```
**********************************************************
   82421 Thu Aug 15 17:31:03 2013
new/usr/src/uts/common/fs/zfs/dsl_dataset.c
4046 dsl_dataset_t ds_dir->dd_lock is highly contended
Reviewed by: Eric Schrock <eric.schrock@delphix.com>
Reviewed by: George Wilson <george.wilson@delphix.com>
**********************************************************
______unchanged_portion_omitted_

  82 void
  83 dsl_dataset_block_born(dsl_dataset_t *ds, const blkptr_t *bp, dmu_tx_t *tx)
  84 {
  85         int used = bp_get_dsize_sync(tx->tx_pool->dp_spa, bp);
  86         int compressed = BP_GET_PSIZE(bp);
  87         int uncompressed = BP_GET_UCSIZE(bp);
  88         int64_t delta;

  90         dprintf_bp(bp, "ds=%p", ds);

  92         ASSERT(dmu_tx_is_syncing(tx));
  93         /* It could have been compressed away to nothing */
  94         if (BP_IS_HOLE(bp))
  95                 return;
  96         ASSERT(BP_GET_TYPE(bp) != DMU_OT_NONE);
  97         ASSERT(DMU_OT_IS_VALID(BP_GET_TYPE(bp)));
  98         if (ds == NULL) {
  99                 dsl_pool_mos_diduse_space(tx->tx_pool,
 100                     used, compressed, uncompressed);
 101                 return;
 102         }

 104         dmu_buf_will_dirty(ds->ds_dbuf, tx);

 105         mutex_enter(&ds->ds_dir->dd_lock);
 105         mutex_enter(&ds->ds_lock);
 106         delta = parent_delta(ds, used);
 107         ds->ds_phys->ds_referenced_bytes += used;
 108         ds->ds_phys->ds_compressed_bytes += compressed;
 109         ds->ds_phys->ds_uncompressed_bytes += uncompressed;
 110         ds->ds_phys->ds_unique_bytes += used;
 111         mutex_exit(&ds->ds_lock);
 112         dsl_dir_diduse_space(ds->ds_dir, DD_USED_HEAD, delta,
 113             compressed, uncompressed, tx);
 114         dsl_dir_transfer_space(ds->ds_dir, used - delta,
 115             DD_USED_REFRSRV, DD_USED_HEAD, tx);
 117         mutex_exit(&ds->ds_dir->dd_lock);
 116 }

 118 int
 119 dsl_dataset_block_kill(dsl_dataset_t *ds, const blkptr_t *bp, dmu_tx_t *tx,
 120     boolean_t async)
 121 {
 122         if (BP_IS_HOLE(bp))
 123                 return (0);

 125         ASSERT(dmu_tx_is_syncing(tx));
 126         ASSERT(bp->blk_birth <= tx->tx_txg);

 128         int used = bp_get_dsize_sync(tx->tx_pool->dp_spa, bp);
 129         int compressed = BP_GET_PSIZE(bp);
 130         int uncompressed = BP_GET_UCSIZE(bp);

 132         ASSERT(used > 0);
 133         if (ds == NULL) {
 134                 dsl_free(tx->tx_pool, tx->tx_txg, bp);
 135                 dsl_pool_mos_diduse_space(tx->tx_pool,
```



```
 136                     -used, -compressed, -uncompressed);
 137                 return (used);
 138         }
 139         ASSERT3P(tx->tx_pool, ==, ds->ds_dir->dd_pool);

 141         ASSERT(!dsl_dataset_is_snapshot(ds));
 142         dmu_buf_will_dirty(ds->ds_dbuf, tx);

 144         if (bp->blk_birth > ds->ds_phys->ds_prev_snap_txg) {
 145                 int64_t delta;

 147                 dprintf_bp(bp, "freeing ds=%llu", ds->ds_object);
 148                 dsl_free(tx->tx_pool, tx->tx_txg, bp);

 152                 mutex_enter(&ds->ds_dir->dd_lock);
 150                 mutex_enter(&ds->ds_lock);
 151                 ASSERT(ds->ds_phys->ds_unique_bytes >= used ||
 152                     !DS_UNIQUE_IS_ACCURATE(ds));
 153                 delta = parent_delta(ds, -used);
 154                 ds->ds_phys->ds_unique_bytes -= used;
 155                 mutex_exit(&ds->ds_lock);
 156                 dsl_dir_diduse_space(ds->ds_dir, DD_USED_HEAD,
 157                     delta, -compressed, -uncompressed, tx);
 158                 dsl_dir_transfer_space(ds->ds_dir, -used - delta,
 159                     DD_USED_REFRSRV, DD_USED_HEAD, tx);
 163                 mutex_exit(&ds->ds_dir->dd_lock);
 160         } else {
 161                 dprintf_bp(bp, "putting on dead list: %s", "");
 162                 if (async) {
 163                         /*
 164                          * We are here as part of zio's write done callback,
 165                          * which means we're a zio interrupt thread.  We can't
 166                          * call dsl_deadlist_insert() now because it may block
 167                          * waiting for I/O.  Instead, put bp on the deferred
 168                          * queue and let dsl_pool_sync() finish the job.
