Pegasus Users Group
MEETING
GeoEDF: A Framework for Geospatial Research Workflows

Rajesh Kalyanam
Research Scientist, Purdue Research Computing

02/25/21
GeoEDF Vision

Researchers spend up to 80% of their time “wrangling data.”

Remote data repos, smart devices, streaming data

Geospatial Data Framework

Reusable Data Connectors
Pluggable Data Processors
Integrated Active Learning
Resource Interoperability Interfaces

GeoEDF Geospatial Data Framework (GUI & API)

Cyberinfrastructure
(Campus, XSEDE, HUBzero, Geospatial Tools, storage, Solr,...)

Make Science FAIR

OUR DATA WORKFLOW - Final
1. Go to the science gateway
2. Define "my_workflow.yml" (or use tool GUI if needed)
3. Ask GeoEDF to execute!
4. Data and workflow automatically published to science gateway

Remote data directly usable in code, seamless workflow
Complexity abstracted away
Reusable data connectors, processors, and workflows
Automatic provenance capture & data annotation => FAIR
The GeoEDF Project

An Extensible Geospatial Data Framework Towards FAIR Science

To help data-driven sciences to be more Findable, Accessible, Interoperable, Reusable

funded by NSF CSSI program award #: 1835822, Oct 2018 - Sep 2023
Multidisciplinary Project Leadership

Jian Jin
- Plant phenotyping & sensors
- Ag & Biological Engineering

Venkatesh Merwade
- Flood modeling & visualization
- Civil Engineering

Carol Song
- Cyberinfrastructure
- Research Computing

Jack Smith
- Water Quality & resource management
- Marshall University

Uris Baldos
- Sustainable development
- Agricultural Economics
Multidisciplinary domains often need to access diverse datasets and integrate them with existing models.
Reality!

Working through the specifics reveals the more messy details!
Streamline data wrangling in research workflows

Enable researchers to break down a complex research task into a collection of data acquisition and processing sub-tasks

Promote FAIR science principles

Integrate GeoEDF and cyberinfrastructure to implicitly & explicitly promote FAIR science principles

Provide reusable and scalable workflow building blocks

Improve the efficiency of day-to-day research workflows by enabling standardization, reuse, composition, and scalable execution
GeoEDF Components

**Reusable Data Connectors**
Implement various data access protocols, enable data acquisition from popular repositories.

**Reusable Data Processors**
Implement domain agnostic & domain specific geospatial processing operations.

**Plug-and-play Workflow Composer**
Enable the composition of individual connectors & processors into complex workflows.

**GeoEDF**
Enable researchers to conceive of geospatial data driven workflows as a sequence of data acquisition and processing steps that can be carried out using pre-existing or user contributed connectors and processors.
<table>
<thead>
<tr>
<th>Data Source</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>MODIS, SMAP, other Earthdata DAACs</td>
</tr>
<tr>
<td>USGS</td>
<td>Elevation, land use, hydrography, Gage, NLDI</td>
</tr>
<tr>
<td>USDA</td>
<td>Soil, land cover, land use</td>
</tr>
<tr>
<td>CUASHI</td>
<td>Rainfall, Hydroshare resources</td>
</tr>
<tr>
<td>EarthStat</td>
<td>Crop data</td>
</tr>
<tr>
<td>FAO</td>
<td>Arable land, harvest data</td>
</tr>
<tr>
<td>CIESIN</td>
<td>Population data</td>
</tr>
<tr>
<td>EPA</td>
<td>Water quality</td>
</tr>
<tr>
<td>Others (no API yet)</td>
<td>Open Data Cubes, Google Earth Engine, ESS-Dive</td>
</tr>
</tbody>
</table>
# Data Processor Examples

