Subject: [PATCH v4 00/10] NFS: callback threads containerization
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 13:59:59 GMT
View Forum Message <> Reply to Message

v4: rebased on trondmy/linux-next

Hi, Trond.
This patch set is the last significant part of client containerization.
It's pending for a long time already. And I really feel, that it's ready for
inclusion.
Without this patch set per-net callback resources management is broken.
Hoping you can take it for next kernel.

The following series implements...

Stanislav Kinsbursky (10):
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

fs/nfs/callback.c | 302 ++----------------
fs/nfs/callback.h | 4 -
fs/nfs/netns.h | 3 +
fs/nfs/nfs4client.c | 2
fs/nfs/nfs4state.c | 6 +
5 files changed, 211 insertions(+), 106 deletions(-)

Subject: [PATCH v4 01/10] NFS: pass net to nfs_callback_down()
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:06 GMT
View Forum Message <> Reply to Message

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
fs/nfs/callback.c | 4 ++--
fs/nfs/callback.h | 2 +-
fs/nfs/nfs4client.c | 2 +-
3 files changed, 4 insertions(+), 4 deletions(-)

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 4c8459e..51297b2 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -301,7 +301,7 @@ out_err:
 /*
  * Kill the callback thread if it's no longer being used.
  */
-void nfs_callback_down(int minorversion)
+void nfs_callback_down(int minorversion, struct net *net)
 {
  struct nfs_callback_data *cb_info = &nfs_callback_info[minorversion];

@@ -309,7 +309,7 @@ void nfs_callback_down(int minorversion)
  cb_info->users--;
  if (cb_info->users == 0 && cb_info->task != NULL) {
   kthread_stop(cb_info->task);
-  svc_shutdown_net(cb_info->serv, &init_net);
+  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/callback.h b/fs/nfs/callback.h
index b44d7b1..3094044 100644
--- a/fs/nfs/callback.h
+++ b/fs/nfs/callback.h
@@ -194,7 +194,7 @@ extern __be32 nfs4_callback_recall(struct cb_recallargs *args, void
*dummy,
        struct cb_process_state *cps);
 #if IS_ENABLED(CONFIG_NFS_V4)
 extern int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt);
-extern void nfs_callback_down(int minorversion);
+extern void nfs_callback_down(int minorversion, struct net *net);
 extern int nfs4_validate_delegation_stateid(struct nfs_delegation *delegation,
         const nfs4_stateid *stateid);
 extern int nfs4_set_callback_sessionid(struct nfs_client *clp);
diff --git a/fs/nfs/nfs4client.c b/fs/nfs/nfs4client.c
index b895629..1dfaa13 100644
--- a/fs/nfs/nfs4client.c
+++ b/fs/nfs/nfs4client.c
@@ -85,7 +85,7 @@ error:
 static void nfs4_destroy_callback(struct nfs_client *clp)
 {
  if (__test_and_clear_bit(NFS_CS_CALLBACK, &clp->cl_res_state))
-  nfs_callback_down(clp->cl_mvops->minor_version);
+  nfs_callback_down(clp->cl_mvops->minor_version, &init_net);
 }
```

static void nfs4_shutdown_client(struct nfs_client *clp)

Subject: [PATCH v4 02/10] NFS: callback service creation function introduced
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:11 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 |   63 +++++++++++++++++++++++++++++++++++++++++++-----------
 1 files changed, 49 insertions(+), 14 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 51297b2..18efeb5 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -217,12 +217,50 @@ 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_create_svc: no kthread, %d users??\n",
+   cb_info->users);
+
+ serv = svc_create(&nfs4_callback_program, NFS4_CALLBACK_BUFSIZE, NULL);
+ if (!serv) {
+  printk(KERN_ERR "nfs_callback_create_svc: create service failed\n");
+  return ERR_PTR(-ENOMEM);
+ }
+ /* As there is only one thread we need to over-ride the
+  * default maximum of 80 connections
+  */
+ serv->sv_maxconn = 1024;
+ dprintk("nfs_callback_create_svc: 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];
@@ -232,19 +270,17 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  struct net *net = &init_net;

  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;
- }
- /* As there is only one thread we need to over-ride the
-  * default maximum of 80 connections
```

```
-  */
- serv->sv_maxconn = 1024;

  ret = svc_bind(serv, net);
  if (ret < 0) {
@@ -285,16 +321,15 @@ 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 v4 03/10] NFS: move per-net callback thread initialization to nfs_callback_up_net()
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:16 GMT

View Forum Message <> Reply to Message

---

v4:
1) Callback transport creation routine selection by version simlified.

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   |  133 +++++++++++++++++++++++++++++++++++-------------------
 fs/nfs/nfs4client.c |    2 -
 2 files changed, 87 insertions(+), 48 deletions(-)
```

