
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;
}
```
