
Subject: [PATCH 04/25] Add vfsmount writer count
Posted by [Dave Hansen](#) on Mon, 11 Dec 2006 22:30:04 GMT
[View Forum Message](#) <> [Reply to Message](#)

Signed-off-by: Dave Hansen <haveblue@us.ibm.com>

```
lxc-dave/fs/namespace.c      | 63 ++++++
lxc-dave/fs/super.c          | 66 ++++++
lxc-dave/include/linux/fs.h   |  2 +
lxc-dave/include/linux/mount.h | 18 ++++++
4 files changed, 144 insertions(+), 5 deletions(-)
```

```
diff -puN fs/namespace.c~03-24-add-vfsmount-writer-count fs/namespace.c
--- lxc/fs/namespace.c~03-24-add-vfsmount-writer-count 2006-12-11 14:21:58.000000000 -0800
+++ lxc-dave/fs/namespace.c 2006-12-11 14:21:58.000000000 -0800
@@ -57,6 +57,7 @@ struct vfsmount *alloc_vfsmnt(const char
    if (mnt) {
        memset(mnt, 0, sizeof(struct vfsmount));
        atomic_set(&mnt->mnt_count, 1);
+       atomic_set(&mnt->mnt_writers, 0);
        INIT_LIST_HEAD(&mnt->mnt_hash);
        INIT_LIST_HEAD(&mnt->mnt_child);
        INIT_LIST_HEAD(&mnt->mnt_mounts);
@@ -894,6 +895,60 @@ out_unlock:
    return err;
}

+int mnt_make_readonly(struct vfsmount *mnt)
+{
+    int ret = 0;
+
+    WARN_ON(__mnt_is_readonly(mnt));
+
+    /*
+     * This flag set is actually redundant with what
+     * happens in do_remount(), but since we do this
+     * under the lock, anyone attempting to get a write
+     * on it after this will fail.
+     */
+    spin_lock(&mnt->mnt_sb->s_mnt_writers_lock);
+    if (!atomic_read(&mnt->mnt_writers))
+        mnt->mnt_flags |= MNT_READONLY;
+    else
+        ret = -EBUSY;
+    spin_unlock(&mnt->mnt_sb->s_mnt_writers_lock);
+    return ret;
```

```

+}
+
+int mnt_want_write(struct vfsmount *mnt)
+{
+ int ret = 0;
+repeat:
+ if (atomic_add_unless(&mnt->mnt_writers, 1, 0))
+ return 0;
+
+ spin_lock(&mnt->mnt_sb->s_mnt_writers_lock);
+ if (__mnt_is_readonly(mnt)) {
+ ret = -EROFS;
+ goto out;
+ }
+ if (atomic_add_return(1, &mnt->mnt_writers) != 1) {
+ atomic_dec(&mnt->mnt_writers);
+ spin_unlock(&mnt->mnt_sb->s_mnt_writers_lock);
+ goto repeat;
+ }
+ atomic_inc(&mnt->mnt_sb->s_mnt_writers);
+out:
+ spin_unlock(&mnt->mnt_sb->s_mnt_writers_lock);
+ return ret;
+}
+
+void mnt_drop_write(struct vfsmount *mnt)
+{
+ if (!atomic_dec_and_lock(&mnt->mnt_writers,
+ &mnt->mnt_sb->s_mnt_writers_lock))
+ return;
+
+ atomic_dec(&mnt->mnt_sb->s_mnt_writers);
+ spin_unlock(&mnt->mnt_sb->s_mnt_writers_lock);
+}
+
+/*
+ * recursively change the type of the mountpoint.
+ */
@@ -962,6 +1017,7 @@ out:
path_release(&old_nd);
return err;
}
+EXPORT_SYMBOL_GPL(mnt_want_write);

/*
 * change filesystem flags. dir should be a physical root of filesystem.
@@ -985,6 +1041,8 @@ static int do_remount(struct nameidata *

