GeoEDF: An Extensible Geospatial Data Framework for FAIR Science

Rajesh Kalyanam
Lan Zhao
Carol Song
rkalyana@purdue.edu
lanzhao@purdue.edu
cxsong@purdue.edu
Research Computing, Purdue University
West Lafayette, Indiana

Venkatesh Merwade
vmerwade@purdue.edu
Lyles School of Civil Engineering, Purdue University
West Lafayette, Indiana

Jian Jin
jianjin@purdue.edu
Agricultural and Biological Engineering, Purdue University
West Lafayette, Indiana

Uris Baldos
ubaldos@purdue.edu
Agricultural Economics, Purdue University
West Lafayette, Indiana

Jack Smith
smith1106@marshall.edu
Marshall University
Huntington, West Virginia

ABSTRACT

Collaborative scientific research is now increasingly conducted online in web-based research platforms termed "science gateways". Most science gateways provide common capabilities including data management and sharing, scientific code development, high performance computing (HPC) integration, and scientific workflow execution of varying automation. Despite the availability of scientific workflow frameworks such as Pegasus and workflow definition languages such as the Common Workflow Language (CWL), in practice typical workflows on science gateways still involve a mix of non-reusable code, desktop tools, and intermediate data wrangling. With the growing emphasis on FAIR (Findable, Accessible, Interoperable, Reusable) science, such mixed workflows present a significant challenge to ensuring compliance to these principles. These challenges are further compounded in the earth sciences where researchers spend inordinate amounts of time manually acquiring, wrangling, and processing earth observation data from repositories managed by organizations such as NASA, USGS, etc. Our extensible geospatial data framework, GeoEDF is designed to address these challenges, making remote datasets directly usable in computational code and facilitating earth science workflows that execute entirely in a science gateway. In this paper we describe the design of GeoEDF, current implementation status, and future work.

PEARC '20, July 26–30, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6689-2/20/07…$15.00
https://doi.org/10.1145/3311790.3396631

CCS CONCEPTS

• **Information systems** → **Geographic information systems**; **Computing platforms**; • **Applied computing** → *Earth and atmospheric sciences*; • **Software and its engineering** → **Development frameworks and environments**; *Software libraries and repositories.*

KEYWORDS

geospatial, data framework, scientific workflow, FAIR science

ACM Reference Format:
Rajesh Kalyanam, Lan Zhao, Carol Song, Venkatesh Merwade, Jian Jin, Uris Baldos, and Jack Smith. 2020. GeoEDF: An Extensible Geospatial Data Framework for FAIR Science. In *Practice and Experience in Advanced Research Computing (PEARC '20), July 26–30, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3311790.3396631

1 INTRODUCTION

Researchers in the earth sciences and various other interdisciplinary fields such as resource sustainability and agricultural economics often have to utilize diverse data sets in their simulation and modeling workflows. For instance, hydrologists utilize a combination of streamflow, land use, and land cover data from USGS, and precipitation and evapotranspiration data from NASA in flood modeling. Agricultural economists utilize crop yield data from the Food and Agriculture Organization of the United Nations (FAO), as well as population density and environmental data from the IPUMS Terra repository to predict the effects of climate change on food scarcity around the world. Due to the inherent massive volume, high dimensionality, heterogeneity of data formats, and variability in access protocols across these various repositories, the process of filtering, acquiring and transforming these data sources into datasets usable in a simulation model is often tedious and time consuming. Furthermore, researchers often have to repeat this data wrangling process several times if simulations need to be run for different spatial or temporal ranges. While most researchers are technically adept at

writing code to automate some of these processes, such software is typically custom-built and designed for a specific execution environment and/or research project and not immediately generalizable or reusable.

Consider also, the recent emphasis on the FAIR (Findable, Accessible, Interoperable, Reusable) principles for scientific research that are intended to enhance the discoverability and reusability of scientific artifacts. Manual data wrangling processes often result in some data artifacts relevant to a simulation result not being truly accessible, or lacking sufficient metadata to enable discoverability and research reproducibility. A natural and obvious solution is to transition scientific workflows to execute entirely in a science gateway, so that automated data annotation and publication can accompany workflow execution; simplifying the pathway to FAIR science.

