ParslFest 2024

Scalable Earth Observation ML Workflow in Climate Applications

Takuya Kurihana,Tyler J. Skluzacek, Rafael Ferreira da Silva, Valentine Anantharaj
Oak Ridge National Laboratory

OAK RIDGE National Laboratory

U.S. DEPARTMENT OF ENERGY

High quality & large-volume of Earth Observation (**EO**) datasets are distributed across multiple platforms

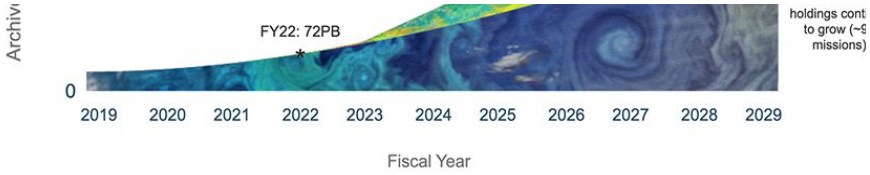
https://www.earthdata.nasa.gov/eosdis/cloud-evolution

ESDS Program

Earthdata Cloud Evolution

More About EOSDIS

With the launch of new, high-data-volume Earth observation missions, NASA's ability to effectively ingest,
and archive large amounts of data requires the most cost-effective, flexible, and scalable data-manageme
architectures and technologies. To meet these demands, NASA's Earth Science Data Systems (ESDS) Progr
implemented a strategic vision to develop and operate multiple components of NASA's Earth Observing S
and Information System (EOSDIS) in a commercial cloud environment.



https://www.earthdata.nasa.gov/esds/esds-highlights/2022-esds-highlights

amazon web services

Google Cloud

Microsoft Azure

11,000+ Data products

ARM

OAK RIDGE National Laboratory

PB

Overview and NCICS/CISESS Contributions

NOAA's Big Data Program (BDP) is designed to facilitate public use of key environmental datasets by providing copies of NOAA's information in the Cloud, allowing users to do analyses of data and extract information without having to transfer and store these massive datasets themselves.

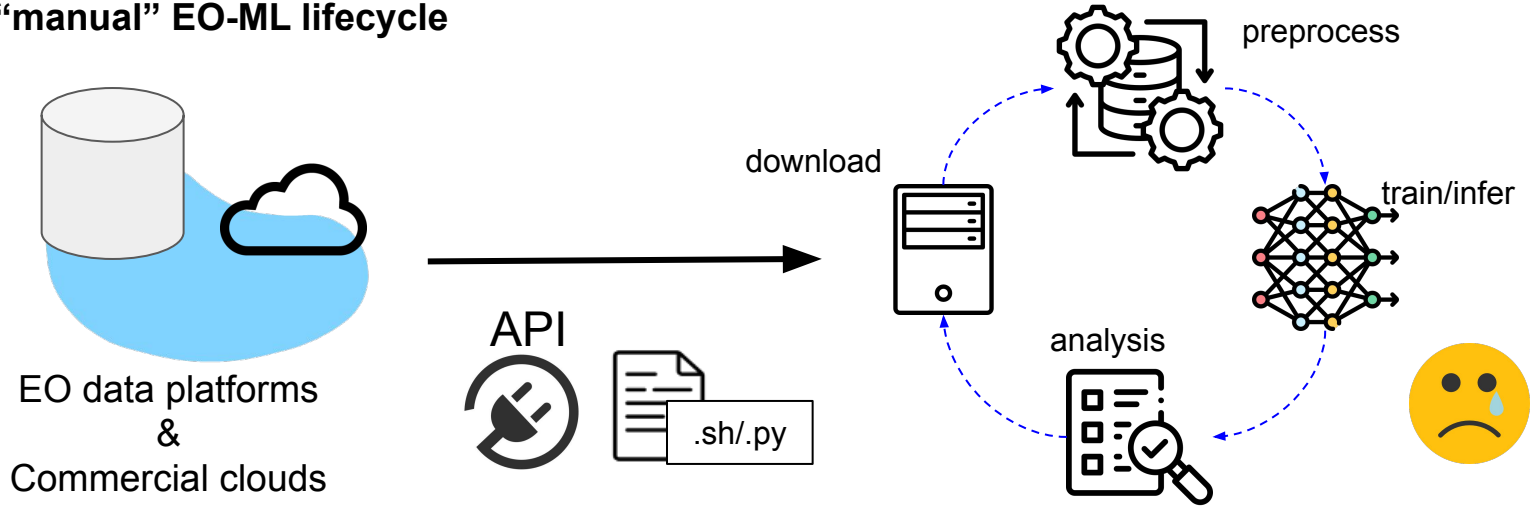
https://ncics.org/data/noaa-big-data-project/