```

```

    down_write(&sb->s_umount);
    err = do_remount_sb(sb, flags, data, 0);
+ if (!(sb->s_flags & MS_RDONLY))
+ mnt_flags |= MNT_SB_WRITABLE;
    if (!err)
        nd->mnt->mnt_flags = mnt_flags;
    up_write(&sb->s_umount);
@@ -992,6 +1050,7 @@ static int do_remount(struct nameidata *
    security_sb_post_remount(nd->mnt, flags, data);
    return err;
}
+EXPORT_SYMBOL_GPL(mnt_drop_write);

static inline int tree_contains_unbindable(struct vfsmount *mnt)
{
@@ -1125,6 +1184,8 @@ int do_add_mount(struct vfsmount *newmnt
    if (S_ISLNK(newmnt->mnt_root->d_inode->i_mode))
        goto unlock;

+ if (!(newmnt->mnt_sb->s_flags & MS_RDONLY))
+ mnt_flags |= MNT_SB_WRITABLE;
    newmnt->mnt_flags = mnt_flags;
    if ((err = graft_tree(newmnt, nd)))
        goto unlock;
@@ -1416,6 +1477,8 @@ long do_mount(char *dev_name, char *dir_
    ((char *)data_page)[PAGE_SIZE - 1] = 0;

    /* Separate the per-mountpoint flags */
+ if (flags & MS_RDONLY)
+ mnt_flags |= MNT_READONLY;
    if (flags & MS_NOSUID)
        mnt_flags |= MNT_NOSUID;
    if (flags & MS_NODEV)
diff -puN fs/super.c~03-24-add-vfsmount-writer-count fs/super.c
--- lxc/fs/super.c~03-24-add-vfsmount-writer-count 2006-12-11 14:21:58.000000000 -0800
+++ lxc-dave/fs/super.c 2006-12-11 14:21:58.000000000 -0800
@@ -94,6 +94,8 @@ static struct super_block *alloc_super(s
    s->s_qcop = sb_quotactl_ops;
    s->s_op = &default_op;
    s->s_time_gran = 1000000000;
+ atomic_set(&s->s_mnt_writers, 0);
+ spin_lock_init(&s->s_mnt_writers_lock);
}
out:
    return s;
@@ -577,6 +579,53 @@ static void mark_files_ro(struct super_b
    file_list_unlock();
}

```

```

+static void __sb_mounts_mod_flag(struct super_block *sb, int set, int flag)
+{
+ struct list_head *p;
+
+ spin_lock(&vfsmount_lock);
+ list_for_each(p, &sb->s_vfsmounts) {
+ struct vfsmount *mnt =
+ list_entry(p, struct vfsmount, mnt_sb_list);
+ if (set)
+ mnt->mnt_flags |= flag;
+ else
+ mnt->mnt_flags &= ~flag;
+ }
+ spin_unlock(&vfsmount_lock);
+}
+
+static void sb_mounts_set_flag(struct super_block *sb, int flag)
+{
+ __sb_mounts_mod_flag(sb, 1, flag);
+}
+static void sb_mounts_clear_flag(struct super_block *sb, int flag)
+{
+ __sb_mounts_mod_flag(sb, 0, flag);
+}
+
+static int sb_remount_ro(struct super_block *sb)
+{
+ return fs_may_remount_ro(sb);
+ spin_lock(&sb->s_mnt_writers_lock);
+ if (atomic_read(&sb->s_mnt_writers) > 0) {
+ spin_unlock(&sb->s_mnt_writers_lock);
+ return -EBUSY;
+ }
+
+ sb_mounts_clear_flag(sb, MNT_SB_WRITABLE);
+ spin_unlock(&sb->s_mnt_writers_lock);
+
+ return 0;
+}
+
+static void sb_remount_rw(struct super_block *sb)
+{
+ spin_lock(&sb->s_mnt_writers_lock);
+ sb_mounts_set_flag(sb, MNT_SB_WRITABLE);
+ spin_unlock(&sb->s_mnt_writers_lock);
+}
+

