Clowder: Open Source Data Management for Long Tail Data

Luigi Marini
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
lmarini@illinois.edu

Indira Gutierrez-Polo
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
indiragp@illinois.edu

Rob Kooper
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
kooper@illinois.edu

Sandeep Puthanveetil
Satheesan
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
sandeeps@illinois.edu

Maxwell Burnette
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
mburnet2@illinois.edu

Jong Lee
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
jonglee1@illinois.edu

Todd Nicholson
University of Illinois at
Urbana-Champaign
Coordinated Science Laboratory
Urbana, Illinois
tcnichol@illinois.edu

Yan Zhao
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
yanzhao3@illinois.edu

Kenton McHenry
University of Illinois at
Urbana-Champaign
National Center for Supercomputing
Applications
Urbana, Illinois
mchenry@illinois.edu

ABSTRACT

Clowder is an open source data management system to support data curation of long tail data and metadata across multiple research domains and diverse data types. Institutions and labs can install and customize their own instance of the framework on local hardware or on remote cloud computing resources to provide a shared service to distributed communities of researchers. Data can be ingested directly from instruments or manually uploaded by users and then shared with remote collaborators using a web front end. We discuss some of the challenges encountered in designing and developing a system that can be easily adapted to different scientific areas including digital preservation, geoscience, material science, medicine, social science, cultural heritage and the arts. Some of these challenges include support for large amounts of data, horizontal scaling of domain specific preprocessing algorithms, ability to provide new data visualizations in the web browser, a comprehensive Web service API for automatic data ingestion and curation, a suite of social

annotation and metadata management features to support data annotation by communities of users and algorithms, and a web based front-end to interact with code running on heterogeneous clusters, including HPC resources.

CCS CONCEPTS

• **Information systems** → **Data provenance**; **RESTful web services**; • **Computer systems organization** → **Cloud computing**; • **Applied computing** → *Document metadata*;

KEYWORDS

data management, data curation, scientific gateways, metadata management, linked data

ACM Reference Format:

Luigi Marini, Indira Gutierrez-Polo, Rob Kooper, Sandeep Puthanveetil Satheesan, Maxwell Burnette, Jong Lee, Todd Nicholson, Yan Zhao, and Kenton McHenry. 2018. Clowder: Open Source Data Management for Long Tail Data. In *PEARC '18: Practice and Experience in Advanced Research Computing, July 22–26, 2018, Pittsburgh, PA, USA*. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3219104.3219159>

1 INTRODUCTION

The past decade has seen a proliferation of scientific gateways and software to create them. Scientific gateways provide web based access to framework of software tools, traditionally High Performance Computing (HPC) resources, for researchers to run applications on remote resources [33]. While traditionally projects like TeraGrid

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA

© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.

ACM ISBN 978-1-4503-6446-1/18/07...\$15.00

<https://doi.org/10.1145/3219104.3219159>

have focused on access to infrastructure for computational sciences [9], projects like XSEDE have started focusing on non-traditional HPC computational use cases [22]. For example, many researchers in data-driven fields struggle with how to best manage their data products and analytics. With more instruments producing data at higher rates, research communities have been struggling to adopt effective data management plans and technologies to store, curate and share their data products. Part of the challenge is the long tail of data, where different communities, small research teams and individual experiments have specific requirements on how the data is stored and catalogued [16]. Each community or individual scientist working on small projects produce a large portion of the total scientific output and do not have many resources to make the data accessible to the wider public [35]. With funding agencies requiring data management plans upfront [13, 25], storing data on local hard drives is not feasible anymore. Cloud storage solutions such as DropBox, Google Drive, etc. can only provide basic support for smaller files organized in hierarchies of folders, but fall short when it comes to metadata curation, access control, sharing and publishing for long term preservation of potentially large files and collections.

A wealth of information and data is accumulated over time as researchers execute experiments and collect data from instruments. While this data is being analyzed, users need simple ways to access and share intermediate data products. These potentially volatile data products are important during the research phase and require flexible access control policies. Data transfer is also an issue for use cases with large amounts of data being produced. Costs for data transfer can be prohibitive for small labs. Simulations running on HPC resources, sensors continuously monitoring the environment and instruments deployed in the labs can produce large volumes of data. Researchers might be better served with a data management system deployed next to the computational resource or physical instruments. Furthermore, each research team has their own metadata standards and analysis codes that they want to apply to the data as it is uploaded to the system.

Clowder aims at simplifying the task of setting up local instances of data management systems, next to the data or instruments and accessible over the web, and Clowder lets each community customize each instance with analysis code, metadata standards and web visualizations to simplify data storing, access and sharing within a community. Integration with existing systems and software requires a flexible architecture and the ability to deploy to different hardware infrastructures. The framework attempts to fulfill these requirements by leveraging stable open source software, supporting flexible metadata definitions and offloading expensive operations to a cloud based extraction bus. Clowder has been developed and leveraged in a wide variety of use cases, including material science research [24], cultural heritage [31], digital curation [23, 26], clinical research informatics [36], geosciences [11], environmental monitoring [2, 15], digital humanities [18, 29], social science [28], as well as industry collaborations.

