The Management of Shares

Tom Haynes
tdh@sun.com
Sun Microsystems, Inc.

Doug McCallum
Doug.McCallum@sun.com
Sun Microsystems, Inc.

Problems

- Scalability
- Management

Scalability – Root of All Evil

- share(1M) and sharetab(4)
  - > Designed for small number of shares
    - >Can not share subdirs
    - >Disks were rare
  - > Actions were rare
    - >Shares occur at boot
    - >Sharetab accessed on mounts

/etc/dfs/sharetab

- Needed to be persistent
- Kernel memory was tight
- Let's put in /etc!
- *By the way, it should really be read-only.*
  - We have cases where customers modify it, expecting shares to come online

Management – Designed for NFS

```
$ more /etc/dfs/fstypes
nfs NFS Utilities
autofs AUTOFS Utilities
cachefs CACHEFS Utilities
```

- autofs and cachefs are no-ops
- Native CIFS implementation is on the way
- ZFS shares not stored in /etc/dfs/dfstab

sharemgr
Improved file share management

- Introduces concept of *share groups*
- Integration with SMF
- Extensible via plugin modules
- Fully scriptable CLI-based management
- CLI-based command for managing protocol separate from shares

Share groups

- Named groups hold collection of shares
- Configuration at the group level
  - common configuration properties
  - enable/disable by group
- Share level override of properties
- Group “default” for backward compatibility
- Group “zfs” provides handle to ZFS managed shares

Integration with SMF

- Each group is implemented as an SMF *service instance*
  - instances can start/stop in parallel
  - configuration properties stored in SMF repository
  - new service instance created for each group
- ZFS shares started via SMF but configuration is stored in ZFS properties
  - dataset with a “sharenfs” property appears as sub-group
  - there is a single “zfs” service instance
- Protocol services *depend* on the group services

Share groups and adding shares

```
$ sharemgr create homedirs
$ sharemgr set -P nfs -p nosuid=true homedirs
$ sharemgr add-share -s /export/home/john homedirs
$ sharemgr add-share -s /export/home/bill homedirs
$ sharemgr show -vp homedirs
homedirs nfs=(nosuid=true)
          /export/home/john
          /export/home/bill
$ svcs group
online    13:00:09 svc:/network/shares/group:zfs
online    13:00:17 svc:/network/shares/group:default
online    14:05:04 svc:/network/shares/group:homedirs
```

Adding security to “homedirs”

```
$ sharemgr set -P nfs -S sys -p rw=“*” ro=rohost homedirs
$ sharemgr show -vp homedirs
  homedirs nfs=(nosuid=true) nfs:sys=(rw=* ro=rohost)
                   /export/home/john
                   /export/home/bill
$ share
  -@homedirs /export/home/john sec=sys,rw,ro=rohost,nosuid “”
  -@homedirs /export/home/bill sec=sys,rw,ro=rohost,nosuid “”
```

Old school still works

```
$ share -F nfs -o sec=sys,rw,ro=rohost /data
$ share
  -@homedirs /export/home/john sec=sys,rw,ro=rohost,nosuid “”
  -@homedirs /export/home/bill sec=sys,rw,ro=rohost,nosuid “”
  - /data rw “”
$ sharemgr show -vp
  default nfs=()
          /data nfs=() nfs:sys=(rw=* ro=rohost)
  homedirs nfs=(nosuid=true) nfs:sys=(rw=* ro=rohost)
          /export/home/john
          /export/home/bill
```

ZFS shares (assume pool “data”)

```
$ zfs create data/dirs
$ zfs create data/dirs/user1
$ zfs create data/dirs/user2
$ zfs set sharenfs=on data/dirs
$ sharemgr show zfs
  zfs
      data/dirs         nfs=()
          /data/dirs
          /data/dirs/user1
          /data/dirs/user2
```

