Jet Propulsion Laboratory
California Institute of Technology

podaac
Physical Oceanography Distributed Active Archive Center

Recent Advances in Cloud Computing and Cloud Data Optimization

Edward M. Armstrong[1], Victoria McDonald[1], Nikki Tebaldi[1], Jorge Vazquez[1], Dean Henze[2],

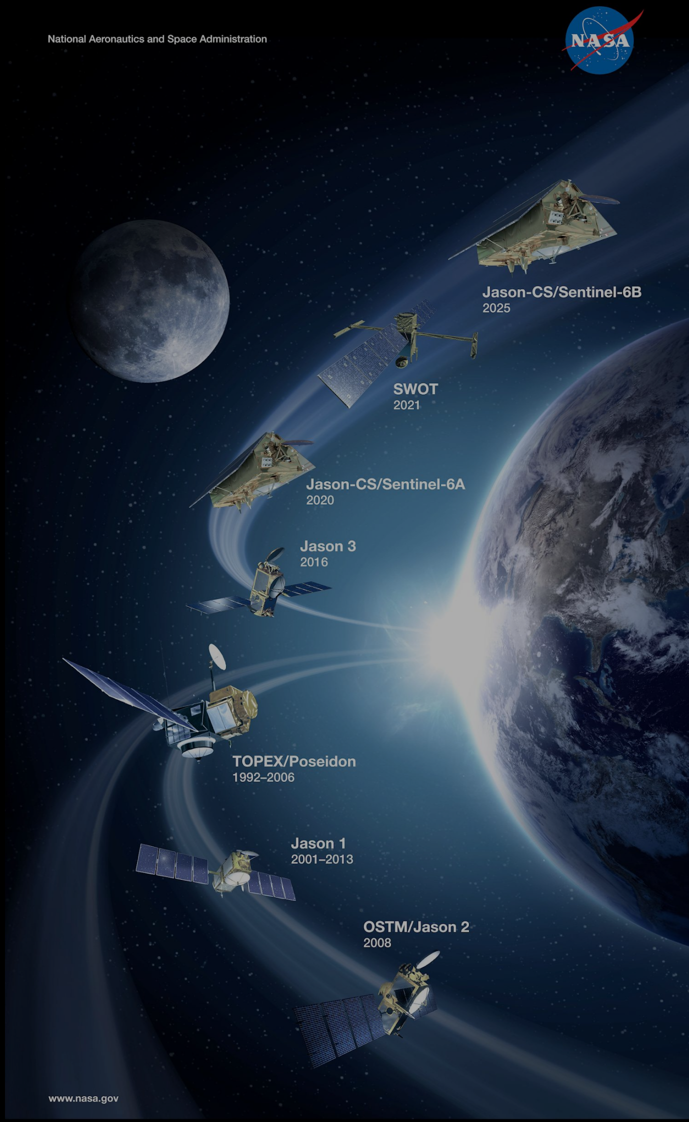
[1] *Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA USA*