In this paper we describe the Clowder framework. Section 2 provides a very brief overview of related work. Section 3 describes the core Clowder data model and how resources can be shared. Section 4 explains how Clowder stores metadata. Section 5 describes how

the extraction bus is used to enable auto-curation and information extraction. Section 6 shows how Clowder supports developing data visualizations previewers in the web browser. Section 7 describes the software architecture and implementation of the system. Section 8 concludes with lessons learned and future plans.

2 RELATED WORK

The software landscape of tools and services for data management is rich with options, each focusing on specific needs [19]. CKAN¹ is an open source solution that each institution can deploy independently [6], it is easily extensible through its API. It allows files to be created under an organization account instead of individual user accounts and supports different roles for user authorization. One of its unique features is the ability of tracking a unique trail of the updates on a dataset by showing all changes since its deposit [34]. Zenodo² is a free central service and open source software. Figshare³ is a central service free up to a certain level of use and closed source. All three services have options for private storage and allow researchers to determine when to make their datasets public.

The aforementioned systems focus on the publication of the final data products and the generation of Digital Object Identifiers. They lack community features for interaction during the research process or tools for validating that all necessary metadata has been added to the objects before final publication. DSpace⁴ is an exception with its support of domain-specific metadata schemas. CKAN does allow "extras" key-value pairs to be used for storing domain specific metadata, but does not follow any formal ontologies for these fields. In B2Share [7] its possible to setup custom metadata entries when selecting a project where the dataset will be uploaded into [6].

DataONE [20] provides distributed infrastructure to enable collaboration across different domains focused on biological and environmental communities. It leverages existing data repositories referred as *Member Nodes* that implement the DataONE API. There are also specialized nodes called *Coordinating Nodes* where more sophisticated services are provided including duplication of data across nodes, store mirrored contents of the metadata, manage access control rules, and identities from different providers. With this infrastructure it is able to provide a global, federated data network.

The SeedMe platform [10] allows users to upload an arbitrary number of data files and organizes them into collections. The owner of the data is limited to allowing other users to view those files, no user besides the owner can update the collection. Metadata can be added to collections by users and threaded discussion can take place in the files to facilitate collaboration. This platform is focused on collecting the results of HPC jobs and automated video encoding to assemble image sequences into videos at multiple resolutions.

3 DATA MODEL

Clowder helps manage data of any type by organizing files and metadata in folders, datasets, collections and spaces (Fig. 1). Datasets are the main resource type and contain files, folders and generic metadata. Each dataset has a name, description, owner and license.

¹<https://ckan.org/>

²<https://zenodo.org/>

³<https://figshare.com/>

⁴<https://www.duraspace.org/dspace/>

Optionally, datasets and files can be tagged with keywords for discovery, receive comments by users, and include generic metadata added manually by users or automatically by metadata extractors. Files can be uploaded to datasets and organized within folders. Files also include ownership information, license, tags, comments, and generic metadata. Collections of datasets can only contain datasets and nested collection but have no generic metadata associated with them. Collections provide ways of organizing datasets. All resources can be created, edited and deleted via the web service API or using the web interface.

Spaces enforce access control and allow administrators of a space to share datasets and collections with specific users using role-based access control. Possible actions within the system may require a specific permission, for example adding and removing datasets from a space, or editing information about the space. Permissions are organized in roles which are assigned to users in spaces. The default permissions are: viewer, editor and administrator. Custom permissions can be defined by an administrator by specifying a new set of permissions. For example, the SEAD project [23] defines a contributor role. Users with this role are able to add data, but can't edit or delete them. There are four different types of spaces available: personal, private, trial and public. Each user has a personal space whose contents are not visible to other users. In a private space, only people that are part of the space can see the details and files within the space, but other users within Clowder can request access to it and see some basic information about the space, including the name, description and logo. The contents of public spaces are open to anyone. Trial spaces have to be approved by a Clowder administrator before they are visible by other users not in the space. These permissions can be overwritten at the dataset level within a space.

Clowder also includes a set of social annotation features, allowing users to tag and leave comments on files and datasets. Users can follow other users, files, datasets, collections, and spaces. Each user has a home page, where users are notified of events associated with the resources they have created or followed. Events include when a dataset was created, deleted, added or removed from a space, a file was uploaded, metadata was updated, users were added to a space, and similar events for collections and files. Users can choose to be notified by email of new events hourly, daily or weekly, if they chose so. These features allow collaborators to have discussions about datasets of interest within the system and be alerted when relevant data is updated.

The different configurations of the data model allow for each community to use spaces, datasets and collections to organize their data as they see fit. For example, the Intensive Managed Landscapes Critical Zone Observatory (IML-CZO) [2] uses spaces to represent different categories of data collected at physical sensor locations including: atmospheric, soil, surface water, among others. Inside the atmospheric space, collections for each type of atmospheric data are available: temperature, climate, and gas fluxes. Each dataset is named after the location where the data was collected, for example, Allerton. Within a dataset, there are two folders: raw, and processed, where users store data collected directly from instruments, or data processed which has been through quality control, outliers or erroneous data was removed, or data was converted to a different unit.

