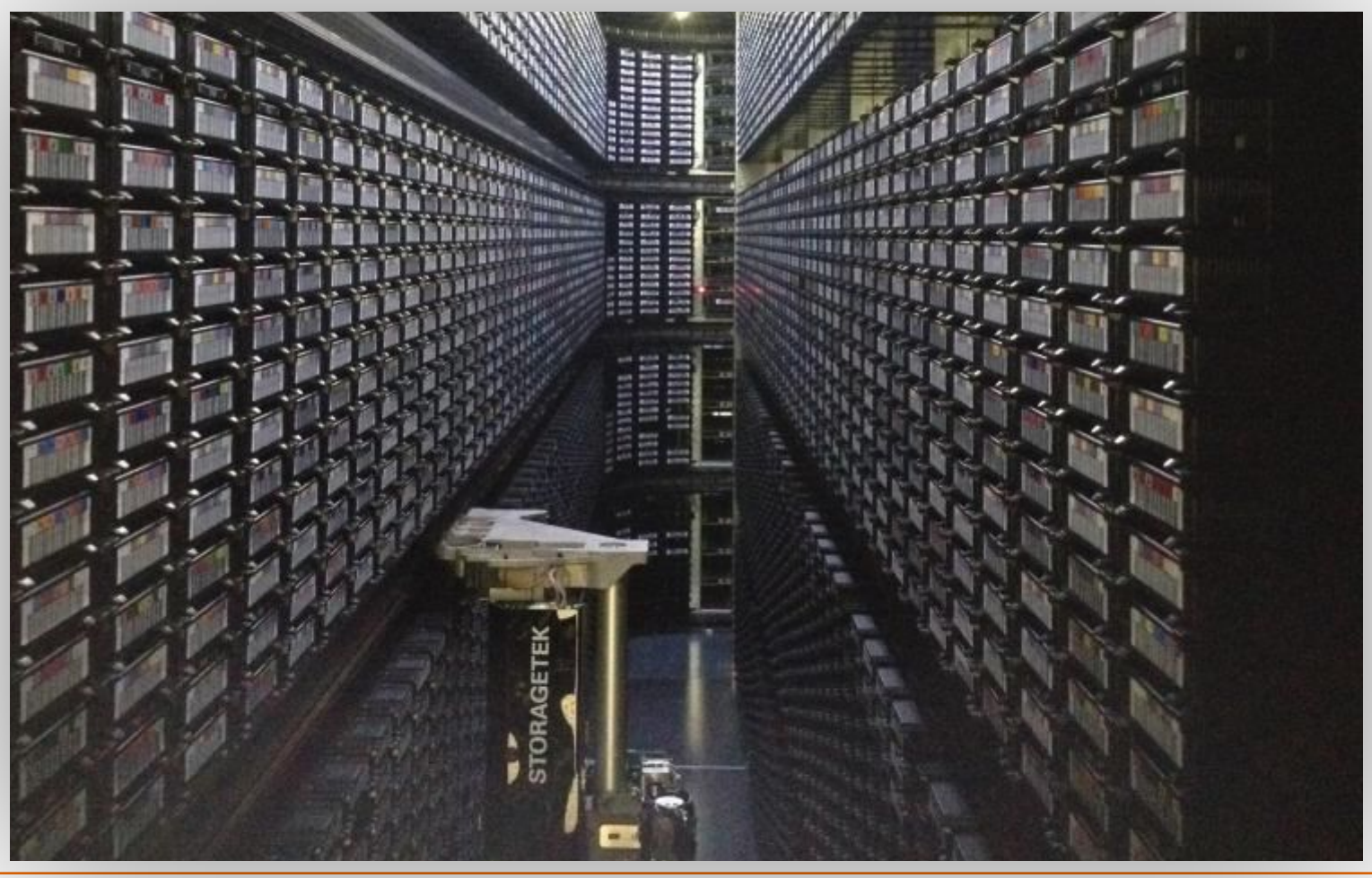
