Subject: [PATCH v2 00/12] NFS: callback threads containerization
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:35:47 GMT
View Forum Message <> Reply to Message

v2: Rebased on Bruce's tree, "for-3.5" branch

This patch set depeneds on "SUNRPC: separate per-net data creation from service creation" patch set sent earlier.

The following series implements...

Stanislav Kinsbursky (11):
      NFS: pass net to nfs_callback_down()
      NFS: callback service creation function introduced
      NFS: move per-net callback thread initialization to nfs_callback_up_net()
      NFS: callback up - transport backchannel cleanup
      NFS: callback service start function introduced
      NFS: callback up - users counting cleanup
      NFS: make nfs_callback_tcpport per network context
      NFS: make nfs_callback_tcpport6 per network context
      NFS; callback per-net usage counting introduced
      NFS: add debug messages to callback down function
      NFS: get module in idmap PipeFS notifier callback

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

```
 fs/nfs/callback.c  |  288 ++++++++++++++++++++++++++++++++++++-----------------
 fs/nfs/callback.h  |    4 -
 fs/nfs/client.c    |    6 +
 fs/nfs/idmap.c     |    9 +-
 fs/nfs/netns.h     |    3 +
 fs/nfs/nfs4state.c |    7 +
 6 files changed, 212 insertions(+), 105 deletions(-)
```

Subject: [PATCH v2 02/12] NFS: callback service creation function introduced
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:02 GMT
View Forum Message <> Reply to Message

This function creates service if it's not exist, or increase usage counter of
the existent, and returns pointer to it.
Usage counter will be droppepd by svc_destroy() later in nfs_callback_up().

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

```
 fs/nfs/callback.c |   55 ++++++++++++++++++++++++++++++++++++++++----------
 1 files changed, 45 insertions(+), 10 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index f3d7ee2..b34ccc6 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -242,12 +242,46 @@ static inline void nfs_callback_bc_serv(u32 minorversion, struct
rpc_xprt *xprt,
 }
 #endif /* CONFIG_NFS_V4_1 */

+static struct svc_serv *nfs_callback_create_svc(int minorversion)
+{
+ struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
+ struct svc_serv *serv;
+
+ /*
+  * Check whether we're already up and running.
+  */
+ if (cb_info->task) {
+  /*
+   * Note: increase service usage, because later in case of error
+   * svc_destroy() will be called.
+   */
+  svc_get(cb_info->serv);
+  return cb_info->serv;
+ }
+
+ /*
+  * Sanity check: if there's no task,
+  * we should be the first user ...
+  */
+ if (cb_info->users)
+  printk(KERN_WARNING "nfs_callback_up: no kthread, %d users??\n",
+   cb_info->users);
+
+ serv = svc_create(&nfs4_callback_program, NFS4_CALLBACK_BUFSIZE, NULL);
+ if (!serv) {
+  printk(KERN_ERR "lockd_up: create service failed\n");
+  return ERR_PTR(-ENOMEM);
+ }
+ dprintk("nfs_callback_up: service created\n");
+ return serv;
+}
+
 /*
  * Bring up the callback thread if it is not already up.
```

```
  */
 int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
 {
- struct svc_serv *serv = NULL;
+ struct svc_serv *serv;
  struct svc_rqst *rqstp;
  int (*callback_svc)(void *vrqstp);
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
@@ -257,15 +291,17 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  struct net *net = current->nsproxy->net_ns;

  mutex_lock(&nfs_callback_mutex);
+
+ serv = nfs_callback_create_svc(minorversion);
+ if (IS_ERR(serv)) {
+  ret = PTR_ERR(serv);
+  goto err_create;
+ }
+
  if (cb_info->users++ || cb_info->task != NULL) {
   nfs_callback_bc_serv(minorversion, xprt, cb_info);
   goto out;
  }
- serv = svc_create(&nfs4_callback_program, NFS4_CALLBACK_BUFSIZE, NULL);
- if (!serv) {
-  ret = -ENOMEM;
-  goto out_err;
- }

  ret = svc_bind(serv, net);
  if (ret < 0) {
@@ -306,16 +342,15 @@ out:
   * on both success and failure so that the refcount is 1 when the
   * thread exits.
   */
- if (serv)
-  svc_destroy(serv);
+ svc_destroy(serv);
+err_create:
  mutex_unlock(&nfs_callback_mutex);
  return ret;
 out_err:
  dprintk("NFS: Couldn't create callback socket or server thread; "
   "err = %d\n", ret);
  cb_info->users--;
- if (serv)
-  svc_shutdown_net(serv, net);
+ svc_shutdown_net(serv, net);
```

```
 goto out;
}
```

Subject: [PATCH v2 03/12] NFS: move per-net callback thread initialization to nfs_callback_up_net()
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:10 GMT
View Forum Message <> Reply to Message

This new function in now called before nfs_minorversion_callback_svc_setup()).

Also few small changes:
1) current network namespace in nfs_callback_up() was replaced by transport net.
2) svc_shutdown_net() was moved prior to callback usage counter decrement
(because in case of per-net data allocation faulure svc_shutdown_net() have to
be skipped).

