iRODS Object Store on Galaxy Server:
Application of iRODS to a Real Time, Multi-user System

KAIVAN KAMALI, NATE CORAOR, MARIUS VAN DEN BEEK, JOHN CHILTON, AND ANTON NEKRUTENKO

PENN STATE UNIVERSITY

Outline

1. Galaxy Intro
2. IRODS in Galaxy
3. Summary

What is Galaxy?

Galaxy (https://galaxyproject.org) is an open-source platform for data analysis. It enables users to

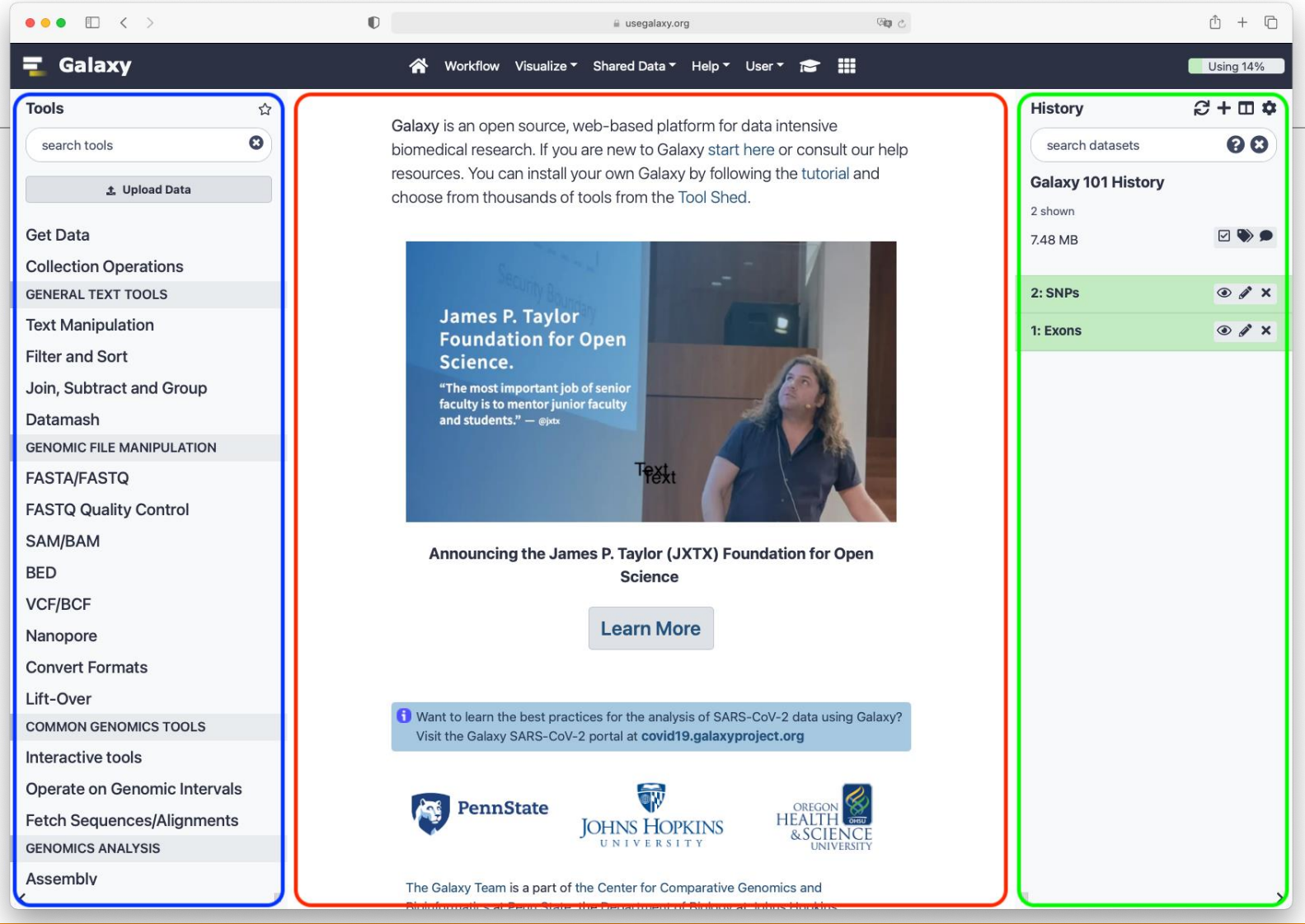
1. Use tools from various domains through its graphical web interface
2. Run code in interactive environments such as Jupyter or RStudio
3. Manage data by sharing and publishing results, workflows, and visualizations
4. Ensure reproducibility by capturing the necessary information to repeat data analyses

Why Galaxy?