Figure 1: Clowder Data Model.

4 FLEXIBLE METADATA REPRESENTATION

Clowder distinguishes between required metadata on its resources, such as time of creation, author, license, etc. and optional generic metadata. The former is required and managed by the system for every resource (collection, dataset, file). The latter is optional and consists of generic statements using community-specific ontologies made by agents on a resource at a particular time. Two types of agents are currently supported: users and extractors. Users of the system can manually add metadata from the web interface by selecting from a predefined list of definitions. Extractors are independent and trusted processes running outside of the main Clowder application. More information about extractors is presented in the next section. Agents are defined as PROV agents [5]. In the future we hope to extend support of the PROV standard to include entities and activities.

JSON Linked Data (JSON-LD) [3] is used to encode optional metadata. The flexibility of JSON coupled with the rigor of Linked Data and Semantic Web technologies allows the application to accept structured metadata in a format that is portable and generic. Figures 2.a and 2.b provide examples of user specified and extractor defined metadata. Figure 2.a shows the addition of a CSDMS [27] standard vocabulary entry by a user. The content body contains a key value pair with a label *CSDMS Standard Name* and a value of *atmosphere_air_increment_of_pressure*. The *@context* subdocument specifies a unique URI for the label according to the JSON-LD rules. The agent is defined as of type *cat:user* (the namespace is defined in the linked context definition), with a *name* and a *user_id* in the form of URL to uniquely identify the user that can be resolved to a linked document with more information about the user.

Figure 2.b shows a extractor created metadata entry with a different type of user, in this case a *cat:extractor* and an *extractor_id* that points to a different type of URL resolving to more information about that extractor, such as version, authors, source code location, Docker Hub image id, software libraries used, bibliography or relevant papers. In this case the *content* subdocument contains a different key value pair also uniquely defined in the *@context* subdocument. While in the first case the option of adding a CSDMS Standard Name was registered with the system as a user metadata definition and available to the user to pick from, in the second case only the extractor knew ahead of time what kind of information it

Figure 2: Metadata encoded as JSON-LD.

was going to attach to the specific resource. The ability to provide a context at run time means that any standard vocabulary and ontology can be submitted to the system without any prior knowledge by the system. Extractors are not limited in the type of metadata that they can add. This greatly simplify the task of extending the system with community specific software and ontologies.

The list of metadata definitions a user can pick from is managed at the Clowder instance level by an administrator and at a space level by space administrators. Default metadata definitions include a subset of Dublin Core metadata definitions [32] like abstract, alternative title, reference, among others. Each metadata definition is defined by a label, description, URI identifying the term, and type. Currently available user metadata types are string, list, date and time, scientific variable, location and a option URL listing all entries for the list type.

By default, the system stores each document in a MongoDB instance. This simplifies development of new features and enables scalability in ways not usually available with open source RDF triple stores. All metadata is included when downloading a dataset using the BagIt [1] standard, a widely used system for archiving and storing digital information. This information is also available through an OAI-ORE [4] endpoint.

4.1 Search

Clowder supports broad search capabilities that look across collections, datasets and filenames across both user-specified and extractor-specified metadata and returns results grouped by resource type. Accessible via a search box in the menu, this search leverages Elasticsearch fuzzy matching to order results with nearest matches first. Clowder also indexes author, sibling resource relationships, descriptions, tags, comments and metadata. Users can browse ordered and weighted lists of tags to locate similar groups of resources quickly.

In addition to this basic search, Clowder's Advanced Search interface supports more detailed searching over metadata. Users can define one or more terms using several comparison operators with and/or grouping rules and an autocomplete system will suggest extant metadata fields grouped by source (i.e. a specific extractor or user) to reduce ambiguity in cases where multiple metadata objects have identically-named fields within different nesting structures.

Advanced Search is also driven by Elasticsearch, which indexes every incoming metadata field. The first time a new field path is encountered in new metadata - typically `extractorname.metadata.path.field` in the case of nested JSON objects - Elasticsearch will attempt to determine the field's data type and create a mapping. This means that all subsequent metadata using that same path and field name must adhere to the original data type or be rejected. Elasticsearch is highly scalable with one Clowder deployment containing an index of over 10 million documents.

5 AUTOMATIC METADATA EXTRACTION

Clowder provides an extraction bus to automatically extract information from raw files and support executing analytics on the data that does not require user input. In many cases, a community will have software that needs to be executed as a pre-processing step when data is ingested into the system. This automatic curation complements manual curation and is extremely important in cases where the size of the data is large or the data streams into the system from instruments and automated scripts.