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

```
---
 fs/nfs/callback.c |  125 +++++++++++++++++++++++++++++++++++++++++--------------------
 1 files changed, 78 insertions(+), 47 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index b34ccc6..2ecea36 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -64,6 +64,32 @@ static struct kernel_param_ops param_ops_portnr = {

 module_param_named(callback_tcpport, nfs_callback_set_tcpport, portnr, 0644);

+static int nfs4_callback_up_net(struct svc_serv *serv, struct net *net)
+{
+ int ret;
+
+ ret = svc_create_xprt(serv, "tcp", net, PF_INET,
+    nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
+ if (ret <= 0)
+  goto out_err;
+ nfs_callback_tcpport = ret;
+ dprintk("NFS: Callback listener port = %u (af %u, net %p)\n",
+   nfs_callback_tcpport, PF_INET, net);
+
+ ret = svc_create_xprt(serv, "tcp", net, PF_INET6,
+    nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
+ if (ret > 0) {
+  nfs_callback_tcpport6 = ret;
+  dprintk("NFS: Callback listener port = %u (af %u, net %p)\n",
+    nfs_callback_tcpport6, PF_INET6, net);
```

```
+ } else if (ret != -EAFNOSUPPORT)
+  goto out_err;
+ return 0;
+
+out_err:
+ return (ret) ? ret : -ENOMEM;
+}
+
 /*
  * This is the NFSv4 callback kernel thread.
  */
@@ -105,36 +131,21 @@ nfs4_callback_svc(void *vrqstp)
 static struct svc_rqst *
 nfs4_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
 {
- int ret;
-
- ret = svc_create_xprt(serv, "tcp", xprt->xprt_net, PF_INET,
-     nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
- if (ret <= 0)
-  goto out_err;
- nfs_callback_tcpport = ret;
- dprintk("NFS: Callback listener port = %u (af %u)\n",
-    nfs_callback_tcpport, PF_INET);
-
- ret = svc_create_xprt(serv, "tcp", xprt->xprt_net, PF_INET6,
-     nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
- if (ret > 0) {
-  nfs_callback_tcpport6 = ret;
-  dprintk("NFS: Callback listener port = %u (af %u)\n",
-     nfs_callback_tcpport6, PF_INET6);
- } else if (ret == -EAFNOSUPPORT)
-  ret = 0;
- else
-  goto out_err;
-
  return svc_prepare_thread(serv, &serv->sv_pools[0], NUMA_NO_NODE);
-
-out_err:
- if (ret == 0)
-  ret = -ENOMEM;
- return ERR_PTR(ret);
 }

 #if defined(CONFIG_NFS_V4_1)
+static int nfs41_callback_up_net(struct svc_serv *serv, struct net *net)
+{
+ /*
```

```
+  * Create an svc_sock for the back channel service that shares the
+  * fore channel connection.
+  * Returns the input port (0) and sets the svc_serv bc_xprt on success
+  */
+ return svc_create_xprt(serv, "tcp-bc", net, PF_INET, 0,
+         SVC_SOCK_ANONYMOUS);
+}
+
 /*
  * The callback service for NFSv4.1 callbacks
  */
@@ -177,19 +188,6 @@ static struct svc_rqst *
 nfs41_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
 {
  struct svc_rqst *rqstp;
- int ret;
-
- /*
-  * Create an svc_sock for the back channel service that shares the
-  * fore channel connection.
-  * Returns the input port (0) and sets the svc_serv bc_xprt on success
-  */
- ret = svc_create_xprt(serv, "tcp-bc", xprt->xprt_net, PF_INET, 0,
-         SVC_SOCK_ANONYMOUS);
- if (ret < 0) {
-  rqstp = ERR_PTR(ret);
-  goto out;
- }

  /*
   * Save the svc_serv in the transport so that it can
@@ -205,7 +203,6 @@ nfs41_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
   svc_xprt_put(serv->sv_bc_xprt);
   serv->sv_bc_xprt = NULL;
  }
-out:
  dprintk("--> %s return %ld\n", __func__,
   IS_ERR(rqstp) ? PTR_ERR(rqstp) : 0);
  return rqstp;
@@ -229,6 +226,11 @@ static inline void nfs_callback_bc_serv(u32 minorversion, struct rpc_xprt
*xprt,
   xprt->bc_serv = cb_info->serv;
 }
 #else
+static int nfs41_callback_up_net(struct svc_serv *serv, struct net *net)
+{
+ return 0;
+}
```

```
+
 static inline int nfs_minorversion_callback_svc_setup(u32 minorversion,
   struct svc_serv *serv, struct rpc_xprt *xprt,
   struct svc_rqst **rqstpp, int (**callback_svc)(void *vrqstp))
@@ -242,6 +244,36 @@ static inline void nfs_callback_bc_serv(u32 minorversion, struct rpc_xprt
*xprt,
 }
 #endif /* CONFIG_NFS_V4_1 */

+static int nfs_callback_up_net(int minorversion, struct svc_serv *serv, struct net *net)
+{
+ int ret;
+
+ dprintk("NFS: create per-net callback data; net=%p\n", net);
+
+ ret = svc_bind(serv, net);
+ if (ret < 0) {
+  printk(KERN_WARNING "NFS: bind callback service failed\n");
+  goto err_bind;
+ }
+
+ if (minorversion) {
+  ret = nfs41_callback_up_net(serv, net);
+  if (ret < 0)
+   goto err_socks;
+ }
+
+ if (ret == 0)
+  ret = nfs4_callback_up_net(serv, net);
+ if (ret < 0)
+  goto err_socks;
+ return 0;
+
+err_socks:
+ svc_rpcb_cleanup(serv, net);
+err_bind:
+ return ret;
+}
+
 static struct svc_serv *nfs_callback_create_svc(int minorversion)
 {
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
@@ -288,7 +320,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  char svc_name[12];
  int ret = 0;
  int minorversion_setup;
- struct net *net = current->nsproxy->net_ns;
+ struct net *net = xprt->xprt_net;
```

```
  mutex_lock(&nfs_callback_mutex);

@@ -303,11 +335,9 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
    goto out;
   }

- ret = svc_bind(serv, net);
- if (ret < 0) {
-   printk(KERN_WARNING "NFS: bind callback service failed\n");
-   goto out_err;
- }
+ ret = nfs_callback_up_net(minorversion, serv, net);
+ if (ret < 0)
+   goto err_net;

   minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
       serv, xprt, &rqstp, &callback_svc);
@@ -347,10 +377,11 @@ err_create:
   mutex_unlock(&nfs_callback_mutex);
   return ret;
  out_err:
+ svc_shutdown_net(serv, net);
+err_net:
   dprintk("NFS: Couldn't create callback socket or server thread; "
     "err = %d\n", ret);
   cb_info->users--;
- svc_shutdown_net(serv, net);
   goto out;
  }
```

