Subject: [PATCH 3/6] Lockd: per-net up and down routines introduced
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:08:05 GMT
View Forum Message <> Reply to Message

This patch introduces per-net Lockd initialization and destruction routines.
The logic is the same as in global Lockd up and down routines. Probably the
solution is not the best one. But at least it looks clear.
So per-net "up" routine are called only in case of lockd is running already. If
per-net resources are not allocated yet, then service is being registered with
local portmapper and lockd sockets created.
Per-net "down" routine is called on every lockd_down() call in case of global
users counter is not zero.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

---
 fs/lockd/svc.c | 47 ++--
 include/linux/sunrpc/svc.h | 2 ++
 net/sunrpc/svc.c | 3 ++-
 3 files changed, 49 insertions(+), 3 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index b461733..86e17e8 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -251,6 +251,45 @@ out_err:
  return err;
 }

+static int lockd_up_net(struct net *net)
+{
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+ struct svc_serv *serv = nlmsvc_rqst->rq_server;
+ int error;
+
+ if (ln->nlmsvc_users)
+ return 0;
+
+ error = svc_rpcb_setup(serv, net);
+ if (error)
+ goto err_rpcb;
+
+ error = make_socks(serv, net);
+ if (error < 0)
+ goto err_socks;
+ return 0;
+
+err_socks:

```
+ svc_rpcb_cleanup(serv, net);
+err_rpcb:
+ return error;
+}
+
+static void lockd_down_net(struct net *net)
+{
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+ struct svc_serv *serv = nlmsvc_rqst->rq_server;
+
+ if (ln->nlmsvc_users) {
+  if (--ln->nlmsvc_users == 0)
+   svc_shutdown_net(serv, net);
+ } else {
+  printk(KERN_ERR "lockd_down_net: no users! task=%p, net=%p\n",
+    nlmsvc_task, net);
+  BUG();
+ }
+}
+
 /*
  * Bring up the lockd process if it's not already up.
  */
@@ -264,8 +303,10 @@ int lockd_up(void)
  /*
  * Check whether we're already up and running.
  */
- if (nlmsvc_rqst)
+ if (nlmsvc_rqst) {
+  error = lockd_up_net(net);
   goto out;
+ }

  /*
  * Sanity check: if there's no pid,
@@ -338,8 +379,10 @@ lockd_down(void)
 {
  mutex_lock(&nlmsvc_mutex);
  if (nlmsvc_users) {
-  if (--nlmsvc_users)
+  if (--nlmsvc_users) {
+   lockd_down_net(current->nsproxy->net_ns);
    goto out;
+  }
  } else {
   printk(KERN_ERR "lockd_down: no users! task=%p\n",
    nlmsvc_task);
diff --git a/include/linux/sunrpc/svc.h b/include/linux/sunrpc/svc.h
```

```
index 7b65495..51b29ac 100644
--- a/include/linux/sunrpc/svc.h
+++ b/include/linux/sunrpc/svc.h
@@ -414,6 +414,7 @@ struct svc_procedure {
 /*
  * Function prototypes.
  */
+int svc_rpcb_setup(struct svc_serv *serv, struct net *net);
 void svc_rpcb_cleanup(struct svc_serv *serv, struct net *net);
 struct svc_serv *svc_create(struct svc_program *, unsigned int,
       void (*shutdown)(struct svc_serv *, struct net *net));
@@ -426,6 +427,7 @@ struct svc_serv *  svc_create_pooled(struct svc_program *, unsigned int,
 int     svc_set_num_threads(struct svc_serv *, struct svc_pool *, int);
 int     svc_pool_stats_open(struct svc_serv *serv, struct file *file);
 void    svc_destroy(struct svc_serv *);
+void    svc_shutdown_net(struct svc_serv *, struct net *);
 int     svc_process(struct svc_rqst *);
 int     bc_svc_process(struct svc_serv *, struct rpc_rqst *,
    struct svc_rqst *);
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index 78abac4..4153846 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -369,7 +369,7 @@ svc_pool_for_cpu(struct svc_serv *serv, int cpu)
  return &serv->sv_pools[pidx % serv->sv_nrpools];
 }

-static int svc_rpcb_setup(struct svc_serv *serv, struct net *net)
+int svc_rpcb_setup(struct svc_serv *serv, struct net *net)
 {
  int err;

@@ -381,6 +381,7 @@ static int svc_rpcb_setup(struct svc_serv *serv, struct net *net)
  svc_unregister(serv, net);
  return 0;
 }
+EXPORT_SYMBOL_GPL(svc_rpcb_setup);

 void svc_rpcb_cleanup(struct svc_serv *serv, struct net *net)
 {
```