



Smart On-Demand Analysis of Multi-Temporal and Full Resolution SAR ARDs in Multi-Cloud & HPC

AIST-18-0085

ESTF 2021

Thursday, July 1, 2021

Hook Hua¹, **Science Data System:** Gerald Manipon¹, Mohammed Karim¹, Marjorie Lucas¹, Zhangfan Xing¹, Joseph Jacob¹, Alex Dunn¹, Rishi Verma¹, Dustin Lo¹, Susan Neely¹

Flood and Damage Assessment: Sang-Ho Yun¹, Jungkyo Jung¹

Solid Earth Science: Eric Fielding¹, David Bekaert¹, Susan Owen¹

Machine Learning: Gian Franco Sacco

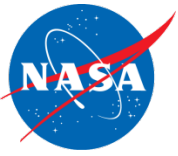
Student Intern: David Tran

¹Jet Propulsion Laboratory, California Institute of Technology



Background

- Motivation
 - *Increasing **gap** between SDS in cloud capability vs algorithm development needs*
 - SAR data can aid in **decision making** for floods, earthquakes, and other monitoring and response scenarios where rapid information for **situational awareness** is required.
 - Increasing international SAR observations
 - SAR intrinsically **high** data volume, compute, and variety of algorithm analysis methods.
- Analytic Collaborative Framework (ACF)
 - *Address disconnect between **algorithm development and large-scale Science Data Systems (SDSes)** in the cloud*
 - *Enables more rapid time to market from algorithm development to data product generation, production, validation*
 - *Facilitating algorithm development of **multi-temporal and full resolution SAR analysis***
 - *Prototype on-demand processing to “Analysis Ready Data (**ARD**)”-like data for SAR*

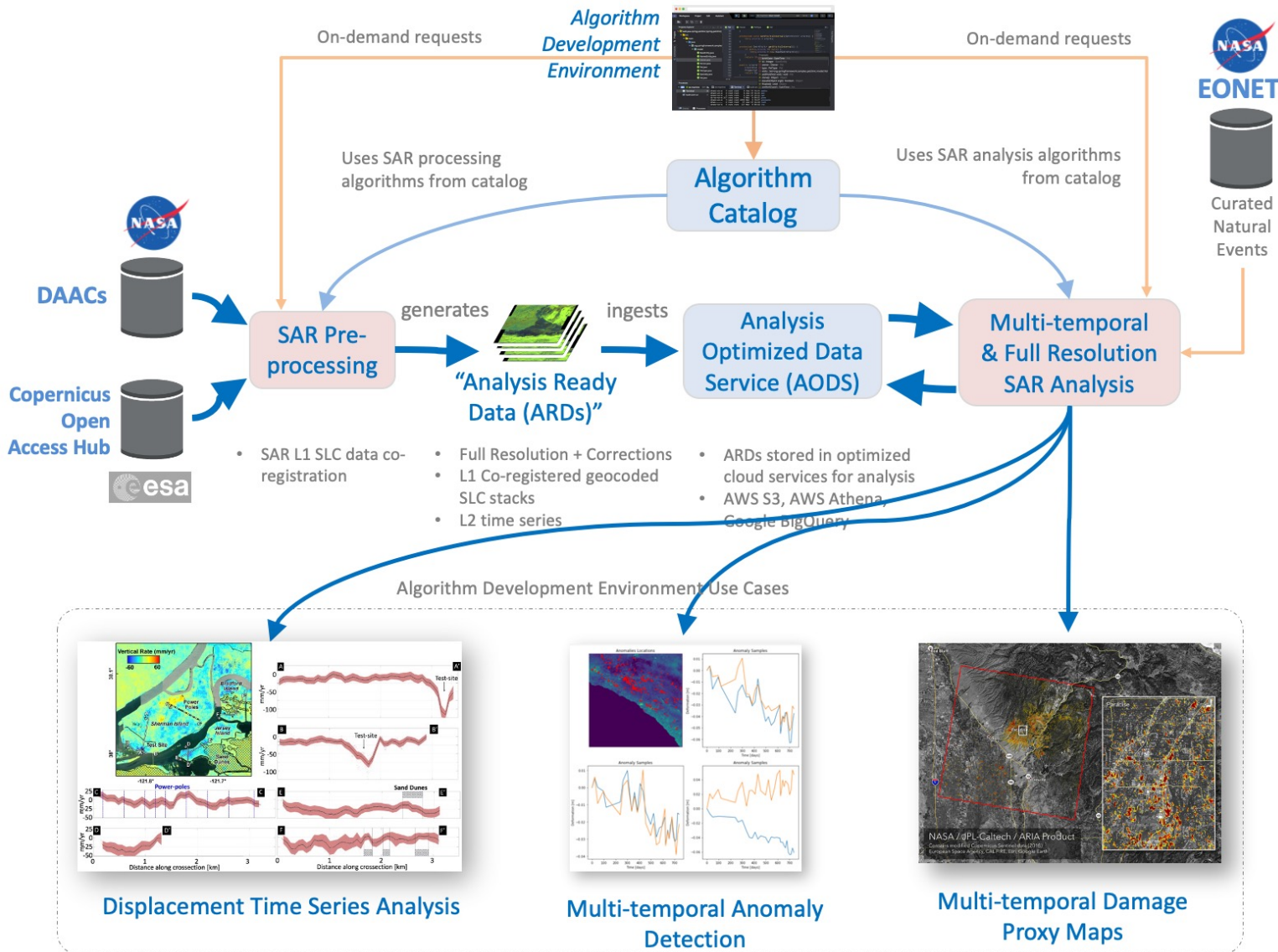


Objectives

- Address need for **rapid & scalable algorithm development** environment
 - Provides pathways for algorithms to **run at large-scale** science data systems and corresponding **efficient handling of voluminous datasets**.
- Increase **accessibility** of **multi-sensor SAR analysis** to users
 - Assess Analysis Ready Data (**ARD**)-like on-demand generation to ease SAR use
- Assess more more **cost-efficient** compute approaches for these larger L2 and L3 analysis, which is already becoming a bottleneck for effective algorithm development and analysis.
 - Demonstrate **multi-cloud** (AWS, Google Cloud Platform, Azure) and **NASA HEC** (Pleiades) approach to on-demand processing
 - Leverage **Machine Learning**-based cost optimization across multi-cloud



