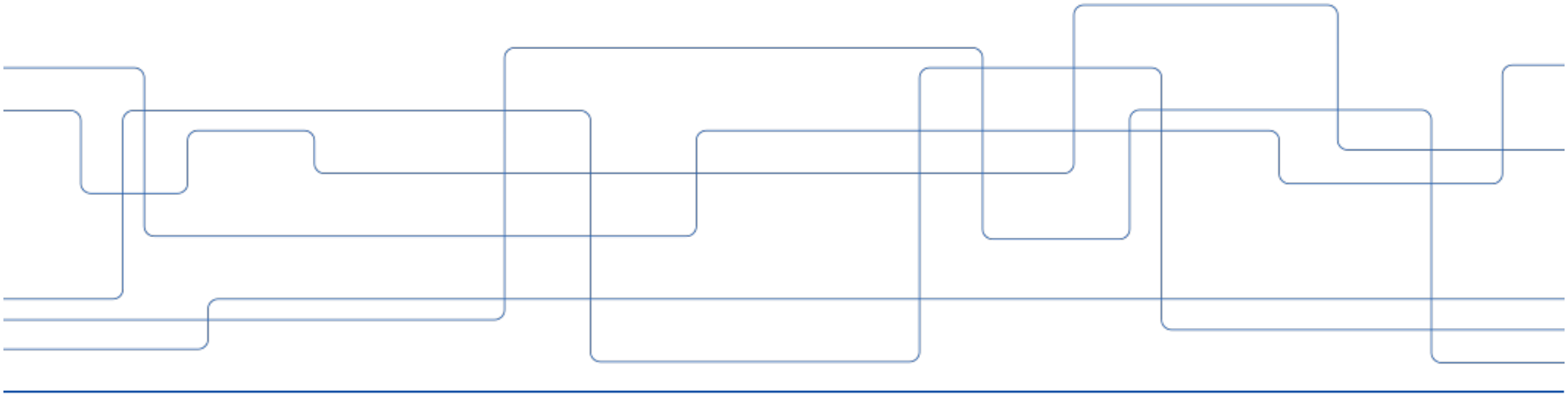
KTH ROYAL INSTITUTE OF TECHNOLOGY

Parallel Data Migration Between GPFS Filesystems via the iRODS Rule Engine

Ilari Korhonen, PDC Center for High Performance Computing

The 12th Annual iRODS Users Group Meeting

June 9th 2020, The Internet



Background

- PDC is a HPC center based at KTH, Stockholm, Sweden
- We are a member of SNIC (a consortium) which facilitates HPC in Sweden
- Naturally we also do high performance and scalable data storage and hence, data management
- At PDC we host a part of the SNIC-funded national storage platform Swestore, which has two separate islands, one running dCache, one running iRODS, both distributed systems nationwide, users being provisioned from a common system
- The storage subsystems hosting the data iRODS manages are heterogeneous, currently we use both ZFS and GPFS, the GPFS tier being the performance tier, a landing zone and ZFS the secondary tier for reliable and scalable capacity
- At KTH we are running a GPFS cluster hosting the filesystems for the iRODS landing zone
- The GPFS cluster and its (physical) filesystems are due for an upgrade!

GPFS Cluster Upgrade

- We procured and accepted the storage system late 2017 from IBM, it is an ESS system running on POWER8 and was set up with GPFS v4.2.3 (which is the previous major release)
- At the time, we were aware that IBM was about to release a new major version of GPFS (5.0), which was rumoured to have numerous enhancements, some of them very relevant to us - especially the space efficiency with respect to on-disk blocks allocation
- GPFS 4.2.3 is able to split a physical on-disk block into 32 sub-blocks, which can be used for file allocations, which is fine for a small block size, we use 8 MiB
- GPFS 5.0.0 on the other hand is capable of a variable sub-block size, and for 8 MiB physical blocks it can be split into 512 of 16 KiB sub-blocks
- Since we have the burden of storing a variety of workloads, some of which include a large number of small files, we prepared from the beginning for a future on-disk format upgrade to 5.0.x

GPFS Cluster Upgrade

- To facilitate an upcoming upgrade, we prepared ourselves with splitting the available storage into two physical GPFS filesystems
- This enables us to do an upgrade w/o going to tape, or replicas on file systems (unless because of an unexpected error / emergency)
- We can simply upgrade one on-disk filesystem at a time and migrate the data online using iRODS w/o any visible effect to the users
- The only part of this which is planned to be done offline is the GPFS cluster upgrade, meaning firmware, operating system and software upgrades.
- Also the previously mentioned is theoretically possible to be done online according to IBM manuals, however recommended to be done offline
- The upgrade procedure was planned for late May / early June, but has now been postponed due to the system administrator being on sick leave.

