Design Patterns in Web Gateways for Scientific Data

Shreyas Cholia
Data and Analytics Services, NERSC

Joint Facilities User Forum on Data-Intensive Computing, Oakland, June 17, 2014
Introduction
Overview

• Motivation

• Patterns
  – Web Data Sharing
  – End-to-End Frameworks
  – API-driven access
  – Web IDEs
  – Federated Identity
  – Social Data

• Conclusions
Motivation for Science Gateways

• People expect web interfaces and applications for usability

X don’t want generic options/tools not applicable to your science
X don’t want to deal with backend, middleware, UNIX CLI etc.
Harnessing The Power Of The Web

• Web interfaces enable science-centric views to data
• Enable new discoveries and insights through collaborative tools and rich interfaces

https://openmsi.nersc.gov

https://materialsproject.org
Science is Collaborative

- Science is now a collaborative effort
- Large teams of people
- Requires shared access to compute and data resources
Patterns in Science Gateways

Attempt to catalog the interesting features of science gateways that span science domains in order to gain some insight on how to best expose scientific data over the web.
Data Sharing Over The Web
Data Sharing Over The Web

• OSTP requirements around data management
• What is the basic infrastructure required to make this happen?
Data Sharing Over The Web

- Store data on persistent global filesystem
- Designate an area for export
- Mount export area RO on a web node using a shared FS
- Run a webserver on web node to share with world
- Add AuthNZ to web server configuration as needed
Scaling and Growth

• Scale out – add more nodes + loadbalancer
• Use sophisticated interfaces to search/browse data
  – HTML+JS, PHP, Web Frameworks
• High performance data transfer tools
  – Globus
  – Xrootd + webdav
  – Custom downloaders
End-to-End Frameworks
End-to-End Frameworks

- Open source frameworks that can be customized for specific science use cases:
  - HubZero
  - Galaxy
  - OGCE

- Takes a plugin + configuration based approach to domain specific customizations.

- Very useful for managing canned workflows.

- Community and management tools often baked-in
End-to-End Frameworks

• Plugin/Portlet approach to development requires developers to learn the vernacular of the framework.

• Applications tend to be built around framework constraints rather than science use-cases. Rich set of front-end technologies in HTML5 and JavaScript much harder to leverage in this context.

• If you want a blank-slate that starts with science use cases and builds up interface from here, you might try a different approach.
API Driven Access
API Driven Access

- Web APIs provide programmatic access to data and resources to developers over the web
- Access to data as well-defined objects allows users to develop their own custom applications and code
- Build web application around this API using the software stack of your choice
The 1-minute intro to REST APIs

- Use HTTP verbs in conjunction with URLs
  - GET, POST, PUT, DELETE
- Verb + URL + parameters = function call
- Return structured data
- For instance (from NEWT API):
  
  Structured Output (JSON):
  ```json
  {"status": "OK",
  "output":["status": "up", "system": "hopper"], {"status": "up", "system": "carver"}, "error": ""
  ```
• Enable heavy duty server side operations and long running tasks

• Perform sub-selection of data – pull down only what is needed in client and render it in the browser
  – eg. OpenDAP connects HDF5 datasets with web clients
eg. ALS Spot Suite

- All the client side logic resides in HTML5/JavaScript
- Make callouts to REST API for all server-side information
Aside - MVC Frameworks

• Increasingly, web frameworks are following Model-View-Controller pattern
  – eg. Django, Ruby-On-Rails, Backbone.js

• Enables separation of user interfaces (V) from backend data models (M) and scientific libraries for interacting with data (C)

• Provide you with just enough scaffolding to build out application without imposing front-end application constraints.

• Web APIs lend themselves nicely to this model
Web IDEs
Web IDEs

- Combine the expressiveness of command line tools with web GUIs
- RStudio, Mathematica, iPython Notebook
Authentication

• **Authentication:** verifying the identity of the user
  – Are you who you say you are?

• **Local authentication**
  – Identity tied to local user credentials from institution that hosts the gateway

• **Federated authentication**
  – Identity verification delegated to external party (such as user’s home institution, or verified 3rd party)
  – External party returns an assertion to the gateway with ID information.
Federated Identity

• One identity to rule them all
• Allow users to log-in using common credentials from their home institution / 3rd party provider
• Assumestrustrelationshipbetween science gateway and identity provider
• Map federated ID to local account
• Examples:
  – Globus Nexus (OAuth), CILogon (Shibboleth)
• Enables access across users from multiple institutions
Communities and Social Data

- Generate a discussion around science results
- Combine Science Gateways with social interaction tools
  - Disqus
  - Forums integrated and cross-linked with datasets
  - Social Media integration
- Referencing and sharing results and user generated datasets
- Users can upload datasets
KBase Narrative

- Allows you to reference data and run models from other users in system
Data Upload and Provenance

• Users upload data to portals
• But this introduces challenges wrt data management
  – Vetting/Validating the data
  – Managing Provenance
  – Data ownership and licensing
  – Public/Private/Protected data
The End

- Probably not a complete list
  - Intended to spur further discussion on other interesting patterns

- NERSC Science Gateways
  - [http://portal.nersc.gov](http://portal.nersc.gov)

- Contact:
  - Shreyas Cholia: scholia@lbl.gov

- Questions? Comments?
Authorization

• **Authorization: Access privileges for a given identity**
  – What is the user allowed to do?

• **Data portals see some combination of the following sharing modes:**
  – Public – data are publicly viewable to all users
  – Protected – data are only visible to members within a designated group or collaboration
  – Private – data are private to the individual user