edward.m.armstrong@jpl.nasa.gov

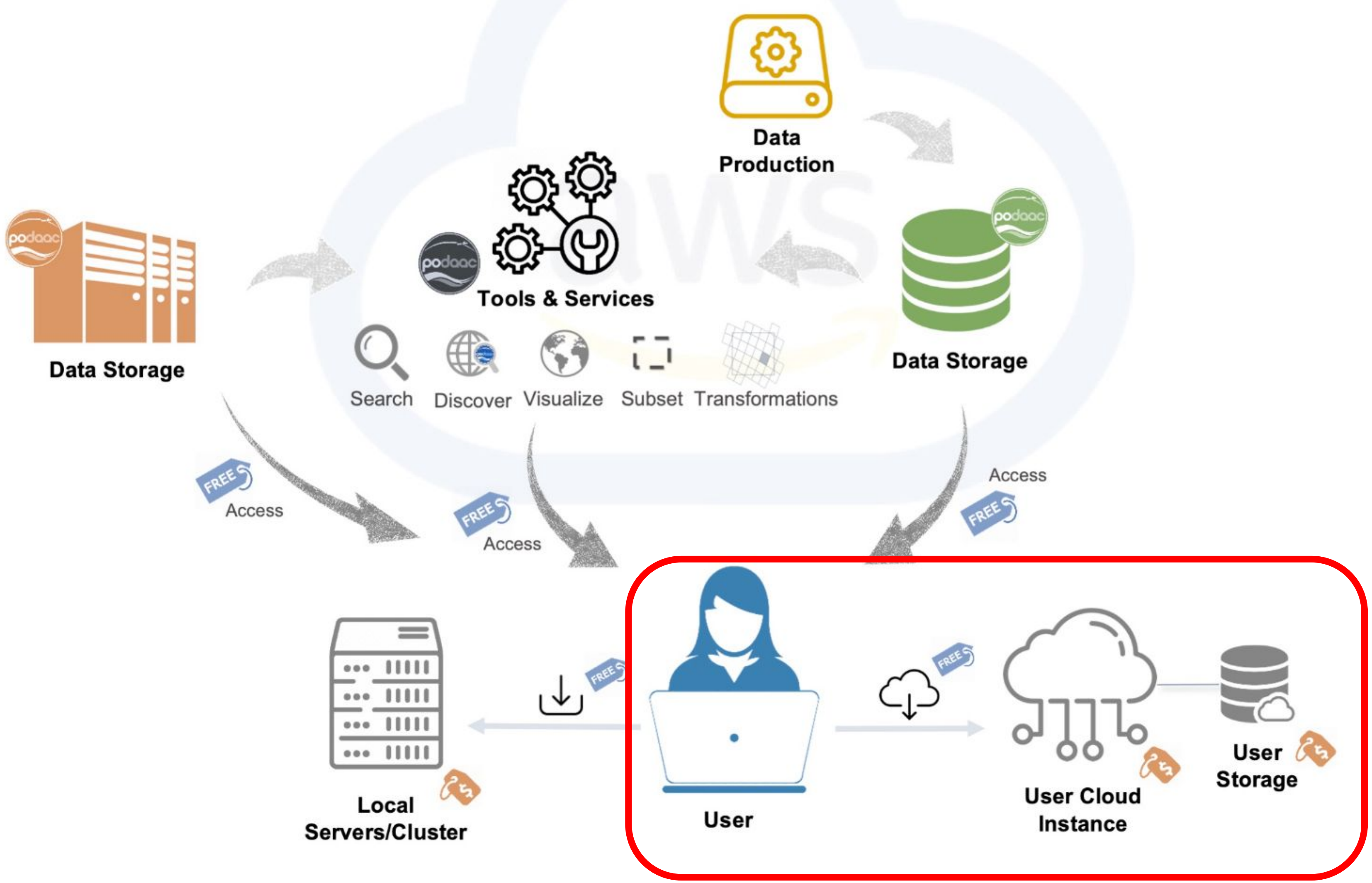
[2] *APR Corporation, Pasadena, CA*

https://podaac.jpl.nasa.gov/

10 June 2024
GHRSST 25th User Symposium and Science Team Meeting



NASA Earthdata Transition to the Cloud



Today: scientific analysis in the cloud….

…but you can still download and compute locally for sure

Progress since the last meeting

- See GHRSST-24 presentation: *Optimizing Data and Services in the Cloud for User Exploitation*
  - New paradigms for data storage, access and most importantly computing
  - Cloud computing “next to the data” in Amazon Web Services (AWS) family of services
- Update on parallel computing using Dask and Coiled
  - Using ECS, Dask with a management service called Coiled to perform subsetting and scientific analyses like EOF
  - Metrics: performance, speed, reproducibility
- Update on scientific time series computing using AWS lambda
  - Scalable and Serverless compute on the MUR SST time series and others
  - Test case for a one year daily average SST time series, and extension to cross comparison of multi SST datasets
  - Metrics: performance,scalability, efficiency, cost
- HDF5/netCDF4 container optimization
- New data stores (e.g. Kerchunk, Zarr)

1) Scientific Cloud Computing with EC2/Dask/Coiled

What is EC2?

- Virtual machines in AWS. Can install software, run code, create files, etc. on this machine.
- EC2 instances can be created with varying numbers of processors, processor types, RAM, etc depending on needs for a scientific computation.

What is Dask?

- Python package providing options for parallelizing code/computations.
- Can be utilized with EC2 instances of varying sizes (CPU/memory) to accelerate computations.

What is Coiled?