Objectives / Tech Advance



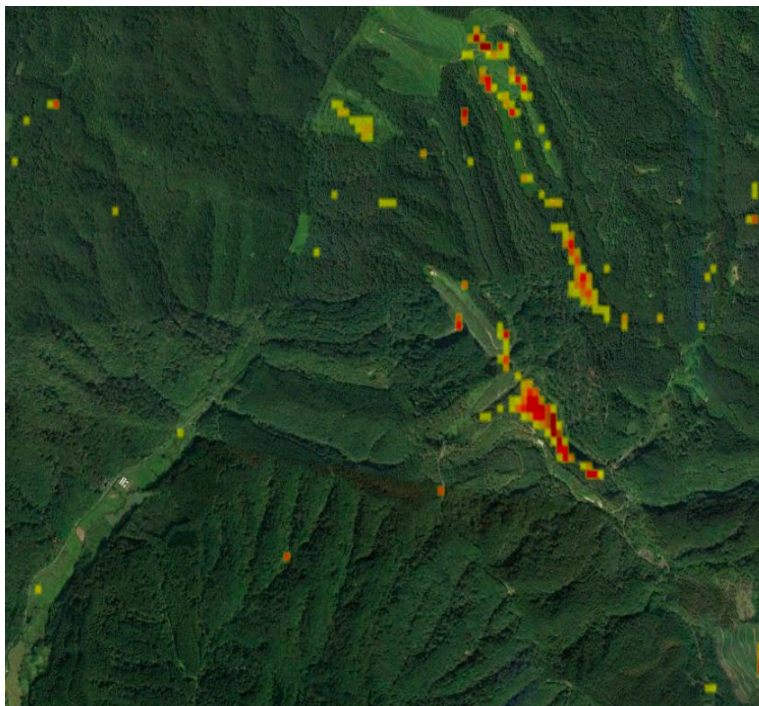


Need for Algorithm Development—*at Scale*

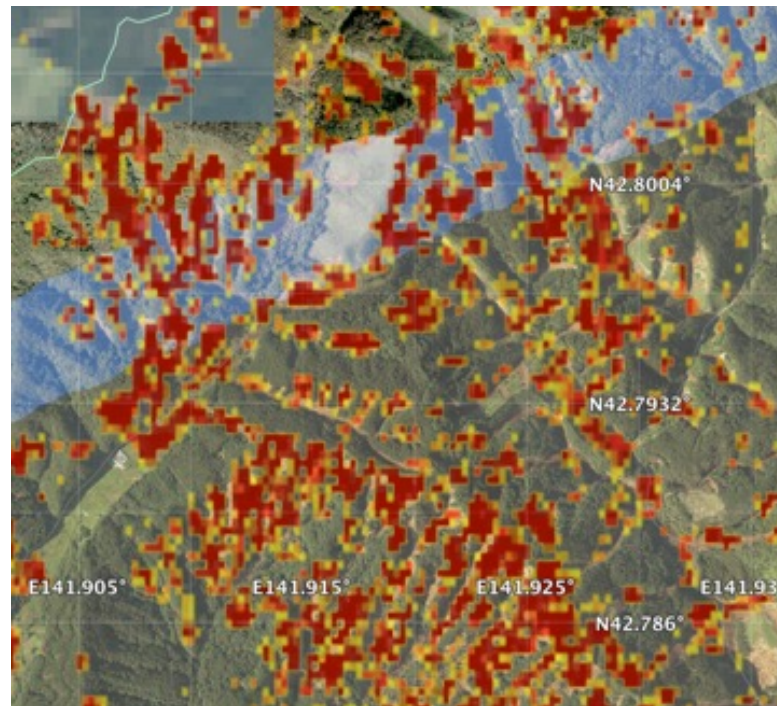
Source: Sang-Ho Yun, Jungkyo Jung

DPM1

DPM2/3



Before/After Scenes
Processing: 1 hour
"Downloading": 1.5 hours

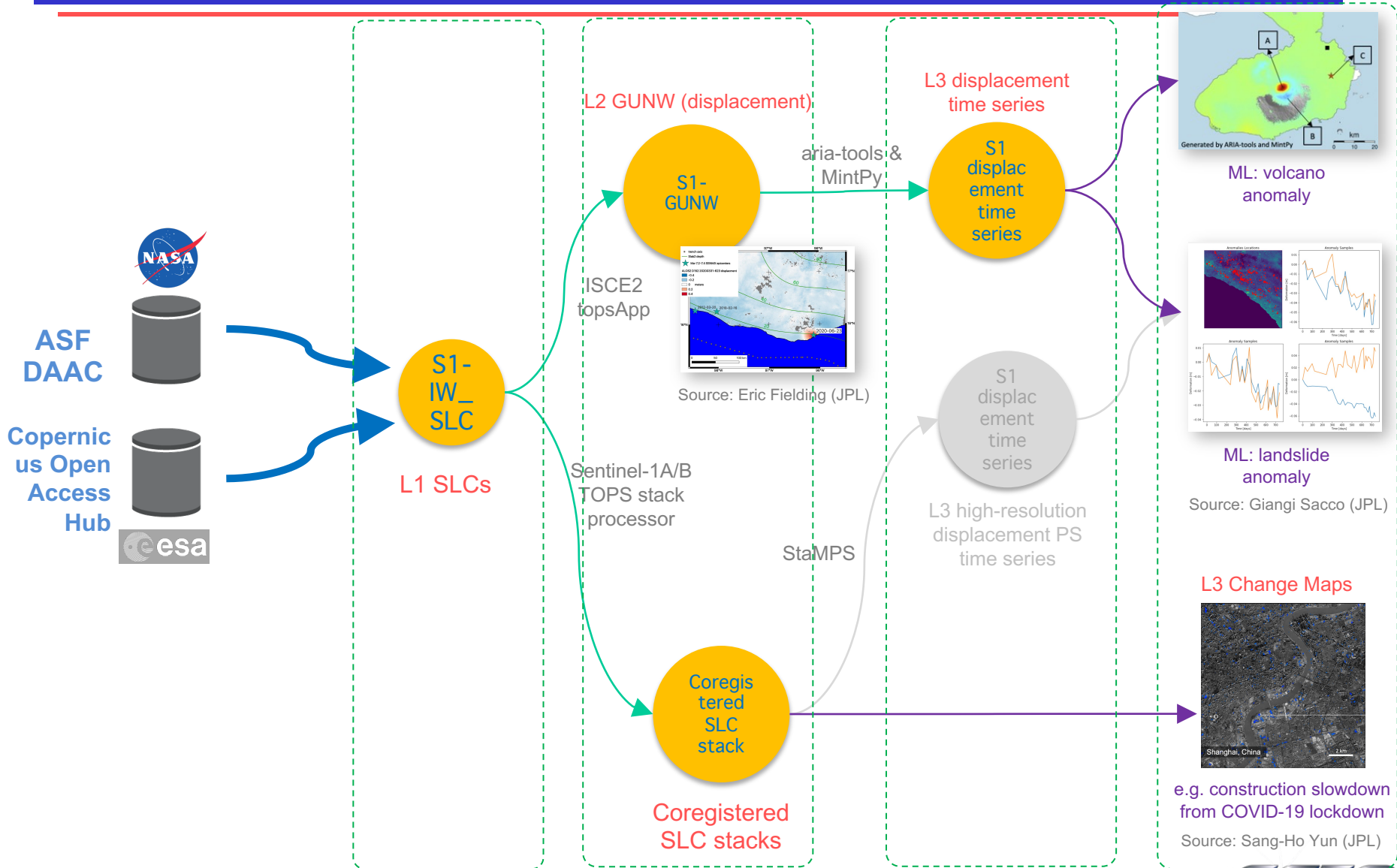


Time Series of Scenes
Processing: 26 days
"Downloading": 40 hours

Landslides Triggered by the M6.6 Hokkaido Earthquake (Sept 2018)



Example On-demand SAR Products and Analysis with Sentinel-1 A/B





NISAR and SWOT On-demand Needs

This AIST's technology demonstration is in alignment with NISAR and SWOT's on-demand needs :

1. Type A: “Tunable” On-Demand Processing

- *“Bring your own parameters” scenario*
- Trigger SDS to run standard product PGEs with custom tunable parameters.
 - Example: Re-run L2 GUNW generation but with nearest 3 neighbor pairing strategy (small-scale and large-scale processing in AWS).

2. Type B: Science Notebook Development Environment (for L1-L3 Cal/Val and ADT)

- *“Bring your own code” scenario*
- A Jupyter notebook algorithm development environment that is collocated with SDS
 - Example: Running ISCE3 in a Jupyter notebook next to L1 SLC data generated by SDS
- Running notebooks at-scale in SDS
 - Example: Running global biomass estimate using custom L2 biomass model

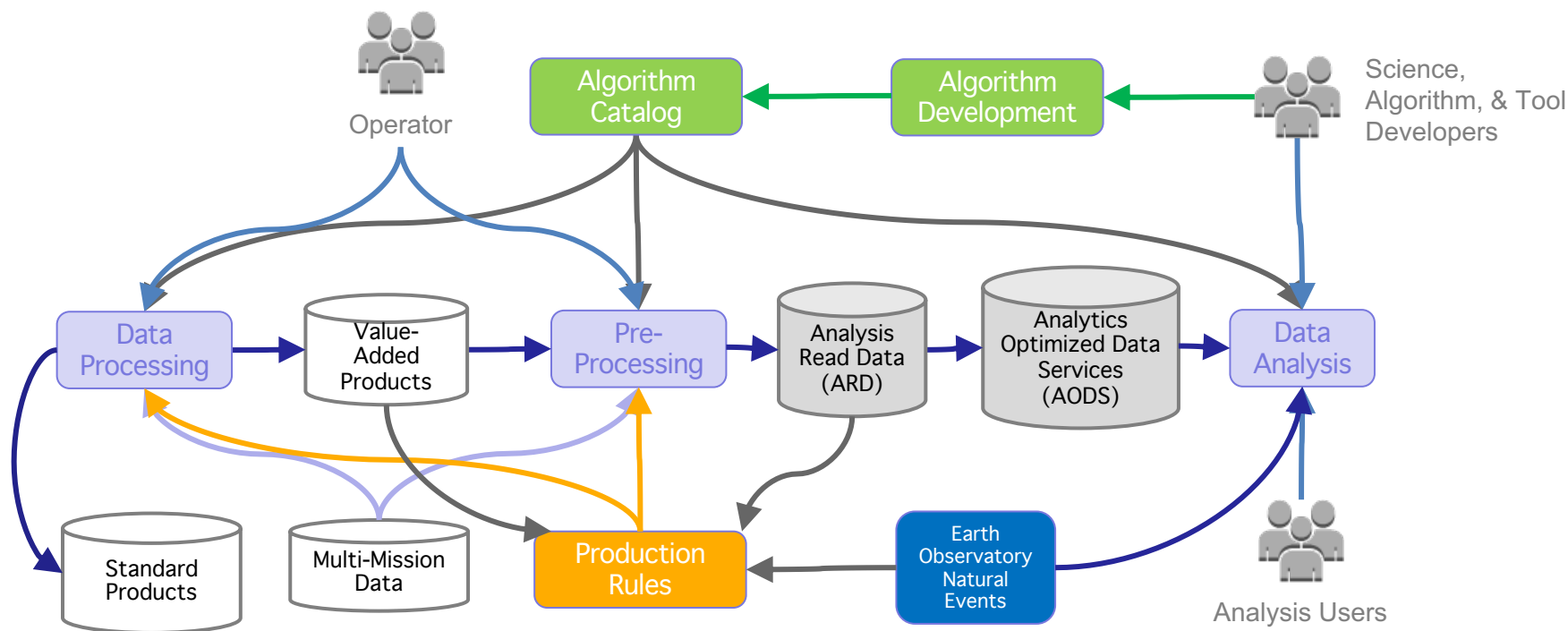
3. Type C: Automatic Generation of Custom Products in Keep-Up Mode

“Subscription” scenario

- Triggering your own code or custom parameters based on new data stream
- Allows custom code for urgent response and forward stream processing.
 - Example: Set up a variant of coherence change detection algorithm to run automatically for any new L1 SLC acquisitions.



