Cloud Dataverse Team

Ang Li
Benjamin Corn
Sneha Pradhan
Will Norman

Cloud Dataverse Project Proposal

Dataverse is an open-source web application allowing individuals to explore, share, cite, and analyze research data. The platform promotes fair use practices within the community, allowing for replication of prior research and the publishers of the research data to receive visibility and credit for their work. The Cloud Dataverse project is an extension of the Dataverse project, currently at Harvard, enabling users to compute over data sets found within a Dataverse on the cloud through big data frameworks such as Hadoop and Spark.

1. Vision and Goals Of The Project:

The main objective of the project is to create an integration between the current Dataverse project and the MOC such that users of the Dataverse project can access target data and run compute jobs via the cloud.

Key goals of the project include :

- Providing users of the Dataverse project with a simple way to interact with the MOC through a Dataverse instance with cloud integration.
- Allowing users to compute over datasets using Hadoop or Spark through Sahara as the current implementation requires users to compute locally or manually setup compute clusters.
- Harness the prior integration of Swift Object Store during the provisioning of compute environments.
- Working alongside the teams of related projects in the MOC to achieve an efficient workflow model and shipping high quality code.

Users/Personas Of The Project:
Customer: Massachusetts Open Cloud(MOC), Dataverse Team (Harvard)
Users: Data Consumers, Data Publishers, System Administrators
System administrators will be able to manage the Dataverse settings such as the Sahara endpoints and control permissions via Swift. Data consumers will be able to launch a compute environment via the Cloud Dataverse UI to complete computations for a selected dataset.

2. Scope and Features Of The Project:

The current implementation of the Dataverse application requires the user to download a dataset for local computing. These datasets, in many cases, are large in scale, often times requiring large storage and compute requirements. The scope of the Cloud Dataverse project is to solve this problem by delivering a streamlined user experience for setting up a compute environment directly from Dataverse through a simple process flow.
By provisioning compute clusters in Hadoop or Spark through Sahara on demand, users are able to compute on the cloud, rather than on their local machines. Our integration includes authenticating users with OpenStack through Keystone, ensuring file permissions are handled appropriately with Swift, and providing users with a compute environment to fit their computation needs.

This project does not include any further work on the integration of Swift with the Dataverse platform, nor does it continue the development of the harvesting project.

3. Solution Concept

Global Architectural Structure of the Project:

The main components of our system includes:

- OpenStack Sahara: A cloud service supporting a variety of data processing frameworks such as Hadoop, Spark or Storm. Sahara communicates with other OpenStack services such as Nova and Swift to deploy the specified cluster with essential configurations for data analytic jobs.
- OpenStack Swift: Offers cloud storage service with simple APIs. It is built for scale and designed to be a highly available, durable data storage service. OpenStack Sahara launches data analytics jobs such as MapReduce using OpenStack Nova and Swift.
- OpenStack Nova: Compute service by OpenStack for hosting and managing cloud computing systems.
- OpenStack Keystone: Authentication service provided by the OpenStack community that helps manage user permissions and authorization.
- MOC: Public Cloud where this project's instance of Dataverse will run,