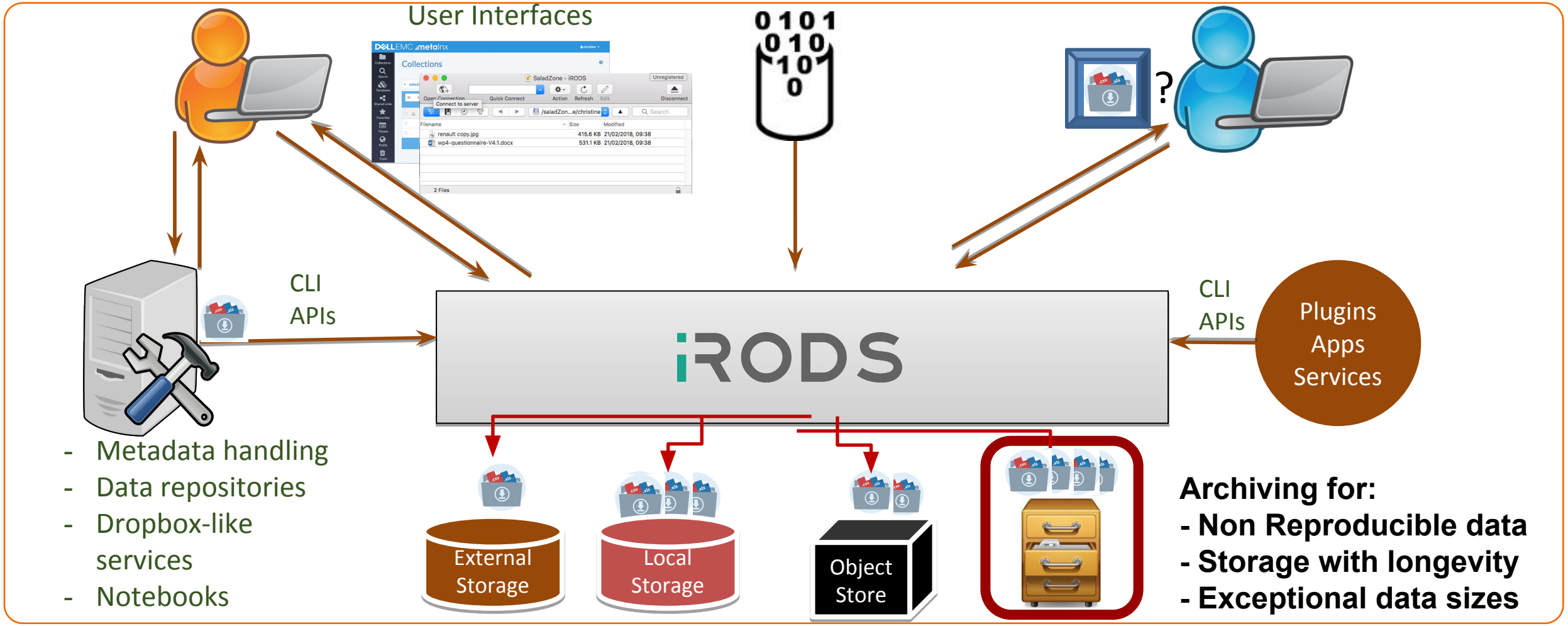
Data Archiving in iRODS



