
Subject: [PATCH v2 6/6] SUNRPC: split SUNRPC PipeFS dentry and private pipe data creation

Posted by [Stanislav Kinsbursky](#) on Mon, 26 Dec 2011 11:46:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch is a final step towards removing PipeFS inode references from kernel code other than PipeFS itself. It makes all kernel SUNRPC PipeFS users depends on pipe private data, which state depend on their specific operations, etc.

This patch completes SUNRPC PipeFS preparations and allows to create pipe private data and PipeFS dentries independently.

Next step will be making SUNRPC PipeFS dentries allocated by SUNRPC PipeFS network namespace aware routines.

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

```
---
fs/nfs/blocklayout/blocklayout.c | 16 ++++++---
fs/nfs/blocklayout/blocklayout.h | 2 +
fs/nfs/blocklayout/blocklayoutdev.c | 2 +
fs/nfs/blocklayout/blocklayoutdm.c | 2 +
fs/nfs/idmap.c | 28 ++++++++-----
include/linux/sunrpc/rpc_pipe_fs.h | 7 +++-
net/sunrpc/auth_gss/auth_gss.c | 54 ++++++++-----
net/sunrpc/rpc_pipe.c | 54 ++++++++-----
8 files changed, 107 insertions(+), 58 deletions(-)
```

```
diff --git a/fs/nfs/blocklayout/blocklayout.c b/fs/nfs/blocklayout/blocklayout.c
```

```
index 281ae95..2f4ede1 100644
```

```
--- a/fs/nfs/blocklayout/blocklayout.c
```

```
+++ b/fs/nfs/blocklayout/blocklayout.c
```

```
@@ -46,7 +46,7 @@ MODULE_LICENSE("GPL");
```

```
MODULE_AUTHOR("Andy Adamson <andros@citi.umich.edu>");
```

```
MODULE_DESCRIPTION("The NFSv4.1 pNFS Block layout driver");
```

```
-struct dentry *bl_device_pipe;
```

```
+struct rpc_pipe *bl_device_pipe;
```

```
wait_queue_head_t bl_wq;
```

```
static void print_page(struct page *page)
```

```
@@ -991,16 +991,23 @@ static int __init nfs4blocklayout_init(void)
```

```
if (ret)
```

```
goto out_putrpc;
```

```
- bl_device_pipe = rpc_mkpipe(path.dentry, "blocklayout", NULL,
```

```
- &bl_upcall_ops, 0);
```

```
+ bl_device_pipe = rpc_mkpipe_data(&bl_upcall_ops, 0);
```

```
path_put(&path);
```

```

    if (IS_ERR(bl_device_pipe)) {
        ret = PTR_ERR(bl_device_pipe);
        goto out_putrpc;
    }
+ bl_device_pipe->dentry = rpc_mkpipe_dentry(path.dentry, "blocklayout",
+     NULL, bl_device_pipe);
+ if (IS_ERR(bl_device_pipe->dentry)) {
+     ret = PTR_ERR(bl_device_pipe->dentry);
+     goto out_destroy_pipe;
+ }
out:
    return ret;

+out_destroy_pipe:
+ rpc_destroy_pipe_data(bl_device_pipe);
out_putrpc:
    rpc_put_mount();
out_remove:
@@ -1014,7 +1021,8 @@ static void __exit nfs4blocklayout_exit(void)
    __func__);

    pnfs_unregister_layoutdriver(&blocklayout_type);
- rpc_unlink(bl_device_pipe);
+ rpc_unlink(bl_device_pipe->dentry);
+ rpc_destroy_pipe_data(bl_device_pipe);
    rpc_put_mount();
}

diff --git a/fs/nfs/blocklayout/blocklayout.h b/fs/nfs/blocklayout/blocklayout.h
index 42acf7e..046b513 100644
--- a/fs/nfs/blocklayout/blocklayout.h
+++ b/fs/nfs/blocklayout/blocklayout.h
@@ -159,7 +159,7 @@ struct bl_msg_hdr {
    u16 totalen; /* length of entire message, including hdr itself */
};

-extern struct dentry *bl_device_pipe;
+extern struct rpc_pipe *bl_device_pipe;
extern wait_queue_head_t bl_wq;

#define BL_DEVICE_UMOUNT          0x0 /* Umount--delete devices */
diff --git a/fs/nfs/blocklayout/blocklayoutdev.c b/fs/nfs/blocklayout/blocklayoutdev.c
index 8101919..949b624 100644
--- a/fs/nfs/blocklayout/blocklayoutdev.c
+++ b/fs/nfs/blocklayout/blocklayoutdev.c
@@ -146,7 +146,7 @@ nfs4_blk_decode_device(struct nfs_server *server,

    dprintk("%s CALLING USERSPACE DAEMON\n", __func__);

