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;
 }
```