https://www.arm.gov/news/features/post/88126

Manual & disconnected processes in ML workflow are time-consuming & computationally intensive

Complex data curation & post-analysis process are often required in EO-ML pipeline

Example of "manual" EO-ML lifecycle

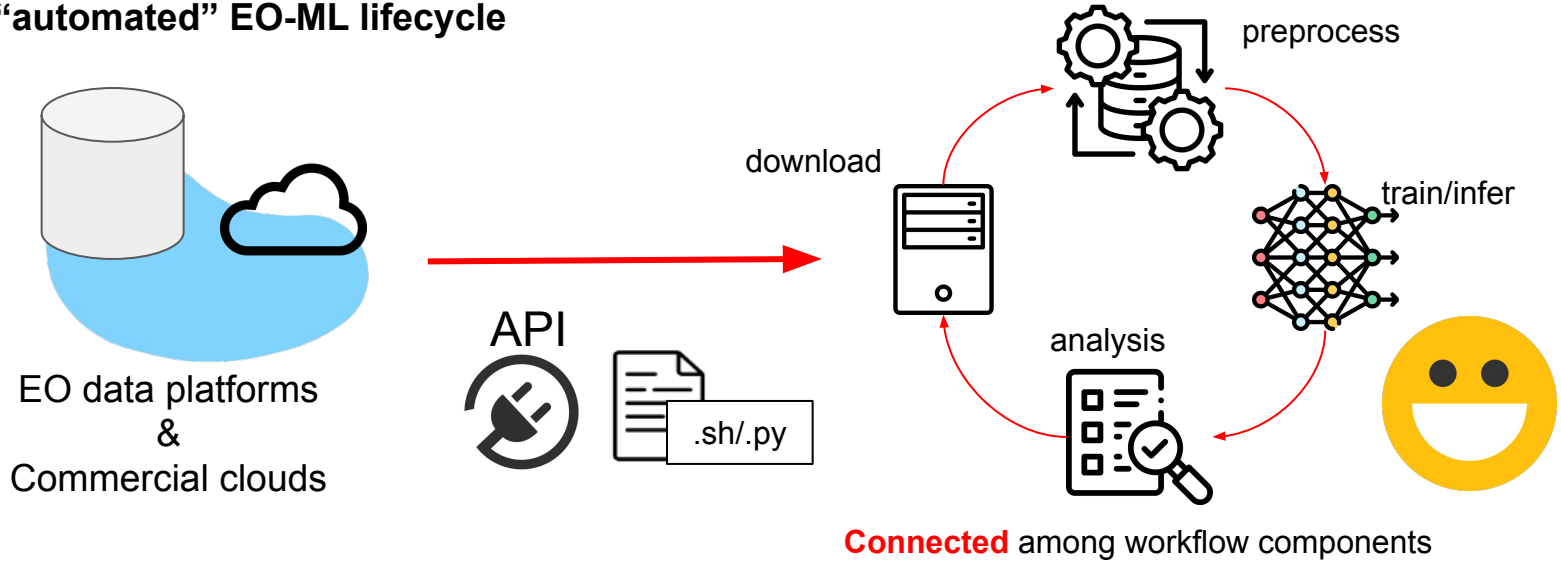


Disconnected among workflow components

Globus software tools to harness the potential of EO-ML workflow for accelerating scientific discovery

