Science Platforms: Interoperability and other Missing Pieces

Gregory Dubois-Felsmann
Caltech/IPAC-LSST

7 December 2018

Comments in the session

- Being able to transport a notebook and associated software from one to another, and use it on the different datasets available at different sites
- Seamlessly performing cross-mission queries
- Need for metadata and documentation for the platforms with which we're not familiar. Need to understand the quality and resolution of the data
- Seamless access to internationally distributed datasets; require information about the science quality of the data you are getting back
- Interoperable authentication - no need for multiple accounts
- Common batch job submission mechanism
- Need a lot of different datasets; avoid as much as possible the need for users to have to understand the meaning of the tables & columns from different datasets; interoperable data models (part of the traditional IVOA remit) -- metadata, UCD's

Comments in the session (2)

- Worried about imposing too many constraints on data services, archives, etc.; too much normalization limits flexibility (and discovery)
- Discussion of CAOM2, UCDs, etc.; standardization of observational metadata vs. catalogs
- Interoperability for science platforms should be seen to be growing out of the existing body of work in the IVOA, and not a wholly new, huge, VAO-like enterprise
- Container-level interoperability
- If I move a container to a different site, do the local data services at the destination speak the same language?
- Could users define a personal environment that could be replicated at multiple sites?
- Very important to communicate exactly what (processing?) was successful, what can be trusted, what work units were completed?

What Interoperability?

Clearly there are many flavors of interoperability that are relevant to Science Platforms

- Interoperability of data services over the network (classic IVOA topic)
- Interoperability of source code that accesses data
  - I want to use the same (or minimally modified) code to access (say) imaging survey data from many sources
  - I want to use the same (or minimally modified) code to access data whether it’s local or remote (orthogonalize user knowledge of how to code and user knowledge of where the data are)
- Interoperability of scientifically similar data models from multiple sources
- Interoperability of user code at the level of a “unit of analysis”
  - Notebooks
  - Containers
  - Software packages (3rd-party or user’s personal software)
- Interoperability of resource access (e.g., to parallel processing, batch systems)

What Interoperability?

Clearly there are many flavors of interoperability that are relevant to Science Platforms

- Interoperability of data services over the network (classic IVOA topic)
- Interoperability of source code that accesses data
  - I want to use the same (or minimally modified) code to access (say) imaging survey data from many sources
  - I want to use the same (or minimally modified) code to access data whether it's local or remote (orthogonalize user knowledge of how to code and user knowledge of where the data are)
- Interoperability of scientifically similar data models from multiple sources
- Interoperability of user code at the level of a "unit of analysis"
  - Notebooks
  - Containers
  - Software packages (3rd-party or user's personal software)
- Interoperability of resource access (e.g., to parallel processing, batch systems)

Evolution of a key set of VO Standards

Standard	Purpose	Adopted	Proposed
TAP	Catalog data access, table metadata access	1.0, 2010	1.1, 2018
ADQL	Catalog query language	2.0, 2008	2.1, 2018
ObsCore/ObsTAP	Basic observation data model and service definition	1.1, 2017	
SIA	Simplified image metadata queries	2.0, 2015	
SODA	On-demand operations like image cutouts	1.0, 2017	
HiPS	All-sky images and catalogs	1.0, 2017	
MOC	All-sky Boolean coverage maps	1.0, 2014	
VOSpace	Remote access to file-oriented data	2.1, 2018	
Supporting Standards			
VOTable	Tabular data transport with metadata	1.3, 2013	
DataLink	Two protocols supporting connecting datasets	1.0, 2015	
DALI	Framework for service definitions	1.1, 2017	
VOSI	Service self-description standards	1.1, 2017	
UWS	Protocol for interacting with asynchronous services	1.1, 2016	
VO-DML	Data model language	1.0, 2017	
UCD1+	Controlled vocabulary for semantics of individual data items	1.3, 2018	

Unfinished business in the existing IVOA ambit

- Fulfilling the promise of the registry and of interoperable discoverability
  - Universal and uniform access to data discovery
  - In practice the quality of registry data is still highly variable, as is the standards-compliance of the registered services
- Interoperable data models that capture a large fraction of the public astronomical data space
- Fully exploiting the standards we have

