Physical Oceanography Distributed Active Archive Center

PO.DAAC Web Service Modernization

Myche McAuley, Michael Gangl, Stepheny Perez, Suresh Vannan

Contact: michael.e.gangl@jpl.nasa.gov

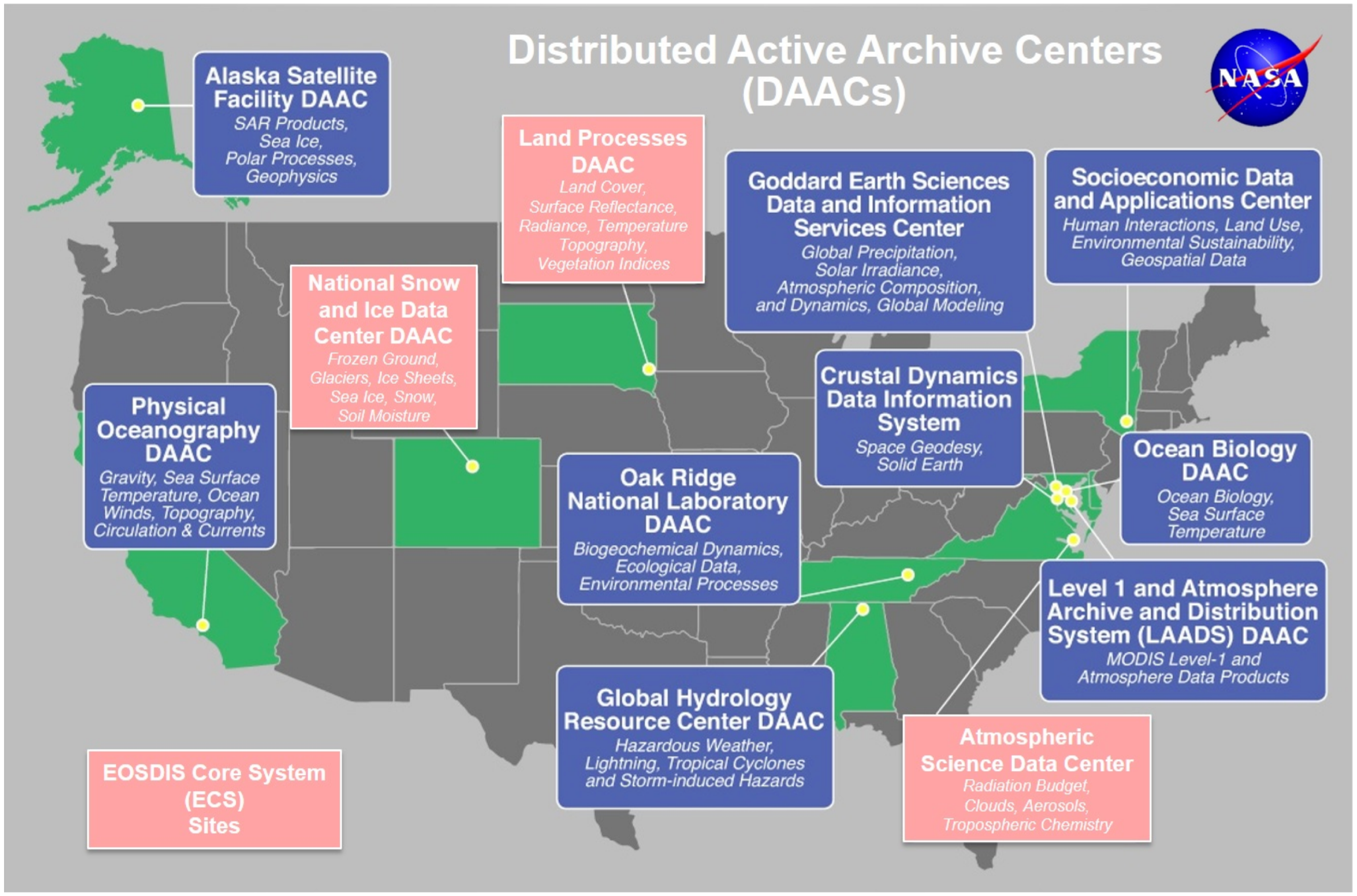
November 5th, 2019
Planetary Data System MC 2019
Flagstaff, Arizona

