Storage Optimization for NASA's Massive Archive of Earth Imagery and Visualized Data - NASA's Global Imagery Browse Services

IN11B-2119
2024 AGU Fall Meeting, Washington, DC
CL#24-3917

Mauricio Hess-Flores[1], Joe T. Roberts [1], Matthew W. Graber [1], Jacqueline Ryan [1], Oleg Pariser [1], Grace Llewellyn [1], Zhangfan Xing [1], Lucian Plesea [2], Minnie Wong [3], Ryan A. Boller [4], Angela Li [4]

(1) Jet Propulsion Laboratory, California Institute of Technology, (2) ESRI, (3) Vantage Systems, Inc., (4) NASA Goddard Space Flight Center

Abstract

NASA's Global Imagery Browse Services (GIBS) delivers an enormous amount of Earth observation imagery through publicly accessible, standard web services. Billions of image tiles have been served to users around the world, across over 1100 Earth Science data products. GIBS serves as the backbone to popular NASA websites such as *Worldview* (web client for GIBS imagery) and *Eyes on the Earth*, and has become popular due to its straightforward integration into GIS applications such as QGIS and ArcGIS. This year, GIBS completed its multi-year transition from an on-premises implementation to a cloud-native implementation, allowing for greater scalability to accommodate new Earth Science missions with massive amounts of data, such as PACE, SWOT, and NISAR. This transition involved creating a new open-source cloud-optimized implementation of *OnEarth*, the open-source image server of GIBS. Currently, imagery served is pre-rendered and stored in S3 buckets as PNG and JPG MRFs.

Due to ever-growing storage demands and associated costs, the OnEarth team recently added support for a number of GDAL-supported image compression algorithms such as *LERC*, *brunsli*, and *ZenJPEG*, each serving an important purpose across different parts of the image collection. For example, LERC provides floating-point precision that is crucial for scientific usability and is not available through formats such as JPEG, which suffers from compression loss in pre-rendering as images are converted and compressed, and PNG, whose 8-bit depth incurs precision loss. Furthermore, LERC support is a subset of a larger effort to investigate potential solutions for imagery to be generated dynamically from cloud-optimized data. For compression and visualization of JPEG data, ZenJPEG improves the storage of *NoData* values, while brunsli can compress JPEG images on average 22% with no additional loss. A deep dive into the pros, cons, and performance evaluation of these compression algorithms will be performed.

GIBS and the Meta Raster Format (MRF)

- **GIBS** provides fast and open access to data products for web clients such as **Worldview**. Both were created in 2011, as a combo of Goddard's near real-time (NRT) product generation and JPL's mapping technology. Tens of millions of daily requests are handled through the WMTS, WMS, and TWMS protocols, and fully in the cloud since February, 2024.
- The main data type in GIBS and its image server, **OnEarth**, is known as *Meta Raster Format*, or **MRF**, which combines raster storage with tile web services and cloud computing. This format was developed at JPL in the 2000's, for storing tiled imagery mosaics at different spatial resolutions. It has been built directly into GDAL, and is supported by software such as ArcGIS and QGIS. OnEarth provides tools for generating MRF files and handling time periods.

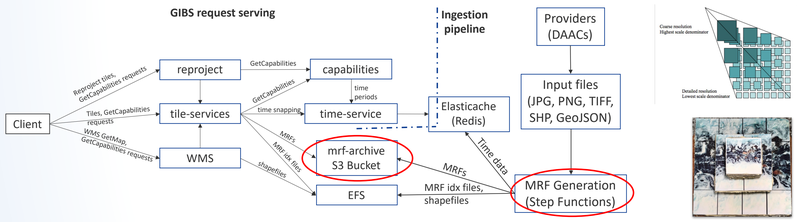


Figure 1. GIBS request serving with OnEarth and ingestion pipeline. Compression occurs during MRF generation (circled in red). MRFs provide access to an indexed heap of regular tiles (blocks) controlled by an XML file, usually organized as a pyramid of overviews, with level zero being the full resolution image (right).

Evaluated compression algorithms

***LERC* (Limited Error Raster Compression):**

- Compression format that can divide a raster into a number of pixel blocks, in which each pixel can be quantized and bit stuffed based on a number of block statistics, including the per-pixel maximum error allowable (zero error yields lossless compression).
- Usage: LERC handles floating-point data, which enables richer scientific analysis

***Brunsli* (JPEG compression):**

- Fast lossless JPEG recompressor, included in the draft for the new JPEG XL standard. It preserves data in a more efficient container and allows for an average 22% decrease in file size while allowing the original JPEG image to be recovered byte-by-byte. JPEG tiles can be served on-the-fly from a Brunsli MRF via the Apache httpd module *mod_brunsli*.
- Usage: Brunsli can be applied on any JPEG layer to further reduce file size

***ZenJPEG* (JPEG with transparency):**

- Implementation to enable the correct storage of *NoData* values with JPEG for improved lossy compression at 8 or 12 bits per channel, basically enabling transparency for JPEGs. The bit mask organized as 8x8 in 2D; compressed by run-length encoding (RLE).
- Usage: ZenJPEG results in smaller file sizes for layers with *NoData* or transparency values, as opposed to PNGs, which results in larger file sizes.

