Draft version July 10, 2019
Typeset using LaTeX **modern** style in AASTeX62

The demise of the filesystem and multi level service architecture

William O'Mullane.,[1] Niall Gaffney,[2] Frossie Economou,[1] Arfon M. Smith,[3]
J. Ross Thomson,[4] and Tim Jenness[1]

[1]*Large Synoptic Survey Telescope (LSST/AURA)*
[2]*Texas Advanced Computing Center*
[3]*Space Telescope Science Institute*
[4]*Google*

(Dated: July 10, 2019)

Abstract

Many astronomy data centres still work on filesystems. Industry has moved on; current practice in computing infrastructure is to achieve Big Data scalability using object stores rather than POSIX file systems. This presents us with opportunities for portability and reuse of software underlying processing and archive systems but it also causes problems for legacy implementations in current data centers.

Keywords: Astronomy, Astrophysics, Data Management, Computing, HPC, HTC, Networking, Cloud, Files

1. INTRODUCTION

Executive summary: the filesystem notion limits our ability to scale processing and has long since been dropped by industry. Astronomy needs to move on with a new architecture.

Object Stores are nothing new to Astronomy. Flexible Image Transport System (FITS) and IRAF tapes are examples of object stores that were used when file systems were unable to handle the volumes of data being produced. Even then, standards for migrating from Object Store to file systems were created, allowing for objects to be retrieved into a predefined namespace on disk. The explosion of individual POSIX disk capacity and POSIX-like file systems have produced generations of researchers who have never used an Object Store. While this growth has supported data systems up till now, the size and complexity of data being produced by surveys and even pointed telescope archives is reaching scales where the requirements placed on file access by the POSIX standard are significantly hindering our ability to work with data. Different parallel file systems have different strengths and weaknesses.

At large scale, data service providers such as Dropbox and Amazon Web Services (AWS) do not store files in POSIX systems. Rather they present the illusion of directory structure

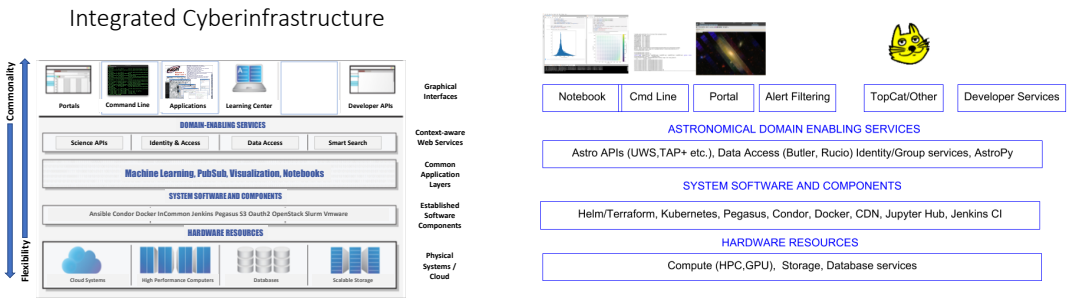

Figure 1. Industry standard Cyber infrastructure model (left) and an astronomy instantiation of such a model (right)

layered over large scale object stores. This allows for faster file access, with only CRUD style functions taking place on each object. Further, as the pseudo-filesystem layer is simply a view of structure typically provided by a graph database, users can arrange or potentially have real time query driven structure for the file organization, removing how many now organize data through a sea of nested symlinks. Some provide local POSIX caches of that users view of the pseudo-filesystem, allowing for POSIX style applications to access the files with `fopen`, `fscan`, and `fclose` standard commands. Additionally, these providers do not show users how data are stored. One can simply request data in the format needed (e.g., Excel, Comma Separated Values (CSV), or as I assume it is the JSON format the web apps use for sheets). At scale, applications often forgo such POSIX layers and simply use the CRUD interfaces to the objects to load them into memory, act on the objects, and then update or delete them, in the format they need as input and with the format they naively produce. Further, the data providers need not update their data archive when formats change, simply provide a new updated data access format that can be fueled by legacy data formats.

It is time for Astronomy data researchers to follow this curve. As users have migrated from using tools on their laptops to support collaboration while reducing individuals need to manage their systems (Jupyter Hubs, Overleaf, Google Slides), so should astronomical data processing and analysis. We propose the adoption of a common Astronomical data access Application Programming Interface (API) layer.

