Cloud-Native Repositories for Big Scientific Data

Ryan P. Abernathey
Lamont–Doherty Earth Observatory of Columbia University

Tom Augspurger
Anaconda, Inc.

Anderson Banihirwe
National Center for Atmospheric Research

Charles C. Blackmon-Luca
Lamont–Doherty Earth Observatory of Columbia University

Timothy J. Crone
Lamont–Doherty Earth Observatory of Columbia University

Chelle L. Gentemann
Farallon Institute

Joseph J. Hamman
National Center for Atmospheric Research

Naomi Henderson
Lamont–Doherty Earth Observatory of Columbia University

Chiara Lepore
Lamont–Doherty Earth Observatory of Columbia University

Theo A. McCaie
Met Office, UK. University of Exeter, UK

Niall H. Robinson
Met Office, UK. University of Exeter, UK

Richard P. Signell
US Geological Survey

***Abstract*—Scientific data has traditionally been distributed via downloads from data server to local computer. This way of working suffers from limitations as scientific datasets grow towards the petabyte scale. A "cloud-native data repository," as defined in this paper, offers several advantages over traditional data repositories—performance, reliability, cost-effectiveness, collaboration, reproducibility, creativity, downstream impacts, and access & inclusion. These objectives motivate a set of best practices for cloud-native data repositories: analysis-ready data, cloud-optimized (ARCO) formats, and loose coupling with data-proximate computing. The Pangeo Project has developed a prototype implementation of these principles by using open-source scientific Python tools. By providing an ARCO data catalog together with on-demand, scalable distributed computing, Pangeo enables users to process big data at rates exceeding 10 GB/s. Several challenges must be resolved in order to realize cloud computing's full potential for scientific research, such as organizing funding, training users, and enforcing data privacy requirements.**

CONTEMPORARY SCIENCE abounds with large, complex datasets used by many researchers. For example, thousands of climate scientists do research using the same multi-petabyte Climate Model Intercomparison Project (CMIP) simulation datasets [1]. The Human Cell Atlas and the Sloan Digital Sky Survey play similar roles in biology and astronomy respectively. These datasets offer exciting potential for new discoveries on important scientific problems and also represent an ideal target for exploitation by emerging machine-learning approaches. The science community's approach to infrastructure, however, may be holding us back from realizing this potential.

Traditionally, scientific data has been distributed via a "download model", wherein scientists download individual data files to local computers for analysis. This model requires that datasets be relatively small, or that users only want to look at a small piece of a larger dataset. But many of the most exciting scientific problems involve looking at the *entirety of very large datasets* in order to identify universal patterns and answer grand challenges. The download model poses several challenges for this mode of analysis. After downloading many files, scientists typically have to do extensive processing and organizing to make them useful for data analysis; this creates a barrier to reproducibility, since a scientist's analysis code must account for this unique "local" organization. Furthermore, the sheer size of the datasets (many terabytes to petabytes) can make downloading effectively impossible. Analysis of such data volumes also can benefit from parallel / distributed computing, which is not always readily available on local computers. Finally, this model reinforces inequality between privileged institutions that have the resources to host local copies of the data and those that don't. This restricts who can participate in science.

Cloud computing, with its ability to place large datasets and massive computational resources in close proximity, seems to offer an ideal solution to these problems. However, there are many different possible ways to organize and structure cloud computing for data-driven science. Proprietary platforms, like Google Earth Engine [2], are one-stop-shop solutions that provide both data and computing. They are powerful but generally controlled by a single company, with limited flexibility and modularity. In contrast, *open architectures* assume data will be distributed over the Internet and seek interoperability between different data catalogs and computational tools [3]. We contend that open architectures are the best path forward for big-data scientific infrastructure.

Cloud-based science platforms built on open architecture require many elements: software for interactive data analysis, machine learning, and visualization; elastically scaled computing resources; and access to data–not to mention examples and documentation, and resources for maintenance. As with any good system architecture, these components should be as modular as possible. While cloud computing frameworks are fairly well established, we feel that the data component is a missing link which is holding back adoption of cloud computing in multiple scientific domains.

In this article, we attempt to define a *cloud-native data repository* and explain how it is different from a traditional data repository. We put forward a set of objectives motivating the need for such repositories and outline an opinionated set of best practices for implementing cloud-native data repositories. We then describe a specific implementation by the Pangeo Project and conclude by enumerating some future challenges for building and maintaining cloud-native data repositories.

Related Work

Most related work on this topic has focused on data-proximate computing tightly coupled to data. Google Earth Engine was a pioneering effort in the field of geospatial analytics and remains a highly influential product in this space [2]. SciServer is a popular general-purpose data-proximate computing environment, and its deployment at Johns Hopkins provides large datasets from astronomy, cosmology, turbulence, genomics, and oceanography, together with a range of computing tools [4]. Also relevant are so-called "analytical databases"—tools that enable users to run complex server-side queries on multidimensional array datasets. Prominent examples include SciDB [5] and Rasdaman [6]. These tools are powerful and effective; however, they suffer from a structural problem: the same provider is responsible

Published by the IEEE Computer Society

for both data and computing. Our approach is somewhat unique in that we argue for decoupling of data storage from the computing platform, but without sacrificing performance.

