Step-by-Step Guide to Building a Big Data Portal – pangeo – Medium

Jun 12, 2018

The volume of scientific datasets is growing at an exponential rate, and scientists are struggling to keep up. This Big Data crisis is one of the central motivators for the Pangeo Project. Some examples of Big Datasets in my fields of oceanography and climate science include:

Examples of similar magnitude abound in scientific fields like astronomy, bioinformatics, and high-energy physics, not to mention industry sectors like finance and manufacturing.

Organizations struggle to provide access to such datasets due to the technical difficulty of storing and transmitting such large volumes of data. Different approaches taken by different data providers have lead to a fragmentation of technology. Big Data portals abound, but they all work differently, making it harder for users to generalize workflows and merge data from different sources. Furthermore, scientists may want to share Big Data they generated from their own experiments, observations, or simulations directly, without the involvement of an official data provider.

The Short Version

Today I am happy to announce the *Pangeo Definitive Guide to Building a Big Data Portal* ©. Drumroll please…

1. Place your Big Data in cloud object storage in a self-describing, cloud-optimized format.
2. Congratulations! Your Big Data Portal is now complete. Take the rest of the day off.

The Details

While the *Definitive Guide* © is a complete plan, I suppose there are a few details that could be elaborated.

What is Big Data?

Big data is anything more than 100 GB, the size of a typical hard drive in a laptop. Next question…

What is cloud object storage?

Cloud object storage is a service offered by all of the major cloud providers (Amazon S3, Google Cloud Storage, Microsoft Azure Blob Storage). Cloud object storage allows you to upload an arbitrary collection of bytes (your data) with a unique identifier. These bytes can be retrieved at any time using this identifier. Cloud storage has effectively infinite capacity and scalability. If the data are retrieved from within the same cloud region, the transfer is fast and free.

But what about the portal part?

Since you have Big Data, the most effective way for users to work with your data is not to download it *out of* the cloud but rather to use distributed computing to process it from *within* the cloud, using cloud computing orchestrated with a platform such as Pangeo. In the cloud, users bring the computing to your data, not the other way around.

So if your data are self-describing, cloud-optimized, and stored in cloud object storage, *you don't actually need a portal*. The data are already ready for direct access and scientific use. That's great news, because portals take work to create and maintain. And despite the best intentions of data providers, portals are usually annoying. As a user of scientific data, whenever I encounter a fancy portal, the first thing I do is just look for the link to the FTP server where I can access the raw files. Then I use wget to suck them out into my personal dark replica.

Of course, there are still valuable and important services for data providers to provide, such as catalogs and tools for searching / data discovery. But when it comes to serving the actual data, the *Definitive Guide* © is all it takes: just point me to the address of the data I want within cloud object storage.

What is a "self-describing, cloud optimized format"?

This is the crux.

It's easiest to start with an example: the Cloud Optimized GeoTIFF (COG).

> A Cloud Optimized GeoTIFF (COG) is a regular GeoTIFF file, aimed at being hosted on a HTTP file server, with an internal organization that enables more efficient workflows on the cloud. It does this by leveraging the ability of clients issuing HTTP GET range requests to ask for just the parts of a file they need.

For a great overview of COGs, check out this article by Chris Holmes:

Before the cloud, the GeoTIFF format was already an established standard for geographic image data. It is a "self-describing" format because metadata such as map projection, coordinate system, ellipsoid, datum, etc. is embedded directly in the file. This eliminates the need for an extra file, service, or website to hold the metadata. Self-describing data is fairly common in geosciences.

In order to facilitate the transition to the cloud, the geospatial community came together and re-engineered the format to become "cloud optimized." The COG allows the metadata to be easily extracted without downloading the whole file, and for small pieces of the image to be downloaded one at a time, or in parallel. Open source geospatial processing software packages such as `GDAL` and `rasterio` have been updated to understand COGs and take advantage of their cloud-friendly attributes. COGs are both a robust archival format and analysis-ready—users don't have to copy or transcode the data to a new format in order to do science with it. This is important for Big Data, since one copy is already more than enough.

How can we generalize the COG concept to all scientific data? This is a hard problem. At the risk of oversimplifying, I contend that, at its core, most geoscience data falls into one of two categories:

1. **Tabular Data**: rows and columns; anything you could put in an Excel spreadsheet or CSV file. For example, a table of significant earthquakes.

2. **Multi-dimensional Arrays**: anything you could put in a netCDF or HDF file. For example, satellite observations of ocean temperature mapped onto a latitude, longitude, time grid.