2. RECOMMENDATIONS

1. Develop a community wide architecture supporting Science as a Service following an industry standard layered architecture for astronomy processing and data access as depicted in Figure 1.
2. Compel all funded funded astronomical projects with *software deliverables* to leverage the architecture and APIs for data access.
3. Develop *data and metadata transform services* to support these common API layers to aid interoperation.
4. Adopt and subscribe to a *common federated identity service* (e.g., InCommon or Globus Auth) that is supported by most university and research organizations.

3. COMMODITY SERVICES AND SOFTWARE BASED COMMUNITY ARCHITECTURE

The astronomy and astrophysics community have historically relied on the development and use of bespoke software and hardware infrastructure to solve challenges related to the managing and analyzing datasets at a scale that was difficult to find in industry or other scientific domains. These requirements are no longer unique and we have access to a wealth of open source software, commodity hardware, and managed cloud services (offered by commercial providers and federally-funded institutions) that are well positioned to meet the needs of astronomers and astrophysicists (Momcheva et al. 2019; Bektesevic et al. 2019). By providing documentation and reference implementations of the astronomy stack using these technologies and making it easier for researchers and missions to access cloud computing services, we can reduce operations costs, accelerate time to science, and increase the scientific return on Federally-funded research in astronomy and astrophysics.

Figure 1 shows the layers of the Cyber Infrastructure (CI) from the interfaces for service access exposed at multiple levels, the common domain wide enabled services, and a collection of system level components that support the higher levels of the CI. The lower down the diagram are commodity layers based on well established and supported components. As one moves up from these layers, more abstraction can be done to expose these pieces in domain, or highly specific subdomain, level interfaces. By making these abstractions, more universal service can be developed that can be applied more globally across the entirety of the CI as a whole.

An example of this would be authentication, where each university or agency may provide their own authentication method but unifying services like CILogon can bring those together to give global spaced identity for a wide range of users based on disparate authentication systems. By providing this structure along with a reference architecture of these System Services based on well supported software components, providers are easily able to both deploy and support these common services which enable cross mission and center interoperability. This structure also reflects how this architecture allows for greater reusability as one gets closer to the actual implementation of these services while supporting greater flexibility and general usability as one works further from the core components. Alternatives such as using github authentication may be more flexible but lack the rigor of InCommon which assures the individual is a member of an academic body — we must also work with these industry standards though.

This service architecture should be based on using standard reusable software from many of the established standards developed outside of astronomy (e.g., common authentication mechanisms such as CILogon, standard data and metadata management systems). Standard API interfaces should also be used to expose these components to higher level APIs. Data formatting and metadata structure can be exposed at the service level, allowing for more data and metadata reuse. An example of a storage agnostic data/metadata layer is the Large Synoptic Survey Telescope (LSST) data butler (Jenness et al. 2018). Creation has been driven by the need to abstract data access away from algorithms. The algorithm code

deals with Python objects and never directly with data formats or even storage. The data butler is a lot more than a data access layer, it includes a full registry of all data objects and how to locate them. This means it may sit atop a filesystem or an object store — a prototype S3 plugin is now available and we are testing it in a pipeline. The data butler is astronomy oriented — it has a built in understanding of certain relationships such as between observations and calibrations or between observations and region of the sky. Since all metadate is held in a registry provenance queries can be answered by the butler and it already can act as the registry to find objects in an object store.

Part of a Cyber Infrastructure model such as depicted in Figure 1 is an object store oriented API — this should be used for data sharing. Such APIs exist such as Amazon's S3. The API layers must *expose both data and metadata in common transferable and transformable formats* (e.g., Common Archive Observation Model (CAOM)).

There must also be an authentication source for users at institutions without such means and for citizen science efforts.

4. WHY WE SHOULD KILL THE FILESYSTEM

Users should not care and repositories should not be tied to legacy formats and storage representations because of legacy constraints at other repositories. The rest of the world has already moved on, Google, Amazon, GitHub, Netflix etc. do not host large filesystems and and can scale because they are not limited by this antiquated formalism.

Filesystems with name spaces are very fragile at large scale. As we get larger data sets we have to trick the filesystem to not run out of Inodes, we make countless sub directories to cope with our thousands of files. This is turn leads to countless hours spent fighting over how to organize files the *right way* in a filesystem. Countless years have been spent fighting over data formats (FITS vs Hierarchical Data Format (HDF) (Mink et al. 2015) vs CSV vs Pandas). If we move code then perhaps the filesystem is not organised in the same manner and the code may not work — remote access to allow caching is not always an option.