GPFS Cluster Upgrade

- Planned procedure (roughly):
    1. Turn off iRODS, after waiting for users to disconnect
    2. Upgrade all firmware images, reboot H/W (several times)
    3. Power off physical storage enclosures (to safeguard all data on disk)
    4. Upgrade operating system on the ESS management node
    5. Provision upgraded node images into the ESS I/O nodes
    6. Reboot all nodes, and bring the cluster back online
    7. Power on physical storage and verify GPFS RAID status
    8. Reformat GPFS filesystem `fs0` into the new on-disk format
    9. Migrate online using iRODS, all iRODS resources from `fs1` -> `fs0`
    10. Reformat GPFS filesystem `fs1` into the new on-disk format
    11. Rebalance data, splitting resources evenly between new filesystems

Parallel Data Migration using iRODS

- The prerequisite for the previous plan was of course, that `fs0` was already cleared out 😄
- This was done already a while ago via the parallel execution provided by the new rule engine in iRODS 4.2.x

- Essentially a two-phase process:

    1. Mass replicate data objects from one set of resources to a mirroring set
    2. Mass trim old copies from the source resources, draining the filesystem

```
syncRescAtPath
{
    # get all object replicas present at source and loop over
    foreach (*row0 in SELECT COLL_NAME, DATA_NAME WHERE COLL_NAME LIKE '*collPath%' AND DATA_RESC_NAME = '*sourceRescName')
    {
        *skipObj = 0;

        *collName = *row0.COLL_NAME;
        *dataName = *row0.DATA_NAME;

        *objPath=*row0.COLL_NAME++"/"++*row0.DATA_NAME;

        # loop over resources where data object is present
        foreach (*row1 in SELECT DATA_RESC_NAME WHERE COLL_NAME = *collName AND DATA_NAME = *dataName)
        {
            # we skip this object if present at target
            if (*row1.DATA_RESC_NAME == *targetRescName)
            {
                *skipObj = 1;
                writeLine("stdout", "*sourceRescName -> *targetRescName: skipping object path '*objPath'");
            }
        }

        # otherwise we enqueue a replication job for this object
        if (*skipObj == 0)
        {
            writeLine("stdout", "*sourceRescName -> *targetRescName: enqueue replication job for object path '*objPath'");

            delay("<PLUSET>0m</PLUSET>")
            {
                msiDataObjRepl(*objPath, "rescName=*sourceRescName++++destRescName=*targetRescName++++irodsAdmin=", *status);
                writeLine("serverLog", "ASYNC: syncRescAtPath: *sourceRescName -> *targetRescName: replicated objPath '*objPath', sta
            }
        }
    }
}
INPUT *sourceRescName="fs0resc0", *targetRescName="fs1-fs0resc0", *collPath="/snic.se/projects/operations"
OUTPUT ruleExecOut
```


```
trimRescAtPath
{
    # get all object replicas present at source and loop over
    foreach (*row in SELECT COLL_NAME, DATA_NAME WHERE COLL_NAME LIKE '*collPath%' AND DATA_RESC_NAME = '*sourceResc')
    {
        *collName = *row.COLL_NAME;
        *dataName = *row.DATA_NAME;

        *objPath=*row.COLL_NAME++"/"++*row.DATA_NAME;
        writeLine("stdout", "*sourceResc: enqueue trim job for object path '*objPath'");

        delay("<PLUSET>0m</PLUSET>")
        {
            msiDataObjTrim(*objPath, *sourceResc, "null", "2", "irodsAdmin", *status);
            writeLine("serverLog", "ASYNC: trimRescAtPath: *sourceResc: trimmed objPath '*objPath', status=*status");
        }
    }
}
INPUT *sourceResc="fs0resc0", *collPath="/snic.se/projects/operations"
OUTPUT ruleExecOut
```

```
irods_server_config:
  advanced_settings:
    maximum_number_of_concurrent_rule_engine_server_processes: 16
    rule_engine_server_sleep_time_in_seconds: 1
```

