iRODS Policy: Read-only local analysis staging policy for BRAIN-I

Terrell Russell
Renaissance Computing Institute (RENCI)
UNC Chapel Hill
unc@terrellrussell.com

Michelle Itano
Neuroscience Microscopy Core (NMC)
UNC Chapel Hill
itano@unc.edu

Jason Stein
Stein Lab, Department of Genetics
UNC Chapel Hill
jason_stein@med.unc.edu

Oleh Krupa
Stein Lab, Department of Genetics
UNC Chapel Hill
ok37@email.unc.edu

ABSTRACT

The BRAIN-I project, a collaboration to image and study mouse brains between the Renaissance Computing Institute (RENCI) and the UNC Neuroscience Microscopy Core (NMC) at the UNC Neuroscience Center at the UNC-Chapel Hill School of Medicine, has been working on policy to allow researchers to do local analysis of data that is already under management by iRODS. This paper will explain the design decisions, the policy that was developed and deployed, and how it works alongside the rest of the BRAIN-I configuration.

Keywords

iRODS, data management, policy, brain images, neuroscience

INTRODUCTION

This project is a collaboration between multiple groups on the same campus. This includes team members from the Renaissance Computing Institute (RENCI), Steve Cox, Director of Software Architecture, and Terrell Russell, Chief Technologist for the iRODS Consortium. The UNC Neuroscience Microscopy Core (NMC) in the School of Medicine is represented by Michelle Itano, the Core's Director. The Stein Lab in UNC-Chapel Hill Department of Genetics is represented by Jason Stein, Lab Director, and Oleh Krupa, Ph.D.

The Stein Lab studies how variations in the genome affect the structure and development of the brain, and in doing so, create risk for neuropsychiatric illnesses. One of the lab's research projects involves using high-powered optical microscopes to create extremely detailed images of mouse brains.

This paper represents the work done to formalize and codify the process of image data ingest and staging with an eye for local analysis by lab technicians and managers.

ARCHITECTURE

This policy project began with iRODS already in place and NFSRODS[1] providing access to image analysis toolsets.

