable in web applications. We are looking to potentially incorporate other compression methods such as JPEG-XR which can handle higher bit depths while being optimized for speed. JPEG-XR to date has not been used in many geospatial applications due to the lack of a suitable container. A 12bit/channel implementation of JPEG does exist in GDAL as part of the TIF support, but is not widely supported. JPEG12 bit is relatively fast to decompress and has minimal effect on the pixel geometry. It is therefore valuable for the compression of panchromatic imagery where the lossy artefacts have minimal effect. One recommendation for reducing the size of scenes that have a higher resolution pan band, is to compress the pan band using lossy compression while using lossless compression on the multispectral imagery that is used for analysis.

Most Lossy compression methods are controlled by a quality parameter that controls the size of the resulting file, but does not control the maximum error of the pixels. ‘Controlled Lossy’ compression enables a tolerance to be defined that sets the maximum deviation that a compressed pixel may vary from the original value. A practical example is the compression of elevation data. Elevation often needs to be stored as floating point, but the source data often contains noise that is beyond the accuracy or precision of the measurements. Such data does not compress well using lossless compression and most lossy compression methods will result in uncontrolled accuracy degradation.

Esri has developed a new compression method called LERC (Limited Error Raster Compression) that was designed to provide such controlled lossy compression, while also being very efficient, such that it utilizes very few CPU cycles both to compress and decompress the data. The patented algorithm identifies the appropriate scaling to be applied to groups of pixels such that the each group can be quantized and efficiently compressed. LERC is used extensively in ArcGIS for the transmission of elevation data, but has also found to be very effective for the compression of imagery.

For imagery of analytical value lossless compression is required, this is especially true for the multispectral imagery from high resolution optical satellites and airborne cameras. A number of lossless compression algorithms exist including lossless JPEG2000, PNG, LZW and Deflate. JPEG2000 although providing the highest compression, has by far the highest CPU Load to decompress. From the other standard compressions Deflate provides a good compromise for good lossless compression and relatively low CPU load.

By setting tolerance to 0.5 for integer data, LERC also acts as a very fast lossless compression. The simplicity of LERC has enabled it to be coded in JavaScript and so can be incorporated into web applications that can directly work on the pixel data values. LERC also includes check sums that can be used to verify the integrity of the data, which can in some cases be compromised during the copying or moving of massive data volumes. Esri has added optional support for LERC to the MRF format.

We did an evaluation of the different lossless compression methods for a sample of high bit depth (> 8bit) imagery from Landsat 8, WorldView 3, Pleiades and UltraCam imagery.
The following table summarizes the typical difference in compression speed, resulting file size and time to read all pixels, as a factor of Deflate.

Compression Method	Compression Speed	Size	Read Time
Deflate	1.00	1.00	1.00
DeflateP2*	0.76	0.92	1.26
JPEG2000	0.56	0.62	8.68
LZW	2.92	1.20	1.17
PNG	0.41	0.90	1.99
LERC	3.00	0.81	0.94

DeflateP2 – Is deflate with horizontal differencing

The table above shows that LERC is very fast, being about 3x faster to write than Deflate while providing about 20% more compression and being a slightly faster to read. In comparison to Lossless JPEG2000, LERC is about 5x faster to write, 9x faster to read, but results is about 30% larger files.

1.5 Conclusion

MRF provides an optimized format for the storage of imagery in both cloud and enterprise environments. There are many cases where it is advantageous to transform the data to MRF when moving it to cloud or slower access storage environments. It has a simple structure that enables high performant implementations. For lossy compression MRF currently utilizes JPEG, but may be expanded to other compressions. For lossless compression None, Deflate, PNG or LERC compression can be currently used. The LERC compression provides further advantages in providing both lossless and controlled lossy compression, while being faster to both compress and decompress.

REFERENCES

*1 - OnEarth and MRF Now Available on GitHub https://wiki.earthdata.nasa.gov/display/GIBS/2014/02/04/OnEarth+and+MRF+Now+Available+on+GitHub