Subject: Re: [PATCH] incorrect direct io error handling
Posted by David Chinner on Wed, 20 Dec 2006 14:26:31 GMT
View Forum Message <> Reply to Message

On Tue, Dec 19, 2006 at 09:07:12AM +0300, Dmitriy Monakhov wrote:
> David Chinner <dgc@sgi.com> writes:
> > On Mon, Dec 18, 2006 at 04:22:44PM +0300, Dmitriy Monakhov wrote:
> >> diff --git a/mm/filemap.c b/mm/filemap.c
> >> index 8332c77..7c571dd 100644
> >> --- a/mm/filemap.c
> >> +++ b/mm/filemap.c

<snip stuff>

> > You comment in the first hunk that i_mutex may not be held here,
> > but there's no comment in __generic_file_aio_write_nolock() that the
> > i_mutex must be held for !S_ISBLK devices.
> Any one may call directly call generic_file_direct_write() with i_mutex not held.

Only block devices based on the implementation (i.e. buffered I/O is
done here). but one can't call vmtruncate without the i_mutex held,
so if a filesystem is calling generic_file_direct_write() it won't
be able to use __generic_file_aio_write_nolock() without the i_mutex
held (because it can right now if it doesn't need the buffered I/O
fallback path), then

> >
> >> @@ -2341,6 +2353,13 @@ ssize_t generic_file_aio_write_nolock(st
> >> ssize_t ret;
> >>
> >> BUG_ON(iocb->ki_pos != pos);
> >> +/*
> >> + * generic_file_buffered_write() may be called inside
> >> + * __generic_file_aio_write_nolock() even in case of
> >> + * O_DIRECT for non S_ISBLK files. So i_mutex must be held.
> >> + */
> >> +if (!S_ISBLK(inode->i_mode))
> >> + BUG_ON(!mutex_is_locked(&inode->i_mutex));
> >>
> >> ret = __generic_file_aio_write_nolock(iocb, iov, nr_segs,
> >> &iocb->ki_pos);
> >
> > I note that you comment here in generic_file_aio_write_nolock(),
> > but it's not immediately obvious that this is refering to the
> > vmtruncate() call in __generic_file_aio_write_nolock().
> This is not about vmtruncate(). __generic_file_aio_write_nolock() may
> call generic_file_buffered_write() even in case of O_DIRECT for !S_ISBLK, and

No, the need for i_mutex is currently dependent on doing direct I/O
and the return value from generic_file_buffered_write().
A filesystem that doesn't fall back to buffered I/O (e.g. XFS) can currently
use generic_file_aio_write_nolock() without needing to hold i_mutex.

Your change prevents that by introducing a vmtruncate() before the
generic_file_buffered_write() return value check, which means that a
filesystem now _must_ hold the i_mutex when calling
generic_file_aio_write_nolock() even when it doesn't do buffered I/O
through this path.

> generic_file_buffered_write() has documented locking rules (i_mutex held).
> IMHO it is important to explicitly document this . And after we realize
> that i_mutex always held, vmtruncate() may be safely called.

I don't think changing the locking semantics of
generic_file_aio_write_nolock() to require a lock for all
filesystem-based users is a good way to fix a filesystem specific
direct I/O problem which can be easily fixed in filesystem specific
code - i.e. call vmtruncate() in ext3_file_write() on failure....

Cheers,

Dave.
--
Dave Chinner
Principal Engineer
SGI Australian Software Group