SURF

Data Management Platform

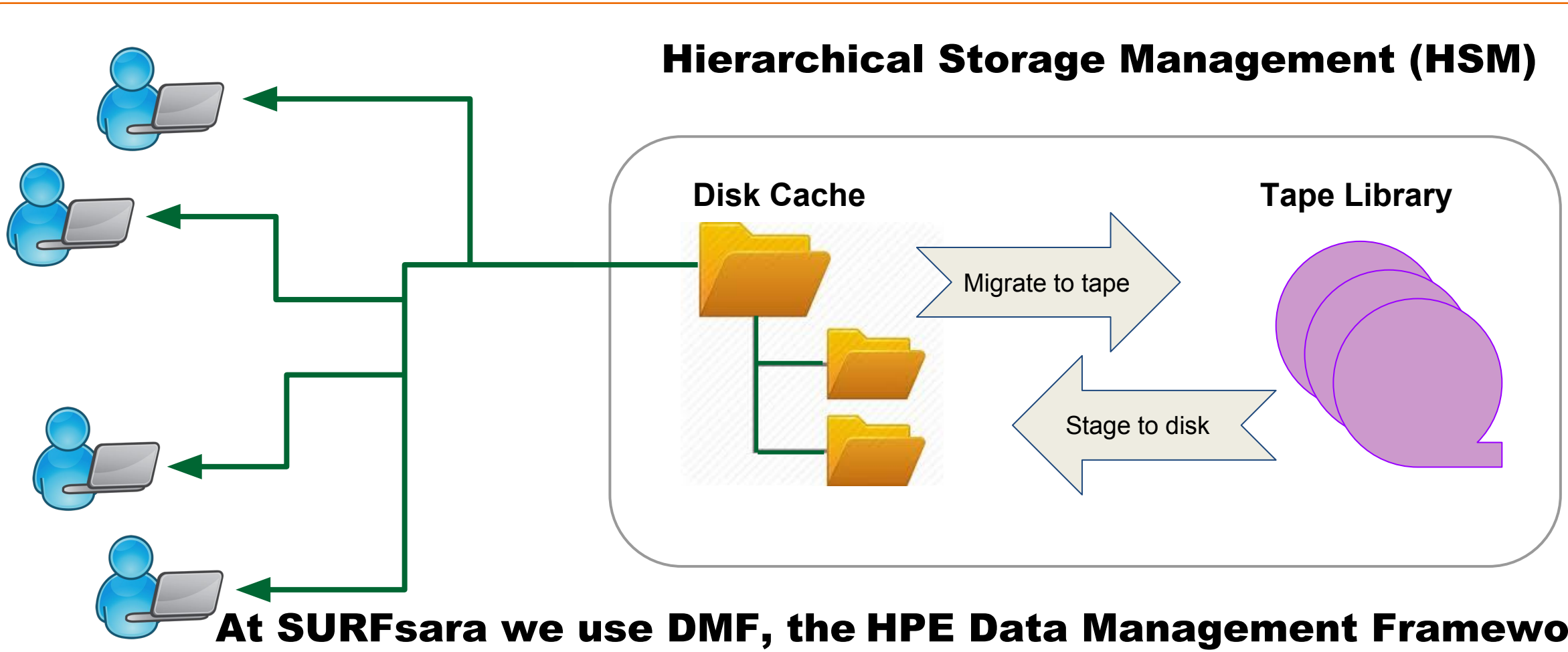


- Metadata handling
- Data repositories
- Dropbox-like services
- Notebooks

Archiving for:
- **Non Reproducible data**
- **Storage with longevity**
- **Exceptional data sizes**

