Linux Kernel Evolution

vs

OpenAFS

Marc Dionne

Edinburgh - 2012

The stage

- Linux is widely deployed as an OpenAFS client platform
- Many large OpenAFS sites rely heavily on Linux on both servers and clients
- The OpenAFS Linux client includes a kernel module
  - Sensitive to kernel changes

The battle

- Linux perspective
  - All useful drivers and modules are in-tree, or should be in the tree
  - Changing the module API/ABI is not a problem – in-tree code is adapted as part of the change
- OpenAFS perspective
  - Can't join the party – incompatible license
  - Must adapt on its own, can't benefit from kernel developers
  - Can't have all the goodies - part of the API is out of reach

Since Oct 2006

- **28** kernel releases (2.6.19 – 3.6)
  - 292 876 commits
  - 17 216 184 lines changed in 49 983 files
- Estimate of > 100 OpenAFS commits linked directly or closely to kernel changes
- Kernel releases with no impact on OpenAFS:

0

Linux development process

- Fast
  - New release every ~3 months
  - No fixed schedule, released when it's ready
  - .. but fairly consistent
- Fast moving
  - Thousands of commits per release
  - Tens of thousands of lines of code changed
- Big
  - Close to 1000 developers involved in each release
  - Heavy corporate participation

The code

- Linux releases are cut directly from “mainline” - master branch of Linus' tree
- 2 week merge window per cycle
  - Followed by ~10 weeks of fixes and stabilization over 6-9 RC releases
- Stable releases are handled by separate maintainers, in separate trees
  - Many active stable releases in parallel
  - Some releases are tagged as long term

linux-next

- Tree for integration testing
- Contains code targeted for next release cycle
  - Most, but not all subsystems
- Rebuilt from scratch daily – expensive to follow
- Not all code in -next will make it to mainline in the following cycle
- Not all code will show up in -next before hitting mainline

How we try to keep up

- Continuously run kernels very close to mainline
- Follow linux-kernel, linux-fsdevel discussions and patches
  - particular attention to vfs layer
  - .. and other related lists
- Frequent builds and tests of current OpenAFS master

How we try to keep up

- Keep an eye out for new warnings
  - Often a symptom of an API/ABI change
- Do real testing
  - Not all changes can be detected at compile time
- Keep an eye on the VFS tree
- Occasional test of linux-next

The result

- OpenAFS master supports most Linux kernel releases before they're released
  - Usually early in the RC cycle
- But stable releases are a challenge
  - There's a speed mismatch
- .. and getting these changes to distributions is also challenge
  - Schedules are not in sync
  - Many have custom patches or packaging

The fixes

- Some fixes are mostly mechanical
- Typical case :
  - A new configure test to identify a new behaviour
  - Conditional code (ifdefs) to do things the new way
  - In some cases, new compatibility helpers to hide the ifdef maze
- Even when the fix is trivial, it may need a lot of packaging
- Unfortunately many changes require more analysis

Challenges

- VFS changes are often merged late in the cycle
  - Better lately
- Many VFS changes appear in mainline with little notice
- Compatibility with older releases
  - Risk of breaking support for an older kernel
  - Impossible to test everything
  - Use mitigating strategies for configure tests
- Sprawling feature tests
  - make -j 16 all = 14.7s
  - ./configure = 80.7s

More challenges

- Keeping the code manageable and readable
  - Keep ifdef jungle under control
- Distributions
  - Have their own schedule, packaging, custom patches, bug reporting, maintainers
- Shrinking API
  - Many useful debug features are off limits – ex: lockdep
  - Can't support RT kernel, Fedora rawhide, etc
  - So far core functionality has been spared

Highlights

Syscall table

- OpenAFS relied on modifying the syscall table to hook the setgroups call and preserve PAGs
- In the early 2.6 kernels, the syscall table was unexported and made read-only
- The new “keyring” feature is now used to implement PAGs internally
- Special PAG groups are still set for legacy reasons – they are no longer used to determine PAG membership

Inode abstraction

- Client keeps references to disk cache files so it can quickly open them as needed
- Traditional reference on Unix systems was the inode number
- On Linux, some filesystems can't guarantee stable inode numbers
  - Problem reports (xfs, reiserfs) led to filesystem restrictions in afsd (ext2/3)
- Linux 2.6.25: the API to open a file by inode number is no longer available