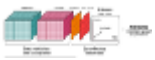
Tabular data, despite being heterogeneous and complex, is already well-supported on the cloud. That's because most data that comes from business is tabular data, and business has already basically made the transition to cloud. For example, the Apache Parquet format is a standard for storing tabular data which can be read and written from most popular programming languages. Parquet plays very well with cloud storage, as described in this article:

So if your Big Data is tabular data, use something like Parquet.

Multi-dimensional arrays are very common in scientific research; there is some overlap with the data covered by GeoTIFF, but multi-dimensional arrays are much more generic. Some examples of multi-dimensional array data are:

- 3D Output from climate models.
- Multi-spectral imagery obtained from microscopes, satellites, telescopes, or aircraft.
- Radar scans of glaciers.
- Ocean hydrographic measurements from ARGO floats.

On one level, multidimensional array data is conceptually simple because it tends to be homogeneous (all the values are the same data-type). However, there can be complex relationships between different arrays; for example, one array (*longitude*) might be the coordinates for a different array (*temperature*). In Earth Science, these relationships are encoded according to the Climate and Forecast (CF) Conventions, which are commonly used in netCDF or HDF files.



Multidimensional Array Dataset Schematic (from Xarray Documentation; created by Stephan Hoyer)

A fundamental problem today is that ***there is no widely accepted standard for cloud-optimized netCDF data***. The traditional formats were meant to be read from local files and do not work efficiently in cloud object storage. An overview of the technical challenges can be found in an excellent blog post entitled HDF in the Cloud by Matthew Rocklin.

For now, the Pangeo project has been experimenting with the Zarr format. Zarr provides and implementation of chunked, compressed, multi-dimensional arrays compatible with the scientific python ecosystem. These arrays can be stored in memory, on regular hard disk, or on cloud storage. Reading is entirely thread-safe, meaning that access can be parallelized easily. Arbitrary metadata, including units, coordinates, provenance, etc., can be embedded in the Zarr data structures. Most of all, *Zarr is simple*: the zarr specification is easy to understand. For these reasons, it was straightforward to create a Zarr backend for Xarray which effectively allows Zarr stores to act like netCDF or HDF files from the user's perspective. Pangeo is currently storing around 30 TB of data in Zarr format on Google Cloud Storage, and we are adding more every week. This data is directly accessible from pangeo.pydata.org, our flagship deployment of the Pangeo platform.

So while there is no widely accepted standard for cloud-optimized netCDF data today, it is clearly technically possible. Hopefully our community will soon converge on a solution that can be adopted widely. In the meantime, you are forgiven for not putting your netCDF data in the cloud just yet…

The Caveats

This article was deliberately provocative. My goal is to push back against what I see as the over-engineering and unnecessary complexity of online data access portals. The great potential of cloud computing is to *bypass the download step* of data analysis. This is a paradigm shift so profound that it calls for data providers to completely rethink their basic principles of operation.

But while the technical pieces are mostly already solved, the transition to cloud brings many unresolved social and financial questions:

- *Can we trust cloud providers with our precious scientific data?* When talking to data managers, I'm surprised how much I encounter this question. For me, it's a no brainer that cloud storage is more robust, scalable, and durable than on-premises hosting. But many colleagues harbor serious doubts about both the technical infrastructure and the business model of the cloud. They worry that cloud providers are planning some sort of bait-and-switch—to lure users into the cloud only to triple the prices once they have the data. Cloud providers could help allay these fears by communicating better with the scientific community.
- *How do we manage costs in the cloud?* In the cloud, the mix of expenses changes drastically; cloud storage costs about 5x more per GB than local storage servers, but all of the expenses related to local hosting facilities (power, cooling, system administration, etc.) disappear. It's not straightforward to compare. Moreover, data in the cloud is probably more than 5x as useful than data locked up behind a portal. If users access the data from cloud computing, there are no egress charges, but this requires users to change their way of working…
- *How do we convince people to use cloud computing?* Scientists are open to changing the way they work if there are obvious benefits. Projects like Project Jupyter have proven that interactive, browser-based computing is the preferred way to do exploratory data science. Emerging services such as Zero to Jupyterhub, Binder, and Pangeo make it easy to deploy notebooks in the cloud. I'm optimistic that we can make the switch almost without users even noticing.

In future blog posts, I will elaborate on how we can confront these challenges and help usher in a new era of scientific cloud computing, in which the scientific process is more efficient, more reproducible, and more fun!