Subject: Re: [PATCH] incorrect error handling inside generic_file_direct_write
Posted by Dmitriy Monakhov on Tue, 12 Dec 2006 10:18:38 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> On Tue, 12 Dec 2006 15:20:52 +0300
> Dmitriy Monakhov <dmonakhov@sw.ru> wrote:
>
>> > XFS (at least) can call generic_file_direct_write() with i_mutex not held.
>> > And vmtruncate() expects i_mutex to be held.
>> >
>> > I guess a suitable solution would be to push this problem back up to the
>> > callers: let them decide whether to run vmtruncate() and if so, to ensure
>> > that i_mutex is held.
>> >
>> > The existence of generic_file_aio_write_nolock() makes that rather messy
>> > though.
>> This means we may call generic_file_aio_write_nolock() without i_mutex, right?
>> but call trace is :
>>   generic_file_aio_write_nolock()
>>   ->generic_file_buffered_write() /* i_mutex not held here */
>> but according to filemaps locking rules: mm/filemap.c:77
>>  ..
>>  *  ->i_mutex   (generic_file_buffered_write)
>>  *    ->mmap_sem  (fault_in_pages_readable->do_page_fault)
>>  ..
>> I'm confused a litle bit, where is the truth?
>
> xfs_write() calls generic_file_direct_write() without taking i_mutex for
> O_DIRECT writes.
Yes, but my quastion is about __generic_file_aio_write_nolock().
As i understand _nolock sufix means that i_mutex was already locked
by caller, am i right ?
If yes, than __generic_file_aio_write_nolock() is beter place for vmtrancate()
acclivity after generic_file_direct_write() has fail.
Signed-off-by: Dmitriy Monakhov <dmonakhov@openvz.org>
-------

diff --git a/mm/filemap.c b/mm/filemap.c
index 7b84dc8..723d2ca 100644
--- a/mm/filemap.c
+++ b/mm/filemap.c
@@ -2282,6 +2282,15 @@ __generic_file_aio_write_nolock(struct k

   written = generic_file_direct_write(iocb, iov, &nr_segs, pos,
       ppos, count, ocount);
+  if (written < 0) {

```
+   loff_t isize = i_size_read(inode);
+   /*
+    * generic_file_direct_write() may have instantiated
+    * a few blocks outside i_size. Trim these off again.
+    */
+   if (pos + count > isize)
+    vmtruncate(inode, isize);
+ }
   if (written < 0 || written == count)
    goto out;
   /*
```

---