```

```

/**
 * do_remount_sb - asks filesystem to change mount options.
 * @sb: superblock in question
@@ -588,7 +637,8 @@ static void mark_files_ro(struct super_b
 */
int do_remount_sb(struct super_block *sb, int flags, void *data, int force)
{
- int retval;
+ int retval = 0;
+ int sb_started_ro = (sb->s_flags & MS_RDONLY);

#ifdef CONFIG_BLOCK
if (!(flags & MS_RDONLY) && bdev_read_only(sb->s_bdev))
@@ -601,13 +651,14 @@ int do_remount_sb(struct super_block *sb

/* If we are remounting RDONLY and current sb is read/write,
   make sure there are no rw files opened */
- if ((flags & MS_RDONLY) && !(sb->s_flags & MS_RDONLY)) {
+ if ((flags & MS_RDONLY) && !sb_started_ro) {
if (force)
mark_files_ro(sb);
- else if (!fs_may_remount_ro(sb))
- return -EBUSY;
+ else
+ retval = sb_remount_ro(sb);
+ if (retval)
+ return retval;
}
-
if (sb->s_op->remount_fs) {
lock_super(sb);
retval = sb->s_op->remount_fs(sb, &flags, data);
@@ -615,6 +666,9 @@ int do_remount_sb(struct super_block *sb
if (retval)
return retval;
}
+ if (!(flags & MS_RDONLY) && sb_started_ro)
+ sb_remount_rw(sb);
+
sb->s_flags = (sb->s_flags & ~MS_RMT_MASK) | (flags & MS_RMT_MASK);
return 0;
}
@@ -903,6 +957,8 @@ vfs_kern_mount(struct file_system_type *

mnt->mnt_mountpoint = mnt->mnt_root;
mnt->mnt_parent = mnt;
+ if (!(mnt->mnt_sb->s_flags & MS_RDONLY))
+ mnt->mnt_flags |= MNT_SB_WRITABLE;

```

```

up_write(&mnt->mnt_sb->s_umount);
free_secdata(secdata);
return mnt;
diff -puN include/linux/fs.h~03-24-add-vfsmount-writer-count include/linux/fs.h
--- lxc/include/linux/fs.h~03-24-add-vfsmount-writer-count 2006-12-11 14:21:58.000000000 -0800
+++ lxc-dave/include/linux/fs.h 2006-12-11 14:21:58.000000000 -0800
@@ -968,6 +968,8 @@ struct super_block {
    struct list_head s_io; /* parked for writeback */
    struct hlist_head s_anon; /* anonymous dentries for (nfs) exporting */
    struct list_head s_vfsmounts;
+ atomic_t s_mnt_writers; /* vfsmounts with active writers */
+ spinlock_t s_mnt_writers_lock; /* taken when mounts change rw state */
    struct list_head s_files;

    struct block_device *s_bdev;
diff -puN include/linux/mount.h~03-24-add-vfsmount-writer-count include/linux/mount.h
--- lxc/include/linux/mount.h~03-24-add-vfsmount-writer-count 2006-12-11 14:21:58.000000000 -0800
+++ lxc-dave/include/linux/mount.h 2006-12-11 14:21:58.000000000 -0800
@@ -28,6 +28,8 @@ struct mnt_namespace;
#define MNT_NOATIME 0x08
#define MNT_NODIRATIME 0x10
#define MNT_RELATIME 0x20
+#define MNT_READONLY 0x40 /* does the user want this to be r/o? */
+#define MNT_SB_WRITABLE 0x80 /* does the SB currently allow writes? */

#define MNT_SHRINKABLE 0x100

@@ -45,6 +47,7 @@ struct vfsmount {
    struct list_head mnt_mounts; /* list of children, anchored here */
    struct list_head mnt_child; /* and going through their mnt_child */
    atomic_t mnt_count;
+ atomic_t mnt_writers;
    int mnt_flags;
    int mnt_expiry_mark; /* true if marked for expiry */
    char *mnt_devname; /* Name of device e.g. /dev/dsk/hda1 */
@@ -65,6 +68,21 @@ static inline struct vfsmount *mntget(st
    return mnt;
}

+static inline int __mnt_is_readonly(struct vfsmount *mnt)
+{
+ return (mnt->mnt_flags & MNT_READONLY) ||
+      !(mnt->mnt_flags & MNT_SB_WRITABLE);
+}
+
+static inline void __mnt_unmake_readonly(struct vfsmount *mnt)
+{

```

```
+ WARN_ON(!__mnt_is_readonly(mnt));
+ mnt->mnt_flags &= ~MNT_READONLY;
+}
+
+extern int mnt_make_readonly(struct vfsmount *mnt);
+extern int mnt_want_write(struct vfsmount *mnt);
+extern void mnt_drop_write(struct vfsmount *mnt);
+extern void mntput_no_expire(struct vfsmount *mnt);
+extern void mnt_pin(struct vfsmount *mnt);
+extern void mnt_unpin(struct vfsmount *mnt);
```

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