```

```

    add_wait_queue(&bl_wq, &wq);
- rc = rpc_queue_upcall(RPC_I(bl_device_pipe->d_inode)->pipe, &msg);
+ rc = rpc_queue_upcall(bl_device_pipe, &msg);
    if (rc < 0) {
        remove_wait_queue(&bl_wq, &wq);
        rv = ERR_PTR(rc);
diff --git a/fs/nfs/blocklayout/blocklayoutdm.c b/fs/nfs/blocklayout/blocklayoutdm.c
index 3c38244..631f254 100644
--- a/fs/nfs/blocklayout/blocklayoutdm.c
+++ b/fs/nfs/blocklayout/blocklayoutdm.c
@@ -66,7 +66,7 @@ static void dev_remove(dev_t dev)
    msg.len = sizeof(bl_msg) + bl_msg.totallen;

    add_wait_queue(&bl_wq, &wq);
- if (rpc_queue_upcall(RPC_I(bl_device_pipe->d_inode)->pipe, &msg) < 0) {
+ if (rpc_queue_upcall(bl_device_pipe, &msg) < 0) {
    remove_wait_queue(&bl_wq, &wq);
    goto out;
}
diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index 4194129..096e374 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -327,7 +327,7 @@ struct idmap_hashtable {
};

struct idmap {
- struct dentry *idmap_dentry;
+ struct rpc_pipe *idmap_pipe;
    wait_queue_head_t idmap_wq;
    struct idmap_msg idmap_im;
    struct mutex idmap_lock; /* Serializes upcalls */
@@ -352,6 +352,6 @@ int
nfs_idmap_new(struct nfs_client *clp)
{
    struct idmap *idmap;
+ struct rpc_pipe *pipe;
    int error;

    BUG_ON(clp->cl_idmap != NULL);
@@ -360,14 +361,23 @@ nfs_idmap_new(struct nfs_client *clp)
    if (idmap == NULL)
        return -ENOMEM;

- idmap->idmap_dentry = rpc_mkpipe(clp->cl_rpcclient->cl_path.dentry,
- "idmap", idmap, &idmap_upcall_ops, 0);
- if (IS_ERR(idmap->idmap_dentry)) {
-     error = PTR_ERR(idmap->idmap_dentry);

```

```

+ pipe = rpc_mkpipe_data(&idmap_upcall_ops, 0);
+ if (IS_ERR(pipe)) {
+ error = PTR_ERR(pipe);
+ kfree(idmap);
+ return error;
+ }

+ if (clp->cl_rpcclient->cl_path.dentry)
+ pipe->dentry = rpc_mkpipe_dentry(clp->cl_rpcclient->cl_path.dentry,
+ "idmap", idmap, pipe);
+ if (IS_ERR(pipe->dentry)) {
+ error = PTR_ERR(pipe->dentry);
+ rpc_destroy_pipe_data(pipe);
+ kfree(idmap);
+ return error;
+ }
+ idmap->idmap_pipe = pipe;
+ mutex_init(&idmap->idmap_lock);
+ mutex_init(&idmap->idmap_im_lock);
+ init_waitqueue_head(&idmap->idmap_wq);
@@ -385,7 +395,9 @@ nfs_idmap_delete(struct nfs_client *clp)

+ if (!idmap)
+ return;
- rpc_unlink(idmap->idmap_dentry);
+ if (idmap->idmap_pipe->dentry)
+ rpc_unlink(idmap->idmap_pipe->dentry);
+ rpc_destroy_pipe_data(idmap->idmap_pipe);
+ clp->cl_idmap = NULL;
+ kfree(idmap);
+ }
@@ -506,7 +518,7 @@ nfs_idmap_id(struct idmap *idmap, struct idmap_hashtable *h,
+ msg.len = sizeof(*im);

+ add_wait_queue(&idmap->idmap_wq, &wq);
- if (rpc_queue_upcall(RPC_I(idmap->idmap_dentry->d_inode)->pipe, &msg) < 0) {
+ if (rpc_queue_upcall(idmap->idmap_pipe, &msg) < 0) {
+ remove_wait_queue(&idmap->idmap_wq, &wq);
+ goto out;
+ }
@@ -567,7 +579,7 @@ nfs_idmap_name(struct idmap *idmap, struct idmap_hashtable *h,

+ add_wait_queue(&idmap->idmap_wq, &wq);

- if (rpc_queue_upcall(RPC_I(idmap->idmap_dentry->d_inode)->pipe, &msg) < 0) {
+ if (rpc_queue_upcall(idmap->idmap_pipe, &msg) < 0) {
+ remove_wait_queue(&idmap->idmap_wq, &wq);
+ goto out;