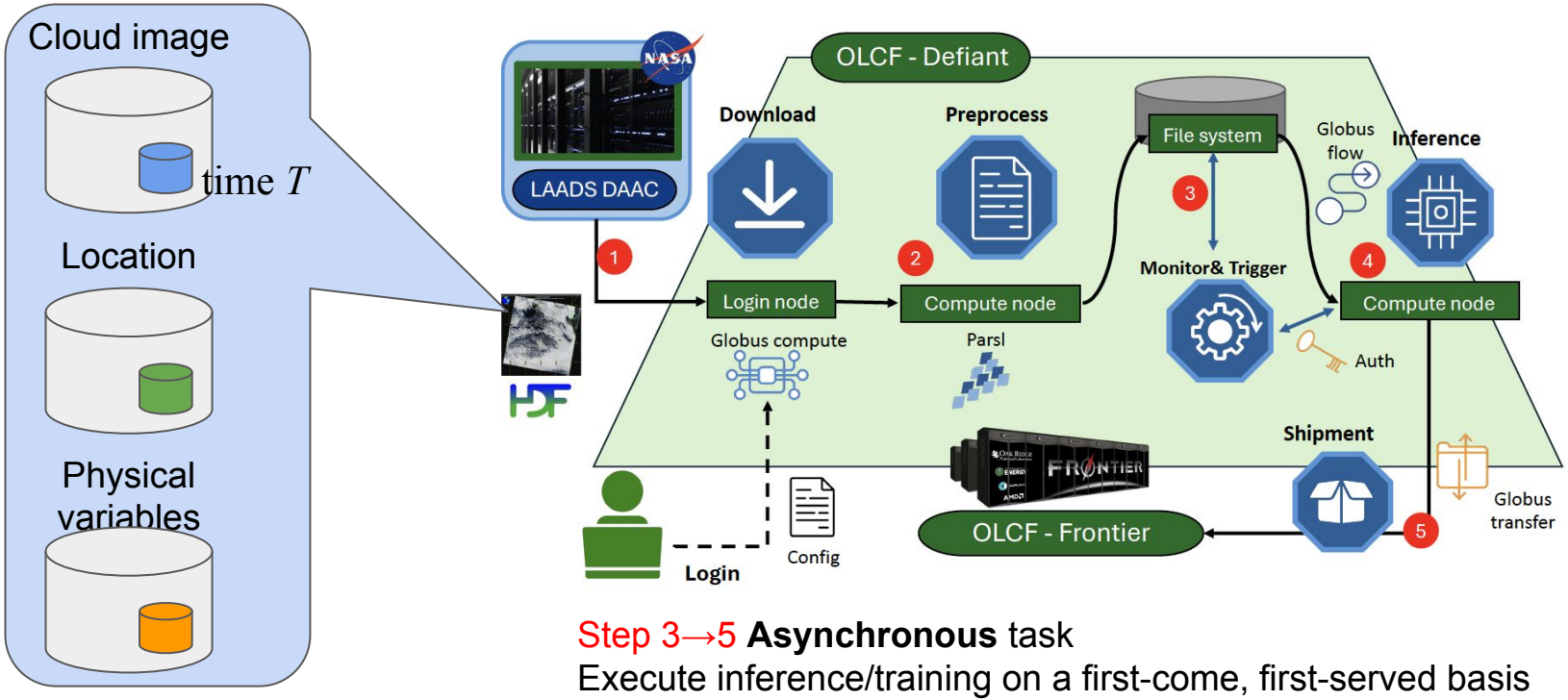
Orchestrate data collection, movement, and processing across multiple computational platforms.

