Subject: [PATCH 2/2] incorrect direct io error handling (v7)
Posted by Dmitriy Monakhov on Mon, 12 Mar 2007 20:19:16 GMT
View Forum Message <> Reply to Message

Changes against v6:
 - Handle direct_io failure inside generic_file_direct_write() as it was
   recommend by Andrew (during discussion v1), and by Nick (during
   discussion v6).
 - change comments, make it more clear.
 - one more time check what __generic_file_aio_write_nolock() always called
   under i_mutex for non blkdev files.
Tested with: fsstress, manual direct_io tests

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, and off corse only for non
blkdev files.
We need truncate any block beyond i_size after write have failed , do in simular
generic_file_buffered_write() error path. We may safely call vmtruncate() here
because i_mutex always held for non blkdev files.

TEST_CASE:
open("/mnt/test/BIG_FILE", O_WRONLY|O_CREAT|O_DIRECT, 0666) = 3
write(3, "aaaaaaaaaaaaaaa"..., 104857600) = -1 ENOSPC (No space left on device)

#stat /mnt/test/BIG_FILE
  File: `/mnt/test/BIG_FILE'
  Size: 0   Blocks: 110896   IO Block: 1024   regular empty file
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<^^^^^^^^file size is less than biggest block idx

Device: fe07h/65031d   Inode: 14   Links: 1
Access: (0644/-rw-r--r--) Uid: (0/   root)   Gid: (0/   root)
Access: 2007-01-24 20:03:38.000000000 +0300
Modify: 2007-01-24 20:03:38.000000000 +0300
Change: 2007-01-24 20:03:39.000000000 +0300

#fsck.ext3 -f /dev/VG/test
e2fsck 1.39 (29-May-2006)
Pass 1: Checking inodes, blocks, and sizes
Inode 14, i_size is 0, should be 56556544. Fix<y>? yes
Pass 2: Checking directory structure
....

```
Signed-off-by: Monakhov Dmitriy <dmonakhov@openvz.org>
---
 mm/filemap.c |   28 ++++++++++++++++++++++++----
 1 files changed, 24 insertions(+), 4 deletions(-)

diff --git a/mm/filemap.c b/mm/filemap.c
index 8bd1ea4..95d49fe 100644
--- a/mm/filemap.c
+++ b/mm/filemap.c
@@ -1932,8 +1932,10 @@ generic_file_direct_write(struct kiocb *iocb, const struct iovec *iov,
  /*
   * Sync the fs metadata but not the minor inode changes and
   * of course not the data as we did direct DMA for the IO.
-  * i_mutex is held, which protects generic_osync_inode() from
-  * livelocking.  AIO O_DIRECT ops attempt to sync metadata here.
+  * i_mutex is held in case of DIO_LOCKING, which protects
+  * generic_osync_inode() from livelocking. If it is not held, then
+  * the filesystem must prevent this livelock. AIO O_DIRECT ops
+  * attempt to sync metadata here.
   */
  if ((written >= 0 || written == -EIOCBQUEUED) &&
      ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
@@ -2155,8 +2157,26 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct iovec
*iov,
   loff_t endbyte;
   ssize_t written_buffered;

+  /*
+   * In case of non blockdev we may fail to buffered I/O.
+   * So i_mutex must be held.
+   */
+  if (!S_ISBLK(inode->i_mode))
+   BUG_ON(!mutex_is_locked(&inode->i_mutex));
+
   written = generic_file_direct_write(iocb, iov, &nr_segs, pos,
        ppos, count, ocount);
+  /*
+   * If host is not S_ISBLK generic_file_direct_write() may
+   * have instantiated a few blocks outside i_size  files
+   * Trim these off again.
+   */
+  if (unlikely(written < 0) && !S_ISBLK(inode->i_mode)) {
+   loff_t isize = i_size_read(inode);
+   if (pos + count > isize)
+    vmtruncate(inode, isize);
+  }
```

```
+
   if (written < 0 || written == count)
    goto out;
   /*
@@ -2261,8 +2281,8 @@ ssize_t generic_file_aio_write(struct kiocb *iocb, const struct iovec
*iov,
 EXPORT_SYMBOL(generic_file_aio_write);

 /*
- * Called under i_mutex for writes to S_ISREG files.   Returns -EIO if something
- * went wrong during pagecache shootdown.
+ * Called under i_mutex for writes to S_ISREG files in case of DIO_LOCKING.
+ * Returns -EIO if something went wrong during pagecache shootdown.
  */
 static ssize_t
 generic_file_direct_IO(int rw, struct kiocb *iocb, const struct iovec *iov,
--
1.5.0.1
```