Subject: [PATCH] ext3: ext3_symlink should use GFP_NOFS allocations inside
Posted by Kirill Korotaev on Fri, 10 Mar 2006 08:03:36 GMT
View Forum Message <> Reply to Message

Andrew,

Resending a patch with a separate helper as requested by Al Viro:

This patch fixes illegal __GFP_FS allocation inside ext3
transaction in ext3_symlink().
Such allocation may re-enter ext3 code from
try_to_free_pages. But JBD/ext3 code keeps a pointer to current
journal handle in task_struct and, hence, is not reentrable.

This bug led to "Assertion failure in journal_dirty_metadata()" messages.

http://bugzilla.openvz.org/show_bug.cgi?id=115

Signed-Off-By: Andrey Savochkin <saw@saw.sw.com.sg>
Signed-Off-By: Kirill Korotaev <dev@openvz.org>

Thanks,
Kirill
P.S. against 2.6.16-rc5

```
--- ./fs/ext3/namei.c.symlnkfix 2006-03-10 10:24:05.000000000 +0300
+++ ./fs/ext3/namei.c 2006-03-10 10:24:49.000000000 +0300
@@ -2141,7 +2141,7 @@ retry:
    * We have a transaction open.  All is sweetness.  It also sets
    * i_size in generic_commit_write().
    */
-  err = page_symlink(inode, symname, l);
+  err = __page_symlink(inode, symname, l, GFP_NOFS);
   if (err) {
    ext3_dec_count(handle, inode);
    ext3_mark_inode_dirty(handle, inode);
--- ./fs/namei.c.symlnkfix 2006-03-10 10:24:05.000000000 +0300
+++ ./fs/namei.c 2006-03-10 10:34:58.000000000 +0300
@@ -2613,13 +2613,16 @@ void page_put_link(struct dentry *dentry
  }
 }

-int page_symlink(struct inode *inode, const char *symname, int len)
+int __page_symlink(struct inode *inode, const char *symname, int len,
+  gfp_t gfp_mask)
 {
  struct address_space *mapping = inode->i_mapping;
- struct page *page = grab_cache_page(mapping, 0);
```

```
+ struct page *page;
  int err = -ENOMEM;
  char *kaddr;

+ page = find_or_create_page(mapping, 0,
+   mapping_gfp_mask(mapping) | gfp_mask);
  if (!page)
   goto fail;
  err = mapping->a_ops->prepare_write(NULL, page, 0, len-1);
@@ -2654,6 +2657,11 @@ fail:
  return err;
 }

+int page_symlink(struct inode *inode, const char *symname, int len)
+{
+ return __page_symlink(inode, symname, len, GFP_KERNEL);
+}
+
 struct inode_operations page_symlink_inode_operations = {
  .readlink = generic_readlink,
  .follow_link = page_follow_link_light,
@@ -2672,6 +2680,7 @@ EXPORT_SYMBOL(lookup_one_len);
 EXPORT_SYMBOL(page_follow_link_light);
 EXPORT_SYMBOL(page_put_link);
 EXPORT_SYMBOL(page_readlink);
+EXPORT_SYMBOL(__page_symlink);
 EXPORT_SYMBOL(page_symlink);
 EXPORT_SYMBOL(page_symlink_inode_operations);
 EXPORT_SYMBOL(path_lookup);
--- ./include/linux/fs.h.symlnkfix 2006-03-10 10:24:05.000000000 +0300
+++ ./include/linux/fs.h 2006-03-10 10:27:40.000000000 +0300
@@ -1669,6 +1669,8 @@ extern int vfs_follow_link(struct nameid
 extern int page_readlink(struct dentry *, char __user *, int);
 extern void *page_follow_link_light(struct dentry *, struct nameidata *);
 extern void page_put_link(struct dentry *, struct nameidata *, void *);
+extern int __page_symlink(struct inode *inode, const char *symname, int len,
+  gfp_t gfp_mask);
 extern int page_symlink(struct inode *inode, const char *symname, int len);
 extern struct inode_operations page_symlink_inode_operations;
 extern int generic_readlink(struct dentry *, char __user *, int);
```

Subject: Re: [PATCH] ext3: ext3_symlink should use GFP_NOFS allocations inside
Posted by Al Viro on Fri, 10 Mar 2006 08:31:36 GMT

View Forum Message <> Reply to Message

On Fri, Mar 10, 2006 at 11:08:29AM +0300, Kirill Korotaev wrote:
> + page = find_or_create_page(mapping, 0,

> +   mapping_gfp_mask(mapping) | gfp_mask);

> +int page_symlink(struct inode *inode, const char *symname, int len)
> +{
> + return __page_symlink(inode, symname, len, GFP_KERNEL);

s/GFP_KERNEL/0/; if somebody has e.g. GFP_NOFS in their mapping flags,
you end up breaking their code. We really pass extra flags to be added
to default ones; page_symlink() should pass 0.

Subject: Re: [PATCH] ext3: ext3_symlink should use GFP_NOFS allocations inside
Posted by dev on Fri, 10 Mar 2006 08:41:44 GMT
View Forum Message <> Reply to Message

>>+ page = find_or_create_page(mapping, 0,
>>+   mapping_gfp_mask(mapping) | gfp_mask);
>
>
>>+int page_symlink(struct inode *inode, const char *symname, int len)
>>+{
>>+ return __page_symlink(inode, symname, len, GFP_KERNEL);
>
>
> s/GFP_KERNEL/0/; if somebody has e.g. GFP_NOFS in their mapping flags,
> you end up breaking their code. We really pass extra flags to be added
> to default ones; page_symlink() should pass 0.
thanks for noticing this.
fixed and resend.

Thanks,
Kirill