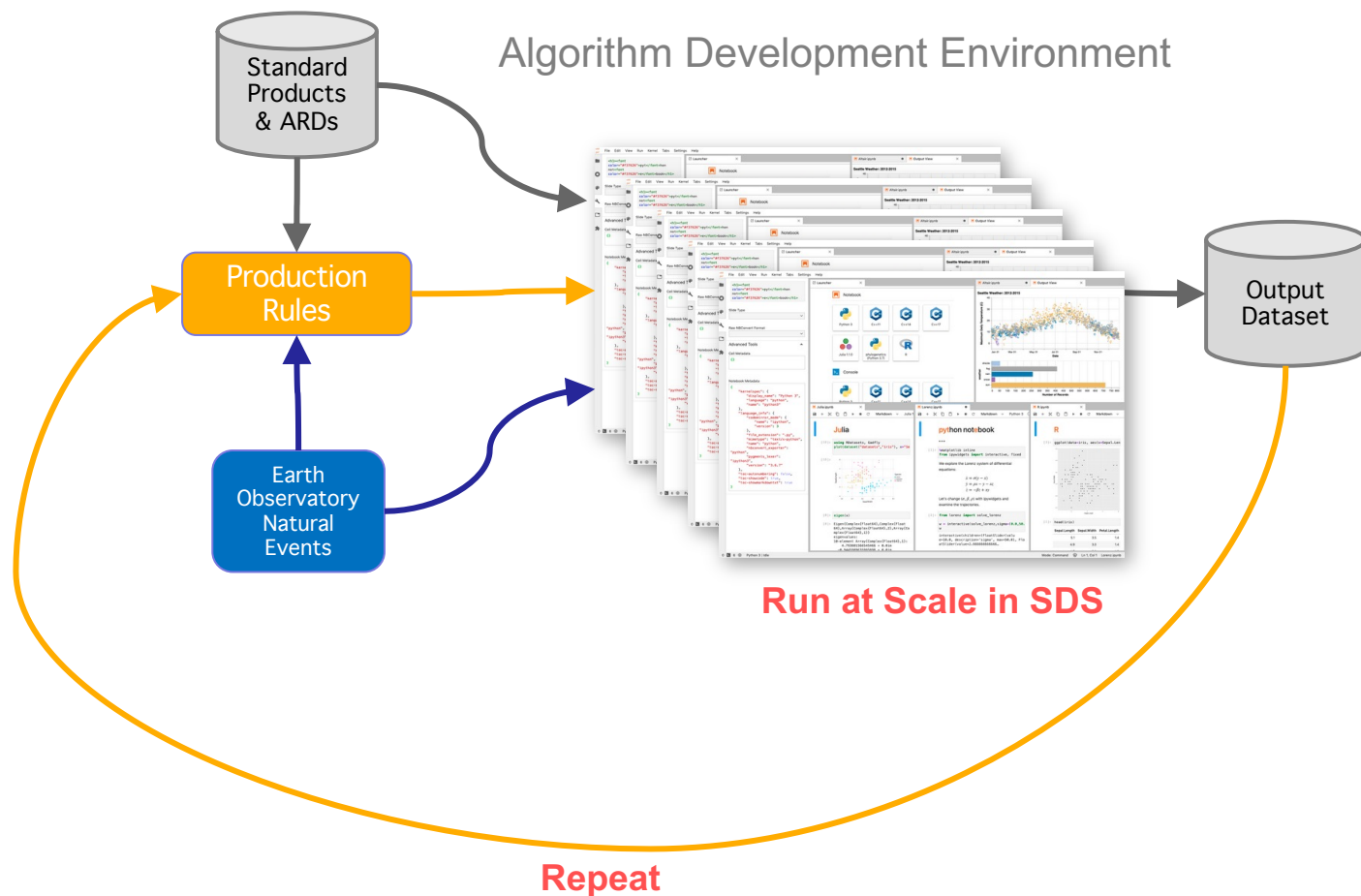
Key Concepts



- Algorithm development environment (Jupyter notebooks)
- Collocated in cloud with science data processing
- Algorithm test bed –at scale
- “ARD-like” SAR data for easier analysis
- Events catalog to natural events
- Production Rules Triggers to link events to automated analysis via user’s notebooks

