
Subject: [PATCH 6/12] gfs2: make gfs2_glock.gl_owner_pid be a struct pid *
Posted by [Pavel Emelianov](#) on Tue, 29 Jan 2008 13:55:47 GMT

[View Forum Message](#) <> [Reply to Message](#)

The gl_owner_pid field is used to get the lock owning task by its pid, so make it in a proper manner, i.e. by using the struct pid pointer and pid_task() function.

The pid_task() becomes exported for the gfs2 module.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
fs/gfs2/glock.c | 19 ++++++++-----
fs/gfs2/incore.h | 2 +-
2 files changed, 13 insertions(+), 8 deletions(-)
```

```
diff --git a/fs/gfs2/glock.c b/fs/gfs2/glock.c
```

```
index 80e09c5..5fe585f 100644
```

```
--- a/fs/gfs2/glock.c
```

```
+++ b/fs/gfs2/glock.c
```

```
@@ -334,7 +334,7 @@ int gfs2_glock_get(struct gfs2_sbd *sdp, u64 number,
```

```
    gl->gl_state = LM_ST_UNLOCKED;
```

```
    gl->gl_demote_state = LM_ST_EXCLUSIVE;
```

```
    gl->gl_hash = hash;
```

```
- gl->gl_owner_pid = 0;
```

```
+ gl->gl_owner_pid = NULL;
```

```
    gl->gl_ip = 0;
```

```
    gl->gl_ops = glops;
```

```
    gl->gl_req_gh = NULL;
```

```
@@ -631,7 +631,7 @@ static void gfs2_glmutex_lock(struct gfs2_glock *gl)
```

```
    wait_on_holder(&gh);
```

```
    gfs2_holder_uninit(&gh);
```

```
    } else {
```

```
- gl->gl_owner_pid = current->pid;
```

```
+ gl->gl_owner_pid = get_pid(task_pid(current));
```

```
    gl->gl_ip = (unsigned long)__builtin_return_address(0);
```

```
    spin_unlock(&gl->gl_spin);
```

```
    }
```

```
@@ -652,7 +652,7 @@ static int gfs2_glmutex_trylock(struct gfs2_glock *gl)
```

```
    if (test_and_set_bit(GLF_LOCK, &gl->gl_flags)) {
```

```
        acquired = 0;
```

```
    } else {
```

```
- gl->gl_owner_pid = current->pid;
```

```
+ gl->gl_owner_pid = get_pid(task_pid(current));
```

```
    gl->gl_ip = (unsigned long)__builtin_return_address(0);
```

```
    }
```

```
    spin_unlock(&gl->gl_spin);
```

@@ -668,12 +668,17 @@ static int gfs2_glmutex_trylock(struct gfs2_glock *gl)

```
static void gfs2_glmutex_unlock(struct gfs2_glock *gl)
{
+ struct pid *pid;
+
  spin_lock(&gl->gl_spin);
  clear_bit(GLF_LOCK, &gl->gl_flags);
- gl->gl_owner_pid = 0;
+ pid = gl->gl_owner_pid;
+ gl->gl_owner_pid = NULL;
  gl->gl_ip = 0;
  run_queue(gl);
  spin_unlock(&gl->gl_spin);
+
+ put_pid(pid);
}
```

/**

@@ -1877,13 +1882,13 @@ static int dump_glock(struct glock_iter *gi, struct gfs2_glock *gl)

```
  print_dbg(gi, " gl_ref = %d\n", atomic_read(&gl->gl_ref));
  print_dbg(gi, " gl_state = %u\n", gl->gl_state);
  if (gl->gl_owner_pid) {
- gl_owner = find_task_by_pid(gl->gl_owner_pid);
+ gl_owner = pid_task(gl->gl_owner_pid, PIDTYPE_PID);
  if (gl_owner)
    print_dbg(gi, " gl_owner = pid %d (%s)\n",
-    gl->gl_owner_pid, gl_owner->comm);
+    pid_nr(gl->gl_owner_pid), gl_owner->comm);
  else
    print_dbg(gi, " gl_owner = %d (ended)\n",
-    gl->gl_owner_pid);
+    pid_nr(gl->gl_owner_pid));
  } else
    print_dbg(gi, " gl_owner = -1\n");
  print_dbg(gi, " gl_ip = %lu\n", gl->gl_ip);
```

diff --git a/fs/gfs2/incore.h b/fs/gfs2/incore.h

index 1339996..8ad1c3f 100644

--- a/fs/gfs2/incore.h

+++ b/fs/gfs2/incore.h

@@ -182,7 +182,7 @@ struct gfs2_glock {

```
  unsigned int gl_hash;
  unsigned int gl_demote_state; /* state requested by remote node */
  unsigned long gl_demote_time; /* time of first demote request */
- pid_t gl_owner_pid;
+ struct pid *gl_owner_pid;
  unsigned long gl_ip;
  struct list_head gl_holders;
```

```
    struct list_head gl_waiters1; /* HIF_MUTEX */
diff --git a/kernel/pid.c b/kernel/pid.c
index e4e5fc8..4776915 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -367,6 +367,7 @@ struct task_struct *pid_task(struct pid *pid, enum pid_type type)
 }
    return result;
 }
+EXPORT_SYMBOL(pid_task);
```

```
/*
 * Must be called under rcu_read_lock() or with tasklist_lock read-held.
--
```

1.5.3.4