Unfinished business in the existing IVOA ambit

- Fulfilling the promise of the registry and of interoperable discoverability

- Interoperable data models that capture a large fraction of the public astronomical data space
  - ObsCore and CAOM2 (non-IVOA but a de-facto standard) do a good job for observation metadata and are being used more and more widely
    - ObsTAP and SIAv2 provide protocols for accessing ObsCore-structured data
    - TAP services can support access to CAOM2, but query patterns are complex and difficult for new users to reproduce (see language bindings discussion below)
  - Standardization of a core data model for object/source/detection catalogs has not gotten above threshold for wide usability
    - VO-DML, approved this year, provides a language for this
    - Still trying to work out standards for annotating data with data models
    - No widely acceptable core Source data model yet exists
    - IMO: Need to focus on building this up incrementally (capture 75% of the value with 25% of the effort?)

Unfinished business in the existing IVOA ambit

- Fulfilling the promise of the registry and of interoperable discoverability

- Interoperable data models that capture a large fraction of the public astronomical data space

- Fully exploiting the standards we have
  - High-quality, regularly validated services
  - Clean registries
  - Extensive use of DataLink to allow navigating from data to related data

Client-side access to IVOA services

- Greatest focus has been on data-center to tool-developer relationship
- The development of language bindings to IVOA network protocols and data serializations, or even their standardization, *are not part of the IVOA's remit*
  - The IVOA has no mandate to formally favor any particular language (and doing so would be a form of anti-interoperability)

Client-side access to IVOA services

- Science-user-centric language bindings have been around for years, but not a focus of effort until recently
  - Tension:
    the people who are good at designing and implementing language bindings appropriate to a given language (e.g., Python)
    are generally not the same as
    the people with in-depth understanding of the standards they may be implementing against, and even less so the ones understanding the "big picture" of the architecture: how the IVOA standards are envisioned to work together
  - A language binding that too-explicitly exposes the existence of the underlying protocols is not necessarily what the community wants
    - Astronomers would naturally like to think about domain objects: observations, sources, spectra, and not about formal protocols

Python existing practice: Compare Astroquery and PyVO

- Astroquery is organized around a template for query interfaces, which *may* be used to implement modules that provide access to astronomical data services
  - Interoperability is at the Python level, assuming the template is followed, and makes no assumptions about standardization of the underlying data services
  - Availability of standardized (IVOA) data services does allow many Astroquery modules to share implementation code

- PyVO is explicitly oriented around the existence of the IVOA protocols and provides 1:1 bindings to them
  - Using PyVO requires understanding the existence and purpose of the protocols

- Vision: build out an Astroquery-like interface that hides protocols but takes advantage underneath of high-quality, robust bindings to the protocols
  - Especially: one that encourages the development of a DataLink-based ecosystem

The promise of data modeling

- Progress in the availability and use of rigorous common data models for (subsets of) published astronomical datasets (for observations, catalogs) *enables exposing that data through common language-specific models* (e.g., Python classes)

- Baby steps: recreating Astropy “SkyCoord” objects in the Python bindings to IVOA services that return spherical coordinates
  - How far can this go?
  - Is it in conflict with efficient in-memory data models for tabular data?
  - How much does it help astronomers to have rows in a source database representable as Python objects?

