

---

Subject: RE: [PATCH] incorrect error handling inside generic\_file\_direct\_write  
Posted by [kenneth.w.chen](#) on Fri, 15 Dec 2006 18:53:18 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Christoph Hellwig wrote on Friday, December 15, 2006 2:44 AM

```
> So we're doing the sync_page_range once in __generic_file_aio_write
> with i_mutex held.
>
>
> > 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))) {
>
> And then another time after it's unlocked, this seems wrong.
```

I didn't invent that mess though.

I should've ask the question first: in 2.6.20-rc1, generic\_file\_aio\_write will call sync\_page\_range twice, once from \_\_generic\_file\_aio\_write\_nolock and once within the function itself. Is it redundant? Can we delete the one in the top level function? Like the following?

```
--- ./mm/filemap.c.orig 2006-12-15 09:02:58.000000000 -0800
+++ ./mm/filemap.c 2006-12-15 09:03:19.000000000 -0800
@@ -2370,14 +2370,6 @@ ssize_t generic_file_aio_write(struct ki
     ret = __generic_file_aio_write_nolock(iocb, iov, nr_segs,
        &iocb->ki_pos);
     mutex_unlock(&inode->i_mutex);
-
- if (ret > 0 && ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
-     ssize_t err;
-
-     err = sync_page_range(inode, mapping, pos, ret);
-     if (err < 0)
-         ret = err;
- }
    return ret;
}
EXPORT_SYMBOL(generic_file_aio_write);
```

---