Galaxy allows for *accessible*, *reproducible*, and *transparent* computational research

- *Accessibility*: Galaxy's simple user interface provides access to computational tools without requiring knowledge of programming languages
- *Reproducibility*: Galaxy captures sufficient information about every step in an analysis for it to be repeated
- *Transparency*: Galaxy enables sharing of any Galaxy object (datasets, histories, workflows), either publicly, or with specific individuals

Galaxy Interface



Galaxy Availability

Galaxy is available:

- As a free, public, web-based platform, supported by the Galaxy Project
- As open-source software that can be downloaded, installed and customized to address specific needs
- Public web servers hosted by other organizations -- Some have opted to make those servers available to others

Galaxy ToolShed

- ToolShed serves as an "App Store" for all Galaxy instances
- Free service for tool developers and Galaxy admins to host and share Galaxy tools
- Tool developers upload tools to ToolShed
  - Tools are made available to thousands of Galaxy users
- Admins install ToolShed tools on their Galaxy instance
  - ToolShed allows populating any Galaxy instance with thousands of freely available tools

Getting Your Tool Into Galaxy

- Three steps to get your tool into Galaxy
  1. Develop a Conda package for the tool
     - Conda is the de facto standard in many communities to deploy software easily and reproducibly
  2. Create a Galaxy wrapper
     - Wrapper: a formal description of all inputs, outputs and parameters of your tool
     - So that Galaxy can generate a GUI out of it and later a command to send to the compute cluster
  3. Request tool installation on Galaxy instance
     - A Pull Request that needs to be approved

Galaxy Tool Wrapper

```
<tool id="seqtk_seq" name="Convert to FASTA (seqtk)" version="0.1.0">
    <requirements>
        <requirement type="package" version="1.2">seqtk</requirement>
    </requirements>
    <command detect_errors="exit_code"><![CDATA[
        seqtk seq -A '$input1' > '$output1'
    ]]></command>
    <inputs>
        <param type="data" name="input1" format="fastq" />
    </inputs>
    <outputs>
        <data name="output1" format="fasta" />
    </outputs>
    <help><![CDATA[
        TODO: Fill in help.
    ]]></help>
</tool>
```

Planemo

- Tool development is significantly facilitated by using *Planemo*
  - Command-line utilities to assist in developing Galaxy tools
  - Instead of manually creating XML files, Planemo 'tool_init' command generates the boilerplate XML
  - Planemo 'lint' command allows for review of tool XML for validity
  - Planemo 'test' and 'serve' commands allow for running tool tests and serving the tool on a local Galaxy instance