Objectives for Cloud-Native Data Repositories

Cloud-native data repositories offer many advantages over the download model. Here we enumerate the objectives that motivate the need for such repositories. These objectives inform the best practices of the next section.

Performance: The initial motivation for moving analysis to the cloud is often to overcome performance bottlenecks associated with a local computing environment. These bottlenecks can be in the form of storage (limited disk space), I/O throughput, network bandwidth, and CPU, all of which can limit the speed of data analysis. Cloud computing offers the opportunity to scale resources to overcome performance bottlenecks, and cloud-native data repositories should first-and-foremost aim to enable high-performance data analysis.

Reliability: Traditional methods of data access often rely on custom servers that must be maintained by data providers and can crash under heavy loads. Cloud-native data repositories should shift the reliability burden to the cloud provider and take advantage of industry-driven innovation in this area.

Cost-Effectiveness: A clear cost inefficiency in the status-quo download model is that the same datasets are stored many times over on local hard drives. However, the personal computers used to analyze them are likely only active for a small fraction of the day; when they are active, they suffer from the performance limitations described above. Cloud-native data repositories should overcome these inefficiencies by storing only one copy of the data, where it is accessible to on-demand, elastically scaled computing. System architecture should also make it possible for different entities to shoulder costs of data and computing respectively.

Collaboration: One limitation for collaborative data science workflows on personal computers is the dependence on the local filesytem paths for data access. This dependence creates friction when collaborators attempt to exchange code and find that they don't have the same data or have organized their files differently. Cloud data generally uses a global namespace (absolute URLs with `https://` or `s3://` prefix) which renders analysis code portable. As a result, cloud-native data repositories should enable scientists to share functional code snippets, creating a faster and more fluid collaboration workflow.

Reproducibility: Reproducibility of data science projects requires open access to at least three elements: the code, the software environment, and the data. Local-filesystem data dependencies therefore also limit reproducibility, in the same way they limit collaboration. Cloud-native data repositories should complement reproducibility tools like Binder (https://mybinder.readthedocs.io/), which provides cloud-based code execution in a user-specified environment, allowing for compute-proximate data access.

Creativity: The download model limits scientists' creativity by locking them in to working with the data they have already downloaded and cleaned. In contrast, cloud-native data repositories should enable scientists to quickly pivot to new datasets as their work evolves organically.

Downstream Impacts: Many data providers, particularly in geospatial fields, are eager to help businesses leverage their data for economic gain. Modern technology companies, particularly startups, overwhelmingly use cloud computing to deploy their products. Cloud-native data repositories should enable and encourage a downstream ecosystem of commercial exploitation.

Access & Inclusion: The download model presumes that scientists have funding and infrastructure at their institutions to purchase powerful personal workstations and disk drives, plus sufficient bandwidth to download big datasets. This is not the case in many countries in the developing world, or even within parts of the United States in historically excluded communities. Cloud-native data repositories should help reduce these inequities by shifting the infrastructure burden to the cloud, where costs can be borne by centralized data and infrastructure providers.

Some of these objectives can be realized by moving analysis to traditional high-performance-computing (HPC) platforms, such as the supercomputers sponsored by NSF's XSEDE program. HPC can undoubtedly achieve excellent

performance for data-intensive workflows, and many HPC centers host vast volumes of scientific data precisely for this purpose. However, because access to HPC systems is heavily limited, the more social objectives on our list (collaboration, reproducibility, downstream impacts, access & inclusion) remain harder to realize in an HPC environment. In this article we focus on public clouds, while recognizing that HPC will remain an important tool for many data-intensive applications. HPC centers are also increasingly adopting cloud-style technologies, so this distinction will likely blur in the future.

Best Practices

Motivated by the above objectives, in this section we put forth some best practices for cloud-native scientific data repositories. These practices are informed by several years of experimentation and development, drawn from our experience in data-intensive scientific fields, including Earth-system science, neuroscience, and astronomy.

FAIR Data

The starting point for cloud-native data repositories is the widely lauded FAIR Principles: data should be Findable, Accessible, Interoperable, and Reuseable [7]. Adoption of these principles is growing within many scientific fields. Cloud-native repositories should build on these successful principles by providing persistent identifiers, rich metadata, machine-readable catalogs and metadata, and standards-compliant formats and protocols. At the same time, the motivations for moving to cloud computing pose some challenges to the FAIR lexicon. How do you make a petabyte of data "accessible?" Answering this question requires going beyond the FAIR framework and considering the full scientific workflow.