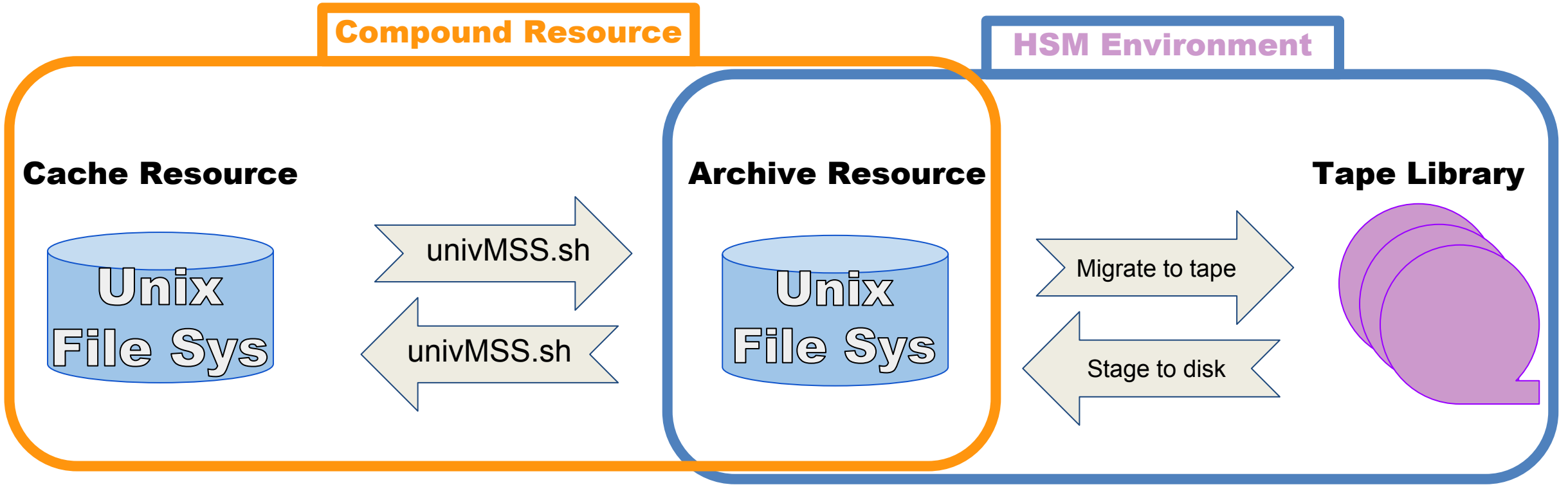
SURF

Near-Line Storage



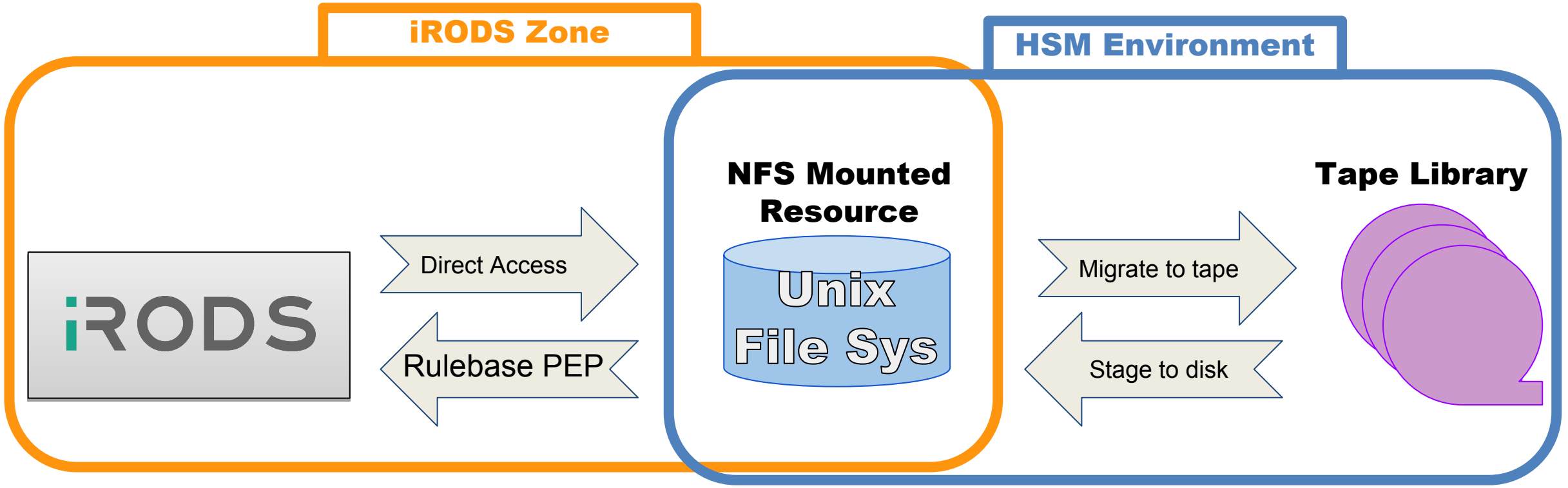
At SURFsara we use DMF, the HPE Data Management Framework