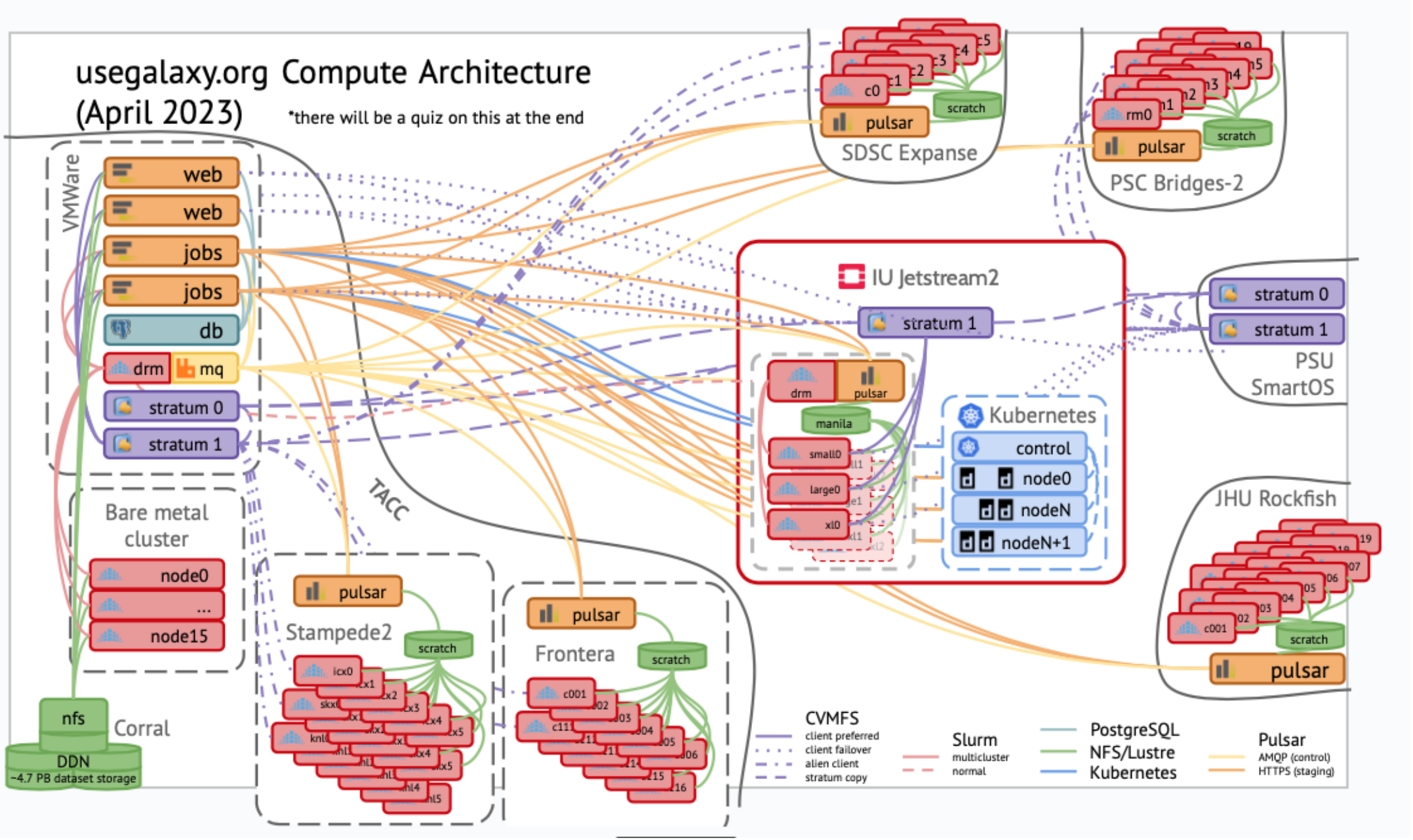
Galaxy Servers

Galaxy Training Network (GTN)

Collection of tutorials developed & maintained by the worldwide Galaxy community

Tutorials for scientists, developers, and admins

Tutorials have slides, hands on section, datasets, workflows, and videos


usegalaxy.org Compute Architecture
(April 2023)
*there will be a quiz on this at the end
VMWare
web
web
jobs
jobs
db
drm
mq
stratum 0
stratum 1
TACC
Bare metal cluster
node0
...
node15
nfs
Corral
DDN
~4.7 PB dataset storage
pulsar
Stampede2
scratch
pulsar
Frontera
scratch
c0
pulsar
scratch
SDSC Expanse
rm0
pulsar
scratch
PSC Bridges-2
IU Jetstream2
stratum 1
drm
pulsar
manila
small0
large0
xl0
Kubernetes
control
node0
nodeN
nodeN+1
stratum 0
stratum 1
PSU SmartOS
JHU Rockfish
c001
scratch
pulsar
CVMFS
client preferred
client failover
alien client
stratum copy
Slurm
multicluster
normal
PostgreSQL
NFS/Lustre
Kubernetes
Pulsar
AMQP (control)
HTTPS (staging)

Outline

1. Galaxy Intro
2. IRODS in Galaxy
3. Summary

Galaxy ObjectStore & Backend

- *ObjectStore:* Galaxy's data virtualization technology
  - Makes it possible to store data on a variety of persistence media & define data distribution policy

- Backend: any persistence media that ObjectStore can be configured to read/write from/to
  - Local storage (e.g., disk)
  - Cloud (e.g. S3)
  - Integrated Rule-Oriented Data Store (iRODS)

