LCOV - code coverage report

Current view:	directory - fs/ext4 - block_validity.c (source / functions)		Found
Test:	kernel_2_6_31_ext4_round_3.info	**Lines:**	118
Date:	2009-10-24	**Functions:**	6

```
                :  1 /*
                :  2  *  linux/fs/ext4/block_validity.c
                :  3  *
                :  4  * Copyright (C) 2009
                :  5  * Theodore Ts'o (tytso@mit.edu)
                :  6  *
                :  7  * Track which blocks in the filesystem are metadata blocks that
                :  8  * should never be used as data blocks by files or directories.
                :  9  */
                : 10
                : 11 #include <linux/time.h>
                : 12 #include <linux/fs.h>
                : 13 #include <linux/namei.h>
                : 14 #include <linux/quotaops.h>
                : 15 #include <linux/buffer_head.h>
                : 16 #include <linux/module.h>
                : 17 #include <linux/swap.h>
                : 18 #include <linux/pagemap.h>
                : 19 #include <linux/version.h>
                : 20 #include <linux/blkdev.h>
                : 21 #include <linux/mutex.h>
                : 22 #include "ext4.h"
                : 23
                : 24 struct ext4_system_zone {
                : 25         struct rb_node  node;
                : 26         ext4_fsblk_t    start_blk;
                : 27         unsigned int    count;
                : 28 };
                : 29
                : 30 static struct kmem_cache *ext4_system_zone_cachep;
                : 31
                : 32 int __init init_ext4_system_zone(void)
              0 : 33 {
              0 : 34         ext4_system_zone_cachep = KMEM_CACHE(ext4_system_zone,
                : 35                                              SLAB_RECLAIM_ACCOUNT);
              0 : 36         if (ext4_system_zone_cachep == NULL)
              0 : 37                 return -ENOMEM;
              0 : 38         return 0;
                : 39 }
                : 40
                : 41 void exit_ext4_system_zone(void)
              0 : 42 {
              0 : 43         kmem_cache_destroy(ext4_system_zone_cachep);
              0 : 44 }
                : 45
                : 46 static inline int can_merge(struct ext4_system_zone *entry1,
                : 47                           struct ext4_system_zone *entry2)
                : 48 {
              0 : 49         if ((entry1->start_blk + entry1->count) == entry2->start_blk)
              0 : 50                 return 1;
              0 : 51         return 0;
                : 52 }
```

```
  53                :
  54                : /*
  55                :  * Mark a range of blocks as belonging to the "system zone" --- that
  56                :  * is, filesystem metadata blocks which should never be used by
  57                :  * inodes.
  58                :  */
  59                : static int add_system_zone(struct ext4_sb_info *sbi,
  60                :                            ext4_fsblk_t start_blk,
  61                :                            unsigned int count)
  62              0 : {
  63              0 :         struct ext4_system_zone *new_entry = NULL, *entry;
  64              0 :         struct rb_node **n = &sbi->system_blks.rb_node, *node;
  65              0 :         struct rb_node *parent = NULL, *new_node = NULL;
  66                :
  67              0 :         while (*n) {
  68              0 :                 parent = *n;
  69              0 :                 entry = rb_entry(parent, struct ext4_system_zone, node);
  70              0 :                 if (start_blk < entry->start_blk)
  71              0 :                         n = &(*n)->rb_left;
  72              0 :                 else if (start_blk >= (entry->start_blk + entry->count))
  73              0 :                         n = &(*n)->rb_right;
  74                :                 else {
  75              0 :                         if (start_blk + count > (entry->start_blk +
  76                :                                                  entry->count))
  77              0 :                                 entry->count = (start_blk + count -
  78                :                                                 entry->start_blk);
  79              0 :                         new_node = *n;
  80              0 :                         new_entry = rb_entry(new_node, struct ext4_system_zone,
  81                :                                              node);
  82              0 :                         break;
  83                :                 }
  84                :         }
  85                :
  86              0 :         if (!new_entry) {
  87              0 :                 new_entry = kmem_cache_alloc(ext4_system_zone_cachep,
  88                :                                              GFP_KERNEL);
  89              0 :                 if (!new_entry)
  90              0 :                         return -ENOMEM;
  91              0 :                 new_entry->start_blk = start_blk;
  92              0 :                 new_entry->count = count;
  93              0 :                 new_node = &new_entry->node;
  94                :
  95                :                 rb_link_node(new_node, parent, n);
  96              0 :                 rb_insert_color(new_node, &sbi->system_blks);
  97                :         }
  98                :
  99                :         /* Can we merge to the left? */
 100              0 :         node = rb_prev(new_node);
 101              0 :         if (node) {
 102              0 :                 entry = rb_entry(node, struct ext4_system_zone, node);
 103              0 :                 if (can_merge(entry, new_entry)) {
 104              0 :                         new_entry->start_blk = entry->start_blk;
 105              0 :                         new_entry->count += entry->count;
 106              0 :                         rb_erase(node, &sbi->system_blks);
 107              0 :                         kmem_cache_free(ext4_system_zone_cachep, entry);
 108                :                 }
 109                :         }
 110                :
 111                :         /* Can we merge to the right? */
 112              0 :         node = rb_next(new_node);
 113              0 :         if (node) {
 114              0 :                 entry = rb_entry(node, struct ext4_system_zone, node);
 115              0 :                 if (can_merge(new_entry, entry)) {
 116              0 :                         new_entry->count += entry->count;
 117              0 :                         rb_erase(node, &sbi->system_blks);
```

```
118                0 :                         kmem_cache_free(ext4_system_zone_cachep, entry);
119                  :                 }
120                  :         }
121                0 :         return 0;
122                  : }
123                  :
124                  : static void debug_print_tree(struct ext4_sb_info *sbi)
125                  : {
126                  :         struct rb_node *node;
127                  :         struct ext4_system_zone *entry;