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 18efeb5..a53b4e5 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -39,6 +39,32 @@ static struct svc_program nfs4_callback_program;

 unsigned short nfs_callback_tcpport6;

+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.
  */
@@ -80,36 +106,21 @@ 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", &init_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", &init_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
   */
@@ -152,19 +163,6 @@ 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", &init_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
@@ -180,7 +178,6 @@ 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;
@@ -204,6 +201,11 @@ 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))
@@ -217,6 +219,44 @@ 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;
+ }
+
+ switch (minorversion) {
```

```
+  case 0:
+   ret = nfs4_callback_up_net(serv, net);
+   break;
+  case 1:
+   ret = nfs41_callback_up_net(serv, net);
+   break;
+  default:
+   printk(KERN_ERR "NFS: unknown callback version: %d\n",
+     minorversion);
+   ret = -EINVAL;
+   break;
+ }
+
+ if (ret < 0) {
+  printk(KERN_ERR "NFS: callback service start failed\n");
+  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];
@@ -267,7 +307,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  char svc_name[12];
  int ret = 0;
  int minorversion_setup;
- struct net *net = &init_net;
+ struct net *net = xprt->xprt_net;

  mutex_lock(&nfs_callback_mutex);

@@ -282,11 +322,9 @@ 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);
@@ -326,10 +364,11 @@ 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;
 }

diff --git a/fs/nfs/nfs4client.c b/fs/nfs/nfs4client.c
index 1dfaa13..a89aebb 100644
--- a/fs/nfs/nfs4client.c
+++ b/fs/nfs/nfs4client.c
@@ -85,7 +85,7 @@ error:
 static void nfs4_destroy_callback(struct nfs_client *clp)
 {
  if (__test_and_clear_bit(NFS_CS_CALLBACK, &clp->cl_res_state))
-  nfs_callback_down(clp->cl_mvops->minor_version, &init_net);
+  nfs_callback_down(clp->cl_mvops->minor_version, clp->cl_net);
 }

 static void nfs4_shutdown_client(struct nfs_client *clp)
```

Subject: [PATCH v4 04/10] NFS: callback up - transport backchannel cleanup
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:21 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 a53b4e5..a528cb7 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -104,7 +104,7 @@ 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);
 }
@@ -160,16 +160,10 @@ 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);
@@ -184,21 +178,25 @@ 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)
@@ -207,14 +205,14 @@ 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 */
@@ -318,7 +316,7 @@ 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;
   }

@@ -326,11 +324,13 @@ 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 v4 05/10] NFS: callback service start function introduced
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:26 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 a528cb7..5d5f9d1 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -217,6 +217,46 @@ 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;
@@ -299,12 +339,8 @@ 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);
@@ -324,34 +360,10 @@ 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
@@ -363,7 +375,8 @@ 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 v4 06/10] 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 5d5f9d1..64e87ec 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -229,6 +229,9 @@ 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) {
@@ -292,6 +295,8 @@ 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;
 }

@@ -340,7 +345,7 @@ 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);
@@ -351,11 +356,6 @@ 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;
@@ -364,13 +364,14 @@ 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);
@@ -378,11 +379,8 @@ 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 v4 07/10] NFS: make nfs_callback_tcpport per network context
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:36 GMT
View Forum Message <> Reply to Message

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c | 6 ++++--