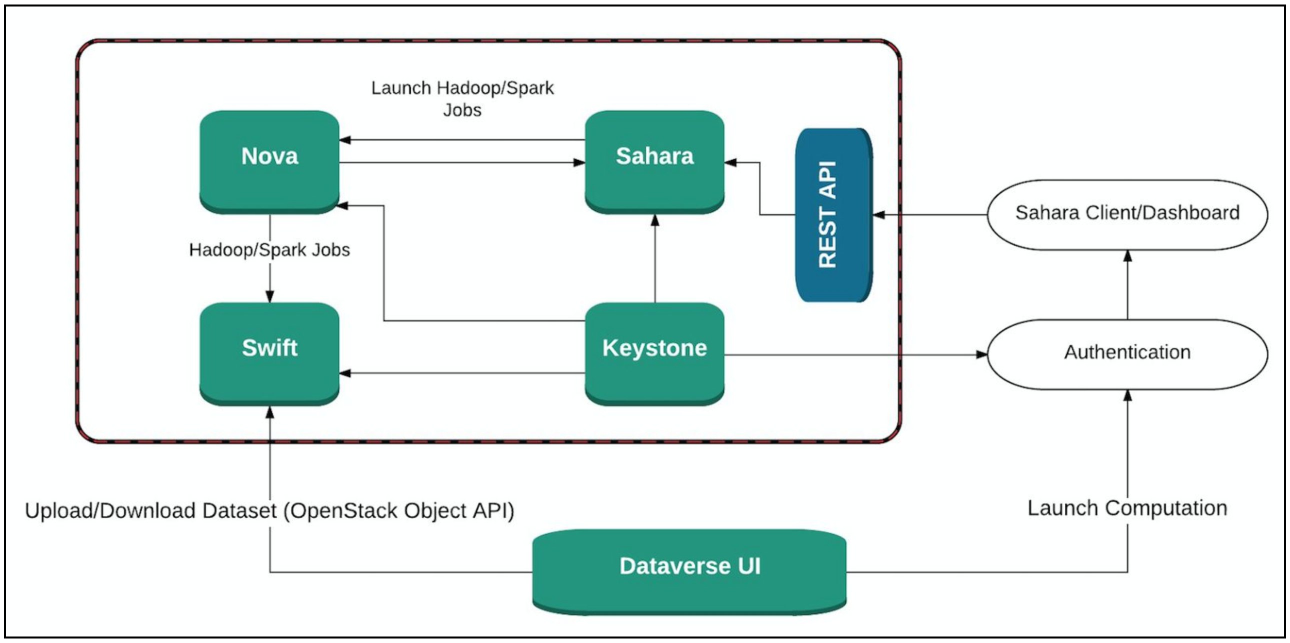

Figure 1

The above figure presents the Cloud Dataverse global architectural structure. The Dataverse UI is the link or integration that our project will be working on. There are two use cases associated with the Dataverse UI. The first use case allows users to upload or download the Datasets through the Dataverse UI through traditional local downloads. The Datasets are stored or retrieved from Swift. The second use case allows users to launch compute environments through the Dataverse UI. To do so the system provides users with the endpoint of MOC where the user must authenticate and then get access to the OpenStack Sahara client or the OpenStack dashboard and conduct the data analytic computation using the clusters provisioned through Sahara.

Design Implications and Discussion:

- Storage of the datasets: The storage of the datasets as well as metadata in OpenStack Swift have already been done as one of the projects by the Dataverse Team. However, we need to link this storage system to OpenStack Nova so the users can have access to the desired data. This is where Sahara comes into play.
- Provisioning compute clusters: The direct way of provisioning Hadoop/Spark clusters is through OpenStack Sahara. Sahara utilizes templates for provisioning compute clusters and these options must be made available to the users through the Cloud Dataverse UI. Configuration parameters ought to be presented in a simple manner while allowing the user to enable advanced configuration options if necessary.
- Keystone: Keystone provides API client authentication for OpenStack. It is essential for accessing OpenStack components such as Sahara and Swift. The effectiveness of

authentication from the Dataverse credentials to Keystone is important to user experience and security. Details of the Keystone APIs are to be carefully studied in one of the coming sprints.

4. <u>Acceptance criteria</u>

Minimum acceptance criteria for Cloud Dataverse includes providing a running instance of Dataverse on the MOC, which is populated with data sets, a streamlined user interface with the ability to spin up a compute cluster from a data set page through OpenStack Sahara, and creating a cluster on either Spark or Hadoop. The user experience should be clean, simple, and effective; much like Amazon Public Datasets or Amazon Elastic MapReduce. Administrative options should be included to manage the endpoints for the Sahara Service. Lastly, proper authentication protocols must be established between Dataverse and the OpenStack environment when a user proceeds with launching a compute environment.

5. <u>Release Planning:</u>

Our trello link will be: https://trello.com/b/kEB7i2E2/cloud-dataverse

<u>Sprint 1 (Feb 5th - Feb 12th):</u>

- Install the Dataverse project in an instance in MOC. This will allow us to understand the backend we will be working with.
- Understand the various components and technical aspects of the projects such as OpenStack Swift, Sahara, and Nova.
- Become familiar with the work done on integrating Swift and the harvesting functionality

Sprint 2 (Feb 13th - Feb 23rd):

- Start on the UI for the different types of users such as the consumers as well as the publishers.
- Work on the authentication protocol portion of the project through OpenStack Keystone.

Release # 1 (Feb 23rd):

- As the demo will be on the 23rd of February, this will be our Minimal Viable Product (MVP). We plan to have the basic UI that will integrate Dataverse with OpenStack. We will have the authentication for the user login.

Sprint 3: (Feb 24th - March 30th):

- Work with connecting UI to Sahara for provisioning Hadoop/Spark clusters
- Development of administrative UI in Dataverse for Sahara / Compute related parameters

Release # 2 (March 30th):

- This release will include the compute feature that will allow the users to spin up Hadoop or Spark cluster jobs provisioned by OpenStack Sahara.

Sprint 5: (March 30 - April 27th)

- Polish the UI for a sleek, smooth and user-friendly experience
- Work on the back-end with Sahara and Spark so that everything works efficiently.
- Check for bugs and clean up the work

Release # 3 (April 28th):

- Final release of the end product.