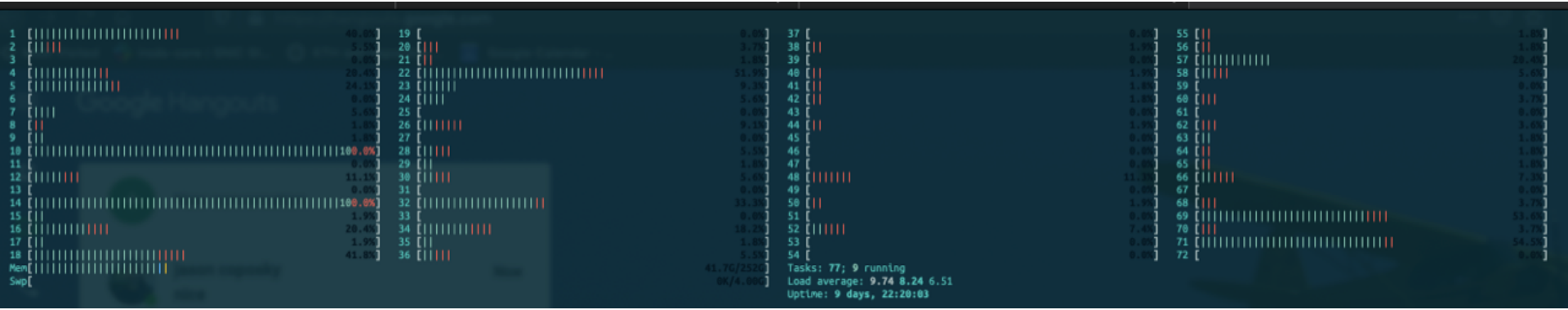
```
climbingcatfish$ for proj in blaah; do for i in {0..3}; do irule -F syncRescAtPath.r
"*sourceRescName='fs0resc${i}'" "*targetRescName='fs1-fs0resc${i}'" \
        "*collPath='/snic.se/projects/${proj}'" | tee syncRescAtPath-projects-${proj}-fs0resc$
{i}-$(date --iso-8601=seconds).txt; done; done
```

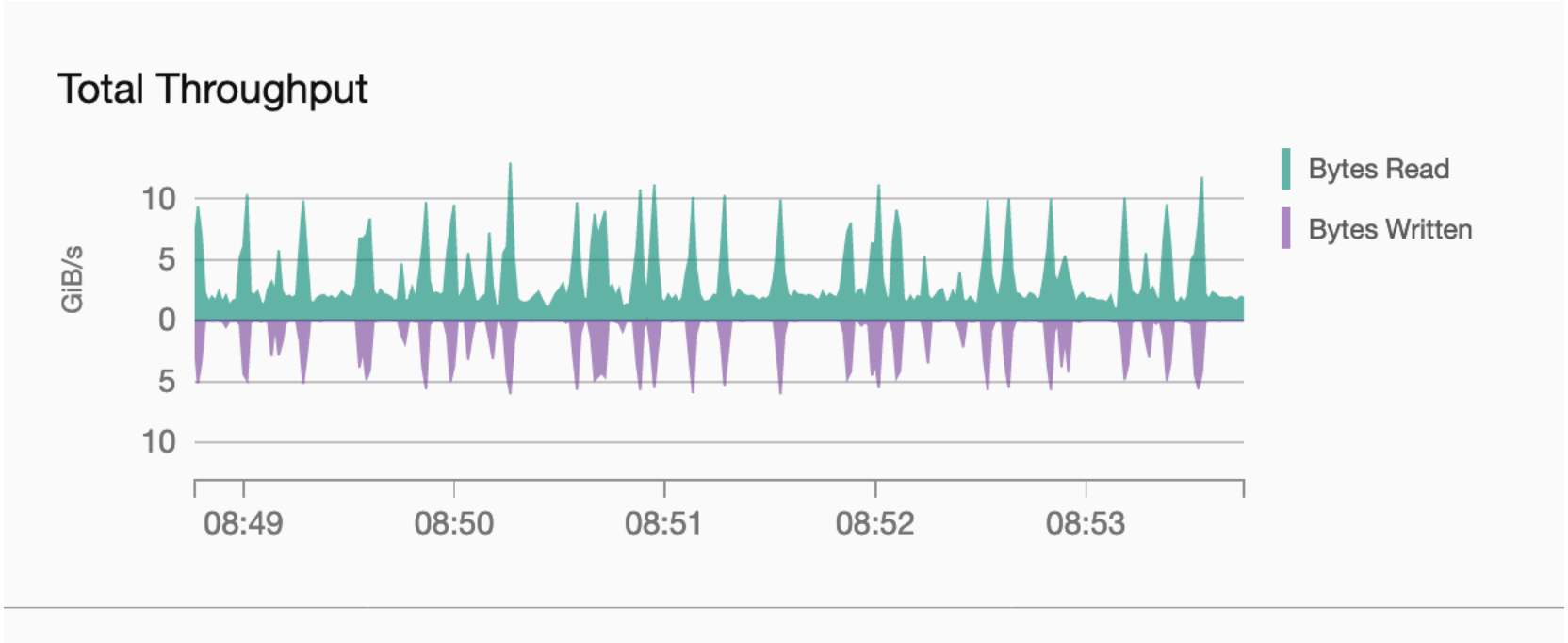
```
Nov 16 17:32:40 pid:27246 remote addresses: 127.0.0.1 ERROR: cllConnect: SQLConnect failed: -1
Nov 16 17:32:40 pid:27246 remote addresses: 127.0.0.1 ERROR: cllConnect: SQLConnect failed:odbcEntry=iRODS
Catalog,user=irods,pass=XXXXX
Nov 16 17:32:40 pid:27246 remote addresses: 127.0.0.1 ERROR: cllConnect:          SQLSTATE: 08001
Nov 16 17:32:40 pid:27246 remote addresses: 127.0.0.1 ERROR: cllConnect:  Native Error Code: 101
Nov 16 17:32:40 pid:27246 remote addresses: 127.0.0.1 ERROR: cllConnect: [unixODBC]FATAL:  remaining connection slo
are reserved for non-replication superuser connections
```

```
irods_server_config:
  advanced_settings:
    maximum_number_of_concurrent_rule_engine_server_processes: 8
    rule_engine_server_sleep_time_in_seconds: 5
```

```
ICAT=# select count(*) from pg_stat_activity;
 count
-------
  1019
(1 row)

capelin$ ps aux | grep irodsServer | wc -l
1021
```


