Subject: [PATCH][SHMEM] Factor out sbi->free_inodes manipulations
Posted by Pavel Emelianov on Thu, 22 Nov 2007 16:57:22 GMT
View Forum Message <> Reply to Message

The shmem_sb_info structure has a number of free_inodes. This
value is altered in appropriate places under spinlock and with
the sbi->max_inodes != 0 check.

Consolidate these manipulations into two helpers.

This is minus 30 bytes of shmem.o and minus 4 :) lines of code.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/mm/shmem.c b/mm/shmem.c
index e0d4b2e..abba17c 100644
--- a/mm/shmem.c
+++ b/mm/shmem.c
@@ -205,6 +205,29 @@ static void shmem_free_blocks(struct inode *inode, long pages)
  }
 }

+static int shmem_inc_inodes(struct shmem_sb_info *sbinfo)
+{
+ if (sbinfo->max_inodes) {
+  spin_lock(&sbinfo->stat_lock);
+  if (!sbinfo->free_inodes) {
+   spin_unlock(&sbinfo->stat_lock);
+   return -ENOMEM;
+  }
+  sbinfo->free_inodes--;
+  spin_unlock(&sbinfo->stat_lock);
+ }
+ return 0;
+}
+
+static void shmem_dec_inodes(struct shmem_sb_info *sbinfo)
+{
+ if (sbinfo->max_inodes) {
+  spin_lock(&sbinfo->stat_lock);
+  sbinfo->free_inodes++;
+  spin_unlock(&sbinfo->stat_lock);
+ }
+}
+
 /*
```

```
 * shmem_recalc_inode - recalculate the size of an inode
 *
@@ -777,11 +800,7 @@ static void shmem_delete_inode(struct inode *inode)
   }
  }
  BUG_ON(inode->i_blocks);
- if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
-  sbinfo->free_inodes++;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ shmem_dec_inodes(sbinfo);
  clear_inode(inode);
 }

@@ -1370,15 +1389,8 @@ shmem_get_inode(struct super_block *sb, int mode, dev_t dev)
  struct shmem_inode_info *info;
  struct shmem_sb_info *sbinfo = SHMEM_SB(sb);

- if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
-  if (!sbinfo->free_inodes) {
-   spin_unlock(&sbinfo->stat_lock);
-   return NULL;
-  }
-  sbinfo->free_inodes--;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ if (shmem_inc_inodes(sbinfo))
+  return NULL;

  inode = new_inode(sb);
  if (inode) {
@@ -1422,11 +1434,8 @@ shmem_get_inode(struct super_block *sb, int mode, dev_t dev)
     NULL);
    break;
   }
- } else if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
-  sbinfo->free_inodes++;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ } else
+  shmem_dec_inodes(sbinfo);
  return inode;
 }

@@ -1676,15 +1685,8 @@ static int shmem_link(struct dentry *old_dentry, struct inode *dir,
```

```
struct dentr
   * but each new link needs a new dentry, pinning lowmem, and
   * tmpfs dentries cannot be pruned until they are unlinked.
   */
- if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
-  if (!sbinfo->free_inodes) {
-   spin_unlock(&sbinfo->stat_lock);
-   return -ENOSPC;
-  }
-  sbinfo->free_inodes--;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ if (shmem_inc_inodes(sbinfo))
+  return -ENOSPC;

  dir->i_size += BOGO_DIRENT_SIZE;
  inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
@@ -1699,14 +1701,8 @@ static int shmem_unlink(struct inode *dir, struct dentry *dentry)
 {
  struct inode *inode = dentry->d_inode;

- if (inode->i_nlink > 1 && !S_ISDIR(inode->i_mode)) {
-  struct shmem_sb_info *sbinfo = SHMEM_SB(inode->i_sb);
-  if (sbinfo->max_inodes) {
-   spin_lock(&sbinfo->stat_lock);
-   sbinfo->free_inodes++;
-   spin_unlock(&sbinfo->stat_lock);
-  }
- }
+ if (inode->i_nlink > 1 && !S_ISDIR(inode->i_mode))
+  shmem_dec_inodes(SHMEM_SB(inode->i_sb));

  dir->i_size -= BOGO_DIRENT_SIZE;
  inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
```