Subject: [PATCH v2 04/12] NFS: callback up - transport backchannel cleanup
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:17 GMT
View Forum Message <> Reply to Message

No need to assign transports backchannel server explicitly in
nfs41_callback_up() - there is nfs_callback_bc_serv() function for this.
By using it, nfs4_callback_up() and nfs41_callback_up() can be called without
transport argument.

Note: service have to be passed to nfs_callback_bc_serv() instead of callback,
since callback link can be uninitialized.

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

```
---
 fs/nfs/callback.c |   34 ++++++++++++++++++----------------
 1 files changed, 17 insertions(+), 17 deletions(-)
```

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 2ecea36..3e6aead 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -129,7 +129,7 @@ nfs4_callback_svc(void *vrqstp)
  * Prepare to bring up the NFSv4 callback service
  */
 static struct svc_rqst *
-nfs4_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
+nfs4_callback_up(struct svc_serv *serv)
 {
  return svc_prepare_thread(serv, &serv->sv_pools[0], NUMA_NO_NODE);
 }
@@ -185,16 +185,10 @@ nfs41_callback_svc(void *vrqstp)
  * Bring up the NFSv4.1 callback service
  */
 static struct svc_rqst *
-nfs41_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
+nfs41_callback_up(struct svc_serv *serv)
 {
  struct svc_rqst *rqstp;

- /*
-  * Save the svc_serv in the transport so that it can
-  * be referenced when the session backchannel is initialized
-  */
- xprt->bc_serv = serv;
-
  INIT_LIST_HEAD(&serv->sv_cb_list);
  spin_lock_init(&serv->sv_cb_lock);
  init_waitqueue_head(&serv->sv_cb_waitq);
@@ -209,21 +203,25 @@ nfs41_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
 }

 static inline int nfs_minorversion_callback_svc_setup(u32 minorversion,
-  struct svc_serv *serv, struct rpc_xprt *xprt,
+  struct svc_serv *serv,
   struct svc_rqst **rqstpp, int (**callback_svc)(void *vrqstp))
 {
  if (minorversion) {
-  *rqstpp = nfs41_callback_up(serv, xprt);
+  *rqstpp = nfs41_callback_up(serv);
   *callback_svc = nfs41_callback_svc;
  }
  return minorversion;
 }
```

```
 static inline void nfs_callback_bc_serv(u32 minorversion, struct rpc_xprt *xprt,
-  struct nfs_callback_data *cb_info)
+  struct svc_serv *serv)
 {
   if (minorversion)
-  xprt->bc_serv = cb_info->serv;
+  /*
+   * Save the svc_serv in the transport so that it can
+   * be referenced when the session backchannel is initialized
+   */
+  xprt->bc_serv = serv;
 }
 #else
 static int nfs41_callback_up_net(struct svc_serv *serv, struct net *net)
@@ -232,14 +230,14 @@ static int nfs41_callback_up_net(struct svc_serv *serv, struct net *net)
 }

 static inline int nfs_minorversion_callback_svc_setup(u32 minorversion,
-  struct svc_serv *serv, struct rpc_xprt *xprt,
+  struct svc_serv *serv,
    struct svc_rqst **rqstpp, int (**callback_svc)(void *vrqstp))
 {
   return 0;
 }

 static inline void nfs_callback_bc_serv(u32 minorversion, struct rpc_xprt *xprt,
-  struct nfs_callback_data *cb_info)
+  struct svc_serv *serv)
 {
 }
 #endif /* CONFIG_NFS_V4_1 */
@@ -331,7 +329,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
   }

   if (cb_info->users++ || cb_info->task != NULL) {
-  nfs_callback_bc_serv(minorversion, xprt, cb_info);
+  nfs_callback_bc_serv(minorversion, xprt, serv);
    goto out;
   }

@@ -339,11 +337,13 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
   if (ret < 0)
    goto err_net;

+ nfs_callback_bc_serv(minorversion, xprt, serv);
+
   minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
-     serv, xprt, &rqstp, &callback_svc);
```