Previous Archive Connection



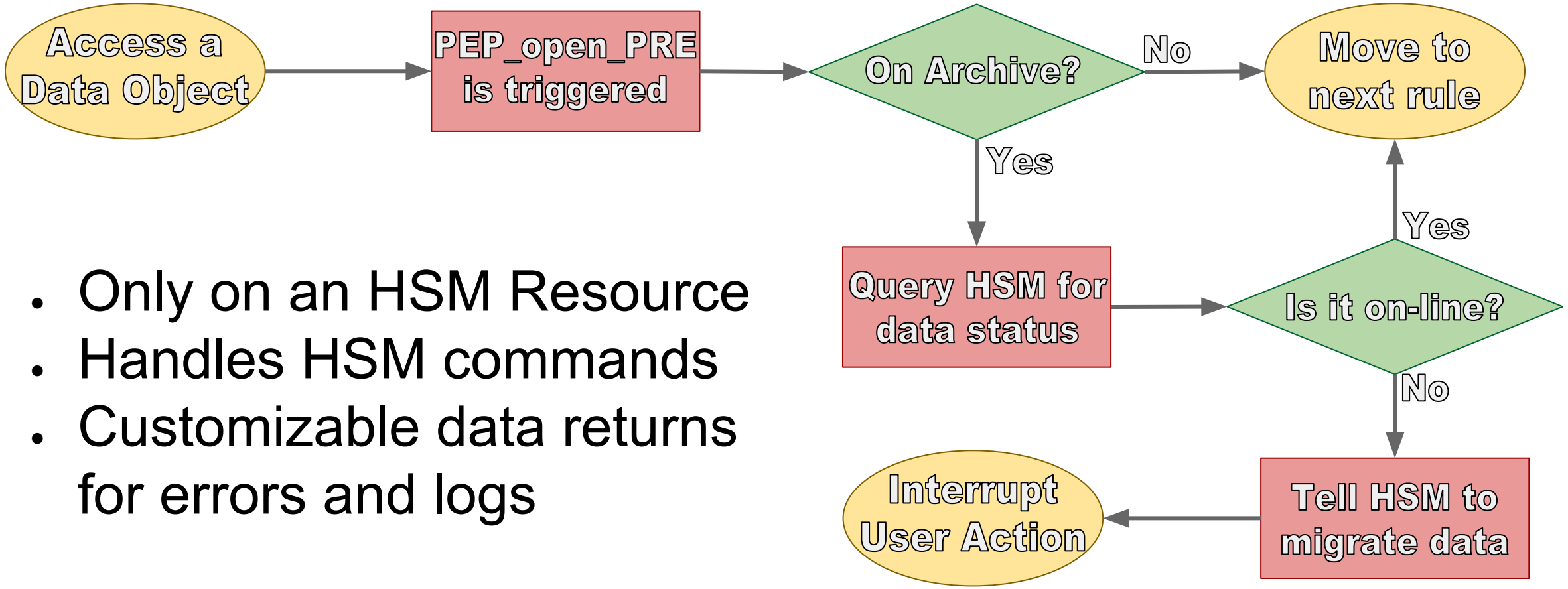
- **Communication gap: the HSM cannot talk to iRODS**
- **Extensive monitoring on iRODS cache and HSM cache**

Current Archive Connection



- **iRODS Looking directly into the Unix File System**
- **iRODS can query an HSM directly for feedback**

Rule Workflow



- Only on an HSM Resource
- Handles HSM commands
- Customizable data returns for errors and logs

SURF

Example Interruptions

Interruption for data which is offline.

```
matthews$ iget test
ERROR: getUtil: get error for ./test status = -1101000 RULE_FAILED_ERR
Level 0: DEBUG: matthews:127.0.0.1 tried to access (iget) /surf/home/matthews/test but it was not staged from tape.
```

Variant of interrupt with auto-stage enabled.

```
matthews$ iget test
ERROR: getUtil: get error for ./test status = -1101000 RULE_FAILED_ERR
Level 0: DEBUG: /surf/home/matthews/test is still on tape, but queued to be staged. Current data staged:       42%.
```

Exceptions are in place to prevent rule conflicts and to force further processing of the rule base. This makes our Archive rule base transparent to existing policy.