The extraction bus is implemented using independent metadata extractors running as external processes and communicating back to the Clowder instance over an external publish/subscribe messaging bus, in our case RabbitMQ, and the Clowder web service API. Messages are sent asynchronously to the message broker. When certain events occur within the system, for example a file is uploaded or moved from one dataset to another, a message is sent out to the broker using the Advanced Message Queuing Protocol (AMQP). The message includes the id of the modified resource, the type of event, and any extra information specific to that even type. This information is encoded as JSON and provided as the payload of the AMQP message. Based on the event type, routing information is added to the message that will direct the message broker to deliver the message to the appropriate queues. For example in case of a file that is uploaded, the message can be routed based on the mimetype of the file. Extractors register themselves with the RabbitMQ at startup on an specific queue. When a message is delivered to an extractor, the extractor executes the predefined custom code. Depending on the task, the extractor will query the Clowder web service API, just like any other external client of the system would, to retrieve required information, for example download the raw bytes of a file on which it needs to execute. The extractor can

implement an algorithm directly within its source code, using any library available, launch any executable, or communicate with existing services, such as HPC batch queues. The results of extractions are communicated back to Clowder using the standard Clowder web service API. Running extractors as independent processes also simplifies the process of creating new ones. RabbitMQ supports many programming languages through its client libraries making it easy for the community to extend the framework.

The RabbitMQ message broker allows us to separate extractors from the core and provides a buffer between the two. This buffer, or queue, prevents loss of messages in case the extractor is not available. Once the extractor is back online it can continue to process the messages from the queue. The message broker will persist the message queue to disk in case of failures so that no messages will be lost. By using a publish-subscribe bus, we can make sure that the system can scale under high loads. When deploying a Clowder instance, only those extractors that the community is interested in are deployed, allowing the community to pick and choose what type of information is appropriate for that domain. By leveraging RabbitMQ direct exchanges, different Clowder instances can also share extractors to minimize the number of total extractors deployed in case of multiple Clowder deployments running on shared resources.

To support different workloads for each extractor type at runtime, the Brown Dog project [30] has implemented a way to monitor these queues and scale the number of running extractor up and down as needed. Each extractor type is deployed as a Docker Swarm service and the replication factor is modified based on queue size and extractor specific configuration parameters. Docker hosts in the swarm are also added and removed at runtime based on overall traffic. This effort is also simplified by the fact that all extractor are dockerized and published to Docker hub.

5.1 PyClowder

To simplify the development of new extractors, we have created PyClowder⁵, a Python package that wraps many useful Clowder API functions. The package was initially developed to provide logic for registering extractors with a Clowder instance, communicating with RabbitMQ and fetching new event messages for processing based on message parameters. By default it also includes automated retrying of failed extraction events and easily configurable integration with centralized logging systems. Over time PyClowder has been expanded to include a variety of CRUD operations in the different Clowder resource types. Ongoing development will simplify writing new extractors using PyClowder by providing ways of executing simple Python functions as extractors.

5.2 Example Extractors

A wide variety of extractors have been written to support specific use cases. Many are available in the Clowder NCSA Bitbucket project⁶. The Brown Dog project keeps a list of known extractors in a wiki page⁷. Some examples include a Tree Delineation Extractor

to identify tree canopy polygons from LiDAR data, a Route Greenness Extractor to calculate the greenest route between a source and a destination, and a OCR Extractor which uses the open source Tesseract software to extract printed text from images. A project to study the immunoregulatory properties of Mesenchymal stem cells (MSC) during islet cell transplants [36] leveraged Clowder to provide a web front end for researchers to easily upload Excel files containing time series data from lab analysis and extractors to extract new data from these Excel spreadsheets and load it in a custom time series database for custom visualizations and machine learning.

Extractors can also be orchestrated to create more complex workflows. For example, in the TERRA-REF project [8], images from various cameras scanning fields are stitched together every day after a full scan is complete. This requires coordination between several extractors in sequence: the first will convert raw data to common image formats. Next, a coordinating extractor will collate lists of images from potentially many parallel converters and assemble lists by date until sufficient coverage exists of the study area for that date. The coordinator then triggers a stitching extractor to mosaic together up to 11,000 images per day into one large image product for uploading into Clowder. Finally these mosaicked images are passed along to analysis extractors that extract traits and characteristics from the imagery that would not be possible at the individual image level. The Clowder architecture allows TERRA-REF to deploy many simultaneous extractors for each part of this process and still orchestrate combined outputs while maintaining speed and scalability.

For compute intensive tasks and to leverage HPC environments to process data, Clowder provides an HPC extractor⁸. This is a generic extractor, which based on configurations settings, listens for specific files (based on the file type) being uploaded to Clowder. Once it receives a message from RabbitMQ, instead of processing the file on its own, it creates and submits a batch queue job in an HPC resource. In the HPC environment, once the job gets started, the file will be processed by the original extractor that was developed for processing such file types. This original extractor generates metadata after processing the file, which is then sent back to Clowder using its RESTful APIs. It also writes the status messages to a log file, which is read by the HPC extractor remotely and sent to Clowder. For this to work, the original extractor must be installed within the HPC environment. The generic HPC extractor can be set up in any environment and can be configured to remotely communicate with an HPC machine and create jobs in its batch processing queue. We have used this extractor to process video files on SDSC Gordon Supercomputer as part of the Video Analytics Tableau [17] and Brown Dog project [26].

