Subject: [PATCH 0/6] Lockd: make it network namespace aware
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:07:38 GMT
View Forum Message <> Reply to Message

With this patch set Lockd will be able to handle lock requests from different
network namespaces separately.

Main ideas of the patch set are:

1) per-net Lockd users counter and resources.

2) nlmsvc_users counter become global one (equal to sum of all per-net
counters).

3) On lockd_up() call:
a) if nlmsvc_users if equal to 0, then lockd thread is started.
b) if current per-net counter equal to 0, then per-net resources are allocated
(lockd_up_net() function).
c) global and current net users counters are increased by one.

4) On lockd_down() call:
a) global and current net users counters are decreased by one.
b) if current per-net counter become equal to 0, then per-net resources are
allocated (lockd_down_net() function).
c) if nlmsvc_users become equal to 0, then lockd thread is stopped.

The following series consists of:

Stanislav Kinsbursky (6):
    Lockd: create permanent lockd sockets in current network namespace
    Lockd: pernet usage counter introduced
    Lockd: per-net up and down routines introduced
    LockD: make nlm hosts network namespace aware
    LockD: make NSM network namespace aware
    Lockd: shutdown NLM hosts in network namespace context

```
 fs/lockd/clntlock.c        |    3 +
 fs/lockd/host.c            |   42 ++++++++++++---
 fs/lockd/mon.c             |   13 +++--
 fs/lockd/netns.h           |   12 ++++
 fs/lockd/svc.c             |  117 ++++++++++++++++++++++++++++++++++++++------
 fs/nfs/client.c            |    1
 include/linux/lockd/bind.h |    1
 include/linux/lockd/lockd.h |    5 +-
 include/linux/sunrpc/svc.h |    2 +
```

net/sunrpc/svc.c | 3 +
10 files changed, 166 insertions(+), 33 deletions(-)
create mode 100644 fs/lockd/netns.h

Subject: [PATCH 1/6] Lockd: create permanent lockd sockets in current network namespace
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:07:48 GMT
View Forum Message <> Reply to Message

This patch parametrizes Lockd permanent sockets creation routine by network namespace context.
It also replaces hard-coded init_net with current network namespace context in Lockd sockets creation routines.
This approach looks safe, because Lockd is created during NFS mount (or NFS server start) and thus socket is required exactly in current network namespace context. But in the same time it means, that Lockd sockets inherits first Lockd requester network namespace. This issue will be fixed in further patches of the series.

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

---
fs/lockd/svc.c | 23 ++++++++++++++----------
1 files changed, 13 insertions(+), 10 deletions(-)

```
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 55fea92..26d8b78 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -189,27 +189,29 @@ lockd(void *vrqstp)
 }

 static int create_lockd_listener(struct svc_serv *serv, const char *name,
-      const int family, const unsigned short port)
+      struct net *net, const int family,
+      const unsigned short port)
 {
  struct svc_xprt *xprt;

- xprt = svc_find_xprt(serv, name, &init_net, family, 0);
+ xprt = svc_find_xprt(serv, name, net, family, 0);
  if (xprt == NULL)
-  return svc_create_xprt(serv, name, &init_net, family, port,
+  return svc_create_xprt(serv, name, net, family, port,
        SVC_SOCK_DEFAULTS);
  svc_xprt_put(xprt);
  return 0;
```

```
 }

-static int create_lockd_family(struct svc_serv *serv, const int family)
+static int create_lockd_family(struct svc_serv *serv, struct net *net,
+          const int family)
 {
  int err;

- err = create_lockd_listener(serv, "udp", family, nlm_udpport);
+ err = create_lockd_listener(serv, "udp", net, family, nlm_udpport);
  if (err < 0)
   return err;

- return create_lockd_listener(serv, "tcp", family, nlm_tcpport);
+ return create_lockd_listener(serv, "tcp", net, family, nlm_tcpport);
 }

 /*
@@ -222,16 +224,16 @@ static int create_lockd_family(struct svc_serv *serv, const int family)
  * Returns zero if all listeners are available; otherwise a
  * negative errno value is returned.
  */
-static int make_socks(struct svc_serv *serv)
+static int make_socks(struct svc_serv *serv, struct net *net)
 {
  static int warned;
  int err;

- err = create_lockd_family(serv, PF_INET);
+ err = create_lockd_family(serv, net, PF_INET);
  if (err < 0)
   goto out_err;

- err = create_lockd_family(serv, PF_INET6);
+ err = create_lockd_family(serv, net, PF_INET6);
  if (err < 0 && err != -EAFNOSUPPORT)
   goto out_err;

@@ -252,6 +254,7 @@ int lockd_up(void)
 {
  struct svc_serv *serv;
  int  error = 0;
+ struct net *net = current->nsproxy->net_ns;

  mutex_lock(&nlmsvc_mutex);
  /*
@@ -275,7 +278,7 @@ int lockd_up(void)
   goto out;
```

```
 }

- error = make_socks(serv);
+ error = make_socks(serv, net);
  if (error < 0)
   goto destroy_and_out;
```

