Subject: [PATCH v2 00/15] Lockd: grace period containerization
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:55:45 GMT
View Forum Message <> Reply to Message

Bruce, I feel this patch set is ready for inclusion.

v2:
1) Rebase on Bruce's "for-3.6" branch.

This patch set makes grace period and hosts reclaiming network namespace
aware.

Main ideas:
1) moving of

 unsigned long next_gc;
 unsigned long nrhosts;

 struct delayed_work grace_period_end;
 struct lock_manager lockd_manager;
 struct list_head grace_list;

to per-net Lockd data.

2) moving of

 struct lock_manager nfsd4_manager;

to per-net NFSd data.

3) shutdown + gc of NLM hosts done now network namespace aware.

4) restart_grace() now works only for init_net.

The following series implements...

Stanislav Kinsbursky (15):
    LockD: mark host per network namespace on garbage collect
    LockD: make garbage collector network namespace aware.
    LockD: manage garbage collection timeout per networks namespace
    LockD: manage used host count per networks namespace
    Lockd: host complaining function introduced
    Lockd: add more debug to host shutdown functions
    LockD: manage grace period per network namespace
    LockD: make lockd manager allocated per network namespace
    NFSd: make nfsd4_manager allocated per network namespace context.

SUNRPC: service request network namespace helper introduced
LockD: manage grace list per network namespace
LockD: pass actual network namespace to grace period management functions
Lockd: move grace period management from lockd() to per-net functions
NFSd: make grace end flag per network namespace
NFSd: make boot_time variable per network namespace

```
fs/lockd/grace.c          |   16 +++++--
fs/lockd/host.c           |   92 ++++++++++++++++++++++++++++++------------
fs/lockd/netns.h          |    7 +++
fs/lockd/svc.c            |   43 ++++++++++++--------
fs/lockd/svc4proc.c       |   13 +++--
fs/lockd/svclock.c        |   16 +++----
fs/lockd/svcproc.c        |   15 ++++--
fs/lockd/svcsubs.c        |   19 +++++---
fs/nfs/callback_xdr.c     |    4 +-
fs/nfsd/export.c          |    4 +-
fs/nfsd/netns.h           |    4 ++
fs/nfsd/nfs4idmap.c       |    4 +-
fs/nfsd/nfs4proc.c        |   18 ++++---
fs/nfsd/nfs4state.c       |  104 ++++++++++++++++++++++++++++++++-------------
fs/nfsd/state.h           |    4 +-
include/linux/fs.h        |    5 +-
include/linux/lockd/lockd.h |    6 +-
include/linux/sunrpc/svc.h  |    2 +
18 files changed, 231 insertions(+), 145 deletions(-)
```

Subject: [PATCH v2 01/15] LockD: mark host per network namespace on garbage collect

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:55:54 GMT

View Forum Message <> Reply to Message

This is required for per-network NLM shutdown and cleanup.
This patch passes init_net for a while.

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

```
---
fs/lockd/host.c           |    3 ++-
fs/lockd/svcsubs.c        |   19 +++++++++++------
include/linux/lockd/lockd.h |    2 +-
3 files changed, 16 insertions(+), 8 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index eb75ca7..2c5f41b 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
```

```
@@ -628,13 +628,14 @@ nlm_gc_hosts(void)
  struct hlist_head *chain;
  struct hlist_node *pos, *next;
  struct nlm_host *host;
+ struct net *net = &init_net;

  dprintk("lockd: host garbage collection\n");
  for_each_host(host, pos, chain, nlm_server_hosts)
   host->h_inuse = 0;

  /* Mark all hosts that hold locks, blocks or shares */
- nlmsvc_mark_resources();
+ nlmsvc_mark_resources(net);

  for_each_host_safe(host, pos, next, chain, nlm_server_hosts) {
   if (atomic_read(&host->h_count) || host->h_inuse
diff --git a/fs/lockd/svcsubs.c b/fs/lockd/svcsubs.c
index 2240d38..0deb5f6 100644
--- a/fs/lockd/svcsubs.c
+++ b/fs/lockd/svcsubs.c
@@ -309,7 +309,8 @@ nlm_release_file(struct nlm_file *file)
  * Helpers function for resource traversal
  *
  * nlmsvc_mark_host:
- * used by the garbage collector; simply sets h_inuse.
+ * used by the garbage collector; simply sets h_inuse only for those
+ * hosts, which passed network check.
  * Always returns 0.
  *
  * nlmsvc_same_host:
@@ -320,12 +321,15 @@ nlm_release_file(struct nlm_file *file)
  * returns 1 iff the host is a client.
  * Used by nlmsvc_invalidate_all
  */
+
 static int
-nlmsvc_mark_host(void *data, struct nlm_host *dummy)
+nlmsvc_mark_host(void *data, struct nlm_host *hint)
 {
  struct nlm_host *host = data;

- host->h_inuse = 1;
+ if ((hint->net == NULL) ||
+     (host->net == hint->net))
+  host->h_inuse = 1;
  return 0;
 }
```

```
@@ -358,10 +362,13 @@ nlmsvc_is_client(void *data, struct nlm_host *dummy)
  * Mark all hosts that still hold resources
  */
 void
-nlmsvc_mark_resources(void)
+nlmsvc_mark_resources(struct net *net)
 {
- dprintk("lockd: nlmsvc_mark_resources\n");
- nlm_traverse_files(NULL, nlmsvc_mark_host, NULL);
+ struct nlm_host hint;
+
+ dprintk("lockd: nlmsvc_mark_resources for net %p\n", net);
+ hint.net = net;
+ nlm_traverse_files(&hint, nlmsvc_mark_host, NULL);
 }

 /*
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index f04ce6a..50e31a2 100644
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -279,7 +279,7 @@ void    nlmsvc_release_call(struct nlm_rqst *);
 __be32    nlm_lookup_file(struct svc_rqst *, struct nlm_file **,
     struct nfs_fh *);
 void    nlm_release_file(struct nlm_file *);
-void    nlmsvc_mark_resources(void);
+void    nlmsvc_mark_resources(struct net *);
 void    nlmsvc_free_host_resources(struct nlm_host *);
 void    nlmsvc_invalidate_all(void);
```

Subject: [PATCH v2 02/15] LockD: make garbage collector network namespace aware.

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:03 GMT

View Forum Message <> Reply to Message

```
---
 fs/lockd/host.c |   22 ++++++++++++++---------
 1 files changed, 13 insertions(+), 9 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 2c5f41b..991274a 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -45,7 +45,7 @@ static unsigned long  next_gc;
 static unsigned long  nrhosts;
 static DEFINE_MUTEX(nlm_host_mutex);
```

```
-static void   nlm_gc_hosts(void);
+static void   nlm_gc_hosts(struct net *net);

 struct nlm_lookup_host_info {
  const int  server;  /* search for server|client */
@@ -345,7 +345,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
  mutex_lock(&nlm_host_mutex);

  if (time_after_eq(jiffies, next_gc))
-  nlm_gc_hosts();
+  nlm_gc_hosts(net);

  chain = &nlm_server_hosts[nlm_hash_address(ni.sap)];
  hlist_for_each_entry(host, pos, chain, h_hash) {
@@ -588,7 +588,7 @@ nlm_shutdown_hosts_net(struct net *net)
  }

  /* Then, perform a garbage collection pass */
- nlm_gc_hosts();
+ nlm_gc_hosts(net);
  mutex_unlock(&nlm_host_mutex);
 }

@@ -623,27 +623,31 @@ nlm_shutdown_hosts(void)
  * mark & sweep for resources held by remote clients.
  */
 static void
-nlm_gc_hosts(void)
+nlm_gc_hosts(struct net *net)
 {
  struct hlist_head *chain;
  struct hlist_node *pos, *next;
  struct nlm_host *host;
- struct net *net = &init_net;

- dprintk("lockd: host garbage collection\n");
- for_each_host(host, pos, chain, nlm_server_hosts)
+ dprintk("lockd: host garbage collection for net %p\n", net);
+ for_each_host(host, pos, chain, nlm_server_hosts) {
+  if (net && host->net != net)
+   continue;
   host->h_inuse = 0;
+ }

  /* Mark all hosts that hold locks, blocks or shares */
  nlmsvc_mark_resources(net);

  for_each_host_safe(host, pos, next, chain, nlm_server_hosts) {
```

```
+  if (net && host->net != net)
+   continue;
   if (atomic_read(&host->h_count) || host->h_inuse
    || time_before(jiffies, host->h_expires)) {
    dprintk("nlm_gc_hosts skipping %s "
-    "(cnt %d use %d exp %ld)\n",
+    "(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);
    continue;
   }
   nlm_destroy_host_locked(host);
```

Subject: [PATCH v2 03/15] LockD: manage garbage collection timeout per networks namespace

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:11 GMT

View Forum Message <> Reply to Message

This patch moves next_gc to per-net data.

Note: passed network can be NULL (when Lockd kthread is exiting of Lockd
module is removing).

