Subject: [PATCH v2 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:17:52 GMT
View Forum Message <> Reply to Message

This patch set was created in context of clone of git
branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.

v2:
1) "over-put" of PipeFS mount point in 1st patch of the series fixed, which
allow allows to bisect the patch set.
2) rebased on tag v3.2

This patch set depends on previous patch sets titled:
1) "SUNRPC: initial part of making pipefs work in net ns"
2) "SUNPRC: cleanup PipeFS for network-namespace-aware users"

This patch set is a first part of reworking SUNPRC PipeFS users.
It makes SUNRPC clients using PipeFS nofitications for directory and GSS pipes
dentries creation. With this patch set RPC clients and GSS auth creations
routines doesn't force SUNRPC PipeFS mount point creation which actually means,
that they now can work without PipeFS dentries.

The following series consists of:

Stanislav Kinsbursky (6):
SUNRPC: handle RPC client pipefs dentries by network namespace aware routines
SUNRPC: handle GSS AUTH pipes by network namespace aware routines
SUNRPC: subscribe RPC clients to pipefs notifications
SUNRPC: remove RPC client pipefs dentries after unregister
SUNRPC: remove RPC pipefs mount point manipulations from RPC clients code
SUNRPC: remove RPC PipeFS mount point reference from RPC client

```
fs/nfs/idmap.c                |   4 +
fs/nfsd/nfs4callback.c        |   2 -
include/linux/nfs.h           |   2 -
include/linux/sunrpc/auth.h   |   2 +
include/linux/sunrpc/clnt.h   |   2 -
net/sunrpc/auth_gss/auth_gss.c |  101 ++++++++++++++++++++++------
net/sunrpc/clnt.c             |  150 ++++++++++++++++++++++++++++++++++++++++---------
net/sunrpc/rpc_pipe.c         |   19 +++--
net/sunrpc/sunrpc.h           |   2 +
9 files changed, 217 insertions(+), 67 deletions(-)
```

Subject: [PATCH v2 1/6] SUNRPC: handle RPC client pipefs dentries by network namespace aware routines
Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:18:01 GMT
View Forum Message <> Reply to Message

v2:
1) "Over-put" of PipeFS mount point fixed. Fix is ugly, but allows to bisect
the patch set. And it will be removed later in the series.

This patch makes RPC clients PipeFs dentries allocations in it's owner network
namespace context.
RPC client pipefs dentries creation logic has been changed:
1) Pipefs dentries creation by sb was moved to separated function, which will
be used for handling PipeFS mount notification.
2) Initial value of RPC client PipeFS dir dentry is set no NULL now.

RPC client pipefs dentries cleanup logic has been changed:
1) Cleanup is done now in separated rpc_remove_pipedir() function, which takes
care about pipefs superblock locking.

Also this patch removes slashes from cb_program.pipe_dir_name and from
NFS_PIPE_DIRNAME to make rpc_d_lookup_sb() work. This doesn't affect
vfs_path_lookup() results in nfs4blocklayout_init() since this slash is cutted
off anyway in link_path_walk().

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

---
 fs/nfsd/nfs4callback.c | 2 -
 include/linux/nfs.h | 2 -
 net/sunrpc/clnt.c | 97 ++++++++++++++++++++++++++++++++----------------
 3 files changed, 66 insertions(+), 35 deletions(-)

diff --git a/fs/nfsd/nfs4callback.c b/fs/nfsd/nfs4callback.c
index 7748d6a..51526b7 100644
--- a/fs/nfsd/nfs4callback.c
+++ b/fs/nfsd/nfs4callback.c
@@ -622,7 +622,7 @@ static struct rpc_program cb_program = {
  .nrvers   = ARRAY_SIZE(nfs_cb_version),
  .version = nfs_cb_version,
  .stats   = &cb_stats,
- .pipe_dir_name = "/nfsd4_cb",
+ .pipe_dir_name = "nfsd4_cb",
 };