Subject: [PATCH 2/6] Lockd: pernet usage counter introduced
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:07:57 GMT
View Forum Message <> Reply to Message

Lockd is going to be shared between network namespaces - i.e. going to be able
to handle lock requests from different network namespaces. This means, that
network namespace related resources have to be allocated not once (like now),
but for every network namespace context, from which service is requested to
operate.
This patch implements Lockd per-net users accounting. New per-net counter is
used to determine, when per-net resources have to be freed.

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

```
---
 fs/lockd/netns.h |   12 ++++++++++
 fs/lockd/svc.c   |   45 +++++++++++++++++++++++++++++++++++++++++++++---
 2 files changed, 54 insertions(+), 3 deletions(-)
 create mode 100644 fs/lockd/netns.h

diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
new file mode 100644
index 0000000..ce227e0
--- /dev/null
+++ b/fs/lockd/netns.h
@@ -0,0 +1,12 @@
+#ifndef __LOCKD_NETNS_H__
+#define __LOCKD_NETNS_H__
+
+#include <net/netns/generic.h>
+
+struct lockd_net {
+ unsigned int nlmsvc_users;
+};
+
+extern int lockd_net_id;
+
+#endif
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 26d8b78..b461733 100644
```

```
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -35,6 +35,8 @@
 #include <linux/lockd/lockd.h>
 #include <linux/nfs.h>

+#include "netns.h"
+
 #define NLMDBG_FACILITY  NLMDBG_SVC
 #define LOCKD_BUFSIZE  (1024 + NLMSVC_XDRSIZE)
 #define ALLOWED_SIGS  (sigmask(SIGKILL))
@@ -50,6 +52,8 @@ static struct task_struct *nlmsvc_task;
 static struct svc_rqst  *nlmsvc_rqst;
 unsigned long   nlmsvc_timeout;

+int lockd_net_id;
+
 /*
  * These can be set at insmod time (useful for NFS as root filesystem),
  * and also changed through the sysctl interface.  -- Jamie Lokier, Aug 2003
@@ -315,8 +319,12 @@ int lockd_up(void)
 destroy_and_out:
  svc_destroy(serv);
 out:
- if (!error)
+ if (!error) {
+  struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+  ln->nlmsvc_users++;
   nlmsvc_users++;
+ }
  mutex_unlock(&nlmsvc_mutex);
  return error;
 }
@@ -499,24 +507,55 @@ module_param_call(nlm_tcpport, param_set_port, param_get_int,
 module_param(nsm_use_hostnames, bool, 0644);
 module_param(nlm_max_connections, uint, 0644);

+static int lockd_init_net(struct net *net)
+{
+ return 0;
+}
+
+static void lockd_exit_net(struct net *net)
+{
+}
+
+static struct pernet_operations lockd_net_ops = {
```

```
+ .init = lockd_init_net,
+ .exit = lockd_exit_net,
+ .id = &lockd_net_id,
+ .size = sizeof(struct lockd_net),
+};
+
+
 /*
  * Initialising and terminating the module.
  */

 static int __init init_nlm(void)
 {
+ int err;
+
 #ifdef CONFIG_SYSCTL
+ err = -ENOMEM;
  nlm_sysctl_table = register_sysctl_table(nlm_sysctl_root);
- return nlm_sysctl_table ? 0 : -ENOMEM;
-#else
+ if (nlm_sysctl_table == NULL)
+  goto err_sysctl;
+#endif
+ err = register_pernet_subsys(&lockd_net_ops);
+ if (err)
+  goto err_pernet;
  return 0;
+
+err_pernet:
+#ifdef CONFIG_SYSCTL
+ unregister_sysctl_table(nlm_sysctl_table);
 #endif
+err_sysctl:
+ return err;
 }

 static void __exit exit_nlm(void)
 {
  /* FIXME: delete all NLM clients */
  nlm_shutdown_hosts();
+ unregister_pernet_subsys(&lockd_net_ops);
 #ifdef CONFIG_SYSCTL
  unregister_sysctl_table(nlm_sysctl_table);
 #endif
```

