MetaCentrum

DMlite VFS plugin

František Dvořák

CESNET

DPM Workshop 2013, Edinburgh

Historic overview

- Motivation
  - easy deployment
  - directly use various filesystems
  - performance
- Initial version
  - written by Alejandro Alvarez Ayllon
  - "skeleton" VFS plugin
  - full read-only functionality
- CESNET's proposal of EGI mini-project
  - "Enabling and Integrating Further Storage Resources in EGI"
  - rejected in EGI
  - participate anyway – funded by NGI-CZ (CESNET)

Work done

- Catalog (namespace)
  - . . . next slides
- Authn (user and groups management)
  - text files, in "csv" format
  - supports additional attributes
  - attributes named *uid*, *gid* as *uid_t*, *git_t* type
  - other attributes as string type
- PoolDriver/Handler, IODriver/Handler, PoolManager
  - write support
  - one static pool
  - only basic functionality, not thoroughly tested
  - any requirements? (add/remove pools, what would it mean?, . . .)
  - cleanups needed (separate plugin parts, missing pieces, . . .)
- Other stuff
  - configurable local prefixes (*NSPrefix*, *DiskPrefix*)
  - global authz in dmlite.conf (regular expressions, *AllowRead*, *AllowWrite*)
  - debugging, syslog

Implementation

- extra metadata needed
  - run under apache (dpmmgr user)
  - no extra privileges for VFS plugin
  - cannot use owner, permissions and size directly on filesystem
- based on User Extended Attributes
  - SL5 not supported
  - SL6 and ext3: user_xattr mount option needed
  - symbolic links cannot have user xattrs
    - metadata from their target or dummy values
    - only root (uid 0) can stat dangling links
  - attr_set() vs fsetxattr() APIs
- using locally mounted FS
  - noatime mount option vs readDir()/readDirx() calls
  - different meaning of st_nlink returned from stat()

Implementation(2)

- access by inode numbers
  - hard to search files by inode numbers on filesystems
  - no extra fileid catalog
  - LRU cache (last 200 extendedStat()/readDirx() calls)

```
std::map<ino_t,std::string> inodes;
std::list<ino_t> inodes_lru;
```

- access by Replica File Names
  - hard to search catalog entry tree for RFN
  - additional directory tree (*NSPrefix*/.#vfs.replicas/)
  - mapping from RFN entries to catalog entries
  - information about replicas remain in catalog entries (RFN, status, type, ...)

Tests

- dmlite tests: success rate 75%
  - 2 fails: st_nlink problem
  - 3 fails: disk support (TODO: unfinished pools, I/O driver)
  - 1 fail: access to "/" (TODO)
  - 1 fail: missing INode interface
- WebDAV
  - download
  - upload
  - experiments with separated NS/disk (temporarily hardcoded disk node hostname in VFS plugin)
- handmade scan utility ("recursive upload")
  - "recursive walk" mirroring local FS to namespace
  - wild local data (dangling links, various permissions, . . .)

Benchmarks

- "recursive upload" catalog benchmark
  - 398 files/sec (one disk), 423 files/sec (two disks)
  - 37x (one disk), 32x (two disks) slower than `rm -rf`
  - no replicas, only create(), setSize(), setUtime(), makeDir()
  - CPU between 90-99%
- each operation performs permissions check
  - from root toward deeper level
  - user xattrs: owner, unix perms, acl

Questions and Discussion

Questions?