```
+     serv, &rqstp, &callback_svc);
  if (!minorversion_setup) {
   /* v4.0 callback setup */
-  rqstp = nfs4_callback_up(serv, xprt);
+  rqstp = nfs4_callback_up(serv);
   callback_svc = nfs4_callback_svc;
  }
```

Subject: [PATCH v2 05/12] NFS: callback service start function introduced
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:24 GMT
View Forum Message <> Reply to Message

This is just a code move, which from my POW makes code looks better.
I.e. now on start we have 3 different stages:
1) Service creation.
2) Service per-net data allocation.
3) Service start.

Patch also renames goto label "out_err:" into "err_start:" to reflect new
changes.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c | 77 ++++++++++++++++++++++++++++++++++++----------------
 1 files changed, 45 insertions(+), 32 deletions(-)

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 3e6aead..8643ff7 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -242,6 +242,46 @@ static inline void nfs_callback_bc_serv(u32 minorversion, struct rpc_xprt
*xprt,
 }
 #endif /* CONFIG_NFS_V4_1 */

+static int nfs_callback_start_svc(int minorversion, struct rpc_xprt *xprt,
+      struct svc_serv *serv)
+{
+ struct svc_rqst *rqstp;
+ int (*callback_svc)(void *vrqstp);
+ struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
+ char svc_name[12];
+ int ret;
+ int minorversion_setup;
+
+ nfs_callback_bc_serv(minorversion, xprt, serv);
+
```

```
+ minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
+     serv, &rqstp, &callback_svc);
+ if (!minorversion_setup) {
+  /* v4.0 callback setup */
+  rqstp = nfs4_callback_up(serv);
+  callback_svc = nfs4_callback_svc;
+ }
+
+ if (IS_ERR(rqstp))
+  return PTR_ERR(rqstp);
+
+ svc_sock_update_bufs(serv);
+
+ sprintf(svc_name, "nfsv4.%u-svc", minorversion);
+ cb_info->serv = serv;
+ cb_info->rqst = rqstp;
+ cb_info->task = kthread_run(callback_svc, cb_info->rqst, svc_name);
+ if (IS_ERR(cb_info->task)) {
+  ret = PTR_ERR(cb_info->task);
+  svc_exit_thread(cb_info->rqst);
+  cb_info->rqst = NULL;
+  cb_info->task = NULL;
+  return PTR_ERR(cb_info->task);
+ }
+ dprintk("nfs_callback_up: service started\n");
+ return 0;
+}
+
 static int nfs_callback_up_net(int minorversion, struct svc_serv *serv, struct net *net)
 {
  int ret;
@@ -312,12 +352,8 @@ static struct svc_serv *nfs_callback_create_svc(int minorversion)
 int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
 {
  struct svc_serv *serv;
- struct svc_rqst *rqstp;
- int (*callback_svc)(void *vrqstp);
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
- char svc_name[12];
  int ret = 0;
- int minorversion_setup;
  struct net *net = xprt->xprt_net;

  mutex_lock(&nfs_callback_mutex);
@@ -337,34 +373,10 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  if (ret < 0)
   goto err_net;
```

```
- nfs_callback_bc_serv(minorversion, xprt, serv);
-
- minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
-     serv, &rqstp, &callback_svc);
- if (!minorversion_setup) {
-  /* v4.0 callback setup */
-  rqstp = nfs4_callback_up(serv);
-  callback_svc = nfs4_callback_svc;
- }
-
- if (IS_ERR(rqstp)) {
-  ret = PTR_ERR(rqstp);
-  goto out_err;
- }
-
- svc_sock_update_bufs(serv);
+ ret = nfs_callback_start_svc(minorversion, xprt, serv);
+ if (ret < 0)
+  goto err_start;

- sprintf(svc_name, "nfsv4.%u-svc", minorversion);
- cb_info->serv = serv;
- cb_info->rqst = rqstp;
- cb_info->task = kthread_run(callback_svc, cb_info->rqst, svc_name);
- if (IS_ERR(cb_info->task)) {
-  ret = PTR_ERR(cb_info->task);
-  svc_exit_thread(cb_info->rqst);
-  cb_info->rqst = NULL;
-  cb_info->task = NULL;
-  goto out_err;
- }
 out:
  /*
   * svc_create creates the svc_serv with sv_nrthreads == 1, and then
@@ -376,7 +388,8 @@ out:
 err_create:
  mutex_unlock(&nfs_callback_mutex);
  return ret;
-out_err:
+
+err_start:
  svc_shutdown_net(serv, net);
 err_net:
  dprintk("NFS: Couldn't create callback socket or server thread; "
```

Subject: [PATCH v2 06/12] NFS: callback up - users counting cleanup

Usage coutner now increased only is the service was started sccessfully.
Even if service is running already, then goto is not required anymore, because
service creation and start will be skipped.
With this patch code looks clearer.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c | 22 ++++++++++------------
 1 files changed, 10 insertions(+), 12 deletions(-)

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 8643ff7..11201dd 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -254,6 +254,9 @@ static int nfs_callback_start_svc(int minorversion, struct rpc_xprt *xprt,

  nfs_callback_bc_serv(minorversion, xprt, serv);

+ if (cb_info->task)
+  return 0;
+
  minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
      serv, &rqstp, &callback_svc);
  if (!minorversion_setup) {
@@ -309,6 +312,8 @@ static int nfs_callback_up_net(int minorversion, struct svc_serv *serv,
struct n
 err_socks:
  svc_rpcb_cleanup(serv, net);
 err_bind:
+ dprintk("NFS: Couldn't create callback socket: err = %d; "
+   "net = %p\n", ret, net);
  return ret;
 }

@@ -353,7 +358,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
 {
  struct svc_serv *serv;
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];
- int ret = 0;
+ int ret;
  struct net *net = xprt->xprt_net;

  mutex_lock(&nfs_callback_mutex);
@@ -364,11 +369,6 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
   goto err_create;
  }
```