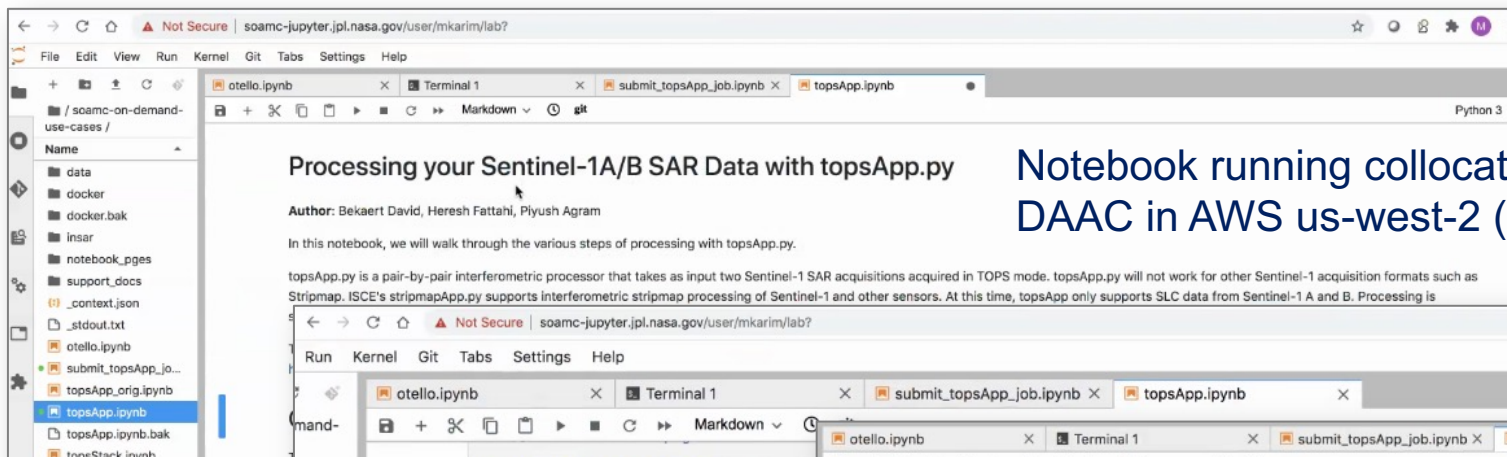


Concept of Jupyter Notebooks Orchestration at Scale

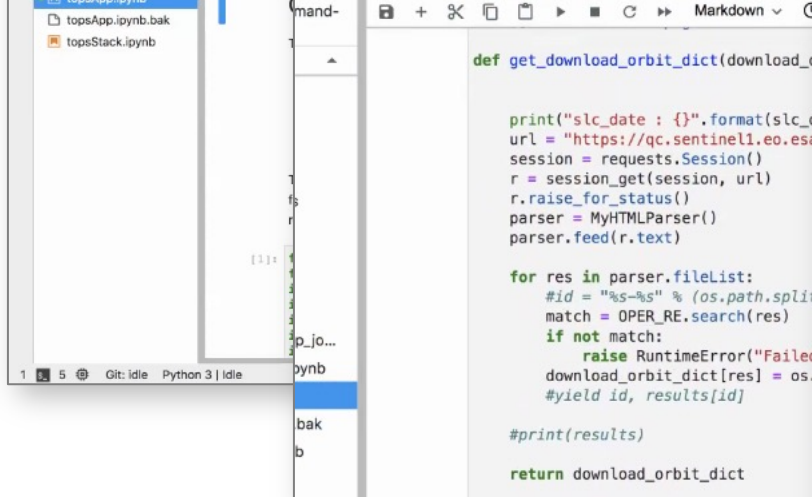




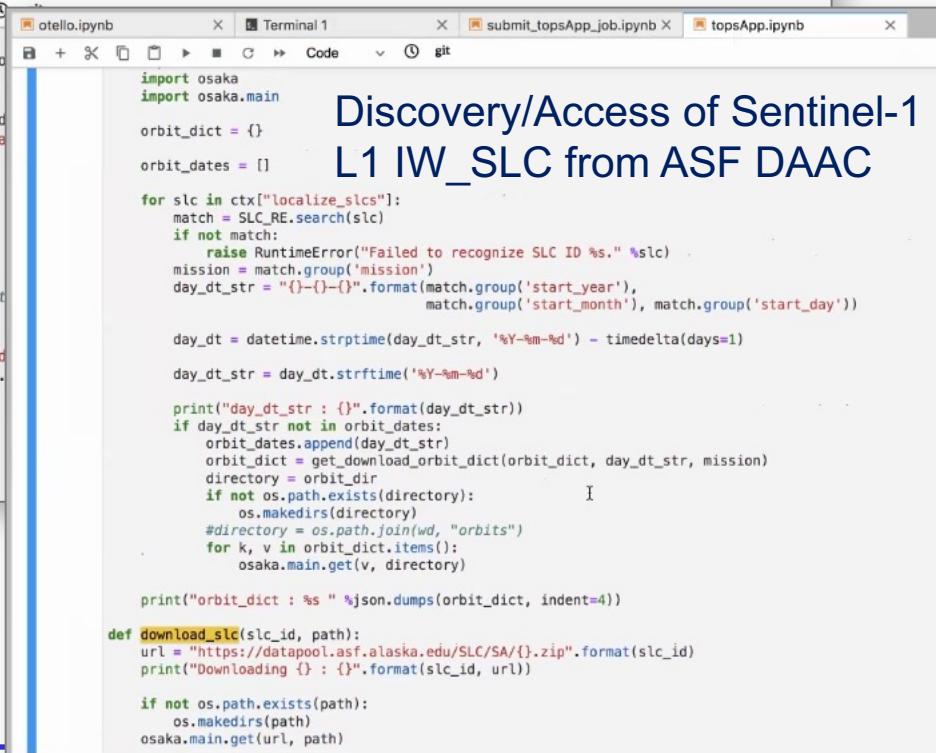
SAR Algorithms in Jupyter Notebooks Collocated with DAAC in AWS



Notebook running collocated with ASF DAAC in AWS us-west-2 (Oregon region)



Discovery/Access of Sentinel-1 ancillary orbits from ESA



Discovery/Access of Sentinel-1 L1 IW_SLC from ASF DAAC



SAR Algorithms in Jupyter Notebooks Collocated with DAAC in AWS

jupyterhub topsApp Last Checkpoint: 19 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
bold.ttf) normal normal 700 condensed) = 0.5349999999999999
2021-06-29 23:01:20,722 - matplotlib.font_manager - DEBUG - findfont: score(<Font 'DejaVu Sans' (DejaVuSans-Oblique.t
tf) oblique normal 400 normal) = 1.05
2021-06-29 23:01:20,723 - matplotlib.font_manager - DEBUG - findfont: score(<Font 'DejaVu Sans' (DejaVuSansCondensed-
Oblique.ttf) oblique normal 400 condensed) = 1.25
2021-06-29 23:01:20,723 - matplotlib.font_manager - DEBUG - findfont: score(<Font 'DejaVu Sans' (DejaVuSans-Bold.ttf)
normal normal 700 normal) = 0.33499999999999996
2021-06-29 23:01:20,724 - matplotlib.font_manager - DEBUG - findfont: Matching sans-serif:style=normal:variant=norma
l:weight=normal:stretch=normal:raise=19.0 to DejaVu Sans ('/opt/conda/lib/python3.8/site-packages/matplotlib/mpl-data/
fonts/ttf/DejaVuSans.ttf') with score of 0.9390009.
```

3.3 Generate the Product

This step generate the topsApp Products. This involves:

- Creating a product id,
- Creating product dir (with the same name as id),
- Moving all the files related to product to product dir
- Generating Met json (Metadata file)
- Generating Datasets.json (Dataset file) More information about metadata files can be found at : <https://hsvds.copernicus.eu/wiki/Scenes+TTS+scenes/265049377/Product+Metadata+Files>

Sentinel-1A/B GUNW processing

Not Secure | soamc-jupyter.jpl.nasa.gov/user/mkarim/lab?

Kernel Git Tabs Settings Help

otello.ipynb Terminal 1 submit_topsApp_job.ipynb topsApp.ipynb

PLOT INTERFEROGRAM

```
[22]: os.chdir(insar_dir)
plotdata('merged/filt_topophase.unw.geo', band=2,
         title='UNW GEO FILT IFG [rad]', colormap='jet',
         colorbar_orientation='vertical', datamin=-20, datamax=50)
#plotdata('merged/filt_topophase_2stage.unw.geo', 2, title='UNW 2-stage GEO FILT IFG [rad]', colormap='jet', colorbar_orientation='vertical')
```



Automating Science Notebooks into Executable Containers

- Enable running same Jupyter notebooks at scale in SDS
 - Enables running large analysis with notebooks across collection of data
- Automated generation of Jupyter notebooks as executable containers
 - Building annotated science notebooks to execute with open source tool `papermill`, then Containerize, and deploy to SDS—to run at scale

```
from typing import List

#Input Parameters: location
min_lon = -155.65979003906253 # type: number
max_lon = -155.34118652343753 # type: number
min_lat = 19.596019240312508 # type: number
max_lat = 19.833892782537767 # type: number

reference_date = "2020-12-20"
secondary_date = "2020-12-08"

#Optional Parameters
satellite = 'Sentinel-1B'
satellite_mode = 'D'
track = 87
version = "v2.0.4"

#Tunable Parameters
sensor_name = "SENTINEL1"
swaths: List[int] = [1, 2, 3]
range_looks = 7
azimuth_looks = 3
do_unwrap = "False"
unwrapper_name = "snaphu_mcf"
do_denseoffsets = "False"

...

The parameters below related to hysds integration and user generally dont need to update them.
HySDS represents Hybrid Science Data System (HySDS) where we can process a noteeok PGE as job and run in scale.
More information about these parameters can be found in :https://hysds-core.atlassian.net/wiki/spaces/HYS/pages/199
About HySDS: https://hysds-core.atlassian.net/wiki/spaces/HYS/
```

```
In [ ]: 1 msgs = ["Hello, world!"]

In [ ]: 1 for msg in msgs:
        2     print(msg)
```



ARD-like Coregistered SLC Stack Generation

soamc-jupyter.jpl.nasa.gov/user/mkarim/notebooks/soamc-on-demand-use-cases/topsStack.ipynb

jupyterhub topsStack (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

Sentinel-1 TOPS stack processor

The detailed algorithm for stack processing of TOPS data can be find here:

- Fattahi, H., P. Agram, and M. Simons (2016), A Network-Based Enhanced Spectral Diversity Approach for TOPS Time-Series Analysis, IEEE Transactions on Geoscience and Remote Sensing, 55(2), 777-786, doi:10.1109/TGRS.2016.2614925.

The scripts provides support for Sentinel-1 TOPS stack processing. Currently supported workflows include a coregistered stack of SLC, interferograms, offsets, and coherence.

Be sure [default] credentials in ~/.aws/credentials are valid and the ~/.netrc file has been changed to include valid credentials for urs.earthdata.nasa.gov.

```
In [ ]:
import os
import json
from math import floor, ceil
import json
import re
import osaka
import osaka.main
from builtins import str
import os, sys, re, json, logging, traceback, requests, argparse
from datetime import datetime
from pprint import pprint
from requests.packages.urllib3.exceptions import (InsecureRequestWarning,
                                                  InsecurePlatformWarning)
try: from html.parser import HTMLParser
except: from html.parser import HTMLParser

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
requests.packages.urllib3.disable_warnings(InsecurePlatformWarning)
PROCESSING_START=datetime.now().strftime("%Y-%m-%dT%H:%M:%S")
print(PROCESSING_START)

PGE_BASE=os.getcwd()
print(PGE_BASE)