iRODS UGM 2021 June 8-11, 2021, Virtual

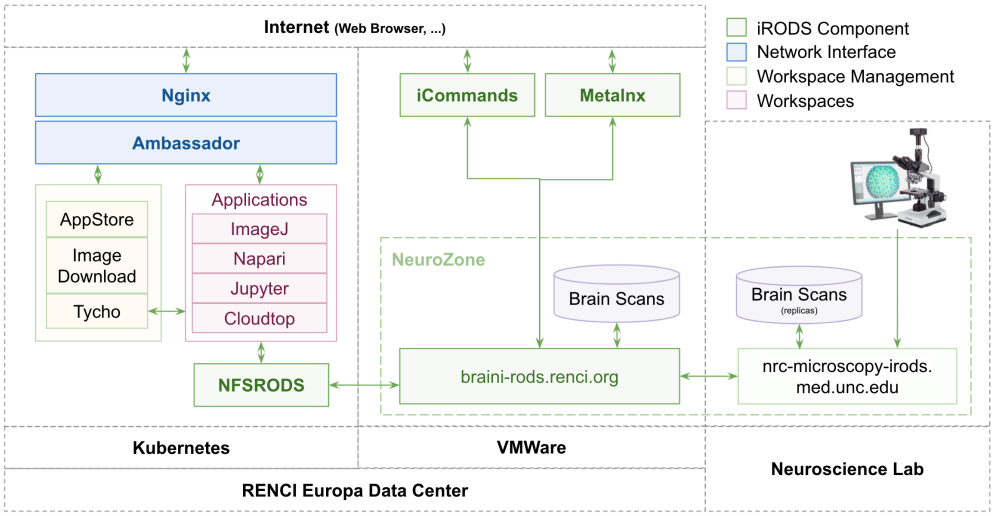

Figure 1. BRAIN-I Architecture

Figure 1 shows the connections between the RENCI Europa Data Center and the Neuroscience Lab in the School of Medicine. The iRODS Catalog Service Provider (`braini-rods.renci.org`) runs in a long-running virtual machine provisioned by VMWare connected to long-term storage provisioned and backed up by RENCI. Various iRODS clients connect to this provider, including the iCommands[2], Metalnx[3], and NFSRODS. NFSRODS runs in the Kubernetes[4] cluster and provides NFSv4.1 access to Data Objects within iRODS to various other applications made available by Kubernetes. An iRODS Catalog Service Consumer is running in the Neuroscience Lab and provides local storage closer to the microscope(s), used for ingest. The policy described in this paper is concerned with the relatively hands-free management of the ingested data and then the ongoing analysis of that data.

USE CASES

There are four main use cases defined and agreed to by the stakeholders of this project. These were developed over 2-3 months in late 2020 through brainstorming sessions, collaborative editing, and reflective meetings and cover the desired and expected workflows of lab members at the time.

The stakeholders involved included scientists doing the capture, lab technicians, the lab director, the microscopy core director, the RENCI infrastructure team, the iRODS development team, and the iRODS administrator of this Zone.

The four main use cases and their envisioned solutions are provided and discussed in the next four subsections, numbered NMC1 through NMC4.

NMC1 - Import of data to iRODS

A lab member saves image data into a well-known location on a local disk attached to the iRODS server (Maybe a folder named 'iRODS import'). (Maybe this can be automated later, but for now... just 'saved/moved' is good enough). The lab member expects the data to appear in iRODS, perhaps with some particular permissions, perhaps with some particular metadata (provided or extracted or associated). The lab expects the data to be managed/moved off the local disk, so it doesn't fill up. Ideally, once in iRODS, there would be a backup (replica) made to a second more secure location. Once copied into another iRODS location with more storage, the import files are removed from the local disk.

SOLUTION - Storage Tiering[5] (and possibly Automated Ingest[6])

NMC2 - Import of data to iRODS and local analysis

A lab member saves image data into a well-known location on a local disk attached to the iRODS server (Maybe a folder named 'iRODS import'). (Maybe this can be automated later, but for now... just 'saved/moved' is good enough). The lab member expects the data to appear in iRODS, perhaps with some particular permissions, perhaps with some particular metadata (provided or extracted or associated). The lab expects the data to be managed/moved off the local disk, so it doesn't fill up. Ideally, once in iRODS, there would be a backup (repica) made to a second more secure location. Once copied into another iRODS location with more storage, the import files are moved to an 'analysis' folder on the local iRODS server.

SOLUTION - Storage Tiering (and labels for 'local analysis') local analysis could be read-only in the vault, perhaps with permissions changed(?!), or another local copy if read/write was important. And any products of local analysis would get routed back through the front door of iRODS. Also possible... NFSRODS, but concerns about network latency are real.

NMC3 - Local iRODS analysis

Analysis of files on the local (NMC) iRODS server from the 'analysis' folder using the local hardware accessible. Could be a custom python script, Matlab code, or through iRODS on ImageJ or Napari. Ideally, would utilize as much computational power as is available, without making it impossible for another user to run a job. Not sure if this can be set to somehow utilize up to 100% or 90% power, but then if another job is started to drop down to maybe 75% or something like that? Or maybe only do that if the job is set to last more than 6 hours, or some relatively arbitrary 'long time'. Leaving 25% for other jobs to still run during that time period? Otherwise for 'shorter' times jobs would be run sequentially with full computational power? Would ideally save analyzed files into the 'iRODS import' folder where again it would be copied and moved to a larger external iRODS server, and removed from the local drive to free up space. Files remaining in the 'analysis' folder would still be accessible to run code on locally.

SOLUTION - Handled by the solution to NMC2

NMC4 - Grant external user iRODS access to published data

Published data would be put on an iRODS protected folder and available post publication for download. Ideally, this would be a very quick process, not requiring a person to verify the request validity. But would also be good to mark how many times files had been downloaded, track who downloaded (maybe just by email address and verifying that address was real?), and if possible allow users to only download specific file sets and not the entire file set. Would want to also ensure that downloading this data wouldn't take up all the bandwidth for the iRODS server.