Analysis-Ready Data

We like to imagine that a scientist's job is to conceive new hypotheses, design instruments, build models, analyze and visualize data in interesting ways, contemplate results deeply, draw conclusions, and communicate findings. The reality is that many data-driven scientific disciplines are bogged down with the work of data downloading, cleaning, and preparation. This is a form of scientific toil, defined here as work that is "manual, repetitive, automatable" (from https://landing.google.com/sre/sre-book/chapters/eliminating-toil/). The outcome of this effort is *analysis-ready data* (ARD), which is ready for immediate exploration, visualization, and analysis. These ARDs are extremely valuable when shared openly, acting to accelerate research. The term ARD was coined in reference to geospatial imagery [8], where making data ready for analysis means acquiring a stack of images from a specific location, aligning them precisely at a pixel level, and performing various corrections to account for atmospheric distortion. We find that the concept of ARD generalizes well to many different fields; computational oceanographers, astronomers, and genomicists all have similar problems wrangling data. While the details of ARD production vary, the existence of this toil, and the resulting drain on scientists' productivity, is ubiquitous.

Traditionally, the toil of producing ARD has occurred in the margins of science, rarely acknowledged or described in publications but nevertheless a vital step in many workflows. ARD production has not been an area for collaboration; rather, it is a private activity, undertaken on a one-off basis for each project. As such, this effort is repeated every time the data is used, rather than once when it is generated. However, as data volumes grow into the "big data" regime (many terabytes to petabytes), the task of creating ARD becomes increasingly onerous. *We recommend that cloud-native repositories seek to publish ARD.* Doing so requires working closely with scientists from the community, who understand how to create ARD for their discipline. Publishing ARD does not mean abandoning the raw, primary data; instead the provenance chain between primary data and ARD must be carefully documented.

A key general attribute of ARD is a focus on meaningful, complete datasets, rather than individual data files (a.k.a. "granules"). This lightens the cognitive load for the data user. With ARD, scientists should be able to "open" a complete dataset which contains the full range of data needed for their analysis, rather than manually looping over many files. There are many ways to achieve this goal from a technical standpoint. For example, many files can be pre-concatenated

into a single massive file; however, such files can be cumbersome to move around. A more scalable approach involves tighter integration between software and data, allowing many individual files to be addressed as a single virtual object. Many modern data analysis tools implement such functionality; in the following section, we will describe a specific Python implementation in detail.

An example display of ARD is shown in figure 1.

Cloud-Optimized Data

The single greatest technical difference between local and cloud-based data environments is the reliance on object storage. Personal computers and HPC systems use POSIX filesystems to store data, and the expectation that data will be accessed via filesystem calls is often baked in to analysis software. In contrast, all modern public cloud platforms provide an object storage service as the cheapest and most scalable way to store massive amounts of data (for example, Amazon S3, Google Cloud Storage [GCS], and Azure Blob Storage). With object storage, data are read and written via HTTP calls. While cloud computing environments can certainly be configured with local hard disks, we argue that the objectives of performance, cost-effectiveness, and reproducibility can only truly be achieved when data access bypasses the filesystem and goes straight for the object store.

One approach to the data access problem is to fool the analysis environment into thinking that it is dealing with files, when in fact it is dealing with object storage. This trick can be played at the operating system level, by using Filesystem in Userspace (FUSE), or at the application level, e.g. by creating compatibility layers that wrap object storage with filesystem-friendly Application Programming Interface (APIs; e.g. fsspec https://filesystem-spec.readthedocs.io/en/latest/). In our experience, such approaches may work but usually fail to achieve optimal performance, due to both the the added complexity and the reliance on legacy, non-cloud-optimized file formats.

To understand why, it is crucial to recognize that local filesystems and object storage have very different performance characteristics. Local disks and filesystems offer low latency (in the order of milliseconds), but their throughput is limited by the physics of the device: roughly 100 MB/s for HDDs and 500 MB/s for SSDs. Because object storage services rely on HTTP calls, latency is relatively high (100 ms or more), but because they allow efficient parallel access, the throughput on large cloud platforms is essentially unlimited when coupled with distributed computing. (Experiments using the PyWren (http://pywren.io/) distributed serverless computing framework reported read / write throughputs in excess of 50 GB/s on S3 [9].)

While in principle the access protocol is independent of the file format itself, in practice, scientific data formats, and accompanying access libraries, often assume a certain access protocol. Many prevalent scientific data formats were developed only with POSIX filesystem access in mind, before object storage was prevalent. Examples include the popular Hierarchical Data Format 5 (HDF5) format as well as domain-specific formats like Network Common Data Form (NetCDF; geoscience), Flexible Image Transport System (FITS; astronomy) and ROOT (high-energy physics). Many of these formats, and their accompanying access libraries, perform numerous small seek / read operations when opening files and reading data. Therefore, even if the application can be tricked into thinking that it is accessing a filesystem when really it is talking to object storage, the accrued latency associated with these operations translates into very slow performance. Mitigating these challenges may require substantial refactoring of the access libraries, and this work is underway in several communities.