In [ ]:
SLC_RE = re.compile(r'(?<mission>S1\w)_IW_SLC_...?'+
r'(?<start_year>\d{4})(?<start_month>\d{2})(?<start_day>\d{2})'+
r'(?<start_hour>\d{2})(?<start_min>\d{2})(?<start_sec>\d{2})'+
r'(?<end_year>\d{4})(?<end_month>\d{2})(?<end_day>\d{2})'+
```

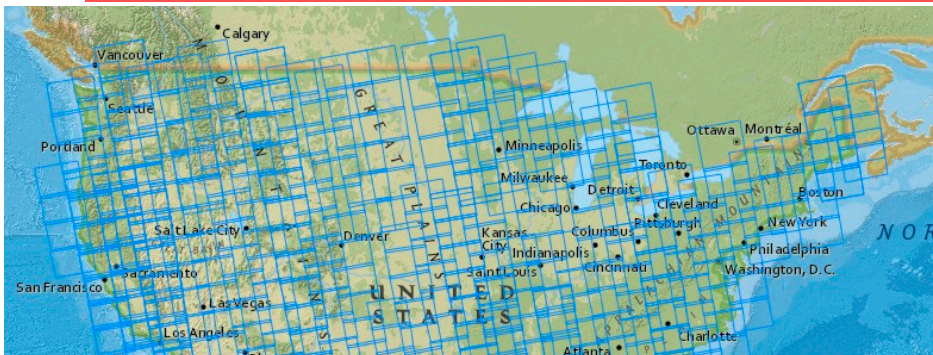
- Coregistration of of SLCs into geocoded stacks
- ARD-like stack as basis of other SAR analysis
 - Damage proxy maps
 - Flood proxy map
 - High resolution displacement time series
- Ported to run in Jupyter notebook and deployable into SDS
- Updates to align with latest ISCE2 open source development
- Benchmarked and optimized performance runs with multi-core parallelization

	c5d.9xlarge (36 vCPU, 72 GiB)	c5.24xlarge (96 vCPU, 192 GiB)	x1e.2xlarge (8 vCPU, 244 GiB)
1 year (~30 SLCs, 4 bursts)	7 hrs, 24 mins, 46 secs	4 hrs, 38 mins, 33 secs	
2 years (~60 SLCs, 4 bursts)	13 hrs, 37 mins, 39 secs	8 hrs, 16 min, 46 secs	
1.7 years (54 SLCs) Beirut	2.63 hrs (50% HT)		
2.7 years (84 SLCs) Beirut	4.09 hrs (50% HT)		
0.7 years (26 SLCs) Beirut			2.76 hrs (50% HT)





Example Potential of SAR Analysis Notebooks at Scale



(upper-left) Sentinel-1A/B ascending track over U.S. : ~650 parallel stack processor jobs running at scale

- Approach for ARD-like Sentinel-1 SLC stack generation—at scale
 - Each SLC footprint stack processing is deployed to run at scale in SDS via Containerized Jupyter notebooks
- Parallelization speed up
 - **Coarse grain parallelization:** scale up parallel SLC stack notebooks to run in parallel in SDS in AWS
 - **Fine graine parallelization:** each notebook leverages multi-core processing
- Addressing costs
 - Leverage **lower costs AWS spot market instances** for deploying Jupyter notebooks at scale
 - Dispatch same jobs to NASA Pleiades for “free” compute (via SBU allocations)
 - * *Operational costs of these kinds of large processing jobs are outside the scope of this AIST technology demonstration*

1076 x 1-year (~30 SLCs) coregistered SLC stacks:

10-months to process in parallel 36-core machine

vs

8 hours in this on-demand ACF

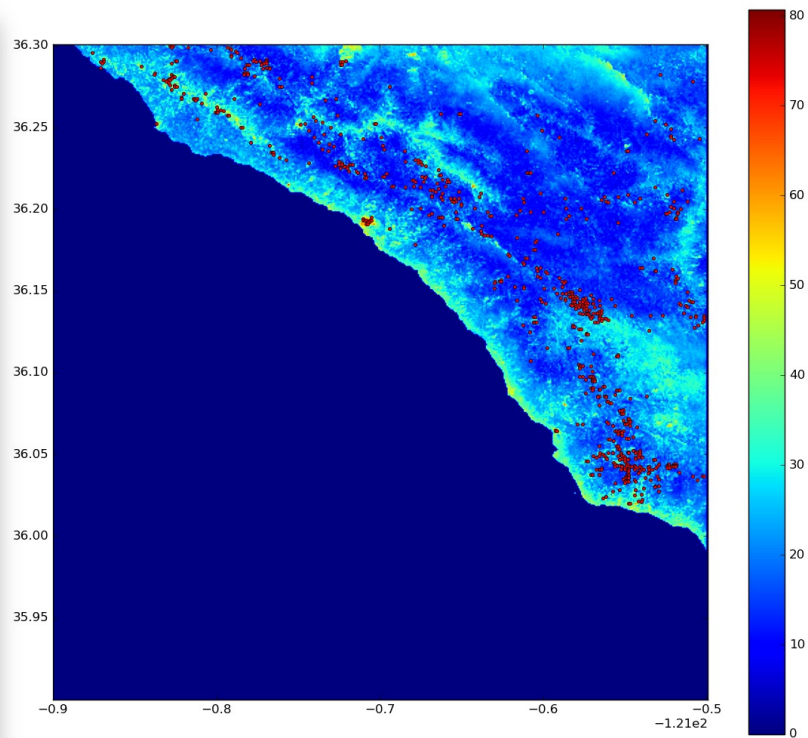
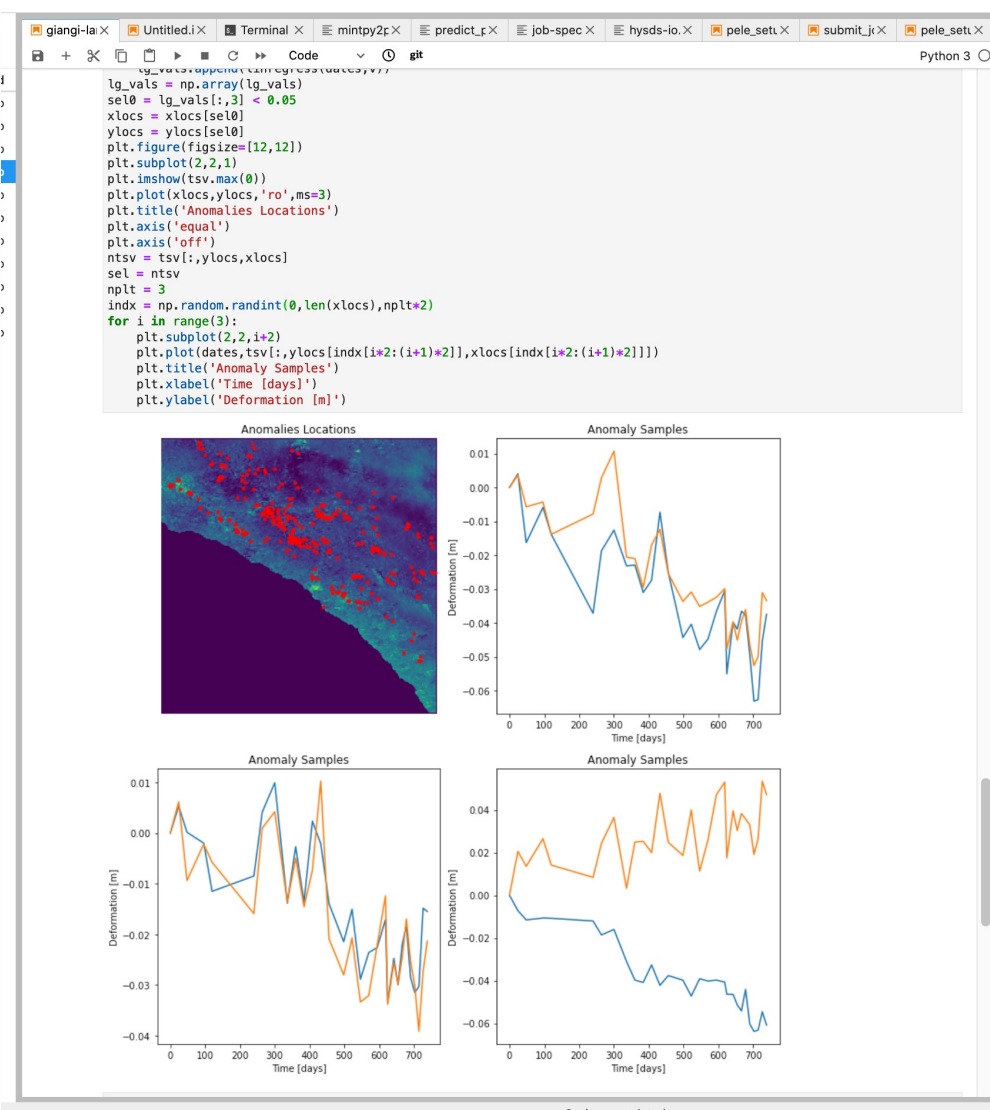
→ Enables more rapid algorithm development



(lower-left) Sentinel-1A/B descending track over U.S. : ~426 parallel stack processor jobs running at scale



Notebooks at Scale: ML-based Deformation Anomaly Detection for Potential Landslides



(above) The heatmap of the max displacement. Time series spread shows the locations of prediction as red dots

(left) Jupyter notebook for neural network-based machine learning for detecting temporal signals in MintPy-generated displacement time series products. Notebook can be deployed as Docker container running at scale over large geographical areas.



