Subject: Re: [Q] missing unused dentry in prune_dcache()?
Posted by David Howells on Fri, 27 Oct 2006 10:42:49 GMT
View Forum Message <> Reply to Message

Vasily Averin <vvs@sw.ru> wrote:

> Therefore I believe that my patch is optimal solution.

I'm not sure that prune_dcache() is particularly optimal. If we're looking to prune for a specific superblock, it may scan most of the dentry_unused list several times, once for each dentry it eliminates.

Imagine the list with a million dentries on it. Imagine further that all the dentries you're trying to eliminate are up near the head end: you're going to have to scan most of the list several times unnecessarily; if you're asked to kill 128 dentries, you might wind up examining on the order of 100,000,000 dentries, 99% of which you scan 128 times.

I wonder if this could be improved by making the assumption that there won't be any entries inserted tailwards of where we've just looked. The problem is that if dcache_lock is dropped, we've no way of keeping track of the current position without inserting a marker into the list.

Now we could do the marker thing quite easily. We'd have to insert a dummy dcache entry, probably with d_sb pointing to some special location that is recognised as saying "that dentry is a marker".

We could do something like the attached patch, for example. Note that the patch compiles, but I haven't tested it. It also uses a big chunk of stack space for the marker. It ought to be safe enough with respect to the other functions that touch that list - all of those deal with specific dentries or look for dentries by superblock.

David

```
diff --git a/fs/dcache.c b/fs/dcache.c
index eab1bf4..a1cae74 100644
--- a/fs/dcache.c
+++ b/fs/dcache.c
@@ -395,18 +395,27 @@ static void prune_one_dentry(struct dent
  * This function may fail to free any resources if
  * all the dentries are in use.
  */
-
+
 static void prune_dcache(int count, struct super_block *sb)
 {
+ struct dentry marker = {
```

```
+  .d_sb = (struct super_block *) &prune_dcache,
+ };
+
+ struct list_head *tmp;
+
  spin_lock(&dcache_lock);
+ list_add_tail(&marker.d_lru, &dentry_unused);
+
  for (; count ; count--) {
   struct dentry *dentry;
-  struct list_head *tmp;
   struct rw_semaphore *s_umount;

   cond_resched_lock(&dcache_lock);

-  tmp = dentry_unused.prev;
+  tmp = marker.d_lru.prev;
+  list_del_init(&marker.d_lru);
+
   if (sb) {
    /* Try to find a dentry for this sb, but don't try
     * too hard, if they aren't near the tail they will
@@ -418,9 +427,18 @@ static void prune_dcache(int count, stru
     skip--;
     tmp = tmp->prev;
    }
+  } else {
+   /* We may not be the only pruner */
+   while (tmp != &dentry_unused) {
+    dentry = list_entry(tmp, struct dentry, d_lru);
+    if (dentry->d_sb !=
+        (struct super_block *) &prune_dcache)
+     break;
+   }
   }
   if (tmp == &dentry_unused)
    break;
+  list_add(&marker.d_lru, tmp);
   list_del_init(tmp);
   prefetch(dentry_unused.prev);
    dentry_stat.nr_unused--;
@@ -439,7 +457,7 @@ static void prune_dcache(int count, stru
   /* If the dentry was recently referenced, don't free it. */
   if (dentry->d_flags & DCACHE_REFERENCED) {
    dentry->d_flags &= ~DCACHE_REFERENCED;
-    list_add(&dentry->d_lru, &dentry_unused);
+    list_add(&dentry->d_lru, &marker.d_lru);
     dentry_stat.nr_unused++;
```

```
     spin_unlock(&dentry->d_lock);
    continue;
@@ -478,12 +496,10 @@ static void prune_dcache(int count, stru
    up_read(s_umount);
   }
   spin_unlock(&dentry->d_lock);
-  /* Cannot remove the first dentry, and it isn't appropriate
-   * to move it to the head of the list, so give up, and try
-   * later
-   */
-  break;
+  list_add(&dentry->d_lru, &marker.d_lru);
+  dentry_stat.nr_unused++;
  }
+ list_del(&marker.d_lru);
  spin_unlock(&dcache_lock);
 }
```