Subject: [PATCH 3/6] Lockd: per-net up and down routines introduced

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)
 {
```

Subject: [PATCH 4/6] LockD: make nlm hosts network namespace aware
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:08:13 GMT
View Forum Message <> Reply to Message

This object depends on RPC client, and thus on network namespace.
So let's make it's allocation and lookup in network namespace context.

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

```
---
 fs/lockd/clntlock.c         |    3 ++-
 fs/lockd/host.c             |   16 ++++++++++++++--
 fs/nfs/client.c             |    1 +
 include/linux/lockd/bind.h  |    1 +
 include/linux/lockd/lockd.h |    4 +++-
 5 files changed, 21 insertions(+), 4 deletions(-)

diff --git a/fs/lockd/clntlock.c b/fs/lockd/clntlock.c
index 8d4ea83..ba1dc2e 100644
--- a/fs/lockd/clntlock.c
+++ b/fs/lockd/clntlock.c
@@ -62,7 +62,8 @@ struct nlm_host *nlmclnt_init(const struct nlmclnt_initdata *nlm_init)

  host = nlmclnt_lookup_host(nlm_init->address, nlm_init->addrlen,
        nlm_init->protocol, nlm_version,
-       nlm_init->hostname, nlm_init->noresvport);
+       nlm_init->hostname, nlm_init->noresvport,
+       nlm_init->net);
  if (host == NULL) {
   lockd_down();
   return ERR_PTR(-ENOLCK);
diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 6f29836..9ebd91d 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -17,6 +17,8 @@
 #include <linux/lockd/lockd.h>
 #include <linux/mutex.h>

+#include <linux/sunrpc/svc_xprt.h>
+
 #include <net/ipv6.h>

 #define NLMDBG_FACILITY  NLMDBG_HOSTCACHE
@@ -54,6 +56,7 @@ struct nlm_lookup_host_info {
  const char  *hostname; /* remote's hostname */
  const size_t  hostname_len; /* it's length */
  const int  noresvport; /* use non-priv port */
+ struct net  *net;  /* network namespace to bind */
 };

 /*
@@ -155,6 +158,7 @@ static struct nlm_host *nlm_alloc_host(struct nlm_lookup_host_info *ni,
  INIT_LIST_HEAD(&host->h_reclaim);
  host->h_nsmhandle  = nsm;
  host->h_addrbuf    = nsm->sm_addrbuf;
```

```
+ host->net    = ni->net;

 out:
  return host;
@@ -206,7 +210,8 @@ struct nlm_host *nlmclnt_lookup_host(const struct sockaddr *sap,
          const unsigned short protocol,
          const u32 version,
          const char *hostname,
-         int noresvport)
+         int noresvport,
+         struct net *net)
 {
  struct nlm_lookup_host_info ni = {
   .server  = 0,
@@ -217,6 +222,7 @@ struct nlm_host *nlmclnt_lookup_host(const struct sockaddr *sap,
   .hostname = hostname,
   .hostname_len = strlen(hostname),
   .noresvport = noresvport,
+  .net  = net,
  };
  struct hlist_head *chain;
  struct hlist_node *pos;
@@ -231,6 +237,8 @@ struct nlm_host *nlmclnt_lookup_host(const struct sockaddr *sap,

  chain = &nlm_client_hosts[nlm_hash_address(sap)];
  hlist_for_each_entry(host, pos, chain, h_hash) {
+  if (host->net != net)
+   continue;
   if (!rpc_cmp_addr(nlm_addr(host), sap))
    continue;

@@ -318,6 +326,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
  struct nsm_handle *nsm = NULL;
  struct sockaddr *src_sap = svc_daddr(rqstp);
  size_t src_len = rqstp->rq_daddrlen;
+ struct net *net = rqstp->rq_xprt->xpt_net;
  struct nlm_lookup_host_info ni = {
   .server  = 1,
   .sap  = svc_addr(rqstp),
@@ -326,6 +335,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
   .version = rqstp->rq_vers,
   .hostname = hostname,
   .hostname_len = hostname_len,
+  .net  = net,
  };

  dprintk("lockd: %s(host='%*s', vers=%u, proto=%s)\n", __func__,
@@ -339,6 +349,8 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
```