SOLUTION - Storage Tiering and labeling of particular Collections or Data Objects - policy can fire and run the 'publish()' function, whatever that is defined to be.

DESIGN GOALS

After agreeing on the interesting use cases, the commonalities were identified and turned into design goals that would satisfy the use cases. These were presented as three 'reasons' to move data around the iRODS Zone.

First, the initial ingest of microscope data would be handled by Automatic Storage Tiering, providing a hands-free migration from the local storage in the lab to the long-term storage in the RENCI data center.

The second design goal was manual targeting of interesting data to NMC for local analysis. This could be performed by anyone with appropriate permission in the system and would have a relatively quick effect of staging the data for the scientists to run their local software tools.

The third design goal was manual targeting, using the same mechanism, to a future location as 'published' or to prepare for being published by an as-yet-determined pipeline. This last design goal was deemed to be less important

than the first two, and would later become 'future work'.

Proposed Solution

To satisfy the design goals, the following configuration and algorithm was proposed.

There would be three different iRODS storage resources, `nmc-ingest` and `nmc-analysis` located on the lab workstation, and `renciResc`, theh primary long-term storage at RENCI.

For the first use case (NMC1), the iRODS automated Ingest tool would be configured to register new files 'in-place' into `nmc-ingest`. The iRODS Storage Tiering framework would be configured to automatically migrate data (replicate and trim) from `nmc-ingest` to `renciResc` after a file had been at rest for 12 hours.

For the local analysis use cases (NMC2 and NMC3), there would be two dynamic policy enforcement points (PEPs) that would fire whenever a Collection or Data Object was tagged with the iRODS AVU of `nmc` and `analysis` (no Unit). After metadata was added, the policy would replicate the Collection or Data Object to `nmc-analysis` and set physical permissions to read-only for other unix users on the NMC workstation to be able to perform analysis. After metadata was removed, the replica(s) on the NMC workstation would be trimmed. The original replica(s) would still remain in the long-term storage at RENCI.

The publishing use case (NMC4) would also be handled by a PEP firing after metadata was added to a Collection or Data Object and replicate, then checksum, and then generate a DOI in a public-accessible area of the server.

Takeaways

The main takeaways from this proposed solution were four-fold:

1) There would always be a replica in primary storage at RENCI after the initial ingest and migration progress. This migration could be from nearly 'real-time' to '1 hour' to '1 day' or '1 week' and could be changed at a later date. Additionally, the 'minimum restage tier' could be set to `renciResc`, so no data ever restages back to the NMC workstation upon retrieval or use.

2) Manual tagging and replication gives control to the lab participants to manage their limited disk space on the lab workstation.

3) Manual tagging will be easily extended to provide for a future publication workflow.

4) Existing and future Cloud Apps will pull from the always-available and in-the-same-datacenter `renciResc`, without triggering any replication or data movement other than to the tool itself.

IMPLEMENTED POLICY

As part of the BRAIN-I project, this iRODS policy set defines the policies for data analysis, replication, and retention in the NMC.

`https://github.com/irods/irods_policy_examples/tree/main/nmc_analysis`[7]

There are two parts of the policy managing the data flow within the iRODS Zone:

Automatic

The iRODS Storage Tiering Framework (Figure 2) is handling newly ingested data and moving it into the long-term storage housed at RENCI. RENCI is providing storage and visualization tooling that prioritizes that local, long-term storage. No new policy was written for BRAIN-I for this deployment. The Tiering Framework handled the use case requirements out-of-the-box and only required configuration.