Our GeoEDF framework is designed based on these considerations of simplifying, generalizing, and fostering reusability of scientific workflows while enabling FAIR science out-of-the-box. The GeoEDF framework is composed of three core elements: reusable *data connectors* that abstract away the specifics of acquiring data from large remote repositories; reusable *data processors* that abstract various domain-independent and domain-specific geospatial data processing building blocks; and *plug-and-play workflows* that can be declaratively composed of pre-existing data connectors and processors. The GeoEDF *workflow engine* transforms this declarative workflow for execution in a variety of target environments. We describe each of these elements and the technical implementation details in section 3. The broader cyberinfrastructure features enabling FAIR science are described in section 5. We conclude with a description of the current implementation status, discussion of broader impacts, and next steps.

2 BACKGROUND AND RELATED WORK

While most science gateways provide data management and scientific code development and execution capabilities, scientific workflows on such platforms are typically manual or fairly prescriptive in nature – often resembling a sequence of high performance computing jobs. By contrast, GeoEDF encourages researchers to approach scientific workflows from a high-level, logical perspective as a sequence of generalizable data acquisition and processing steps. Such generalization aids in effective reusability of common elements across different scientific workflows. For instance, acquisition of data for a specific time range from a particular NASA-hosted FTP server when viewed as a more general process of filtering and downloading a set of a files from a FTP server can then be applied to any FTP-based data repository. Similarly, the specific processing task of aggregating evapotranspiration data from a HDF file across the polygons in a watershed boundary dataset from an ESRI Shapefile can be more generally conceived as a "masking" operator applicable to any pair of files in the HDF and Shapefile formats. We should note here that geographic information systems (GIS) such as QGIS provide a large library of such domain-agnostic geospatial processing operations. Our intent in GeoEDF is not to replicate this entire library. In a high-level conception of geospatial scientific workflows, some operations may naturally correspond to typical processing operations supported by most GIS systems, however they are by no means exhaustive. Modern earth science workflows are composed of a diverse set of generalizable operations including statistical analysis, machine learning-based modeling and prediction, and calibration. We next provide a brief survey of prior work that is most closely related to GeoEDF and representative of common approaches towards scientific data management and workflow development thus far, illustrating some of the key gaps that GeoEDF fills.

2.1 GABBs/HUBzero

The GeoEDF project builds on existing software building blocks that enable researchers to manage geospatial data and conduct manual end-to-end workflows in a science gateway. These Geospatial Data Analysis Building Blocks (GABBs) [13] provide geospatial data management, visualization toolkits, and geospatial processing tools for science gateways built on the HUBzero [10] cyberinfrastructure (CI) platform. In addition to a feature-rich content management system (CMS), HUBzero provides a complete Linux execution environment that can be used to develop and deploy scientific tools that run in containers on scalable execution infrastructure. The HUBzero tool environment is also integrated with various campus and national high-performance computing (HPC) resources such as XSEDE and supports HPC job submission, monitoring, and data transfer.

Seamless integration between the GABBs geospatial data management building block and the HUBzero tool environment facilitates manual end-to-end workflows where HUBzero tools can be executed in sequence without requiring any intermediate data transfers. However, scientists are still responsible for bringing data from remote repositories to GABBs data management before it can be used in these workflows. While GeoEDF has been designed to be independent of any specific cyberinfrastructure framework, we use the HUBzero-based science gateway MyGeoHub [8] as the primary deployment and dissemination venue. GeoEDF will extend GABBs to support automated end-to-end workflows that no longer require manual data acquisition as a first step. We also leverage GABBs data management and its seamless integration with the HUBzero scientific tool environment to facilitate FAIR science compliance for scientific workflows in MyGeoHub. Details of this integration are described in section 5.

2.2 Cyberinfrastructures Supporting Workflows

While GeoEDF can be seen to share similarities with scientific workflow software, it is a level of abstraction removed from them. The GeoEDF workflow engine can and does leverage existing scientific workflow systems (specifically Pegasus [4]) to instantiate and execute the logical GeoEDF workflow. It can be argued that workflow systems such as Galaxy [5] and Kepler [3] do operate at the same level of abstraction as GeoEDF. However, these systems only support certain specific data access protocols or repositories, requiring the user to manually upload any other datasets to the workflow executor. Systems such as Apache Taverna [7] and Airavata [9] only support operations implemented as web-services with SOAP, WSDL, or REST interfaces. Moreover, these systems require data to

be transferred to the workflow execution servers via SSH or SCP data transfers.