However, an attractive alternative to refactoring legacy data formats is to adopt more modern *cloud-optimized* formats that were designed from inception for use with object storage. Example cloud-optimized formats include AVRO, ORC, and Parquet for tabular data and TileDB Embedded and Zarr for multi-dimensional array data. Cloud-optimized GeoTIFF is a popular solution for geospatial raster data which extends a popular format with cloud-friendly features. These formats all share certain fundamental characteristics:

1) The metadata describing structure and content can be read quickly, allowing client-

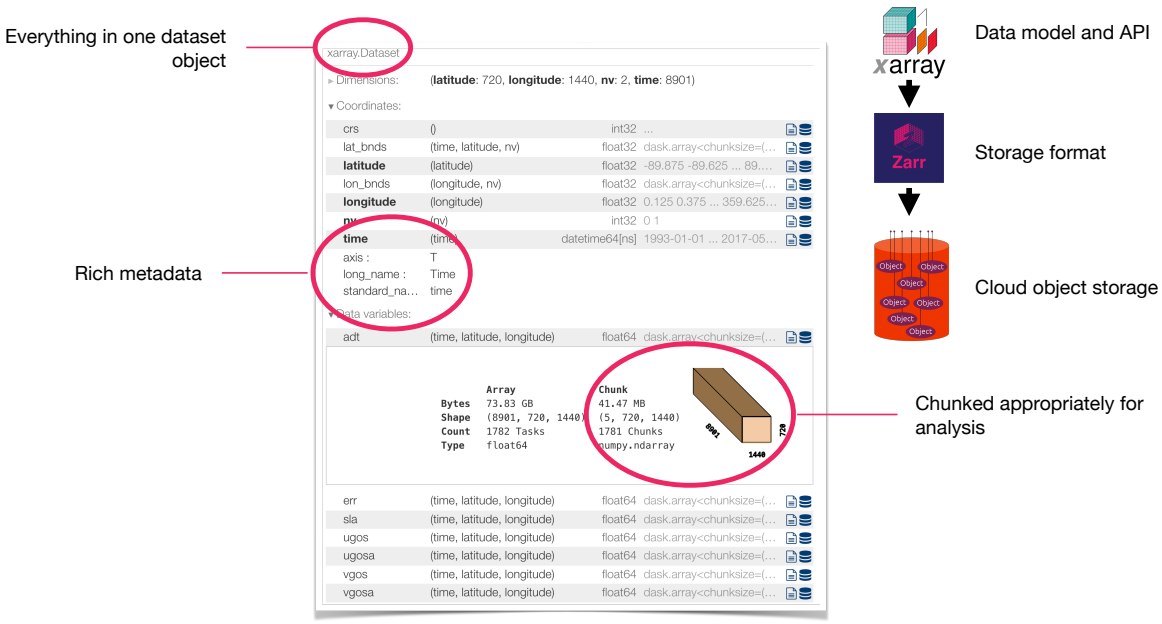

Figure 1. Example of ARD from the Pangeo Data Catalog. The high-level data model and rich visual presentation of the dataset are provided by Xarray. Under the hood, the data are stored in the Zarr format in cloud object storage. Use of the Dask framework allows for "lazy loading," meaning that we can view a very large dataset without loading it into memory. From https://catalog.pangeo.io/browse/master/ocean/sea_surface_height/.

side code to construct virtual representations of large, complex datasets.

2) Data can be read using the HTTP protocol, without the assumption of filesystem paths.
3) Data are organized in internal groupings (e.g. shards / chunks / tiles) that allow for efficient subsetting and distributed processing.

Combining cloud-optimized formats with object storage essentially transforms the object storage service into a high-performance REST API for data access, but without the burden of operating any additional infrastructure. Cloud-optimized formats are thus essential to achieving performance and cost-effectiveness. Taking full advantage of chunked data formats also requires cooperation from the analysis software, and indeed many distributed computing frameworks couple well with cloud-optimized formats.

As much as practical, we recommend that cloud-native data repositories adopt cloud-optimized formats. This may require transforming legacy formats upon ingestion. This activity can often be coupled with the process of making data analysis-ready, resulting in analysis-ready, cloud-optimized (ARCO) data. ARCO data is the gold standard of cloud-native data repositories. However, such transcoding can be expensive, time consuming, and fragile; the open question of how to best serve legacy formats from cloud-native repositories is taken up later in the section on future challenges.

Data-Proximate Computing

With traditional data repositories, users bring the data to their computer (the download model). In cloud-native data repositories, users bring their computing to the data; this is called data-proximate computing [10]. In contrast to the vertically integrated proprietary cloud platforms for data analysis, open-access cloud-native data repositories should not be coupled directly to a specific computing solution; instead they should seek maximum interoperability with a federation of computing services and access paradigms. This will foster an organic and evolving ecosystem between data providers and consumers, a federation of data and computing providers. Below we enumerate a few specific data-proximate computing approaches to consider.

Interactive computing is an essential element of modern data science. Interactive computing allows a data scientist to explore complex datasets, test ideas, receive visual feedback in the form of figures, and iterate rapidly to refine their analysis. Effective interactive computing requires a rich, dynamic user interface, which is easy to achieve in a local environment but more difficult with remote (e.g. cloud-based) computing systems. The Jupyter project has emerged as a leading tool for interactive computing in recent years [11], allowing users to interact with remote systems through a web browser. Jupyter in the cloud pairs very well with cloud-native data repositories as a solution for interactive, data-proximate computing. Many cloud platforms and third-party service providers offer managed Jupyter hosting. The Jupyter project also provides a comprehensive recipe for deploying your own cloud-based Jupyter Hub environment (https://zero-to-jupyterhub.readthedocs.io/). RStudio Server provides a comparable experience for R users.