 169                          */
 170                         bplist_append(&ds->ds_pending_deadlist, bp);
 171                 } else {
 172                         dsl_deadlist_insert(&ds->ds_deadlist, bp, tx);
 173                 }
 174                 ASSERT3U(ds->ds_prev->ds_object, ==,
 175                     ds->ds_phys->ds_prev_snap_obj);
 176                 ASSERT(ds->ds_prev->ds_phys->ds_num_children > 0);
 177                 /* if (bp->blk_birth > prev prev snap txg) prev unique += bs */
 178                 if (ds->ds_prev->ds_phys->ds_next_snap_obj ==
 179                     ds->ds_object && bp->blk_birth >
 180                     ds->ds_prev->ds_phys->ds_prev_snap_txg) {
 181                         dmu_buf_will_dirty(ds->ds_prev->ds_dbuf, tx);
 182                         mutex_enter(&ds->ds_prev->ds_lock);
 183                         ds->ds_prev->ds_phys->ds_unique_bytes += used;
 184                         mutex_exit(&ds->ds_prev->ds_lock);
 185                 }
 186                 if (bp->blk_birth > ds->ds_dir->dd_origin_txg) {
 187                         dsl_dir_transfer_space(ds->ds_dir, used,
 188                             DD_USED_HEAD, DD_USED_SNAP, tx);
 189                 }
 190         }
 191         mutex_enter(&ds->ds_lock);
 192         ASSERT3U(ds->ds_phys->ds_referenced_bytes, >=, used);
 193         ds->ds_phys->ds_referenced_bytes -= used;
 194         ASSERT3U(ds->ds_phys->ds_compressed_bytes, >=, compressed);
 195         ds->ds_phys->ds_compressed_bytes -= compressed;
 196         ASSERT3U(ds->ds_phys->ds_uncompressed_bytes, >=, uncompressed);
 197         ds->ds_phys->ds_uncompressed_bytes -= uncompressed;
 198         mutex_exit(&ds->ds_lock);
```

```
 200         return (used);
 201 }
```

_____***unchanged_portion_omitted_***

```
 594 static int
 595 dsl_dataset_namelen(dsl_dataset_t *ds)
 596 {
 597         int result;

 599         if (ds == NULL) {
 600                 result = 3;     /* "mos" */
 601         } else {
 602                 result = dsl_dir_namelen(ds->ds_dir);
 603                 VERIFY0(dsl_dataset_get_snapname(ds));
 604                 if (ds->ds_snapname[0]) {
 605                         ++result;       /* adding one for the @-sign */
 606                         if (!MUTEX_HELD(&ds->ds_lock)) {
 607                                 mutex_enter(&ds->ds_lock);
 608                                 result += strlen(ds->ds_snapname);
 609                                 mutex_exit(&ds->ds_lock);
 610                         } else {
 611                                 result += strlen(ds->ds_snapname);
 612                         }
 613                 }
 614         }

 616         return (result);
 617 }
```

```
 590 void
 591 dsl_dataset_rele(dsl_dataset_t *ds, void *tag)
 592 {
 593         dmu_buf_rele(ds->ds_dbuf, tag);
 594 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   35916 Thu Aug 15 17:31:07 2013
new/usr/src/uts/common/fs/zfs/dsl_dir.c
4046 dsl_dataset_t ds_dir->dd_lock is highly contended
Reviewed by: Eric Schrock <eric.schrock@delphix.com>
Reviewed by: George Wilson <george.wilson@delphix.com>
**********************************************************
______unchanged_portion_omitted_

 838 /* call from syncing context when we actually write/free space for this dd */
 839 void
 840 dsl_dir_diduse_space(dsl_dir_t *dd, dd_used_t type,
 841     int64_t used, int64_t compressed, int64_t uncompressed, dmu_tx_t *tx)
 842 {
 843         int64_t accounted_delta;

 845         /*
 846          * dsl_dataset_set_refreservation_sync_impl() calls this with
 847          * dd_lock held, so that it can atomically update
 848          * ds->ds_reserved and the dsl_dir accounting, so that
 849          * dsl_dataset_check_quota() can see dataset and dir accounting
 850          * consistently.