2.3 Data and Software Frameworks

In the Clowder data management cyberinfrastructure, an extensible set of "extractors" can be used to process files either automatically (via a message queue monitoring file upload events) or on-demand. However, no straightforward approach exists for stringing extractors together into a pipeline. Cyverse's [6] Discovery Environment (DE) is the most closely related workflow system to GeoEDF. However, DE requires pipeline components to be Docker containers with heavily specialized user interfaces. In terms of geospatial data processing, recent systems such as GeoSPARK, Hadoop-GIS, and SpatialHadoop focus primarily on supporting big spatial data processing and only provide software for importing local files into big spatial data structures that support geospatial operations and transformations. The CyberGIS project [12] has developed several web-based platforms, applications, and libraries supporting data-intensive, scalable geospatial analytics capabilities. However, to the best of our knowledge, CyberGIS does not support declarative end-to-end workflows that can seamlessly access and process remote datasets.

3 GEOEDF FRAMEWORK

The GeoEDF framework is designed to enable earth science researchers to envision and formulate their scientific workflows as a logical sequence of data acquisition and processing steps which directly map onto GeoEDF connectors and processors respectively. The syntax and semantics of connectors (to a great extent) and processors (to a lesser extent) is designed with significant emphasis on their generality and reusability. We describe each of the core elements of the GeoEDF framework and the workflow engine that validates, instantiates, and transforms a logical workflow composed of connectors and processors for execution on compute resources.

3.1 Connectors

GeoEDF connectors abstract away the specifics of connecting to and transferring data from a data source. For instance, the user of a connector need not be aware of the implementation of a specific protocol for data query, filtering, and transfer; the implementation of authentication/authorization handshakes; and the setup and execution of iterative loops for batch file transfers. The design, syntax, and semantics of connectors is inspired by the concept of Logstash [1] plugins. Each connector can be composed of three different types of plugins: *input*, *filter*, and *output*. A connector is then a combination of exactly one input plugin, zero or more filter plugins, and at-most one output plugin. Before describing these components, we first present a motivating example for this design choice.

Consider the NASA MODIS Evapotranspiration dataset [2] which can be used to calculate the water energy balance or soil water status which are key to various hydrological simulations. This dataset is available as 8-day composites at a publicly accessible website with direct download URLs to individual data files such as those shown in Figure 1. Hydrological simulations typically range over different time periods or spatial regions (often involving one or more watersheds). Thus, a key data preparation step is to download all evapotranspiration data for the temporal and spatial region of interest of the simulation. Any programmatic or scripted approach to downloading the relevant data involves first determining the necessary values for key parts of the direct download URL in Figure 1 that encode the year, day of year, and MODIS grid identifier. In the case where a watershed represents the spatial region of interest, the watershed boundary will need to be transformed into the set of numeric identifiers of MODIS grids that intersect the watershed.

The straightforward approach to building a data accessor for evapotranspiration data would be to write code that accepts a temporal range (e.g. start and end dates) and a geospatial vector file encoding the watershed boundary as inputs, determines the download URL(s), and downloads the corresponding HDF files. However, a more generalizable and modular approach would be to separately develop code that downloads a file given its URL, and code that constructs the various salient pieces of that URL. In this specific case, these might include code that accepts a temporal range and returns a set of date strings (in a desired format), and code that accepts a vector file and returns a set of MODIS grid identifiers for grids that intersect the vector dataset. It stands to reason that this modular code can be reused out-of-the-box for a variety of URL combinations that include a date string (with different formats) and/or MODIS grid identifier at different locations in the URL. Broadly speaking, an input plugin implements a specific data access protocol (e.g. HTTP for acquiring files given a URL), while a filter plugin provides a set of values for use in the arguments to an input (or filter as we will show later) plugin. Output plugins are the same as input plugins except the direction of data transfer is reversed. Whereas filter plugins in Logstash filter individual data records ***after*** they have been retrieved by an input plugin, here the filter plugin is designed to filter a file repository's contents ***before*** any actual file transfer from the repository occurs.

3.1.1 Syntax and Semantics. GeoEDF input, filter, and output connector plugins are designed as Python classes implementing a standardized interface for their plugin type. The standardized interface enables the workflow engine to be agnostic to a plugin's specifics when executing it. Plugin classes can have an arbitrary set of instance variables relevant to the purpose of that plugin, each of which can either be required or optional. A connector is a composition of instances of plugin classes that satisfies certain syntactic and semantic rules. YAML syntax is used to encode this composition with YAML *nodes* representing plugin class instances and an associative key-value pair *mapping* under each node representing bindings for the class instance variables. Figure 2 illustrates one such connector definition consisting of an instance of the `NASAInput` plugin, and three filter plugins: `DateTimeFilter`, `StringFilter`, and `MODISFilter`.

