```
**********************************************************
   44738 Wed Nov  6 12:52:06 2013
new/usr/src/uts/common/fs/zfs/dmu_tx.c
4188 assertion failed in dmu_tx_hold_free(): dn_datablkshift != 0
Reviewed by: George Wilson <george.wilson@delphix.com>
Reviewed by: Christopher Siden <christopher.siden@delphix.com>
**********************************************************
______unchanged_portion_omitted_
```

```
void
dmu_tx_hold_free(dmu_tx_t *tx, uint64_t object, uint64_t off, uint64_t len)
{
        dmu_tx_hold_t *txh;
        dnode_t *dn;
        int err;
        zio_t *zio;

        ASSERT(tx->tx_txg == 0);

        txh = dmu_tx_hold_object_impl(tx, tx->tx_objset,
            object, THT_FREE, off, len);
        if (txh == NULL)
                return;
        dn = txh->txh_dnode;
        dmu_tx_count_dnode(txh);

        if (off >= (dn->dn_maxblkid+1) * dn->dn_datablksz)
                return;
        if (len == DMU_OBJECT_END)
                len = (dn->dn_maxblkid+1) * dn->dn_datablksz - off;

        /*
         * For i/o error checking, we read the first and last level-0
         * blocks if they are not aligned, and all the level-1 blocks.
         *
         * Note:  dbuf_free_range() assumes that we have not instantiated
         * any level-0 dbufs that will be completely freed.  Therefore we must
         * exercise care to not read or count the first and last blocks
         * if they are blocksize-aligned.
         */
        if (dn->dn_datablkshift == 0) {
                if (off != 0 || len < dn->dn_datablksz)
                        dmu_tx_count_write(txh, 0, dn->dn_datablksz);
        } else {
                /* first block will be modified if it is not aligned */
                if (!IS_P2ALIGNED(off, 1 << dn->dn_datablkshift))
                        dmu_tx_count_write(txh, off, 1);
                /* last block will be modified if it is not aligned */
                if (!IS_P2ALIGNED(off + len, 1 << dn->dn_datablkshift))
                        dmu_tx_count_write(txh, off+len, 1);
        }

        /*
         * Check level-1 blocks.
         */
        if (dn->dn_nlevels > 1) {
                int shift = dn->dn_datablkshift + dn->dn_indblkshift -
                    SPA_BLKPTRSHIFT;
                uint64_t start = off >> shift;
                uint64_t end = (off + len) >> shift;

                ASSERT(dn->dn_datablkshift != 0);
                ASSERT(dn->dn_indblkshift != 0);

                /*
                 * dnode_reallocate() can result in an object with indirect
```

```
                 * blocks having an odd data block size.  In this case,
                 * just check the single block.
                 */
                if (dn->dn_datablkshift == 0)
                        start = end = 0;

                zio = zio_root(tx->tx_pool->dp_spa,
                    NULL, NULL, ZIO_FLAG_CANFAIL);
                for (uint64_t i = start; i <= end; i++) {
                        uint64_t ibyte = i << shift;
                        err = dnode_next_offset(dn, 0, &ibyte, 2, 1, 0);
                        i = ibyte >> shift;
                        if (err == ESRCH)
                                break;
                        if (err) {
                                tx->tx_err = err;
                                return;
                        }

                        err = dmu_tx_check_ioerr(zio, dn, 1, i);
                        if (err) {
                                tx->tx_err = err;
                                return;
                        }
                }
                err = zio_wait(zio);
                if (err) {
                        tx->tx_err = err;
                        return;
                }
        }

        dmu_tx_count_free(txh, off, len);
}
______unchanged_portion_omitted_
```