<table>
<thead>
<tr>
<th>Domain</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>Reproject, resample, format transformation, filter, mosaic, clip/mask, aggregate (spatial &amp; temporal), visualization, reclassification</td>
</tr>
<tr>
<td>Hydrology</td>
<td>Terrain analysis, flood models</td>
</tr>
<tr>
<td>Digital Ag</td>
<td>Query, spatial/temporal filter, ML training, decision support</td>
</tr>
<tr>
<td>Sustainability</td>
<td>Downsampling, (weighted) aggregate, FEWS models</td>
</tr>
</tbody>
</table>
Plug-and-play Workflow Composer

- **Workflow Framework** defining
  - Standardized interfaces for connectors and processors
  - Syntax and semantics of defining and composing instances of connectors and processors into scientific workflows

- **Workflow Engine** transforming
  - “Declarative”, abstract workflows into code executing on heterogeneous compute resources
GeoEDF in a nutshell

Workflows are sequences of connector and processor instances

Connector and Processor instances specified in YAML

Connector and Processor Python classes
Example Hydrologic Workflow

Apply GeoEDF principles

NASA DAAC

HDF File(s)

NASAIinput Connector

HDF File(s)

HDFEOSShapefileMask Processor

Shapefile

Weighted LAI per region

Watershed map

MODIS Leaf Area Index (LAI)

Mask + Aggregate

Hydrologic model
The GeoEDF Workflow

$1:
Input:
NASAINput:
url: http://files.ntsg.umt.edu/data/NTSG_Products/MOD16/MOD16A2.105_MERRAGMAO/{file}
user: rkalyana
password:
Filter:
file:
PathFilter:
   pattern: ‘Y%{year}/D001/*.h00v08*.hdf’
year:
DateTimeFilter:
   pattern: ‘%Y’
start: 01/01/2000
end: 12/31/2005
period: 1Y

$2:
HDFShapefileEOSMask:
hdf file: $1
shapefile: /home/rkalyana/subs1.shp

- Filters provide bindings for variables
- They promote modularization and can implement complex spatial and temporal filtering
- Filters help restrict the data that is actually “downloaded”
 Workflow Concretization using Pegasus

- Connectors need to bind filter variables in order; arbitrary number of variable bindings may be generated; each binding “retrieves” arbitrary number of files

- Processors may need to process an arbitrary number of files retrieved by a connector

- Each connector or processor turns into its own “sub-workflow”

- Top-level DAX builds and executes these sub-workflows as it goes

- Sub-workflows only transfer back data necessary to construct the next ”stage” sub-workflow; viz., filter values, file listing

- Final step returns outputs

- *Connectors/processors can have arbitrary software dependencies (containerization is a good idea!)

- **Public-private keypair generated for each workflow to encrypt sensitive strings (viz. any field left blank for user input in workflow definition)
Connector/Processor Contribution Process

(1) Contribute connectors/processors via GitHub PRs

(2) Detect changes, build Singularity container, push to registry server

(3) Query registry for list of connector, processor containers
Cyberinfrastructure Integration
Dec 2020
- Sample connectors, processors
- CI/CD pipeline in place
- Proof-of-concept Jupyter notebook based on Pegasus dev container

Feb 2021
- Execution on HPC
- Integration with MyGeoHub gateway

Mar 2021
- Release v1.0 available on MyGeoHub, PyPI, GitHub
- Sample end-to-end research workflows
- Documentation

Summer 2021
- Workflow monitoring
- Data annotation hooks
- CI interoperability and external CI install
Our Pegasus Feedback

**Cyberinfrastructure Integration**

- Best practices for setting up Pegasus to support (a) multiple users, (b) secure sensitive information (e.g., catalogs, keys)
- Middleware layer with a thin API interface?

**New Features**

- Support for conditionals, loop-until?
- High-level monitoring, i.e., what task in what sub-workflow is currently executing?
Thank You!

Where to find us:

- Project Repository: [https://github.com/geoedf](https://github.com/geoedf)
- MyGeoHub CI: [https://mygeohub.org](https://mygeohub.org)
- Email: Carol Song [cxsong@purdue.edu], Rajesh Kalyanam [rkalyana@purdue.edu]