A key syntactic rule for plugin composition has been outlined before: exactly one input plugin, zero or more filter plugins, and at-most one output plugin. The remaining syntactic and semantic rules primarily concern the use of filter plugins. Recall that a filter plugin provides a set of values for use in the arguments of other plugins. Syntactically in the YAML, this relationship is represented using a *variable* in a plugin's bindings and creating a separate YAML node for this variable, whose content is an instance of a filter plugin that

http://files.ntsg.umt.edu/data/.../Y2001/D001/MOD16A2.A2000001.h00v08.105.2013121200130.hdf

Figure 1: Direct download URL for a NASA MODIS Evapotranspiration HDF file for 01/01/2001. Note that the URL encodes the year, Y2001, day of year, D001, and MODIS grid number, h00v08 that the data in this HDF file corresponds to.

provides the variable's values. In the case of Figure 2, the variable `year` is bound by the `DateTimeFilter`, and the variable `file` is bound by the `StringFilter`. The `DateTimeFilter` generates date strings given a date format string and start and end date. Filter plugins can themselves have dynamic bindings via the use of other filter plugins. For instance, the variable `grid` here is being bound by the third `MODISFilter` plugin. The `MODISFilter` generates MODIS grid identifiers in the *h##v##* format (horizontal and vertical grid indices) given a region of interest in the form of an ESRI Shapefile. The use of filter plugins for dynamic binding is key to the generality and reusability of GeoEDF connectors. Rules governing the use of variables and filters include: requiring a variable to be bound to exactly one filter, disallowing circular dependencies, and disallowing variables to use the same name as the connector's plugins' instance variables.

3.2 Processors

GeoEDF processors resemble the standard computational unit in typical scientific workflows. They encapsulate some domain-independent (such as geospatial processing or data format conversion), or domain-specific processing operation (e.g. applying a flood model, downsampling to a different grid resolution). GeoEDF processors are again designed as Python classes. This does not restrict the generality of processors; Python's support for subprocess execution ensures that processors can encapsulate existing scientific code written in a variety of other programming languages. However, by this design choice, we promote the conceptualization of processors as generalized, reusable pieces of computation. For instance, a monolithic script that obtains elevation data from USGS as separate tiles, mosaics them together, determines a region of interest from a watershed Shapefile, and then clips the raster to this region of interest may make various assumptions about the datasets. These might include the geospatial map projection that has been used in the Shapefile and elevation data, the location where intermediate results have been saved, or the number of tiles that are usually returned. Separating out each of these operations as its own processor ensures both code reusability and adaptability to various execution environments. Just like connectors, using a processor involves defining an instance of a processor class in YAML syntax as illustrated in Figure 3. The primary validation in the case of a processor is to ensure all required instance variables have been bound and that no filter variables are used.

3.3 Plug-and-play Workflows

GeoEDF connectors and processors can be chained together into workflows where connectors acquire the necessary data which is then transformed and processed by various processors. The YAML syntax for workflows builds on the syntax for connectors and processors with numeric indices used to identify workflow stages and input-output dependencies between the various stages. One such workflow combining the connector and processor examples from before is illustrated in Figure 4. Each workflow stage has a numeric identifier (e.g. $1, $2) and the input-output dependency between the connector's data output and a subsequent processor is encoded by using the connector's identifier $1 as a variable binding in the processor. During execution, the $1 argument to the processor will be replaced with each file acquired by the connector.

3.3.1 Usage and Development. As part of the GeoEDF project, an initial set of connector plugin and processor classes are being developed for a few key science drivers. The connector plugins will include support for key data access protocols such as HTTP, FTP, and OpenDAP, and to access repositories hosted by organizations such as NASA, USGS, USDA, etc. Organizational repositories often require custom input plugins since they typically involve an authentication handshake before allowing data access. Processors will include certain common, domain-independent geospatial operations, file format converters, and domain-specific simulation and analysis code from the science use cases. GeoEDF is an open-source, community-driven project; contributors will be able to contribute connector and processor classes to a central repository so other users can utilize them in their scientific workflows. Our design for facilitating this open-source, community-driven approach will be presented in Section 4.

3.4 Workflow Engine

We are currently developing the GeoEDF workflow engine that is responsible for transforming a "logical" GeoEDF workflow expressed in YAML syntax into a concrete workflow that can be executed in different environments. After evaluating various scientific workflow systems, we have decided to utilize the Pegasus workflow management system [4] due to its support for dynamic workflows, diverse execution environments (through site configurations), and automated data staging and transfer optimizations. GeoEDF workflows are necessarily dynamic; connectors can retrieve an arbitrary number of files and efficiency dictates that each of these files are then processed in parallel by processors down the line. Thus, a GeoEDF workflow can only be concretized one workflow stage at a time. Pegasus supports dynamic workflows by allowing workflow jobs to dynamically generate "sub-workflows" which can then be executed after the job generating them has run.