- An independent third party software system for managing a Dask cluster. The commercial arm of Dask
- Software as a Service (Saas) but you have to pay; free trial available

Workspace Challenges - Service and Software Knowledge and Expertise

Scientific Computing with Python and Dask	AWS Infrastructure	Other software/technical knowledge
Effective Python, NumPy, Xarray, Dask syntax Understanding of parallelization methods and which one to apply Code and cluster optimization	Basic setup/usage Spinning up & connecting to EC2 instances Lambda functions Requirements for multi-VM, distributed clusters? Cost Tracking	Docker Linux shell / bash scripting Ensuring VM's have EDL creds, AWS creds, software environments Anything else?

Using Coiled

Scientific Computing with Python and Dask	AWS Infrastructure	Other software/technical knowledge
Effective Python, NumPy, Xarray, Dask syntax Understanding of parallelization methods and which one to apply Code and cluster optimization	Basic setup/usage ~~Spinning up & connecting to EC2 instances~~ ~~Lambda functions~~ ~~Requirements for multi-VM, distributed clusters?~~ ~~Cost Tracking~~	~~Docker~~ ~~Linux shell / bash scripting~~ ~~Ensuring VM's have EDL creds, AWS creds, software environments~~ ~~Anything else?~~

Setup

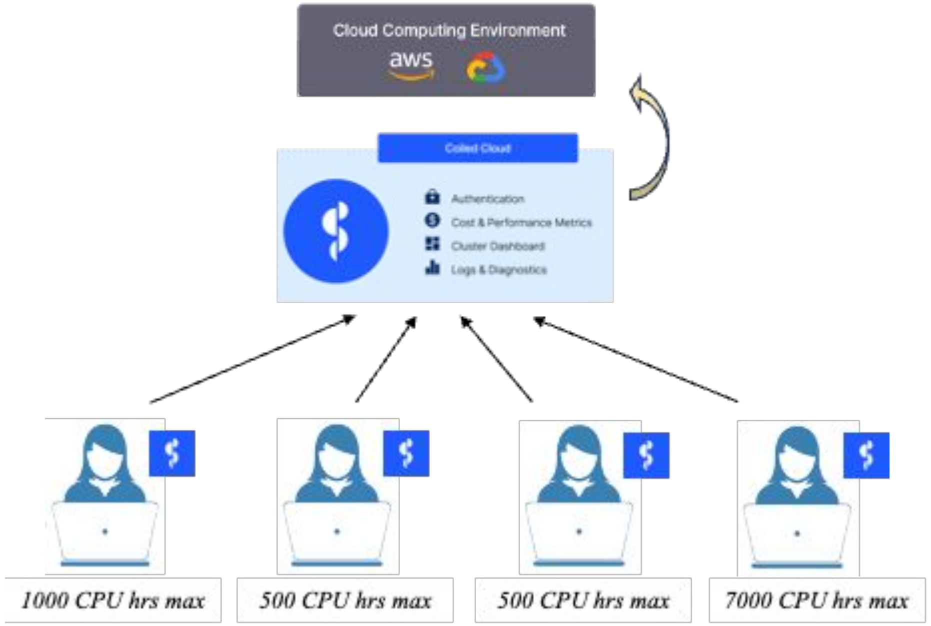
1. Sign up (e.g. with Github / Google account)
2. Download Coiled: “pip install coiled”
3. Connect to an AWS account.

Multiple Coiled accounts can be part of the same workspace, connected to a single AWS account.

Pricing

All AWS costs incurred are your own.

Extra Coiled costs: First 10,000 CPU hours per month free, [$0.05/CPU-hour beyond that]().



Coiled Application #1: SST/SSH correlation

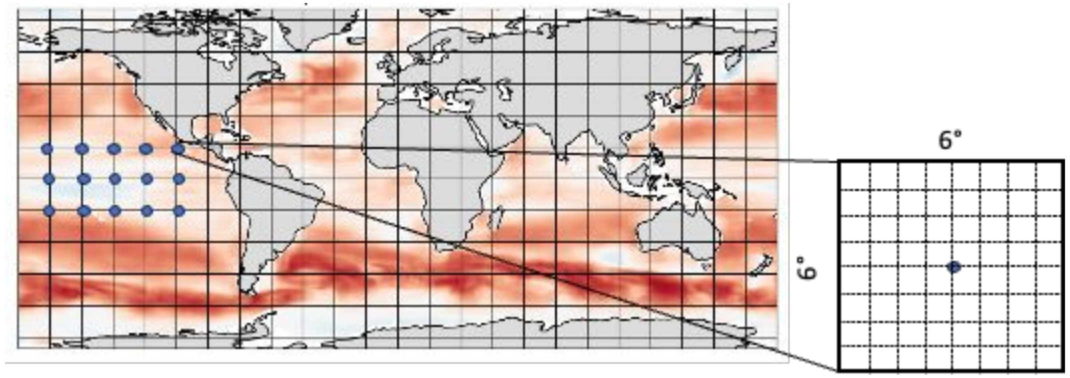
PO.DAAC-hosted, gridded SSH and SST data sets

