
Subject: [PATCH] ipv4/ipvs: Convert to kthread API
Posted by [ebiederm](#) on Thu, 19 Apr 2007 06:55:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

Modify startup of ipvs sync threads to use kthread_run instead of a weird combination of calling kernel_thread to start a fork_sync_thread whose whole purpose in life was to call kernel_thread again starting the actually sync thread which called daemonize.

To use kthread_run I had to move the name calculation from sync_thread into start_sync_thread resulting in a small amount of code motion.

The result is simpler and more maintainable piece of code.

Cc: Wensong Zhang <wensong@linux-vs.org>

Cc: Julian Anastasov <ja@ssi.bg>

Cc: Simon Horman <horms@verge.net.au>

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

net/ipv4/ipvs/ip_vs_sync.c | 49 ++++++++-----

1 files changed, 12 insertions(+), 37 deletions(-)

diff --git a/net/ipv4/ipvs/ip_vs_sync.c b/net/ipv4/ipvs/ip_vs_sync.c

index 7ea2d98..c4be9dc 100644

--- a/net/ipv4/ipvs/ip_vs_sync.c

+++ b/net/ipv4/ipvs/ip_vs_sync.c

@@ -29,6 +29,7 @@

#include <linux/in.h>

#include <linux/igmp.h> /* for ip_mc_join_group */

#include <linux/udp.h>

+#include <linux/kthread.h>

#include <net/ip.h>

#include <net/sock.h>

@@ -750,34 +751,23 @@ static int sync_thread(void *startup)

DECLARE_WAITQUEUE(wait, current);

mm_segment_t oldmm;

int state;

- const char *name;

/* increase the module use count */

ip_vs_use_count_inc();

- if (ip_vs_sync_state & IP_VS_STATE_MASTER && !sync_master_pid) {

```

+ if (ip_vs_sync_state & IP_VS_STATE_MASTER && !sync_master_pid)
    state = IP_VS_STATE_MASTER;
- name = "ipvs_syncmaster";
- } else if (ip_vs_sync_state & IP_VS_STATE_BACKUP && !sync_backup_pid) {
+ else if (ip_vs_sync_state & IP_VS_STATE_BACKUP && !sync_backup_pid)
    state = IP_VS_STATE_BACKUP;
- name = "ipvs_syncbackup";
- } else {
+ else {
    IP_VS_BUG();
    ip_vs_use_count_dec();
    return -EINVAL;
}

- daemonize(name);
-
    oldmm = get_fs();
    set_fs(KERNEL_DS);

- /* Block all signals */
- spin_lock_irq(&current->siglock);
- siginitsetinv(&current->blocked, 0);
- recalc_sigpending();
- spin_unlock_irq(&current->siglock);
-
    /* set the maximum length of sync message */
    set_sync_mesg_maxlen(state);

@@ -815,29 +805,11 @@ static int sync_thread(void *startup)
    return 0;
}

-
-static int fork_sync_thread(void *startup)
-{
- pid_t pid;
-
- /* fork the sync thread here, then the parent process of the
-  sync thread is the init process after this thread exits. */
- repeat:
- if ((pid = kernel_thread(sync_thread, startup, 0)) < 0) {
- IP_VS_ERR("could not create sync_thread due to %d... "
- "retrying.\n", pid);
- msleep_interruptible(1000);
- goto repeat;
- }
-
- return 0;

```

```

-}
-
-
int start_sync_thread(int state, char *mcast_ifn, __u8 syncid)
{
    DECLARE_COMPLETION_ONSTACK(startup);
- pid_t pid;
+ struct task_struct *task;
+ const char *name;

    if ((state == IP_VS_STATE_MASTER && sync_master_pid) ||
        (state == IP_VS_STATE_BACKUP && sync_backup_pid))
@@ -852,16 +824,19 @@ int start_sync_thread(int state, char *mcast_ifn, __u8 syncid)
    strcpy(ip_vs_master_mcast_ifn, mcast_ifn,
        sizeof(ip_vs_master_mcast_ifn));
    ip_vs_master_syncid = syncid;
+ name = "ipvs_syncmaster";
} else {
    strcpy(ip_vs_backup_mcast_ifn, mcast_ifn,
        sizeof(ip_vs_backup_mcast_ifn));
    ip_vs_backup_syncid = syncid;
+ name = "ipvs_syncbackup";
}

repeat:
- if ((pid = kernel_thread(fork_sync_thread, &startup, 0)) < 0) {
- IP_VS_ERR("could not create fork_sync_thread due to %d... "
-     "retrying.\n", pid);
+ task = kthread_run(sync_thread, &startup, name);
+ if (IS_ERR(task)) {
+ IP_VS_ERR("could not create sync_thread due to %ld... "
+     "retrying.\n", PTR_ERR(task));
    msleep_interruptible(1000);
    goto repeat;
}
--
1.5.0.g53756

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
