

Subject: [PATCH][RFC] incorrect direct io error handling (v3)
Posted by [Dmitriy Monakhov](#) on Wed, 24 Jan 2007 19:04:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

incorrect direct io error handling (v3)

Changes from v2:

- Remove BUG_ON(!mutex_is_locked(..)) for non blkdev.
- vmtruncate() called from generic_file_aio_write().
- depends on patch titled:
[PATH][RFC] mm: Move common segments checks to separate function

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 similar generic_file_buffered_write() error path. Initially i've proposed do it in __generic_file_aio_write_nolock() with explicit guarantee i_mutex always held, but not everybody was agree with it. So we may safely call vmtruncate() inside generic_file_aio_write(), here i_mutex already locked.

TEST_CASE:

```
open("/mnt/test/BIG_FILE", O_WRONLY|O_CREAT|O_DIRECT, 0666) = 3
write(3, "aaaaaaaaaaaaaaaa"..., 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: Dmitriy Monakhov <dmonakhov@openvz.org>

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

index d01abb6..96840e5 100644

--- a/mm/filemap.c

+++ b/mm/filemap.c

@@ -2058,8 +2058,9 @@ generic_file_direct_write(struct kiocb *
/*

* 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 may not being held, if so some specific locking
+ * ordering must protect generic_osync_inode() from livelocking.
+ * AIO O_DIRECT ops attempt to sync metadata here.
*/

if ((written >= 0 || written == -EIOCBQUEUED) &&
((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
@@ -2365,6 +2366,17 @@ ssize_t generic_file_aio_write(struct ki
 &iocb->ki_pos);
 mutex_unlock(&inode->i_mutex);

+ if (unlikely(ret < 0 && (file->f_flags & O_DIRECT))) {
+ ssize_t cnt = generic_segment_checks(nr_segs, iov, VERIFY_READ);
+ 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 (cnt > 0 && (pos + cnt > isize))
+ vmtruncate(inode, isize);
+ }
+
if (ret > 0 && ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
 ssize_t err;

@@ -2377,8 +2389,8 @@ ssize_t generic_file_aio_write(struct ki
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.
+ * May be called without i_mutex for writes to S_ISREG files.
+ * 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,

[View Forum Message](#) <> [Reply to Message](#)

Dmitriy Monakhov <dmonakhov@sw.ru> wrote:

[illegible]

```

> 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: Dmitry Monakhov <dmonakhov@openvz.org>
> -----
>

```

```

diff --git a/mm/filemap.c b/mm/filemap.c
> index d01abb6..96840e5 100644
> --- a/mm/filemap.c
> +++ b/mm/filemap.c
> @@ -2058,8 +2058,9 @@ generic_file_direct_write(struct kiocb *
> /*
>  * 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 may not being held, if so some specific locking
> + * ordering must protect generic_osync_inode() from livelocking.
> + * AIO O_DIRECT ops attempt to sync metadata here.
>  */
> if ((written >= 0 || written == -EIOCBQUEUED) &&
>     ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
> @@ -2365,6 +2366,17 @@ ssize_t generic_file_aio_write(struct ki
>     &iocb->ki_pos);
>     mutex_unlock(&inode->i_mutex);
>
> + if (unlikely(ret < 0 && (file->f_flags & O_DIRECT))) {
> +     ssize_t cnt = generic_segment_checks(nr_segs, iov, VERIFY_READ);
> +     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 (cnt > 0 && (pos + cnt > isize))
> +         vmtruncate(inode, isize);
> + }
> +

```

vmtruncate() really wants i_mutex to be held. Can't we do that here?

Subject: Re: [PATCH][RFC] incorrect direct io error handling (v3)
 Posted by [Dmitriy Monakhov](#) on Fri, 26 Jan 2007 07:59:30 GMT
[View Forum Message](#) <> [Reply to Message](#)

Andrew Morton <akpm@osdl.org> writes:

> On Wed, 24 Jan 2007 22:05:06 +0300

> Dmitriy Monakhov <dmonakhov@sw.ru> wrote:

>

>> incorrect direct io error handling (v3)

>> Changes from v2:

>> - Remove BUG_ON(!mutex_is_locked(..)) for non blkdev.

>> - vmtruncate() called from generic_file_aio_write().

>> - depends on patch titled:

>> [PATH][RFC] mm: Move common segments checks to separate function

>

> drat, I skipped that patch due to rejects, and because Nick is working on

> things in the same area.

>

[skip]

>> if ((written >= 0 || written == -EIOCBQUEUED) &&

>> ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {

>> @@ -2365,6 +2366,17 @@ ssize_t generic_file_aio_write(struct ki

>> &iocb->ki_pos);

>> mutex_unlock(&inode->i_mutex);

>>

>> + if (unlikely(ret < 0 && (file->f_flags & O_DIRECT))) {

>> + ssize_t cnt = generic_segment_checks(nr_segs, iov, VERIFY_READ);

>> + 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 (cnt > 0 && (pos + cnt > isize))

>> + vmtruncate(inode, isize);

>> + }

>> +

>

> vmtruncate() really wants i_mutex to be held. Can't we do that here?

Yepp 110% true, baaahh it looks like my brain wasn't clear at the time i wrote this. We have to do vmtruncate() before dropping i_mutex , right after __generic_file_aio_write_nolock() call , i'm sorry to waste your time.