
Subject: Re: [PATCH] incorrect direct io error handling
Posted by [David Chinner](#) on Mon, 18 Dec 2006 22:15:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

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
> @@ -2044,8 +2044,9 @@ generic_file_direct_write(struct kiocb *
> /*
>  * Sync the fs metadata but not the minor inode changes and
>  * of course not the data as we did direct DMA for the IO.
> - * i_mutex is held, which protects generic_osync_inode() from
> - * livelocking. AIO O_DIRECT ops attempt to sync metadata here.
> + * i_mutex may not being held (XFS does this), if so some specific locking
> + * ordering must protect generic_osync_inode() from livelocking.
> + * AIO O_DIRECT ops attempt to sync metadata here.
> */
> if ((written >= 0 || written == -EIOCBQUEUED) &&
>     ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
> @@ -2279,6 +2280,17 @@ __generic_file_aio_write_nolock(struct k
>
>     written = generic_file_direct_write(iocb, iov, &nr_segs, pos,
>     ppos, count, ocount);
> + /*
> +  * If host is not S_ISBLK generic_file_direct_write() may
> +  * have instantiated a few blocks outside i_size files
> +  * Trim these off again.
> +  */
> + if (unlikely(written < 0) && !S_ISBLK(inode->i_mode)) {
> +     loff_t isize = i_size_read(inode);
> +     if (pos + count > isize)
> +         vmtruncate(inode, isize);
> + }
> +
> if (written < 0 || written == count)
>     goto out;
```

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.

```
> @@ -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 referring to the `vmtruncate()` call in `__generic_file_aio_write_nolock()`.

IOWs, wouldn't it be better to put this comment and check in `__generic_file_aio_write_nolock()` directly above the `vmtruncate()` call that cares about this?

```

> @@ -2383,8 +2402,8 @@ ssize_t generic_file_aio_write(struct ki
> EXPORT_SYMBOL(generic_file_aio_write);
>
> /*
> - * Called under i_mutex for writes to S_ISREG files. Returns -EIO if something
> - * went wrong during pagecache shutdown.
> + * May be called without i_mutex for writes to S_ISREG files. XFS does this.
> + * Returns -EIO if something went wrong during pagecache shutdown.
> */

```

Not sure you need to say "XFS does this" - other filesystems may do this in the future.....

Cheers,

Dave.

--

Dave Chinner
Principal Engineer
SGI Australian Software Group