```
- if (cb_info->users++ || cb_info->task != NULL) {
-  nfs_callback_bc_serv(minorversion, xprt, serv);
-  goto out;
- }
-
  ret = nfs_callback_up_net(minorversion, serv, net);
  if (ret < 0)
   goto err_net;
@@ -377,13 +377,14 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  if (ret < 0)
   goto err_start;

-out:
+ cb_info->users++;
  /*
   * svc_create creates the svc_serv with sv_nrthreads == 1, and then
   * svc_prepare_thread increments that. So we need to call svc_destroy
   * on both success and failure so that the refcount is 1 when the
   * thread exits.
   */
+err_net:
  svc_destroy(serv);
 err_create:
  mutex_unlock(&nfs_callback_mutex);
@@ -391,11 +392,8 @@ err_create:

 err_start:
  svc_shutdown_net(serv, net);
-err_net:
- dprintk("NFS: Couldn't create callback socket or server thread; "
-  "err = %d\n", ret);
- cb_info->users--;
- goto out;
+ dprintk("NFS: Couldn't create server thread; err = %d\n", ret);
+ goto err_net;
 }

 /*
```

Subject: [PATCH v2 07/12] NFS: make nfs_callback_tcpport per network context
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:38 GMT
View Forum Message <> Reply to Message

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c | 7 ++++---

```
 fs/nfs/callback.h  |    1 -
 fs/nfs/netns.h     |    1 +
 fs/nfs/nfs4state.c |    5 ++++-
 4 files changed, 9 insertions(+), 5 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 11201dd..9380184 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -24,6 +24,7 @@
 #include "nfs4_fs.h"
 #include "callback.h"
 #include "internal.h"
+#include "netns.h"

 #define NFSDBG_FACILITY NFSDBG_CALLBACK

@@ -39,7 +40,6 @@ static DEFINE_MUTEX(nfs_callback_mutex);
 static struct svc_program nfs4_callback_program;

 unsigned int nfs_callback_set_tcpport;
-unsigned short nfs_callback_tcpport;
 unsigned short nfs_callback_tcpport6;
 #define NFS_CALLBACK_MAXPORTNR (65535U)

@@ -67,14 +67,15 @@ module_param_named(callback_tcpport, nfs_callback_set_tcpport,
portnr, 0644);
 static int nfs4_callback_up_net(struct svc_serv *serv, struct net *net)
 {
  int ret;
+ struct nfs_net *nn = net_generic(net, nfs_net_id);

  ret = svc_create_xprt(serv, "tcp", net, PF_INET,
     nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
  if (ret <= 0)
   goto out_err;
- nfs_callback_tcpport = ret;
+ nn->nfs_callback_tcpport = ret;
  dprintk("NFS: Callback listener port = %u (af %u, net %p)\n",
-   nfs_callback_tcpport, PF_INET, net);
+   nn->nfs_callback_tcpport, PF_INET, net);

  ret = svc_create_xprt(serv, "tcp", net, PF_INET6,
     nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
diff --git a/fs/nfs/callback.h b/fs/nfs/callback.h
index 6d900cf..8006959 100644
--- a/fs/nfs/callback.h
+++ b/fs/nfs/callback.h
```

```
@@ -208,7 +208,6 @@ extern int nfs4_set_callback_sessionid(struct nfs_client *clp);
 #define NFS41_BC_MAX_CALLBACKS 1

 extern unsigned int nfs_callback_set_tcpport;
-extern unsigned short nfs_callback_tcpport;
 extern unsigned short nfs_callback_tcpport6;

 #endif /* __LINUX_FS_NFS_CALLBACK_H */
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index aa14ec3..8e4e8f8 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -18,6 +18,7 @@ struct nfs_net {
  struct list_head nfs_volume_list;
 #ifdef CONFIG_NFS_V4
  struct idr cb_ident_idr; /* Protected by nfs_client_lock */
+ unsigned short nfs_callback_tcpport;
 #endif
  spinlock_t nfs_client_lock;
 };
diff --git a/fs/nfs/nfs4state.c b/fs/nfs/nfs4state.c
index 0f43414..5d68b3b 100644
--- a/fs/nfs/nfs4state.c
+++ b/fs/nfs/nfs4state.c
@@ -57,6 +57,8 @@
 #include "internal.h"
 #include "pnfs.h"

+#include "netns.h"
+
 #define OPENOWNER_POOL_SIZE 8

 const nfs4_stateid zero_stateid;
@@ -71,10 +73,11 @@ int nfs4_init_clientid(struct nfs_client *clp, struct rpc_cred *cred)
  };
  unsigned short port;
  int status;
+ struct nfs_net *nn = net_generic(clp->net, nfs_net_id);

