

---

Subject: [PATCH] shrink\_dcache\_sb speedup  
Posted by [den](#) on Fri, 14 Sep 2007 08:37:12 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Denis V. Lunev <[den@openvz.org](mailto:den@openvz.org)>

This patch makes shrink\_dcache\_sb consistent with dentry pruning policy.

On the first pass we iterate over dentry unused list and prepare some dentries for removal.

However, since the existing code moves evicted dentries to the beginning of the LRU it can happen that fresh dentries from other superblocks will be inserted *\*before\** our dentries.

This can result in significant slowdown of shrink\_dcache\_sb(). Moreover, for virtual filesystems like unionfs which can call dput() during dentries kill existing code results in  $O(n^2)$  complexity.

We observed 2 minutes shrink\_dcache\_sb() with only 35000 dentries.

To avoid this effects we propose to isolate sb dentries at the end of LRU list.

Signed-off-by: Denis V. Lunev <[den@openvz.org](mailto:den@openvz.org)>

Signed-off-by: Kirill Korotaev <[dev@openvz.org](mailto:dev@openvz.org)>

Signed-off-by: Andrey Mirkin <[amirkin@openvz.org](mailto:amirkin@openvz.org)>

-----

```
--- ./fs/dcache.c.shrink 2007-09-14 10:25:21.000000000 +0400
+++ ./fs/dcache.c 2007-09-14 10:26:08.000000000 +0400
@@ -553,18 +553,18 @@ void shrink_dcache_sb(struct super_block
 * superblock to the most recent end of the unused list.
 */
spin_lock(&dcache_lock);
- list_for_each_safe(tmp, next, &dentry_unused) {
+ list_for_each_prev_safe(tmp, next, &dentry_unused) {
    dentry = list_entry(tmp, struct dentry, d_lru);
    if (dentry->d_sb != sb)
        continue;
- list_move(tmp, &dentry_unused);
+ list_move_tail(tmp, &dentry_unused);
}

/*
 * Pass two ... free the dentries for this superblock.
 */
repeat:
```

```

- list_for_each_safe(tmp, next, &dentry_unused) {
+ list_for_each_prev_safe(tmp, next, &dentry_unused) {
    dentry = list_entry(tmp, struct dentry, d_lru);
    if (dentry->d_sb != sb)
        continue;
--- ./include/linux/list.h.shrink 2007-08-10 16:58:49.000000000 +0400
+++ ./include/linux/list.h 2007-09-14 10:26:08.000000000 +0400
@@ -478,6 +478,18 @@ static inline void list_splice_init_rcu(
    pos = n, n = pos->next)

/**
+ * list_for_each_prev_safe - iterate over a list backwards safe against removal
+ * of list entry
+ * @pos: the &struct list_head to use as a loop cursor.
+ * @n: another &struct list_head to use as temporary storage
+ * @head: the head for your list.
+ */
+#define list_for_each_prev_safe(pos, n, head) \
+ for (pos = (head)->prev, n = pos->prev; \
+      prefetch(pos->prev), pos != (head); \
+      pos = n, n = pos->prev)
+
+/**
 * list_for_each_entry - iterate over list of given type
 * @pos: the type * to use as a loop cursor.
 * @head: the head for your list.

```

---