Caltech

Overview

- Earthdata And the DAACs
- Modernization
    - Why modernize?
    - Emphasis on APIs
    - Architecture / Microservices
- Lessons Learned

Earthdata And the DAACs


Distributed Active Archive Centers (DAACs)
NASA
Alaska Satellite Facility DAAC
SAR Products, Sea Ice, Polar Processes, Geophysics
Land Processes DAAC
Land Cover, Surface Reflectance, Radiance, Temperature Topography, Vegetation Indices
Goddard Earth Sciences Data and Information Services Center
Global Precipitation, Solar Irradiance, Atmospheric Composition, and Dynamics, Global Modeling
Socioeconomic Data and Applications Center
Human Interactions, Land Use, Environmental Sustainability, Geospatial Data
National Snow and Ice Data Center DAAC
Frozen Ground, Glaciers, Ice Sheets, Sea Ice, Snow, Soil Moisture
Crustal Dynamics Data Information System
Space Geodesy, Solid Earth
Physical Oceanography DAAC
Gravity, Sea Surface Temperature, Ocean Winds, Topography, Circulation & Currents
Oak Ridge National Laboratory DAAC
Biogeochemical Dynamics, Ecological Data, Environmental Processes
Ocean Biology DAAC
Ocean Biology, Sea Surface Temperature
Level 1 and Atmosphere Archive and Distribution System (LAADS) DAAC
MODIS Level-1 and Atmosphere Data Products
Global Hydrology Resource Center DAAC
Hazardous Weather, Lightning, Tropical Cyclones and Storm-induced Hazards
Atmospheric Science Data Center
Radiation Budget, Clouds, Aerosols, Tropospheric Chemistry
EOSDIS Core System (ECS) Sites

Why Modernize

- Management Reasons:
  - Maximize scientific impact
  - Attract new/best talent
    - developers = happy ? increase_productivity() : update_resumes()
  - Add services and functionality faster/cheaper
  - Reduce cost due to egress from cloud
  - Attract proposals, work, new datasets to the DAAC(s)
  - Better adapt the next generation of change
- User Reasons
  - **Reduce download time** from years to minutes - Data too large for average user (SWOT will be petabyte scale)
  - ***Consistent API*** within PO.DAAC and ultimately ALL DAACs

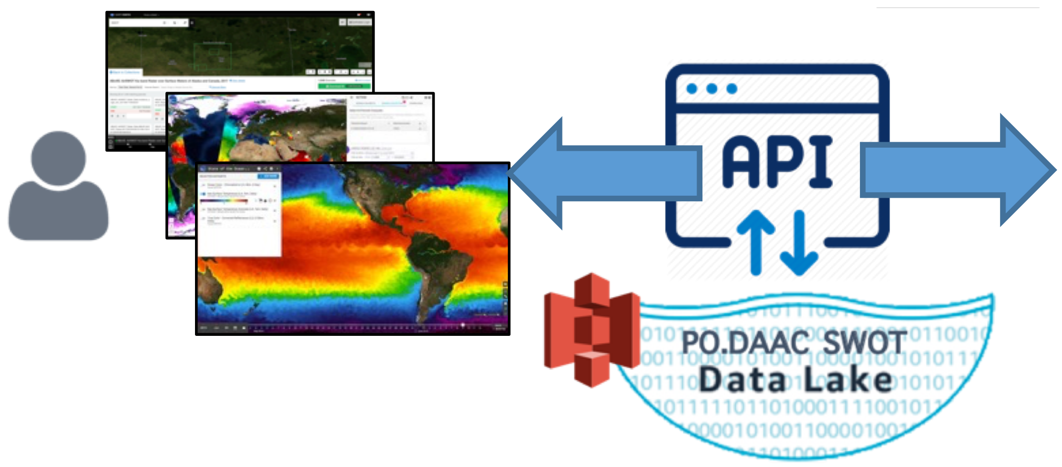
Why Modernize

- We should have been doing all of these things all along… but it’s hard without buy in from sponsors *(buy in = money = time = resources)*
- PO.DAAC’s (and other DAAC’s) move to the cloud affords us the time and resources to “do it right”
  - New technology stacks need services and tooling re-built or at least re-adapted
  - EOSDIS already implemented a common ingest and archive system, so it’s a common base to build off of