  if (test_bit(NFS4CLNT_LEASE_CONFIRM, &clp->cl_state))
   goto do_confirm;
- port = nfs_callback_tcpport;
+ port = nn->nfs_callback_tcpport;
  if (clp->cl_addr.ss_family == AF_INET6)
   port = nfs_callback_tcpport6;
```

Subject: [PATCH v2 08/12] NFS: make nfs_callback_tcpport6 per network context
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:46 GMT
View Forum Message <> Reply to Message

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

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 9380184..d25085f 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -40,7 +40,6 @@ static DEFINE_MUTEX(nfs_callback_mutex);
 static struct svc_program nfs4_callback_program;

 unsigned int nfs_callback_set_tcpport;
-unsigned short nfs_callback_tcpport6;
 #define NFS_CALLBACK_MAXPORTNR (65535U)

 static int param_set_portnr(const char *val, const struct kernel_param *kp)
@@ -80,9 +79,9 @@ static int nfs4_callback_up_net(struct svc_serv *serv, struct net *net)
  ret = svc_create_xprt(serv, "tcp", net, PF_INET6,
     nfs_callback_set_tcpport, SVC_SOCK_ANONYMOUS);
  if (ret > 0) {
-  nfs_callback_tcpport6 = ret;
+  nn->nfs_callback_tcpport6 = ret;
   dprintk("NFS: Callback listener port = %u (af %u, net %p)\n",
-    nfs_callback_tcpport6, PF_INET6, net);
+    nn->nfs_callback_tcpport6, PF_INET6, net);
  } else if (ret != -EAFNOSUPPORT)
   goto out_err;
  return 0;
diff --git a/fs/nfs/callback.h b/fs/nfs/callback.h
index 8006959..86b5671 100644
--- a/fs/nfs/callback.h
+++ b/fs/nfs/callback.h
@@ -208,6 +208,5 @@ extern int nfs4_set_callback_sessionid(struct nfs_client *clp);
 #define NFS41_BC_MAX_CALLBACKS 1

 extern unsigned int nfs_callback_set_tcpport;
-extern unsigned short nfs_callback_tcpport6;

 #endif /* __LINUX_FS_NFS_CALLBACK_H */
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index 8e4e8f8..bd72a55 100644
```

```
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -19,6 +19,7 @@ struct nfs_net {
 #ifdef CONFIG_NFS_V4
  struct idr cb_ident_idr; /* Protected by nfs_client_lock */
  unsigned short nfs_callback_tcpport;
+ unsigned short nfs_callback_tcpport6;
 #endif
  spinlock_t nfs_client_lock;
 };
diff --git a/fs/nfs/nfs4state.c b/fs/nfs/nfs4state.c
index 5d68b3b..bee4bc1 100644
--- a/fs/nfs/nfs4state.c
+++ b/fs/nfs/nfs4state.c
@@ -79,7 +79,7 @@ int nfs4_init_clientid(struct nfs_client *clp, struct rpc_cred *cred)
   goto do_confirm;
  port = nn->nfs_callback_tcpport;
  if (clp->cl_addr.ss_family == AF_INET6)
-  port = nfs_callback_tcpport6;
+  port = nn->nfs_callback_tcpport6;

  status = nfs4_proc_setclientid(clp, NFS4_CALLBACK, port, cred, &clid);
  if (status != 0)
```

Subject: [PATCH v2 09/12] NFS; callback per-net usage counting introduced
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:36:54 GMT
View Forum Message <> Reply to Message

This patch also introduces refcount-aware nfs_callback_down_net() wrapper for
svc_shutdown_net().

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

```
---
 fs/nfs/callback.c |   19 ++++++++++++++++--
 fs/nfs/netns.h    |    1 +
 2 files changed, 18 insertions(+), 2 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index d25085f..dbedd7e 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -285,10 +285,25 @@ static int nfs_callback_start_svc(int minorversion, struct rpc_xprt *xprt,
  return 0;
 }

+static void nfs_callback_down_net(u32 minorversion, struct svc_serv *serv, struct net *net)
+{
```

```
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+
+ if (--nn->cb_users[minorversion])
+  return;
+
+ dprintk("NFS: destroy per-net callback data; net=%p\n", net);
+ svc_shutdown_net(serv, net);
+}
+
 static int nfs_callback_up_net(int minorversion, struct svc_serv *serv, struct net *net)
 {
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
  int ret;

+ if (nn->cb_users[minorversion]++)
+  return 0;
+
  dprintk("NFS: create per-net callback data; net=%p\n", net);

  ret = svc_bind(serv, net);
@@ -391,7 +406,7 @@ err_create:
  return ret;

 err_start:
- svc_shutdown_net(serv, net);
+ nfs_callback_down_net(minorversion, serv, net);
  dprintk("NFS: Couldn't create server thread; err = %d\n", ret);
  goto err_net;
 }
@@ -404,10 +419,10 @@ void nfs_callback_down(int minorversion, struct net *net)
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];

  mutex_lock(&nfs_callback_mutex);
+ nfs_callback_down_net(minorversion, cb_info->serv, net);
  cb_info->users--;
  if (cb_info->users == 0 && cb_info->task != NULL) {
   kthread_stop(cb_info->task);
-  svc_shutdown_net(cb_info->serv, net);
   svc_exit_thread(cb_info->rqst);
   cb_info->serv = NULL;
   cb_info->rqst = NULL;
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index bd72a55..db3e495 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -20,6 +20,7 @@ struct nfs_net {
  struct idr cb_ident_idr; /* Protected by nfs_client_lock */
  unsigned short nfs_callback_tcpport;
```

```
  unsigned short nfs_callback_tcpport6;
+ int cb_users[NFS4_MAX_MINOR_VERSION + 1];
 #endif
  spinlock_t nfs_client_lock;
 };
```

Subject: [PATCH v2 10/12] NFS: add debug messages to callback down function
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:37:02 GMT
View Forum Message <> Reply to Message

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c | 2 ++
 1 files changed, 2 insertions(+), 0 deletions(-)

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index dbedd7e..2db796a 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -423,7 +423,9 @@ void nfs_callback_down(int minorversion, struct net *net)
   cb_info->users--;
   if (cb_info->users == 0 && cb_info->task != NULL) {
     kthread_stop(cb_info->task);
+   dprintk("nfs_callback_down: service stopped\n");
     svc_exit_thread(cb_info->rqst);
+   dprintk("nfs_callback_down: service destroyed\n");
     cb_info->serv = NULL;
     cb_info->rqst = NULL;
     cb_info->task = NULL;
```

Subject: [PATCH v2 11/12] Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:37:09 GMT
View Forum Message <> Reply to Message

From: root (none) <root@sighted>

---
 fs/nfs/client.c | 4 +++-
 fs/nfs/idmap.c | 5 ++++-
 2 files changed, 7 insertions(+), 2 deletions(-)

