Subject: Re: [PATCH] AB-BA deadlock in drop_caches sysctl (resend, the one sent was for 2.6.18)
Posted by den on Tue, 04 Dec 2007 08:00:22 GMT
View Forum Message <> Reply to Message

```
Andrew Morton wrote:
> On Mon, 3 Dec 2007 16:52:47 +0300
> "Denis V. Lunev" <den@openvz.org> wrote:
>
>> There is a AB-BA deadlock regarding drop_caches sysctl. Here are the code
>> paths:
>>
>> drop_pagecache
>>   spin_lock(&inode_lock);
>>   invalidate_mapping_pages
>>     try_to_release_page
>>       ext3_releasepage
>>         journal_try_to_free_buffers
>>           __journal_try_to_free_buffer
>>      spin_lock(&journal->j_list_lock);
>>
>> __journal_temp_unlink_buffer (called under journal->j_list_lock by comments)
>>   mark_buffer_dirty
>>     __set_page_dirty
>>       __mark_inode_dirty
>>         spin_lock(&inode_lock);
>>
>> The patch tries to address the issue - it drops inode_lock before digging into
>> invalidate_inode_pages. This seems sane as inode hold should not gone from the
>> list and should not change its place.
>>
>> Signed-off-by: Denis V. Lunev <den@openvz.org>
>> --
>> diff --git a/fs/drop_caches.c b/fs/drop_caches.c
>> index 59375ef..4ac80d8 100644
>> --- a/fs/drop_caches.c
>> +++ b/fs/drop_caches.c
>> @@ -14,15 +14,27 @@ int sysctl_drop_caches;
>>
>>  static void drop_pagecache_sb(struct super_block *sb)
>>  {
>> - struct inode *inode;
>> + struct inode *inode, *old;
>>
>> + old = NULL;
>>   spin_lock(&inode_lock);
>>   list_for_each_entry(inode, &sb->s_inodes, i_sb_list) {
>>    if (inode->i_state & (I_FREEING|I_WILL_FREE))
>>     continue;
>> -  __invalidate_mapping_pages(inode->i_mapping, 0, -1, true);
>> +  __iget(inode);
>> +  spin_unlock(&inode_lock);
>> +
>> +  if (old != NULL)
>> +   iput(old);
>> +  invalidate_mapping_pages(inode->i_mapping, 0, -1);
>> +  old = inode;
>> +
>> +  spin_lock(&inode_lock);
>>   }
>>   spin_unlock(&inode_lock);
>> +
>> + if (old != NULL)
>> +  iput(old);
>>  }
>
```

> We need to hold onto inode_lock while walking sb->s_inodes. Otherwise the
> inode which we're currently looking at could get removed from i_sb_list and
> bad things will happen (drop_pagecache_sb will go infinite, or will oops, I
> guess).

as far as I understand, there are the following place removing inode
from i_sb_list:
- generic_delete_inode (via iput_final)
- generic_forget_inode (via iput_final)
- hugetlbfs_forget_inode
- dispose_list after the check under inode_lock for i_count

So, the patch is sane from disappearing point of view:
- I hold inode under inode_lock
- and iput it after new inode to clean has been found and hold

Nevertheless we'll think a bit about ext3 fix. Though other staff like
gfs2 etc can also be affected.

Regards,
 Den