

RobinHood in Technical Computing Environments

Daniel Kobras

science + computing ag

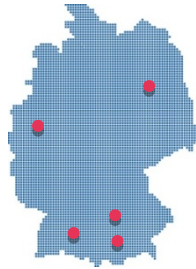
IT-Dienstleistungen und Software für anspruchsvolle Rechnernetze

Tübingen | München | Berlin | Düsseldorf

Established

1989

Offices



Tübingen
Munich
Berlin
Düsseldorf
Ingolstadt
Beijing



Employees

275

Share Holders

Atos SE (100%)

Turnaround 2013

30,70 Mio. Euro

Portfolio

IT Services for complex computing environments
Comprehensive solutions for Linux- und Windows-based **HPC**
scVENUS system management software for efficient
administration of homogeous and heterogeneous networks

Environments

What we work with

- Many flavours of technical computing environments
 - HPC clusters (Linux)
 - Pre/postprocessing servers (Linux)
 - Workstations (Windows/Linux)
 - Fileservers (Lustre, GPFS, Appliances, homebrew Linux NFS+Samba)
- Storage historically evolved
 - Past: scattered across many individual fileservers
 - Later: aimed to consolidate on central (scale-out) storage
 - Now: scattered across many scale-out storage systems
- Self-censorship: Admins avoid long-running tasks on large FSes
 - hard to stay informed about FS usage patterns

What we use RobinHood for

- RobinHood for us is:
 - **A looking glass into the usage of large filesystems, ie.**
 - A fast and convenient database for filesystem metadata
 - An efficient alternative for du/find on large filesystems
 - A near-online backup/replica for Lustre-MDTs
 - Occasionally even: a tmpfs manager
- RobinHood for us might become:
 - A core ingredient for multi-site storage replication (active/passive)
 - An HSM system (linked to TSM)
 - A federated metadata engine spanning multiple filesystems

Past and Current Use Cases

Flexible „Soft Quota“

- Don't force hard quotas upon users
 - Still keep an eye on space usage
 - More flexible than real quota:
 - Per user/group
 - Per subtree
 - Per file type
 - Combinations thereof
 - Independent of quota support in underlying filesystem
- essentially just `find/du -s`
- except that you wouldn't run `find/du -s` because it's too slow

Detect abnormal usage patterns

- Variant of the „soft quota“ theme
 - Monitor daily changes in space/inode usage for specific subtrees
 - Alert on abnormal changes, eg.
 - Involuntary removal of larger portions of FS (rbh-du)
 - Exceptionally large files or directories (alert trigger)
 - File names containing common problematic characters (alert trigger)
- If we can't avoid the troublemakers, we at least want to know about them as soon as possible
- **More and more use cases pop up in day to day administration once it's cheap and easy to implement them**

Scratch cleaner

- Out-of-the-box functionality of RobinHood's tmpfsmgr
- Special requirements by customer
- Couldn't be expressed in terms of purge policies
- Use RobinHood for scanning and population of DB
- Use custom Python script to implement policy, DB queries, and actual purging

One Tool To Find Them

- Several types of filesystems in use:
 - Lustre (1.8/2.5)
 - GPFS
 - PanFS
 - (automounted) NFS
- RobinHood provides consistent interface across all filesystems
- Easy to adjust existing scripts (du -> rbh-du, find -> rbh-find)
- No stability issues (barring FS/interconnect instabilities)
- Typical scan speeds:
 - ~350 entries/sec (PanFS)
 - ~1000 entries/sec (NFS over IPoIB)
 - ~3,000-5,000 entries/sec (Lustre/GPFS)

Future Use Cases

Project Permissions

- Project directories require pre-defined set of permissions
- Partly enforced by configuration/ACLs
- Occasionally corrupted by pilot errors
- Should be continually fixed by automated script
- Requires RobinHood to support mode/perms
- Already possible with custom DB queries
- Not yet supported in rbh-find

Replicated Storage System

- „Poor man's Metro Cluster“ for large datasets
- Low(ish) fill rate but high overall volume of data
- Traditional restore from backup doesn't meet recovery time objective
- Two Lustre systems at different data centres
- Long-range IB interconnect
- Activate Lustre changelogs
- Use RobinHood Backup for continuous (async) replication from primary to secondary system
- In case of disaster, switch primary/secondary relationship
 - Minimize data loss in case of disaster
 - Fast return to operation

Federated Filesystem Database

- Run queries across all filesystems at site
- eg. find all filesystem objects owned by former employee when account is locked down
- Think:
rbh-find -f '*' -user robin
gpfs01:/home/robin/foo.txt
lustre1:/proj1/bar.sim
netapp2:/scr/companysecret.docx
clusternode23:/tmp/hoverboard.jt
- Optimal design unclear:
 - Central DB server: facilitates queries
 - Multiple DB servers: avoids performance bottleneck

- Looking glass functionality: Makes it easy to know what's going on in your FSES
- Cheap and flexible metadata queries: Allows you to perform operations you otherwise wouldn't dare to do on a large FS
- Filesystem agnostic: Provides consistent interface across arbitrary filesystems, easy to use and adapt
- Backup and HSM functionality: not yet, but will follow soon
- Nice to have (aka things I should send a patch for):
 - Requires: lustre-modules | lustre-client-modules
 - More comprehensive support of find options
 - Choose scheduler of scan runs:
 - Start x hours after end of last scan (current behaviour)
 - Start every x hours (unless still running)

Vielen Dank für Ihre Aufmerksamkeit.

Daniel Kobras

science + computing ag

www.science-computing.de

Telefon 07071 9457-0

info@science-computing.de