Integration of NASA EONET Events to Automate Triggering of Deformation Processing

- Goal: to provide natural events as “triggers” for automating data processing with “notebook algorithms”
- NASA Earth Observatory Natural Event Tracker (EONET)
 - Providing a curated source of continuously updated natural event metadata.
- Curated Events
 - **Severe Storms:** Tropical Cyclones
 - National Hurricane Center
 - Joint Typhoon Warning Center
 - **Volcanoes**
 - Smithsonian/USGS Weekly Volcanic Activity Report
 - **Wildfires**
 - Alberta Wildfire
 - British Columbia Wildfire Service
 - California Department of Forestry and Fire Protection
 - InciWeb
 - Manitoba Wildfire Program
 - Pacific Disaster Center
 - **Sea and Lake Ice:** icebergs
 - National Ice Center

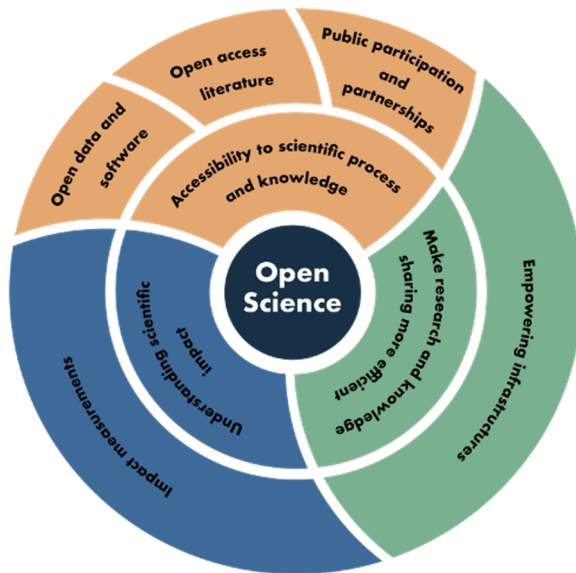
HyGDS Faceted Search interface showing filters for event type and dataset. The 'event' filter is set to 'event (1605)' and the 'dataset' filter is set to 'eonet-severe-storms (1026)'. A map of Hawaii is visible in the background.

Continuous ingest of EONET events into analysis environment

EONET Earth Observatory Natural Event Tracker. The page features a header with the EONET logo and a main content area with introductory text about the repository and API.



Open Science Implications for ACF



"From Open Data to Open Science." Earth and Space Science [doi:10.1029/2020EA001562]
<https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2020EA001562>

<https://nasadaacs.eos.nasa.gov/esds/open-science>

- Open Data
 - NASA has free and open data distributed by DAACs
- Open Access
 - DAACs provide access to open data
- Open Source
 - Algorithms, software, code, production system artifacts are open source
 - Development done in the open
- Open Cyberinfrastructure
 - Analysis platforms are open, accessible, interoperable, contributable
- Collaboration
 - Scientific research is sharable in collaborative environments
 - Development of algorithms (and system) are done in open and collaborative manner
- Provenance and reproducibility
 - Are the data production and analysis steps preserved in a way that can be reproduced?
 - By the same system and/or by others?
- Open Knowledge Dissemination
 - Sharing and publishing in open-access journals
- Impacts
 - Measuring science impact of earth science data records (ESDRs)

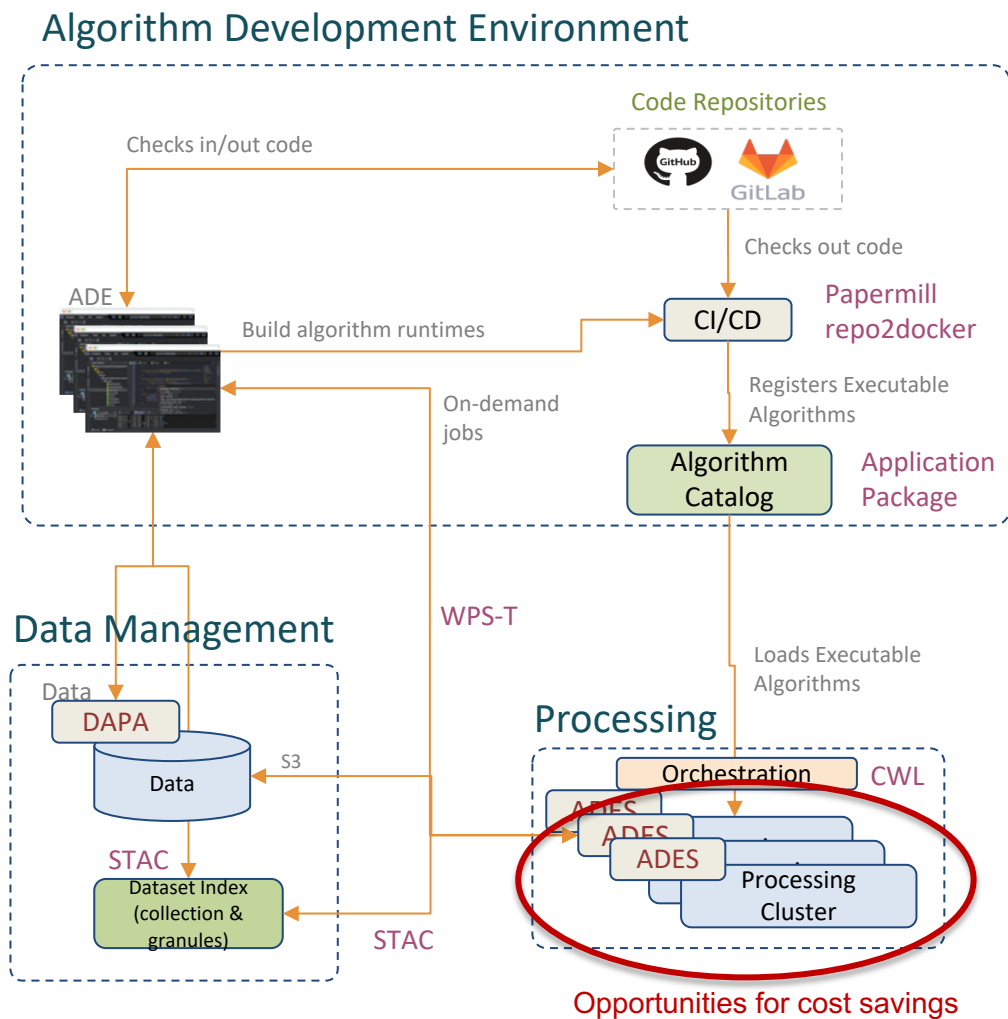


Open Geospatial Consortium (OGC) Relevance to Open Science and ACF

- Adhering to these OGC specs helps to open up the ACF approach towards **Open Science**
- Instills open and **collaborative** algorithm development, **distributed** teams, and **interoperability** across systems
- OGC is an **international standards body**.
 - *They do not define implementations.*