Disabling/Enabling a Share Group

```
$ sharemgr list -v
  default enabled    nfs
  zfs     enabled    nfs
  homedirs enabled   nfs
$ sharemgr disable homedirs
$ sharemgr list -v
  default enabled    nfs
  zfs     enabled    nfs
  homedirs disabled  nfs
$ sharemgr enable homedirs
  sharemgr list -v
  default enabled    nfs
  zfs     enabled    nfs
  homedirs enabled   nfs
```

Future of sharemgr

- Tighter Integration with ZFS
  - ZFS will use sharemgr API
- CIFS Server Integration
  - CIFS protocol plugin is being prototyped
  - sharemgr API needs enhancements to fully support
- Can not process share groups in parallel
  - File lock on /etc/dfs/sharetab
  - We'll fix this one by the end of the presentation.

Case Study – unshareall(1M)

```
$ file `which unshareall`
/usr/sbin/unshareall:   executable /sbin/sh script
```

- Removes all shares from the sharetab

Old implementation

```
59 for i in $fsys
60 do
61        for path in `sed -n "s/^\([^    ]*\)[   ]*[^   ]*[     ]*${i}.*/\1/p" /etc/dfs/sharetab`
62        do
63                /usr/sbin/unshare -F $i $path
64        done
65 done
```

Cost

- Currently reads the sharetab 3 times
  - Due to the sed run against $fsys
- Causes N forks
- Causes the sharetab to be read N times
- Has to rewrite the sharetab N times

With sharemgr

```
50  if [ "$fsys" ]          # for each file system ...
51  then
52          fsys=`echo $fsys|tr ',' ' '`
53          for i in $fsys
54          do
55                  /usr/sbin/sharemgr stop -P $fsys -a
56          done
57  else                    # for every file system ...
58          /usr/sbin/sharemgr stop -a
59  fi
```

Cost

- Does not read the sharetab per fstype
- Can cause 3 forks or just 1
- Causes the sharetab to be read N times
- Has to rewrite the sharetab N times

What if we got the sharetab off disk?

- In memory, no need to read/rewrite N times

Scalability: shares in the wild

- ZFS testing is driving larger share sizes
- Numbers of shares on jurassic
  - 12 shares before ZFS
  - 300 shares with first introduction
  - 1300 shares a week later

In Kernel Sharetab
Improved share storage

- Scalability
  - Want to kick off share groups in parallel
  - Do not want to hit disk to authenticate a NFS request
- Portability
  - Want a solution for CIFS
- Ownership
  - Kernel should own shares

Design Considerations

- Sharetab has to be persistent when power is on
  - Used to be nuked on boot
  - Can't put sharetab in user space of mountd
    - It can be restarted
- Do not want it to be protocol specific
  - Mountd connotates NFSv2/3
- No clue what 3rd party applications are doing with /etc/dfs/sharetab

More Design Considerations

- ZFS wants to delegate filesystem creation to non-root users
  - Security
    - ZFS uses ACLs for security, not RBAC
    - Really do not want to use setuid scripts/programs
  - Regular file is owned by root
- Want to remove file locks on /etc/dfs/sharetab

Pseudo-FS implementation

- Store the shares in the kernel
  - Hash tables on path name
- Create a new module sharefs
- Sharemgr is the only application allowed to write
  - ZFS has to call into sharemgr
  - Sharemgr uses a syscall to pass shares
- Readers access a psuedo-fs: /etc/dfs/sharetab

GFS for code reuse

- ZFS, objfs, and ctfs use gfs
- Abstract framework for pseudo-fs
  - vfs and vnode ops call into gfs
  - gfs handles generic tasks
  - Calls into code specific routines

uts/common/fs/gfs.c

uts/common/sys/gfs.h

Future work

- Get share groups into memory
  - Large host lists not shared
- API to get shares
  - Extend sharemgr to get shares from kernel
  - File I/O is sole *published* means of access
- Callbacks to inform consumers of changes
  - Currently, consumers periodically stat the file

Questions

Tom Haynes
tdh@sun.com