```
  chain = &nlm_server_hosts[nlm_hash_address(ni.sap)];
  hlist_for_each_entry(host, pos, chain, h_hash) {
+  if (host->net != net)
+   continue;
   if (!rpc_cmp_addr(nlm_addr(host), ni.sap))
    continue;

@@ -431,7 +443,7 @@ nlm_bind_host(struct nlm_host *host)
    .to_retries = 5U,
   };
   struct rpc_create_args args = {
-   .net  = &init_net,
+   .net  = host->net,
    .protocol = host->h_proto,
    .address = nlm_addr(host),
    .addrsize = host->h_addrlen,
diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 2328dcb..1a5cd49 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -707,6 +707,7 @@ static int nfs_start_lockd(struct nfs_server *server)
   .nfs_version = clp->rpc_ops->version,
   .noresvport = server->flags & NFS_MOUNT_NORESVPORT ?
     1 : 0,
+  .net  = clp->net,
  };

  if (nlm_init.nfs_version > 3)
diff --git a/include/linux/lockd/bind.h b/include/linux/lockd/bind.h
index fbc48f8..11a966e 100644
--- a/include/linux/lockd/bind.h
+++ b/include/linux/lockd/bind.h
@@ -42,6 +42,7 @@ struct nlmclnt_initdata {
  unsigned short  protocol;
  u32   nfs_version;
  int   noresvport;
+ struct net  *net;
 };

 /*
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index 8949167..94b3d13 100644
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -67,6 +67,7 @@ struct nlm_host {
  struct list_head h_reclaim; /* Locks in RECLAIM state */
  struct nsm_handle *h_nsmhandle; /* NSM status handle */
```

```
  char   *h_addrbuf; /* address eyecatcher */
+ struct net  *net;  /* host net */
 };

 /*
@@ -222,7 +223,8 @@ struct nlm_host  *nlmclnt_lookup_host(const struct sockaddr *sap,
      const unsigned short protocol,
      const u32 version,
      const char *hostname,
-     int noresvport);
+     int noresvport,
+     struct net *net);
 void    nlmclnt_release_host(struct nlm_host *);
 struct nlm_host  *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
      const char *hostname,
```

Subject: [PATCH 5/6] LockD: make NSM network namespace aware
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:08:21 GMT
View Forum Message <> Reply to Message

NLM host is network namespace aware now.
So NSM have to take it into account.

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

```
---
 fs/lockd/mon.c |   13 +++++++------
 1 files changed, 7 insertions(+), 6 deletions(-)

diff --git a/fs/lockd/mon.c b/fs/lockd/mon.c
index c196030..7ef14b3 100644
--- a/fs/lockd/mon.c
+++ b/fs/lockd/mon.c
@@ -62,14 +62,14 @@ static inline struct sockaddr *nsm_addr(const struct nsm_handle *nsm)
  return (struct sockaddr *)&nsm->sm_addr;
 }

-static struct rpc_clnt *nsm_create(void)
+static struct rpc_clnt *nsm_create(struct net *net)
 {
  struct sockaddr_in sin = {
   .sin_family  = AF_INET,
   .sin_addr.s_addr = htonl(INADDR_LOOPBACK),
  };
  struct rpc_create_args args = {
-  .net   = &init_net,
+  .net   = net,
```

```
   .protocol  = XPRT_TRANSPORT_UDP,
   .address  = (struct sockaddr *)&sin,
   .addrsize  = sizeof(sin),
@@ -83,7 +83,8 @@ static struct rpc_clnt *nsm_create(void)
  return rpc_create(&args);
 }

-static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct nsm_res *res)
+static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct nsm_res *res,
+     struct net *net)
 {
  struct rpc_clnt *clnt;
  int  status;
@@ -99,7 +100,7 @@ static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct
nsm_res *res)
   .rpc_resp = res,
  };

- clnt = nsm_create();
+ clnt = nsm_create(net);
  if (IS_ERR(clnt)) {
   status = PTR_ERR(clnt);
   dprintk("lockd: failed to create NSM upcall transport, "
@@ -149,7 +150,7 @@ int nsm_monitor(const struct nlm_host *host)
   */
  nsm->sm_mon_name = nsm_use_hostnames ? nsm->sm_name : nsm->sm_addrbuf;

