Zachary Garoutte
CEREO Living Atlas

File Compression Strategies for Cloud Storage in CEREO Living Atlas

Purpose

To minimize storage usage and reduce bandwidth costs, files uploaded to the Living Atlas platform should be compressed before being stored in Google Cloud Storage in order to allow us to keep the costs of running the cloud server to a minimum. This document outlines common compression methods, their advantages, and ease of implementation within a FastAPI backend.

Compression Methods

1. ZIP (Deflate)

- **Description**: Standard archive format combining compression and packaging.
- **Pros**:
  - Universally supported across platforms (Windows, macOS, Linux, browsers).
  - Easy to implement (`zipfile` in Python).
  - Can handle multiple files in one archive.
- **Cons**:
  - Compression ratio moderate; less efficient for very large media (video, images).
- **Best Use**: General-purpose compression for mixed content.

2. GZIP

- **Description**: Stream-based compression using DEFLATE algorithm.
- **Pros**:
  - High compression ratio for text, CSV, JSON, logs.
  - Widely supported.
  - Easy to implement (`gzip` in Python).
- **Cons**:
  - Works on single files only (not archives).
  - Slightly slower than ZIP for decompression.
- **Best Use**: Compressing text-heavy individual files.

3. BZ2 (bzip2)

- **Description**: Block-sorting compression (Burrows–Wheeler transform).
- **Pros**:
  - Higher compression ratio than ZIP/GZIP for text and structured data.
- **Cons**:
  - Slower compression speed.

    - Not as widely supported as ZIP
- **Best Use**: When maximum compression is more important than speed.

4. XZ (LZMA)

- **Description**: High-ratio compression algorithm (used in `.xz` archives).
- **Pros**:
    - Excellent compression ratio (better than bz2).
- **Cons**:
    - Slowest among standard methods.
    - Not natively supported on all OSs, so would have to decompress the file before sending it to the device to make sure the file actually works on that device. Which can be done easily in python but might slow download speeds as you have to download the whole uncompressed file.
- **Best Use**: Large static datasets where storage savings outweigh speed.

My Recommendation:

My recommendation is going to vary based on what the types of files are that we want to store, if we are wanting to store long text documents or just files that can potentially get large I would recommend the XZ compression method, while its slower then the others it is going to compress the files to a greater extent than any of the other methods. However if file size has a set limit and we limit things like photos to only png types being allowed as png types are naturally almost fully compressed we could use the tradition zip method of compression, and using zip compression would also allow users to upload a whole named folder that can be downloaded vs XZ is going to be limited to single file format. So in conclusion the ZIP method is convenient, native on any device, and allows folder format. However the XZ method is going to be more scalable and could save a dramatic amount of storage usage as the application grows making the costs of running the application cheaper in the long run.