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

```
---
 fs/lockd/host.c  |   12 ++++++++---
 fs/lockd/netns.h |    1 +
 2 files changed, 10 insertions(+), 3 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 991274a..3636734 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -21,6 +21,8 @@

 #include <net/ipv6.h>

+#include "netns.h"
+
 #define NLMDBG_FACILITY  NLMDBG_HOSTCACHE
 #define NLM_HOST_NRHASH  32
 #define NLM_HOST_REBIND  (60 * HZ)
@@ -41,7 +43,6 @@ static struct hlist_head nlm_client_hosts[NLM_HOST_NRHASH];
   hlist_for_each_entry_safe((host), (pos), (next), \
       (chain), h_hash)
```

```
-static unsigned long  next_gc;
 static unsigned long  nrhosts;
 static DEFINE_MUTEX(nlm_host_mutex);

@@ -337,6 +338,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
   .hostname_len = hostname_len,
   .net  = net,
  };
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  dprintk("lockd: %s(host='%*s', vers=%u, proto=%s)\n", __func__,
    (int)hostname_len, hostname, rqstp->rq_vers,
@@ -344,7 +346,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,

  mutex_lock(&nlm_host_mutex);

- if (time_after_eq(jiffies, next_gc))
+ if (time_after_eq(jiffies, ln->next_gc))
   nlm_gc_hosts(net);

  chain = &nlm_server_hosts[nlm_hash_address(ni.sap)];
@@ -653,5 +655,9 @@ nlm_gc_hosts(struct net *net)
   nlm_destroy_host_locked(host);
  }

- next_gc = jiffies + NLM_HOST_COLLECT;
+ if (net) {
+  struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+  ln->next_gc = jiffies + NLM_HOST_COLLECT;
+ }
 }
diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index ce227e0..97c6c77 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -5,6 +5,7 @@

 struct lockd_net {
  unsigned int nlmsvc_users;
+ unsigned long next_gc;
 };

 extern int lockd_net_id;
```

Subject: [PATCH v2 04/15] LockD: manage used host count per networks

This patch introduces moves nrhosts in per-net data.
It also adds kernel warning to nlm_shutdown_hosts_net() about remaining hosts
in specified network namespace context.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/host.c | 18 ++++++++++++++++++
 fs/lockd/netns.h | 1 +
 2 files changed, 19 insertions(+), 0 deletions(-)

```
diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 3636734..6c56090 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -173,6 +173,7 @@ out:
 static void nlm_destroy_host_locked(struct nlm_host *host)
 {
  struct rpc_clnt *clnt;
+ struct lockd_net *ln = net_generic(host->net, lockd_net_id);

  dprintk("lockd: destroy host %s\n", host->h_name);

@@ -189,6 +190,7 @@ static void nlm_destroy_host_locked(struct nlm_host *host)
   rpc_shutdown_client(clnt);
  kfree(host);

+ ln->nrhosts--;
  nrhosts--;
 }

@@ -229,6 +231,7 @@ struct nlm_host *nlmclnt_lookup_host(const struct sockaddr *sap,
  struct hlist_node *pos;
  struct nlm_host *host;
  struct nsm_handle *nsm = NULL;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  dprintk("lockd: %s(host='%s', vers=%u, proto=%s)\n", __func__,
    (hostname ? hostname : "<none>"), version,
@@ -263,6 +266,7 @@ struct nlm_host *nlmclnt_lookup_host(const struct sockaddr *sap,
   goto out;

  hlist_add_head(&host->h_hash, chain);
+ ln->nrhosts++;
  nrhosts++;
```

```
  dprintk("lockd: %s created host %s (%s)\n", __func__,
@@ -384,6 +388,7 @@ struct nlm_host *nlmsvc_lookup_host(const struct svc_rqst *rqstp,
  memcpy(nlm_srcaddr(host), src_sap, src_len);
  host->h_srcaddrlen = src_len;
  hlist_add_head(&host->h_hash, chain);
+ ln->nrhosts++;
  nrhosts++;

  dprintk("lockd: %s created host %s (%s)\n",
@@ -592,6 +597,19 @@ nlm_shutdown_hosts_net(struct net *net)
  /* Then, perform a garbage collection pass */
  nlm_gc_hosts(net);
  mutex_unlock(&nlm_host_mutex);
+
+ /* complain if any hosts are left */
+ if (net) {
+  struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+  printk(KERN_WARNING "lockd: couldn't shutdown host module for net %p!\n", net);
+  dprintk("lockd: %lu hosts left in net %p:\n", ln->nrhosts, net);
+  for_each_host(host, pos, chain, nlm_server_hosts) {
+   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->net);
+  }
+ }
 }

 /*
diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index 97c6c77..44c8f0b 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -6,6 +6,7 @@
 struct lockd_net {
  unsigned int nlmsvc_users;
  unsigned long next_gc;
+ unsigned long nrhosts;
 };

 extern int lockd_net_id;
```

Subject: [PATCH v2 05/15] Lockd: host complaining function introduced
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:27 GMT
View Forum Message <> Reply to Message

Just a small cleanup.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/host.c | 57 ++++++++++++++++++++++++++++++----------------------------
 1 files changed, 30 insertions(+), 27 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 6c56090..8cbf53d 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -572,6 +572,35 @@ void nlm_host_rebooted(const struct nlm_reboot *info)
  nsm_release(nsm);
 }

+static void nlm_complain_hosts(struct net *net)
+{
+ struct hlist_head *chain;
+ struct hlist_node *pos;
+ struct nlm_host *host;
+
+ if (net) {
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+ if (ln->nrhosts == 0)
+   return;
+ printk(KERN_WARNING "lockd: couldn't shutdown host module for net %p!\n", net);
+ dprintk("lockd: %lu hosts left in net %p:\n", ln->nrhosts, net);
+ } else {
+ if (nrhosts == 0)
+   return;
+ 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) {
+ if (net && host->net != net)
+   continue;
+ 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->net);
+ }
+}
+
 void
 nlm_shutdown_hosts_net(struct net *net)
 {
@@ -598,18 +627,7 @@ nlm_shutdown_hosts_net(struct net *net)
  nlm_gc_hosts(net);

```
  mutex_unlock(&nlm_host_mutex);

- /* complain if any hosts are left */
- if (net) {
-  struct lockd_net *ln = net_generic(net, lockd_net_id);
-
-  printk(KERN_WARNING "lockd: couldn't shutdown host module for net %p!\n", net);
-  dprintk("lockd: %lu hosts left in net %p:\n", ln->nrhosts, net);
-  for_each_host(host, pos, chain, nlm_server_hosts) {
-   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->net);
-  }
- }
+ nlm_complain_hosts(net);
 }

 /*
@@ -619,22 +637,7 @@ nlm_shutdown_hosts_net(struct net *net)
 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 net %p)\n",
-    host->h_name, atomic_read(&host->h_count),
-    host->h_inuse, host->h_expires, host->net);
-  }
- }
 }

 /*
```

Subject: [PATCH v2 06/15] Lockd: add more debug to host shutdown functions
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:35 GMT
View Forum Message <> Reply to Message

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

```
---
 fs/lockd/host.c |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 8cbf53d..0084ab8 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -608,11 +608,10 @@ nlm_shutdown_hosts_net(struct net *net)
  struct hlist_node *pos;
  struct nlm_host *host;

- dprintk("lockd: shutting down host module\n");
  mutex_lock(&nlm_host_mutex);

  /* First, make all hosts eligible for gc */
- dprintk("lockd: nuking all hosts...\n");
+ dprintk("lockd: nuking all hosts in net %p...\n", net);
  for_each_host(host, pos, chain, nlm_server_hosts) {
   if (net && host->net != net)
    continue;
@@ -637,6 +636,7 @@ nlm_shutdown_hosts_net(struct net *net)
 void
 nlm_shutdown_hosts(void)
 {
+ dprintk("lockd: shutting down host module\n");
  nlm_shutdown_hosts_net(NULL);
 }
```

Subject: [PATCH v2 07/15] LockD: manage grace period per network namespace
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:43 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/netns.h |    2 ++
 fs/lockd/svc.c   |   17 ++++++++++------
 2 files changed, 13 insertions(+), 6 deletions(-)

diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index 44c8f0b..94653ae 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -7,6 +7,8 @@ struct lockd_net {
  unsigned int nlmsvc_users;
  unsigned long next_gc;
  unsigned long nrhosts;
```

```
+
+ struct delayed_work grace_period_end;
 };

 extern int lockd_net_id;
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 80938fd..70c4177 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -95,21 +95,22 @@ static void grace_ender(struct work_struct *not_used)
  locks_end_grace(&lockd_manager);
 }

-static DECLARE_DELAYED_WORK(grace_period_end, grace_ender);
-
 static void set_grace_period(void)
 {
  unsigned long grace_period = get_lockd_grace_period();
+ struct lockd_net *ln = net_generic(&init_net, lockd_net_id);

  locks_start_grace(&lockd_manager);
- cancel_delayed_work_sync(&grace_period_end);
- schedule_delayed_work(&grace_period_end, grace_period);
+ cancel_delayed_work_sync(&ln->grace_period_end);
+ schedule_delayed_work(&ln->grace_period_end, grace_period);
 }

 static void restart_grace(void)
 {
  if (nlmsvc_ops) {
-  cancel_delayed_work_sync(&grace_period_end);
+  struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+
+  cancel_delayed_work_sync(&ln->grace_period_end);
   locks_end_grace(&lockd_manager);
   nlmsvc_invalidate_all();
   set_grace_period();
@@ -124,6 +125,7 @@ lockd(void *vrqstp)
 {
  int  err = 0, preverr = 0;
  struct svc_rqst *rqstp = vrqstp;
+ struct lockd_net *ln = net_generic(&init_net, lockd_net_id);

  /* try_to_freeze() is called from svc_recv() */
  set_freezable();
@@ -184,7 +186,7 @@ lockd(void *vrqstp)
   svc_process(rqstp);
  }
```