Emphasis on APIs

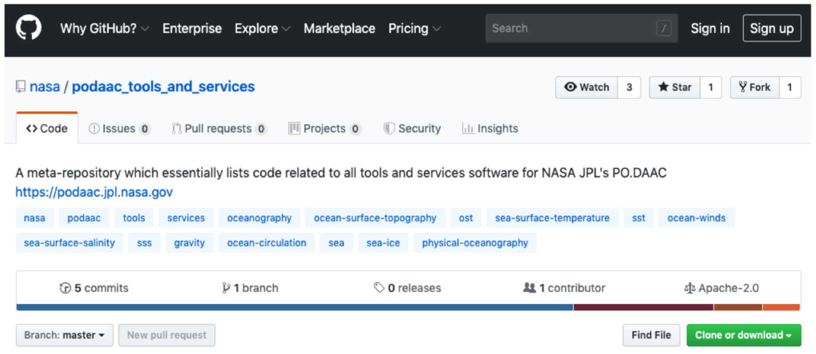
- OpenAPI spec for any public facing service!
  - Clearly state contract between user and service
  - Allows users to generate a client in “any” language they’d like
- Don’t hide: APIs become a first class citizens
  - Sing from the hills about your APIs.
- Leverage APIs for your own development
  - No shortcuts use the same APIs your users use
  - Tools (e.g. UIs, portals, websites) should use the same APIs. That is, don’t customize* services to their frontends

Emphasis on APIs



YouTube
jupyter
Webinar
User Handbook
Recipes

Github and Github forum for community adoption


Why GitHub? Enterprise Explore Marketplace Pricing
Search
Sign in Sign up
Watch 3 Star 1 Fork 1
Code Issues 0 Pull requests 0 Projects 0 Security Insights
Branch: master New pull request Find File Clone or download