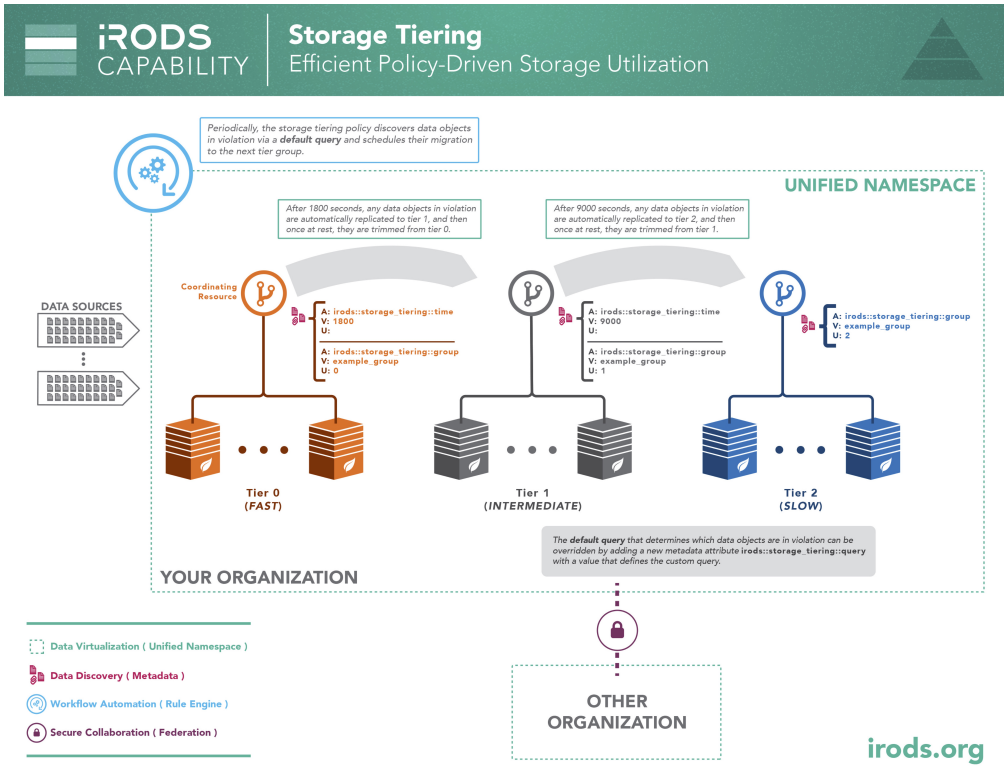

Figure 2. iRODS Storage Tiering Framework

Manual

When NMC staff want to run local analysis on data already in the iRODS namespace, they can 'tag' the data of interest, and this policy will manage the replication to their local machine, set permissions, and prevent removal of that data from the system until it has been 'untagged'. Once 'untagged', the data will be trimmed from the researchers' local storage and remain housed only in long-term storage at RENCI.

This custom policy was designed, written, deployed, and tested for BRAIN-I specifically. However, this mark-and-sweep approach is generic enough it could be extended to meet similar workflow requirements in other domains.

The sweeper does three things every time it fires (configured to be every 30 seconds). It checks which collections and data objects need to be enqueued for replication, and it checks which data objects need to be trimmed.

```
callback.nmc_replicate_dataobjs_under_tagged_collections();
callback.nmc_replicate_tagged_dataobjs();
callback.nmc_trim_untagged_dataobjs_on_target_resource();
```

The other PEPs only run pre-checks to see whether the items being operated on are 'tagged' and therefore protected from being manipulated because they are 'in analysis mode'.

TESTING

To prove that the policy was behaving as expected, a series of BATS[8] tests were written to assert that the system was in a good state before and after each required operation. These scenarios include the simple tagging and untagging