6 DATA VISUALIZATIONS

Clowder supports the need for custom data visualizations and previewing large data files by allowing developers to package and deploy JavaScript-based data previewers within Clowder instances. A data previewer renders a pre-computed data preview or generates a custom visualization on the fly based on available metadata and

⁵<https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/pyclowder2/>

⁶<https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/>

⁷<https://opensource.ncsa.illinois.edu/confluence/x/YGFBQ>

⁸<https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/extractors-hpc/browse/python>

raw bytes. A data preview, in many cases, is just a reduced size version of the input data resource that helps users to get a sneak peek of the actual raw data (e.g. a lower resolution version of a movie video). In some cases, it can be a data resource converted to different formats for rendering inside the web browser. In some other cases a preview can be a zero-byte placeholder file to trigger a data previewer that renders a complex visualization.

For example, OGG format videos are not supported by some browsers. To ensure that a video file in that format gets displayed across all browsers, a video preview extractor processes this file and generates a lower resolution MP4 video file and attaches it to the original file as one of its data previews. Now, a video data previewer whose supported file formats include MP4 can render this video preview. Thus a user is now able to play a video file that otherwise would not have been playable on the browser. Previews, thumbnails, and visualizations generated by video previewers were extensively used by the Video Analysis Tableau project. The project used Clowder as a platform to study videos. The video analysis extractors developed as part of the project were contributed back into Clowder [17].

Figure 3: Visualization generated by the person tracking previewer showing people being tracked across different video frames.

More complex data previewers can generate and render visualizations using the original input file and metadata generated by

an extractor. The person tracking extractor⁹ developed using the discriminatively trained deformable part models [12, 14] and continuous energy minimization [21] processes a video file and generates metadata that includes locations of different people identified in each video frame. Along with this metadata, a zero-byte data preview in custom file format is uploaded by this extractor. The person tracking data previewer using its JSON format manifest file to listen to this custom file format and when it sees such a data preview attached to a video file, it parses the JSON-LD metadata associated with that file and obtains the person tracking metadata. This metadata is processed to generate a person tracking visualization that contains colored bounding boxes around people being tracked and a colored bar graph that shows when those people appear in the video (Fig. 3).

A data previewer includes a JSON format manifest file that contains metadata about itself and one or more JavaScript files that contain the logic for rendering a data preview or visualization. This also includes the JavaScript libraries used by this data previewer, if any. The manifest file contains the name of the previewer, path to its main JS file, and the input data MIME types for which this previewer will be run by Clowder. When a resource page (file, dataset, or collection) is rendered, Clowder uses the manifest file to select which data previewers should be executed for that resource. Each resulting preview is rendered in its own tab.

6.1 Geospatial Previewers and Extractors

As an example of how extractors, previewers and external services can work together to extend Clowder for the needs of a specific use case, we describe how Clowder supports visualizing geospatial data. This implementation includes a GeoTiff extractor, a shapefile extractor, a geospatial previewer and an instance Geoserver. If enabled, when a user uploads a GeoTiff or a zip file containing a shapefile, the system provides an interactive map on the dataset or file page that visualizes the geospatial raster or vector based information.

Figure 4 shows how when an ESRI shapefile is uploaded to Clowder, a geospatial extractor registers the file with a Geoserver. The URL of the generated WMS layer is then added by the extractor as generic metadata back to Clowder. If generic metadata with the required keys (for example “WMS Layer URL”) is available on the file, the geospatial previewer is launched and an OpenLayers-based map is rendered together with the WMS layer. It is worth noting that the previewer and extractor in this case work together but are also independent. The metadata can be added using a different extractor or client and the metadata can be used by a different previewer or client. In the future, other OGC services such as Web Feature Service (WFS), and Catalog Service for the Web (CSW) can be leveraged by Clowder using the Geoserver and a similar setup.

7 ARCHITECTURE AND IMPLEMENTATION

The Clowder framework follows a typical web application layered architecture with an HTML/Javascript web front end and a JSON RESTful web service API at the top (Fig. 5). The former is used by end users to manually interact with the system, while the latter is

⁹<https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/extractors-person-tracking/>

Figure 4: Geospatial support using metadata extractors and data previewers.

used by external clients for machine-to-machine interaction. Almost all functionality available in the web GUI is available from the web API and vice versa. This allows projects to write rich clients that are tailored to the needs of a specific use case. Supported authentication methods include a local login/password option, OAuth2 providers (Google, Twitter, Facebook), ORCID, CILogon, Atlassian Crowd and LDAP.