```
  flush_signals(current);
- cancel_delayed_work_sync(&grace_period_end);
+ cancel_delayed_work_sync(&ln->grace_period_end);
  locks_end_grace(&lockd_manager);
  if (nlmsvc_ops)
   nlmsvc_invalidate_all();
@@ -589,6 +591,9 @@ module_param(nlm_max_connections, uint, 0644);

 static int lockd_init_net(struct net *net)
 {
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+ INIT_DELAYED_WORK(&ln->grace_period_end, grace_ender);
  return 0;
 }
```

Subject: [PATCH v2 08/15] LockD: make lockd manager allocated per network namespace

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:50 GMT

View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/netns.h |    2 ++
 fs/lockd/svc.c   |   18 ++++++++++--------
 2 files changed, 12 insertions(+), 8 deletions(-)

diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index 94653ae..e78650c 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -1,6 +1,7 @@
 #ifndef __LOCKD_NETNS_H__
 #define __LOCKD_NETNS_H__

+#include <linux/fs.h>
 #include <net/netns/generic.h>

 struct lockd_net {
@@ -9,6 +10,7 @@ struct lockd_net {
  unsigned long nrhosts;

  struct delayed_work grace_period_end;
+ struct lock_manager lockd_manager;
 };

 extern int lockd_net_id;
```

```
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 70c4177..a9c436b 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -87,12 +87,14 @@ static unsigned long get_lockd_grace_period(void)
   return nlm_timeout * 5 * HZ;
 }

-static struct lock_manager lockd_manager = {
-};
-
-static void grace_ender(struct work_struct *not_used)
+static void grace_ender(struct work_struct *grace)
 {
- locks_end_grace(&lockd_manager);
+ struct delayed_work *dwork = container_of(grace, struct delayed_work,
+        work);
+ struct lockd_net *ln = container_of(dwork, struct lockd_net,
+         grace_period_end);
+
+ locks_end_grace(&ln->lockd_manager);
 }

 static void set_grace_period(void)
@@ -100,7 +102,7 @@ static void set_grace_period(void)
  unsigned long grace_period = get_lockd_grace_period();
  struct lockd_net *ln = net_generic(&init_net, lockd_net_id);

- locks_start_grace(&lockd_manager);
+ locks_start_grace(&ln->lockd_manager);
  cancel_delayed_work_sync(&ln->grace_period_end);
  schedule_delayed_work(&ln->grace_period_end, grace_period);
 }
@@ -111,7 +113,7 @@ static void restart_grace(void)
   struct lockd_net *ln = net_generic(&init_net, lockd_net_id);

   cancel_delayed_work_sync(&ln->grace_period_end);
-  locks_end_grace(&lockd_manager);
+  locks_end_grace(&ln->lockd_manager);
   nlmsvc_invalidate_all();
   set_grace_period();
  }
@@ -187,7 +189,7 @@ lockd(void *vrqstp)
  }
  flush_signals(current);
  cancel_delayed_work_sync(&ln->grace_period_end);
- locks_end_grace(&lockd_manager);
+ locks_end_grace(&ln->lockd_manager);
```

```
 if (nlmsvc_ops)
  nlmsvc_invalidate_all();
 nlm_shutdown_hosts();
```

Subject: [PATCH v2 09/15] NFSd: make nfsd4_manager allocated per network namespace context.
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:56:58 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfsd/netns.h     |    2 ++
 fs/nfsd/nfs4state.c |   32 ++++++++++++++++++++++----------
 2 files changed, 21 insertions(+), 13 deletions(-)

diff --git a/fs/nfsd/netns.h b/fs/nfsd/netns.h
index 3936563..e99767d 100644
--- a/fs/nfsd/netns.h
+++ b/fs/nfsd/netns.h
@@ -34,6 +34,8 @@ struct nfsd_net {

  struct cache_detail *idtoname_cache;
  struct cache_detail *nametoid_cache;
+
+ struct lock_manager nfsd4_manager;
 };

 extern int nfsd_net_id;
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index ab0a02a..fad2408 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -45,6 +45,8 @@
 #include "vfs.h"
 #include "current_stateid.h"

+#include "netns.h"
+
 #define NFSDDBG_FACILITY                NFSDDBG_PROC

 /* Globals */
@@ -3115,22 +3117,21 @@ out:
  return status;
 }

-static struct lock_manager nfsd4_manager = {
-};
```

```
-
 static bool grace_ended;

 static void
-nfsd4_end_grace(void)
+nfsd4_end_grace(struct net *net)
 {
+ struct nfsd_net *nn = net_generic(net, nfsd_net_id);
+
  /* do nothing if grace period already ended */
  if (grace_ended)
   return;

  dprintk("NFSD: end of grace period\n");
  grace_ended = true;
- nfsd4_record_grace_done(&init_net, boot_time);
- locks_end_grace(&nfsd4_manager);
+ nfsd4_record_grace_done(net, boot_time);
+ locks_end_grace(&nn->nfsd4_manager);
  /*
   * Now that every NFSv4 client has had the chance to recover and
   * to see the (possibly new, possibly shorter) lease time, we
@@ -3153,7 +3154,7 @@ nfs4_laundromat(void)
  nfs4_lock_state();

  dprintk("NFSD: laundromat service - starting\n");
- nfsd4_end_grace();
+ nfsd4_end_grace(&init_net);
  INIT_LIST_HEAD(&reaplist);
  spin_lock(&client_lock);
  list_for_each_safe(pos, next, &client_lru) {
@@ -4687,6 +4688,8 @@ set_max_delegations(void)
 int
 nfs4_state_start(void)
 {
+ struct net *net = &init_net;
+ struct nfsd_net *nn = net_generic(net, nfsd_net_id);
  int ret;

  /*
@@ -4696,10 +4699,10 @@ nfs4_state_start(void)
   * to that instead and then do most of the rest of this on a per-net
   * basis.
   */
- get_net(&init_net);
- nfsd4_client_tracking_init(&init_net);
+ get_net(net);
+ nfsd4_client_tracking_init(net);
```

```
  boot_time = get_seconds();
- locks_start_grace(&nfsd4_manager);
+ locks_start_grace(&nn->nfsd4_manager);
  grace_ended = false;
  printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
         nfsd4_grace);
@@ -4722,8 +4725,8 @@ nfs4_state_start(void)
 out_free_laundry:
  destroy_workqueue(laundry_wq);
 out_recovery:
- nfsd4_client_tracking_exit(&init_net);
- put_net(&init_net);
+ nfsd4_client_tracking_exit(net);
+ put_net(net);
  return ret;
 }

@@ -4764,9 +4767,12 @@ __nfs4_state_shutdown(void)
 void
 nfs4_state_shutdown(void)
 {
+ struct net *net = &init_net;
+ struct nfsd_net *nn = net_generic(net, nfsd_net_id);
+
  cancel_delayed_work_sync(&laundromat_work);
  destroy_workqueue(laundry_wq);
- locks_end_grace(&nfsd4_manager);
+ locks_end_grace(&nn->nfsd4_manager);
  nfs4_lock_state();
  __nfs4_state_shutdown();
  nfs4_unlock_state();
```

Subject: [PATCH v2 10/15] SUNRPC: service request network namespace helper introduced

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:06 GMT

View Forum Message <> Reply to Message

This is a cleanup patch - makes code looks simplier.
It replaces widely used rqstp->rq_xprt->xpt_net by introduced SVC_NET(rqstp).

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

```
---
 fs/lockd/host.c           |    2 +-
 fs/nfs/callback_xdr.c     |    4 ++--
 fs/nfsd/export.c          |    4 ++--
 fs/nfsd/nfs4idmap.c       |    4 ++--
 include/linux/sunrpc/svc.h |    2 ++
```

```
 5 files changed, 9 insertions(+), 7 deletions(-)

diff --git a/fs/lockd/host.c b/fs/lockd/host.c
index 0084ab8..f9b22e5 100644
--- a/fs/lockd/host.c
+++ b/fs/lockd/host.c
@@ -331,7 +331,7 @@ 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 net *net = SVC_NET(rqstp);
  struct nlm_lookup_host_info ni = {
   .server  = 1,
   .sap  = svc_addr(rqstp),
diff --git a/fs/nfs/callback_xdr.c b/fs/nfs/callback_xdr.c
index e64b01d..742ff4f 100644
--- a/fs/nfs/callback_xdr.c
+++ b/fs/nfs/callback_xdr.c
@@ -863,7 +863,7 @@ static __be32 nfs4_callback_compound(struct svc_rqst *rqstp, void
*argp, void *r
   .drc_status = 0,
   .clp = NULL,
   .slotid = NFS4_NO_SLOT,
-  .net = rqstp->rq_xprt->xpt_net,
+  .net = SVC_NET(rqstp),
  };
  unsigned int nops = 0;

@@ -879,7 +879,7 @@ static __be32 nfs4_callback_compound(struct svc_rqst *rqstp, void
*argp, void *r
   return rpc_garbage_args;

  if (hdr_arg.minorversion == 0) {
-  cps.clp = nfs4_find_client_ident(rqstp->rq_xprt->xpt_net, hdr_arg.cb_ident);
+  cps.clp = nfs4_find_client_ident(SVC_NET(rqstp), hdr_arg.cb_ident);
   if (!cps.clp || !check_gss_callback_principal(cps.clp, rqstp))
    return rpc_drop_reply;
  }
diff --git a/fs/nfsd/export.c b/fs/nfsd/export.c
index ba23349..d77acd0 100644
--- a/fs/nfsd/export.c
+++ b/fs/nfsd/export.c
@@ -929,7 +929,7 @@ struct svc_export *
 rqst_exp_get_by_name(struct svc_rqst *rqstp, struct path *path)
 {
  struct svc_export *gssexp, *exp = ERR_PTR(-ENOENT);
- struct nfsd_net *nn = net_generic(rqstp->rq_xprt->xpt_net, nfsd_net_id);
```

```
+ struct nfsd_net *nn = net_generic(SVC_NET(rqstp), nfsd_net_id);
  struct cache_detail *cd = nn->svc_export_cache;

  if (rqstp->rq_client == NULL)
@@ -960,7 +960,7 @@ struct svc_export *
 rqst_exp_find(struct svc_rqst *rqstp, int fsid_type, u32 *fsidv)
 {
  struct svc_export *gssexp, *exp = ERR_PTR(-ENOENT);
- struct nfsd_net *nn = net_generic(rqstp->rq_xprt->xpt_net, nfsd_net_id);
+ struct nfsd_net *nn = net_generic(SVC_NET(rqstp), nfsd_net_id);
  struct cache_detail *cd = nn->svc_export_cache;

  if (rqstp->rq_client == NULL)
diff --git a/fs/nfsd/nfs4idmap.c b/fs/nfsd/nfs4idmap.c
index dae36f1..fdc91a6 100644
--- a/fs/nfsd/nfs4idmap.c
+++ b/fs/nfsd/nfs4idmap.c
@@ -546,7 +546,7 @@ idmap_name_to_id(struct svc_rqst *rqstp, int type, const char *name,
u32 namelen
   .type = type,
  };
  int ret;
- struct nfsd_net *nn = net_generic(rqstp->rq_xprt->xpt_net, nfsd_net_id);
+ struct nfsd_net *nn = net_generic(SVC_NET(rqstp), nfsd_net_id);

  if (namelen + 1 > sizeof(key.name))
   return nfserr_badowner;
@@ -571,7 +571,7 @@ idmap_id_to_name(struct svc_rqst *rqstp, int type, uid_t id, char *name)
   .type = type,
  };
  int ret;
- struct nfsd_net *nn = net_generic(rqstp->rq_xprt->xpt_net, nfsd_net_id);
+ struct nfsd_net *nn = net_generic(SVC_NET(rqstp), nfsd_net_id);

  strlcpy(key.authname, rqst_authname(rqstp), sizeof(key.authname));
  ret = idmap_lookup(rqstp, idtoname_lookup, &key, nn->idtoname_cache, &item);
diff --git a/include/linux/sunrpc/svc.h b/include/linux/sunrpc/svc.h
index 40e0a27..d83db80 100644
--- a/include/linux/sunrpc/svc.h
+++ b/include/linux/sunrpc/svc.h
@@ -278,6 +278,8 @@ struct svc_rqst {
  struct task_struct *rq_task; /* service thread */
 };