```
diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 44cd70f..ae29d4f 100644
```

```
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_

 #ifdef CONFIG_NFS_V4
  err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
- if (err)
+ if (err) {
+  put_net(clp->net);
   goto error_cleanup;
+ }

  spin_lock_init(&clp->cl_lock);
  INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index b7f348b..c9c3e1b 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -487,7 +487,7 @@ nfs_idmap_delete(struct nfs_client *clp)
  kfree(idmap);
 }

-static int __rpc_pipefs_event(struct nfs_client *clp, unsigned long event,
+static noinline int __rpc_pipefs_event(struct nfs_client *clp, unsigned long event,
         struct super_block *sb)
 {
  int err = 0;
@@ -533,8 +533,11 @@ static struct nfs_client *nfs_get_client_for_event(struct net *net, int
event)

  spin_lock(&nn->nfs_client_lock);
  list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
+  BUG_ON(atomic_read(&clp->cl_count) == 0);
   if (clp->rpc_ops != &nfs_v4_clientops)
    continue;
+  BUG_ON(clp->cl_idmap == NULL);
+  BUG_ON(clp->cl_idmap->idmap_pipe == NULL);
   cl_dentry = clp->cl_idmap->idmap_pipe->dentry;
   if (((event == RPC_PIPEFS_MOUNT) && cl_dentry) ||
       ((event == RPC_PIPEFS_UMOUNT) && !cl_dentry))
```

Subject: [PATCH v2 12/12] NFS: get module in idmap PipeFS notifier callback
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 07:37:17 GMT
View Forum Message <> Reply to Message

This is bug fix.

Notifier callback is called from SUNRPC module. So before dereferencing NFS
module we have to make sure, that it's alive.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/idmap.c | 4 ++++
 1 files changed, 4 insertions(+), 0 deletions(-)

diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index c9c3e1b..f55b80a 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -557,12 +557,16 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long
event,
  struct nfs_client *clp;
  int error = 0;

+ if (!try_module_get(THIS_MODULE))
+ return 0;
+
  while ((clp = nfs_get_client_for_event(sb->s_fs_info, event))) {
   error = __rpc_pipefs_event(clp, event, sb);
   nfs_put_client(clp);
   if (error)
    break;
  }
+ module_put(THIS_MODULE);
  return error;
 }

Subject: [PATCH v3] NFS: put net on idr allocation failure
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 08:03:45 GMT
View Forum Message <> Reply to Message

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/client.c | 4 +++-
 1 files changed, 3 insertions(+), 1 deletions(-)

diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 44cd70f..ae29d4f 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_

 #ifdef CONFIG_NFS_V4

```
  err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
- if (err)
+ if (err) {
+  put_net(clp->net);
   goto error_cleanup;
+ }