Compression performance

Compression and server-side performance testing for LERC was based on creating MRFs via GDAL (for LERC) and OnEarth's *mrfgen* tool (for brunsli and ZenJPEG) in order to compare file sizes and compression timing. All numbers were averaged across multiple runs.

LERC benchmarks:

			LERC precision							
			0		0.001		Histogram-derived value (parentheses)		LZW (baseline)	
Variable	Data range	Units							Baseline	
CO	[0, 0.000003]	kb	1015307	98.74%	4181	0.41%	15121 (10e-8)	1.47%	1028223	100.00%
		ms	657.914	265.70%	11.183	4.52%	16.391	6.62%	247.619	100.00%
NO2	[10e-11, 10e-9]	kb	2046416	99.43%	4161	0.20%	550120 (10e-13)	26.73%	2058201	100.00%
		ms	414.459	148.07%	11.290	4.03%	34.391	12.29%	279.904	100.00%
PM25_RH35_GCC	[0.269339, 696.068909]	kb	1795600	99.30%	1330819	73.60%	1796318 (0.001)	99.34%	1808247	100.00%
		ms	682.405	186.44%	11.264	3.08%	11.363	3.10%	366.023	100.00%

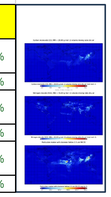


Table 1. Compression file sizes and average timing for LERC vs LZW compression. LERC compression rates outperform those of LZW even when the precision parameter (LERC_PREC) is set to 0, and is increasingly faster when the precision value increases. At 0, the algorithm looks for the best block size, compressing the data multiple times (similarly to PNG), which makes things slower.
Granule: GEOS-CF.v01.rpl.aqc_tavg_1hr_g1440x721_v1.20231112_0030z.nc4
Output size: 1440x721, data type: float 32; data resolution: 0.25x0.25; latitude resolution: 0.25; longitude resolution: 0.25

Variable	Data range	Units	LERC (PREC = 0)		LZW	
Aerosol Optical Thickness 550 Land Best Estimate	[0.02, 0.155]	kb	12882	59.12%	21791	100.00%
		ms	3.584	74.10%	4.836	100.00%
Angstrom Exponent Land Ocean Best Estimate	[0.093, 1.556]	kb	1605748	89.90%	1786226	100.00%
		ms	7.753	92.16%	8.413	100.00%

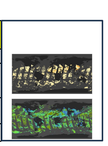


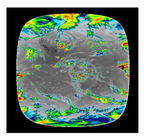
Table 2. Compression rates and timing for VIIRS data, using LERC (with zero precision) versus LZW compression. Even at zero precision, speeds increase relative to LZW as a function of data sparsity.
Size: 402x402, data type: Float32

Brunsli benchmarks:

Input and MRF size: 20480 x 10240		mrfgen run time (s)	gdal_translate - main call (ms)	MRF size (MB)	JPG size (KB)
brunsli on	avg	377.1362	0.2421	47	112
	stddev	4.2195	0.0105		
brunsli off	avg	296.3310	0.2442	62.7	138
	stddev	4.4474	0.0139		
Ratio brunsli on/off		127.27%	99.12%	74.96%	81.16%

Table 3. MRF and JPEG compression rates using Brunsli on GOES data, as tested with 'mrfgen' within an OnEarth container running on a Mac M1 machine.

ZenJPEG benchmarks:



	GOES dataset		OCI-PACE dataset	
MRF compression type	MRF generation run time (s)	MRF size (KB)	MRF generation run time (s)	MRF size (KB)
ZenJPEG	14.092	1272726	148.034	30724315
PNG	16.483	8890209	436.423	36364382
Ratio ZenJPEG/PNG	85.49%	14.32%	33.92%	84.49%

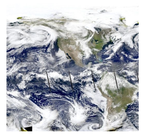


Table 4. ZenJPEG compression performance versus PNG, for the GOES (left) and OCI-PACE (right) datasets.

Client-side performance

Client-side performance is being measured only for LERC, given that client-side infrastructure has only been set up for this particular algorithm so far.