128                0 :         int first = 1;
129                  :
130                0 :         printk(KERN_INFO "System zones: ");
131                0 :         node = rb_first(&sbi->system_blks);
132                0 :         while (node) {
133                0 :                 entry = rb_entry(node, struct ext4_system_zone, node);
134                0 :                 printk("%s%llu-%llu", first ? "" : ", ",
135                  :                        entry->start_blk, entry->start_blk + entry->count - 1);
136                0 :                 first = 0;
137                0 :                 node = rb_next(node);
138                  :         }
139                0 :         printk("\n");
140                  : }
141                  :
142                  : int ext4_setup_system_zone(struct super_block *sb)
143               94 : {
144               94 :         ext4_group_t ngroups = ext4_get_groups_count(sb);
145               94 :         struct ext4_sb_info *sbi = EXT4_SB(sb);
146                  :         struct ext4_group_desc *gdp;
147                  :         ext4_group_t i;
148               94 :         int flex_size = ext4_flex_bg_size(sbi);
149                  :         int ret;
150                  :
151               94 :         if (!test_opt(sb, BLOCK_VALIDITY)) {
152               94 :                 if (EXT4_SB(sb)->system_blks.rb_node)
153                0 :                         ext4_release_system_zone(sb);
154               94 :                 return 0;
155                  :         }
156                0 :         if (EXT4_SB(sb)->system_blks.rb_node)
157                0 :                 return 0;
158                  :
159                0 :         for (i=0; i < ngroups; i++) {
160                0 :                 if (ext4_bg_has_super(sb, i) &&
161                  :                     ((i < 5) || ((i % flex_size) == 0)))
162                0 :                         add_system_zone(sbi, ext4_group_first_block_no(sb, i),
163                  :                                         sbi->s_gdb_count + 1);
164                0 :                 gdp = ext4_get_group_desc(sb, i, NULL);
165                0 :                 ret = add_system_zone(sbi, ext4_block_bitmap(sb, gdp), 1);
166                0 :                 if (ret)
167                0 :                         return ret;
168                0 :                 ret = add_system_zone(sbi, ext4_inode_bitmap(sb, gdp), 1);
169                0 :                 if (ret)
170                0 :                         return ret;
171                0 :                 ret = add_system_zone(sbi, ext4_inode_table(sb, gdp),
172                  :                                       sbi->s_itb_per_group);
173                0 :                 if (ret)
174                0 :                         return ret;
175                  :         }
176                  :
177                0 :         if (test_opt(sb, DEBUG))
178                  :                 debug_print_tree(EXT4_SB(sb));
179                0 :         return 0;
180                  : }
181                  :
182                  : /* Called when the filesystem is unmounted */
```

```
   183                :  void ext4_release_system_zone(struct super_block *sb)
   184               94 :  {
   185               94 :          struct rb_node  *n = EXT4_SB(sb)->system_blks.rb_node;
   186                  :          struct rb_node  *parent;
   187                  :          struct ext4_system_zone *entry;
   188                  :
   189              188 :          while (n) {
   190                  :                  /* Do the node's children first */
   191                0 :                  if (n->rb_left) {
   192                0 :                          n = n->rb_left;
   193                0 :                          continue;
   194                  :                  }
   195                0 :                  if (n->rb_right) {
   196                0 :                          n = n->rb_right;
   197                0 :                          continue;
   198                  :                  }
   199                  :                  /*
   200                  :                   * The node has no children; free it, and then zero
   201                  :                   * out parent's link to it.  Finally go to the
   202                  :                   * beginning of the loop and try to free the parent
   203                  :                   * node.
   204                  :                   */
   205                0 :                  parent = rb_parent(n);
   206                0 :                  entry = rb_entry(n, struct ext4_system_zone, node);
   207                0 :                  kmem_cache_free(ext4_system_zone_cachep, entry);
   208                0 :                  if (!parent)
   209                0 :                          EXT4_SB(sb)->system_blks.rb_node = NULL;
   210                0 :                  else if (parent->rb_left == n)
   211                0 :                          parent->rb_left = NULL;
   212                0 :                  else if (parent->rb_right == n)
   213                0 :                          parent->rb_right = NULL;
   214                0 :                  n = parent;
   215                  :          }
   216               94 :          EXT4_SB(sb)->system_blks.rb_node = NULL;
   217               94 :  }
   218                  :
   219                  :  /*
   220                  :   * Returns 1 if the passed-in block region (start_blk,
   221                  :   * start_blk+count) is valid; 0 if some part of the block region
   222                  :   * overlaps with filesystem metadata blocks.
   223                  :   */
   224                  :  int ext4_data_block_valid(struct ext4_sb_info *sbi, ext4_fsblk_t start_blk,
   225                  :                             unsigned int count)
   226        555886331 :  {
   227                  :          struct ext4_system_zone *entry;
   228        555886331 :          struct rb_node *n = sbi->system_blks.rb_node;
   229                  :
   230       1110786023 :          if ((start_blk <= le32_to_cpu(sbi->s_es->s_first_data_block)) ||
   231                  :              (start_blk + count > ext4_blocks_count(sbi->s_es)))
   232                0 :                  return 0;
   233        555886331 :          while (n) {
   234                0 :                  entry = rb_entry(n, struct ext4_system_zone, node);
   235                0 :                  if (start_blk + count - 1 < entry->start_blk)
   236                0 :                          n = n->rb_left;
   237                0 :                  else if (start_blk >= (entry->start_blk + entry->count))
   238                0 :                          n = n->rb_right;
   239                  :                  else
   240                0 :                          return 0;
   241                  :          }
   242        555886331 :          return 1;
   243                  :  }
   244                  :
```