Example of "automated" EO-ML lifecycle

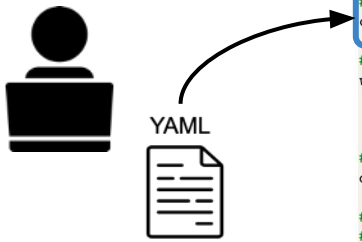


Target: *Self-supervised deep learning to classify satellite clouds images*

Total volume ~ 1PB



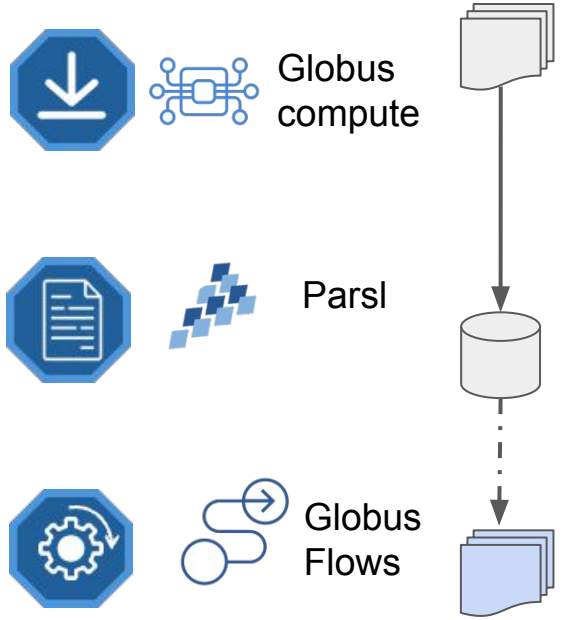
Globus Computes/Flow enable to easily create a seamless & flexibly multi-facility ML workflow for domain scientists



User defined workflow configurations

```yaml
# auth IDs
token: "user token generated from LAADS"
endpoint_id: "globus compute endpoint UUID"

# parameters for downloading
dowload:
    savepath: /path/to/save/downloading/data

# parameters for preprocess
preprocess:
    prefix: "MOD021KM"
    nodes: 2
    ncores: 10

# parameters for inference
inference:
    outputdir: /path/to/output
    flow_id: "globus flow UUID"
    modeldir: /path/to/ trained_AI_model
    compute_function_id: "inference function UUID"
```

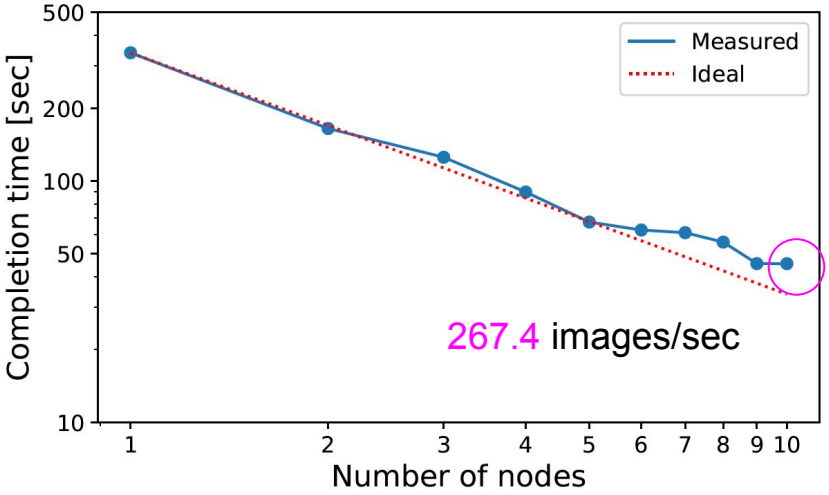
```python
# 0: Load YAML config file
config = load_config("./config.yaml")

# 1: Download
with Executor(config["endpoint_id"]) as gce:
    f = gce.submit(
            downloader,
            config["download"]["savepath"], config["token"])
## get a list of downloaded files
dl_files = f.result()

# 2: Preprocess
## Define parsl config
parsl_config=Config(
    HighThroughputExecutor(
        provider=SlurmProvider(
            # control number of workers launching
            nodes_per_block=config["preprocess"]["nodes"],
            cores_per_node=config["preprocess"]["ncores"],
            ...) )])

futures= [preprocessing(
            filename,
            "./config.yaml")
            for filename in dl_files]
## get result
results = [ r.result() for r in futures])

# 3 & 4 & 5: Monitor & Inference & Shipment
trigger = FileTrigger(
        config["donwnload"]["savepath"],
        FlowRunner=inference2transfer,
    )
## asynchronously trigger flow run to inference & shipment
trigger.run()
```