Data Distribution

When you have multiple backends, can define nested relationship between them

- *Hierarchical* backends
- *Distributed* backends

	Where data is read from?	**Where data is written to?**
Hierarchical	first backend where data exists	always the first backend
Distributed	first backend where data exists	pseudo-randomly selected backend

iRODS Server for Galaxy

- Hosted on Texas Advanced Computing Center (TACC) at the University of Texas at Austin
  - Galaxy test server (test.galaxyproject.org) and Galaxy Main server (www.usegalaxy.org) both configured to have iRODS as an object store
- Galaxy codebase is in Python
  - Uses Python iRODS Client (PRC) to interact with iRODS
  - PRC v0.9.0+ (along with iRODS server v4.2.9+) supports multi-threaded put/get
  - Provides performance similar to iCommands

Alpha/Beta Testing

- Ran upload/download operations via scripts
- Switched a number of Galaxy developers to iRODS backend
- Monitored Galaxy and iRODS server logs
  - Did not observe any errors or performance issues

- Contacted a large number of Galaxy power users for iRODS beta testing
- Overrode object store access for those users to iRODS
- Monitored Galaxy and iRODS server logs
  - Did not observe any errors or performance issues

Galaxy's iRODS ObjectStore

iRODS parameters are specified in an ObjectStore XML configuration file

```
<?xml version="1.0"?>
<object_store type="irods">
    <auth username="rods" password="rods" />
    <resource name="demoResc" />
    <zone name="tempZone" />
    <connection host="localhost" port="1247" timeout="30" refresh_time="30" connection_pool_monitor_interval="60" />
    <cache path="database/object_store_cache_irods" size="1" />
    <extra_dir type="job_work" path="database/job_working_directory_irods" />
    <extra_dir type="temp" path="database/tmp_irods" />
</object_store>
```

IRODS Configuration Parameters

- **refresh_time**
  - The connection pool in Python iRODS Client (PRC) maintains a set of idle and active connections
  - When app needs a connection, it pops the idle set, and pushes it onto active set
  - Occasionally, if connection popped from the idle set was created a long time ago, saw NetworkException errors in the Galaxy log
  - Seemed older connections would get dropped
  - Introduced 'refresh_time', so if the popped connection was created more than 'refresh_time' seconds ago, it is destroyed, and a new connection is created
  - Change made to PRC

IRODS Configuration Parameters

- connection_pool_monitor_interval
  - Galaxy has facilities to run periodic jobs
  - Before job is run, the worker process creates an irods session, which contains a connection pool
  - The worker process maintins the irods session for subsequent call (possibly hours later)
  - Connections in connection pool go stale, but since they are not used, they are not refreshed based on refresh_time.
  - Created a new thread in Galaxy, that monitors connection pool of all sessions, and destroys connections that are stale
- The thread runs every connection_pool_monitor_interval seconds

Outline

1. Galaxy Intro
2. IRODS in Galaxy
3. Summary

Summary & Future Work

- After CyVerse (https://cyverse.org/about), Galaxy is one of the few applications of iRODS to a real time, multi-user system
- We may want to add a second iRODS server
  - A proxy server sits in front of the two iRODS server instances, for routing and load balancing
  - Provides redundancy and fault tolerance
- Investigate using iRODS storage tiering to seamlessly move objects between slow and fast storage based on their last access time
  - Move files not recently used to slower storage, to free up faster storage

Thank you!

We would like to thank all the iRODS team members for their support!

Questions/Comments?

References

1. Vahid Jalili, et. al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update, *Nucleic Acids Research*, Volume 48, Issue W1, 02 July 2020, Pages W395–W402, https://doi.org/10.1093/nar/gkaa434
2. A Short Introduction to Galaxy (https://training.galaxyproject.org/training-material/topics/introduction/tutorials/galaxy-intro-short/slides.html#1)
3. Introduction to Galaxy. https://training.galaxyproject.org/training-material/topics/introduction/slides/introduction.html#1
4. Galaxy ObjectStore. https://galaxyproject.org/admin/objectstore/
5. Dockerized iRODS Server. https://github.com/kxk302/irods
6. Galaxy Tool wrappers. https://github.com/bgruening/galaxytools
7. Building Galaxy Tools Using Planemo. https://planemo.readthedocs.io/en/latest/writing_standalone.html