Tasks: 77; 9 running
Load average: 9.74 8.24 6.51
Uptime: 9 days, 22:20:03


Total Throughput
GiB/s
10
5
0
5
10
08:49
08:50
08:51
08:52
08:53
Bytes Read
Bytes Written

And… results were mostly good but…

```
Nov 16 23:36:10 pid:23718 NOTICE: dataCreate: l3Create of /gpfs/fs1/iRODS/fs0resc3/Vault/projects/
icos/[path1] failed, status = -38000
Nov 16 23:36:10 pid:23718 NOTICE: dataCreate: l3Create of /gpfs/fs1/iRODS/fs0resc3/Vault/projects/
icos/[path1] failed, status = -38000
Nov 16 23:36:10 pid:23718 DEBUG: msiDataObjRepl: rsDataObjRepl failed /snic.se/projects/icos/
[path1], status = -38000
caused by: DEBUG: msiDataObjRepl: rsDataObjRepl failed /snic.se/projects/icos/[path1], status =
-38000
Nov 16 23:36:11 pid:23718 NOTICE: dataCreate: l3Create of /gpfs/fs1/iRODS/fs0resc3/Vault/projects/
icos/[path2] failed, status = -38000
Nov 16 23:36:11 pid:23718 NOTICE: dataCreate: l3Create of /gpfs/fs1/iRODS/fs0resc3/Vault/projects/
icos/[path2] failed, status = -38000
Nov 16 23:36:11 pid:23718 DEBUG: msiDataObjRepl: rsDataObjRepl failed /snic.se/projects/icos/
[path2], status = -38000
caused by: DEBUG: msiDataObjRepl: rsDataObjRepl failed /snic.se/projects/icos/[path2], status =
-38000
```

Cleaning up

```
$ for file in syncRescAtPath-projects-icos-fs0resc*2019-11-18*.txt.gz; do zcat $file | grep -v
skipping; done
fs0resc3 -> fs1-fs0resc3: enqueue replication job for object path '/snic.se/projects/icos/[path1]'
fs0resc3 -> fs1-fs0resc3: enqueue replication job for object path '/snic.se/projects/icos/[path2]'
```

```
climbingcatfish$ iquest "%s" "select COLL_NAME where COLL_PARENT_NAME = '/snic.se/projects'" |
while read objpath; do for resc in {fs1-,}fs0resc{0..3}; do iquest "object count (${resc}/$
{objpath}): %s" "select count(DATA_ID) where COLL_NAME like '${objpath}%' and DATA_RESC_NAME = '$
{resc}'"; done; echo "---"; done | tee snic.se-projects-objcounts-$(date --iso-8601=seconds).txt
```

```
climbingcatfish$ for i in {0..3}; do irule -F syncRescAtPath.r "*sourceRescName='fs0resc${i}'"
"*targetRescName='fs1-fs0resc${i}'" "*collPath='/snic.se/'" | tee syncRescAtPath-snic.se-fs0resc$
{i}-$(date --iso-8601=seconds).txt; done
```

