
Subject: RE: [PATCH] incorrect error handling inside generic_file_direct_write
Posted by [kenneth.w.chen](#) on Wed, 13 Dec 2006 02:43:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Andrew Morton wrote on Tuesday, December 12, 2006 2:40 AM
> On Tue, 12 Dec 2006 16:18:32 +0300
> Dmitriy Monakhov <dmonakhov@sw.ru> wrote:
>
> > > 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 little bit, where is the truth?
> > >
> > > xfs_write() calls generic_file_direct_write() without taking i_mutex for
> > > O_DIRECT writes.
> > Yes, but my question is about __generic_file_aio_write_nolock().
> > As I understand _nolock suffix means that i_mutex was already locked
> > by caller, am I right ?
>
> Nope. It just means that __generic_file_aio_write_nolock() doesn't take
> the lock. We don't assume or require that the caller took it. For example
> the raw driver calls generic_file_aio_write_nolock() without taking
> i_mutex. Raw isn't relevant to the problem (although ocfs2 might be). But
> we cannot assume that all callers have taken i_mutex, I think.

I think we should also clean up generic_file_aio_write_nolock. This was brought up a couple of weeks ago and I gave up too early. Here is my second attempt.

How about the following patch, I think we can kill generic_file_aio_write_nolock and merge both *file_aio_write_nolock into one function, then

```
generic_file_aio_write
ocfs2_file_aio_write
blk_dev->aio_write
```

all points to a non-lock version of __generic_file_aio_write(). First two already hold i_mutex, while the block device's aio_write method doesn't require i_mutex to be held.

Signed-off-by: Ken Chen <kenneth.w.chen@intel.com>

```
diff -Nurp linux-2.6.19/drivers/char/raw.c linux-2.6.19.ken/drivers/char/raw.c
--- linux-2.6.19/drivers/char/raw.c 2006-11-29 13:57:37.000000000 -0800
```

```

+++ linux-2.6.19.ken/drivers/char/raw.c 2006-12-12 16:41:39.000000000 -0800
@@ -242,7 +242,7 @@ static const struct file_operations raw_
    .read = do_sync_read,
    .aio_read = generic_file_aio_read,
    .write = do_sync_write,
-   .aio_write = generic_file_aio_write_nolock,
+   .aio_write = __generic_file_aio_write,
    .open = raw_open,
    .release = raw_release,
    .ioctl = raw_ioctl,
diff -Nurp linux-2.6.19/fs/block_dev.c linux-2.6.19.ken/fs/block_dev.c
--- linux-2.6.19/fs/block_dev.c 2006-11-29 13:57:37.000000000 -0800
+++ linux-2.6.19.ken/fs/block_dev.c 2006-12-12 16:47:58.000000000 -0800
@@ -1198,7 +1198,7 @@ const struct file_operations def_blk_fop
    .read = do_sync_read,
    .write = do_sync_write,
    .aio_read = generic_file_aio_read,
-   .aio_write = generic_file_aio_write_nolock,
+   .aio_write = __generic_file_aio_write,
    .mmap = generic_file_mmap,
    .fsync = block_fsync,
    .unlocked_ioctl = block_ioctl,
diff -Nurp linux-2.6.19/fs/ocfs2/file.c linux-2.6.19.ken/fs/ocfs2/file.c
--- linux-2.6.19/fs/ocfs2/file.c 2006-11-29 13:57:37.000000000 -0800
+++ linux-2.6.19.ken/fs/ocfs2/file.c 2006-12-12 16:42:09.000000000 -0800
@@ -1107,7 +1107,7 @@ static ssize_t ocfs2_file_aio_write(stru
    /* communicate with ocfs2_dio_end_io */
    ocfs2_iocb_set_rw_locked(iocb);

-   ret = generic_file_aio_write_nolock(iocb, iov, nr_segs, iocb->ki_pos);
+   ret = __generic_file_aio_write(iocb, iov, nr_segs, iocb->ki_pos);

    /* buffered aio wouldn't have proper lock coverage today */
    BUG_ON(ret == -EIOCBQUEUED && !(filp->f_flags & O_DIRECT));
diff -Nurp linux-2.6.19/include/linux/fs.h linux-2.6.19.ken/include/linux/fs.h
--- linux-2.6.19/include/linux/fs.h 2006-11-29 13:57:37.000000000 -0800
+++ linux-2.6.19.ken/include/linux/fs.h 2006-12-12 16:41:58.000000000 -0800
@@ -1742,7 +1742,7 @@ extern int file_send_actor(read_descript
int generic_write_checks(struct file *file, loff_t *pos, size_t *count, int isblk);
extern ssize_t generic_file_aio_read(struct kiocb *, const struct iovec *, unsigned long, loff_t);
extern ssize_t generic_file_aio_write(struct kiocb *, const struct iovec *, unsigned long, loff_t);
-extern ssize_t generic_file_aio_write_nolock(struct kiocb *, const struct iovec *,
+extern ssize_t __generic_file_aio_write(struct kiocb *, const struct iovec *,
    unsigned long, loff_t);
extern ssize_t generic_file_direct_write(struct kiocb *, const struct iovec *,
    unsigned long *, loff_t, loff_t *, size_t, size_t);
diff -Nurp linux-2.6.19/mm/filemap.c linux-2.6.19.ken/mm/filemap.c
--- linux-2.6.19/mm/filemap.c 2006-11-29 13:57:37.000000000 -0800

