Subject: [PATCH 06/14] sysfs: Rewrite sysfs_get_dentry
Posted by ebiederm on Tue, 31 Jul 2007 10:27:17 GMT
View Forum Message <> Reply to Message

Currently sysfs_get_dentry is very hairy and requires all kinds
of locking magic. This patch rewrites sysfs_get_dentry to
not use the cached sd->s_dentry, and instead simply lookup
and create dcache entries.

This lays the foundation for removing s_dentry and the
current hairy sysfs_assoc_lock logic.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
---
 fs/sysfs/dir.c |  117 +++++++++++++++++++++++--------------------------------
 1 files changed, 49 insertions(+), 68 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 6933f36..f22d60c 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -20,6 +20,8 @@ spinlock_t sysfs_assoc_lock = SPIN_LOCK_UNLOCKED;
 static spinlock_t sysfs_ino_lock = SPIN_LOCK_UNLOCKED;
 static DEFINE_IDA(sysfs_ino_ida);

+static void sysfs_attach_dentry(struct sysfs_dirent *sd, struct dentry *dentry);
+
 /**
  * sysfs_link_sibling - link sysfs_dirent into sibling list
  * @sd: sysfs_dirent of interest
@@ -66,10 +68,10 @@ void sysfs_unlink_sibling(struct sysfs_dirent *sd)
  * sysfs_get_dentry - get dentry for the given sysfs_dirent
  * @sd: sysfs_dirent of interest
  *
- * Get dentry for @sd.  Dentry is looked up if currently not
- * present.  This function climbs sysfs_dirent tree till it
- * reaches a sysfs_dirent with valid dentry attached and descends
- * down from there looking up dentry for each step.
+ * Get dentry for @sd.
+ *
+ * This function descends the sysfs dentry tree from the root
+ * populating it if necessary until it reaches the dentry for @sd.
  *
  * LOCKING:
  * Kernel thread context (may sleep)
@@ -77,85 +79,58 @@ void sysfs_unlink_sibling(struct sysfs_dirent *sd)
  * RETURNS:
  * Pointer to found dentry on success, ERR_PTR() value on error.
```

```
  */
-struct dentry *sysfs_get_dentry(struct sysfs_dirent *sd)
+struct dentry *__sysfs_get_dentry(struct sysfs_dirent *sd, int create)
 {
  struct sysfs_dirent *cur;
  struct dentry *parent_dentry, *dentry;
- int i, depth;
-
- /* Find the first parent which has valid s_dentry and get the
-  * dentry.
-  */
- mutex_lock(&sysfs_mutex);
- restart0:
- spin_lock(&sysfs_assoc_lock);
- restart1:
- spin_lock(&dcache_lock);
-
- dentry = NULL;
- depth = 0;
- cur = sd;
- while (!cur->s_dentry || !cur->s_dentry->d_inode) {
-  if (cur->s_flags & SYSFS_FLAG_REMOVED) {
-   dentry = ERR_PTR(-ENOENT);
-   depth = 0;
-   break;
-  }
-  cur = cur->s_parent;
-  depth++;
- }
- if (!IS_ERR(dentry))
-  dentry = dget_locked(cur->s_dentry);
+ struct qstr name;
+ struct inode *inode;

- spin_unlock(&dcache_lock);
- spin_unlock(&sysfs_assoc_lock);
+ parent_dentry = NULL;
+ dentry = dget(sysfs_sb->s_root);

- /* from the found dentry, look up depth times */
- while (depth--) {
-  /* find and get depth'th ancestor */
-  for (cur = sd, i = 0; cur && i < depth; i++)
+ do {
+  /* Find the first ancestor I have not looked up */
+  cur = sd;
+  while (cur->s_parent != dentry->d_fsdata)
    cur = cur->s_parent;
```

```
-  /* This can happen if tree structure was modified due
-   * to move/rename.  Restart.
-   */
-  if (i != depth) {
-   dput(dentry);
-   goto restart0;
-  }
-
-  sysfs_get(cur);
-
-  mutex_unlock(&sysfs_mutex);
-
   /* look it up */
-  parent_dentry = dentry;
-  dentry = lookup_one_len_kern(cur->s_name, parent_dentry,
-         strlen(cur->s_name));
   dput(parent_dentry);
-
-  if (IS_ERR(dentry)) {
-   sysfs_put(cur);
-   return dentry;
+  parent_dentry = dentry;
+  name.name = cur->s_name;
+  name.len = strlen(cur->s_name);
+  dentry = d_hash_and_lookup(parent_dentry, &name);
+  if (dentry)
+   continue;
+  if (!create)
+   goto out;
+  dentry = d_alloc(parent_dentry, &name);
+  if (!dentry) {
+   dentry = ERR_PTR(-ENOMEM);
+   goto out;
   }
-
-  mutex_lock(&sysfs_mutex);
-  spin_lock(&sysfs_assoc_lock);
-
-  /* This, again, can happen if tree structure has
-   * changed and we looked up the wrong thing.  Restart.
-   */
-  if (cur->s_dentry != dentry) {
+  inode = sysfs_get_inode(cur);
+  if (!inode) {
    dput(dentry);
-   sysfs_put(cur);
-   goto restart1;
```

```
+   dentry = ERR_PTR(-ENOMEM);
+   goto out;
   }
+  d_instantiate(dentry, inode);
+  sysfs_attach_dentry(cur, dentry);
+ } while (cur != sd);

-  spin_unlock(&sysfs_assoc_lock);
+out:
+ dput(parent_dentry);
+ return dentry;
+}

-  sysfs_put(cur);
- }
+struct dentry *sysfs_get_dentry(struct sysfs_dirent *sd)
+{
+ struct dentry *dentry;

+ mutex_lock(&sysfs_mutex);
+ dentry = __sysfs_get_dentry(sd, 1);
  mutex_unlock(&sysfs_mutex);
  return dentry;
 }
@@ -750,6 +725,12 @@ static struct dentry * sysfs_lookup(struct inode *dir, struct dentry
*dentry,
  struct inode *inode;

  mutex_lock(&sysfs_mutex);
+
+ /* Guard against races with sysfs_get_dentry */
+ result = d_hash_and_lookup(dentry->d_parent, &dentry->d_name);
+ if (result)
+  goto out;
+
  for (sd = parent_sd->s_children; sd; sd = sd->s_sibling) {
   if (sysfs_type(sd) &&
       !strcmp(sd->s_name, dentry->d_name.name))
--
1.5.1.1.181.g2de0
```