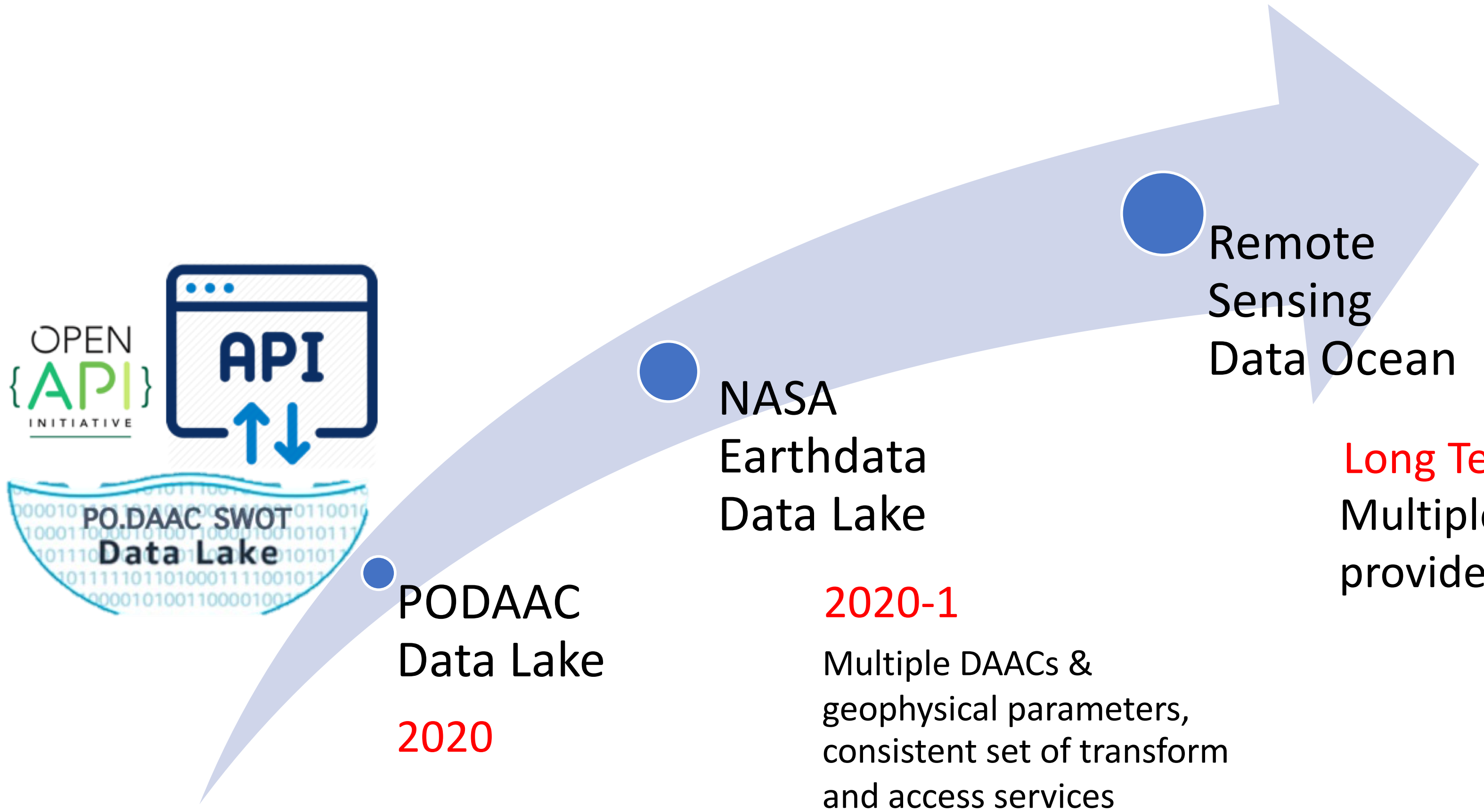
nasa / podaac_tools_and_services

A meta-repository which essentially lists code related to all tools and services software for NASA JPL's PO.DAAC
https://podaac.jpl.nasa.gov

nasa podaac tools services oceanography ocean-surface-topography ost sea-surface-temperature sst ocean-winds sea-surface-salinity sss gravity ocean-circulation sea sea-ice physical-oceanography