+#define SVC_NET(svc_rqst) (svc_rqst->rq_xprt->xpt_net)
+
 /*
  * Rigorous type checking on sockaddr type conversions
```

```
 */
```

Subject: [PATCH v2 11/15] LockD: manage grace list per network namespace
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:13 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/grace.c |   14 ++++++++++---
 fs/lockd/netns.h |    1 +
 fs/lockd/svc.c   |    1 +
 3 files changed, 13 insertions(+), 3 deletions(-)

diff --git a/fs/lockd/grace.c b/fs/lockd/grace.c
index 183cc1f..8dbaff7 100644
--- a/fs/lockd/grace.c
+++ b/fs/lockd/grace.c
@@ -4,8 +4,10 @@

 #include <linux/module.h>
 #include <linux/lockd/bind.h>
+#include <net/net_namespace.h>
+
+#include "netns.h"

-static LIST_HEAD(grace_list);
 static DEFINE_SPINLOCK(grace_lock);

 /**
@@ -21,8 +23,11 @@ static DEFINE_SPINLOCK(grace_lock);
  */
 void locks_start_grace(struct lock_manager *lm)
 {
+ struct net *net = &init_net;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
  spin_lock(&grace_lock);
- list_add(&lm->list, &grace_list);
+ list_add(&lm->list, &ln->grace_list);
  spin_unlock(&grace_lock);
 }
 EXPORT_SYMBOL_GPL(locks_start_grace);
@@ -54,6 +59,9 @@ EXPORT_SYMBOL_GPL(locks_end_grace);
  */
 int locks_in_grace(void)
 {
- return !list_empty(&grace_list);
```

```
+ struct net *net = &init_net;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+ return !list_empty(&ln->grace_list);
 }
 EXPORT_SYMBOL_GPL(locks_in_grace);
diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index e78650c..4eee248 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -11,6 +11,7 @@ struct lockd_net {

  struct delayed_work grace_period_end;
  struct lock_manager lockd_manager;
+ struct list_head grace_list;
 };

 extern int lockd_net_id;
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index a9c436b..834dfe2 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -596,6 +596,7 @@ static int lockd_init_net(struct net *net)
  struct lockd_net *ln = net_generic(net, lockd_net_id);

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

Subject: [PATCH v2 12/15] LockD: pass actual network namespace to grace period management functions

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:22 GMT

View Forum Message <> Reply to Message

Passed network namespace replaced hard-coded init_net

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

```
---
 fs/lockd/grace.c          |    6 ++----
 fs/lockd/svc.c            |   16 ++++++++++-------
 fs/lockd/svc4proc.c       |   13 +++++++------
 fs/lockd/svclock.c        |   16 ++++++++++++++++
 fs/lockd/svcproc.c        |   15 +++++++++++++++
 fs/nfsd/nfs4proc.c        |   18 ++++++++++++++++++
 fs/nfsd/nfs4state.c       |   29 ++++++++++++++++++
```

```
 include/linux/fs.h          |    5 +++--
 include/linux/lockd/lockd.h |    4 ++--
 10 files changed, 67 insertions(+), 58 deletions(-)

diff --git a/fs/lockd/grace.c b/fs/lockd/grace.c
index 8dbaff7..6d1ee72 100644
--- a/fs/lockd/grace.c
+++ b/fs/lockd/grace.c
@@ -21,9 +21,8 @@ static DEFINE_SPINLOCK(grace_lock);
  *
  * This function is called to start a grace period.
  */
-void locks_start_grace(struct lock_manager *lm)
+void locks_start_grace(struct net *net, struct lock_manager *lm)
 {
- struct net *net = &init_net;
  struct lockd_net *ln = net_generic(net, lockd_net_id);

  spin_lock(&grace_lock);
@@ -57,9 +56,8 @@ EXPORT_SYMBOL_GPL(locks_end_grace);
  * to answer ordinary lock requests, and when they should accept only
  * lock reclaims.
  */
-int locks_in_grace(void)
+int locks_in_grace(struct net *net)
 {
- struct net *net = &init_net;
  struct lockd_net *ln = net_generic(net, lockd_net_id);

  return !list_empty(&ln->grace_list);
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 834dfe2..68271c2 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -97,12 +97,12 @@ static void grace_ender(struct work_struct *grace)
  locks_end_grace(&ln->lockd_manager);
 }

-static void set_grace_period(void)
+static void set_grace_period(struct net *net)
 {
  unsigned long grace_period = get_lockd_grace_period();
- struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

- locks_start_grace(&ln->lockd_manager);
+ locks_start_grace(net, &ln->lockd_manager);
  cancel_delayed_work_sync(&ln->grace_period_end);
```

```
  schedule_delayed_work(&ln->grace_period_end, grace_period);
 }
@@ -110,12 +110,13 @@ static void set_grace_period(void)
 static void restart_grace(void)
 {
  if (nlmsvc_ops) {
-  struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+  struct net *net = &init_net;
+  struct lockd_net *ln = net_generic(net, lockd_net_id);

   cancel_delayed_work_sync(&ln->grace_period_end);
   locks_end_grace(&ln->lockd_manager);
   nlmsvc_invalidate_all();
-  set_grace_period();
+  set_grace_period(net);
  }
 }

@@ -127,7 +128,8 @@ lockd(void *vrqstp)
 {
  int  err = 0, preverr = 0;
  struct svc_rqst *rqstp = vrqstp;
- struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+ struct net *net = &init_net;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  /* try_to_freeze() is called from svc_recv() */
  set_freezable();
@@ -141,7 +143,7 @@ lockd(void *vrqstp)
   nlm_timeout = LOCKD_DFLT_TIMEO;
  nlmsvc_timeout = nlm_timeout * HZ;

- set_grace_period();
+ set_grace_period(net);

  /*
   * The main request loop. We don't terminate until the last
diff --git a/fs/lockd/svc4proc.c b/fs/lockd/svc4proc.c
index 9a41fdc..4a43d25 100644
--- a/fs/lockd/svc4proc.c
+++ b/fs/lockd/svc4proc.c
@@ -11,6 +11,7 @@
 #include <linux/time.h>
 #include <linux/lockd/lockd.h>
 #include <linux/lockd/share.h>
+#include <linux/sunrpc/svc_xprt.h>

 #define NLMDBG_FACILITY  NLMDBG_CLIENT
```

```
@@ -151,7 +152,7 @@ nlm4svc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;

  /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
   resp->status = nlm_lck_denied_grace_period;
   return rpc_success;
  }
@@ -161,7 +162,7 @@ nlm4svc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
   return resp->status == nlm_drop_reply ? rpc_drop_reply :rpc_success;

  /* Try to cancel request. */
- resp->status = nlmsvc_cancel_blocked(file, &argp->lock);
+ resp->status = nlmsvc_cancel_blocked(SVC_NET(rqstp), file, &argp->lock);

  dprintk("lockd: CANCEL        status %d\n", ntohl(resp->status));
  nlmsvc_release_host(host);
@@ -184,7 +185,7 @@ nlm4svc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;

  /* Don't accept new lock requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
   resp->status = nlm_lck_denied_grace_period;
   return rpc_success;
  }
@@ -194,7 +195,7 @@ nlm4svc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
   return resp->status == nlm_drop_reply ? rpc_drop_reply :rpc_success;

  /* Now try to remove the lock */
- resp->status = nlmsvc_unlock(file, &argp->lock);
+ resp->status = nlmsvc_unlock(SVC_NET(rqstp), file, &argp->lock);

  dprintk("lockd: UNLOCK        status %d\n", ntohl(resp->status));
  nlmsvc_release_host(host);
@@ -321,7 +322,7 @@ nlm4svc_proc_share(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;

  /* Don't accept new lock requests during grace period */
- if (locks_in_grace() && !argp->reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !argp->reclaim) {
   resp->status = nlm_lck_denied_grace_period;
   return rpc_success;
  }
@@ -354,7 +355,7 @@ nlm4svc_proc_unshare(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;
```

```
  /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
  }
diff --git a/fs/lockd/svclock.c b/fs/lockd/svclock.c
index e46353f..afe4488 100644
--- a/fs/lockd/svclock.c
+++ b/fs/lockd/svclock.c
@@ -26,7 +26,7 @@
 #include <linux/kernel.h>
 #include <linux/sched.h>
 #include <linux/sunrpc/clnt.h>
-#include <linux/sunrpc/svc.h>
+#include <linux/sunrpc/svc_xprt.h>
 #include <linux/lockd/nlm.h>
 #include <linux/lockd/lockd.h>
 #include <linux/kthread.h>
@@ -447,11 +447,11 @@ nlmsvc_lock(struct svc_rqst *rqstp, struct nlm_file *file,
    goto out;
  }

- if (locks_in_grace() && !reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !reclaim) {
    ret = nlm_lck_denied_grace_period;
    goto out;
  }
- if (reclaim && !locks_in_grace()) {
+ if (reclaim && !locks_in_grace(SVC_NET(rqstp))) {
    ret = nlm_lck_denied_grace_period;
    goto out;
  }
@@ -559,7 +559,7 @@ nlmsvc_testlock(struct svc_rqst *rqstp, struct nlm_file *file,
    goto out;
  }

- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    ret = nlm_lck_denied_grace_period;
    goto out;
  }
@@ -603,7 +603,7 @@ out:
  * must be removed.
  */
 __be32
-nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
```

```
+nlmsvc_unlock(struct net *net, struct nlm_file *file, struct nlm_lock *lock)
 {
  int error;

@@ -615,7 +615,7 @@ nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
     (long long)lock->fl.fl_end);

  /* First, cancel any lock that might be there */
- nlmsvc_cancel_blocked(file, lock);
+ nlmsvc_cancel_blocked(net, file, lock);

  lock->fl.fl_type = F_UNLCK;
  error = vfs_lock_file(file->f_file, F_SETLK, &lock->fl, NULL);
@@ -631,7 +631,7 @@ nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
  * The calling procedure must check whether the file can be closed.
  */
 __be32
-nlmsvc_cancel_blocked(struct nlm_file *file, struct nlm_lock *lock)
+nlmsvc_cancel_blocked(struct net *net, struct nlm_file *file, struct nlm_lock *lock)
 {
  struct nlm_block *block;
  int status = 0;
@@ -643,7 +643,7 @@ nlmsvc_cancel_blocked(struct nlm_file *file, struct nlm_lock *lock)
     (long long)lock->fl.fl_start,
     (long long)lock->fl.fl_end);

- if (locks_in_grace())
+ if (locks_in_grace(net))
   return nlm_lck_denied_grace_period;

  mutex_lock(&file->f_mutex);
diff --git a/fs/lockd/svcproc.c b/fs/lockd/svcproc.c
index d27aab1..de8f2ca 100644
--- a/fs/lockd/svcproc.c
+++ b/fs/lockd/svcproc.c
@@ -11,6 +11,7 @@
 #include <linux/time.h>
 #include <linux/lockd/lockd.h>
 #include <linux/lockd/share.h>
+#include <linux/sunrpc/svc_xprt.h>

 #define NLMDBG_FACILITY  NLMDBG_CLIENT

@@ -175,13 +176,14 @@ nlmsvc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
 {
  struct nlm_host *host;
  struct nlm_file *file;
+ struct net *net = SVC_NET(rqstp);
```

```
  dprintk("lockd: CANCEL        called\n");

  resp->cookie = argp->cookie;

  /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(net)) {
   resp->status = nlm_lck_denied_grace_period;
   return rpc_success;
  }
@@ -191,7 +193,7 @@ nlmsvc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
   return resp->status == nlm_drop_reply ? rpc_drop_reply :rpc_success;

  /* Try to cancel request. */
- resp->status = cast_status(nlmsvc_cancel_blocked(file, &argp->lock));
+ resp->status = cast_status(nlmsvc_cancel_blocked(net, file, &argp->lock));

  dprintk("lockd: CANCEL        status %d\n", ntohl(resp->status));
  nlmsvc_release_host(host);
@@ -208,13 +210,14 @@ nlmsvc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
 {
  struct nlm_host *host;
  struct nlm_file *file;
+ struct net *net = SVC_NET(rqstp);

  dprintk("lockd: UNLOCK        called\n");

  resp->cookie = argp->cookie;

  /* Don't accept new lock requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(net)) {
   resp->status = nlm_lck_denied_grace_period;
   return rpc_success;
  }
@@ -224,7 +227,7 @@ nlmsvc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
   return resp->status == nlm_drop_reply ? rpc_drop_reply :rpc_success;

  /* Now try to remove the lock */
- resp->status = cast_status(nlmsvc_unlock(file, &argp->lock));
+ resp->status = cast_status(nlmsvc_unlock(net, file, &argp->lock));

  dprintk("lockd: UNLOCK        status %d\n", ntohl(resp->status));
  nlmsvc_release_host(host);
@@ -361,7 +364,7 @@ nlmsvc_proc_share(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;
```

```
  /* Don't accept new lock requests during grace period */
- if (locks_in_grace() && !argp->reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !argp->reclaim) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
  }
@@ -394,7 +397,7 @@ nlmsvc_proc_unshare(struct svc_rqst *rqstp, struct nlm_args *argp,
  resp->cookie = argp->cookie;

  /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
  }
diff --git a/fs/nfsd/nfs4proc.c b/fs/nfsd/nfs4proc.c
index 987e719..c9c1c0a 100644
--- a/fs/nfsd/nfs4proc.c
+++ b/fs/nfsd/nfs4proc.c
@@ -354,10 +354,10 @@ nfsd4_open(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  /* Openowner is now set, so sequence id will get bumped.  Now we need
   * these checks before we do any creates: */
  status = nfserr_grace;
- if (locks_in_grace() && open->op_claim_type != NFS4_OPEN_CLAIM_PREVIOUS)
+ if (locks_in_grace(SVC_NET(rqstp)) && open->op_claim_type !=
NFS4_OPEN_CLAIM_PREVIOUS)
    goto out;
  status = nfserr_no_grace;
- if (!locks_in_grace() && open->op_claim_type == NFS4_OPEN_CLAIM_PREVIOUS)
+ if (!locks_in_grace(SVC_NET(rqstp)) && open->op_claim_type ==
NFS4_OPEN_CLAIM_PREVIOUS)
    goto out;

  switch (open->op_claim_type) {
@@ -686,7 +686,8 @@ nfsd4_read(struct svc_rqst *rqstp, struct nfsd4_compound_state *cstate,

  nfs4_lock_state();
  /* check stateid */
- if ((status = nfs4_preprocess_stateid_op(cstate, &read->rd_stateid,
+ if ((status = nfs4_preprocess_stateid_op(SVC_NET(rqstp),
+       cstate, &read->rd_stateid,
        RD_STATE, &read->rd_filp))) {
    dprintk("NFSD: nfsd4_read: couldn't process stateid!\n");
    goto out;
@@ -741,7 +742,7 @@ nfsd4_remove(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
 {
```

```
  __be32 status;

- if (locks_in_grace())
+ if (locks_in_grace(SVC_NET(rqstp)))
   return nfserr_grace;
  status = nfsd_unlink(rqstp, &cstate->current_fh, 0,
         remove->rm_name, remove->rm_namelen);
@@ -760,8 +761,8 @@ nfsd4_rename(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,

  if (!cstate->save_fh.fh_dentry)
   return status;
- if (locks_in_grace() && !(cstate->save_fh.fh_export->ex_flags
-     & NFSEXP_NOSUBTREECHECK))
+ if (locks_in_grace(SVC_NET(rqstp)) &&
+  !(cstate->save_fh.fh_export->ex_flags & NFSEXP_NOSUBTREECHECK))
   return nfserr_grace;
  status = nfsd_rename(rqstp, &cstate->save_fh, rename->rn_sname,
         rename->rn_snamelen, &cstate->current_fh,
@@ -845,7 +846,7 @@ nfsd4_setattr(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,

  if (setattr->sa_iattr.ia_valid & ATTR_SIZE) {
   nfs4_lock_state();
-  status = nfs4_preprocess_stateid_op(cstate,
+  status = nfs4_preprocess_stateid_op(SVC_NET(rqstp), cstate,
     &setattr->sa_stateid, WR_STATE, NULL);
   nfs4_unlock_state();
   if (status) {
@@ -890,7 +891,8 @@ nfsd4_write(struct svc_rqst *rqstp, struct nfsd4_compound_state *cstate,
   return nfserr_inval;

  nfs4_lock_state();
- status = nfs4_preprocess_stateid_op(cstate, stateid, WR_STATE, &filp);
+ status = nfs4_preprocess_stateid_op(SVC_NET(rqstp),
+     cstate, stateid, WR_STATE, &filp);
  if (filp)
   get_file(filp);
  nfs4_unlock_state();
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index fad2408..b926e00 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -2884,7 +2884,8 @@ static void nfsd4_open_deleg_none_ext(struct nfsd4_open *open, int
status)
  * Attempt to hand out a delegation.
  */
 static void
```

```
-nfs4_open_delegation(struct svc_fh *fh, struct nfsd4_open *open, struct nfs4_ol_stateid *stp)
+nfs4_open_delegation(struct net *net, struct svc_fh *fh,
+       struct nfsd4_open *open, struct nfs4_ol_stateid *stp)
 {
  struct nfs4_delegation *dp;
  struct nfs4_openowner *oo = container_of(stp->st_stateowner, struct nfs4_openowner,
oo_owner);
@@ -2905,7 +2906,7 @@ nfs4_open_delegation(struct svc_fh *fh, struct nfsd4_open *open,
struct nfs4_ol_
   case NFS4_OPEN_CLAIM_NULL:
    /* Let's not give out any delegations till everyone's
     * had the chance to reclaim theirs.... */
-   if (locks_in_grace())
+   if (locks_in_grace(net))
     goto out;
    if (!cb_up || !(oo->oo_flags & NFS4_OO_CONFIRMED))
     goto out;
@@ -3039,7 +3040,7 @@ nfsd4_process_open2(struct svc_rqst *rqstp, struct svc_fh *current_fh,
struct nf
  * Attempt to hand out a delegation. No error return, because the
  * OPEN succeeds even if we fail.
  */
- nfs4_open_delegation(current_fh, open, stp);
+ nfs4_open_delegation(SVC_NET(rqstp), current_fh, open, stp);
 nodeleg:
  status = nfs_ok;

@@ -3278,11 +3279,11 @@ out:
 }

 static inline __be32
-check_special_stateids(svc_fh *current_fh, stateid_t *stateid, int flags)
+check_special_stateids(struct net *net, svc_fh *current_fh, stateid_t *stateid, int flags)
 {
  if (ONE_STATEID(stateid) && (flags & RD_STATE))
   return nfs_ok;
- else if (locks_in_grace()) {
+ else if (locks_in_grace(net)) {
   /* Answer in remaining cases depends on existence of
    * conflicting state; so we must wait out the grace period. */
   return nfserr_grace;
@@ -3299,9 +3300,9 @@ check_special_stateids(svc_fh *current_fh, stateid_t *stateid, int flags)
  * that are not able to provide mandatory locking.
  */
 static inline int
-grace_disallows_io(struct inode *inode)
+grace_disallows_io(struct net *net, struct inode *inode)
 {
```

