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;
```

Subject: Re: [PATCH][SHMEM] Factor out sbi->free_inodes manipulations
Posted by Hugh Dickins on Fri, 23 Nov 2007 13:41:55 GMT
View Forum Message <> Reply to Message

Looks good, but we can save slightly more there (depending on config),
and I found your inc/dec names a little confusing, since the count is
going the other way: how do you feel about this version? (I'd like it
better if those helpers could take a struct inode *, but they cannot.)
Hugh

From: Pavel Emelyanov <xemul@openvz.org>

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 42 bytes of shmem.o and minus 4 :) lines of code.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Signed-off-by: Hugh Dickins <hugh@veritas.com>

```
 mm/shmem.c |   72 ++++++++++++++++++++++++++++++++++++------------------------------------
 1 file changed, 34 insertions(+), 38 deletions(-)

--- 2.6.24-rc3/mm/shmem.c 2007-11-07 04:21:45.000000000 +0000
+++ linux/mm/shmem.c 2007-11-23 12:43:28.000000000 +0000
@@ -207,6 +207,31 @@ static void shmem_free_blocks(struct ino
  }
 }

+static int shmem_reserve_inode(struct super_block *sb)
+{
+ 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 -ENOMEM;
+  }
+  sbinfo->free_inodes--;
+  spin_unlock(&sbinfo->stat_lock);
+ }
+ return 0;
+}
+
+static void shmem_free_inode(struct super_block *sb)
+{
+ struct shmem_sb_info *sbinfo = SHMEM_SB(sb);
+ 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
  *
@@ -762,7 +787,6 @@ static int shmem_notify_change(struct de

 static void shmem_delete_inode(struct inode *inode)
 {
- struct shmem_sb_info *sbinfo = SHMEM_SB(inode->i_sb);
  struct shmem_inode_info *info = SHMEM_I(inode);

  if (inode->i_op->truncate == shmem_truncate) {
@@ -777,11 +801,7 @@ static void shmem_delete_inode(struct in
   }
  }
  BUG_ON(inode->i_blocks);
- if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
-  sbinfo->free_inodes++;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ shmem_free_inode(inode->i_sb);
  clear_inode(inode);
 }

@@ -1398,15 +1418,8 @@ shmem_get_inode(struct super_block *sb,
  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_reserve_inode(sb))
+  return NULL;

  inode = new_inode(sb);
  if (inode) {
@@ -1450,11 +1463,8 @@ shmem_get_inode(struct super_block *sb,
       NULL);
    break;
   }
- } else if (sbinfo->max_inodes) {
-  spin_lock(&sbinfo->stat_lock);
```

```
-  sbinfo->free_inodes++;
-  spin_unlock(&sbinfo->stat_lock);
- }
+ } else
+  shmem_free_inode(sb);
  return inode;
 }

@@ -1797,22 +1807,14 @@ static int shmem_create(struct inode *di
 static int shmem_link(struct dentry *old_dentry, struct inode *dir, struct dentry *dentry)
 {
  struct inode *inode = old_dentry->d_inode;
- struct shmem_sb_info *sbinfo = SHMEM_SB(inode->i_sb);

  /*
   * No ordinary (disk based) filesystem counts links as inodes;
   * 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_reserve_inode(inode->i_sb))
+  return -ENOSPC;