5 commits · 1 branch · 0 releases · 1 contributor · Apache-2.0

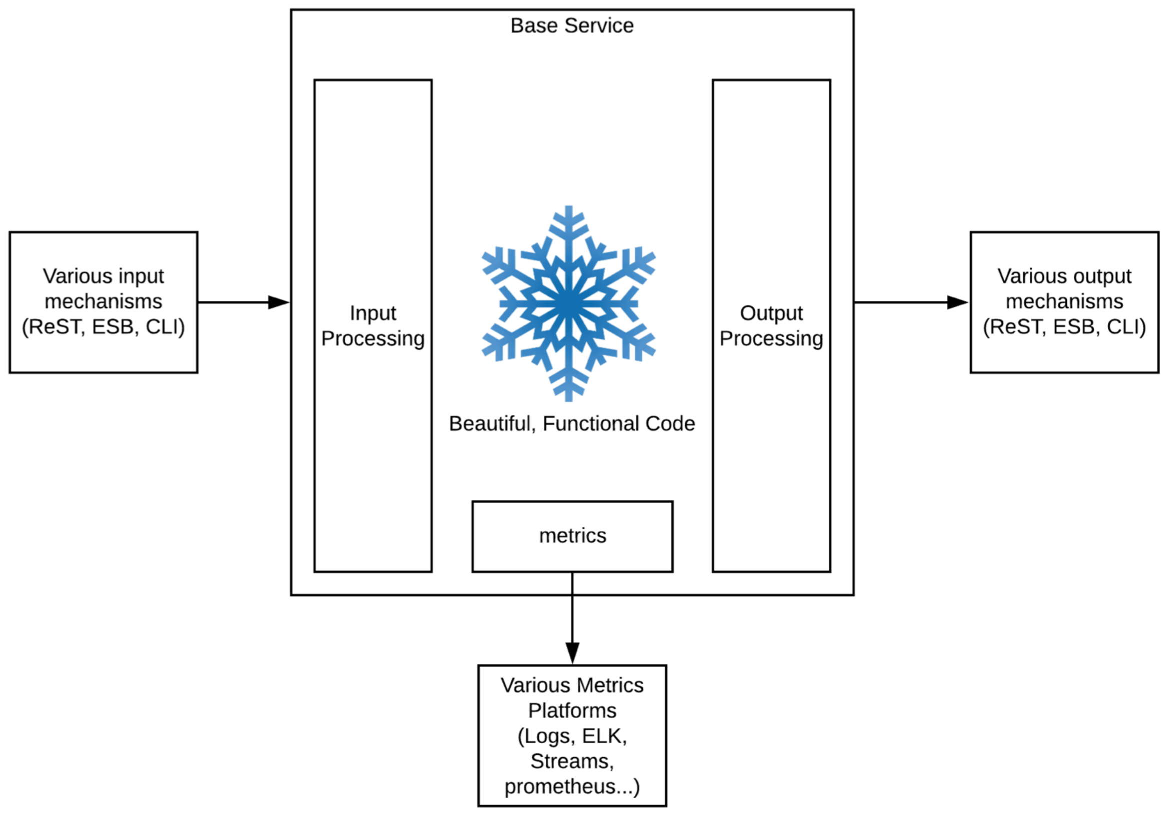
Emphasis on APIs



Architecture / Microservices

- PO.DAAC is moving towards a microservice architecture
  - This may or may not be right for you.
  - This may or may not be right for you.
  - Reuse deployment, test frameworks, build pipelines, etc.
  - Reduce complexity and time of delivering functionality to the end user
  - Reduce the complexity and time of getting developers trained, started, and outputting useful products *(while learning amaaaaazing development practices)*

Architecture / Microservices


Base Service
Various input mechanisms (ReST, ESB, CLI)
Input Processing
Beautiful, Functional Code
metrics
Output Processing
Various output mechanisms (ReST, ESB, CLI)
Various Metrics Platforms (Logs, ELK, Streams, prometheus...)

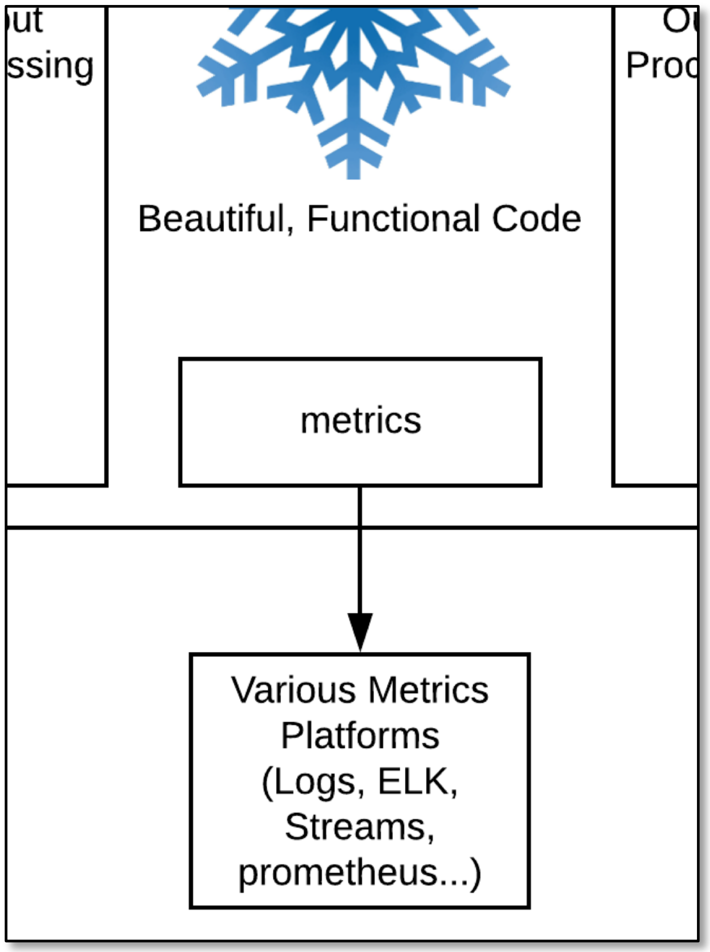
Choose your own metaphor

Each development snowflake is essentially a beautiful, Eden-esque courtyard surrounded by a standardized functional-yet-unsexy wrapper that can best be defined as embracing the brutalism aesthetic

We're cranking out assembly line car chassis with customizable engines. At the end of the day, you can get in the car and know how the gas pedal (development) , brakes (test… haha), steering wheel (deployment) are all expected to work, regardless of the engine in the car.

