Subject: [PATCH v2 1/3] lockd: per-net NSM client creation and destruction helpers introduced
Posted by Stanislav Kinsbursky on Tue, 18 Sep 2012 09:37:12 GMT
View Forum Message <> Reply to Message

NSM RPC client can be required on NFSv3 umount, when child reaper is dying (and
destroying it's mount namespace). It means, that current nsproxy is set to
NULL already, but creation of RPC client requires UTS namespace for gaining
hostname string.
This patch introduces reference counted NFS RPC clients creation and
destruction helpers (similar to RPCBIND RPC clients).

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Cc: <stable@vger.kernel.org>
---
 fs/lockd/mon.c | 51 ++--
 fs/lockd/netns.h | 4 ++++
 fs/lockd/svc.c | 1 +
 3 files changed, 54 insertions(+), 2 deletions(-)

diff --git a/fs/lockd/mon.c b/fs/lockd/mon.c
index 7ef14b3..38f240e 100644
--- a/fs/lockd/mon.c
+++ b/fs/lockd/mon.c
@@ -19,6 +19,8 @@

 #include <asm/unaligned.h>

+#include "netns.h"
+
 #define NLMDBG_FACILITY NLMDBG_MONITOR
 #define NSM_PROGRAM 100024
 #define NSM_VERSION 1
@@ -70,7 +72,7 @@ static struct rpc_clnt *nsm_create(struct net *net)
  };
  struct rpc_create_args args = {
   .net   = net,
- .protocol = XPRT_TRANSPORT_UDP,
+ .protocol = XPRT_TRANSPORT_TCP,
   .address = (struct sockaddr *)&sin,
   .addrsize = sizeof(sin),
   .servername = "rpc.statd",
@@ -83,6 +85,51 @@ static struct rpc_clnt *nsm_create(struct net *net)
  return rpc_create(&args);
 }

+__maybe_unused static struct rpc_clnt *nsm_client_get(struct net *net)
+{

```
+ static DEFINE_MUTEX(nsm_create_mutex);
+ struct rpc_clnt *clnt;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+ spin_lock(&ln->nsm_clnt_lock);
+ if (ln->nsm_users) {
+  ln->nsm_users++;
+  clnt = ln->nsm_clnt;
+  spin_unlock(&ln->nsm_clnt_lock);
+  goto out;
+ }
+ spin_unlock(&ln->nsm_clnt_lock);
+
+ mutex_lock(&nsm_create_mutex);
+ clnt = nsm_create(net);
+ if (!IS_ERR(clnt)) {
+  ln->nsm_clnt = clnt;
+  smp_wmb();
+  ln->nsm_users = 1;
+ }
+ mutex_unlock(&nsm_create_mutex);
+out:
+ return clnt;
+}
+
+__maybe_unused static void nsm_client_put(struct net *net)
+{
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+ struct rpc_clnt *clnt = ln->nsm_clnt;
+ int shutdown = 0;
+
+ spin_lock(&ln->nsm_clnt_lock);
+ if (ln->nsm_users) {
+  if (--ln->nsm_users)
+   ln->nsm_clnt = NULL;
+  shutdown = !ln->nsm_users;
+ }
+ spin_unlock(&ln->nsm_clnt_lock);
+
+ if (shutdown)
+  rpc_shutdown_client(clnt);
+}
+
 static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct nsm_res *res,
     struct net *net)
 {
@@ -111,7 +158,7 @@ static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct
nsm_res *res,
```

```
  memset(res, 0, sizeof(*res));

  msg.rpc_proc = &clnt->cl_procinfo[proc];
- status = rpc_call_sync(clnt, &msg, 0);
+ status = rpc_call_sync(clnt, &msg, RPC_TASK_SOFTCONN);
  if (status < 0)
   dprintk("lockd: NSM upcall RPC failed, status=%d\n",
     status);
diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index 4eee248..5010b55 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -12,6 +12,10 @@ struct lockd_net {
  struct delayed_work grace_period_end;
  struct lock_manager lockd_manager;
  struct list_head grace_list;
+
+ spinlock_t nsm_clnt_lock;
+ unsigned int nsm_users;
+ struct rpc_clnt *nsm_clnt;
 };

 extern int lockd_net_id;
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 31a63f8..7e35587 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -596,6 +596,7 @@ static int lockd_init_net(struct net *net)

  INIT_DELAYED_WORK(&ln->grace_period_end, grace_ender);
  INIT_LIST_HEAD(&ln->grace_list);
+ spin_lock_init(&ln->nsm_clnt_lock);
  return 0;
 }
```