of items in the iRODS Zone, but also their recursive counterparts. When an item is tagged for analysis, neither it nor its descendents can be removed manually, and this must be enforced by policy.

```
$ git clone https://github.com/bats-core/bats-core
$ time bash bats-core/bin/bats test_nmc_analysis.bats
 ✓ tag a collection
 ✓ tag a data object
 ✓ untag a collection
 ✓ untag a data object
 ✓ overwrite a tagged data object
 ✓ overwrite a data object under a tagged collection
 ✓ trim a tagged data object - DISALLOWED
 ✓ trim a data object under a tagged collection - DISALLOWED
 ✓ remove a tagged data object - DISALLOWED
 ✓ remove a tagged collection - DISALLOWED
 ✓ remove a data object under a tagged collection - DISALLOWED
 ✓ remove a collection under a tagged collection - DISALLOWED
 ✓ remove a collection containing a tagged data object - DISALLOWED
 ✓ remove a collection containing a tagged collection - DISALLOWED
 ✓ untag an enqueued data object - DISALLOWED
 ✓ untag a collection with an enqueued descendent data object - DISALLOWED

16 tests, 0 failures

real    2m4.745s
user    0m8.606s
sys     0m2.172s
```

STATUS

At the time of this paper, the BRAIN-I project had no items in active analysis, but had nearly 1 million data objects under management representing nearly 13TB of storage.

```
$ iquest "%s" "select count(DATA_ID) where RESC_NAME = 'nmc-ingest'"
0

$ iquest "%s" "select count(DATA_ID) where RESC_NAME = 'nmc-analysis'"
0

$ iquest "%s" "select count(DATA_ID) where RESC_NAME = 'renciResc'"
921670

$ iquest "%s" "select sum(DATA_SIZE) where RESC_NAME = 'renciResc'" \
  | awk '{print $1/1024^3 " GB "}'
12743.7 GB
```

LIMITATIONS AND FUTURE WORK

The original proposed solution included ingest, tiering, local analysis, and publication, but these were not all implemented in the initial rollout of this policy.

The Automated Ingest could be deployed with a Landing Zone pattern to catch and process the data coming from the microscopes. This was not implemented due to the microscope not being directly attached to the network. Lab personnel currently need to manually transfer the data to the NMC workstation before running `iput`.

Publication can easily be added once more analysis is done and useful data products are ready to be shared with others.

Additional microscopes could be added to this configuration without much trouble. The additional concerns would appear as namespacing problems and should be easily mitigated with good metadata and naming practices.

As the NMC becomes accustomed to having their images available in a platform like BRAIN-I, we expect additional labs to become interested in having something similar or being added to the BRAIN-I infrastructure directly. This should also scale as needed, since additional storage capacity could be added at RENCI and/or at other labs or data centers. The unified namespace of iRODS already accounts for growth and expansion of this kind.

SUMMARY

This paper covers the design process and results of providing a read-only local analysis staging policy for the BRAIN-I project at UNC-Chapel Hill. Live code is available and can serve as a prototype of policy for similar use cases.

iRODS proves flexible and powerful enough to provide transparency and control for a relatively small lab while demonstrating that the platform is capable of driving this type of policy-based approach for any organization.

REFERENCES

[1] Draughn, Kory; Russell, Terrell. iRODS Client: NFSRODS 2.0 (2021). Draughn `https://irods.org/uploads/2021/Draughn-iRODS-NFSRODS_2.0-paper.pdf`

[2] iRODS Client: iCommands `https://github.com/irods/irods_client_icommands`

[3] Zhou, Bo; Draughn, Kory; Coposky, Jason; Russell, Terrell; Conway, Mike; iRODS Client: Metalnx 2.4.0 with GalleryView (2021) `https://irods.org/uploads/2021/Zhou-iRODS-Metalnx_2.4.0_with_GalleryView-slides.pdf`

[4] Kubernetes `https://kubernetes.io/`

[5] iRODS Unified Storage Tiering Rule Engine Plugin. `https://github.com/irods/irods_capability_storage_tiering`

[6] Xu, Hao; King, Alan; Russell, Terrell; Coposky, Jason; de Torcy, Antoine; iRODS Capability: Automated Ingest (2018) `https://irods.org/uploads/2018/Xu-RENCI-Automated_Ingest-paper.pdf`

[7] Russell, Terrell: UNC Neuroscience Microscopy Core (NMC) iRODS Policy (2021). `https://github.com/irods/irods_policy_examples/tree/main/nmc_analysis`

[8] Bats-core: Bash Automated Testing System (2018). `https://github.com/bats-core/bats-core`