Subject: Re: [PATCH 2/2] mm: incorrect direct io error handling (v6)
Posted by Nick Piggin on Mon, 12 Mar 2007 12:14:47 GMT
View Forum Message <> Reply to Message

On Mon, Mar 12, 2007 at 12:23:00PM +0300, Dmitriy Monakhov wrote:
> Nick Piggin <npiggin@suse.de> writes:
>
> > On Mon, Mar 12, 2007 at 11:55:30AM +0300, Dmitriy Monakhov wrote:
> >> Nick Piggin <npiggin@suse.de> writes:
> >>
> >> > On Mon, Mar 12, 2007 at 10:58:10AM +0300, Dmitriy Monakhov wrote:
> > >
> >> >> @@ -2240,6 +2241,29 @@ ssize_t generic_file_aio_write(struct kiocb *iocb, const struct iovec *iov,
> >> >> mutex_lock(&inode->i_mutex);
> >> >> ret = __generic_file_aio_write_nolock(iocb, iov, nr_segs,
> >> >> &iocb->ki_pos);
> >> >> + /*
> >> >> + * If __generic_file_aio_write_nolock has failed.
> >> >> + * This may happen because of:
> >> >> + * 1) Bad segment found (failed before actual write attempt)
> >> >> + * 2) Segments are good, but actual write operation failed
> >> >> + *   and may have instantiated a few blocks outside i_size.
> >> >> + *   a) in case of buffered write these blocks was already
> >> >> + *   trimmed by generic_file_buffered_write()
> >> >> + *   b) in case of O_DIRECT these blocks weren't trimmed yet.
> >> >> + *
> >> >> + * In case of (2b) these blocks have to be trimmed off again.
> >> >> + */
> >> >> + if (unlikely(ret < 0 && file->f_flags & O_DIRECT)) {
> >> >> + unsigned long nr_segs_avail = nr_segs;
> >> >> + size_t count = 0;
> >> >> + if (!generic_segment_checks(iov, &nr_segs_avail, &count,
> >> >> +   VERIFY_READ)) {
> >> >> +   /*It is (2b) case, because segments are good*/
> >> >> +   loff_t isize = i_size_read(inode);
> >> >> +   if (pos + count > isize)
> >> >> +   vmtruncate(inode, isize);
> >> >> + }
> >> >> + }
> > >
> >> > OK, but wouldn't this be better to be done in the actual direct IO
> >> > functions themselves? Thus you could be sure that you have the 2b case,
> >> > and the code would be less fragile to something changing?
> >> Ohh, We can't just call vmtruncate() after generic_file_direct_write()
> >> failure while __generic_file_aio_write_nolock() becase where is no guarantee
> >> what i_mutex held. In fact all existing fs always invoke
> >> __generic_file_aio_write_nolock() with i_mutex held in case of S_ISREG files,

> >> but this was't explicitly demanded and documented. I've proposed to do it in
> >> previous versions of this patch, because it this just document current state
> >> of affairs, but David Chinner wasn't agree with it.
> >
> > It seemed like it was documented in the comments that you altered in this
> > patch...
> >
> > How would such a filesystem that did not hold i_mutex propose to fix the
> > problem?
> >
> > The burden should be on those filesystems that might not want to hold
> > i_mutex here, to solve the problem nicely, rather than generic code to take
> > this ugly code.
> Ok then what do you think about this version http://lkml.org/lkml/2006/12/18/103
> witch was posted almost month ago :)

That seems better, but people might take issue with the fact that it has
to make the check for S_ISREG files. I don't know... people with more
knowledge of the vfs+fs side of things might have better input.
