```
**********************************************************
    4460 Tue Apr 23 15:32:37 2013
new/usr/src/uts/common/fs/zfs/zrlock.c
3746 ZRLs are racy
Submitted by:   Justin Gibbs <justing@spectralogic.com>
**********************************************************
_____unchanged_portion_omitted_

```

```
 70 void
 71 #ifdef  ZFS_DEBUG
 72 zrl_add_debug(zrlock_t *zrl, const char *zc)
 73 #else
 74 zrl_add(zrlock_t *zrl)
 75 #endif
 76 {
 77         uint32_t n = (uint32_t)zrl->zr_refcount;

 79         while (1) {
 80 #endif /* ! codereview */
 81                 while (n != ZRL_LOCKED) {
 82                         uint32_t cas = atomic_cas_32(
 83                             (uint32_t *)&zrl->zr_refcount, n, n + 1);
 84                         if (cas == n) {
 85                                 ASSERT((int32_t)n >= 0);
 86 #ifdef  ZFS_DEBUG
 87                                 if (zrl->zr_owner == curthread) {
 88                                         DTRACE_PROBE2(zrlock__reentry,
 89                                             zrlock_t *, zrl, uint32_t, n);
 90                                 }
 91                                 zrl->zr_owner = curthread;
 92                                 zrl->zr_caller = zc;
 93 #endif
 94                                 return;
 95                         }
 96                         n = cas;
 97                 }

 98                 mutex_enter(&zrl->zr_mtx);
 99                 while (zrl->zr_refcount == ZRL_LOCKED) {
100                         cv_wait(&zrl->zr_cv, &zrl->zr_mtx);
101                 }
 84         ASSERT(zrl->zr_refcount >= 0);
 85         zrl->zr_refcount++;
 86 #ifdef  ZFS_DEBUG
 87         zrl->zr_owner = curthread;
 88         zrl->zr_caller = zc;
 89 #endif
102                 mutex_exit(&zrl->zr_mtx);
103         }
104 #endif /* ! codereview */
105 }

107 void
108 zrl_remove(zrlock_t *zrl)
109 {
110         uint32_t n;

112         n = atomic_dec_32_nv((uint32_t *)&zrl->zr_refcount);
113         ASSERT((int32_t)n >= 0);
114 #ifdef  ZFS_DEBUG
115         if (zrl->zr_owner == curthread) {
116                 zrl->zr_owner = NULL;
117                 zrl->zr_caller = NULL;
118         }
119 #endif
120 }
```



```
122 int
123 zrl_tryenter(zrlock_t *zrl)
124 {
125         uint32_t n = (uint32_t)zrl->zr_refcount;

127         if (n == 0) {
128                 uint32_t cas = atomic_cas_32(
129                     (uint32_t *)&zrl->zr_refcount, 0, ZRL_LOCKED);
130                 if (cas == 0) {
131 #ifdef  ZFS_DEBUG
132                         ASSERT(zrl->zr_owner == NULL);
133                         zrl->zr_owner = curthread;
134 #endif
135                         return (1);
136                 }
137         }

139         ASSERT((int32_t)n > ZRL_DESTROYED);

141         return (0);
142 }

144 void
145 zrl_exit(zrlock_t *zrl)
146 {
147         ASSERT(zrl->zr_refcount == ZRL_LOCKED);

149         mutex_enter(&zrl->zr_mtx);
150 #ifdef  ZFS_DEBUG
151         ASSERT(zrl->zr_owner == curthread);
152         zrl->zr_owner = NULL;
153         membar_producer();      /* make sure the owner store happens first */
154 #endif
155         zrl->zr_refcount = 0;
156         cv_broadcast(&zrl->zr_cv);
157         mutex_exit(&zrl->zr_mtx);
158 }

160 int
161 zrl_refcount(zrlock_t *zrl)
162 {
163         ASSERT(zrl->zr_refcount > ZRL_DESTROYED);

165         int n = (int)zrl->zr_refcount;
166         return (n <= 0 ? 0 : n);
167 }

169 int
170 zrl_is_zero(zrlock_t *zrl)
171 {
172         ASSERT(zrl->zr_refcount > ZRL_DESTROYED);

174         return (zrl->zr_refcount <= 0);
175 }

177 int
178 zrl_is_locked(zrlock_t *zrl)
179 {
180         ASSERT(zrl->zr_refcount > ZRL_DESTROYED);

182         return (zrl->zr_refcount == ZRL_LOCKED);
183 }

185 #ifdef  ZFS_DEBUG
186 kthread_t *
```

```
187 zrl_owner(zrlock_t *zrl)
188 {
189         return (zrl->zr_owner);
190 }
191 #endif
```