
Subject: Re: [Q] missing unused dentry in prune_dcache()?
Posted by [David Howells](#) on Thu, 26 Oct 2006 13:23:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Vasily Averin <vvs@sw.ru> wrote:

> I've noticed one more minor issue in your patch: in
> shrink_dcache_for_umount_subtree() function you decrement
> dentry_stat.nr_dentry without dcache_lock.

How about the attached patch?

David

VFS: Fix an error in unused dentry counting

From: David Howells <dhowells@redhat.com>

Fix an error in unused dentry counting in shrink_dcache_for_umount_subtree() in which the count is modified without the dcache_lock held.

Signed-Off-By: David Howells <dhowells@redhat.com>

fs/dcache.c | 22 ++++++-----
1 files changed, 15 insertions(+), 7 deletions(-)

```
diff --git a/fs/dcache.c b/fs/dcache.c
index a1ff91e..eab1bf4 100644
--- a/fs/dcache.c
+++ b/fs/dcache.c
@@ -553,16 +553,17 @@ repeat:
 * - see the comments on shrink_dcache_for_umount() for a description of the
 * locking
 */
-static void shrink_dcache_for_umount_subtree(struct dentry *dentry)
+static unsigned shrink_dcache_for_umount_subtree(struct dentry *dentry)
{
    struct dentry *parent;
+    unsigned detached = 0;

    BUG_ON(!IS_ROOT(dentry));

    /* detach this root from the system */
    spin_lock(&dcache_lock);
    if (!list_empty(&dentry->d_lru)) {
-    dentry_stat.nr_unused--;
+    detached++;
    }
```

```

    list_del_init(&dentry->d_lru);
}
__d_drop(dentry);
@@ -579,7 +580,7 @@ static void shrink_dcache_for_umount_sub
    list_for_each_entry(loop, &dentry->d_subdirs,
        d_u.d_child) {
        if (!list_empty(&loop->d_lru)) {
-       dentry_stat.nr_unused--;
+       detached++;
        list_del_init(&loop->d_lru);
        }

@@ -620,7 +621,7 @@ static void shrink_dcache_for_umount_sub
    atomic_dec(&parent->d_count);

    list_del(&dentry->d_u.d_child);
-   dentry_stat.nr_dentry--; /* For d_free, below */
+   detached++; /* For d_free, below */

    inode = dentry->d_inode;
    if (inode) {
@@ -638,7 +639,7 @@ static void shrink_dcache_for_umount_sub
        * otherwise we ascend to the parent and move to the
        * next sibling if there is one */
    if (!parent)
-   return;
+   return detached;

    dentry = parent;

@@ -663,6 +664,7 @@ static void shrink_dcache_for_umount_sub
void shrink_dcache_for_umount(struct super_block *sb)
{
    struct dentry *dentry;
+   unsigned detached = 0;

    if (down_read_trylock(&sb->s_umount))
        BUG();
@@ -670,11 +672,17 @@ void shrink_dcache_for_umount(struct sup
    dentry = sb->s_root;
    sb->s_root = NULL;
    atomic_dec(&dentry->d_count);
-   shrink_dcache_for_umount_subtree(dentry);
+   detached = shrink_dcache_for_umount_subtree(dentry);

    while (!hlist_empty(&sb->s_anon)) {
        dentry = hlist_entry(sb->s_anon.first, struct dentry, d_hash);
-       shrink_dcache_for_umount_subtree(dentry);

```

```
+ detached += shrink_dcache_for_umount_subtree(dentry);  
+ }  
+  
+ if (detached) {  
+ spin_lock(&dcache_lock);  
+ dentry_stat.nr_unused -= detached;  
+ spin_unlock(&dcache_lock);  
+ }  
+ }
```
