Subject: Re: [PATCH] incorrect error handling inside generic_file_direct_write
Posted by Christoph Hellwig on Fri, 15 Dec 2006 10:43:41 GMT
View Forum Message <> Reply to Message

> +ssize_t
> +__generic_file_aio_write(struct kiocb *iocb, const struct iovec *iov,
> +   unsigned long nr_segs, loff_t pos)

I'd still call this generic_file_aio_write_nolock.

> + loff_t *ppos = &iocb->ki_pos;

I'd rather use iocb->ki_pos directly in the few places ppos is referenced
currently.

>   if (ret > 0 && ((file->f_flags & O_SYNC) || IS_SYNC(inode))) {
> - ssize_t err;
> -
>   err = sync_page_range_nolock(inode, mapping, pos, ret);
>   if (err < 0)
>   ret = err;
>   }

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.