The main web applications is written using the Play Framework and Scala programming language. Both were selected for their performance and scalability. The Play Framework was selected because of its lightweight and stateless approach. Scala was chosen over Java because of its combination of object-oriented and functional programming, static typing, being less verbose and the ability to leverage Java and other JVM based libraries. The Play Framework follows the model-view-controller pattern. Models are shared between web GUI and web API. Even though controllers for each are kept separate, Clowder includes an extra layer of service abstraction below the controllers. Services are Scala traits for which one or more implementations are provided using dependency injection. This means that the code in the controllers is not tightly bound to the code used to communicate with the back end services. For example, the *ByteStorageService* trait abstracts storing and retrieving file bytes. The controllers only know that when storing bytes an *InputStream* needs to be provided. Three implementations are currently available of the *ByteStorageService* to store files on the file system, in MongoDB GridFS or in IRODS. Administrators can specify which implementation to use at runtime through the use of various configuration files. This service abstraction provides flexibility in extending the system to use new infrastructure without having to make drastic changes to the core of the system.

By default, the system stores all information in MongoDB, with metadata indexed in Elasticsearch and the raw files stored on the file system. MongoDB and Elasticsearch were selected for their ability to horizontally scale and their wide adoption and support. Their native support of JSON matches the needs of the web interfaces and services. RabbitMQ is used to implement the metadata extraction bus because of its proven scalability and reliability, the use of the open AMQP specification and the extensive list of client libraries. Thanks to the service abstraction any of these technologies could be replaced in the future. Multiple projects [8, 30] have tested and improved Clowder’s scalability and performance under heavy loads and over large collections.

Some features can be turned on and off through the use of plugins. These are very specific features that not all use cases might need. For

Figure 5: Clowder Architecture.

example the SEAD project developed a staging area for publishing datasets to long-term preservation archives that requires special external web services and permission to communicate to a variety of archives. If turned on and properly configured, the web GUI will provide interfaces to publish datasets and the web API will have endpoints available for external archives to query required information in Clowder throughout the publishing process. Several projects [2, 15] have developed a Geostreaming API as a plugin to Clowder to manage geo-temporal data streams in a custom Postgres database with custom geodashboards as clients. These plugins and others can be turned on and off using configuration files.

8 CONCLUSIONS

Clowder is a data management system designed to support the long tail of data. Projects in different domains have been able to leverage its flexible core, rich data model, robust metadata support, scalable extraction bus, generic web-based visualizations to support a wide variety of use cases. In turn, many of these projects have contributed back to the project and extend the system in novel ways. This interaction between project specific requirements and generic functionality has not been easy to manage. We have been developing an open source governance model which includes a contributing workflow, code of conduct, mailing list and chat channels, but realized early on that nurturing an open source community is a challenge in and of itself. We will continue to do so as we believe that for a data management system to truly be able to support the long tail of data it must include a healthy open source community that goes beyond current funding.

Interacting with the community and having external developers contribute to the framework has also taught us the importance of supporting materials. Libraries like PyClowder, to simplify the task of writing extractors without having to learn all the details of how the system works, were not in the original design but they ended up being extremely helpful. We are currently working on extending the documentation for both end users and software developers. We have begun refactoring the core to erase some of the technical debt accrued over the years. Several software libraries used have new versions that are not backward compatible or are not maintained anymore. Many internal features that were originally developed for specific use cases have not been properly maintained or have

been replaced by better solutions. This results in a code base that is more difficult to maintain than it needs to be.

ACKNOWLEDGMENTS

This research has been funded by grants: NSF #BCS-0941268, #EAR-331906, #ACI-1261582, #ACI-1443013, #OCI-0940824, #OCI-0525308; NIH #1P01AI089556-01A1. Illinois - Indiana Sea Grant #DW92329201, European Commission #RI-261600, XSEDE #OCI-1053575, ARPA-E #DE-AR0000594. We are grateful to all contributors to Clowder (<https://github.com/ncsa/clowder/blob/master/CONTRIBUTORS.md>).

