POSIX file locks

- Migration and failover when each file system is exported by exactly one cluster node

  Bruce merged two patches from Wendy Cheng (after some review and bug fixing) that help with migration.

  Part of the process of migrating a file system is migration of lock state from the source server to the target server. One element of this is the removal of locks on the source server. These two patches provide administrative interfaces

  ```
  /proc/fs/nfsd/unlock_ip
  ```

  and

  ```
  /proc/fs/nfsd/unlock_filesystem
  ```

  for lock removal.

  The sequence of events for migration would be

  - Tear down the IP address
  - Unexport the path
  - Write the IP address to `/proc/fs/nfsd/unlock_ip` to unlock any files locked through the given address, e.g.,

    ```
    $ echo 10.1.1.2 > /proc/fs/nfsd/unlock_ip
    ```

  - Signal target server to begin takeover.

  If the source file system must be unmounted but locks are held from other IP addresses, they can be forcibly removed by writing the mount point to `/proc/fs/nfsd/unlock_filesystem, e.g.,`

  ```
  $ echo /mnt/sfs1 > /proc/fs/nfsd/unlock_filesystem
  ```

- Migration and failover when file systems are exported by multiple cluster nodes

  The current scheme for lock migration simply drops locks on the source server and reacquires them on the target server. Bruce is working on a more ambitious mechanism for client migration.

  Suppose we want to migrate NFS service for a file system from a source server to a target server. Most often, this is accomplished by using the server IP address as an abstraction layer: at first, the IP address points to the source server, later it points to the target server.

  The easy way to migrate locks from the source server to the target server is to use existing NFS reboot recovery mechanisms. With this approach, the target server simply informs clients that is has rebooted. Clients then attempt to reclaim their locks on the target server.

  The target server then enters a "grace period" during which it grants valid lock reclaim requests but refuses all others. Once the grace period has ended, subsequent reclaim requests are refused, lest clients believe that they have held locks continuously, when in actuality conflicting locks may have been granted and released.

  To use reboot recovery for lock migration, we need a way to tell the target server to enter a grace period for the clients migrating from the source server. These clients are easily identifiable: they access the server through the floating IP address.

  This suggests an interface like

  ```
  $ echo 192.168.0.1 /path/to/export > /proc/fs/nfsd/resume_ip
  ```

  Note that the grace period is restricted to an IP address and a file system, so that file systems unaffected by the migration are not put into grace. We also restrict the server to grant reclaims in the given file system only to clients using the given IP address.

- lockd cleanup and multithreading

  Nothing to report.

Leases

- Extend and fix problems with existing lease implementation.

  We fixed a number of lease bugs:

  - Fixed set/break lease race causes by an alloc-might-block-under-BKL bug
  - Fixed a race where a new writer could open a file with outstanding read leases without breaking those leases
  - Added generic per-inode lease enabling/disabling
  - Added support for directory leases
  - Added VFS-level support for breaking both lease types (as appropriate) on: unlink, rename, chmod, chown, creat, mknod, mkdir, symlink, link, and rmdir
  - Due to locking issues and/or because nfsd doesn't want to block while leases are broken (unlike local processes, which must block), added NFS-level support for breaking leases on the NFS operations analogous to those listed above (unlink, rename, etc).

  With these patches, NFSv4 delegations can—for the first time—support RFC-mandated semantics, even in the presence of local applications using the same file system. The patches are under review.

- Implement leases in a Linux cluster filesystem.
- Implement write delegations.
- Implement cluster-coherent write delegations.

  Nothing to report.

RPC reply caching

- Implement an RPC reply cache that persists across reboots.
- Extend state migration to support RPC reply cache migration.
- Implement the NFSv4.1 RPC reply cache using sessions.
- Implement the NFSv4.1 RPC reply cache using persistent sessions.

  Nothing to report.

CIFS/NFS integration

- Ensure the Linux lease implementation meets Samba's needs.
- Implement native NFSv4 ACL support.

  Nothing to report