MEaSUREs gridded SSH Version 2205: 0.17° x 0.17°, 5-day period, https://doi.org/10.5067/SLREF-CDRV3

GHRSST Level 4 MW_OI Global Foundation SST, V5.0: 0.25° x 0.25°, daily, https://doi.org/10.5067/GHMWO-4FR05

Time period of overlap: 1998 – 2020

Analysis: Global map of spatial correlation for each overlapping pair of files, 1808 days in total (3616 files, ~20 GB). Then further statistical assessment of these maps in the future.



Computations for each lat-lon window (after interpolation of SST grid to SSH):

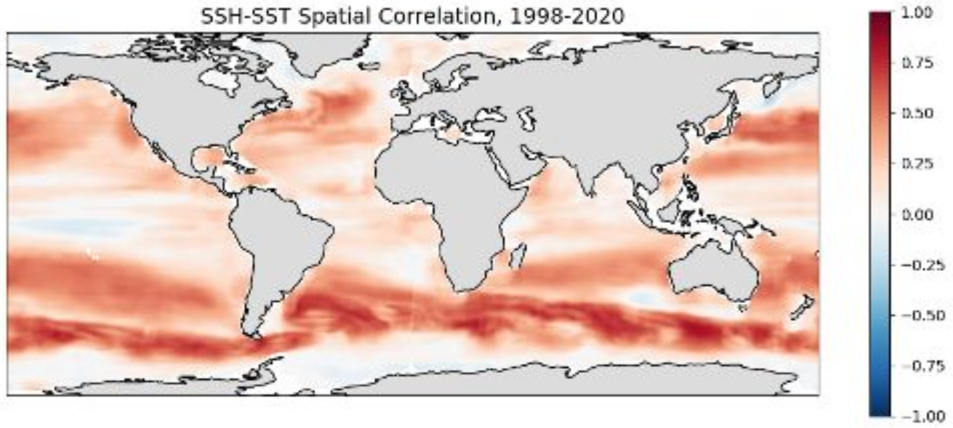
1. Locate SST, SSH data within window
2. Fit 2D linear surfaces to SSH, SST
3. Anomalies = deviations from 2D fit
4. Correlation between anomalies

   ~ 2 million windows per pair of files

#1

Processing the entire record

- Processed entire record (years 1998 – 2022, 1808 pairs of files) at 0.25° x 0.25° resolution, Link to Github.
- At ~25 minutes processing time per pair, it would take ~1 month on a laptop or EC2 instance without parallel computing.
- Ran with a Coiled function, 250 workers, saving output to an S3 bucket. Took ~5 hours for ~$20 (AWS costs, no Coiled costs).



Coiled Summary

- Coiled allows a scientist to focus on the scientific code and results, and abstract the implementation
- Free compute tier will meet most use cases
- Not a silver bullet
  - Still need expertise in python and compute strategies.

2) Experiences with AWS Lambda

Features

- Serverless computing
- Event driven
- Computations scale across large data volumes without provisioning server capacity

