Linux kernel client - Feature #23

fcntl/flock advisory lock support

04/13/2010 09:45 AM - Sage Weil

Status:	Resolved	**% Done:**	0%
Priority:	Normal	**Spent time:**	5.00 hours
Assignee:	Greg Farnum		
Category:			
Target version:	v2.6.36		
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:			

Description

History

#1 - 04/16/2010 04:10 PM - Greg Farnum

- Assignee set to Greg Farnum

#2 - 04/19/2010 10:26 AM - Greg Farnum

- Status changed from New to In Progress

- Start date changed from 04/13/2010 to 04/19/2010

#3 - 04/20/2010 01:35 PM - Greg Farnum

All right, designed a basic interface between the client and the MDS. Going to implement the MDS and messaging parts for an MDS-only solution.

#4 - 05/17/2010 03:43 PM - Greg Farnum

- Status changed from In Progress to Resolved

It should support flock and fcntl locks now. Currently there are no caps for this, so all locking requests are routed through the MDS. It's been tested using the xfstests locktests test, with some additions. It appears to recover properly from a dead MDS.

flock locks haven't been tested, though, since I couldn't find a good test for them. Since the infrastructure is the same as for fcntl locks, though, they should be good to go.

The userspace stuff is in branch "locks" and the kernel client is in branch "flock".

#5 - 05/18/2010 10:03 AM - Greg Farnum

- Status changed from Resolved to In Progress

Found some issues with recovery after all; working on them now.

#6 - 05/18/2010 11:49 AM - Greg Farnum

- Status changed from In Progress to 4

Ahah, file_lock's fl_nspid pointer isn't filled in before calling the filesystem's lock handlers. I've fixed that so it is now.

#7 - 05/21/2010 03:40 PM - Sage Weil

- Target version changed from v2.6.35 to v2.6.36

#8 - 06/23/2010 01:24 PM - Sage Weil

BTW these should all be __le32 etc. if the values go over the wire. And the kclient code that uses them needs to use cpu_to_le##, le##_to_cpu (userspace does the endian conversion magically).

```
+       struct {
+               __u8 rule; /* currently fcntl or flock */
+               __u8 type; /* shared, exclusive, remove*/
+               __u64 pid; /* process id requesting the lock */
+               __u64 pid_namespace;
+               __u64 start; /* initial location to lock */
+               __u64 length; /* num bytes to lock from start */
+               __u8 wait; /* will caller wait for lock to become available? */
+       } __attribute__ ((packed)) filelock_change;
 } __attribute__ ((packed));

 #define CEPH_MDS_FLAG_REPLAY        1  /* this is a replayed op */
@@ -481,6 +494,23 @@ struct ceph_mds_reply_dirfrag {
        __le32 dist[];
 } __attribute__ ((packed));

...

+struct ceph_filelock {
+       __u64 start;/* file offset to start lock at */
+       __u64 length; /* num bytes to lock; 0 for all following start */
+       __s64 client; /* which client holds the lock */
+       __u64 pid; /* process id holding the lock on the client */
+       __u64 pid_namespace;
+       __u8 type; /* shared lock, exclusive lock, or unlock */
+} __attribute__ ((packed));
```

#9 - 06/23/2010 05:30 PM - Greg Farnum

Hmm, this was halfway done before but it's in properly now.

#10 - 08/03/2010 10:10 AM - Greg Farnum

- Status changed from 4 to Resolved

This stuff got pushed into unstable yesterday.