```
- return locks_in_grace() && mandatory_lock(inode);
+ return locks_in_grace(net) && mandatory_lock(inode);
 }

 /* Returns true iff a is later than b: */
@@ -3392,7 +3393,7 @@ static __be32 nfsd4_lookup_stateid(stateid_t *stateid, unsigned char
typemask, s
 * Checks for stateid operations
 */
 __be32
-nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
+nfs4_preprocess_stateid_op(struct net *net, struct nfsd4_compound_state *cstate,
       stateid_t *stateid, int flags, struct file **filpp)
 {
  struct nfs4_stid *s;
@@ -3405,11 +3406,11 @@ nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
  if (filpp)
   *filpp = NULL;

- if (grace_disallows_io(ino))
+ if (grace_disallows_io(net, ino))
   return nfserr_grace;

  if (ZERO_STATEID(stateid) || ONE_STATEID(stateid))
-  return check_special_stateids(current_fh, stateid, flags);
+  return check_special_stateids(net, current_fh, stateid, flags);

  status = nfsd4_lookup_stateid(stateid,
NFS4_DELEG_STID|NFS4_OPEN_STID|NFS4_LOCK_STID, &s);
  if (status)
@@ -4106,10 +4107,10 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
   goto out;

  status = nfserr_grace;
- if (locks_in_grace() && !lock->lk_reclaim)
+ if (locks_in_grace(SVC_NET(rqstp)) && !lock->lk_reclaim)
   goto out;
  status = nfserr_no_grace;
- if (!locks_in_grace() && lock->lk_reclaim)
+ if (!locks_in_grace(SVC_NET(rqstp)) && lock->lk_reclaim)
   goto out;

  locks_init_lock(&file_lock);
@@ -4209,7 +4210,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  struct nfs4_lockowner *lo;
  __be32 status;
```

```
- if (locks_in_grace())
+ if (locks_in_grace(SVC_NET(rqstp)))
   return nfserr_grace;

  if (check_lock_length(lockt->lt_offset, lockt->lt_length))
@@ -4702,7 +4703,7 @@ nfs4_state_start(void)
  get_net(net);
  nfsd4_client_tracking_init(net);
  boot_time = get_seconds();
- locks_start_grace(&nn->nfsd4_manager);
+ locks_start_grace(net, &nn->nfsd4_manager);
  grace_ended = false;
  printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
         nfsd4_grace);
diff --git a/fs/nfsd/state.h b/fs/nfsd/state.h
index 495df4e..981ef10 100644
--- a/fs/nfsd/state.h
+++ b/fs/nfsd/state.h
@@ -451,7 +451,8 @@ static inline struct nfs4_ol_stateid *openlockstateid(struct nfs4_stid *s)

 struct nfsd4_compound_state;

-extern __be32 nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
+extern __be32 nfs4_preprocess_stateid_op(struct net *net,
+  struct nfsd4_compound_state *cstate,
   stateid_t *stateid, int flags, struct file **filp);
 extern void nfs4_lock_state(void);
 extern void nfs4_unlock_state(void);
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 17fd887..a1e7727 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -1163,9 +1163,10 @@ struct lock_manager {
  struct list_head list;
 };

-void locks_start_grace(struct lock_manager *);
+struct net;
+void locks_start_grace(struct net *, struct lock_manager *);
 void locks_end_grace(struct lock_manager *);
-int locks_in_grace(void);
+int locks_in_grace(struct net *);

 /* that will die - we need it for nfs_lock_info */
 #include <linux/nfs_fs_i.h>
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index 50e31a2..f5a051a 100644
```

```
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -262,11 +262,11 @@ typedef int   (*nlm_host_match_fn_t)(void *cur, struct nlm_host *ref);
 __be32    nlmsvc_lock(struct svc_rqst *, struct nlm_file *,
         struct nlm_host *, struct nlm_lock *, int,
         struct nlm_cookie *, int);
-__be32    nlmsvc_unlock(struct nlm_file *, struct nlm_lock *);
+__be32    nlmsvc_unlock(struct net *net, struct nlm_file *, struct nlm_lock *);
 __be32    nlmsvc_testlock(struct svc_rqst *, struct nlm_file *,
    struct nlm_host *, struct nlm_lock *,
    struct nlm_lock *, struct nlm_cookie *);
-__be32    nlmsvc_cancel_blocked(struct nlm_file *, struct nlm_lock *);
+__be32    nlmsvc_cancel_blocked(struct net *net, struct nlm_file *, struct nlm_lock *);
 unsigned long   nlmsvc_retry_blocked(void);
 void    nlmsvc_traverse_blocks(struct nlm_host *, struct nlm_file *,
     nlm_host_match_fn_t match);
```

Subject: [PATCH v2 13/15] Lockd: move grace period management from lockd() to per-net functions

Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:29 GMT

View Forum Message <> Reply to Message

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

```
---
 fs/lockd/svc.c |    9 +++------
 1 files changed, 3 insertions(+), 6 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 68271c2..31a63f8 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -128,8 +128,6 @@ lockd(void *vrqstp)
 {
  int  err = 0, preverr = 0;
  struct svc_rqst *rqstp = vrqstp;
- struct net *net = &init_net;
- struct lockd_net *ln = net_generic(net, lockd_net_id);

  /* try_to_freeze() is called from svc_recv() */
  set_freezable();
@@ -143,8 +141,6 @@ lockd(void *vrqstp)
   nlm_timeout = LOCKD_DFLT_TIMEO;
  nlmsvc_timeout = nlm_timeout * HZ;

- set_grace_period(net);
-
  /*
```

```
   * The main request loop. We don't terminate until the last
   * NFS mount or NFS daemon has gone away.
@@ -190,8 +186,6 @@ lockd(void *vrqstp)
    svc_process(rqstp);
   }
   flush_signals(current);
- cancel_delayed_work_sync(&ln->grace_period_end);
- locks_end_grace(&ln->lockd_manager);
   if (nlmsvc_ops)
    nlmsvc_invalidate_all();
   nlm_shutdown_hosts();
@@ -272,6 +266,7 @@ static int lockd_up_net(struct svc_serv *serv, struct net *net)
   error = make_socks(serv, net);
   if (error < 0)
    goto err_socks;
+ set_grace_period(net);
   dprintk("lockd_up_net: per-net data created; net=%p\n", net);
   return 0;

@@ -289,6 +284,8 @@ static void lockd_down_net(struct svc_serv *serv, struct net *net)
   if (ln->nlmsvc_users) {
    if (--ln->nlmsvc_users == 0) {
     nlm_shutdown_hosts_net(net);
+    cancel_delayed_work_sync(&ln->grace_period_end);
+    locks_end_grace(&ln->lockd_manager);
     svc_shutdown_net(serv, net);
     dprintk("lockd_down_net: per-net data destroyed; net=%p\n", net);
    }
```

Subject: [PATCH v2 14/15] NFSd: make grace end flag per network namespace
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:37 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfsd/netns.h     |    1 +
 fs/nfsd/nfs4state.c |    8 +++-----
 2 files changed, 4 insertions(+), 5 deletions(-)

diff --git a/fs/nfsd/netns.h b/fs/nfsd/netns.h
index e99767d..b6deebd 100644
--- a/fs/nfsd/netns.h
+++ b/fs/nfsd/netns.h
@@ -36,6 +36,7 @@ struct nfsd_net {
   struct cache_detail *nametoid_cache;

   struct lock_manager nfsd4_manager;
```

```
+ bool grace_ended;
 };

 extern int nfsd_net_id;
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index b926e00..2c53447 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -3118,19 +3118,17 @@ out:
  return status;
 }

-static bool grace_ended;
-
 static void
 nfsd4_end_grace(struct net *net)
 {
  struct nfsd_net *nn = net_generic(net, nfsd_net_id);

  /* do nothing if grace period already ended */
- if (grace_ended)
+ if (nn->grace_ended)
   return;

  dprintk("NFSD: end of grace period\n");
- grace_ended = true;
+ nn->grace_ended = true;
  nfsd4_record_grace_done(net, boot_time);
  locks_end_grace(&nn->nfsd4_manager);
  /*
@@ -4704,7 +4702,7 @@ nfs4_state_start(void)
  nfsd4_client_tracking_init(net);
  boot_time = get_seconds();
  locks_start_grace(net, &nn->nfsd4_manager);
- grace_ended = false;
+ nn->grace_ended = false;
  printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
        nfsd4_grace);
  ret = set_callback_cred();
```

Subject: [PATCH v2 15/15] NFSd: make boot_time variable per network namespace
Posted by Stanislav Kinsbursky on Wed, 25 Jul 2012 12:57:45 GMT
View Forum Message <> Reply to Message

NFSd's boot_time represents grace period start point in time.