3.4.1 Pegasus Workflow Generation. The GeoEDF workflow engine will parse a GeoEDF YAML workflow and create and execute the necessary Pegasus workflow jobs. As part of this transformation process, an administrative job is inserted at the beginning of each workflow to create a staging directory to store intermediate files during the workflow's execution. Each GeoEDF workflow stage (corresponding to a connector or processor) will be converted into

```
Input:
  NASAInput:
    url: http://files.ntsg.umt.edu/data/.../Y%{year}/D001/%{file}
    username: john
Filter:
  year:
    DateTimeFilter:
      start: 01-01-2001
      end: 01-01-2005
      pattern: yyyy
  file:
    StringFilter:
      pattern: MOD16A2.A2000001.%{grid}.105.2013121200130.hdf
  grid:
    MODISFilter:
      shapefile: watershed.shp
```

Figure 2: YAML definition of a connector that can be used to download MODIS evapotranspiration data from a NASA data repository

```
HDFShapefileMask:
  hdffile: MOD16.hdf
  shapefile: watershed.shp
```

Figure 3: YAML definition of a processor that "masks" a HDF file with a Shapefile, generating a new Shapefile that contains weighted averages of HDF file data for each of the polygons in the input Shapefile

a Pegasus sub-workflow. When transforming a connector into a Pegasus sub-workflow, the variable-filter dependency chain is utilized to set up a sequence of jobs that execute each filter plugin as soon as it is fully bound, before the input plugin and finally the output plugin is executed. When transforming a processor, parallel jobs will be created based on the set of possible bindings for any workflow stages referenced by their index (e.g. $1). If a connector's plugins reference prior workflow stages, the corresponding sub-workflow will contain parallel sequences of jobs.

In the case of the GeoEDF workflow in Figure 4, the sub-workflow for the connector would contain parallel jobs for the `MODISFilter` and `DateTimeFilter`, followed by a job for the `StringFilter`. The job for the `NASAInput` plugin would be executed last by having it depend on completion of the `DateTimeFilter` and `StringFilter` jobs. For the `HDFShapefileMask` processor, as many parallel jobs would be created as there are files returned by the `NASAInput` connector after execution. In order to determine the number of parallel jobs to be created for a particular workflow stage, each sub-workflow will include a "merge" job that collects and merges the names of each of the files generated by the jobs run in that sub-workflow. This merge job outputs a text file with this merged list of files which is then used by the job generating the sub-workflow for the next stage. Since workflow stages can reference zero or more prior stages, the results of the merge jobs from each of the referenced prior stages will be used to construct the appropriate combinatorial set of bindings (and corresponding parallel jobs) for the current stage. The Pegasus workflow that will be produced after the transformation of the GeoEDF workflow in Figure 4, is depicted in Figure 5.

3.4.2 Pegasus Workflow Execution. Since both connectors and processors can conceivably produce a significant number of large files, we will seek to avoid any unnecessary data transfers back and forth between the workflow submission host and the execution environment. The administrative job at the beginning of the workflow will create a data directory on the execution site to hold the data files from each stage so subsequent workflow stages can access them. Since the generation of sub-workflows and their "submission" as a job needs to occur on the submission host, the outputs of the merge jobs are the only intermediate files that will be transferred back. Jobs executing filters will return the set of values generated by the filter as output.

Since processor classes can include arbitrary software dependencies or code that may have been developed in other programming languages, we will containerize them as Singularity containers to enable portability and ease of execution in diverse HPC environments. Since we anticipate connector plugins to be purely-Pythonic, we will use a single Singularity container but separate Conda environments to manage the dependencies for each plugin class. Bind

```
$1:
  Input:
    NASAInput:
      url: http://files.ntsg.umt.edu/data/.../Y%{year}/D001/%{file}
      username: john
  Filter:
    year:
      DateTimeFilter:
        start: 01-01-2001
        end: 01-01-2005
        pattern: yyyy
    file:
      StringFilter:
        pattern: MOD16A2.A2000001.%{grid}.105.2013121200130.hdf
    grid:
      MODISFilter:
        shapefile: watershed.shp
$2:
  HDFShapefileMask:
    hdffile: $1
    shapefile: watershed.shp
```

Figure 4: YAML definition of a workflow that combines the connector and processor from Figures 2 and 3

mounts will be used to make the job data directory available in these containers during execution.