```
 fs/nfs/callback.h  |    1 -
 fs/nfs/netns.h     |    1 +
 fs/nfs/nfs4state.c |    4 +++-
 4 files changed, 8 insertions(+), 4 deletions(-)

diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 64e87ec..94aa9d8 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -23,6 +23,7 @@
 #include "nfs4_fs.h"
 #include "callback.h"
 #include "internal.h"
+#include "netns.h"

 #define NFSDBG_FACILITY NFSDBG_CALLBACK

@@ -42,14 +43,15 @@ unsigned short nfs_callback_tcpport6;
 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 3094044..1c167d1 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 0539de1..1538d3a 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -22,6 +22,7 @@ struct nfs_net {
  struct list_head nfs_volume_list;
 #if IS_ENABLED(CONFIG_NFS_V4)
  struct idr cb_ident_idr; /* Protected by nfs_client_lock */
+ unsigned short nfs_callback_tcpport;
 #endif
  spinlock_t nfs_client_lock;
  struct timespec boot_time;
diff --git a/fs/nfs/nfs4state.c b/fs/nfs/nfs4state.c
index 55148de..afd06b4 100644
--- a/fs/nfs/nfs4state.c
+++ b/fs/nfs/nfs4state.c
@@ -56,6 +56,7 @@
 #include "delegation.h"
 #include "internal.h"
 #include "pnfs.h"
+#include "netns.h"

 #define NFSDBG_FACILITY  NFSDBG_STATE

@@ -73,10 +74,11 @@ int nfs4_init_clientid(struct nfs_client *clp, struct rpc_cred *cred)
  };
  unsigned short port;
  int status;
+ struct nfs_net *nn = net_generic(clp->cl_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 v4 08/10] NFS: make nfs_callback_tcpport6 per network context

Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:41 GMT

View Forum Message <> Reply to Message

```
Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/nfs/callback.c  |    6 ++----
 fs/nfs/callback.h  |    1 -
 fs/nfs/netns.h     |    1 +
 fs/nfs/nfs4state.c |    2 +-
 4 files changed, 4 insertions(+), 6 deletions(-)
```

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 94aa9d8..baafa0f 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -38,8 +38,6 @@ static struct nfs_callback_data
nfs_callback_info[NFS4_MAX_MINOR_VERSION + 1];
 static DEFINE_MUTEX(nfs_callback_mutex);
 static struct svc_program nfs4_callback_program;

-unsigned short nfs_callback_tcpport6;
-
 static int nfs4_callback_up_net(struct svc_serv *serv, struct net *net)
 {
  int ret;
@@ -56,9 +54,9 @@ 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 1c167d1..c07a8d4 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 1538d3a..137238b 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -23,6 +23,7 @@ struct nfs_net {
 #if IS_ENABLED(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;
  struct timespec boot_time;
diff --git a/fs/nfs/nfs4state.c b/fs/nfs/nfs4state.c
index afd06b4..2042bf0 100644
--- a/fs/nfs/nfs4state.c
+++ b/fs/nfs/nfs4state.c
@@ -80,7 +80,7 @@ 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 v4 09/10] NFS: callback per-net usage counting introduced
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:46 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 baafa0f..6dfdc83 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -260,10 +260,25 @@ 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);
@@ -378,7 +393,7 @@ 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;
 }
@@ -391,10 +406,10 @@ 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 137238b..b9c7f9b 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -24,6 +24,7 @@ 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;
  struct timespec boot_time;
```

Subject: [PATCH v4 10/10] NFS: add debug messages to callback down function
Posted by Stanislav Kinsbursky on Mon, 20 Aug 2012 14:00:51 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 6dfdc83..8ed0bc8 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -410,7 +410,9 @@ 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;