Inode abstraction

- Solution: exportfs interface
  - Linux API to get a stable opaque file handle from the filesystem, and later use it to open the file
  - Used by NFSD – supported by all exportable filesystems
- Implemented progressively
  - Minimal change in 1.4 to deal with 2.6.25; create our own inode number based handles for ext2/3
  - Later, call filesystems to generate handles
  - Finally, extend method to pre-2.6.25 kernels
- Side benefit: any exportable filesystem can now be used

Linux 3.0

- Numbering change – no major new feature
- Impact limited to the build system, packaging
- Some discussion about default sysname values

Credentials

- Internal kernel handling of security credentials has evolved
  - Separate structure with a pointer in the task struct
  - RCU based change mechanism
  - Support for new security subsystems – selinux, etc.
- OpenAFS changes
  - Use the new cred structure directly, instead of rolling our own
  - Open cache files with the initial cache manager credentials – resolves issues for systems with selinux and AppArmor

aklog -setpag

- Stopped working at some point – a process was not allowed to change its parent's credentials
- .. but a new syscall now allows a process to set a keyring in its parent
- Currently works for recent kernels

BKL

- “Big Kernel Lock” - global kernel wide lock
- Gradually replaced by more granular locking, RCU
- Last bits removed in kernel 2.6.39
- By that time, OpenAFS master was mostly BKL free
  - .. but making 1.4 safe for BKL removal would have been invasive
  - EOL for new kernel support in 1.4

RCU based path walking

- Major VFS change to reduce lock contention by relying on RCU where possible
- Requires that several VFS callbacks don't sleep
  - But most OpenAFS callbacks take the global lock (GLOCK), and can sleep
- Fallback mechanism
  - filesystems can indicate that they don't support RCU path walking
  - VFS calls back with locks taken
- Significant locking changes (ex: no more dcache_lock)

RCU path walking

- For OpenAFS
  - Return appropriate error codes to trigger the fallback to locking mode
  - Rework locking
  - Resulted in a few hard to diagnose bugs where some configure tests caused the VFS to think we supported RCU mode

IMA

- Integrity Measurement Architecture, activated in Fedora and Red Hat Enterprise kernels
- Hooks into file opens and closes, issues warning for close with no corresponding open
- API was unbalanced
  - Close implicitely called IMA
  - Caller had to call IMA for some opens – ex: dentry_open used by OpenAFS
  - But... IMA calls are GPL only and not accessible to OpenAFS
- Bottom line: impossible to use the API correctly and avoid the flood of syslog warnings

IMA

- All (eventually) ended well
  - API reworked in kernel mainline
  - Backported in time for RHEL 6 release, with customer pressure
  - Affected Fedora reached EOL

Exportfs API

- OpenAFS relies on this API for two uses
  - Tracking and opening disk cache files
  - Exporting AFS files via the NFS translator
- Many revisions to this API over the past few years, some major
- Translator no longer supported – requires GPL only symbols

Looming changes

- vmtruncate
- Kernel and module signing, secure boot
- ...

As of today..

- 3.4 support in official 1.6.1 release
- 3.5 and 3.6 support in master and 1.6 branch
- 3.7 currently still in merge window
- 3.7 RC1 imminent
- 3.7 support looking good
- until...

```
Commit:     8e377d15078a501c4da98471f56396343c407d92
Author:     Jeff Layton <jlayton@redhat.com>

    vfs: unexport getname and putname symbols
    I see no callers in module code.
---
 fs/namei.c |    2 --
 1 files changed, 0 insertions(+), 2 deletions(-)

diff --git a/fs/namei.c b/fs/namei.c
index ca14d84..9cc0fce 100644
--- a/fs/namei.c
+++ b/fs/namei.c
@@ -163,7 +163,6 @@ void putname(const char *name)
        else
                __putname(name);
 }
-EXPORT_SYMBOL(putname);
 #endif

 static int check_acl(struct inode *inode, int mask)
@@ -3964,7 +3963,6 @@ EXPORT_SYMBOL(follow_down_one);
 EXPORT_SYMBOL(follow_down);
 EXPORT_SYMBOL(follow_up);
 EXPORT_SYMBOL(get_write_access); /* nfsd */
-EXPORT_SYMBOL(getname);
 EXPORT_SYMBOL(lock_rename);
```

Thanks!