 static int max_cb_time(void)
diff --git a/include/linux/nfs.h b/include/linux/nfs.h
index 8c6ee44..6d1fb63 100644
--- a/include/linux/nfs.h

```
+++ b/include/linux/nfs.h
@@ -29,7 +29,7 @@
 #define NFS_MNT_VERSION  1
 #define NFS_MNT3_VERSION 3

-#define NFS_PIPE_DIRNAME "/nfs"
+#define NFS_PIPE_DIRNAME "nfs"

 /*
  * NFS stats. The good thing with these values is that NFSv3 errors are
diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index c5f04aa..90e82c5 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -96,52 +96,89 @@ static void rpc_unregister_client(struct rpc_clnt *clnt)
  spin_unlock(&sn->rpc_client_lock);
 }

-static int
-rpc_setup_pipedir(struct rpc_clnt *clnt, char *dir_name)
+static void __rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
+{
+ if (clnt->cl_path.dentry)
+  rpc_remove_client_dir(clnt->cl_path.dentry);
+ clnt->cl_path.dentry = NULL;
+}
+
+static void rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
+{
+ struct super_block *pipefs_sb;
+ int put_mnt = 0;
+
+ pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
+ if (pipefs_sb) {
+  if (clnt->cl_path.dentry)
+   put_mnt = 1;
+  __rpc_clnt_remove_pipedir(clnt);
+  rpc_put_sb_net(clnt->cl_xprt->xprt_net);
+ }
+ if (put_mnt)
+  rpc_put_mount();
+}
+
+static struct dentry *rpc_setup_pipedir_sb(struct super_block *sb,
+        struct rpc_clnt *clnt, char *dir_name)
 {
  static uint32_t clntid;
- struct path path, dir;
```

```
  char name[15];
  struct qstr q = {
   .name = name,
  };
+ struct dentry *dir, *dentry;
  int error;

- clnt->cl_path.mnt = ERR_PTR(-ENOENT);
- clnt->cl_path.dentry = ERR_PTR(-ENOENT);
- if (dir_name == NULL)
-  return 0;
-
- path.mnt = rpc_get_mount();
- if (IS_ERR(path.mnt))
-  return PTR_ERR(path.mnt);
- error = vfs_path_lookup(path.mnt->mnt_root, path.mnt, dir_name, 0, &dir);
- if (error)
-  goto err;
-
+ dir = rpc_d_lookup_sb(sb, dir_name);
+ if (dir == NULL)
+  return dir;
  for (;;) {
   q.len = snprintf(name, sizeof(name), "clnt%x", (unsigned int)clntid++);
   name[sizeof(name) - 1] = '\0';
   q.hash = full_name_hash(q.name, q.len);
-  path.dentry = rpc_create_client_dir(dir.dentry, &q, clnt);
-  if (!IS_ERR(path.dentry))
+  dentry = rpc_create_client_dir(dir, &q, clnt);
+  if (!IS_ERR(dentry))
    break;
-  error = PTR_ERR(path.dentry);
+  error = PTR_ERR(dentry);
   if (error != -EEXIST) {
    printk(KERN_INFO "RPC: Couldn't create pipefs entry"
      " %s/%s, error %d\n",
      dir_name, name, error);
-   goto err_path_put;
+   break;
   }
  }
- path_put(&dir);
+ dput(dir);
+ return dentry;
+}
+
+static int
+rpc_setup_pipedir(struct rpc_clnt *clnt, char *dir_name)
```

```
+{
+ struct super_block *pipefs_sb;
+ struct path path;
+
+ clnt->cl_path.mnt = ERR_PTR(-ENOENT);
+ clnt->cl_path.dentry = NULL;
+ if (dir_name == NULL)
+  return 0;
+
+ path.mnt = rpc_get_mount();
+ if (IS_ERR(path.mnt))
+  return PTR_ERR(path.mnt);
+ pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
+ if (!pipefs_sb) {
+  rpc_put_mount();
+  return -ENOENT;
+ }
+ path.dentry = rpc_setup_pipedir_sb(pipefs_sb, clnt, dir_name);
+ rpc_put_sb_net(clnt->cl_xprt->xprt_net);
+ if (IS_ERR(path.dentry)) {
+  rpc_put_mount();
+  return PTR_ERR(path.dentry);
+ }
  clnt->cl_path = path;
  return 0;
-err_path_put:
- path_put(&dir);
-err:
- rpc_put_mount();
- return error;
 }

 static struct rpc_clnt * rpc_new_client(const struct rpc_create_args *args, struct rpc_xprt *xprt)
@@ -249,10 +286,7 @@ static struct rpc_clnt * rpc_new_client(const struct rpc_create_args
*args, stru
  return clnt;

 out_no_auth:
- if (!IS_ERR(clnt->cl_path.dentry)) {
-  rpc_remove_client_dir(clnt->cl_path.dentry);
-  rpc_put_mount();
- }
+ rpc_clnt_remove_pipedir(clnt);
 out_no_path:
  kfree(clnt->cl_principal);
 out_no_principal:
@@ -477,10 +511,7 @@ rpc_free_client(struct rpc_clnt *clnt)
 {
```

```
  dprintk("RPC:       destroying %s client for %s\n",
     clnt->cl_protname, clnt->cl_server);
- if (!IS_ERR(clnt->cl_path.dentry)) {
-  rpc_remove_client_dir(clnt->cl_path.dentry);
-  rpc_put_mount();
- }
+ rpc_clnt_remove_pipedir(clnt);
  if (clnt->cl_parent != clnt) {
   rpc_release_client(clnt->cl_parent);
   goto out_free;
```