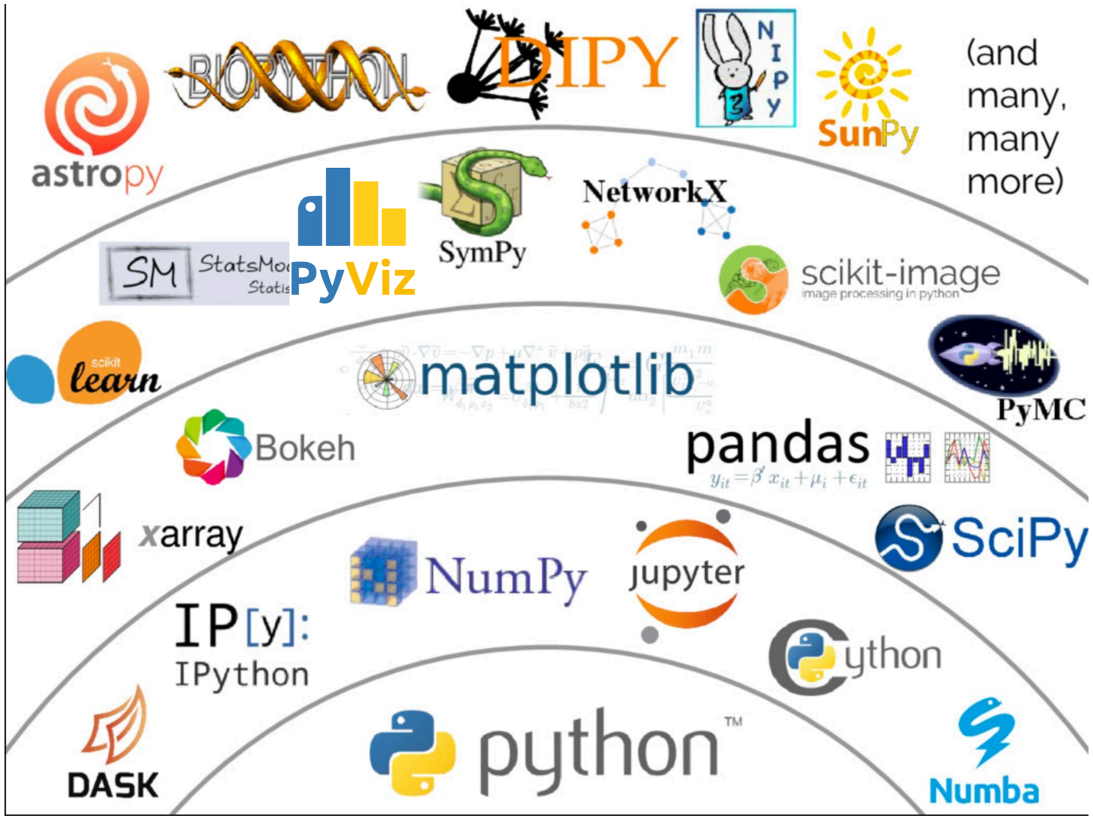
What Interoperability?

Clearly there are many flavors of interoperability that are relevant to Science Platforms

- Interoperability of data services over the network (classic IVOA topic)
- Interoperability of source code that accesses data
  - I want to use the same (or minimally modified) code to access (say) imaging survey data from many sources
  - I want to use the same (or minimally modified) code to access data whether it’s local or remote (orthogonalize user knowledge of how to code and user knowledge of where the data are)
- Interoperability of scientifically similar data models from multiple sources
- Interoperability of user code at the level of a “unit of analysis”
  - Notebooks
  - Containers
  - Software packages (3rd-party or user’s personal software)
- Interoperability of resource access (e.g., to parallel processing, batch systems)