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

```
---
 fs/nfsd/netns.h     |    1 +
 fs/nfsd/nfs4state.c |   39 ++++++++++++++++++++++++++----------
 fs/nfsd/state.h     |    1 +
 3 files changed, 25 insertions(+), 16 deletions(-)

diff --git a/fs/nfsd/netns.h b/fs/nfsd/netns.h
index b6deebd..65c2431 100644
--- a/fs/nfsd/netns.h
+++ b/fs/nfsd/netns.h
@@ -37,6 +37,7 @@ struct nfsd_net {

   struct lock_manager nfsd4_manager;
   bool grace_ended;
+ time_t boot_time;
 };

 extern int nfsd_net_id;
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index 2c53447..d683f32 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -52,7 +52,6 @@
 /* Globals */
 time_t nfsd4_lease = 90;     /* default lease time */
 time_t nfsd4_grace = 90;
-static time_t boot_time;

 #define all_ones {{~0,~0},~0}
 static const stateid_t one_stateid = {
@@ -1055,12 +1054,12 @@ renew_client(struct nfs4_client *clp)

 /* SETCLIENTID and SETCLIENTID_CONFIRM Helper functions */
 static int
-STALE_CLIENTID(clientid_t *clid)
+STALE_CLIENTID(clientid_t *clid, struct nfsd_net *nn)
 {
- if (clid->cl_boot == boot_time)
+ if (clid->cl_boot == nn->boot_time)
   return 0;
  dprintk("NFSD stale clientid (%08x/%08x) boot_time %08lx\n",
-  clid->cl_boot, clid->cl_id, boot_time);
+  clid->cl_boot, clid->cl_id, nn->boot_time);
  return 1;
 }

@@ -1241,8 +1240,9 @@ same_creds(struct svc_cred *cr1, struct svc_cred *cr2)
 static void gen_clid(struct nfs4_client *clp)
```

```
 {
  static u32 current_clientid = 1;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- clp->cl_clientid.cl_boot = boot_time;
+ clp->cl_clientid.cl_boot = nn->boot_time;
  clp->cl_clientid.cl_id = current_clientid++;
 }

@@ -2225,8 +2225,9 @@ nfsd4_setclientid_confirm(struct svc_rqst *rqstp,
  nfs4_verifier confirm = setclientid_confirm->sc_confirm;
  clientid_t * clid = &setclientid_confirm->sc_clientid;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
   return nfserr_stale_clientid;
  nfs4_lock_state();

@@ -2585,8 +2586,9 @@ nfsd4_process_open1(struct nfsd4_compound_state *cstate,
  unsigned int strhashval;
  struct nfs4_openowner *oo = NULL;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

- if (STALE_CLIENTID(&open->op_clientid))
+ if (STALE_CLIENTID(&open->op_clientid, nn))
   return nfserr_stale_clientid;
  /*
   * In case we need it later, after we've already created the
@@ -3094,12 +3096,13 @@ nfsd4_renew(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
 {
  struct nfs4_client *clp;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  nfs4_lock_state();
  dprintk("process_renew(%08x/%08x): starting\n",
    clid->cl_boot, clid->cl_id);
  status = nfserr_stale_clientid;
- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
   goto out;
  clp = find_confirmed_client(clid);
  status = nfserr_expired;
@@ -3129,7 +3132,7 @@ nfsd4_end_grace(struct net *net)
```

```
  dprintk("NFSD: end of grace period\n");
  nn->grace_ended = true;
- nfsd4_record_grace_done(net, boot_time);
+ nfsd4_record_grace_done(net, nn->boot_time);
  locks_end_grace(&nn->nfsd4_manager);
  /*
   * Now that every NFSv4 client has had the chance to recover and
@@ -3235,9 +3238,9 @@ static inline __be32 nfs4_check_fh(struct svc_fh *fhp, struct
nfs4_ol_stateid *s
 }

 static int
-STALE_STATEID(stateid_t *stateid)
+STALE_STATEID(stateid_t *stateid, struct nfsd_net *nn)
 {
- if (stateid->si_opaque.so_clid.cl_boot == boot_time)
+ if (stateid->si_opaque.so_clid.cl_boot == nn->boot_time)
   return 0;
  dprintk("NFSD: stale stateid " STATEID_FMT "!\n",
   STATEID_VAL(stateid));
@@ -3372,10 +3375,11 @@ static __be32 nfsd4_validate_stateid(struct nfs4_client *cl, stateid_t
*stateid)
 static __be32 nfsd4_lookup_stateid(stateid_t *stateid, unsigned char typemask, struct nfs4_stid
**s)
 {
  struct nfs4_client *cl;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  if (ZERO_STATEID(stateid) || ONE_STATEID(stateid))
   return nfserr_bad_stateid;
- if (STALE_STATEID(stateid))
+ if (STALE_STATEID(stateid, nn))
   return nfserr_stale_stateid;
  cl = find_confirmed_client(&stateid->si_opaque.so_clid);
  if (!cl)
@@ -4047,6 +4051,7 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  bool new_state = false;
  int lkflg;
  int err;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  dprintk("NFSD: nfsd4_lock: start=%Ld length=%Ld\n",
   (long long) lock->lk_offset,
@@ -4073,7 +4078,7 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
     sizeof(clientid_t));
```

```
   status = nfserr_stale_clientid;
-  if (STALE_CLIENTID(&lock->lk_new_clientid))
+  if (STALE_CLIENTID(&lock->lk_new_clientid, nn))
    goto out;

   /* validate and update open stateid and open seqid */
@@ -4207,6 +4212,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  struct file_lock file_lock;
  struct nfs4_lockowner *lo;
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  if (locks_in_grace(SVC_NET(rqstp)))
   return nfserr_grace;
@@ -4217,7 +4223,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
  nfs4_lock_state();

  status = nfserr_stale_clientid;
- if (!nfsd4_has_session(cstate) && STALE_CLIENTID(&lockt->lt_clientid))
+ if (!nfsd4_has_session(cstate) && STALE_CLIENTID(&lockt->lt_clientid, nn))
   goto out;

  if ((status = fh_verify(rqstp, &cstate->current_fh, S_IFREG, 0)))
@@ -4366,6 +4372,7 @@ nfsd4_release_lockowner(struct svc_rqst *rqstp,
  struct list_head matches;
  unsigned int hashval = ownerstr_hashval(clid->cl_id, owner);
  __be32 status;
+ struct nfsd_net *nn = net_generic(&init_net, nfsd_net_id);

  dprintk("nfsd4_release_lockowner clientid: (%08x/%08x):\n",
   clid->cl_boot, clid->cl_id);
@@ -4373,7 +4380,7 @@ nfsd4_release_lockowner(struct svc_rqst *rqstp,
  /* XXX check for lease expiration */

  status = nfserr_stale_clientid;
- if (STALE_CLIENTID(clid))
+ if (STALE_CLIENTID(clid, nn))
   return status;

  nfs4_lock_state();
@@ -4700,7 +4707,7 @@ nfs4_state_start(void)
   */
  get_net(net);
  nfsd4_client_tracking_init(net);
- boot_time = get_seconds();
```

```
+ nn->boot_time = get_seconds();
  locks_start_grace(net, &nn->nfsd4_manager);
  nn->grace_ended = false;
  printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
diff --git a/fs/nfsd/state.h b/fs/nfsd/state.h
index 981ef10..e617314 100644
--- a/fs/nfsd/state.h
+++ b/fs/nfsd/state.h
@@ -450,6 +450,7 @@ static inline struct nfs4_ol_stateid *openlockstateid(struct nfs4_stid *s)
 #define WR_STATE         0x00000020

 struct nfsd4_compound_state;
+struct nfsd_net;

 extern __be32 nfs4_preprocess_stateid_op(struct net *net,
   struct nfsd4_compound_state *cstate,
```

Subject: Re: [PATCH v2 09/15] NFSd: make nfsd4_manager allocated per network namespace context.

Posted by bfields on Fri, 27 Jul 2012 21:46:25 GMT

View Forum Message <> Reply to Message

On Wed, Jul 25, 2012 at 04:56:58PM +0400, Stanislav Kinsbursky wrote:

```
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> ---
>  fs/nfsd/netns.h     |    2 ++
>  fs/nfsd/nfs4state.c |   32 ++++++++++++++++++++++----------
>  2 files changed, 21 insertions(+), 13 deletions(-)
>
> diff --git a/fs/nfsd/netns.h b/fs/nfsd/netns.h
> index 3936563..e99767d 100644
> --- a/fs/nfsd/netns.h
> +++ b/fs/nfsd/netns.h
> @@ -34,6 +34,8 @@ struct nfsd_net {
>
>   struct cache_detail *idtoname_cache;
>   struct cache_detail *nametoid_cache;
> +
> + struct lock_manager nfsd4_manager;
>  };
>
>  extern int nfsd_net_id;
> diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
> index ab0a02a..fad2408 100644
> --- a/fs/nfsd/nfs4state.c
> +++ b/fs/nfsd/nfs4state.c
> @@ -45,6 +45,8 @@
```

> #include "vfs.h"
> #include "current_stateid.h"
>
> +#include "netns.h"
> +
> #define NFSDDBG_FACILITY		NFSDDBG_PROC
>
> /* Globals */
> @@ -3115,22 +3117,21 @@ out:
>   return status;
> }
>
> -static struct lock_manager nfsd4_manager = {
> -};
> -
> static bool grace_ended;
>
> static void
> -nfsd4_end_grace(void)
> +nfsd4_end_grace(struct net *net)
> {
> + struct nfsd_net *nn = net_generic(net, nfsd_net_id);
> +
>   /* do nothing if grace period already ended */
>   if (grace_ended)
>   return;

That doesn't look right....

But, OK, looking through later patches I see you make grace_ended
per-namespace later. OK.

--b.