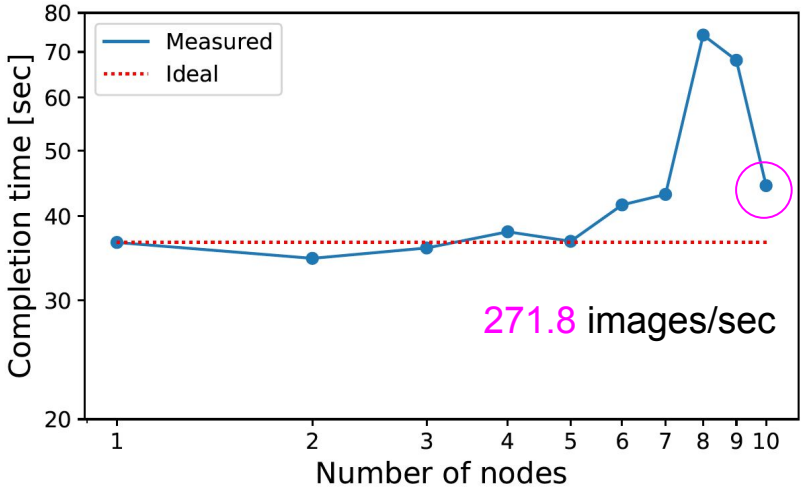
Parsl enables scaling data intensive pipeline

The preprocessing step of our EO-ML workflow achieves optimal scaling by increasing the number of nodes

Strong scaling

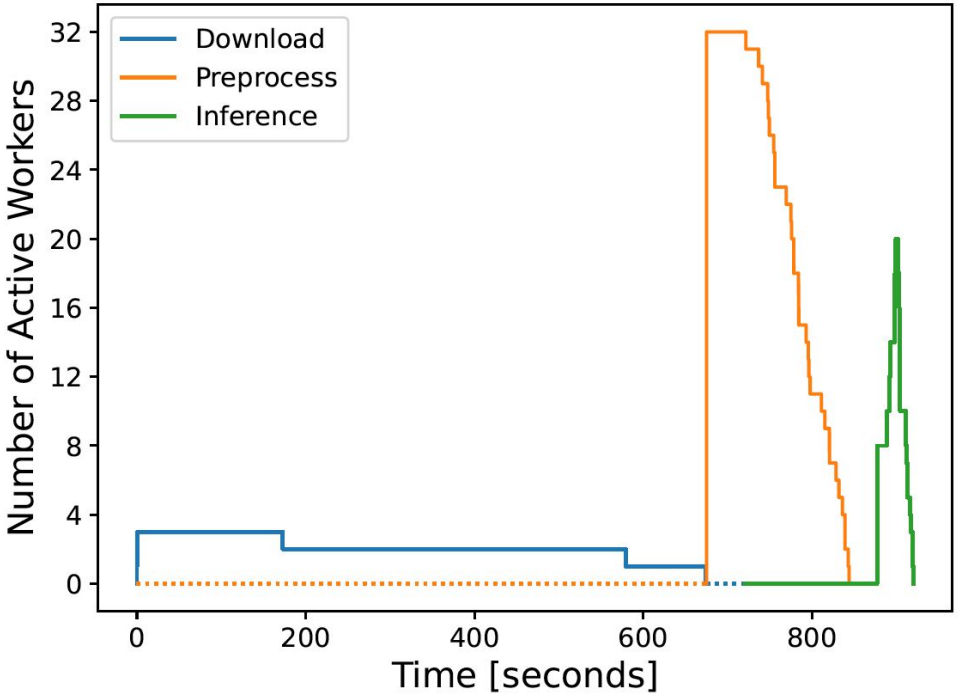


Weak scaling



Flexibly manage resources for workflow components

Specify different computing resources based on nature of tasks & machine availability.



Summary, Future works & Question

kurihanat@ornl.gov

Xloop @SC'24 

Summary:

- Demonstrated a deployment of EO-ML workflow on OLCF for peta-scale EO dataset
- Leveraged Globus Compute, Parsl, and Globus Flows to automate ML lifecycle
- Globus software tools provide excellent scaling performance and fast overhead times.