 851          */
 852         boolean_t needlock = !MUTEX_HELD(&dd->dd_lock);

 854         ASSERT(dmu_tx_is_syncing(tx));
 855         ASSERT(type < DD_USED_NUM);

 857         dmu_buf_will_dirty(dd->dd_dbuf, tx);

 859         if (needlock)
 860                 mutex_enter(&dd->dd_lock);
 861         accounted_delta = parent_delta(dd, dd->dd_phys->dd_used_bytes, used);
 862         ASSERT(used >= 0 || dd->dd_phys->dd_used_bytes >= -used);
 863         ASSERT(compressed >= 0 ||
 864             dd->dd_phys->dd_compressed_bytes >= -compressed);
 865         ASSERT(uncompressed >= 0 ||
 866             dd->dd_phys->dd_uncompressed_bytes >= -uncompressed);
 857         dmu_buf_will_dirty(dd->dd_dbuf, tx);
 867         dd->dd_phys->dd_used_bytes += used;
 868         dd->dd_phys->dd_uncompressed_bytes += uncompressed;
 869         dd->dd_phys->dd_compressed_bytes += compressed;

 871         if (dd->dd_phys->dd_flags & DD_FLAG_USED_BREAKDOWN) {
 872                 ASSERT(used > 0 ||
 873                     dd->dd_phys->dd_used_breakdown[type] >= -used);
 874                 dd->dd_phys->dd_used_breakdown[type] += used;
 875 #ifdef DEBUG
 876                 dd_used_t t;
 877                 uint64_t u = 0;
 878                 for (t = 0; t < DD_USED_NUM; t++)
 879                         u += dd->dd_phys->dd_used_breakdown[t];
 880                 ASSERT3U(u, ==, dd->dd_phys->dd_used_bytes);
 881 #endif
 882         }
 883         if (needlock)
 884                 mutex_exit(&dd->dd_lock);

 886         if (dd->dd_parent != NULL) {
 887                 dsl_dir_diduse_space(dd->dd_parent, DD_USED_CHILD,
 888                     accounted_delta, compressed, uncompressed, tx);
 889                 dsl_dir_transfer_space(dd->dd_parent,
 890                     used - accounted_delta,
 891                     DD_USED_CHILD_RSRV, DD_USED_CHILD, tx);
 892         }
 893 }
```



```
 895 void
 896 dsl_dir_transfer_space(dsl_dir_t *dd, int64_t delta,
 897     dd_used_t oldtype, dd_used_t newtype, dmu_tx_t *tx)
 898 {
 890         boolean_t needlock = !MUTEX_HELD(&dd->dd_lock);

 899         ASSERT(dmu_tx_is_syncing(tx));
 900         ASSERT(oldtype < DD_USED_NUM);
 901         ASSERT(newtype < DD_USED_NUM);

 903         if (delta == 0 || !(dd->dd_phys->dd_flags & DD_FLAG_USED_BREAKDOWN))
 904                 return;

 906         dmu_buf_will_dirty(dd->dd_dbuf, tx);
 899         if (needlock)
 907         mutex_enter(&dd->dd_lock);
 908         ASSERT(delta > 0 ?
 909             dd->dd_phys->dd_used_breakdown[oldtype] >= delta :
 910             dd->dd_phys->dd_used_breakdown[newtype] >= -delta);
 911         ASSERT(dd->dd_phys->dd_used_bytes >= ABS(delta));
 905         dmu_buf_will_dirty(dd->dd_dbuf, tx);
 912         dd->dd_phys->dd_used_breakdown[oldtype] -= delta;
 913         dd->dd_phys->dd_used_breakdown[newtype] += delta;
 908         if (needlock)
 914         mutex_exit(&dd->dd_lock);
 915 }
______unchanged_portion_omitted_
```