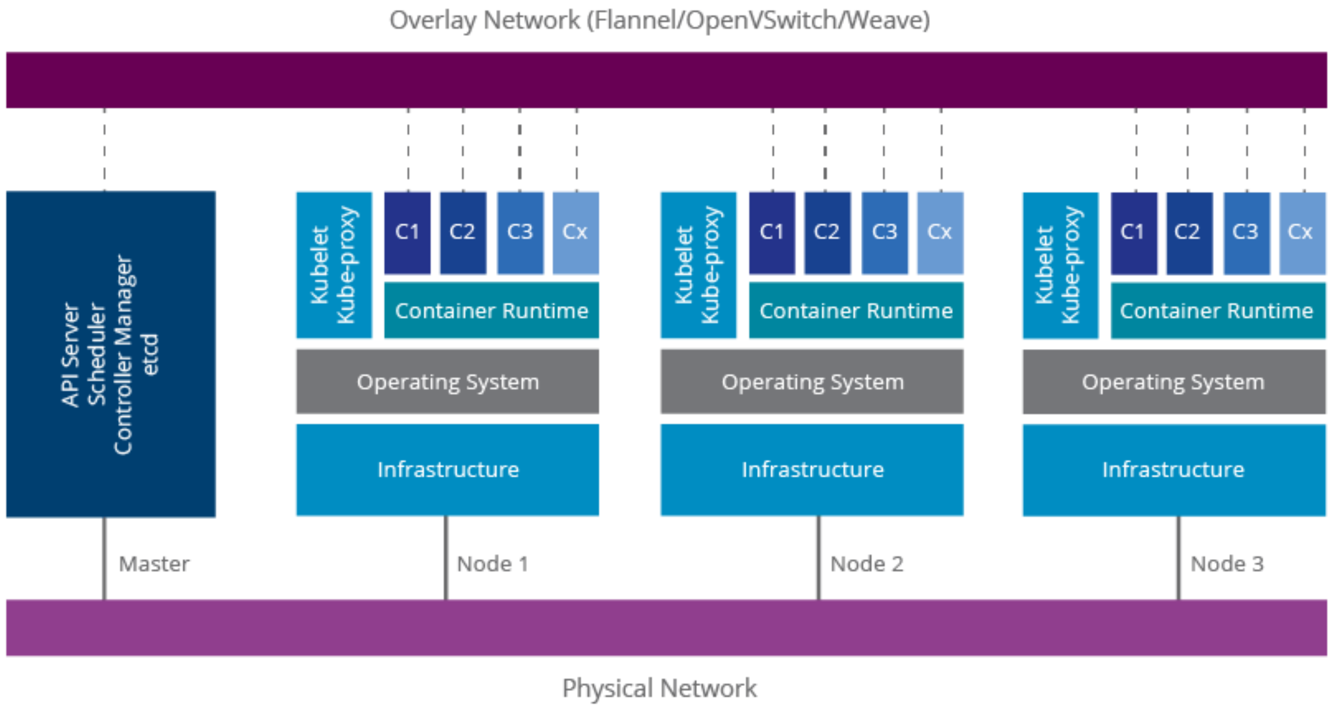
Open-source data analysis codes

- Open-source software efforts are producing robust, mature, and immediately applicable layered libraries that provide out-of-the box solutions for many computational and visualization tasks.
- They are becoming a common language that is known to young people entering our disciplines.



Open-source infrastructure tooling

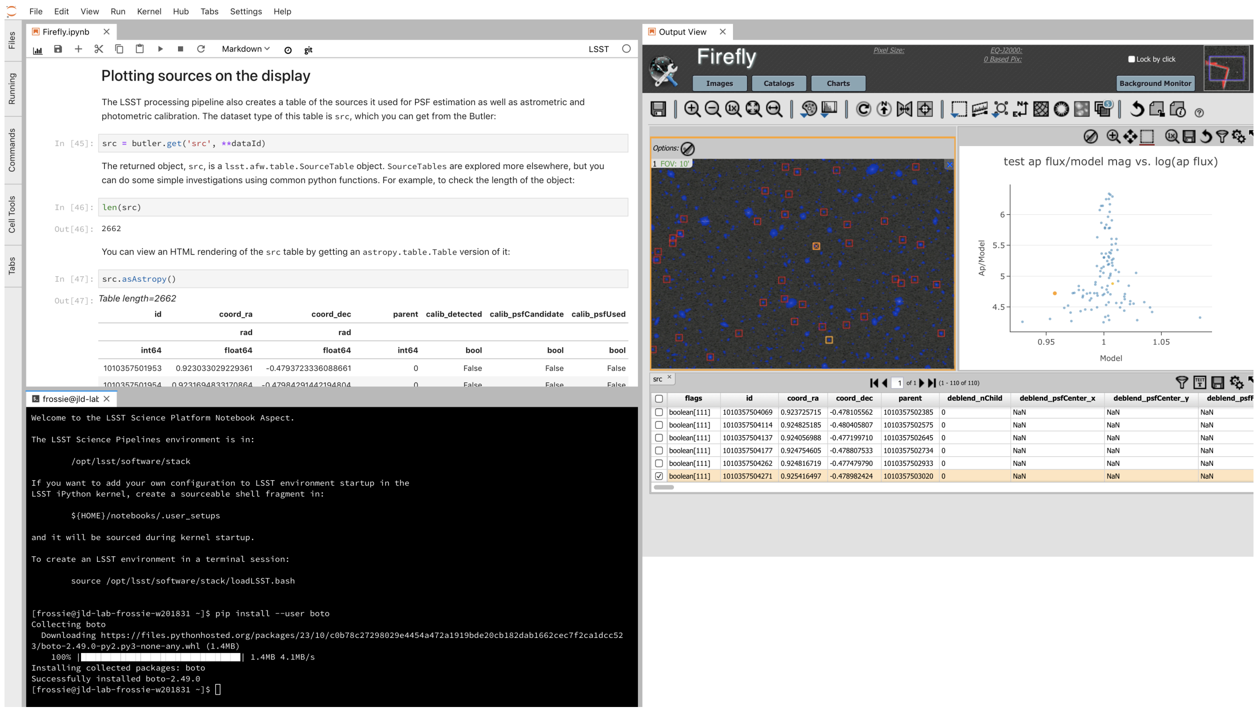
– Containerization frameworks and container-orchestration frameworks such as Docker and Kubernetes are greatly simplifying system administration and application deployment.



kubernetes

Open-source server-side interactive application delivery

– The Web-based notebook interface (here, JupyterLab) is providing a straightforward means of delivering interactive, server-side, next-to-the-data computing to our users.