Future works:

- Complete more seamless workflow with Globus Flows
- Shareable and publicly accessible repository of complete workflows or individual workflow steps
- Multi-DOE facility EO-ML training workflow

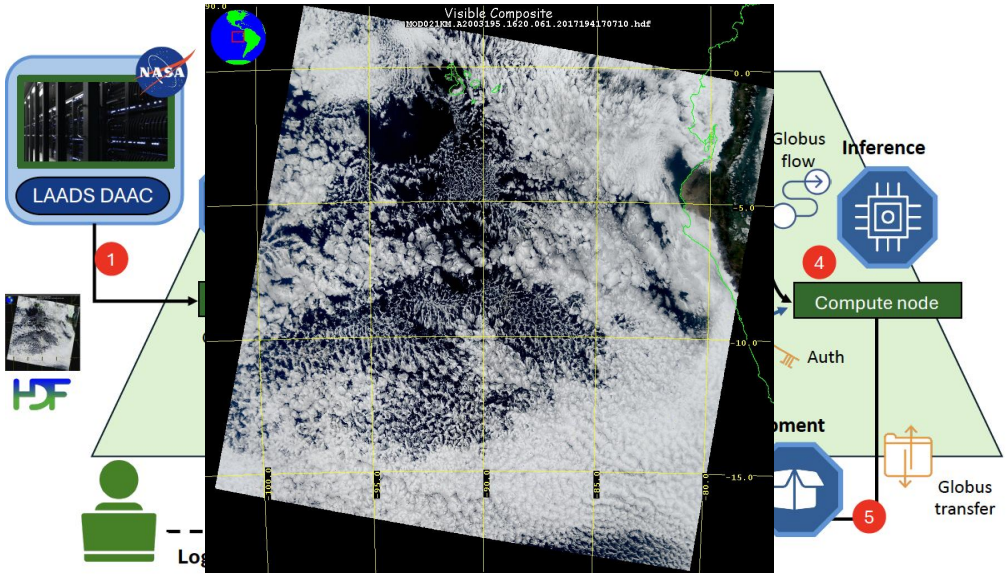
Globus software ecosystem automates **multi-facility** EO-ML workflow execution on OLCF Advanced Computing Ecosystem

Target application:
Self-supervised cloud classification → classify satellite clouds images

Problem state ***before*** automation:

- 3 different satellite image products (~850TB) from NASA
- Downloading was remotely executed by **funcX**
- Preprocess was accelerated by **Parsl**

Question: How to connect data pipeline over multiple facilities?



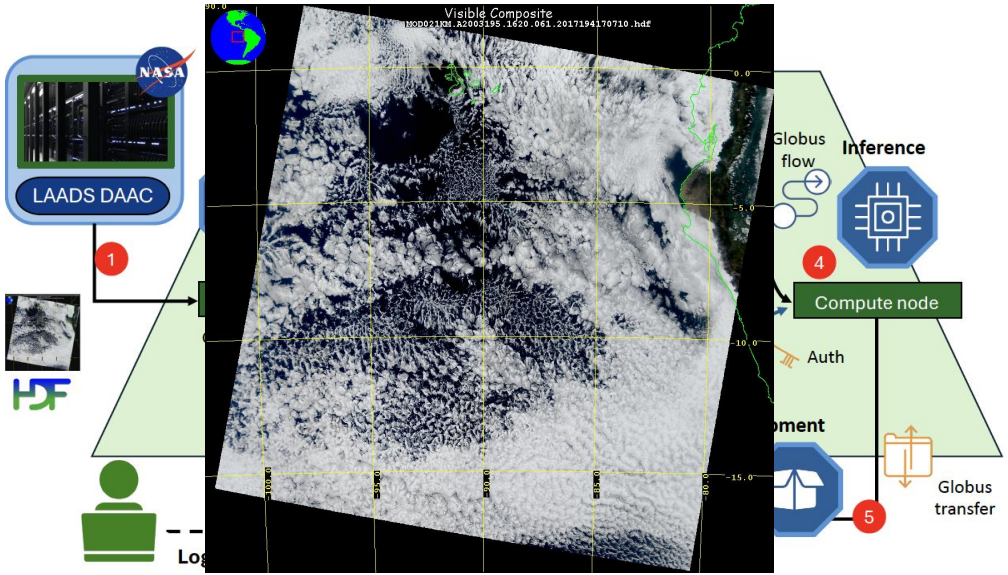
Globus software ecosystem automates **multi-facility** EO-ML workflow execution on OLCF Advanced Computing Ecosystem

