iRODS Client: NFSRODS 2.0

Kory Draughn
Renaissance Computing Institute (RENCI)
UNC Chapel Hill
korydraughn@renci.org

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

ABSTRACT

NFSRODS has been updated to version 2.0 and now provides significant performance improvements and caching capabilities. This talk will cover what has changed and future work.

Keywords

iRODS, client, NFS, NFSv4, data management

INTRODUCTION

Last year's NFSRODS 1.0 release[1] marked the point where iRODS filesystem integration became easy for system administrators to deploy and support. The functionality was in place, but there had been no usage. After a few deployments, the feedback began to come in and with large numbers of users and files, NFSRODS was not capable of keeping up. This work describes the efforts in moving from NFSRODS 1.0 to NFSRODS 2.0[2].

CHANGES SINCE 1.0

Many of the changes in NFSRODS 2.0 are incremental and represent standard software development and refinement. These include updates of its software dependencies, cleanup of warnings, and better integration with a Docker-based deployment environment.

Since iRODS 4.2.9 incorporated the functionality of the Update Collection MTime Rule Engine Plugin[3], that dependency on the server has been relaxed.

More substantial changes are represented by the work done to increase performance for the list operation.

NEW CONFIGURATION

NFSRODS 2.0 introduces six additional configuration options for deployment. These represent four cache timeout settings, a boolean to determine whether overwriting existing data objects is allowed through the NFS interface, and a workaround for an Oracle-backed iRODS deployment.

```
// The refresh time for cached iRODS object type information.
"object_type_refresh_time_in_milliseconds": 300000,

// The refresh time for cached iRODS user permissions information.
"user_permissions_refresh_time_in_milliseconds": 300000,

// The refresh time for cached iRODS user type information.
```

iRODS UGM 2021 June 8-11, 2021, Virtual

```
"user_type_refresh_time_in_milliseconds": 300000,

// The refresh time for cached GenQuery results used to produce the
// output of a list operation.
"list_operation_query_results_refresh_time_in_milliseconds": 30000,

// Specifies whether the force flag should be applied when overwriting
// an existing file. If this option is false, an error will be reported
// back to the client.
"allow_overwrite_of_existing_files": true,

// Allows NFSRODS to make adjustments for running against an Oracle-backed
// iRODS deployment.
"using_oracle_database": false
```

LIST OPERATION

The list operation of NFSRODS 1.0 had performance issues. It could not handle large collections. This was manifested by the return of truncated and inconsistent results. iRODS servers that were backed by Oracle databases were not returning the correct results. Redundant information was being queried from the catalog, and NFSRODS was naively executing identical queries upon every operation without caching any results.

To fix these list operation shortcomings, NFSRODS 2.0 now caches many, if not all, results retrieved from the Jargon (Java)[4] layer which interacts with iRODS. The Oracle support has been corrected and the default log level has been lowered to reduce time spent writing to disk.

Additionally, best practice has been established to turn off color listings in the calling terminal to reduce the number of stat operations and therefore reduce the network traffic with the connected iRODS server.

PERFORMANCE

To test the new speed of NFSRODS 2.0, the following test harness was configured:

- iRODS provider (backed by PostgreSQL) running on Ubuntu 16.04 (32 cores)
- NFSRODS container running on the same machine hosting the provider
- Set NFSRODS log level to INFO
- Mount command: `sudo mount -o port=2050 localhost:/ /mnt/nfsrods`
- Timing command: `time /bin/ls <collection> | wc -l`

Collection Size	1.0	2.0 (no cache)	2.0 (cached)
500	1m7.211s	0m0.857s	0m0.046s
1,000	3m49.246s	0m0.708s	0m0.041s
3,000	31m45.147s	0m2.345s	0m0.045s
6,000	86m56.771s (results truncated)	0m5.274s	0m0.052s
10,000	87m18.675s (results truncated)	0m9.302s	0m0.058s

Table 1. Performance Comparison between NFSRODS 1.0 and 2.0 (n=5)

Table 1 illustrates both the troubles that NFSRODS 1.0 had with large collections as well as the increased performance of NFSRODS 2.0. Each cell represents the mean of 5 runs, except the two largest collection sizes (6,000 and 10,000) for NFSRODS 1.0 where the results were truncated. An initial listing of a collection with 3,000 data objects was reduced by a factor of over 800x from over 31 minutes to just over 2 seconds. Once that listing information had been seen by NFSRODS 2.0 and cached, a second listing returned the results from local memory in 1/20th of a second, another 50x speedup.

These improvements make NFSRODS appear much more transparent to the user and ready for production deployments.

FUTURE WORK

While this optimization work makes NFSRODS usable, there is still work to be done. Large file transfers are still serial and can appear painfully slow to users who are used to the multi-threaded, network saturating performance of the iCommands. Parallel I/O support is now the highest priority feature to be added next.

Support for iRODS metadata, perhaps via extended file attributes will be investigated since iRODS metadata is currently not visible through the NFS network interface. Additionally, test coverage and error messages need to be improved.

SUMMARY

NFSRODS 2.0 provides a significant increase in performance for collection listings. This deployable production release will encourage continued community feedback about the best way to present iRODS to existing tools and applications.

REFERENCES

[1] Draughn, K., Russell, T., Mieczkowski, A., Coposky, J., Conway, M.: iRODS Client: NFSRODS 1.0. 8pp. 2020 iRODS User Group Meeting. (2020) `https://irods.org/uploads/2020/Draughn-iRODS-NFSRODS_v1.0.0-paper.pdf`

[2] iRODS Client - NFSRODS. `https://github.com/irods/irods_client_nfsrods`

[3] iRODS Rule Engine Plugin - Update Collection MTime. `https://github.com/irods/irods_rule_engine_plugin_update_collection_mtime`

[4] Jargon - iRODS Java client library. `https://github.com/DICE-UNC/jargon`