

---

Subject: [PATCH] [NFS]: Lock daemon start/stop rework.

Posted by [den](#) on Wed, 30 Jan 2008 11:41:34 GMT

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

---

The pid of the locking daemon can be substituted with a task struct without a problem. Namely, the value is filled in the context of the lockd thread and used in lockd\_up/lockd\_down.

It is possible to save task struct instead and use it to kill the process. The safety of this operation is guaranteed by the RCU, i.e. task can't disappear without passing a quiescent state.

Signed-off-by: Denis V. Lunev <[den@openvz.org](mailto:den@openvz.org)>

---

fs/lockd/svc.c | 38 ++++++-----  
1 files changed, 25 insertions(+), 13 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c

index 82e2192..4979e70 100644

--- a/fs/lockd/svc.c

+++ b/fs/lockd/svc.c

@@ -48,7 +48,7 @@ EXPORT\_SYMBOL(nlmsvc\_ops);

static DEFINE\_MUTEX(nlmsvc\_mutex);

static unsigned int nlmsvc\_users;

-static pid\_t nlmsvc\_pid;

+static struct task\_struct \*nlmsvc\_task;

static struct svc\_serv \*nlmsvc\_serv;

int nlmsvc\_grace\_period;

unsigned long nlmsvc\_timeout;

@@ -128,7 +128,8 @@ lockd(struct svc\_rqst \*rqstp)

/\*

\* Let our maker know we're running.

\*/

- nlmsvc\_pid = current->pid;

+ rcu\_assign\_pointer(nlmsvc\_task, current);

+

nlmsvc\_serv = rqstp->rq\_server;

complete(&lockd\_start\_done);

@@ -151,7 +152,7 @@ lockd(struct svc\_rqst \*rqstp)

\* NFS mount or NFS daemon has gone away, and we've been sent a

\* signal, or else another process has taken over our job.

\*/

- while ((nlmsvc\_users || !signalled()) && nlmsvc\_pid == current->pid) {

+ while ((nlmsvc\_users || !signalled()) && nlmsvc\_task == current) {

long timeout = MAX\_SCHEDULE\_TIMEOUT;

char buf[RPC\_MAX\_ADDRBUFLen];

```

@@ -200,12 +201,12 @@ lockd(struct svc_rqst *rqstp)
 * Check whether there's a new lockd process before
 * shutting down the hosts and clearing the slot.
 */
- if (!nlmsvc_pid || current->pid == nlmsvc_pid) {
+ if (nlmsvc_task == NULL || current == nlmsvc_task) {
    if (nlmsvc_ops)
        nlmsvc_invalidate_all();
    nlm_shutdown_hosts();
- nlmsvc_pid = 0;
    nlmsvc_serv = NULL;
+ rcu_assign_pointer(nlmsvc_task, NULL);
} else
    printk(KERN_DEBUG
        "lockd: new process, skipping host shutdown\n");
@@ -273,7 +274,7 @@ lockd_up(int proto) /* Maybe add a 'family' option when IPv6 is supported
?? */
/*
 * Check whether we're already up and running.
 */
- if (nlmsvc_pid) {
+ if (nlmsvc_task != NULL) {
    if (proto)
        error = make_socks(nlmsvc_serv, proto);
    goto out;
@@ -329,38 +330,49 @@ void
lockd_down(void)
{
    static int warned;
+ struct task_struct *tsk;

    mutex_lock(&nlmsvc_mutex);
+ rcu_read_lock();
+ tsk = rcu_dereference(nlmsvc_task);
    if (nlmsvc_users) {
        if (--nlmsvc_users)
-         goto out;
+         goto out_rcu_unlock;
    } else
-     printk(KERN_WARNING "lockd_down: no users! pid=%d\n", nlmsvc_pid);
+     printk(KERN_WARNING "lockd_down: no users! pid=%d\n",
+         task_pid_nr(tsk));

- if (!nlmsvc_pid) {
+ if (tsk == NULL) {
    if (warned++ == 0)
        printk(KERN_WARNING "lockd_down: no lockd running.\n");

```

```

- goto out;
+ goto out_rcu_unlock;
}
warned = 0;

- kill_proc(nlmsvc_pid, SIGKILL, 1);
+ send_sig(SIGKILL, tsk, 1);
+ rcu_read_unlock();
+
/*
 * Wait for the lockd process to exit, but since we're holding
 * the lockd semaphore, we can't wait around forever ...
 */
clear_thread_flag(TIF_SIGPENDING);
interruptible_sleep_on_timeout(&lockd_exit, HZ);
- if (nlmsvc_pid) {
+ if (nlmsvc_task != NULL) {
    printk(KERN_WARNING
        "lockd_down: lockd failed to exit, clearing pid\n");
- nlmsvc_pid = 0;
+ rcu_assign_pointer(nlmsvc_task, NULL);
    }
    spin_lock_irq(&current->sigband->siglock);
    recalc_sigpending();
    spin_unlock_irq(&current->sigband->siglock);
out:
    mutex_unlock(&nlmsvc_mutex);
+ return;
+
+out_rcu_unlock:
+ rcu_read_unlock();
+ goto out;
}
EXPORT_SYMBOL(lockd_down);

```

--

1.5.3.rc5

---