Technology Update

Terrell Russell, Ph.D.
@terrellrussell
Chief Technologist, iRODS Consortium

June 25-28, 2019
iRODS User Group Meeting 2019
Utrecht, Netherlands
In The Last Year

<table>
<thead>
<tr>
<th>iRODS Release</th>
<th>Issues Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.12</td>
<td>36</td>
</tr>
<tr>
<td>4.2.4</td>
<td>31</td>
</tr>
<tr>
<td>4.2.5</td>
<td>57</td>
</tr>
<tr>
<td>4.2.6</td>
<td>27</td>
</tr>
</tbody>
</table>

Plugins

- Python Rule Engine Plugin
- Storage Tiering Rule Engine Plugin
- Auditing (AMQP) Rule Engine Plugin
- Update Collection Mtime Rule Engine Plugin
- S3 Resource Plugin
- GSI Authentication Plugin
- Kerberos Authentication Plugin
- Curl Microservice Plugin

Clients

- Python iRODS Client
- Cloud Browser
- Metalnx
- NFSRODS
- Automated Ingest Framework
~/irods $ git shortlog --summary --numbered 4.1.11..4.1.12
  27  Alan King
  11  Terrell Russell
   1  Justin James

~/irods $ git shortlog --summary --numbered 4.2.3..4-2-stable
  39  Alan King
  20  Kory Draughn
  20  Terrell Russell
  14  Andrew Kelly
   6  Jason Coposky
   5  Justin James
   5  Zoey Greer
   5  d-w-moore
   3  Hao Xu
   2  Felix A. Croes
   2  jkgill
   1  Kyle Ferriter
   1  Matt Watson
Ongoing Development Work

- iRODS 4.2.7
- iRODS 4.3.0
- Automated Ingest Capability
- Storage Tiering Capability
- Indexing Capability
- Publishing Capability
- Python iRODS Client (PRC)
- Metalnx
- NFSRODS
- Lustre Integration
- NetCDF Extraction
- Ceph RADOS Resource Plugin
- Cacheless S3 Resource Plugin
- Multipart Transfer, v5 API
- Testing Infrastructure
Steadily filling out the iRODS Data Management Model...

- Auditing - 2017
- Automated Ingest - 2018
- Storage Tiering - 2018
- Indexing - 2019
- Publishing - 2019
- Provenance
- Integrity
- Compliance
Working Groups

Technology Working Group
• Goal: To keep everyone up to date, provide a forum for roadmap discussion and collaboration opportunities

Metadata Templates Working Group
• Goal: To define a standardized process for the application and management of metadata templates by the iRODS Server
  ▪ NIEHS / Data Commons
  ▪ Utrecht / Yoda
  ▪ Maastricht / DataHub+
  ▪ Arizona / CyVerse

Changelog Working Group (Upcoming...)
• Goal: To define a standardized log format from parallel file systems
  ▪ OpenSFS / Lustre
  ▪ IBM / GPFS
  ▪ Panasas / PanFS
  ▪ ThinkParQ / BeeGFS
  ▪ Red Hat / GlusterFS
Last Year and Next Year

- New Libraries
  - Kory Draughn

- irodsDelayServer and Intermediate Replicas
  - Alan King

- Build and Test
  - Jaspreet Gill
New Libraries, Oh My!

Goal: Provide standardized interfaces that simplify common iRODS tasks

Six new libraries (so far):

- **iRODS Query Iterator**
  - Abstracts the GenQuery API making it very easy to fetch information from the catalog
- **iRODS Thread Pool**
- **iRODS Connection Pool**
  - Built with iRODS Thread Pool
- **iRODS Filesystem** (experimental)
  - Implements the ISO C++17 Standard Filesystem Interface for iRODS
- **iRODS IOSTreams** (experimental)
  - Provides standardized interfaces and facilities for reading/writing data objects using different transport protocols (e.g. TCP, UDT, RDMA)
- **iRODS Query Processor**
  - Built with iRODS Query Iterator and iRODS Thread Pool

Benefits:

- Usable in client-side and server-side code
- Developers can accomplish more with less code
- Developers introduce fewer bugs
- Developers can focus on the objective they want to accomplish
- Makes fixing bugs easier

Originally planned for 4.3.0.