```

```

+++ linux-2.6.19.ken/mm/filemap.c 2006-12-12 16:47:58.000000000 -0800
@@ -2219,9 +2219,9 @@ zero_length_segment:
}
EXPORT_SYMBOL(generic_file_buffered_write);

-static ssize_t
-__generic_file_aio_write_nolock(struct kiocb *iocb, const struct iovec *iov,
- unsigned long nr_segs, loff_t *ppos)
+ssize_t
+__generic_file_aio_write(struct kiocb *iocb, const struct iovec *iov,
+ unsigned long nr_segs, loff_t pos)
{
    struct file *file = iocb->ki_filp;
    struct address_space * mapping = file->f_mapping;
@@ -2229,9 +2229,10 @@ __generic_file_aio_write_nolock(struct k
    size_t count; /* after file limit checks */
    struct inode *inode = mapping->host;
    unsigned long seg;
- loff_t pos;
+ loff_t *ppos = &iocb->ki_pos;
    ssize_t written;
    ssize_t err;
+ ssize_t ret;

    ocount = 0;
    for (seg = 0; seg < nr_segs; seg++) {
@@ -2254,7 +2255,6 @@ __generic_file_aio_write_nolock(struct k
    }

    count = ocount;
- pos = *ppos;

    vfs_check_frozen(inode->i_sb, SB_FREEZE_WRITE);

@@ -2332,32 +2332,16 @@ __generic_file_aio_write_nolock(struct k
    }
out:
    current->backing_dev_info = NULL;
- return written ? written : err;
-}
-
-ssize_t generic_file_aio_write_nolock(struct kiocb *iocb,
- const struct iovec *iov, unsigned long nr_segs, loff_t pos)
-{
- struct file *file = iocb->ki_filp;
- struct address_space *mapping = file->f_mapping;
- struct inode *inode = mapping->host;
- ssize_t ret;

```

```

-
- BUG_ON(iocb->ki_pos != pos);
-
- ret = __generic_file_aio_write_nolock(iocb, iov, nr_segs,
-   &iocb->ki_pos);
+ ret = written ? written : err;

    if (ret > 0 && ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
-   ssize_t err;
-
-   err = sync_page_range_nolock(inode, mapping, pos, ret);
-   if (err < 0)
-       ret = err;
-   }
-   return ret;
-   }
-EXPORT_SYMBOL(generic_file_aio_write_nolock);
+EXPORT_SYMBOL(__generic_file_aio_write);

ssize_t generic_file_aio_write(struct kiocb *iocb, const struct iovec *iov,
    unsigned long nr_segs, loff_t pos)
@@ -2370,8 +2354,7 @@ ssize_t generic_file_aio_write(struct ki
    BUG_ON(iocb->ki_pos != pos);

    mutex_lock(&inode->i_mutex);
-   ret = __generic_file_aio_write_nolock(iocb, iov, nr_segs,
-   &iocb->ki_pos);
+   ret = __generic_file_aio_write(iocb, iov, nr_segs, pos);
    mutex_unlock(&inode->i_mutex);

    if (ret > 0 && ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {

```
