ESCO Cloud Analytics Notepad

Capturing ideas and snippets of information about cloud analytics.

Problem Statements

Asynchronous services and workflows

Traditional data and service endpoints have been fairly static. Archives serve data from a generally predefined set of products that are either fixed or growing over time. Services are developed and published and are expected to be available for long time periods. How can we adapt to be able to quickly provide access to a more fluid pool of data that is being fed by new processing made possible by the cloud. How can services be extended to include very long-running jobs such as when we're aggregating results.

Paradigm Changes

Do scientists have to develop new mental models of how data are processed to make better use of the cloud environment? How much can (or should) the intricacies of distributed data analytics be hidden behind facades?

Assimilate new data

How do you feed new data into tools and workflows that have not been used for that kind of data before? What data formats, metadata, data structures, coordinate representations (time, space, spectral), or ancillary variables are needed?

Congruent spatio-temporal views

How can we provide views of data from multiple sources in a way that consumers of the data see a uniform view? Views can be pre-built, such as with datacubes, but can also be computed as needed.

Solution Matrix

A table showing the problem statements from above and the building blocks from below. This is an experimental presentation that is likely to be superseded by a better way of matching building blocks to problem statements.

	Asynchronous services & workflows	Paradigm changes	Assimilate new data	Congruent spatio-temporal views
WCS 2.0				
WCS 2.1				
WCS-T				
OPeNDAP				
Open Data Cube				
WPS 2.0				
WCPS 1.0				

WPS-T				
Common Data Model				
Cloud Optimized GeoTIFF				
EO JSON				
OGC CIS 1.0/1.1				
OGC DGGS				
xarray				
dask				
dask.distributed				
daskernetes				
PyTables				
Jupyter Notebooks				
STAC				
OpenAPI / Swagger				

Building blocks

(Open to suggestions about better categories or names of categories!)

Data Access

- OGC WCS 2.0 - multi-dimensional coverage data access over the Internet (using OGC CIS 1.0)
- OGC WCS 2.1 - provides access to OGC CIS 1.1 data (adds irregular grids, different internal partitioning to accommodate new access patterns, adds JSON and RDF representation.
- OGC WCS-T - defines an extension to the WCS Core for updating coverage offer-ings on a server
- OPeNDAP - discipline-neutral means of requesting and providing data across the World Wide Web
- Open Data Cube - time-series multi-dimensional (space, time, data type) stack of spatially aligned pixels ready for analysis

Data Processing Services

- OGC WPS 2.0 - rules for standardizing inputs and outputs for geospatial processing services
- OGC WCPS 1.0 - protocol-independent language for the extraction, processing, and analysis of multi-dimensional coverages representing sensor, image, or statistics data.
- OGC WPS-T - [preliminary description] extends OGC WPS with two new operations:DeployProcess and UndeployProcess

Data Models & Formats

- Common Data Model - Unidata's abstract data model for scientific datasets, merges netCDF, HDF5, and OPeNDAP data models
- Cloud Optimized GeoTIFF (COG) - GeoTIFF with internal organization that enables more efficient workflows on the cloud via HTTP GET range requests
    - Online COG validator
- EO JSON - a number of efforts to develop JSON specs for coverage data
- OGC CIS 1.0 and 1.1 - Coverage Implementation Specification
- OGC DGGS - Discrete Global Grid Systems, spatial reference system that uses a hierarchical tessellation of cells to partition and address the globe

Data Libraries

- xarray - toolkit for analytics on multi-dimensional arrays for pandas
- dask - flexible parallel computing library for analytic computing

Workflow and orchestration

- dask.distributed - lightweight library for distributed computing in Python. It extends both the concurrent.futures and dask APIs to moderate sized clusters
- daskernetes - deploys Dask workers on Kubernetes clusters using native Kubernetes APIs. It is designed to dynamically launch short-lived deployments of workers during the lifetime of a Python process.

Visualization & Interaction

- PyTables - built on top of the HDF5 library, tool for interactively browsing, processing and searching very large amounts of data
- Jupyter Notebooks - Interactive code execution and visualization

Metadata & Catalogs

- SpatioTemporal Asset Catalog (STAC) - expose Earth observation data as spatiotemporal asset catalogs (possible on-the-fly catalog for cloud pipelines?)

Interoperability Tools

- OpenAPI initiative - standardizing how to describe REST APIs (based on swagger)

References / Links

Other work

NASA

- Cumulus - Cloud-based data ingest, archive, distribution and management system for EOSDIS

Other

- Pangeo - an experimental deployment of JupyterHub, Dask, and XArray on Google Container Engine (GKE) to support atmospheric and oceanographic data analysis on large datasets

Tutorials & Articles

Tutorials

- How to Cloud for Scientists Webinar - YouTube, 51 minutes, Chris Lynnes - surveys a wide variety of cloud computing services that can be leveraged for Earth Observation data analysis
- Cloud Optimized GeoTiff Map Experiment - running on AWS Lambda
- Cloud Native Geospatial - Chris Holmes blog posts on cloud hosted geospatial
    - Some of the posts:
    - Planet's Cloud Native Geospatial Architecture
    - Open Aerial Map's Cloud Native Geospatial Architecture
    - Cloud Native Geospatial Architecture Defined
- AWS explained: the basics - Brief explanation of cloud, Amazon Web Services (AWS)

Articles

- *Fostering Cross-Disciplinary Earth Science Through Datacube Analytics,* P Baumann et al, 2018 - Abstract, Chapter PDF
- *Archive Management of NASA Earth Observation Data to Support Cloud Analysis*, C Lynnes, K Baynes, M McInerney, 2017 - PDF

Questions

- How does one do interprocess communication in the cloud? In the old days there was Shared Memory, Pipes, Sockets, and Files. In the cloud it seems there's primarily HTTP (and maybe sockets) or files.
- How do you decide when it's better to write out a file so you can stop running a process that's costing per minute vs. holding things in memory so you don't incur storage charges?