  spin_lock_init(&clp->cl_lock);
  INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
```

Subject: Re: [PATCH v3] NFS: put net on idr allocation failure
Posted by Myklebust, Trond on Tue, 22 May 2012 14:37:41 GMT
View Forum Message <> Reply to Message

On Tue, 2012-05-22 at 12:03 +0400, Stanislav Kinsbursky wrote:
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> ---
> fs/nfs/client.c | 4 +++-
> 1 files changed, 3 insertions(+), 1 deletions(-)
>
> diff --git a/fs/nfs/client.c b/fs/nfs/client.c
> index 44cd70f..ae29d4f 100644
> --- a/fs/nfs/client.c
> +++ b/fs/nfs/client.c
> @@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_
>
> #ifdef CONFIG_NFS_V4
>   err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
> - if (err)
> + if (err) {
> + put_net(clp->net);
>   goto error_cleanup;
> + }
>
>   spin_lock_init(&clp->cl_lock);
>   INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
>

Let's just move the actual get_net() call to the end of the function.

--
Trond Myklebust
Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com

www.netapp.com

Subject: Re: [PATCH v2 11/12] Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Posted by Myklebust, Trond on Tue, 22 May 2012 14:38:38 GMT
View Forum Message <> Reply to Message

On Tue, 2012-05-22 at 11:37 +0400, Stanislav Kinsbursky wrote:
> From: root (none) <root@sighted>
>

What happened to the Subject: and changelog?

--
Trond Myklebust
Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com
www.netapp.com

Subject: Re: [PATCH v2 11/12] Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 14:39:48 GMT
View Forum Message <> Reply to Message

On 22.05.2012 18:38, Myklebust, Trond wrote:
> On Tue, 2012-05-22 at 11:37 +0400, Stanislav Kinsbursky wrote:
>> From: root (none)<root@sighted>
>>
>
> What happened to the Subject: and changelog?
>

Sorry for this.
That's why I've sent "v3" - added subject and changelog.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v3] NFS: put net on idr allocation failure
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 14:41:35 GMT

On 22.05.2012 18:37, Myklebust, Trond wrote:
> On Tue, 2012-05-22 at 12:03 +0400, Stanislav Kinsbursky wrote:
>> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
>> ---
>> fs/nfs/client.c | 4 +++-
>> 1 files changed, 3 insertions(+), 1 deletions(-)
>>
>> diff --git a/fs/nfs/client.c b/fs/nfs/client.c
>> index 44cd70f..ae29d4f 100644
>> --- a/fs/nfs/client.c
>> +++ b/fs/nfs/client.c
>> @@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct
nfs_client_initdata *cl_
>>
>> #ifdef CONFIG_NFS_V4
>>   err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
>> - if (err)
>> + if (err) {
>> + put_net(clp->net);
>>   goto error_cleanup;
>> + }
>>
>>   spin_lock_init(&clp->cl_lock);
>>   INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
>>
>
> Let's just move the actual get_net() call to the end of the function.
>

Ok. But it will look a bit strange, like this:

clp->net = net;
...
nfs_get_cb_ident_idr
...
get_net(clp->net)

Or I can pass net to nfs_get_cb_ident_idr() as a parameter.
Which solution is more preferable&?

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH v3] NFS: put net on idr allocation failure

On Tue, 2012-05-22 at 18:41 +0400, Stanislav Kinsbursky wrote:
> On 22.05.2012 18:37, Myklebust, Trond wrote:
> > On Tue, 2012-05-22 at 12:03 +0400, Stanislav Kinsbursky wrote:
> >> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
> >> ---
> >> fs/nfs/client.c | 4 +++-
> >> 1 files changed, 3 insertions(+), 1 deletions(-)
> >>
> >> diff --git a/fs/nfs/client.c b/fs/nfs/client.c
> >> index 44cd70f..ae29d4f 100644
> >> --- a/fs/nfs/client.c
> >> +++ b/fs/nfs/client.c
> >> @@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct
nfs_client_initdata *cl_
> >>
> >> #ifdef CONFIG_NFS_V4
> >>   err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
> >> - if (err)
> >> + if (err) {
> >> + put_net(clp->net);
> >>   goto error_cleanup;
> >> + }
> >>
> >>   spin_lock_init(&clp->cl_lock);
> >>   INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
> >>
> >
> > Let's just move the actual get_net() call to the end of the function.
> >
>
> Ok. But it will look a bit strange, like this:
>
> clp->net = net;
> ...
> nfs_get_cb_ident_idr
> ...
> get_net(clp->net)
>
> Or I can pass net to nfs_get_cb_ident_idr() as a parameter.
> Which solution is more preferable&?

Just put the get_net() at the end. It doesn't matter if it looks a bit
odd: the intent is still obvious.

--
Trond Myklebust

Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com
www.netapp.com

Subject: Re: [PATCH v3] NFS: put net on idr allocation failure
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 14:46:20 GMT
View Forum Message <> Reply to Message

On 22.05.2012 18:45, Myklebust, Trond wrote:
> On Tue, 2012-05-22 at 18:41 +0400, Stanislav Kinsbursky wrote:
>> On 22.05.2012 18:37, Myklebust, Trond wrote:
>>> On Tue, 2012-05-22 at 12:03 +0400, Stanislav Kinsbursky wrote:
>>>> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
>>>> ---
>>>> fs/nfs/client.c | 4 +++-
>>>> 1 files changed, 3 insertions(+), 1 deletions(-)
>>>>
>>>> diff --git a/fs/nfs/client.c b/fs/nfs/client.c
>>>> index 44cd70f..ae29d4f 100644
>>>> --- a/fs/nfs/client.c
>>>> +++ b/fs/nfs/client.c
>>>> @@ -176,8 +176,10 @@ static struct nfs_client *nfs_alloc_client(const struct
nfs_client_initdata *cl_
>>>>
>>>> #ifdef CONFIG_NFS_V4
>>>>   err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
>>>> - if (err)
>>>> + if (err) {
>>>> + put_net(clp->net);
>>>>   goto error_cleanup;
>>>> + }
>>>>
>>>>   spin_lock_init(&clp->cl_lock);
>>>>   INIT_DELAYED_WORK(&clp->cl_renewd, nfs4_renew_state);
>>>>
>>>
>>> Let's just move the actual get_net() call to the end of the function.
>>>
>>
>> Ok. But it will look a bit strange, like this:
>>
>> clp->net = net;
>> ...
>> nfs_get_cb_ident_idr
>> ...

>> get_net(clp->net)
>>
>> Or I can pass net to nfs_get_cb_ident_idr() as a parameter.
>> Which solution is more preferable&?
>
> Just put the get_net() at the end. It doesn't matter if it looks a bit
> odd: the intent is still obvious.
>

Sure.

--
Best regards,
Stanislav Kinsbursky

Subject: [PATCH v4] NFS: get client net after idr allocation
Posted by Stanislav Kinsbursky on Tue, 22 May 2012 14:52:56 GMT
View Forum Message <> Reply to Message

Allocation can fail. So instead of put net in case of failure, get net after
allocation.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/client.c | 3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

```
diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index b59419a..b694971 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -172,7 +172,7 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_
  clp->cl_rpcclient = ERR_PTR(-EINVAL);

  clp->cl_proto = cl_init->proto;
- clp->net = get_net(cl_init->net);
+ clp->net = cl_init->net;

 #ifdef CONFIG_NFS_V4
  err = nfs_get_cb_ident_idr(clp, cl_init->minorversion);
@@ -187,6 +187,7 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_
  clp->cl_minorversion = cl_init->minorversion;
  clp->cl_mvops = nfs_v4_minor_ops[cl_init->minorversion];
 #endif
+ get_net(clp->net);
```

```
cred = rpc_lookup_machine_cred("*");
if (!IS_ERR(cred))
 clp->cl_machine_cred = cred;
```