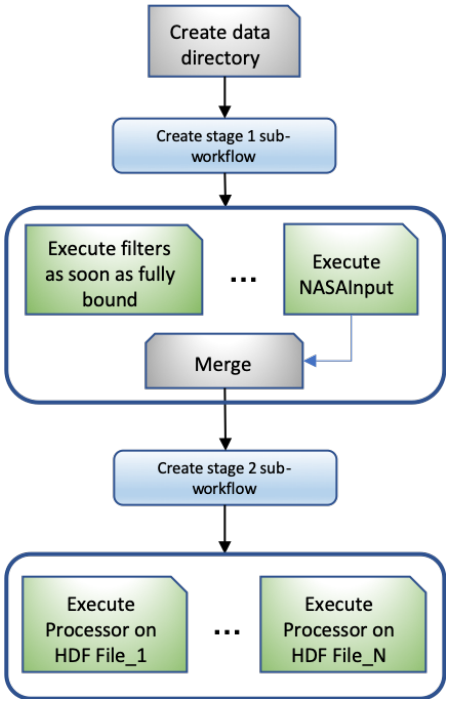

Figure 5: Pegasus workflow corresponding to Figure 4

4 CYBERINFRASTRUCTURE INTEGRATION

We will utilize the MyGeoHub science gateway [8] to develop, test, and deploy GeoEDF. The GeoEDF workflow engine will be installed in MyGeoHub's tool environment to enable users to develop and execute workflows from inside scientific tools and Jupyter notebooks. Local files from the tool environment that are referenced in the workflow will be automatically transferred to the execution environment using Pegasus' data staging capabilities. Due to the seamless integration between GABBs data management and the HUBzero tool environment, any data files managed by GABBs can be used directly in GeoEDF workflows. A small Condor pool will be set up in commercial cloud for testing purposes, while campus clusters at Purdue University and the XSEDE community clusters will be used to allow users to utilize their own cluster allocations to execute their workflows. By default, the final results of executing a workflow will be transferred back to the tool environment. Users can then use the seamless integration between GABBs data management and HUBzero tools to create data publications or share results with other users of MyGeoHub. Connectors will also be developed to utilize the GABBs data service APIs to transfer data from and to MyGeoHub's GABBs-managed data storage to automate this sharing process.

A Singularity registry will be created to host the Singularity containers for the connectors and processors developed for GeoEDF's driving science use cases. Standard open-source contribution pipelines on GitHub will be used to manage connector and processor contributions. A continuous integration/continuous deployment (CI/CD) pipeline will be established using GitHub actions to automatically create and deploy Singularity containers to this registry as connectors and processors are updated or contributed. A standard GitHub

repository template will be provided for connectors and processors so that useful metadata and documentation on their usage can be recorded and processed automatically to produce GitHub pages. These pages will form the primary resource for users to discover and identify existing connectors or processors that can be used for their own workflows. The integration of GeoEDF with MyGeoHub is illustrated in Figure 6.

5 SUPPORTING FAIR SCIENCE

GeoEDF will facilitate FAIR (Findable, Accessible, Interoperable, Reusable) science by automatically associating sufficient annotation with workflow results and by leveraging existing GABBs and HUBzero metadata capabilities. Both the GABBs data management building blocks and the HUBzero data publication process have developed certain key capabilities in metadata management, discoverability, and persistent identification. HUBzero provides a straightforward path from GABBs-managed data storage to persistent data publication with associated digital object identifiers (DOIs). GABBs data management enhances this publication process by automatically transferring any metadata from files in the active data storage to the corresponding data publication. All GABBs-managed metadata uses the Dublin Core Metadata Initiative (DCMI) standard and is indexed to HUBzero's Apache Solr index to aid data discoverability.

The GABBs data management building block automatically processes geospatial files on creation to extract metadata. Most standard geospatial file formats allow for metadata to be encoded along with the file's contents. Such metadata can include the geospatial bounds of the file, projection information, data description, authorship, and creation history. This metadata is extracted and then marshalled into the DCMI format and indexed to the Solr server. Programmatic retrieval and update of this metadata for GABBs-managed files is also supported by the GABBs data service APIs.

In GeoEDF, we will expand on these prior capabilities by developing a similar data service API for HUBzero publications to support publication discoverability and metadata retrieval. In addition, workflows results that are transferred to GABBs-managed storage via a connector will be automatically annotated with provenance details of the workflow through the GABBs data service APIs. This metadata will then be transferred over on data publication.