Cleaning up

```
climbingcatfish$ for resc in {fs1-,}fs0resc{0..3}; do iquest "object count (${resc}): %s" "select
count(DATA_ID)' WHERE DATA_RESC_NAME = '${resc}'"; done
object count (fs1-fs0resc0): 836916
object count (fs1-fs0resc1): 836576
object count (fs1-fs0resc2): 834842
object count (fs1-fs0resc3): 836643
object count (fs0resc0): 836917
object count (fs0resc1): 836576
object count (fs0resc2): 834842
object count (fs0resc3): 836643
```

Cleaning up

```
climbingcatfish$ iquest "%s/%s" "select COLL_NAME, DATA_NAME where DATA_RESC_NAME = 'fs0resc0'" --
no-page | sort > fs0resc0-objpaths.txt
climbingcatfish$ iquest "%s/%s" "select COLL_NAME, DATA_NAME where DATA_RESC_NAME = 'fs1-
fs0resc0'" --no-page | sort > fs1-fs0resc0-objpaths.txt
climbingcatfish$ diff fs0resc0-objpaths.txt fs1-fs0resc0-objpaths.txt
climbingcatfish$ echo $?
1

climbingcatfish$ irm -f /snic.se/trash/orphan/rods#snic.se/bigfile.2151629042

climbingcatfish$ iquest "%s/%s" "select COLL_NAME, DATA_NAME where DATA_RESC_NAME = 'fs0resc0'" --
no-page | sort > fs0resc0-objpaths.txt
climbingcatfish$ iquest "%s/%s" "select COLL_NAME, DATA_NAME where DATA_RESC_NAME = 'fs1-
fs0resc0'" --no-page | sort > fs1-fs0resc0-objpaths.txt
climbingcatfish$ diff fs0resc0-objpaths.txt fs1-fs0resc0-objpaths.txt
climbingcatfish$ echo $?
0
```

Trim Jobs

```
climbingcatfish$ for i in {0..3}; do irule -F trimRescAtPath.r "*sourceResc='fs0resc${i}'"
"*collPath='/snic.se/home'"; done | tee trimRescAtPath-fs0resc${i}-home-$(date --
iso-8601=seconds).txt
climbingcatfish$ for i in {0..3}; do irule -F trimRescAtPath.r "*sourceResc='fs0resc${i}'"
"*collPath='/snic.se/migration'"; done | tee trimRescAtPath-fs0resc${i}-migration-$(date --
iso-8601=seconds).txt
climbingcatfish$ for i in {0..3}; do irule -F trimRescAtPath.r "*sourceResc='fs0resc${i}'"
"*collPath='/snic.se/projects'"; done | tee trimRescAtPath-fs0resc${i}-projects-$(date --
iso-8601=seconds).txt
climbingcatfish$ for i in {0..3}; do irule -F trimRescAtPath.r "*sourceResc='fs0resc${i}'"
"*collPath='/snic.se/'"; done | tee trimRescAtPath-fs0resc${i}-snic.se-$(date --
iso-8601=seconds).txt
```

Trim Jobs and Cleanup

```
climbingcatfish$ for resc in fs0resc{0..3}; do iquest "object count (${resc}): %s" "select
count(DATA_ID) where DATA_RESC_NAME = '${resc}'"; done
object count (fs0resc0): 102
object count (fs0resc1): 94
object count (fs0resc2): 102
object count (fs0resc3): 102
```

```
climbingcatfish$ for resc in fs0resc{0..3}; do iquest "%s/%s" "select COLL_NAME, DATA_NAME where RESC_NAME = '$
{resc}'" > objpaths-${resc}-$(date --iso-8601=seconds).txt; done
```

Thank you!

- Contact information:

  Ilari Korhonen

  Systems Manager
  PDC Center for High Performance Computing
  KTH Royal Institute of Technology
  Stockholm, Sweden

  email: ilarik@kth.se

- Questions ?