Subject: Re: [PATCH] incorrect direct io error handling
Posted by Dmitriy Monakhov on Tue, 19 Dec 2006 06:31:15 GMT
View Forum Message <> Reply to Message

"Chen, Kenneth W" <kenneth.w.chen@intel.com> writes:

> Dmitriy Monakhov wrote on Monday, December 18, 2006 5:23 AM
>> This patch is result of discussion started week ago here:
>> http://lkml.org/lkml/2006/12/11/66
>> changes from original patch:
>> - Update wrong comments about i_mutex locking.
>> - Add BUG_ON(!mutex_is_locked(..)) for non blkdev.
>> - vmtruncate call only for non blockdev
>> LOG:
>> If generic_file_direct_write() has fail (ENOSPC condition) inside
>> __generic_file_aio_write_nolock() it may have instantiated
>> a few blocks outside i_size. And fsck will complain about wrong i_size
>> (ext2, ext3 and reiserfs interpret i_size and biggest block difference as error),
>> after fsck will fix error i_size will be increased to the biggest block,
>> but this blocks contain gurbage from previous write attempt, this is not
>> information leak, but its silence file data corruption. This issue affect
>> fs regardless the values of blocksize or pagesize.
>> We need truncate any block beyond i_size after write have failed , do in simular
>> generic_file_buffered_write() error path. If host is !S_ISBLK i_mutex always
>> held inside generic_file_aio_write_nolock() and we may safely call vmtruncate().
>> Some fs (XFS at least) may directly call generic_file_direct_write()with
>> i_mutex not held. There is no general scenario in this case. This fs have to
>> handle generic_file_direct_write() error by its own specific way (place).
>
>
> I'm puzzled that if ext2 is able to instantiate some blocks, then why does it
> return no space error? Where is the error coming from?
generic_file_aio_write_nolock()
 ->generic_file_direct_write()
   ->generic_file_direct_IO()
     ->ext2_direct_IO(WRITE,...)
       ->blockdev_direct_IO(....,ext2_get_block,...)