  dir->i_size += BOGO_DIRENT_SIZE;
  inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
@@ -1827,14 +1829,8 @@ static int shmem_unlink(struct inode *di
 {
  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_free_inode(inode->i_sb);
```

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

Subject: Re: [PATCH][SHMEM] Factor out sbi->free_inodes manipulations
Posted by Pavel Emelianov on Fri, 23 Nov 2007 13:53:07 GMT
View Forum Message <> Reply to Message

Hugh Dickins wrote:
> Looks good, but we can save slightly more there (depending on config),
> and I found your inc/dec names a little confusing, since the count is
> going the other way: how do you feel about this version? (I'd like it

This is perfect =) Thanks for the masterclass!

> better if those helpers could take a struct inode *, but they cannot.)
> Hugh
>
>
> From: Pavel Emelyanov <xemul@openvz.org>
>
> 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 42 bytes of shmem.o and minus 4 :) lines of code.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
> Signed-off-by: Hugh Dickins <hugh@veritas.com>
> ---
>
> mm/shmem.c | 72 ++++++++++++++++++++++++++++++++++---------------------------
> 1 file changed, 34 insertions(+), 38 deletions(-)
>
> --- 2.6.24-rc3/mm/shmem.c 2007-11-07 04:21:45.000000000 +0000
> +++ linux/mm/shmem.c 2007-11-23 12:43:28.000000000 +0000
> @@ -207,6 +207,31 @@ static void shmem_free_blocks(struct ino
>   }
> }
>
> +static int shmem_reserve_inode(struct super_block *sb)
> +{
> + 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 -ENOMEM;
> +  }
> +  sbinfo->free_inodes--;
> +  spin_unlock(&sbinfo->stat_lock);
> + }
> + return 0;
> +}
> +
> +static void shmem_free_inode(struct super_block *sb)
> +{
> + struct shmem_sb_info *sbinfo = SHMEM_SB(sb);
> + 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
>   *
> @@ -762,7 +787,6 @@ static int shmem_notify_change(struct de
>
>  static void shmem_delete_inode(struct inode *inode)
>  {
> - struct shmem_sb_info *sbinfo = SHMEM_SB(inode->i_sb);
>   struct shmem_inode_info *info = SHMEM_I(inode);
>
>   if (inode->i_op->truncate == shmem_truncate) {
> @@ -777,11 +801,7 @@ static void shmem_delete_inode(struct in
>    }
>   }
>   BUG_ON(inode->i_blocks);
> - if (sbinfo->max_inodes) {
> -  spin_lock(&sbinfo->stat_lock);
> -  sbinfo->free_inodes++;
> -  spin_unlock(&sbinfo->stat_lock);
> - }
> + shmem_free_inode(inode->i_sb);
>   clear_inode(inode);
>  }
>
> @@ -1398,15 +1418,8 @@ shmem_get_inode(struct super_block *sb,
>   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_reserve_inode(sb))
> +  return NULL;
>
>   inode = new_inode(sb);
>   if (inode) {
> @@ -1450,11 +1463,8 @@ shmem_get_inode(struct super_block *sb,
>         NULL);
>     break;
>    }
> - } else if (sbinfo->max_inodes) {
> -  spin_lock(&sbinfo->stat_lock);
> -  sbinfo->free_inodes++;
> -  spin_unlock(&sbinfo->stat_lock);
> - }
> + } else
> +  shmem_free_inode(sb);
>   return inode;
>  }
>
> @@ -1797,22 +1807,14 @@ static int shmem_create(struct inode *di
>  static int shmem_link(struct dentry *old_dentry, struct inode *dir, struct dentry *dentry)
>  {
>   struct inode *inode = old_dentry->d_inode;
> - struct shmem_sb_info *sbinfo = SHMEM_SB(inode->i_sb);
>
>   /*
>    * No ordinary (disk based) filesystem counts links as inodes;
>    * 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_reserve_inode(inode->i_sb))
```

```
> +  return -ENOSPC;
>
>   dir->i_size += BOGO_DIRENT_SIZE;
>   inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
> @@ -1827,14 +1829,8 @@ static int shmem_unlink(struct inode *di
>  {
>   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_free_inode(inode->i_sb);
>
>   dir->i_size -= BOGO_DIRENT_SIZE;
>   inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
>
```

Subject: Re: [PATCH][SHMEM] Factor out sbi->free_inodes manipulations
Posted by akpm on Tue, 27 Nov 2007 05:23:11 GMT
View Forum Message <> Reply to Message