Backported to 4.2.5 and 4.2.6 due to their ease of use and immediate impact.
Old irodsReServer (pre-4.2.5)

- Fork-exec model for synchronous work distribution
  - Maximum of 256 rules processed per wake-up
  - Rules to be run later may block other rules
  - Long-running rules may block entire RE server process

New irodsReServer (4.2.5+)

- Rebuilt with iRODS Query Iterator, Thread Pool, and Connection Pool
- Single-Producer/Multi-Consumer
  - Uses query iterator to page over results
  - Limits query to rules ready to execute
  - Rules execute asynchronously using in-memory queue and thread pool

Rename to irodsDelayServer (4.3.0)

- iRODS Query Processor, distributed rule execution, ...
The Missing Link: Intermediate Replicas

**Intermediate replica**
- Replica is registered in ICAT, but data is not yet at rest
- Indicated with '?' via ils

**Putting a data object into iRODS**
- Register all required replicas (per voting) as intermediate before any data movement
- Finalize info in ICAT upon transfer completion

```bash
# Intermediate state of all replicas - an iput to a replication resource with 3 leaves
$ ils -l
/tempZone/home/rods:
  rods          0 repl;ufs0       0 2019-04-08.15:38 ? foo
  rods          1 repl;ufs1       0 2019-04-08.15:38 ? foo
  rods          2 repl;ufs2       0 2019-04-08.15:38 ? foo

# After initial put is complete and before synchronous replication has completed
$ ils -l
/tempZone/home/rods:
  rods          0 repl;ufs0      12345 2019-04-08.15:38 & foo
  rods          1 repl;ufs1      0 2019-04-08.15:38 ? foo
  rods          2 repl;ufs2      0 2019-04-08.15:38 ? foo

# After replication has succeeded
$ ils -l
/tempZone/home/rods:
  rods          0 repl;ufs0      12345 2019-04-08.15:38 & foo
  rods          1 repl;ufs1     12345 2019-04-08.15:38 & foo
  rods          2 repl;ufs2     12345 2019-04-08.15:38 & foo

Stale replicas will now be indicated with 'X'

$ ils -l
/tempZone/home/rods:
  rods          0 resc1       54321 2019-04-08.15:38 & bar
  rods          1 resc2    12345 2019-04-08.15:38 X bar
iRODS Build and Test - History

July 2011

- Python → Node.js → RabbitMQ → Celery → Eucalyptus

October 2012

- Python → Node.js → ssh → OpenStack

January 2013

- Hudson → Python → OpenStack

October 2013

- Hudson → Python → vSphere long-running VMs

Spring 2015

- Jenkins → Python → Ansible → zone_bundles → vSphere dynamic VMs

Spring 2017

- Moved iRODS build/test logic from Ansible to python modules (per-repository)
- Consolidated to two parameterized Jenkins jobs
iRODS Build and Test - 2018 Promises

- Increase coverage (more plugins in CI)
- Move pipeline scripts to GitHub (no logic in Jenkins)
- Address inconsistency (false reds / pyvmomi errors)
- Containerize Jenkins (easier to test / update / redeploy)
- Possibly move from VMs to containers (speed / fewer moving parts)
iRODS Build and Test - Reality

- Everything would need a custom pipeline and logic
- Need externalized infrastructure for some of the tests
iRODS Build and Test - 2019 Architecture

- Dockerized Jenkins
- All configuration and setup in git
- Launches sibling Docker containers
  - Build OS Images
  - Build iRODS Packages
  - Deploy and Test
    - core, plugins, topology, federation
- Development is same as production
An additional DB would increase this test run by 20 containers (8+8+8+16 = 40)

Dockerized equivalent of the current 4-2-stable release process:

- 3 OS, 3 Databases, 31 test suites, 8 Plugins
  - 3 x 3 x 31 = 279 core containers