The Enabling Features

iRODS Aspects

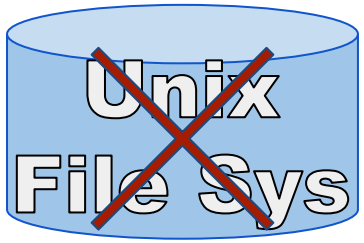
- PEP to interrupt access
- iCAT functions if bit-stream is offline

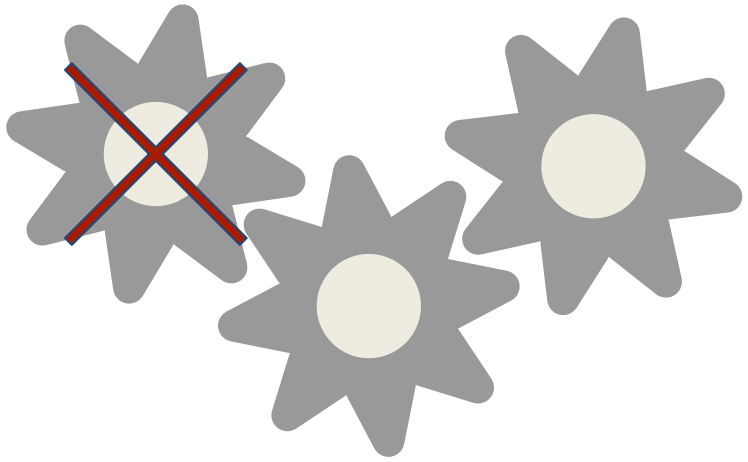
DMF Aspects

- Offline file system visibility
- Automated migration
- Separate several zone ownership rights