6 DISCUSSION

GeoEDF is driven by use cases from scientific domains that heavily utilize earth science data. We briefly describe how research workflows in these domains with significantly benefit from the use of GeoEDF.

6.1 Hydrology

In hydrology, researchers often utilize a wide variety of earth science data from different resource providers. Flood simulations require evapotranspiration and vegetation (leaf cover) data from NASA, peak streamflow and land elevation data from USGS, and SWAT (Soil Water Assessment Tool) models from other cyberinfrastructure (CI) platforms such as Hydroshare [11]. In addition to ensuring that all these datasets are in a common geospatial projection and resolution, various filtering, clipping, and aggregation operations need to be performed before using these datasets in the simulation models. This whole process will need to be repeated for simulations over different time periods or regions of interest. So far, hydrologists typically utilize desktop tools such as QGIS for some of these data preparation steps, and then move to a cyberinfrastructure platform to run the simulations using HPC resources. GeoEDF will completely automate such workflows using the connectors and processors that we have described previously and by providing additional connectors to access resources published by CI platforms such as Hydroshare, avoiding the need for any intermediate manual data transfers.

6.2 Global Resource Sustainability

In agricultural economics, the study of global resource sustainability involves combining data from diverse domains. Global crop yield data published by the U. N. FAO is combined with global administrative boundary data; population and income data from the World Bank; and land cover, land use data from IPUMS TERRA in a general equilibrium model. Since these datasets originate from different domains and organizations with their own data standards and access methods, a significant amount of pre-processing is required before the model can be applied. These pre-processing steps include operations such as downsampling data to a desired grid resolution, aggregating over administrative boundaries, and transforming the results into custom data formats that are required by the equilibrium model. Researchers currently utilize custom code written in various programming languages (Python, R) to perform each of these steps, some of which are carried out in Windows desktops due to the availability of certain data format conversion tools. GeoEDF will completely automate such workflows through the use of connectors to acquire data from the various data sources and processors for geospatial operations such as downsampling and aggregation, data format conversions, and wrapping the custom R code.

6.3 Crop Health Assessment

Recent advances in low-cost handheld devices are providing novel means of assessing crop health so various disease and nutritional deficiency issues can be immediately discovered and mitigated, thus improving crop yields. Hyperspectral imaging sensors attached to a mobile device are now used to image leaves from corn and soybean plants to determine parameters such as nitrogen and water content at high resolution levels. By making these devices available to farmers in the field, point *sensor* data can be collected from various assessment locations in a field and streamed to a central storage database. These data points will need to be collated, processed, and analyzed to uncover and visualize spatial trends at different scales (farm, county, state, etc.). A key use of these data points is in the development of a predictive machine learning (ML) model that can utilize such crop parameters to predict disease markers. However, before the data points can be used in a ML model, they need to be standardized to account for the effect of various environmental factors on the measurements. Cloud cover, temperature, and time of day can all affect the imaging results; thus, it is necessary to standardize the measurements to a fixed time of day and year. GeoEDF will help with both access of various subsets of the collected data

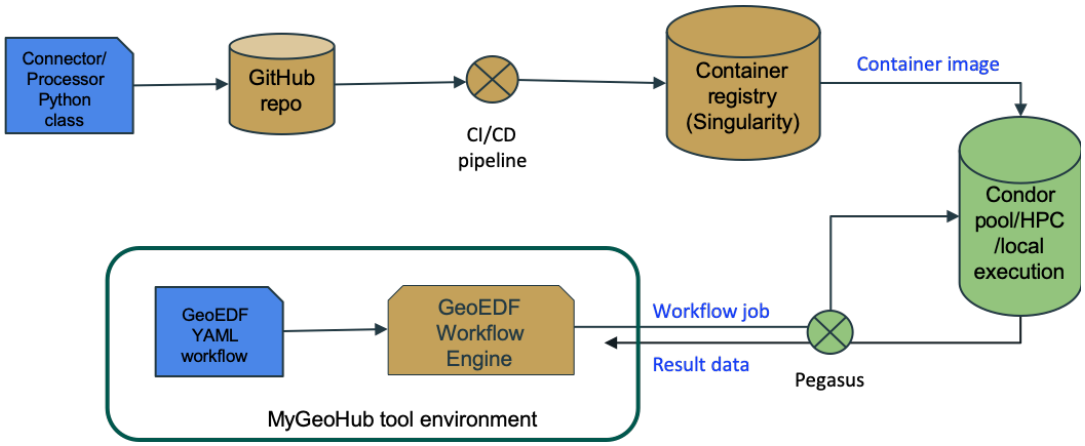

Figure 6: Integration of GeoEDF with the MyGeoHub cyberinfrastructure platform

for different temporal and spatial ranges, as well as with automating the calibration workflow that utilizes both the collected and environmental data to first standardize measurements and then run ML models.