- OGC specifications instills **interoperable architectural design patterns**
- Open and international standards **fosters community** implementations and interoperability via interoperable service interfaces
- Interoperability as basis for **provenance and reproducibility**



Factoring in OGC Interoperability for Open Science



- OGC Earth Observation Applications Pilot
- OGC Testbed 13 to 17 have relevant specifications
- Application Deployment and Execution Service (ADES)
- Web Processing Service Transactional (WPS-T)
- Application Package (portal containerized jobs)
- SpatioTemporal Asset Catalog (STAC)
- Data Access and Processing API (DAPA)
- Command Workflow Language (CWL)



Addressing SAR Analysis Cloud Costs

- Large compute needs and costs of SDSes in both NISAR and SWOT
- Address vendor lock-in issues
- Early cost analysis shows potential for savings across multi-cloud

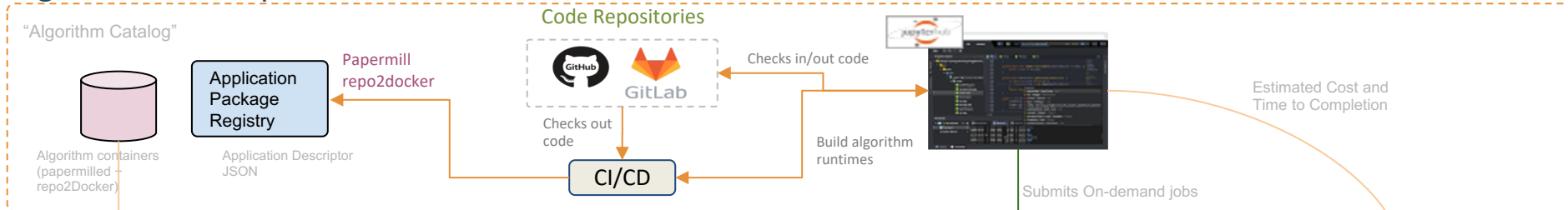
Analysis Example Need	Amazon Web Services (AWS)	Google Cloud Platform (GCP)	Microsoft Azure Cloud
Light analysis on 10 small compute instances	Instance: t3.small (2 Cores, 2 GiB RAM) Region: US West (Oregon) \$0.21 per hour \$149.76 per month [LOWER COSTS]	Instance: N1-STANDARD-2 (2 Cores, 7.5 GiB RAM) Region: Western US \$0.67 per hour \$478.80 per month	Instance: B2S (2 Cores, 4 GiB RAM) Region: US West 2 \$0.69 per hour \$493.20 per month
Moderate analysis on 100 medium compute instances	Instance: t3.xlarge (4 Cores, 16 GiB RAM) Region: US West (Oregon) \$16.64 per hour \$11,980.80 per month	Instance: N1-HIGHMEM-4 (4 Cores, 26 GiB RAM) Region: Western US \$16.58 per hour \$11,934.72 per month	Instance: B4MS 4 Cores, 16 GiB RAM Region: US West 2 \$8.93 per hour \$6,429.60 per month [LOWER COSTS]
Large SAR bulk processing on 1000 large compute instances	Instance: c5.9xlarge (36 Cores, 72 GiB RAM) region: US West (Oregon) \$1,530.00 per hour \$1,101,600.00 per month	Instance: N1-STANDARD-32 (32 Cores, 120 GiB RAM) Region: Western US \$1,064.00 per hour \$766,080.00 per month [LOWER COSTS]	Instance: F32 v2 (32 Cores, 64 GiB RAM) region: US West 2 \$1,361.35 per hour \$980,172.00 per month

* Cost estimates are examples based on publicly advertised "rack rate" costs for spot/preemptive compute. Actuals will vary depending on market demand and cloud reseller rates.

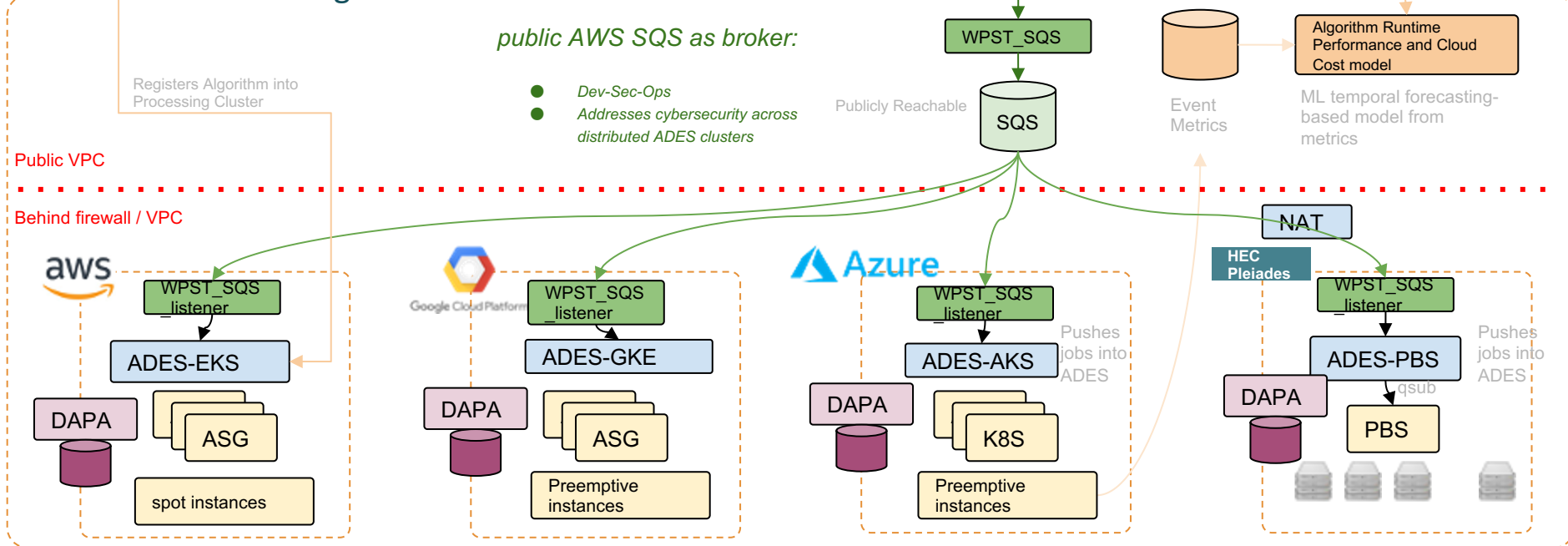


Running Jupyter Notebooks at Scale across Multi-Cloud and HEC

Algorithm Development

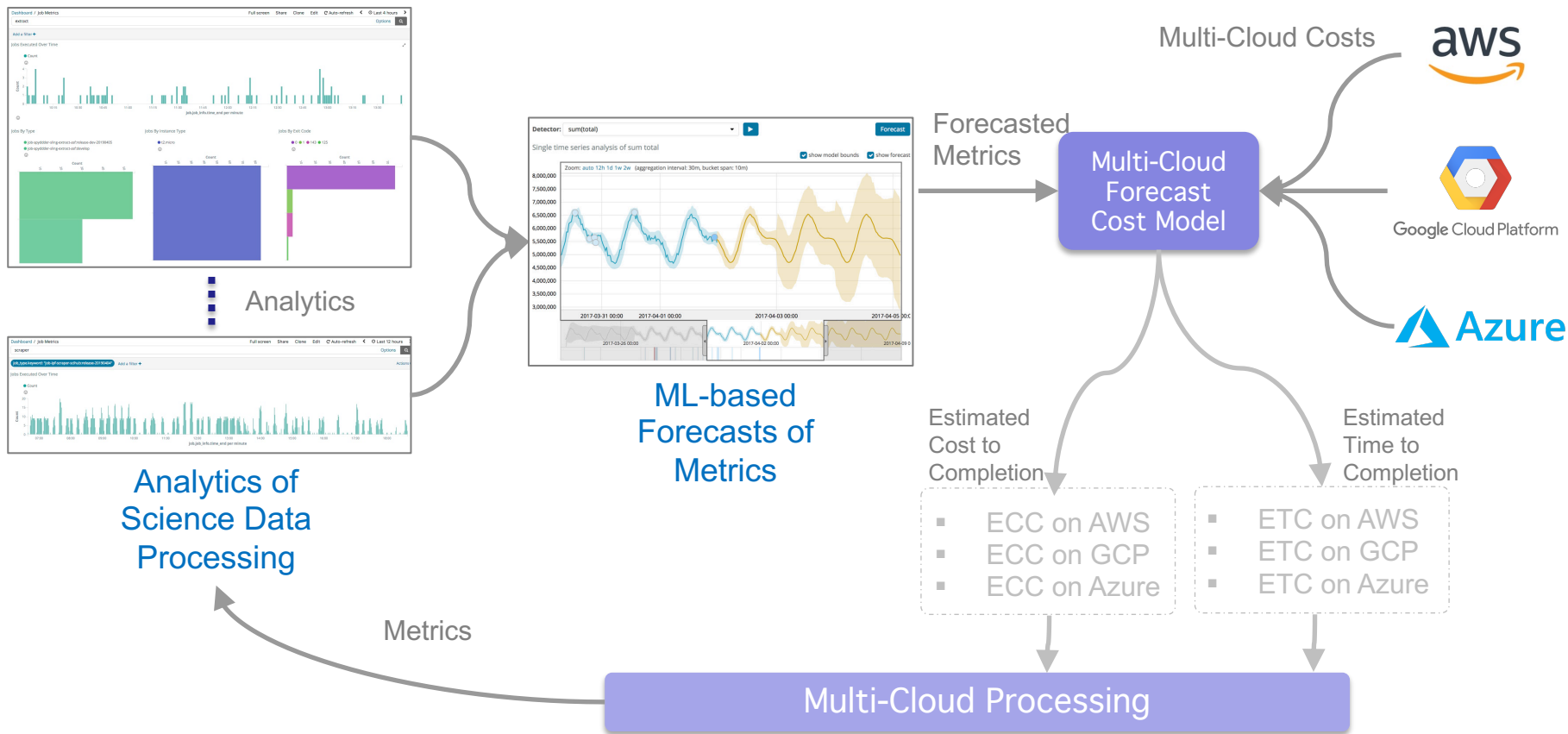


Science Data Processing





Collecting Metrics for ML-based Forecasting and Estimates

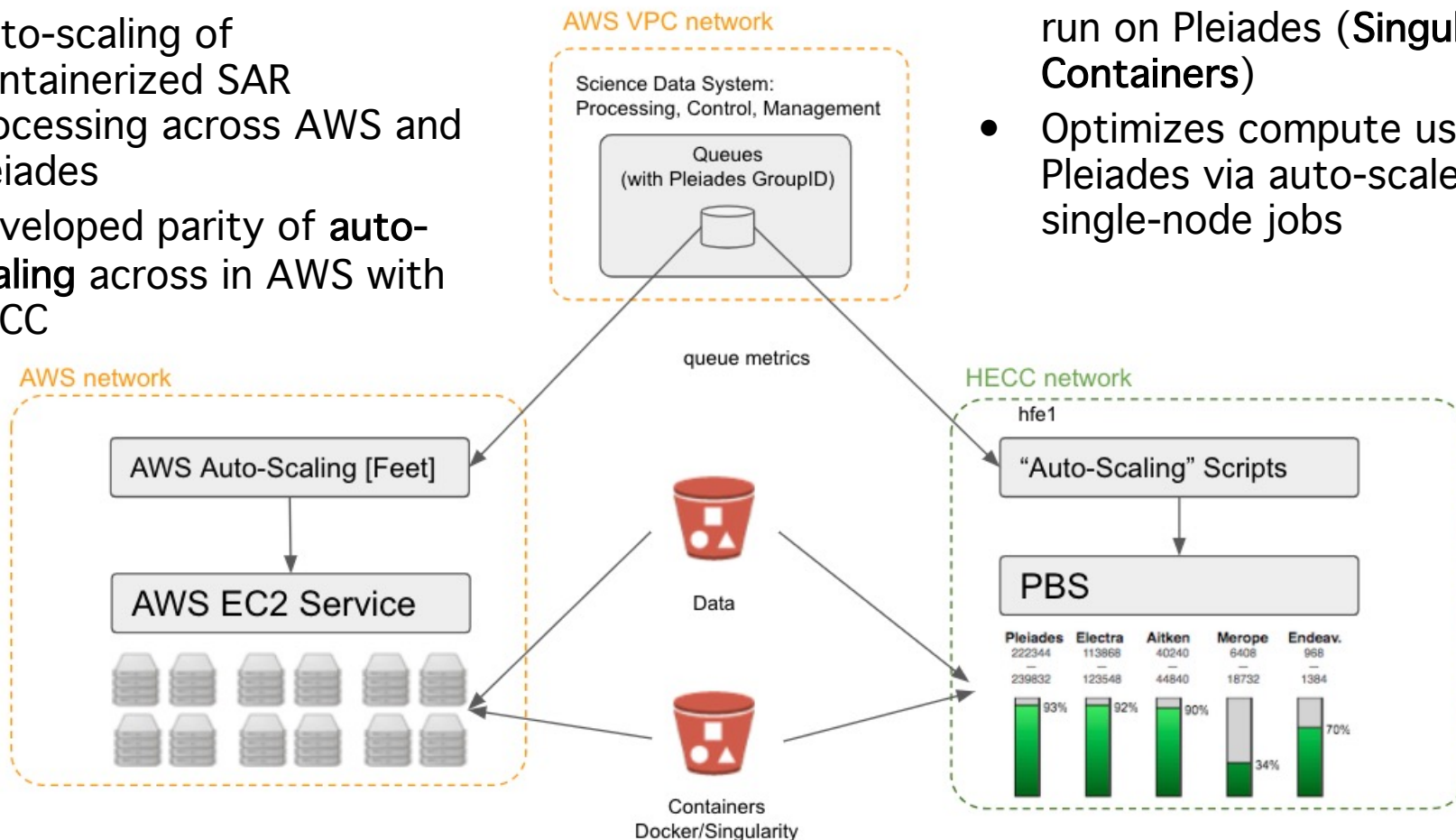




Prototyped Auto-scaling Compute Across AWS and NASA HEC Pleiades

- Initial effort started under ESI funding for ARIA in HEC (PI: Sue Owen)