REFERENCES

- [1] 2018. The BagIt File Packaging Format. (2018). Retrieved June 1, 2018 from <https://tools.ietf.org/html/draft-kunze-bagit-14>
- [2] 2018. IML-CZO Data Management - Critical Zone Observatory Network for Intensively Managed Landscapes. (2018). Retrieved June 1, 2018 from <http://imlczo.ncsa.illinois.edu/>
- [3] 2018. JSON-LD - JSON for Linking Data. (2018). Retrieved June 1, 2018 from <https://json-ld.org/>
- [4] 2018. ORE Specifications and User Guides - Table of Contents. (2018). Retrieved June 1, 2018 from <http://www.openarchives.org/ore/1.0/toc>
- [5] 2018. PROV Model Primer. (2018). Retrieved June 1, 2018 from <https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/#agents-and-responsibility>
- [6] Ricardo Carvalho Amorim, João Aguiar Castro, João Rocha da Silva, and Cristina Ribeiro. 2017. A comparison of research data management platforms: architecture, flexible metadata and interoperability. *Universal Access in the Information Society* 16, 4 (01 Nov 2017), 851–862. <https://doi.org/10.1007/s10209-016-0475-y>
- [7] S. B. Ardestani, C. J. Håkansson, E. Laure, I. Livenson, P. Stranák, E. Dima, D. Blommesteijn, and M. v. d. Sanden. 2015. B2SHARE: An Open eScience Data Sharing Platform. In *2015 IEEE 11th International Conference on e-Science*. 448–453. <https://doi.org/10.1109/eScience.2015.44>
- [8] Maxwell Burnette, Rob Kooper, J.D. Maloney, Gareth S. Rohde, Jeffrey A. Terstriep, Craig Willis, Noah Falhgren, Todd Mockler, Maria Newcomb, Vasit Sagan, Pedro Andrade-Sanchez, Nadia Shakoor, Paheding Sidike, Rick Ward, and David LeBauer. 2018. TERRA-REF Data Processing Infrastructure for Plant Sensing. In *PEARC'18: Practice and Experience in Advanced Research Computing*. ACM, New York, NY, USA. <https://doi.org/10.1145/3219104.3219152> To appear.
- [9] Charles E. Catlett. 2005. TeraGrid: A Foundation for US Cyberinfrastructure. In *Network and Parallel Computing*, Hai Jin, Daniel Reed, and Wenbin Jiang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–1.
- [10] Amit Chourasia, Mona Wong, Dmitry Mishin, David R. Nadeau, and Michael Norman. 2016. SeedMe: A Scientific Data Sharing and Collaboration Platform. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16)*. ACM, New York, NY, USA, Article 48, 6 pages. <https://doi.org/10.1145/2949550.2949590>
- [11] Mostafa M. Elag, Praveen Kumar, Luigi Marini, James D. Myers, Margaret Hedstrom, and Beth A. Plale. 2017. Identification and characterization of information-networks in long-tail data collections. *Environmental Modelling & Software* 94 (2017), 100 – 111. <https://doi.org/10.1016/j.envsoft.2017.03.032>
- [12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. 2010. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 9 (2010), 1627–1645.
- [13] National Science Foundation. 2014. Chapter II - Proposal Preparation Instructions. (Dec 2014). https://www.nsf.gov/pubs/policydocs/pappguide/nsf15001/gpg_2.jsp#IIC2j
- [14] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. 2012. Discriminatively Trained Deformable Part Models, Release 5. <http://people.cs.uchicago.edu/~rbg/latent-release5/>. (09 2012).
- [15] I. Gutierrez-Polo, Y. Zhao, S. Bradley, E. Roeder, M. Pitcel, K. TePas, P. Collingsworth, and L. Marini. 2017. Monitoring Water Quality in the Great Lakes Leveraging Geo-Temporal Cyberinfrastructure. In *2017 IEEE 13th International Conference on e-Science (e-Science)*. 364–373. <https://doi.org/10.1109/eScience.2017.50>
- [16] P Bryan Heidorn. 2008. Shedding light on the dark data in the long tail of science. *Library trends* 57, 2 (2008), 280–299.
- [17] Virginia Kuhn, Alan Craig, Michael Simeone, Sandeep Puthanveetil Satheesan, and Luigi Marini. 2015. The VAT: Enhanced Video Analysis. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure (XSEDE '15)*. ACM, New York, NY, USA, Article 11, 4 pages. <https://doi.org/10.1145/2792745.2792756>
- [18] Alison Langmead, Paul Rodriguez, Sandeep Puthanveetil Satheesan, and Alan Craig. 2017. Extracting Meaningful Data from Decomposing Bodies. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact (PEARC17)*. ACM, New York, NY, USA, Article 41, 8 pages. <https://doi.org/10.1145/3093338.3093368>
- [19] John A Lewis. 2014. Research data management technical infrastructure: A review of options for development at the University of Sheffield. (2014).
- [20] William K. Michener, Suzie Allard, Amber Budden, Robert B. Cook, Kimberly Douglass, Mike Frame, Steve Kelling, Rebecca Koskela, Carol Tenopir, and David A. Vieglais. 2012. Participatory design of DataONE - Enabling cyberinfrastructure for the biological and environmental sciences. *Ecological Informatics* 11 (2012), 5 – 15. <https://doi.org/10.1016/j.ecoinf.2011.08.007> Data platforms in integrative biodiversity research.