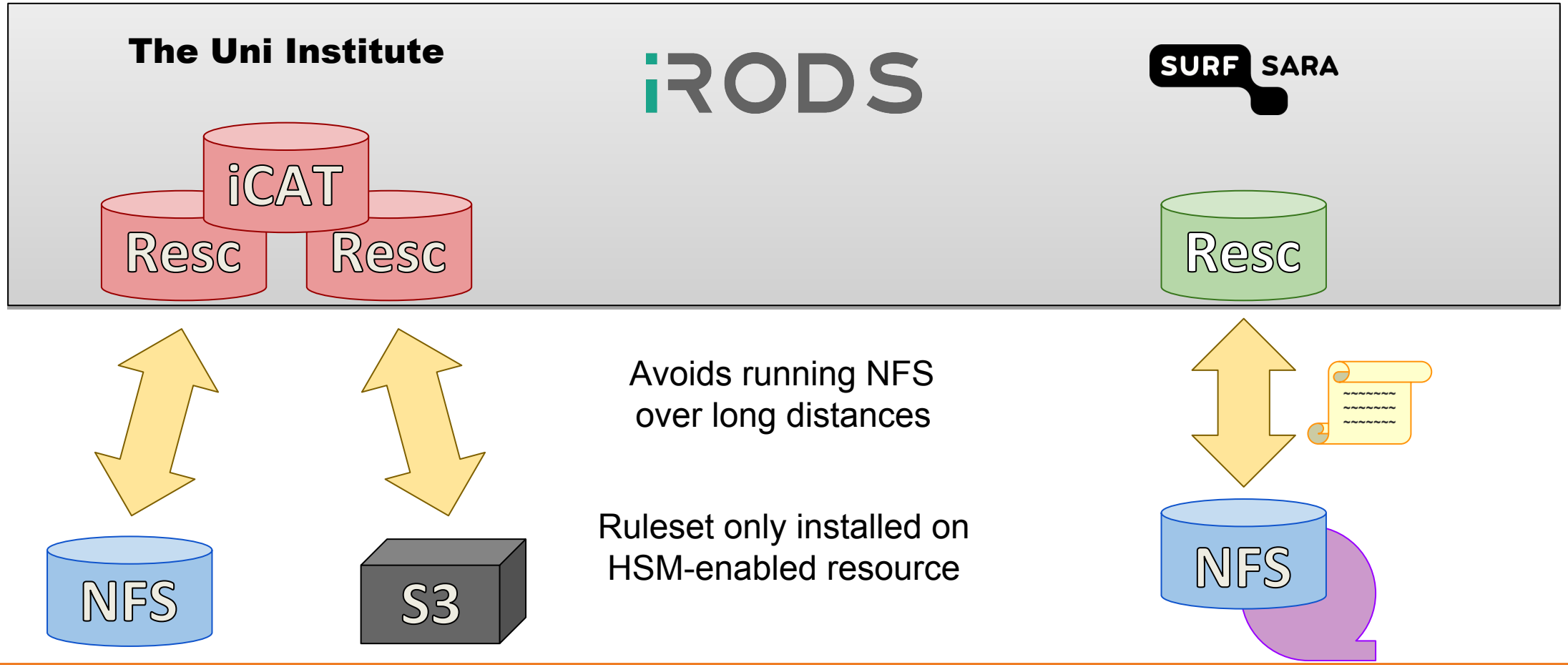
Changing from Old to New

- One less cache resource in the architecture
- iRODS can communicate directly to an HSM
- No extra script handling data movement
- Faster movement of data to tape





Example Setup



Today's Issues & Tomorrow's Plan

Ongoing Issues

- User error handling
- Direct user logon to Resc
- Tar-Ball handling

Long-term Plans

- Microservices
- Site2Site tunnel
- 4.2.x rewrite & tests

Credit Where Credit is Due

SURFsara

Sharif Islam, Arthur Newton, Jurriaan Staathof, Christine Staiger, Robert Verkerk

Maastricht Data Hub

Maarten Coonen, Paul van Schayck

And the entire iRODS Chat Group

Speaker Notes

Slide 1
I'm here on behalf of SURFsara out of Amsterdam, the Netherlands. I am a system admin, handling (currently) 10 iRODS servers.
This slide deck goes over the idea behind connecting iRODS to an archive system, in this case, a tape library. Our library is not a backup solution, but archival storage for data that is used very rarely, data that is non-reproducible and needs archived, or is too large to store conventionally.

Slide 2
A standard data-management setup. Most environments have 4+ pieces of this. I am going to focus on the Archive specifically.

Slide 3
Near-line storage is a way to store data offline while providing an automated system to restore it online. The three pieces are a front end disk cache, a tape library, and the system inbetween that handles migration. The purpose here is that actions are transparent to users, apart from requesting data be brought back online.

Slide 4
In previous attempts to handle this, we at SURFsara presented a compound resource object. This uses a Unixfilesystem disk space as a cache for actively using data. A script is then used to handle transactions between the disk cache and whatever back-end storage device is being utilized. Using S3 as the archive resource is a popular type.
However, with our archive, this created a chain of cache->cache->archive. This causes communication problems between the two systems. iRODS sends a command, but doesn't provide much feedback. The near-line storage finishes, but iRODS is not notified.

Slide 5
By using the iRODS Rule engine, we can eliminate the compound resource object. Instead, we configure the UnixFileSystem from the front-end disk cache. This is done via NFS4 in our environment, however we are toying with the idea of putting iRODS directly over the front-end cache. Within SURFsara, via the NFS4 connection, our iRODS server can write to the archive at about 450MB/s.We force the rule engine to handle all of our interaction with our archive. It will call the commands required, it will interrupt user action if data is offline, and it will monitor/log all actions.