- Auto-scaling of Containerized SAR processing across AWS and Pleiades
- Developed parity of **auto-scaling** across in AWS with HECC

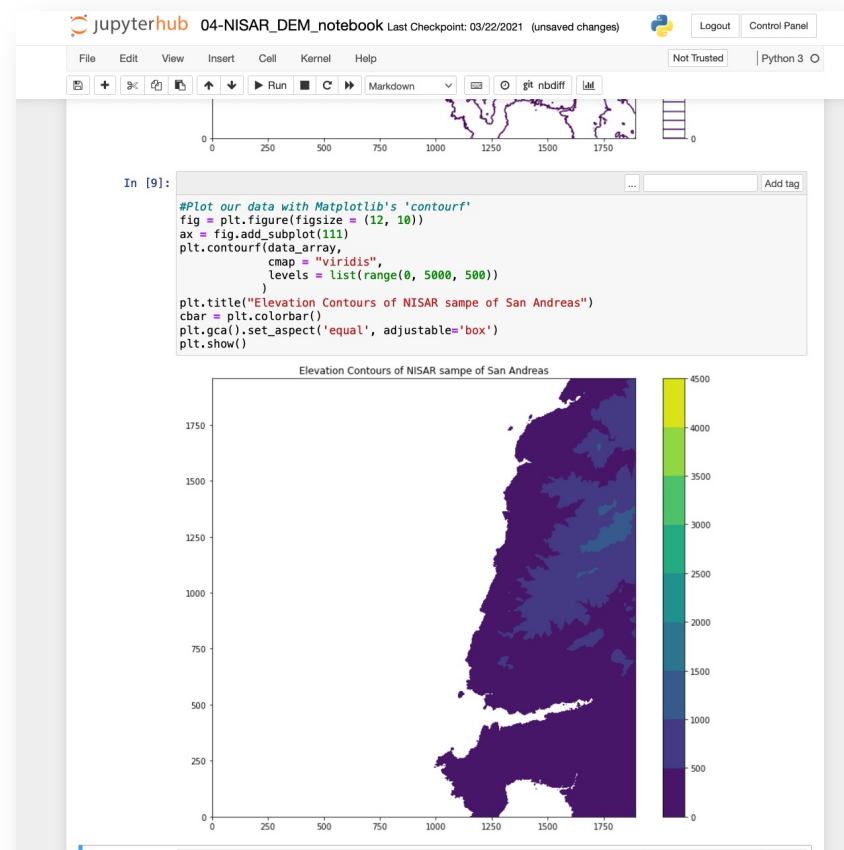
- Algorithms deployed to run at scale in AWS can also run on Pleiades (**Singularity Containers**)
- Optimizes compute use on Pleiades via auto-scaled single-node jobs





NISAR SDS Infusion for On-Demand Processing

- AIST task contributions to **open source** software used by NISAR SDS and SWOT SDS
- **Coordination** with NISAR SDS contributing to the on-demand algorithm development and test bed environment
 - Use cases for Cal/Val and ADT
 - Science team already started exploring science notebooks for algorithms
 - **Algorithm improvement and data product validation**
- NISAR SDS **deployed** on-demand system for NISAR Science Team
 - **Shares common based system developed in this AIST**
 - Algorithms in Jupyter notebooks are deployable as Docker containers running at scale in SDS
 - NISAR SDS current baseline in AWS only





Key Points

- Motivation
 - Address need for rapid & **scalable algorithm development environment** supporting use cases of algorithm development, data product generation, production, and product validation
 - Assess more more **cost-efficient** compute approaches for larger L2 and L3 analysis
- Approach
 - Integrated **algorithm development environment (ADE)** and SDS for running algorithms in Jupyter notebooks to run **on-demand and at scale in SDS**
 - Demonstration SAR algorithms in **Jupyter notebooks** for Sentinel-1 data as proxy for NISAR:
 - **Sentinel-1 coregistered SLC stacks**
 - **Sentinel-1 GUNW**
 - **ML temporal anomaly detection (potential landslides)**
 - On-demand processing across **multi-cloud (AWS, GCP, Azure)** and **NASA HEC Pleiades**
 - Collect processing **metrics to forecast estimate costs/time to completion**



Acronyms

List of Acronyms

- ADE Algorithm Development Environment
- ADES Application Deployment and Execution Service
- ADT Algorithm Development Team
- ARD Analysis Ready Data
- AODS Analysis Optimized Data Services
- AWS Amazon Web Services
- CWL Command Workflow Language (CWL)
- DAPA Data Access and Processing API
- DPM Damage Proxy Map
- EONET Earth Observatory Network Event Tracker
- FPM Flood Proxy Map
- GCP Google Cloud Services
- HEC High End Computing
- HPC High Performance Computing
- HySDS Hybrid Cloud Science Data System
- InSAR Interferometric Synthetic Aperture Radar
- OGC Open Geospatial Consortium
- PGE Product Generation Executive
- PS time series Persistent Scatter time series
- SAR Synthetic Aperture Radar
- SDS Science Data System
- SLC Single Looks Complex
- STAC SpatioTemporal Asset Catalog
- WPS-T Web Processing Service Transactional (WPS-T)