Subject: [PATCH v2 0/3] lockd: use per-net refrence-counted NSM clients
Posted by Stanislav Kinsbursky on Tue, 18 Sep 2012 09:37:06 GMT
View Forum Message <> Reply to Message

v2:
1) NSM transport is TCP based now. And all RPC tasks now called with RPC_TASK_SOFTCONN. The advantage of this is that the kernel could discover when statd is not running and fail the upcall immediately, rather than waiting possibly many seconds for each upcall RPC to time out.
2) XDR layer violation (reference to upper RPC client cl_hostname) was replaced by passing the string as a part of nlm_args structure.

This is a bug fix for https://bugzilla.redhat.com/show_bug.cgi?id=830862.

The problem is that with NFSv4 mount in container (with separated mount namesapce) and active lock on it, dying child reaped of this container will try to umount NFS and doing this will try to create RPC client to send unmonitor request to statd.
But creation of RCP client requires valid current->nsproxy (for operation with utsname()) and during umount on child reaper exit it's equal to zero.

Proposed solution is to introduce refrence-counter per-net NSM client, which is created on fist monitor call and destroyed after the lst monitor call.

The following series implements...

Stanislav Kinsbursky (3):
    lockd: per-net NSM client creation and destruction helpers introduced
    lockd: use rpc client's cl_nodename for id encoding
    lockd: create and use per-net NSM RPC clients on MON/UNMON requests

```
 fs/lockd/mon.c   |   86 ++++++++++++++++++++++++++++++++++++++++++++++++-----------
 fs/lockd/netns.h |    4 +++
 fs/lockd/svc.c   |    1 +
 3 files changed, 74 insertions(+), 17 deletions(-)
```

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

NSM RPC client can be required on NFSv3 umount, when child reaper is dying (and destroying it's mount namespace). It means, that current nsproxy is set to NULL already, but creation of RPC client requires UTS namespace for gaining

hostname string.
This patch introduces reference counted NFS RPC clients creation and
destruction helpers (similar to RPCBIND RPC clients).

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

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

 #include <asm/unaligned.h>

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

+__maybe_unused static struct rpc_clnt *nsm_client_get(struct net *net)
+{
+ static DEFINE_MUTEX(nsm_create_mutex);
+ struct rpc_clnt *clnt;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
+
+ spin_lock(&ln->nsm_clnt_lock);
+ if (ln->nsm_users) {
+ ln->nsm_users++;
+ clnt = ln->nsm_clnt;

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

  msg.rpc_proc = &clnt->cl_procinfo[proc];
- status = rpc_call_sync(clnt, &msg, 0);
+ status = rpc_call_sync(clnt, &msg, RPC_TASK_SOFTCONN);
  if (status < 0)
   dprintk("lockd: NSM upcall RPC failed, status=%d\n",
     status);
```

```
diff --git a/fs/lockd/netns.h b/fs/lockd/netns.h
index 4eee248..5010b55 100644
--- a/fs/lockd/netns.h
+++ b/fs/lockd/netns.h
@@ -12,6 +12,10 @@ struct lockd_net {
  struct delayed_work grace_period_end;
  struct lock_manager lockd_manager;
  struct list_head grace_list;
+
+ spinlock_t nsm_clnt_lock;
+ unsigned int nsm_users;
+ struct rpc_clnt *nsm_clnt;
 };

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

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

Subject: [PATCH v2 2/3] lockd: use rpc client's cl_nodename for id encoding
Posted by Stanislav Kinsbursky on Tue, 18 Sep 2012 09:37:18 GMT
View Forum Message <> Reply to Message

Taking hostname from uts namespace if not safe, because this cuold be performind during umount operation on child reaper death. And in this case current->nsproxy is NULL already.

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

```
diff --git a/fs/lockd/mon.c b/fs/lockd/mon.c
index 38f240e..e0bc36e 100644
--- a/fs/lockd/mon.c
+++ b/fs/lockd/mon.c
@@ -42,6 +42,7 @@ struct nsm_args {
  u32   proc;
```

```
 char   *mon_name;
+ char   *nodename;
 };

 struct nsm_res {
@@ -141,6 +142,7 @@ static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct
nsm_res *res,
   .vers  = 3,
   .proc  = NLMPROC_NSM_NOTIFY,
   .mon_name = nsm->sm_mon_name,
+  .nodename = utsname()->nodename,
  };
  struct rpc_message msg = {
   .rpc_argp = &args,
@@ -477,7 +479,7 @@ static void encode_my_id(struct xdr_stream *xdr, const struct nsm_args
*argp)
 {
  __be32 *p;

- encode_nsm_string(xdr, utsname()->nodename);
+ encode_nsm_string(xdr, argp->nodename);
  p = xdr_reserve_space(xdr, 4 + 4 + 4);
  *p++ = cpu_to_be32(argp->prog);
  *p++ = cpu_to_be32(argp->vers);
```

Subject: [PATCH v2 3/3] lockd: create and use per-net NSM RPC clients on MON/UNMON requests
Posted by Stanislav Kinsbursky on Tue, 18 Sep 2012 09:37:23 GMT
View Forum Message <> Reply to Message

NSM RPC client can be required on NFSv3 umount, when child reaper is dying
(and destroying it's mount namespace). It means, that current nsproxy is set
to NULL already, but creation of RPC client requires UTS namespace for gaining
hostname string.