Target application:
Self-supervised cloud classification → classify satellite clouds images

Problem state ***before*** automation:

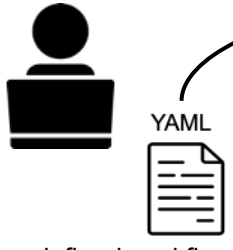
- 3 different satellite image products (~850TB) from NASA
- Downloading was remotely executed by **funcX**
- Preprocess was accelerated by **Parsl**

Question: How to connect data pipeline over multiple facilities?



Use Globus Computes/Flow to easily create a seamless & flexibly multi-facility ML workflow for domain scientists

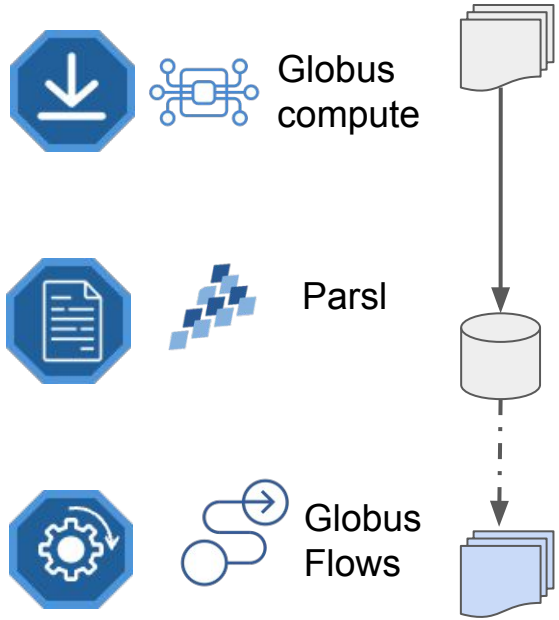
Previously disconnected ML pipelines are now integrated with Globus software tools ecosystem.



User defined workflow configurations

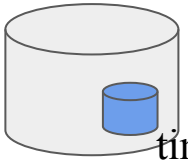
```yaml
# auth IDs
token: "user token generated from LAADS"
endpoint_id: "globus compute endpoint UUID"

# parameters for downloading
dowload:
    savepath: /path/to/save/downloading/data

# parameters for preprocess
preprocess:
    prefix: "MOD021KM"
    nodes: 2
    ncores: 10

# parameters for inference
inference:
    outputdir: /path/to/output
    flow_id: "globus flow UUID"
    modeldir: /path/to/ trained_AI_model
    compute_function_id: "inference function UUID"
```

```python
# 0: Load YAML config file
config = load_config("./config.yaml")

# 1: Download
with Executor(config["endpoint_id"]) as gce:
    f = gce.submit(
            downloader,
            config["download"]["savepath"], config["token"])
## get a list of downloaded files
dl_files = f.result()

# 2: Preprocess
## Define parsl config
parsl_config=Config(
    HighThroughputExecutor(
        provider=SlurmProvider(
            # control number of workers launching
            nodes_per_block=config["preprocess"]["nodes"],
            cores_per_node=config["preprocess"]["ncores"],
            ...) )])

futures= [preprocessing(
            filename,
            "./config.yaml")
            for filename in dl_files]
## get result
results = [ r.result() for r in futures])

# 3 & 4 & 5: Monitor & Inference & Shipment
trigger = FileTrigger(
        config["donwnload"]["savepath"],
        FlowRunner=inference2transfer,
    )
## asynchronously trigger flow run to inference & shipment
trigger.run()
```



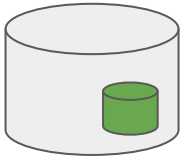
Target application:
Self-supervised deep learning → classify satellite clouds images

Cloud image



time T

Location



Physical variables