Subject: [PATCH v2 2/6] SUNRPC: handle GSS AUTH pipes by network namespace aware routines
Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:18:09 GMT
View Forum Message <> Reply to Message

This patch makes RPC GSS PipeFs pipes allocated in it's RPC client owner
network namespace context.
Pipes creation and destruction now done in separated functions, which takes
care about PipeFS superblock locking.

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

---
 net/sunrpc/auth_gss/auth_gss.c | 95 +++++++++++++++++++++++++++++++---------
 1 files changed, 73 insertions(+), 22 deletions(-)

diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 7544305..164193e 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -759,6 +759,73 @@ gss_pipe_destroy_msg(struct rpc_pipe_msg *msg)
  }
 }

+static void gss_pipes_dentries_destroy(struct rpc_auth *auth)
+{
+ struct gss_auth *gss_auth;
+
+ gss_auth = container_of(auth, struct gss_auth, rpc_auth);
+ rpc_unlink(gss_auth->pipe[0]->dentry);
+ rpc_unlink(gss_auth->pipe[1]->dentry);
+}
+
+static int gss_pipes_dentries_create(struct rpc_auth *auth)
+{
+ int err;

```
+ struct gss_auth *gss_auth;
+ struct rpc_clnt *clnt;
+
+ gss_auth = container_of(auth, struct gss_auth, rpc_auth);
+ clnt = gss_auth->client;
+
+ gss_auth->pipe[1]->dentry = rpc_mkpipe_dentry(clnt->cl_path.dentry,
+           "gssd",
+           clnt, gss_auth->pipe[1]);
+ if (IS_ERR(gss_auth->pipe[1]->dentry))
+  return PTR_ERR(gss_auth->pipe[1]->dentry);
+ gss_auth->pipe[0]->dentry = rpc_mkpipe_dentry(clnt->cl_path.dentry,
+           gss_auth->mech->gm_name,
+           clnt, gss_auth->pipe[0]);
+ if (IS_ERR(gss_auth->pipe[0]->dentry)) {
+  err = PTR_ERR(gss_auth->pipe[0]->dentry);
+  goto err_unlink_pipe_1;
+ }
+ return 0;
+
+err_unlink_pipe_1:
+ rpc_unlink(gss_auth->pipe[1]->dentry);
+ return err;
+}
+
+static void gss_pipes_dentries_destroy_net(struct rpc_clnt *clnt,
+       struct rpc_auth *auth)
+{
+ struct net *net = clnt->cl_xprt->xprt_net;
+ struct super_block *sb;
+
+ sb = rpc_get_sb_net(net);
+ if (sb) {
+  if (clnt->cl_path.dentry)
+   gss_pipes_dentries_destroy(auth);
+  rpc_put_sb_net(net);
+ }
+}
+
+static int gss_pipes_dentries_create_net(struct rpc_clnt *clnt,
+      struct rpc_auth *auth)
+{
+ struct net *net = clnt->cl_xprt->xprt_net;
+ struct super_block *sb;
+ int err = 0;
+
+ sb = rpc_get_sb_net(net);
+ if (sb) {
```

```
+  if (clnt->cl_path.dentry)
+   err = gss_pipes_dentries_create(auth);
+  rpc_put_sb_net(net);
+ }
+ return err;
+}
+
 /*
  * NOTE: we have the opportunity to use different
  * parameters based on the input flavor (which must be a pseudoflavor)
@@ -814,31 +881,16 @@ gss_create(struct rpc_clnt *clnt, rpc_authflavor_t flavor)
   err = PTR_ERR(gss_auth->pipe[0]);
   goto err_destroy_pipe_1;
  }
-
- gss_auth->pipe[1]->dentry = rpc_mkpipe_dentry(clnt->cl_path.dentry,
-           "gssd",
-           clnt, gss_auth->pipe[1]);
- if (IS_ERR(gss_auth->pipe[1]->dentry)) {
-  err = PTR_ERR(gss_auth->pipe[1]->dentry);
+ err = gss_pipes_dentries_create_net(clnt, auth);
+ if (err)
   goto err_destroy_pipe_0;
- }
-
- gss_auth->pipe[0]->dentry = rpc_mkpipe_dentry(clnt->cl_path.dentry,
-           gss_auth->mech->gm_name,
-           clnt, gss_auth->pipe[0]);
- if (IS_ERR(gss_auth->pipe[0]->dentry)) {
-  err = PTR_ERR(gss_auth->pipe[0]->dentry);
-  goto err_unlink_pipe_1;
- }
  err = rpcauth_init_credcache(auth);
  if (err)
-  goto err_unlink_pipe_0;
+  goto err_unlink_pipes;

  return auth;
-err_unlink_pipe_0:
- rpc_unlink(gss_auth->pipe[0]->dentry);
-err_unlink_pipe_1:
- rpc_unlink(gss_auth->pipe[1]->dentry);
+err_unlink_pipes:
+ gss_pipes_dentries_destroy_net(clnt, auth);
 err_destroy_pipe_0:
  rpc_destroy_pipe_data(gss_auth->pipe[0]);
 err_destroy_pipe_1:
@@ -855,8 +907,7 @@ out_dec:
```

```
 static void
 gss_free(struct gss_auth *gss_auth)
 {
- rpc_unlink(gss_auth->pipe[0]->dentry);
- rpc_unlink(gss_auth->pipe[1]->dentry);
+ gss_pipes_dentries_destroy_net(gss_auth->client, &gss_auth->rpc_auth);
  rpc_destroy_pipe_data(gss_auth->pipe[0]);
  rpc_destroy_pipe_data(gss_auth->pipe[1]);
  gss_mech_put(gss_auth->mech);
```