We need to foster better remote collaboration. The laptop is the bane of file sharing. This has changed with cloud based pseudo-file systems but require storage in a single cloud providers infrastructure. By creating a Filesystem as a Service (FSAAS) federated across data and cloud providers, we will win.

This would imply that *POSIX based file access be deprecated in software development* and only used when applications require thread safe data access (something that is currently not possible with FITS files). We should however develop a pseudo-directory structure system to integrate local and remote files into a dynamic namespace for each user and potentially each users use-case (e.g., the Box sync interface or the FUSE based WholeTale file system (Brinckman et al. 2019))

This "Infrastructure as Code" (Morris 2016) approach lowers the bar to entry and allows for easier adoption of more standardized services that will enable large-scale astronomical research in ways that are well demonstrated in plant genomics (CyVerse and Galaxy), natural

hazards (Designsafe), and surface water research (Hydroshare). (See also the decadal paper on Cloud infrastructure by Arfon Smith et. Al.)

4.1. *The catch*

Switching to an object store removes the filesystem bottle neck however it also removes the filesystem index. This implies that a registry of the objects must be contained. We frequently do this anyway usually sucking meta data into a databases to allow searching — just in the case of an object store this would no longer be optional.

5. CONCLUSION

We should agree on an *astronomy stack* of services with agreed interfaces such that we can concentrate on building domain specific layers on top of industry standard tools.

We should stop worrying about filesystems in astronomy — we should agree on a decent API for object storage and a registry to go with it. The registry should build on existing agreements i.e., based on CAOM-2 (Dowler et al. 2007).

Adoption of a limited set of services will aid ease of use and cut down on wasted effort by all data providers.

REFERENCES

Bektesevic, D., Mehta, P., Juric, M., et al. 2019, in American Astronomical Society Meeting Abstracts, Vol. 233, American Astronomical Society Meeting Abstracts #233, 245.05

Brinckman, A., Chard, K., Gaffney, N., et al. 2019, Future Generation Computer Systems, 94, 854

Dowler, P. D., Gaudet, S., Durand, D., et al. 2007, in Astronomical Society of the Pacific Conference Series, Vol. 376, Astronomical Data Analysis Software and Systems XVI, ed. R. A. Shaw, F. Hill, & D. J. Bell, 347

Jenness, T., Bosch, J. F., Schellart, P., et al. 2018, arXiv e-prints, arXiv:1812.08085

Mink, J., Mann, R. G., Hanisch, R., et al. 2015, in Astronomical Society of the Pacific Conference Series, Vol. 495, Astronomical Data Analysis Software an Systems XXIV (ADASS XXIV), ed. A. R. Taylor & E. Rosolowsky, 11

Momcheva, I., Smith, A. M., & Fox, M. 2019, in American Astronomical Society Meeting Abstracts, Vol. 233, American Astronomical Society Meeting Abstracts #233, 457.06

Morris, K. 2016, Infrastructure as Code: Managing Servers in the Cloud, Safari Books Online (O'Reilly Media)

Glossary

algorithm: A computational implementation of a calculation or some method of processing.. 3

API: Application Programming Interface. 2–5

AWS: Amazon Web Services. 1

CAOM: Common Archive Observation Model. 4, 5

CI: Cyber Infrastructure. 3

CRUD: Create Retrieve Update and Destroy. 2

CSV: Comma Separated Values. 2, 4

FITS: Flexible Image Transport System. 1, 4

Flexible Image Transport System: an international standard in astronomy for storing images, tables, and metadata in disk files. See the IAU FITS Standard for details.. 1, 6

FSAAS: Filesystem as a Service. 4

FUSE: a user space filesystem framework. 4

HDF: Hierarchical Data Format. 4

IRAF: Image Reduction and Analysis Facility. 1

JSON: JavaScript Object Notation. 2

LSST: Large Synoptic Survey Telescope. 3

metadata: General term for data about data, e.g., attributes of astronomical objects (e.g. images, sources, astroObjects, etc.) that are characteristics of the objects themselves, and facilitate the organization, preservation, and query of data sets. (E.g., a FITS header contains metadata).. 2–4

pipeline: A configured sequence of software tasks (Stages) to process data and generate data products. Example: Association Pipeline.. 4

POSIX: Portable Operating System Interface. 1, 2

provenance: Information about how LSST images, Sources, and Objects were created (e.g., versions of pipelines, algorithmic components, or templates) and how to recreate them.. 4

stack: a grouping, usually in layers (hence stack), of software packages and services to achieve a common goal. Often providing a higher level set of end user oriented services and tools. 3, 5