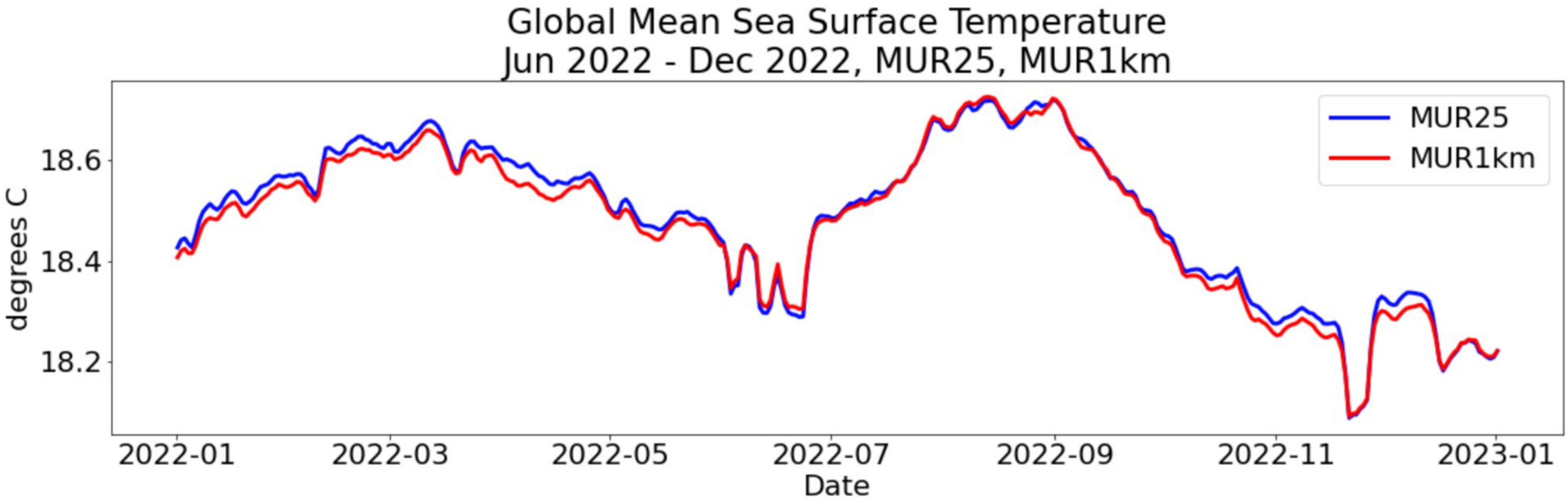
Best for discrete tasks where processes don’t need to communicate with each other

Limitations

- 15 min execution timeout
- 10 GB temporary storage, output needs to be saved somewhere (like S3)
- 1000 concurrent executions (can be increased)
- Memory per Lambda allocated in advance
- Lambdas can’t talk to each other so some use cases need to do calculations in stages

[Limitations documentation]

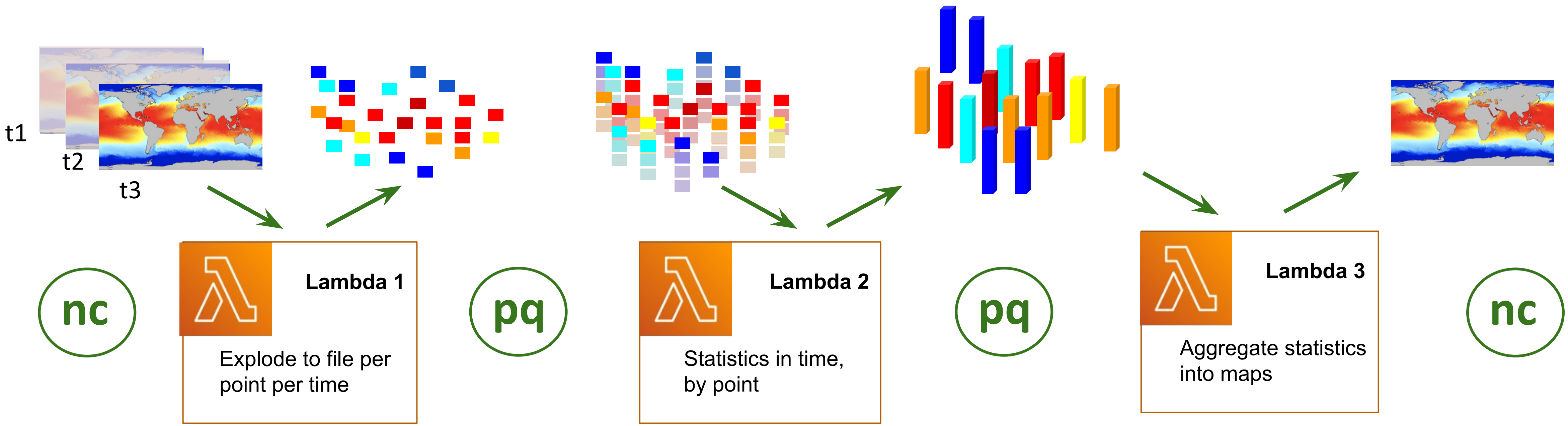
Results



AWS provides a calculator to estimate the cost of running using cloud services. Using the cost estimator, running this tutorial one time as-configured to create a one year time series may result in a charge of **$0.72 USD.**

Scaling up to create a 10-year time series one time would increase the cost to **$7.22 USD**.