This patch creates reference-counted per-net NSM client on first monitor
request and destroys it after last unmonitor request.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Cc: <stable@vger.kernel.org>

```
---
 fs/lockd/mon.c |   37 ++++++++++++++++++++-----------------
 1 files changed, 20 insertions(+), 17 deletions(-)

diff --git a/fs/lockd/mon.c b/fs/lockd/mon.c
index e0bc36e..e4fb3ba 100644
```

```
--- a/fs/lockd/mon.c
+++ b/fs/lockd/mon.c
@@ -7,7 +7,6 @@
  */

 #include <linux/types.h>
-#include <linux/utsname.h>
 #include <linux/kernel.h>
 #include <linux/ktime.h>
 #include <linux/slab.h>
@@ -86,7 +85,7 @@ static struct rpc_clnt *nsm_create(struct net *net)
  return rpc_create(&args);
 }

-__maybe_unused static struct rpc_clnt *nsm_client_get(struct net *net)
+static struct rpc_clnt *nsm_client_get(struct net *net)
 {
  static DEFINE_MUTEX(nsm_create_mutex);
  struct rpc_clnt *clnt;
@@ -113,7 +112,7 @@ out:
  return clnt;
 }

-__maybe_unused static void nsm_client_put(struct net *net)
+static void nsm_client_put(struct net *net)
 {
  struct lockd_net *ln = net_generic(net, lockd_net_id);
  struct rpc_clnt *clnt = ln->nsm_clnt;
@@ -132,9 +131,8 @@ __maybe_unused static void nsm_client_put(struct net *net)
 }

 static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct nsm_res *res,
-    struct net *net)
+    struct rpc_clnt *clnt)
 {
- struct rpc_clnt *clnt;
  int  status;
  struct nsm_args args = {
   .priv  = &nsm->sm_priv,
@@ -142,20 +140,14 @@ static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct
nsm_res *res,
   .vers  = 3,
   .proc  = NLMPROC_NSM_NOTIFY,
   .mon_name = nsm->sm_mon_name,
-  .nodename = utsname()->nodename,
+  .nodename = clnt->cl_nodename,
  };
  struct rpc_message msg = {
```

```
   .rpc_argp = &args,
   .rpc_resp = res,
  };

- clnt = nsm_create(net);
- if (IS_ERR(clnt)) {
-  status = PTR_ERR(clnt);
-  dprintk("lockd: failed to create NSM upcall transport, "
-    "status=%d\n", status);
-  goto out;
- }
+ BUG_ON(clnt == NULL);

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

@@ -166,8 +158,6 @@ static int nsm_mon_unmon(struct nsm_handle *nsm, u32 proc, struct
nsm_res *res,
     status);
  else
   status = 0;
- rpc_shutdown_client(clnt);
- out:
  return status;
 }

@@ -187,6 +177,7 @@ int nsm_monitor(const struct nlm_host *host)
  struct nsm_handle *nsm = host->h_nsmhandle;
  struct nsm_res res;
  int  status;
+ struct rpc_clnt *clnt;

  dprintk("lockd: nsm_monitor(%s)\n", nsm->sm_name);

@@ -199,7 +190,15 @@ 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, host->net);
+ clnt = nsm_client_get(host->net);
+ if (IS_ERR(clnt)) {
+  status = PTR_ERR(clnt);
+  dprintk("lockd: failed to create NSM upcall transport, "
+    "status=%d, net=%p\n", status, host->net);
+  return status;
+ }
+
+ status = nsm_mon_unmon(nsm, NSMPROC_MON, &res, clnt);
  if (unlikely(res.status != 0))
```

```
   status = -EIO;
  if (unlikely(status < 0)) {
@@ -231,9 +230,11 @@ void nsm_unmonitor(const struct nlm_host *host)

  if (atomic_read(&nsm->sm_count) == 1
   && nsm->sm_monitored && !nsm->sm_sticky) {
+  struct lockd_net *ln = net_generic(host->net, lockd_net_id);
+
   dprintk("lockd: nsm_unmonitor(%s)\n", nsm->sm_name);

-  status = nsm_mon_unmon(nsm, NSMPROC_UNMON, &res, host->net);
+  status = nsm_mon_unmon(nsm, NSMPROC_UNMON, &res, ln->nsm_clnt);
   if (res.status != 0)
    status = -EIO;
   if (status < 0)
@@ -241,6 +242,8 @@ void nsm_unmonitor(const struct nlm_host *host)
      nsm->sm_name);
   else
    nsm->sm_monitored = 0;
+
+  nsm_client_put(host->net);
  }
 }
```