

---

Subject: Re: [PATCH] incorrect error handling inside generic\_file\_direct\_write  
Posted by [Andrew Morton](#) on Tue, 12 Dec 2006 09:52:32 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 12 Dec 2006 15:20:52 +0300  
Dmitriy Monakhov <dmonakhov@sw.ru> wrote:

> > XFS (at least) can call generic\_file\_direct\_write() with i\_mutex not held.  
> > And vmtruncate() expects i\_mutex to be held.  
> >  
> > I guess a suitable solution would be to push this problem back up to the  
> > callers: let them decide whether to run vmtruncate() and if so, to ensure  
> > that i\_mutex is held.  
> >  
> > The existence of generic\_file\_aio\_write\_nolock() makes that rather messy  
> > though.  
> This means we may call generic\_file\_aio\_write\_nolock() without i\_mutex, right?  
> but call trace is :  
> generic\_file\_aio\_write\_nolock()  
> ->generic\_file\_buffered\_write() /\* i\_mutex not held here \*/  
> 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.

---