Services – Measure Everything

- Metrics are built in to everything we do
  - Processes (timing of any major operation)
  - Web requests (who what where when)
  - Test and Deployment times
  - Development (Linting, coverage, bug reports)
- Log everything and then build the query mechanism to get *meaningful data. Adapt as needed.*


ut
ssing
Beautiful, Functional Code
Ou
Proc
metrics
Various Metrics Platforms (Logs, ELK, Streams, prometheus...)

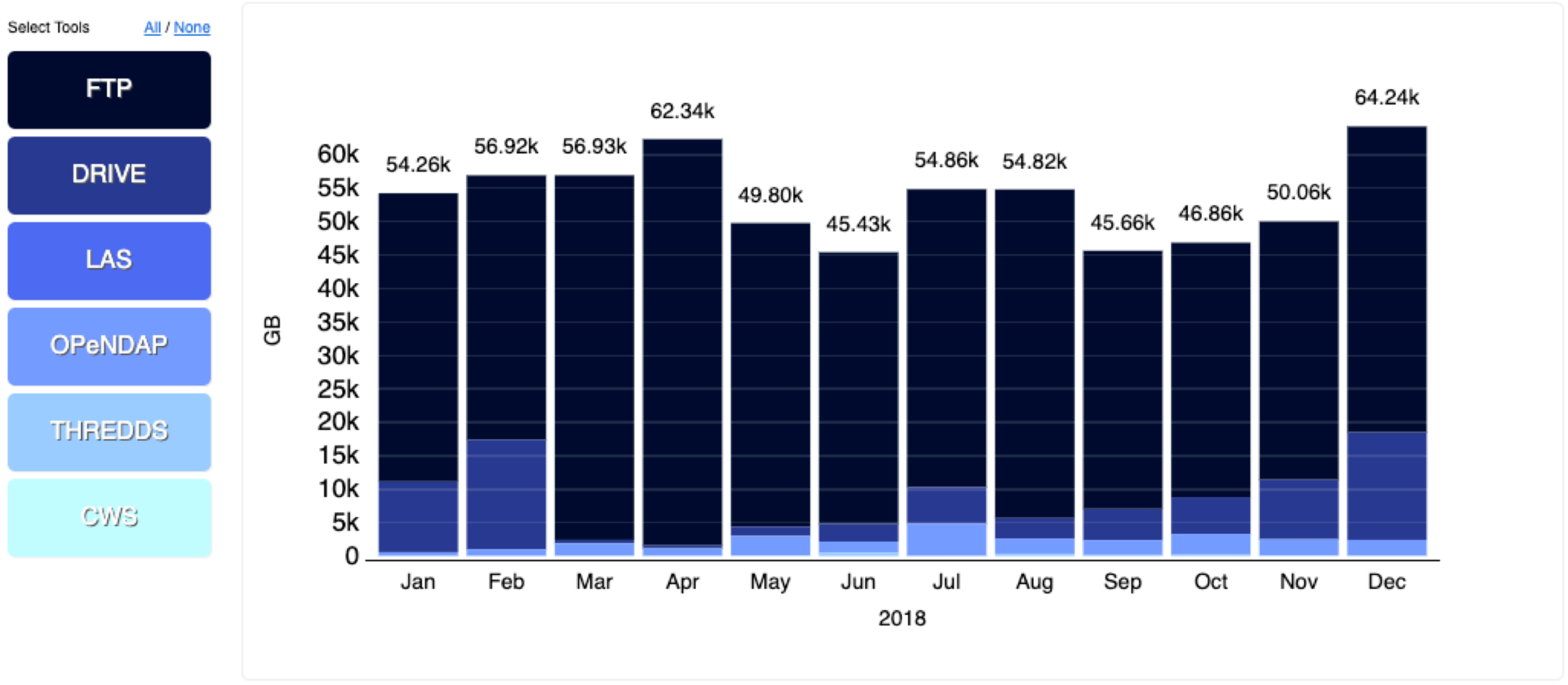
Lessons Learned

- Need buy in from Stakeholders – Sponsors, managers, developers
- “What can be done” and “what can be done by your team” are different things!
- Cultural Change is hard
  - Distribute accountability for success
  - Bring in eager talent