Speaker Notes

Slide 6
Our rule flow. This begins with the access of a data object. Any object, it doesn't matter. Our rule uses the PEP_open_PRE, so any time an object bit-stream is opened, our rule is triggered. The first step is to see if our archive is involved. If not, we continue through the rulebase. If the data is on our archive, then we need to know if the data is online or offline. So iRODS will use msiExecCmd to query the HSM and get a status. Then the rule processes if the data is online or not. If the data is online, we continue through the rule base. If the data is offline, then we need to begin data migration to online and interrupt the user action. Our rule is written to not conflict with any following policy on success.

Slide 7
Our rule engine interrupt feedback. The first option is a literal interrupt. This would require another call to specify data to be staged, and iRODS would send the call to the HSM. The second block is an auto-staging rule. It shows the interrupt of a user action, as well as the output. If a user were to repeatedly run this command, they would see the percentage grow until completed. Then, it would no longer interrupt the action.

Slide 8
These are the combination of features that enable this setup to work.
In iRODS:
-Dynamic policy enforcement points allow us to handle user action before the action occurs
-With iNODE visibility, meta-data and the iCAT function perfectly. Our rule is only triggered if data-stream is accessed.
In DMF:
-In DMF, iNODEs preserve structure. This allows iRODS to "see" the directory structure and relevant information.
-Automated policies handle the disk cache management. iRODS never worries if the cache is full.
-iRODS sees nothing odd about the UnixFileSystem, it is simply an NFS4 mount point.
-Each iRODS zone gets a uniquely owned and controlled location based on service account running iRODS. This grants segregation and control of data for various iRODS instances.

Speaker Notes

Slide 9
Having one less cache disk in the architecture means that we do not have to worry as much about cache overflows. It also means we use less diskspace overall, as the two caches would at least need to be equal size. Allowing iRODS to talk directly to the HSM means it can check the status of data as needed, instead of updating all data objects at a time. Removing a script from handling data movement means less moving parts. This simplifies the path data takes, leaving less places to go wrong. Having one less resting place for data in a cache and one less script to handle transfers means data ends up on tape faster.

Slide 10
Example case, University has an iRODS instance with some NFS storage, local disk space, and maybe a block storage system. They want to add tape, so they contact SURFsara. We create a server that joins their zone via service account. we place our rule policy for handling DMF connectivity. Everything behind the scenes. For the example case, the Uni Institute runs their own iRODS environment. They have an iCAT, two resource servers with some local space, and extended it with an NFS mount and an Amazon S3 resource. But they want something capable of doing 15TB a day for long-term storage. SURFsara has a tape library. By setting up our resource server and joining it to their iRODS Zone, we can advertise our NFS4 mounted front-end disk cache as just another resource object.

Slide 11
Our solution is not without issues. We are having a hard time migrating customized error messages back to other iRODS servers. We also have the age-old problem of “do not put small files on tape”, but that is a use-case specific solution. One example we have personally completed is to tar an entire collection and subcollections. Meta-data stays the same because they apply all their meta-data at this top-level collection.
Additionally, We are bug-hunting for an odd instance of when a user is accessing iRODS through our Archive Resource server. The rule itself seems to be unable to call msiExecCmd if moving data from a resource object on another server.
We also have some forward planning in the works. We want to get away from msiExecCmd and use a microservice. We want to test with site-to-site VPN tunnels, similar to how a geographically separated office would be able to log in, and we need to finish updating things for 4.2.x. There is a working framework, but the code is in need of a scrub.

Slide 12
Credit to those involved, the team members at SURFsara. Their tape library knowledge and iRODS knowledge gave me a good boost in the right direction.
A special thank you to Maastricht Data Hub for being the guinea pigs on the first connections.
And the iRODS Chat Group, for the random issues we bumped into and the feedback provided, as well as the vast history from everyone else’s questions.