Subject: [PATCH v2 3/6] SUNRPC: subscribe RPC clients to pipefs notifications
Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:18:17 GMT
View Forum Message <> Reply to Message

This patch subscribes RPC clients to RPC pipefs notifications. RPC clients
notifier block is registering with pipefs initialization during SUNRPC module
init.
This notifier callback is responsible for RPC client PipeFS directory and GSS
pipes creation. For pipes creation and destruction two additional callbacks
were added to struct rpc_authops.
Note that no locking required in notifier callback because PipeFS superblock
pointer is passed as an argument from it's creation or destruction routine and
thus we can be sure about it's validity.

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

```
---
 include/linux/sunrpc/auth.h    |    2 +
 net/sunrpc/auth_gss/auth_gss.c |   10 ++++--
 net/sunrpc/clnt.c              |   69 ++++++++++++++++++++++++++++++++++++++++-
 net/sunrpc/rpc_pipe.c          |   19 ++++++---
 net/sunrpc/sunrpc.h            |    2 +
 5 files changed, 92 insertions(+), 10 deletions(-)

diff --git a/include/linux/sunrpc/auth.h b/include/linux/sunrpc/auth.h
index febc4db..83f493f 100644
--- a/include/linux/sunrpc/auth.h
+++ b/include/linux/sunrpc/auth.h
@@ -98,6 +98,8 @@ struct rpc_authops {

  struct rpc_cred * (*lookup_cred)(struct rpc_auth *, struct auth_cred *, int);
  struct rpc_cred * (*crcreate)(struct rpc_auth*, struct auth_cred *, int);
+ int   (*pipes_create)(struct rpc_auth *);
+ void   (*pipes_destroy)(struct rpc_auth *);
 };

 struct rpc_credops {
```

```
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 164193e..d7a5719 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -764,8 +764,10 @@ static void gss_pipes_dentries_destroy(struct rpc_auth *auth)
  struct gss_auth *gss_auth;

  gss_auth = container_of(auth, struct gss_auth, rpc_auth);
- rpc_unlink(gss_auth->pipe[0]->dentry);
- rpc_unlink(gss_auth->pipe[1]->dentry);
+ if (gss_auth->pipe[0]->dentry)
+  rpc_unlink(gss_auth->pipe[0]->dentry);
+ if (gss_auth->pipe[1]->dentry)
+  rpc_unlink(gss_auth->pipe[1]->dentry);
 }

 static int gss_pipes_dentries_create(struct rpc_auth *auth)
@@ -1608,7 +1610,9 @@ static const struct rpc_authops authgss_ops = {
  .create  = gss_create,
  .destroy = gss_destroy,
  .lookup_cred = gss_lookup_cred,
- .crcreate = gss_create_cred
+ .crcreate = gss_create_cred,
+ .pipes_create = gss_pipes_dentries_create,
+ .pipes_destroy = gss_pipes_dentries_destroy,
 };

 static const struct rpc_credops gss_credops = {
diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index 90e82c5..4170745 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -98,8 +98,11 @@ static void rpc_unregister_client(struct rpc_clnt *clnt)

 static void __rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
 {
- if (clnt->cl_path.dentry)
+ if (clnt->cl_path.dentry) {
+  if (clnt->cl_auth && clnt->cl_auth->au_ops->pipes_destroy)
+   clnt->cl_auth->au_ops->pipes_destroy(clnt->cl_auth);
   rpc_remove_client_dir(clnt->cl_path.dentry);
+ }
  clnt->cl_path.dentry = NULL;
 }

@@ -181,6 +184,70 @@ rpc_setup_pipedir(struct rpc_clnt *clnt, char *dir_name)
  return 0;
 }
```

```
+static int __rpc_pipefs_event(struct rpc_clnt *clnt, unsigned long event,