Plotting sources on the display

The LSST processing pipeline also creates a table of the sources it used for PSF estimation as well as astrometric and photometric calibration. The dataset type of this table is `src`, which you can get from the Butler:

```
In [45]: src = butler.get('src', **dataId)
```

The returned object, `src`, is a `lsst.afw.table.SourceTable` object. `SourceTables` are explored more elsewhere, but you can do some simple investigations using common python functions. For example, to check the length of the object:

```
In [46]: len(src)
Out[46]: 2662
```

You can view an HTML rendering of the `src` table by getting an `astropy.table.Table` version of it:

```
In [47]: src.asAstropy()
```

Out[47]: *Table length=2662*

id	coord_ra	coord_dec	parent	calib_detected	calib_psfCandidate	calib_psfUsed
	rad	rad				
int64	float64	float64	int64	bool	bool	bool
1010357501953	0.923033029229361	-0.4793723336088661	0	False	False	False
1010357501954	0.9231694833170864	-0.47984291442194804	0	False	False	False

frossie@jld-lab

```
Welcome to the LSST Science Platform Notebook Aspect.

The LSST Science Pipelines environment is in:

        /opt/lsst/software/stack

If you want to add your own configuration to LSST environment startup in the
LSST iPython kernel, create a sourceable shell fragment in:

        ${HOME}/notebooks/.user_setups

and it will be sourced during kernel startup.

To create an LSST environment in a terminal session:

        source /opt/lsst/software/stack/loadLSST.bash


[frossie@jld-lab-frossie-w201831 ~]$ pip install --user boto
Collecting boto
  Downloading https://files.pythonhosted.org/packages/23/10/c0b78c27298029e4454a472a1919bde20cb182dab1662cec7f2ca1dcc52
3/boto-2.49.0-py2.py3-none-any.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 4.1MB/s
Installing collected packages: boto
Successfully installed boto-2.49.0
[frossie@jld-lab-frossie-w201831 ~]$ 
```

src — 1 of 1 (1 - 110 of 110)

☐	flags	id	coord_ra	coord_dec	parent	deblend_nChild	deblend_psfCenter_x	deblend_psfCenter_y	deblend_psfF
☐	boolean[111]	1010357504069	0.923725715	-0.478105562	1010357502385	0	NaN	NaN	NaN
☐	boolean[111]	1010357504114	0.924825185	-0.480405807	1010357502575	0	NaN	NaN	NaN
☐	boolean[111]	1010357504137	0.924056988	-0.477199710	1010357502645	0	NaN	NaN	NaN
☐	boolean[111]	1010357504177	0.924754605	-0.478807533	1010357502734	0	NaN	NaN	NaN
☐	boolean[111]	1010357504262	0.924816719	-0.477479790	1010357502933	0	NaN	NaN	NaN
✓	boolean[111]	1010357504271	0.925416497	-0.478982424	1010357503020	0	NaN	NaN	NaN

So what's missing?

In implementing the "Science Platform" vision of offering computing resources close to the data, the following pieces seem not to be as mature or as seamlessly easy for our users to apply:

- Frameworks for efficient multi-tenant application of parallel tools such as Dask
- Cross-site APIs for managing batch jobs and workflows of batch jobs
- Finer-grained transportability of packaged user analysis code across data centers
  - Including from agency-managed assets to commercial cloud providers – support multiple models!
- Tools for automatically matching code to data across multiple data centers
  - Seems to require additional metadata standardization
  - Need widely-adopted solutions to security concerns about container frameworks
- Interoperable authentication and authorization
- Frameworks for reusability, reproducibility, human-friendly version control, collaborative editing, and citeability that work smoothly with the now-popular notebook-oriented environments
  - Be aware of developing concerns/criticism of a too-pervasive shift to notebooks without this

There is a need for R&D, development of experience bases and lessons-learned, and finally (where appropriate) standardization in these areas.