Layer	Bit depth	Resolution	Average (ms)	Dates
GEOS CF - CO	32-bit floating point	16km	160.849	1/1/2018-1/5/2018
GEOS CF - O3	32-bit floating point	16km	299.755	1/1/2018-1/5/2018
GEOS CF - PM25 RH35 GCC	32-bit floating point	16km	355.776	1/1/2018-1/5/2018
GEOS CF - SO2	32-bit floating point	16km	488.899	1/1/2018-1/5/2018
OMI*	32-bit floating point	1km	2697.054	1/1/2022-1/3/2022
VIIRS nighttime**	16-bit unsigned int	500m	147.902	7/1/2022-7/1/2022

Table 5. Timing numbers for rendering a LERC layer onto the client once a tile request is received (averaged over 40 runs), obtained by zooming out and panning the whole globe and averaging across multiple dates. Timing includes converting raw high-precision data into an image. Open Layers was used.
**OMI = OMI Total NO2 LRC v1 STD*
***VIIRS nighttime = VIIRS_VNP46A1_LERC_v1*

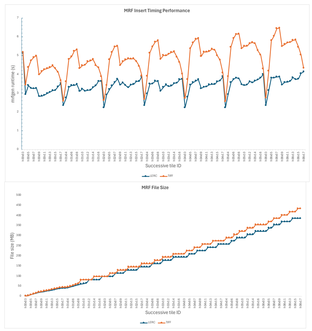
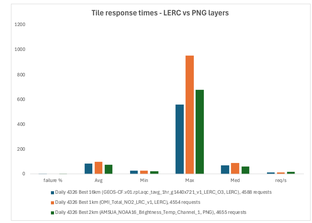


Figure 2. MRF insert performance. Top: timing for LERC vs TIFF as tiles are progressively inserted into an MRF. Bottom: respective MRF file sizes. Repetitive patterns are due to varying amounts of NoData values per tile when going around the globe (each row of tiles).



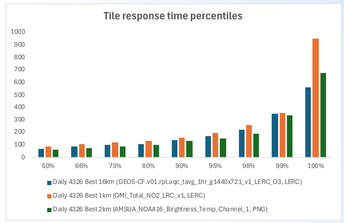


*Figure 3. Tile response time statistics for LERC vs PNG layers (left) and corresponding percentiles (right). **Times are comparable to current GIBS performance with PNG tiles**. Tests were run in an OnEarth instance within AWS and connected directly to the load balancer, to minimize network latency.*

Comparisons of each method

LERC:

- Outperforms LZW in compression rates and compression speed
- Handles floating point data; useful for scientific analysis
- Suited well for sparse datasets with *NoData* values taken into consideration
- Also used in ESRI's CRF format

Considerations:

- Extra work needed to evaluate the data histogram to get the best results

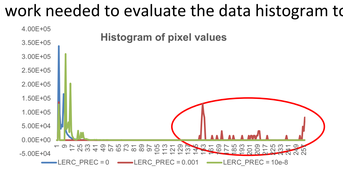
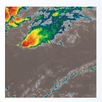


Figure 4. Histogram of pixel values at different LERC precision values. Image degradation occurs if an inappropriate value is chosen

Brunsli:

- Fast reading and writing when compared to PNG and DEFLATE; no tunable parameters



Considerations:

- Can only be used on JPEG layers, and does not support 12-bits-per-sample JPEGs
- Care must be taken with compression artifacts

ZenJPEG:

- Proper management of *NoData* values; zero pixels stored losslessly. No tunable parameters
- Works for 8 and 12 bits, and even in conjunction with Brunsli

Considerations:

- Specific only to MRFs and not to other GDAL-supported formats since the mask is generated and applied at the MRF codec level

Conclusion: optimal handling of compression within our architecture should be handled on a case-by-case basis: LERC is better for science data, brunsli for general JPEG compression, and ZenJPEG for handling *NoData* values. **As an example, MODIS, VIIRS, and GOES spectral JPEG layers make up a significant portion of the GIBS archive and can benefit directly from compression**.

What about other cloud-optimized formats?

- These methods can be applied to other formats, for example COGs
  - Can use LERC with COGs
- However, GIBS uses MRFs because:
  - COGs re-organize GeoTIFFs into cloud-optimized structures, increasing write time
  - MRFs use sparse .idx files to reduce disk space and support incremental file updates, unlike COGs
  - Zarr is not as good for visualizing interactive web maps if chunked temporally
  - Tiles can be read quickly from MRF files, which themselves are fast to generate and write
  - MRF is great for generating global mosaics at standardized grid and time resolutions

Effect of compression on costs

- Costs associated with data storage, data requests, and egress affect costs for AWS S3 and other services. File sizes can be reduced dramatically via compression, especially with LERC, at the expense of precision loss. Egress costs are associated with serving data from cloud storage to a client, based on the amount of data, the number of requests made against buckets and objects, and the transfer location of the data.

Future work

- Our performance evaluation on Meta Raster Format (MRF) generation via LERC, brunsli, and ZenJPEG compression shows encouraging results. All have been incorporated into OnEarth and we are in the process of operationalizing them within the larger GIBS system via updating the ingestion pipeline.
- Besides LERC, brunsli, and ZenJPEG, we plan on evaluating other algorithms such as QB3 and AVIF. We also plan on performing additional benchmarking once all algorithms are fully operational within our system.
- We're evaluating LERC MRFs as a caching format for other common Earth data formats such as NetCDF and HDF.

Please visit GIBS and Worldview at
GIBS: earthdata.nasa.gov/gibs
Worldview: worldview.earthdata.nasa.gov
Contact: onearth@jpl.nasa.gov