- status = nsm_mon_unmon(nsm, NSMPROC_MON, &res);
+ status = nsm_mon_unmon(nsm, NSMPROC_MON, &res, host->net);
  if (unlikely(res.status != 0))
   status = -EIO;
  if (unlikely(status < 0)) {
@@ -183,7 +184,7 @@ void nsm_unmonitor(const struct nlm_host *host)
   && nsm->sm_monitored && !nsm->sm_sticky) {
   dprintk("lockd: nsm_unmonitor(%s)\n", nsm->sm_name);

-  status = nsm_mon_unmon(nsm, NSMPROC_UNMON, &res);
+  status = nsm_mon_unmon(nsm, NSMPROC_UNMON, &res, host->net);
   if (res.status != 0)
    status = -EIO;
   if (status < 0)
```

Subject: [PATCH 6/6] Lockd: shutdown NLM hosts in network namespace context
Posted by Stanislav Kinsbursky on Tue, 31 Jan 2012 11:08:29 GMT
View Forum Message <> Reply to Message

Lockd now managed in network namespace context. And this patch introduces

network namespace related NLM hosts shutdown in case of releasing per-net Lockd
resources.

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

---
 fs/lockd/host.c | 26 ++++++++++++++++++-------
 fs/lockd/svc.c | 4 +++-
 include/linux/lockd/lockd.h | 1 +
 3 files changed, 23 insertions(+), 8 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 9ebd91d..eb75ca7 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -565,12 +565,8 @@ void nlm_host_rebooted(const struct nlm_reboot *info)
  nsm_release(nsm);
 }

-/*
- * Shut down the hosts module.
- * Note that this routine is called only at server shutdown time.
- */
 void
-nlm_shutdown_hosts(void)
+nlm_shutdown_hosts_net(struct net *net)
 {
  struct hlist_head *chain;
  struct hlist_node *pos;
@@ -582,6 +578,8 @@ nlm_shutdown_hosts(void)
  /* First, make all hosts eligible for gc */
  dprintk("lockd: nuking all hosts...\n");
  for_each_host(host, pos, chain, nlm_server_hosts) {
+ if (net && host->net != net)
+   continue;
   host->h_expires = jiffies - 1;
   if (host->h_rpcclnt) {
    rpc_shutdown_client(host->h_rpcclnt);
@@ -592,15 +590,29 @@ nlm_shutdown_hosts(void)
  /* Then, perform a garbage collection pass */
  nlm_gc_hosts();
  mutex_unlock(&nlm_host_mutex);
+}
+
+/*
+ * Shut down the hosts module.
+ * Note that this routine is called only at server shutdown time.
+ */

```
+void
+nlm_shutdown_hosts(void)
+{
+ struct hlist_head *chain;
+ struct hlist_node *pos;
+ struct nlm_host *host;
+
+ nlm_shutdown_hosts_net(NULL);

  /* complain if any hosts are left */
  if (nrhosts != 0) {
   printk(KERN_WARNING "lockd: couldn't shutdown host module!\n");
   dprintk("lockd: %lu hosts left:\n", nrhosts);
   for_each_host(host, pos, chain, nlm_server_hosts) {
-   dprintk("       %s (cnt %d use %d exp %ld)\n",
+   dprintk("       %s (cnt %d use %d exp %ld net %p)\n",
     host->h_name, atomic_read(&host->h_count),
-    host->h_inuse, host->h_expires);
+    host->h_inuse, host->h_expires, host->net);
   }
  }
 }
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 86e17e8..b34100e 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -281,8 +281,10 @@ static void lockd_down_net(struct net *net)
  struct svc_serv *serv = nlmsvc_rqst->rq_server;

  if (ln->nlmsvc_users) {
-  if (--ln->nlmsvc_users == 0)
+  if (--ln->nlmsvc_users == 0) {
+   nlm_shutdown_hosts_net(net);
    svc_shutdown_net(serv, net);
+  }
  } else {
   printk(KERN_ERR "lockd_down_net: no users! task=%p, net=%p\n",
     nlmsvc_task, net);
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index 94b3d13..f04ce6a 100644
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -234,6 +234,7 @@ struct rpc_clnt * nlm_bind_host(struct nlm_host *);
 void    nlm_rebind_host(struct nlm_host *);
 struct nlm_host * nlm_get_host(struct nlm_host *);
 void    nlm_shutdown_hosts(void);
+void    nlm_shutdown_hosts_net(struct net *net);
 void    nlm_host_rebooted(const struct nlm_reboot *);
```

```
/*
```