  - 3 x 3 x 8 = 72 plugin containers
  - 3 x 3 x 2 x Federation subset = ? containers
  - 3 x 3 x 4 x Topology subset = ? containers

<table>
<thead>
<tr>
<th></th>
<th>OS</th>
<th>Database</th>
<th>Containers</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>2</td>
<td>1</td>
<td>2 test suites</td>
<td>4</td>
</tr>
<tr>
<td>Plugins</td>
<td>2</td>
<td>1</td>
<td>2 plugins (1 suite each)</td>
<td>4</td>
</tr>
<tr>
<td>Federation</td>
<td>2</td>
<td>1</td>
<td>2 providers (1 suite each)</td>
<td>4</td>
</tr>
<tr>
<td>Topology</td>
<td>2</td>
<td>1</td>
<td>4 (1 provider + 3 consumers)</td>
<td>8</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>20</strong></td>
</tr>
</tbody>
</table>
iRODS Build and Test - Future

- Make iRODS Jenkins publicly accessible
- Investigate scaling up
- Increase coverage
- Approachable for community developers
  - Confidence
  - Acceptance Criteria
4.3.0 Update

- Checksums moving down into resource plugins
- JSON configuration/schema consolidation
- Use latest releases of irods-externals
- Logging overhaul
4.3.0 Logging Update

Today
- Quiet for well-behaved systems
- Inconsistent formatting
- Incomplete (syslog support)
- Not very helpful in tracking a root cause for errors
- Not very helpful when multiple servers are involved

Design Goals
- Reduce code - Leverage an existing logging library (spdlog, etc.)
- Enable admins to easily capture, process, and analyze activity
- Consistent formatting
- Easily track errors across multiple servers (hostname, timestamp, PID, plugin, etc.)
- Tie into existing infrastructure
- Provide more options for controlling output

<table>
<thead>
<tr>
<th></th>
<th>Local Files (rsyslog)</th>
<th>Remote (rsyslog)</th>
<th>--stdout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packaged</td>
<td>default</td>
<td>centralized logging</td>
<td>Docker-friendly</td>
</tr>
<tr>
<td>Non-Package Install</td>
<td>probably n/a</td>
<td>probably n/a</td>
<td>HPC and development</td>
</tr>
</tbody>
</table>
Policy Composition

With the new libraries, we can rewrite 90% of the internals, and then fix the things that depend on them later, with little expectation of regression, because the interfaces remain the same.

Internally

- We will have a new API... but not really
- Instead, we stepped back and built good tools
  - Allows us to refactor and go faster without breaking the 4.x API
  - This has turned out to be more powerful than expected

Externally

- It's a good story, the ability to compose policy into capabilities
- Can build smaller pieces of functionality which can be composed to help solve larger problems
- We don't have to worry about side effects

Continuation within the Rule Engine Plugin Framework allows administrators to break apart monolithic PEP implementations into reusable components.
Big Picture

Core

- 4.3.0 - Harden and Polish
- 5.0.0 - Simplify API, Drop federation with 3.x

Clients

- GUIs (Metalnx, et al.)
- Onboarding and Syncing (Automated Ingest)
- File System Integration (NFSRODS / CIFSRODS)
- iRODS Console (alongside existing iCommands)

Continue building out policy components (Capabilities)

We want installation and management of iRODS to become about policy design, composition, and configuration.

Please share your:

- use cases
- pain points
- hopes and dreams
Open Source Community Engagement

Get Involved

- Working Groups
- GitHub Issues
- Pull Requests
- Chat List
- Consortium Membership

Tell Others

- Publish, Cite, Advocate, Refer