+    struct super_block *sb)
+{
+ struct dentry *dentry;
+ int err = 0;
+
+ switch (event) {
+ case RPC_PIPEFS_MOUNT:
+  if (clnt->cl_program->pipe_dir_name == NULL)
+   break;
+  dentry = rpc_setup_pipedir_sb(sb, clnt,
+          clnt->cl_program->pipe_dir_name);
+  BUG_ON(dentry == NULL);
+  if (IS_ERR(dentry))
+   return PTR_ERR(dentry);
+  clnt->cl_path.dentry = dentry;
+  if (clnt->cl_auth->au_ops->pipes_create) {
+   err = clnt->cl_auth->au_ops->pipes_create(clnt->cl_auth);
+   if (err)
+    __rpc_clnt_remove_pipedir(clnt);
+  }
+  break;
+ case RPC_PIPEFS_UMOUNT:
+  __rpc_clnt_remove_pipedir(clnt);
+  break;
+ default:
+  printk(KERN_ERR "%s: unknown event: %ld\n", __func__, event);
+  return -ENOTSUPP;
+ }
+ return err;
+}
+
+static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
+       void *ptr)
+{
+ struct super_block *sb = ptr;
+ struct rpc_clnt *clnt;
+ int error = 0;
+ struct sunrpc_net *sn = net_generic(sb->s_fs_info, sunrpc_net_id);
+
+ spin_lock(&sn->rpc_client_lock);
+ list_for_each_entry(clnt, &sn->all_clients, cl_clients) {
+  error = __rpc_pipefs_event(clnt, event, sb);
+  if (error)
+   break;
+ }
+ spin_unlock(&sn->rpc_client_lock);
```

```
+ return error;
+}
+
+static struct notifier_block rpc_clients_block = {
+ .notifier_call = rpc_pipefs_event,
+};
+
+int rpc_clients_notifier_register(void)
+{
+ return rpc_pipefs_notifier_register(&rpc_clients_block);
+}
+
+void rpc_clients_notifier_unregister(void)
+{
+ return rpc_pipefs_notifier_unregister(&rpc_clients_block);
+}
+
 static struct rpc_clnt * rpc_new_client(const struct rpc_create_args *args, struct rpc_xprt *xprt)
 {
  struct rpc_program *program = args->program;
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index d3a0d26..8d74575 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -938,7 +938,7 @@ struct dentry *rpc_create_client_dir(struct dentry *dentry,

 /**
  * rpc_remove_client_dir - Remove a directory created with rpc_create_client_dir()
- * @dentry: directory to remove
+ * @clnt: rpc client
  */
 int rpc_remove_client_dir(struct dentry *dentry)
 {
@@ -1189,17 +1189,24 @@ int register_rpc_pipefs(void)
     init_once);
  if (!rpc_inode_cachep)
   return -ENOMEM;
+ err = rpc_clients_notifier_register();
+ if (err)
+  goto err_notifier;
  err = register_filesystem(&rpc_pipe_fs_type);
- if (err) {
-  kmem_cache_destroy(rpc_inode_cachep);
-  return err;
- }
-
+ if (err)
+  goto err_register;
```

```
  return 0;
+
+err_register:
+ rpc_clients_notifier_unregister();
+err_notifier:
+ kmem_cache_destroy(rpc_inode_cachep);
+ return err;
 }

 void unregister_rpc_pipefs(void)
 {
+ rpc_clients_notifier_unregister();
  kmem_cache_destroy(rpc_inode_cachep);
  unregister_filesystem(&rpc_pipe_fs_type);
 }
diff --git a/net/sunrpc/sunrpc.h b/net/sunrpc/sunrpc.h
index 90c292e..14c9f6d 100644
--- a/net/sunrpc/sunrpc.h
+++ b/net/sunrpc/sunrpc.h
@@ -47,5 +47,7 @@ int svc_send_common(struct socket *sock, struct xdr_buf *xdr,
       struct page *headpage, unsigned long headoffset,
       struct page *tailpage, unsigned long tailoffset);

+int rpc_clients_notifier_register(void);
+void rpc_clients_notifier_unregister(void);
 #endif /* _NET_SUNRPC_SUNRPC_H */
```

