

Subject: [PATCH 2/2] fs: incorrect direct io error handling v8
Posted by [Dmitriy Monakhov](#) on Mon, 19 Mar 2007 07:48:54 GMT
[View Forum Message](#) <> [Reply to Message](#)

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` in case of non blockdev files. At least ext2, ext3 and reiserfs interpret `i_size` and biggest block difference as error. Later fsck will complain about wrong `i_size`. After fsck will fix error `i_size` will be increased to the biggest block. This is bad because this blocks contain gurbage from previous write attempt. And result in silence file data corruption. This issue affect fs regardless the values of blocksize or pagesize. We need truncate any block beyond `i_size` after `generic_file_direct_write()` has failed. In fact all existing fs witch use `__generic_file_aio_write_nolock()` always call it under `i_mutex` for non blockdev files. This patch add correcspnding `BUG_ON()` in order to explicitly demand/document it.

Also fix out of date direct_io locking comments.

TEST CASE:

####TESTCASE BEGIN

```
$touch /mnt/test/BIG FILE
```

```
## at this moment /mnt/test/BIG_FILE size and blocks equal to zero
```

```
open("/mnt/test/BIG_FILE", O_WRONLY|O_CREAT|O_DIRECT, 0666) = 3
```

```
write(3, "aaaaaaaaaaaaa"..., 104857600) = -1 ENOSPC (No space left on device)
```

```
## size and block sould't be changed because write op failed.
```

```
$stat /mnt/test/BIG_FILE
```

File: `/mnt/test/BIG_FILE'

Size: 0 Blocks: 110896 IO Block: 1024 regular empty file

[illegible]

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

□ □ □ □

#####TESTCASE END

Signed-off-by: Dmitriy Monakhov <dmonakhov@openvz.org>

...

mm/filemap.c | 24 ++++++-----

1 files changed, 20 insertions(+), 4 deletions(-)

diff --git a/mm/filemap.c b/mm/filemap.c

index bbef42f..a08900e 100644

--- a/mm/filemap.c

+++ b/mm/filemap.c

@@ -1892,8 +1892,10 @@ 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))) {

@@ -2083,6 +2085,9 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct iovec
*iov,

ssize_t written;

ssize_t err;

+ /* Always called under i_mutex for writes to non blockdev files */

+ BUG_ON(!S_ISBLK(inode->i_mode) &&

+ !mutex_is_locked(&inode->i_mutex));

ocount = 0;

err = generic_iovec_checks(iov, &nr_segs, &ocount, VERIFY_READ);

if (err)

@@ -2117,6 +2122,17 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct iovec
*iov,

written = generic_file_direct_write(iocb, iov, &nr_segs, pos,

ppos, count, ocount);

+ /*

+ * If host is not blockdev 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;

```
/*  
@@ -2221,8 +2237,8 @@ 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 shutdown.  
+ * Called under i_mutex for writes to S_ISREG files in case of DIO_LOCKING.  
+ * Returns -EIO if something went wrong during pagecache shutdown.  
*/
```

```
static ssize_t  
generic_file_direct_IO(int rw, struct kiocb *iocb, const struct iovec *iov,  
--
```

1.4.4.2