Cloud-based Jupyter / RStudio environments run inside virtual machines that can be customized with different levels of computing resources (RAM, CPUs, GPUs) depending on the nature of the analysis and data. However, for truly big data, the cloud offers the opportunity to provision large numbers of compute nodes for a very short time for *parallel distributed computing*. Coupled with "spot pricing" (discounts for preemptable nodes), this creates a cost-effective way to process huge volumes of data. Popular distributed computing frameworks that can be deployed in the cloud include Hadoop, Spark, Dask, and Ray. Serverless computing (e.g. Amazon Web Services [AWS] Lambda) offers an even more scalable solution for distributed processing [9]. Cloud-native data repositories should seek to make their data accessible to and performant with these frameworks, which is mostly achieved by adopting cloud-optimized formats.

A final important application for cloud-native data repositories is *training machine learning models*. Machine learning, and deep learning in particular, benefit from very large, clean, homogeneous datasets on which to train their models—precisely the sorts of datasets that cloud-native data repositories will be providing. Cloud computing environments can provide the sorts of specialized hardware (e.g. GPUs, TPUs) that high-performance machine learning requires, and frameworks such as Kubeflow or Dask-ML can help coordinate complex machine learning experiments in the cloud. Cloud-native data repositories should strive to support machine learning applications such as these through high-throughput data access.

Data-proximate computing should ideally be deployed in the same cloud region as the data itself, as this will maximize performance and eliminate network egress charges. However, budgets permitting, cloud-native data repositories should also enable their data to be accessed openly over the internet, enabling multi-cloud workflows and integration with on-premises computing. A key point is that, while making their data accessible to data-proximate computing, *the data provider itself need not shoulder the computing costs*; these costs can be borne by individual research groups, private companies, or research funding agencies.

Open-Source and Portability

Beyond the technical best practices described above, we also encourage cloud-native data repositories to employ open-source software in a way that renders their underlying data and software portable across cloud platforms. Open-source software facilitates maximum transparency and reproducibility, increasing trust in a system. Particularly for publicly funded projects, releasing open-source software also enables the broadest possible access and impact by permitting others to reuse and adapt the tool; for this reason, many funding agencies now encourage or require open-source development. However, the definition of open-source can become ambiguous for infrastructure deployed in the cloud, since commercial cloud providers rely on many custom, proprietary tools to operate their cloud platforms.

To resolve this dilemma, we cite the open-source cloud user's "bill of rights", defined concisely by Yuvi Panda, core developer of the Jupyter project (https://words.yuvi.in/post/oss-in-the-cloud/). Operators of open cloud architecture should be able to

1) Run their software on any cloud provider they choose to

2) Run their software on a bunch of computers they physically own, with the help of other open-source software only

Adopting these principles can mitigate a major concern for institutions seeking to migrate their data infrastructure to the cloud: the fear of lock-in to a specific cloud provider. These principles constrain architectural choices. For example, the principles dictate that an open cloud-native data repository should probably not use a tool like Google BigQuery as its primary mechanism for storing data, since BigQuery is only available in Google Cloud Platform (GCP). In contrast, a data repository built around a generic object storage service is transferable to virtually any cloud provider.

An Implementation by Pangeo

The Pangeo Project (https://pangeo.io/) is a community organization aimed at advancing open-source software and infrastructure for the analysis of large, complex Earth system data. While initially focused on HPC environments, the Pangeo Project received NSF support for credits in GCP in 2017 and began deploying data analysis environments in the cloud. The best practices defined above emerged from that effort. Here we summarize our implementation of those practices achieved by integrating different open-source software tools from the scientific Python ecosystem. We emphasize that the best practices are general and could be implemented many different ways; the cloud-based genomics community uses a very different technology stack but follows broadly similar architectural principles [12].

Architecture

The Pangeo Cloud Data Catalog consists of ARCO datasets stored in the Zarr format. These datasets are multidimensional arrays, originating from satellite observations and numerical simulations, which conform to the NetCDF data model and the Climate-Forecast (CF) metadata conventions. The Zarr datasets are produced by aggregating many individual NetCDF files (e.g. all the temporal granules from a satellite product), setting an appropriate chunk size (approximately 100 MB), and writing the data to Zarr format directly to GCS in the `US-CENTRAL` region.

The majority of these datasets are cataloged with the Intake Python library (https://intake.readthedocs.io/) using Intake's YAML catalog format. These catalogs are stored in a GitHub repository (https://github.com/pangeo-data/pangeo-datastore), which uses continuous integration to validate new catalog entries. The Intake catalog can be used directly by Python users to interactively browse, search, and open datasets. It is also used to power a Flask-based web application that allows for browsing of the catalog via a public website (https://catalog.pangeo.io/). The data and catalogs are all public, allowing anyone to deploy data-proximate computing; however the "requester pays" setting is enabled. This requires users to pay the egress charges associated with transferring data out of the cloud region. (There are no egress fees for in-region access.)