>
>   dprintk("NFSD: end of grace period\n");
>   grace_ended = true;
> - nfsd4_record_grace_done(&init_net, boot_time);
> - locks_end_grace(&nfsd4_manager);
> + nfsd4_record_grace_done(net, boot_time);
> + locks_end_grace(&nn->nfsd4_manager);
>   /*
>   * Now that every NFSv4 client has had the chance to recover and
>   * to see the (possibly new, possibly shorter) lease time, we
> @@ -3153,7 +3154,7 @@ nfs4_laundromat(void)
>   nfs4_lock_state();
>
>   dprintk("NFSD: laundromat service - starting\n");

```
> - nfsd4_end_grace();
> + nfsd4_end_grace(&init_net);
>   INIT_LIST_HEAD(&reaplist);
>   spin_lock(&client_lock);
>   list_for_each_safe(pos, next, &client_lru) {
> @@ -4687,6 +4688,8 @@ set_max_delegations(void)
>  int
>  nfs4_state_start(void)
>  {
> + struct net *net = &init_net;
> + struct nfsd_net *nn = net_generic(net, nfsd_net_id);
>   int ret;
>
>   /*
> @@ -4696,10 +4699,10 @@ nfs4_state_start(void)
>    * to that instead and then do most of the rest of this on a per-net
>    * basis.
>    */
> - get_net(&init_net);
> - nfsd4_client_tracking_init(&init_net);
> + get_net(net);
> + nfsd4_client_tracking_init(net);
>   boot_time = get_seconds();
> - locks_start_grace(&nfsd4_manager);
> + locks_start_grace(&nn->nfsd4_manager);
>   grace_ended = false;
>   printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
>          nfsd4_grace);
> @@ -4722,8 +4725,8 @@ nfs4_state_start(void)
>  out_free_laundry:
>   destroy_workqueue(laundry_wq);
>  out_recovery:
> - nfsd4_client_tracking_exit(&init_net);
> - put_net(&init_net);
> + nfsd4_client_tracking_exit(net);
> + put_net(net);
>   return ret;
>  }
>
> @@ -4764,9 +4767,12 @@ __nfs4_state_shutdown(void)
>  void
>  nfs4_state_shutdown(void)
>  {
> + struct net *net = &init_net;
> + struct nfsd_net *nn = net_generic(net, nfsd_net_id);
> +
>   cancel_delayed_work_sync(&laundromat_work);
>   destroy_workqueue(laundry_wq);
```

> - locks_end_grace(&nfsd4_manager);
> + locks_end_grace(&nn->nfsd4_manager);
> nfs4_lock_state();
> __nfs4_state_shutdown();
> nfs4_unlock_state();
>

Subject: Re: [PATCH v2 00/15] Lockd: grace period containerization
Posted by bfields on Fri, 27 Jul 2012 21:54:31 GMT
View Forum Message <> Reply to Message

On Wed, Jul 25, 2012 at 04:55:45PM +0400, Stanislav Kinsbursky wrote:
> Bruce, I feel this patch set is ready for inclusion.
>
> v2:
> 1) Rebase on Bruce's "for-3.6" branch.
>
> This patch set makes grace period and hosts reclaiming network namespace
> aware.

On a quick skim--yes, that looks reasonable to me.

It doesn't help with active/active cluster exports, because in that case we need some additional coordination between nfsd's.

But it looks good enough to handle the case where each filesystem is exported from at most one server at a time, which is more than we currently handle.

It's a little late for 3.6. Also I get the impression Al Viro has some lockd rework in progress, which we may want to wait for.

So I'll likely look again into queueing this up for 3.7 once 3.6-rc1 is out.

--b.

>
> Main ideas:
> 1) moving of
>
> unsigned long next_gc;
> unsigned long nrhosts;
>
> struct delayed_work grace_period_end;
> struct lock_manager lockd_manager;
> struct list_head grace_list;

>
> to per-net Lockd data.
>
> 2) moving of
>
> struct lock_manager nfsd4_manager;
>
> to per-net NFSd data.
>
> 3) shutdown + gc of NLM hosts done now network namespace aware.
>
> 4) restart_grace() now works only for init_net.
>
> The following series implements...
>
> ---
>
> Stanislav Kinsbursky (15):
>   LockD: mark host per network namespace on garbage collect
>   LockD: make garbage collector network namespace aware.
>   LockD: manage garbage collection timeout per networks namespace
>   LockD: manage used host count per networks namespace
>   Lockd: host complaining function introduced
>   Lockd: add more debug to host shutdown functions
>   LockD: manage grace period per network namespace
>   LockD: make lockd manager allocated per network namespace
>   NFSd: make nfsd4_manager allocated per network namespace context.
>   SUNRPC: service request network namespace helper introduced
>   LockD: manage grace list per network namespace
>   LockD: pass actual network namespace to grace period management functions
>   Lockd: move grace period management from lockd() to per-net functions
>   NFSd: make grace end flag per network namespace
>   NFSd: make boot_time variable per network namespace
>
>
> fs/lockd/grace.c | 16 +++++--
> fs/lockd/host.c | 92 ++++++++++++++++++++++++++++++------------
> fs/lockd/netns.h | 7 +++
> fs/lockd/svc.c | 43 ++++++++++++--------
> fs/lockd/svc4proc.c | 13 +++--
> fs/lockd/svclock.c | 16 +++----
> fs/lockd/svcproc.c | 15 ++++--
> fs/lockd/svcsubs.c | 19 +++++---
> fs/nfs/callback_xdr.c | 4 +-
> fs/nfsd/export.c | 4 +-
> fs/nfsd/netns.h | 4 ++
> fs/nfsd/nfs4idmap.c | 4 +-
> fs/nfsd/nfs4proc.c | 18 ++++---

> fs/nfsd/nfs4state.c | 104 ++++++++++++++++++++++++++++++----------------
> fs/nfsd/state.h | 4 +-
> include/linux/fs.h | 5 +-
> include/linux/lockd/lockd.h | 6 +-
> include/linux/sunrpc/svc.h | 2 +
> 18 files changed, 231 insertions(+), 145 deletions(-)
>

Subject: Re: [PATCH v2 00/15] Lockd: grace period containerization
Posted by Stanislav Kinsbursky on Mon, 30 Jul 2012 10:03:57 GMT
View Forum Message <> Reply to Message

> On Wed, Jul 25, 2012 at 04:55:45PM +0400, Stanislav Kinsbursky wrote:
>> Bruce, I feel this patch set is ready for inclusion.
>>
>> v2:
>> 1) Rebase on Bruce's "for-3.6" branch.
>>
>> This patch set makes grace period and hosts reclaiming network namespace
>> aware.
>
> On a quick skim--yes, that looks reasonable to me.
>
> It doesn't help with active/active cluster exports, because in that case
> we need some additional coordination between nfsd's.
>
> But it looks good enough to handle the case where each filesystem is
> exported from at most one server at a time, which is more than we
> currently handle.
>
> It's a little late for 3.6. Also I get the impression Al Viro has some
> lockd rework in progress, which we may want to wait for.
>
> So I'll likely look again into queueing this up for 3.7 once 3.6-rc1 is
> out.

Ok.
Will Al Viro's lockd rework be a part of 3.6 kernel?

>
> --b.
>
>>
>> Main ideas:

>> 1) moving of
>>
>> unsigned long next_gc;
>> unsigned long nrhosts;
>>
>> struct delayed_work grace_period_end;
>> struct lock_manager lockd_manager;
>> struct list_head grace_list;
>>
>> to per-net Lockd data.
>>
>> 2) moving of
>>
>> struct lock_manager nfsd4_manager;
>>
>> to per-net NFSd data.
>>
>> 3) shutdown + gc of NLM hosts done now network namespace aware.
>>
>> 4) restart_grace() now works only for init_net.
>>
>> The following series implements...
>>
>> ---
>>
>> Stanislav Kinsbursky (15):
>>   LockD: mark host per network namespace on garbage collect
>>   LockD: make garbage collector network namespace aware.
>>   LockD: manage garbage collection timeout per networks namespace
>>   LockD: manage used host count per networks namespace
>>   Lockd: host complaining function introduced
>>   Lockd: add more debug to host shutdown functions
>>   LockD: manage grace period per network namespace
>>   LockD: make lockd manager allocated per network namespace
>>   NFSd: make nfsd4_manager allocated per network namespace context.
>>   SUNRPC: service request network namespace helper introduced
>>   LockD: manage grace list per network namespace
>>   LockD: pass actual network namespace to grace period management functions
>>   Lockd: move grace period management from lockd() to per-net functions
>>   NFSd: make grace end flag per network namespace
>>   NFSd: make boot_time variable per network namespace
>>
>>
>> fs/lockd/grace.c | 16 +++++--
>> fs/lockd/host.c | 92 ++++++++++++++++++++++++++++++++------------
>> fs/lockd/netns.h | 7 +++
>> fs/lockd/svc.c | 43 ++++++++++++--------
>> fs/lockd/svc4proc.c | 13 +++--

>> fs/lockd/svclock.c | 16 ++++----
>> fs/lockd/svcproc.c | 15 ++++++--
>> fs/lockd/svcsubs.c | 19 +++++++---
>> fs/nfs/callback_xdr.c | 4 +-
>> fs/nfsd/export.c | 4 +-
>> fs/nfsd/netns.h | 4 ++
>> fs/nfsd/nfs4idmap.c | 4 +-
>> fs/nfsd/nfs4proc.c | 18 ++++++---
>> fs/nfsd/nfs4state.c | 104 ++++++

> >
> >So I'll likely look again into queueing this up for 3.7 once 3.6-rc1 is
> >out.
>
>
> Ok.
> Will Al Viro's lockd rework be a part of 3.6 kernel?

Actually I think it mostly won't be.

And this looks pretty safe, really. I've gone ahead and merged it.

--b.

Subject: Re: [PATCH v2 00/15] Lockd: grace period containerization
Posted by Stanislav Kinsbursky on Tue, 31 Jul 2012 14:26:20 GMT
View Forum Message <> Reply to Message

> On Mon, Jul 30, 2012 at 02:03:57PM +0400, Stanislav Kinsbursky wrote:

>>> On Wed, Jul 25, 2012 at 04:55:45PM +0400, Stanislav Kinsbursky wrote:
>>>> Bruce, I feel this patch set is ready for inclusion.
>>>>
>>>> v2:
>>>> 1) Rebase on Bruce's "for-3.6" branch.
>>>>
>>>> This patch set makes grace period and hosts reclaiming network namespace
>>>> aware.
>>>
>>> On a quick skim--yes, that looks reasonable to me.
>>>
>>> It doesn't help with active/active cluster exports, because in that case
>>> we need some additional coordination between nfsd's.
>>>
>>> But it looks good enough to handle the case where each filesystem is
>>> exported from at most one server at a time, which is more than we
>>> currently handle.
>>>
>>> It's a little late for 3.6. Also I get the impression Al Viro has some
>>> lockd rework in progress, which we may want to wait for.
>>>
>>> So I'll likely look again into queueing this up for 3.7 once 3.6-rc1 is
>>> out.
>>
>>
>> Ok.

>> Will Al Viro's lockd rework be a part of 3.6 kernel?
>
> Actually I think it mostly won't be.
>
> And this looks pretty safe, really. I've gone ahead and merged it.
>

Nice to hear. Thanks, Bruce.

--
Best regards,
Stanislav Kinsbursky