Subject: [PATCH v2 4/6] SUNRPC: remove RPC client pipefs dentries after unregister

Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:18:26 GMT

View Forum Message <> Reply to Message

Without this patch we have races:

```
rpc_fill_super    rpc_free_client
rpc_pipefs_event(MOUNT)   rpc_remove_pipedir
spin_lock(&rpc_client_lock);
rpc_setup_pipedir_sb
spin_unlock(&rpc_client_lock);
    spin_lock(&rpc_client_lock);
    (remove from list)
    spin_unlock(&rpc_client_lock);
    MEAMORY LEAKED
```

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

```
 net/sunrpc/clnt.c |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index 4170745..c89ceb8 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -578,7 +578,6 @@ rpc_free_client(struct rpc_clnt *clnt)
 {
  dprintk("RPC:       destroying %s client for %s\n",
    clnt->cl_protname, clnt->cl_server);
- rpc_clnt_remove_pipedir(clnt);
  if (clnt->cl_parent != clnt) {
   rpc_release_client(clnt->cl_parent);
   goto out_free;
@@ -587,6 +586,7 @@ rpc_free_client(struct rpc_clnt *clnt)
   kfree(clnt->cl_server);
 out_free:
  rpc_unregister_client(clnt);
+ rpc_clnt_remove_pipedir(clnt);
  rpc_free_iostats(clnt->cl_metrics);
  kfree(clnt->cl_principal);
  clnt->cl_metrics = NULL;
```

Subject: [PATCH v2 5/6] SUNRPC: remove RPC pipefs mount point manipulations from RPC clients code

Posted by Stanislav Kinsbursky on Wed, 11 Jan 2012 15:18:34 GMT

View Forum Message <> Reply to Message

v2:
1) Updated due to changes in the first patch of the series.

Now, with RPC pipefs mount notifications handling in RPC clients, we can remove
mount point creation and destruction. RPC clients dentries will be created on
PipeFS mount event and removed on umount event.

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

```
---
 net/sunrpc/clnt.c |   19 +++----------------
 1 files changed, 3 insertions(+), 16 deletions(-)

diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index c89ceb8..e3ced30 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -109,17 +109,12 @@ static void __rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
```

```
 static void rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
 {
  struct super_block *pipefs_sb;
- int put_mnt = 0;

  pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
  if (pipefs_sb) {
-  if (clnt->cl_path.dentry)
-   put_mnt = 1;
   __rpc_clnt_remove_pipedir(clnt);
   rpc_put_sb_net(clnt->cl_xprt->xprt_net);
  }
- if (put_mnt)
-  rpc_put_mount();
 }

 static struct dentry *rpc_setup_pipedir_sb(struct super_block *sb,
@@ -165,21 +160,13 @@ rpc_setup_pipedir(struct rpc_clnt *clnt, char *dir_name)
  clnt->cl_path.dentry = NULL;
  if (dir_name == NULL)
   return 0;
-
- path.mnt = rpc_get_mount();
- if (IS_ERR(path.mnt))
-  return PTR_ERR(path.mnt);
  pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
- if (!pipefs_sb) {
-  rpc_put_mount();
-  return -ENOENT;
- }
+ if (!pipefs_sb)
+  return 0;
  path.dentry = rpc_setup_pipedir_sb(pipefs_sb, clnt, dir_name);
  rpc_put_sb_net(clnt->cl_xprt->xprt_net);
- if (IS_ERR(path.dentry)) {
-  rpc_put_mount();
+ if (IS_ERR(path.dentry))
   return PTR_ERR(path.dentry);
- }
  clnt->cl_path = path;
  return 0;
 }
```