```
**********************************************************
   64722 Mon May 28 11:59:40 2012
new/usr/src/uts/common/fs/nfs/nfs4_callback.c
We've seen a situation where the NFS4-client tries endlessly to return a
delegation. This happened after a server reboot. The server responds with
STALE_STATEID to the DELEGRETURN. This case is not directly handled by
nfs4_do_delegreturn. Instead, it triggers a recovery of the clientid, which
in turn triggers a reclaim of all open files for this server.
To find the open files, rtable4 is enumerated for each mount to the server.
This is supposed to reopen the file, so that the next DELEGRETURN will
succeed.
In this case, the rnode is not in the rtable-hash, so it never gets
recovered. This leads to an endless delegreturn-loop, iterated once per
second.
This fix tests on NFS4_STALE_STATEID in combination with the rnode not
being in the hash. In this case, it treats the error as fatal and just
discards the delegation.
**********************************************************
______unchanged_portion_omitted_

1445 int
1446 nfs4_do_delegreturn(rnode4_t *rp, int flags, cred_t *cr,
1447         struct nfs4_callback_globals *ncg)
1448 {
1449         vnode_t *vp = RTOV4(rp);
1450         mntinfo4_t *mi = VTOMI4(vp);
1451         nfs4_lost_rqst_t lost_rqst;
1452         nfs4_recov_state_t recov_state;
1453         bool_t needrecov = FALSE, recovonly, done = FALSE;
1454         nfs4_error_t e = { 0, NFS4_OK, RPC_SUCCESS };

1456         ncg->nfs4_callback_stats.delegreturn.value.ui64++;

1458         while (!done) {
1459                 e.error = nfs4_start_fop(mi, vp, NULL, OH_DELEGRETURN,
1460                     &recov_state, &recovonly);

1462                 if (e.error) {
1463                         if (flags & NFS4_DR_FORCE) {
1464                                 (void) nfs_rw_enter_sig(&mi->mi_recovlock,
1465                                     RW_READER, 0);
1466                                 nfs4delegreturn_cleanup_impl(rp, NULL, ncg);
1467                                 nfs_rw_exit(&mi->mi_recovlock);
1468                         }
1469                         break;
1470                 }

1472                 /*
1473                  * Check to see if the delegation has already been
1474                  * returned by the recovery thread.   The state of
1475                  * the delegation cannot change at this point due
1476                  * to start_fop and the r_deleg_recall_lock.
1477                  */
1478                 if (rp->r_deleg_type == OPEN_DELEGATE_NONE) {
1479                         e.error = 0;
1480                         nfs4_end_op(mi, vp, NULL, &recov_state, needrecov);
1481                         break;
1482                 }

1484                 if (recovonly) {
1485                         /*
1486                          * Delegation will be returned via the
1487                          * recovery framework.  Build a lost request
1488                          * structure, start recovery and get out.
1489                          */
1490                         nfs4_error_init(&e, EINTR);
```



```
1491                         nfs4delegreturn_save_lost_rqst(e.error, &lost_rqst,
1492                             cr, vp);
1493                         (void) nfs4_start_recovery(&e, mi, vp,
1494                             NULL, &rp->r_deleg_stateid,
1495                             lost_rqst.lr_op == OP_DELEGRETURN ?
1496                             &lost_rqst : NULL, OP_DELEGRETURN, NULL,
1497                             NULL, NULL);
1498                         nfs4_end_op(mi, vp, NULL, &recov_state, needrecov);
1499                         break;
1500                 }

1502                 nfs4delegreturn_otw(rp, cr, &e);

1504                 /*
1505                  * Ignore some errors on delegreturn; no point in marking
1506                  * the file dead on a state destroying operation.
1507                  */
1508                 if (e.error == 0 && (nfs4_recov_marks_dead(e.stat) ||
1509                     e.stat == NFS4ERR_BADHANDLE ||
1510                     e.stat == NFS4ERR_STALE ||
1511                     (e.stat == NFS4ERR_STALE_STATEID &&
1512                      !(rp->r_flags & R4HASHED))))
1510                     e.stat == NFS4ERR_STALE))
1513                         needrecov = FALSE;
1514                 else
1515                         needrecov = nfs4_needs_recovery(&e, TRUE, vp->v_vfsp);

1517                 if (needrecov) {
1518                         nfs4delegreturn_save_lost_rqst(e.error, &lost_rqst,
1519                             cr, vp);
1520                         (void) nfs4_start_recovery(&e, mi, vp,
1521                             NULL, &rp->r_deleg_stateid,
1522                             lost_rqst.lr_op == OP_DELEGRETURN ?
1523                             &lost_rqst : NULL, OP_DELEGRETURN, NULL,
1524                             NULL, NULL);
1525                 } else {
1526                         nfs4delegreturn_cleanup_impl(rp, NULL, ncg);
1527                         done = TRUE;
1528                 }

1530                 nfs4_end_op(mi, vp, NULL, &recov_state, needrecov);
1531         }
1532         return (e.error);
1533 }
______unchanged_portion_omitted_
```