Lambda Workflow extension for pixel-by-pixel comparisons



3) Cloud optimization for HDF and netCDF

- HDF5 was built for the pre cloud era when reads to local disks and networked file systems were cheap (fast)
  - In this era the number for read operations was not particularly a high concern
- This paradigm is flipped in the cloud with object store and its key-value interface
  - Reads become longer (expensive).
  - Therefore the read size quantity should optimized as best possible to minimize the number of reads
- The HDF library (and h5py) read internal metadata and raw data in fixed page sizes that are not of optimal size for the cloud in the default mode
- As a further compromise the entire internal metadata contents must be read before any further file I/O is done
  - I.e., the file metadata tells a HDF5 reader where to go in the file (byte offset) to read (or write) the variable (dataset) or chunks of variable

Example of the optimized HDF5 file metadata structure

- After using the *h5repack* command:
  - % h5repack -S PAGE -G 4194304 in_file out_file

```
File space management strategy: H5F_FSPACE_STRATEGY_PAGE
File space page size: 4194304 bytes
Summary of file space information:
  File metadata: 4643132 bytes
  Raw data: 1062927559 bytes
  Amount/Percent of tracked free space: 0 bytes/0.0%
  Unaccounted space: 6171133 bytes
Total space: 1073741824 bytes
```

Performance tests - netCDF4

- Using ‘h5repack’ on netCDF4 is a valid strategy
  - Same input operation to read the total metadata size, repack with an optimized single metadata page size
  - netCDF4 structure and rules maintained; its a netCDF4 output
- Same performance programs implemented as with HDF5 tests
- Caveat: Paged Aggregation cannot be implemented using the native netCDF libraries

Results for **AWS-West S3** endpoints for a LLC netCDF file

File: LLC4320_pre-SWOT_ACC_SMST_20120707.nc [**5 GiB; 1 group; 45 datasets, ~348KiB file metadata, ;** Level 4]

File Page Size	(Read) Buffer Size	Runtime total
Original (default 4KiB)	No Page Strategy Implemented	~ 5 secs
524 KiB Page	524 KiB	~ 0.9 secs
524 KiB Page	1 MiB	~ 0.9 secs

File with **Page** strategy **10% larger** than original.

>5x round trip read improvement

Alternative pathways for netCDF4 optimization

- Use the “experimental” h5netcdf python interface to netCDF to write out an optimized file.
  - **Example 1** (with page aggregation)

```
import h5netcdf
h5netcdf.File('co.nc', mode='w', fs_strategy='page', fs_page_size=4*1024*1024, ...)
```

  - **Example 2** (with metadata block)

```
import h5netcdf
h5netcdf.File('co.nc', mode='w', meta_block_size=4*1024*1024, ...)
```

Example applications of a virtual data store - by Ayush Nag

- Accessing an entire SSH/SST time series for visualization and subsetting
- Leveraging the Kerchunk/Zarr/DMR++ technologies
    - Kerchunk: https://fsspec.github.io/kerchunk/
    - Zarr: https://doi.org/10.5067/DOC/ESCO/ESDS-RFC-048v1
    - DMR++: https://docs.opendap.org/index.php/DMR%2B%2B
- A lightweight JSON externally accessible metadata file is the key
    - https://nbviewer.org/github/podaac/the-coding-club/blob/main/notebooks/SWOT_SSH_dashboard.ipynb
    - Accessible both locally and in the cloud
- See me in the GHRSST GDS Evolution session for more information !

Summary

- Cloud computing: a new ecosystem for scientific computing
  - requires new knowledge, skills, techniques and understanding of its services suite
- Presented ongoing work with cloud computing for time series analysis using Coiled and other AWS services.
  - Tutorials available on our Github site
- For data producers HDF5 (and netCDF4) internal metadata aggregation improves cloud file I/O (sometimes radically)
  - The netCDF4 implementation is a little adhoc but investigations continue
- PO.DAAC is here to help you on your cloud journeys!

Questions to edward.m.armstrong@jpl.nasa.gov