- [21] A. Milan, S. Roth, and K. Schindler. 2014. Continuous Energy Minimization for Multitarget Tracking. *IEEE TPAMI* 36, 1 (2014), 58–72. <https://doi.org/10.1109/TPAMI.2013.103>
- [22] Richard L Moore, Chaitan Baru, Diane Baxter, Geoffrey C Fox, Amit Majumdar, Phillip Papadopoulos, Wayne Pfeiffer, Robert S Sinkovits, Shawn Strande, Mahidhar Tatineni, et al. 2014. Gateways to discovery: Cyberinfrastructure for the long tail of science. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM, 39.
- [23] J. Myers, M. Hedstrom, D. Akmon, S. Payette, B. A. Plale, I. Kouper, S. McCaulay, R. McDonald, I. Suriarachchi, A. Varadaraju, P. Kumar, M. Elag, J. Lee, R. Kooper, and L. Marini. 2015. Towards Sustainable Curation and Preservation: The SEAD Project's Data Services Approach. In *2015 IEEE 11th International Conference on e-Science*. 485–494. <https://doi.org/10.1109/eScience.2015.56>
- [24] P. Nguyen, S. Konstanty, T. Nicholson, T. O'Brien, A. Schwartz-Duval, T. Spila, K. Nahrstedt, R. H. Campbell, I. Gupta, M. Chan, K. Mchenry, and N. Paquin. 2017. 4CeeD: Real-Time Data Acquisition and Analysis Framework for Material-Related Cyber-Physical Environments. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 11–20. <https://doi.org/10.1109/CCGRID.2017.51>
- [25] National Institutes of Health. 2003. NIH Data Sharing Policy and Implementation Guidance. (Mar 2003). Retrieved March 13, 2018 from https://grants.nih.gov/grants/policy/data_sharing/data_sharing_guidance.htm
- [26] Smruti Padhy, Jay Alameda, Rob Kooper, Rui Liu, Sandeep Puthanveetil Satheesan, Inna Zharnitsky, Gregory Jansen, Michael C. Dietze, Praveen Kumar, Jong Lee, Richard Marciano, Luigi Marini, Barbara Minsker, Chris Navarro, Marcus Slavenas, William Sullivan, and Kenton McHenry. 2016. An Architecture for Automatic Deployment of Brown Dog Services at Scale into Diverse Computing Infrastructures. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16)*. ACM, New York, NY, USA, Article 33, 8 pages. <https://doi.org/10.1145/2949550.2949647>
- [27] Scott D Peckham. 2014. The CSDMS standard names: Cross-domain naming conventions for describing process models, data sets and their associated variables. (2014).
- [28] Marshall Scott Poole, Natalie Lambert, Sandeep Puthanveetil Satheesan, Amit Das, Alex Yahja, and Mark Hasegawa-Johnson. 2015. GroupScope: A Framework and Tools for Large Scale Study of Social Processes. (2015).
- [29] Paul Rodriguez, Sandeep Puthanveetil, Jeffrey Will, Elizabeth Wuelfel, and Alan Craig. 2017. Extracting, Assimilating, and Sharing the Results of Image Analysis on the FSA/OWI Photography Collection. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact (PEARC17)*. ACM, New York, NY, USA, Article 42, 6 pages. <https://doi.org/10.1145/3093338.3093365>
- [30] Sandeep Puthanveetil Satheesan, Jay Alameda, Shannon Bradley, Michael Dietze, Benjamin Galewsky, Gregory Jansen, Rob Kooper, Praveen Kumar, Jong Lee, Richard Marciano, Luigi Marini, Christopher M. Navarro Barbara S. Minsker, Arthur Schmidt, Marcus Slavenas, William C. Sullivan, Bing Zhang, Yan Zhao, Inna Zharnitsky, and Kenton McHenry. 2018. Brown Dog: Making the Digital World a Better Place, a Few Files at a Time. In *PEARC'18: Practice and Experience in Advanced Research Computing*. ACM, New York, NY, USA. <https://doi.org/10.1145/3219104.3219132> To appear.
- [31] Constantinos Sophocleous, Luigi Marini, Ropertos Georgiou, Mohammed El-faragy, and Kenton McHenry. 2017. Medici 2: A scalable content management system for cultural heritage datasets. *Code4Lib Journal* (2017).
- [32] Stuart Weibel. 1997. The Dublin Core: A Simple Content Description Model for Electronic Resources. *Bulletin of the American Society for Information Science and Technology* 24, 1 (1997), 9–11. <https://doi.org/10.1002/bult.70>
- [33] Nancy Wilkins-Diehr, Dennis Gannon, Gerhard Klimeck, Scott Oster, and Sudhakar Pamidighantam. 2008. TeraGrid Science Gateways and Their Impact on Science. *Computer* 41, 11 (Nov 2008), 32–41. <https://doi.org/10.1109/MC.2008.470>
- [34] Joss Winn et al. 2013. Open data and the academy: an evaluation of CKAN for research data management. (2013).
- [35] B. Zhang, L. C. Pouchard, P. M. Smith, A. Gasc, and B. C. Pijanowski. 2016. Data storage and sharing for the long tail of science. In *2016 New York Scientific Data Summit (NYSDS)*. 1–9. <https://doi.org/10.1109/NYSDS.2016.7747811>
- [36] Y. Zhao, E. F. Black, L. Marini, K. McHenry, N. Kenyon, R. Patil, A. Balla, and A. Bartholomew. 2016. Automatic glomerulus extraction in whole slide images towards computer aided diagnosis. In *2016 IEEE 12th International Conference on e-Science (e-Science)*. 165–174. <https://doi.org/10.1109/eScience.2016.7870897>