6.4 Water Quality Asssessment

Water quality data is often crowd-sourced from various field sensors deployed in streams and assessment stations. Due to the wide variety of sensors deployed, some of them may stream data directly to a data repository, others may need manual data collection at periodic intervals and conversion into tabular forms, and yet others might require the use of proprietary data access APIs to pull data on-demand from the sensors. The EPA water quality portal (WQP) provides a repository of water quality monitoring data collected from contributors all across the country. Contributors use the Water Quality Exchange (WQX) to submit data to WQP. Since sensors can provide data in various custom formats, various preprocessing and data format conversion steps need to be performed before the collected data can be contributed through the WQX. GeoEDF can help in filtering and accessing data from the various data stores containing data from these sensors and converting them into the format required for WQX through appropriate connectors and processors, aiding in the timely transfer of data to WQP.

7 CURRENT STATUS AND NEXT STEPS

We have completed the design of the YAML syntax for GeoEDF workflows and developed an initial set of connectors and processors for the science use cases described in Section 6. We have also evaluated the creation and execution of dynamic workflows through the Pegasus Python API. We are currently working on the set up of a Singularity registry and the CI/CD pipeline for automatically building Singularity containers for connectors and processors. We are also developing the GeoEDF workflow engine to convert GeoEDF YAML workflows into Pegasus workflows. We anticipate releasing a beta version of the workflow engine in May 2020 as open-source on GitHub.

ACKNOWLEDGMENTS

This project is funded by the NSF award no. 1835822.

REFERENCES

[1] [n.d.]. Logstash. https://www.elastic.co/logstash Accessed 02/02/20.
[2] [n.d.]. MODIS Evapotranspiration Dataset. https://modis.gsfc.nasa.gov/data/dataprod/mod16.php Accessed 02/02/20.
[3] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. 2004. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* IEEE, 423–424.
[4] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira Da Silva, Miron Livny, et al. 2015. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* 46 (2015), 17–35.
[5] Jeremy Goecks, Anton Nekrutenko, James Taylor, Galaxy Team, et al. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 11, 8 (2010), R86.
[6] Stephen A Goff, Matthew Vaughn, Sheldon McKay, Eric Lyons, Ann E Stapleton, Damian Gessler, Naim Matasci, Liya Wang, Matthew Hanlon, Andrew Lenards, et al. 2011. The iPlant collaborative: cyberinfrastructure for plant biology. *Frontiers in plant science* 2 (2011), 34.
[7] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, and Tom Oinn. 2006. Taverna: a tool for building and running workflows of services. *Nucleic acids research* 34, suppl_2 (2006), W729–W732.
[8] Rajesh Kalyanam, Lan Zhao, Carol Song, Larry Biehl, Derrick Kearney, I Luk Kim, Jaewoo Shin, Nelson Villoria, and Venkatesh Merwade. 2019. MyGeoHub—A sustainable and evolving geospatial science gateway. *Future Generation Computer Systems* 94 (2019), 820–832.
[9] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, et al. 2011. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments.* 21–28.
[10] Michael McLennan and Rick Kennell. 2010. HUBzero: a platform for dissemination and collaboration in computational science and engineering. *Computing in Science & Engineering* 12, 2 (2010), 48–53.
[11] David G Tarboton, Ray Idaszak, Jeffery S Horsburgh, Jeff Heard, Dan Ames, Jonathan L Goodall, Larry Band, Venkatesh Merwade, Alva Couch, Jennifer Arrigo, et al. 2014. HydroShare: advancing collaboration through hydrologic data and model sharing. (2014).
[12] Shaowen Wang. 2010. A CyberGIS framework for the synthesis of cyberinfrastructure, GIS, and spatial analysis. *Annals of the Association of American Geographers* 100, 3 (2010), 535–557.
[13] Lan Zhao, Carol X Song, Rajesh Kalyanam, Larry Biehl, Robert Campbell, Leif Delgass, Derrick Kearney, Wei Wan, Jaewoo Shin, and I Luk Kim. 2017. GABBs-Reusable Geospatial Data Analysis Building Blocks for Science Gateways.. In *Gateways 2017.*