The Pangeo Project also operates a data-proximate computing service, called Pangeo Cloud, for about 100 scientists. This service, described at https://pangeo.io/cloud.html, is deployed on both GCP and AWS. All computing services are managed within Kubernetes clusters, which are configured to scale capacity elastically up and down depending on system usage. The infrastructure is based on JupyterHub, and users connect via their web browser. After authentication (via GitHub), the JupyterHub service spawns a Jupyter server for each user, providing a private, interactive Python computing environments pre-loaded with commonly used packages for data access, analysis, and visualization. The user can choose different levels of computing resources (RAM, CPU) depending on their needs.

Users interact with the Data Catalog using the Intake library. Most of the datasets are optimized to be opened and analyzed with Xarray, a Python library that provides a high-level data model and computational API for working with labeled multi-dimensional datasets. Xarray supports "lazy" operations, in which data is not actually loaded from the storage device until explicitly required for computation and visualization. Xarray integrates closely with Dask to enable automatic parallelization of analysis operations. Interactive visualization of Big Data is enabled by the HoloViz (formerly PyViz) tools, which allow dynamic re-rendering on zoom [13].

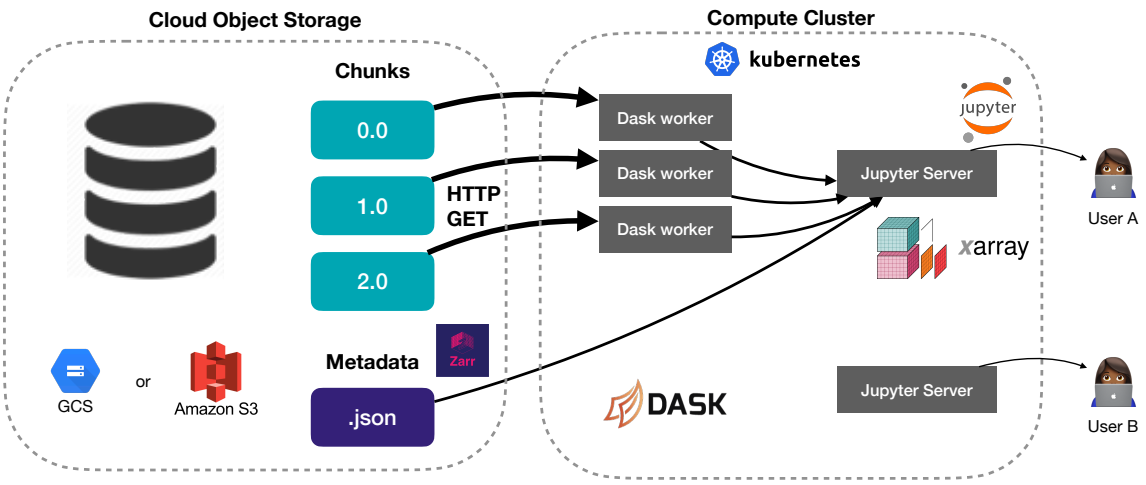

Figure 2. Pangeo architecture diagram. The data repository is hosted in cloud object storage (left), in the Zarr format. Compute nodes inside a Kubernetes cluster (right) fetch data and metadata from the object store. Users connect to the system via Jupyter and write interactive data analysis code in Xarray, which dispatches computations on an adaptively scaling Dask cluster.

To facilitate distributed computing, Pangeo Cloud also includes a deployment of Dask Gateway (https://gateway.dask.org/), a secure, multi-tenant server for managing Dask clusters. Users can provision personal Dask clusters which can be used to parallelize processing of data. Pangeo workloads tend to be heavily I/O bound, and this parallelization enables users to take advantage of high throughput of cloud object storage; each worker pulls its data directly from object storage, with Zarr chunks providing a natural unit of parallelization. The various components of this configuration are illustrated in figure 2. Typically users will scale up a Dask cluster, perform a reduction operation (e.g. mean or standard deviation) on a large dataset, and then switch to local analysis mode for the final visualization and interpretation steps.

Measuring Data Throughput

A central motivation for creating cloud-native data repositories is the desire to overcome the performance limitations of the download model. In Pangeo Cloud, data are not downloaded but rather streamed directly from object storage to compute nodes within the same cloud region. To quantify the speedup this architecture provides, in figure 3 we show results from a benchmarking exercise, adopted from https://github.com/earthcube2020/ec20_abernathey_etal. We measured the rate, in MB/s, at which data can be transferred from storage services to the Dask compute cluster, as a function of the number of distributed read processes. All benchmarking was performed in the GCP US-CENTRAL1 region. We compared four modes of data access:

1) Loading data from an ESGF Open-source Project for a Network Data Access Protocol (OPeNDAP) server. OPeNDAP is a mature and widely used data exchange API in the geosciences [14]. However, this is not a cloud-based service, so it serves as a baseline against which to compare newer technologies.
2) NetCDF-4 (HDF5) files placed directly in Cloud Object Storage. This option is useful to quantify how a legacy format performs in the cloud.
3) Zarr stored in GCS
4) Zarr stored in the Open Storage Network (OSN) S3-compatible object storage service.