Backup

The way things were

- The way things were
  - Majority of users used FTP for whole file access (FTP was replaced by ‘Drive’ in 2019)



TWTW - The way things were

- Second were standard access technologies like OPeNDAP and THREDDS
  - OPENDAP: Variable level subsetting by space, time
  - THREDDS: Aggregations across multiple files (trends)
  - Both of these help to reduce the volume of files downloaded (**Spoiler:** this will be important later).
  - 30% of users utilize these technologies

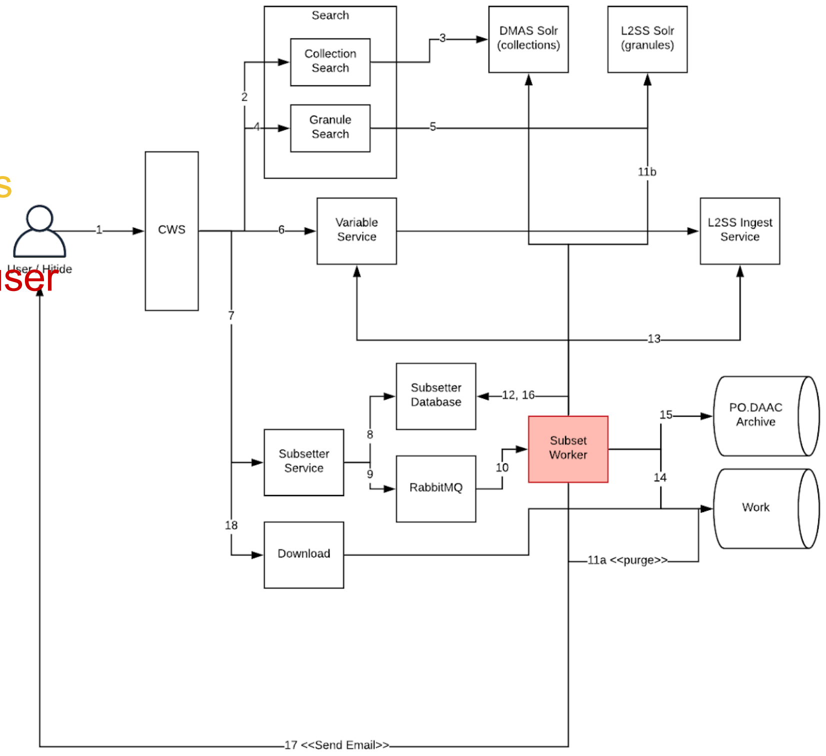
TWTW: Ad Hoc Services

- Services come organically to PO.DAAC
  - E.g. We need something better than OPeNDAP for level2 subsetting, let's build one ourselves.
  - E.g. We need a way of searching on swath data, assuming bounding boxes are accurate is wrong.
- Services are "thrown over the wall" a lot of proposal works ends up dropped in our lap and we need to integrate it.
- This leads to… specialized APIs, no cohesive vocabulary, services are difficult to integrate with existing tools/UIs, users need to learn different APIs not only across DAACs, but within DAACS as well!

TWTW – Coarse Metrics

- High level metrics
  - Archive size
  - Distribution volume, distribution by tool
  - Some web metrics
- Useful for reporting to our sponsor, **but not deep diving…**
  - What datasets are users using in tandem?
  - What type of user is using the API vs the UI
  - How has a change to some code affected our performance (new library, new AMI, etc)

L2SS Current Design



- Movement of data
- Dynamicity of work flows
- Monitoring and Logging
- Synchronous vs Asynchronous
- Separation of concerns
- Simple/Common interface to user
- -ilities:
    1. Scalability,
    2. Maintainability,
    3. Testability
    4. Measurability
- Core AND cloud side

- Complexity of system