On Fri, 23 Nov 2007 13:41:55 +0000 (GMT) Hugh Dickins <hugh@veritas.com> wrote:

> Looks good, but we can save slightly more there (depending on config),
> and I found your inc/dec names a little confusing, since the count is
> going the other way: how do you feel about this version? (I'd like it
> better if those helpers could take a struct inode *, but they cannot.)
> Hugh
>
>
> From: Pavel Emelyanov <xemul@openvz.org>
>
> 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 42 bytes of shmem.o and minus 4 :) lines of code.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
> Signed-off-by: Hugh Dickins <hugh@veritas.com>
> ---
>
>  mm/shmem.c |   72 +++++++++++++++++++++++++++++++-----------------------------
>  1 file changed, 34 insertions(+), 38 deletions(-)
>
> --- 2.6.24-rc3/mm/shmem.c 2007-11-07 04:21:45.000000000 +0000
> +++ linux/mm/shmem.c 2007-11-23 12:43:28.000000000 +0000
> @@ -207,6 +207,31 @@ static void shmem_free_blocks(struct ino
>   }
>  }
>
> +static int shmem_reserve_inode(struct super_block *sb)
> +{
> + 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 -ENOMEM;
> +  }
> +  sbinfo->free_inodes--;
> +  spin_unlock(&sbinfo->stat_lock);
> + }
> + return 0;
> +}
```

It is peculair to (wrongly) return -ENOMEM

```
> + if (shmem_reserve_inode(inode->i_sb))
> +  return -ENOSPC;
```

and to then correct it in the caller..

Something boringly conventional such as the below, perhaps?

```
--- a/mm/shmem.c~shmem-factor-out-sbi-free_inodes-manipulations-fix
+++ a/mm/shmem.c
@@ -212,7 +212,7 @@ static int shmem_reserve_inode(struct su
  spin_lock(&sbinfo->stat_lock);
  if (!sbinfo->free_inodes) {
   spin_unlock(&sbinfo->stat_lock);
-   return -ENOMEM;
+   return -ENOSPC;
  }
  sbinfo->free_inodes--;
  spin_unlock(&sbinfo->stat_lock);
```

```
@@ -1679,14 +1679,16 @@ static int shmem_create(struct inode *di
 static int shmem_link(struct dentry *old_dentry, struct inode *dir, struct dentry *dentry)
 {
  struct inode *inode = old_dentry->d_inode;
+ int ret;

  /*
   * No ordinary (disk based) filesystem counts links as inodes;
   * but each new link needs a new dentry, pinning lowmem, and
   * tmpfs dentries cannot be pruned until they are unlinked.
   */
- if (shmem_reserve_inode(inode->i_sb))
-  return -ENOSPC;
+ ret = shmem_reserve_inode(inode->i_sb);
+ if (ret)
+  goto out;

  dir->i_size += BOGO_DIRENT_SIZE;
  inode->i_ctime = dir->i_ctime = dir->i_mtime = CURRENT_TIME;
@@ -1694,7 +1696,8 @@ static int shmem_link(struct dentry *old
  atomic_inc(&inode->i_count); /* New dentry reference */
  dget(dentry);  /* Extra pinning count for the created dentry */
  d_instantiate(dentry, inode);
- return 0;
+out:
+ return ret;
 }

 static int shmem_unlink(struct inode *dir, struct dentry *dentry)
_
```

Subject: Re: [PATCH][SHMEM] Factor out sbi->free_inodes manipulations
Posted by Hugh Dickins on Tue, 27 Nov 2007 08:36:41 GMT
View Forum Message <> Reply to Message

On Mon, 26 Nov 2007, Andrew Morton wrote:
>
> It is peculair to (wrongly) return -ENOMEM
>
> > + if (shmem_reserve_inode(inode->i_sb))
> > + return -ENOSPC;
>
> and to then correct it in the caller..

Oops, I missed that completely.

> Something boringly conventional such as the below, perhaps?

Much better, thanks.

Hugh