First we note that the OPeNDAP server was both the slowest and the least scalable, saturating throughput at just over 100 MB/s. This is unsurprising, given the fact that the OPeNDAP service is likely backed by a single server and fixed network pipe. The fastest throughputs were

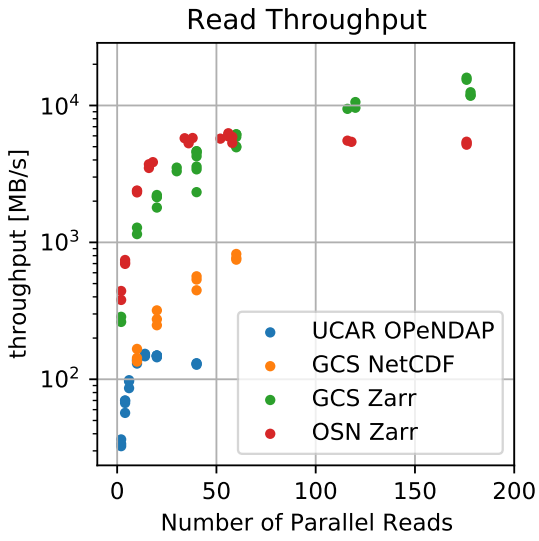

Figure 3. Read throughput of different data access methods. Source data available at https://zenodo.org/record/3829032. Code available at http://gallery.pangeo.io/repos/earthcube2020/ec20_abernathey_etal/cloud_storage.html.

obtained with the cloud-optimized format Zarr on OSN and GCS, both achieving throughputs in excess of 1 GB/s. Interestingly, for modest levels of parallelism (< 50 cores), OSN was actually faster than GCS, despite the data living outside of GCP. However, OSN eventually saturated, while GCS continued to scale, albeit at a slower rate, as more nodes were added. The legacy format (NetCDF-4/HDF5) on object storage also displayed decent scaling within the measured range, but overall speeds were an order of magnitude slower than Zarr. This highlights the value of cloud-optimized formats when working with object storage.

Overall, these results show how Pangeo's implementation of a cloud-native data repository and data-proximate computing has the potential to process data at much faster rates than workflows confined to personal computers. The fastest SSDs on personal computers can deliver throughputs of around 500 MB/s. With a modest number of distributed compute nodes, Pangeo Cloud users can process data at many GB/s. Moreover, with this approach, no data need be downloaded and only one copy of the data need be stored.

Conclusions and Future Challenges

The success of cloud public dataset programs, and the potential of nascent efforts such as Pangeo's cloud data library, shows that cloud-native data repositories can be a viable path forward to help data-intensive scientific fields overcome current infrastructure challenges. Cloud-native scientific platforms require both ARCO data as well as scalable data-proximate computing. However, one service provider need not be responsible for both. We argue for decoupling the storage of ARCO data from the data-proximate computing components. These services should be interoperable yet independent.

Despite the potential of cloud-native data repositories to accelerate scientific discovery, we see several challenges that will limit their adoption in the near future. A central challenge is *funding*. Academic research institutions and funding systems are not accustomed to paying for cloud computing. Structural factors, such as indirect cost policies and the three-year grant funding cycle, favor capital expenditure for equipment over the operational expenditure of cloud computing. Moreover, the potential cost savings associated with cloud-native repositories—namely, the elimination of duplicate local copies of big datasets and associated local computing resources—can only be realized by aggregating over many research groups. The process of moving to the cloud-native model can therefore be thought of as a phase transition to a lower-cost, higher-productivity state for an entire field. Overcoming the activation energy to catalyze this phase transition, however, may require substantial long-term investments and encouragement from funding agencies. We believe that the separation of concerns between cloud-native data repositories and data-proximate computing services will help simplify this transition.

Another challenge is *user education and training*. A truly cloud-native data science workflow will not look exactly like a local one. While novices can learn cloud-native practices from the beginning of their training, more experienced users will have to unlearn certain familiar patterns, such as reliance on local filesystem paths for data storage. Distributed computing tools like Dask can be extremely powerful, but efficient

use of these tools requires learning new concepts. Even experienced HPC users, accustomed to working with parallel processing, may feel confused by the cloud-native concepts of elastic scaling and serverless computing. Likewise, university and lab IT staff are relatively unfamiliar with the administration and management of cloud computing (in comparison to running local servers), and this too requires new training. Coordinated efforts are required to address these educational needs in order to realize the potential of cloud computing for science.

Finally, cloud-native data repositories will have to contend with special challenges around *data access and privacy*. It is perhaps unsurprising that fields like climate science and astronomy have been early adopters of cloud-based data sharing; the data in these fields are mostly open access and without privacy restrictions. Therefore, data can simply be made public and open to all. Without flexible open tools for access control management of cloud-based data, fields with strong privacy concerns will likely be driven towards proprietary, vertically-integrated solutions that are not modular or portable. While cloud providers have powerful permissions functionality, some technical development (and commensurate agency funding) will be required to extend the approaches described here for managing access to less open data sets.

Despite these challenges, we remain very enthusiastic about the potential of cloud computing to transform scientific research in data-intensive fields. We hope that, by helping to define best practices for cloud-native data repositories, we have made a small contribution towards building the infrastructure we need to tackle the some of the great scientific challenges of the future.

