Subject: [PATCH] AB-BA deadlock in drop_caches sysctl (resend, the one sent was for 2.6.18)
Posted by den on Mon, 03 Dec 2007 13:52:03 GMT
View Forum Message <> Reply to Message

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);
 }

 void drop_pagecache(void)
diff --git a/include/linux/fs.h b/include/linux/fs.h
index b3ec4a4..b424dff 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -1649,9 +1649,6 @@ extern int __invalidate_device(struct block_device *);
 extern int invalidate_partition(struct gendisk *, int);
 #endif
 extern int invalidate_inodes(struct super_block *);
-unsigned long __invalidate_mapping_pages(struct address_space *mapping,
-     pgoff_t start, pgoff_t end,
-     bool be_atomic);
 unsigned long invalidate_mapping_pages(struct address_space *mapping,
      pgoff_t start, pgoff_t end);

diff --git a/mm/truncate.c b/mm/truncate.c
index cadc156..5cbe1e7 100644
--- a/mm/truncate.c
+++ b/mm/truncate.c
@@ -266,8 +266,21 @@ void truncate_inode_pages(struct address_space *mapping, loff_t
lstart)
 }
 EXPORT_SYMBOL(truncate_inode_pages);

-unsigned long __invalidate_mapping_pages(struct address_space *mapping,
-    pgoff_t start, pgoff_t end, bool be_atomic)
+/**
+ * invalidate_mapping_pages - Invalidate all the unlocked pages of one inode
+ * @mapping: the address_space which holds the pages to invalidate
+ * @start: the offset 'from' which to invalidate
+ * @end: the offset 'to' which to invalidate (inclusive)
+ *
+ * This function only removes the unlocked pages, if you want to
+ * remove all the pages of one inode, you must call truncate_inode_pages.
```

```
+ *
+ * invalidate_mapping_pages() will not block on IO activity. It will not
+ * invalidate pages which are dirty, locked, under writeback or mapped into
+ * pagetables.
+ */
+unsigned long invalidate_mapping_pages(struct address_space *mapping,
+     pgoff_t start, pgoff_t end)
 {
  struct pagevec pvec;
  pgoff_t next = start;
@@ -308,30 +321,10 @@ unlock:
     break;
    }
    pagevec_release(&pvec);
-  if (likely(!be_atomic))
-   cond_resched();
+  cond_resched();
  }
  return ret;
 }
-
-/**
- * invalidate_mapping_pages - Invalidate all the unlocked pages of one inode
- * @mapping: the address_space which holds the pages to invalidate
- * @start: the offset 'from' which to invalidate
- * @end: the offset 'to' which to invalidate (inclusive)
- *
- * This function only removes the unlocked pages, if you want to
- * remove all the pages of one inode, you must call truncate_inode_pages.
- *
- * invalidate_mapping_pages() will not block on IO activity. It will not
- * invalidate pages which are dirty, locked, under writeback or mapped into
- * pagetables.
- */
-unsigned long invalidate_mapping_pages(struct address_space *mapping,
-     pgoff_t start, pgoff_t end)
-{
- return __invalidate_mapping_pages(mapping, start, end, false);
-}
 EXPORT_SYMBOL(invalidate_mapping_pages);

 /*
```

Subject: Re: [PATCH] AB-BA deadlock in drop_caches sysctl (resend, the one sent was for 2.6.18)
Posted by akpm on Mon, 03 Dec 2007 19:01:43 GMT

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).

drop_caches is bad this way - it has a couple of ranking errors. A suitable fix would be to remove the drop_caches feature, but it seems to be fairly popular as a developer thing. The approach thus far has been "yeah, sorry about that, but drop_caches is only for development and it is root-only anyway".

We could fix this particular issue by changing JBD to run mark_inode_dirty() outside list_lock (which would be a good change independent of the drop_caches issue) but other problems with drop_caches will remain.

One way to fix jbd (and jbd2) would be:

```
static void __journal_temp_unlink_buffer(struct journal_head *jh,
     struct buffer_head **bh_to_dirty)
{
 *bh_to_dirty = NULL;
 ...
 if (test_clear_buffer_jbddirty(bh))
  *bh_to_dirty = bh;
}

{
 struct buffer_head *bh_to_dirty; /* probably needs uninitialized_var() */

 ...
 __journal_temp_unlink_buffer(jh, &bh_to_dirty);
 ...
 jbd_mark_buffer_dirty(bh_to_dirty);
 brelse(bh_to_dirty);
```

 ...
}

static inline void jbd_mark_buffer_dirty(struct buffer_head *bh)
{
 if (bh)
  mark_buffer_dirty(bh);
}

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,
```