ACKNOWLEDGMENT

Abernathey, Lepore, and Crone, and Henderson acknowledge support from NSF award ICER-2026932 and the Gordon and Betty Moore Foundation. Gentemann received support from the Inter-agency Implementation and Advanced Concepts Team (IMPACT) at NASA's Marshall Space Flight Center.

Disclaimer: Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

REFERENCES

1. V. Balaji, K. E. Taylor, M. Juckes, B. N. Lawrence, P. J. Durack, M. Lautenschlager, C. Blanton, L. Cinquini, S. Denvil, M. Elkington, and et al., "Requirements for a global data infrastructure in support of CMIP6," *Geoscientific Model Development*, vol. 11, no. 9, p. 3659–3680, Sep 2018. [Online]. Available: http://dx.doi.org/10.5194/gmd-11-3659-2018
2. N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone," *Remote Sensing of Environment*, vol. 202, p. 18–27, Dec 2017. [Online]. Available: http://dx.doi.org/10.1016/j.rse.2017.06.031
3. J. de La Beaujardière, "A geodata fabric for the 21st century," *Eos*, vol. 100, Nov 2019. [Online]. Available: http://dx.doi.org/10.1029/2019EO136386
4. D. Medvedev, G. Lemson, and M. Rippin, "Sciserver compute: Bringing analysis close to the data," in *Proceedings of the 28th International Conference on Scientific and Statistical Database Management - SSDBM '16*. ACM Press, 2016. [Online]. Available: http://dx.doi.org/10.1145/2949689.2949700
5. P. G. Brown, "Overview of scidb," in *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*. ACM Press, 2010. [Online]. Available: http://dx.doi.org/10.1145/1807167.1807271
6. P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, "The multidimensional database system Rasdaman," in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data - SIGMOD '98*. ACM Press, 1998. [Online]. Available: http://dx.doi.org/10.1145/276304.276386
7. M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, and et al., "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, Mar 2016. [Online]. Available: http://dx.doi.org/10.1038/sdata.2016.18
8. C. Holmes, "Analysis ready data defined," Apr 2018. [Online]. Available: https://medium.com/planet-stories/analysis-ready-data-defined-5694f6f48815
9. E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proceedings of the 2017 Symposium on Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2017, p. 445–451. [Online]. Available: https://doi.org/10.1145/3127479.3128601

10. M. Ramamurthy, “Geoscience cyberinfrastructure in the cloud: Data-proximate computing to address big data and open science challenges,” in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/eScience.2017.63

11. T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, “Jupyter notebooks – a publishing format for reproducible computational workflows,” in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

12. G. Auwera and B. D. O’Connor, *Genomics in the Cloud*. O’Reilly Media, Inc., 2020.

13. R. P. Signell and D. Pothina, “Analysis and visualization of coastal ocean model data in the Cloud,” *Journal of Marine Science and Engineering*, vol. 7, no. 4, pp. 110–121, Apr 2019. [Online]. Available: https://doi.org/10.3390/jmse7040110

14. P. Cornillon, J. Gallagher, and T. Sgouros, “Opendap: Accessing data in a distributed, heterogeneous environment,” *Data Science Journal*, vol. 2, pp. 164–174, 2003.

Ryan Abernathey is an Associate Professor in the Dept. of Earth and Environmental Science at Columbia University. Contact him at rpa@ldeo.columbia.edu.

Tom Augspurger is a software engineer at Anaconda, Inc. Contact him at tom.w.augspurger@gmail.com.

Anderson Banihirwe is a software engineer at the National Center for Atmospheric Research. Contact him at abanihi@ucar.edu.

Charles C. Blackmon-Luca is a software engineer at the Lamont–Doherty Earth Observatory of Columbia University. Contact him at blackmon@ldeo.columbia.edu

Timothy J. Crone is a Lamont Associate Research Professor at the Lamont–Doherty Earth Observatory of Columbia University. Contact him at tjc@ldeo.columbia.edu.

Chelle Gentemann is a Senior Scientist at the Farallon Institute. Contact her at cgentemann@faralloninstitute.org.

Joe Hamman is Technology Director at CarbonPlan and a project scientist at NCAR. Contact him at joe@carbonplan.org.

Naomi Henderson is a Research Scientist at the Lamont–Doherty Earth Observatory of Columbia University. Contact her at naomi@ldeo.columbia.edu.

Chiara Lepore is an Associate Research Scientist at the Lamont–Doherty Earth Observatory of Columbia University. Contact her at clepore@ldeo.columbia.edu.

Theo A. McCaie is Data Engineering Research Lead at the Joint Centre for Excellence in Environmental Intelligence. Contact him at theo.mccaie@informaticslab.co.uk.

Niall H. Robinson is Research and Applications Lead at the Joint Centre for Excellence in Environmental Intelligence, where he is affiliated to the UK Met Office and the University of Exeter. Contact him at niall.robinson@informaticslab.co.uk.

Richard P. Signell is a Research Oceanographer at the US Geological Survey, Woods Hole Coastal and Marine Science Center. Contact him at rsignell@usgs.gov.