```

```

}
diff --git a/include/linux/sunrpc/rpc_pipe_fs.h b/include/linux/sunrpc/rpc_pipe_fs.h
index ad78bea..0808ed2 100644
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -34,6 +34,7 @@ struct rpc_pipe {
    struct delayed_work queue_timeout;
    const struct rpc_pipe_ops *ops;
    spinlock_t lock;
+ struct dentry *dentry;
};

struct rpc_inode {
@@ -77,8 +78,10 @@ extern struct dentry *rpc_create_cache_dir(struct dentry *,
    struct cache_detail *);
extern void rpc_remove_cache_dir(struct dentry *);

-extern struct dentry *rpc_mkpipe(struct dentry *, const char *, void *,
-    const struct rpc_pipe_ops *, int flags);
+struct rpc_pipe *rpc_mkpipe_data(const struct rpc_pipe_ops *ops, int flags);
+void rpc_destroy_pipe_data(struct rpc_pipe *pipe);
+extern struct dentry *rpc_mkpipe_dentry(struct dentry *, const char *, void *,
+    struct rpc_pipe *);
extern int rpc_unlink(struct dentry *);
extern struct vfsmount *rpc_get_mount(void);
extern void rpc_put_mount(void);
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index f99fde5..7544305 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -81,7 +81,7 @@ struct gss_auth {
    * mechanism (for example, "krb5") and exists for
    * backwards-compatibility with older gssd's.
    */
- struct dentry *dentry[2];
+ struct rpc_pipe *pipe[2];
};

/* pipe_version >= 0 if and only if someone has a pipe open. */
@@ -451,7 +451,7 @@ gss_alloc_msg(struct gss_auth *gss_auth, uid_t uid, struct rpc_clnt *clnt,
    kfree(gss_msg);
    return ERR_PTR(vers);
}
- gss_msg->pipe = RPC_I(gss_auth->dentry[vers]->d_inode)->pipe;
+ gss_msg->pipe = gss_auth->pipe[vers];
INIT_LIST_HEAD(&gss_msg->list);
rpc_init_wait_queue(&gss_msg->rpc_waitqueue, "RPCSEC_GSS upcall waitq");
init_waitqueue_head(&gss_msg->waitqueue);

```

```

@@ -801,21 +801,33 @@ gss_create(struct rpc_clnt *clnt, rpc_authflavor_t flavor)
    * that we supported only the old pipe. So we instead create
    * the new pipe first.
    */
- gss_auth->dentry[1] = rpc_mkpipe(clnt->cl_path.dentry,
-   "gssd",
-   clnt, &gss_upcall_ops_v1,
-   RPC_PIPE_WAIT_FOR_OPEN);
- if (IS_ERR(gss_auth->dentry[1])) {
-   err = PTR_ERR(gss_auth->dentry[1]);
+ gss_auth->pipe[1] = rpc_mkpipe_data(&gss_upcall_ops_v1,
+   RPC_PIPE_WAIT_FOR_OPEN);
+ if (IS_ERR(gss_auth->pipe[1])) {
+   err = PTR_ERR(gss_auth->pipe[1]);
+   goto err_put_mech;
+ }

- gss_auth->dentry[0] = rpc_mkpipe(clnt->cl_path.dentry,
-   gss_auth->mech->gm_name,
-   clnt, &gss_upcall_ops_v0,
-   RPC_PIPE_WAIT_FOR_OPEN);
- if (IS_ERR(gss_auth->dentry[0])) {
-   err = PTR_ERR(gss_auth->dentry[0]);
+ gss_auth->pipe[0] = rpc_mkpipe_data(&gss_upcall_ops_v0,
+   RPC_PIPE_WAIT_FOR_OPEN);
+ if (IS_ERR(gss_auth->pipe[0])) {
+   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);
+   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);
@@ -824,9 +836,13 @@ gss_create(struct rpc_clnt *clnt, rpc_authflavor_t flavor)

```

```

    return auth;
err_unlink_pipe_0:
- rpc_unlink(gss_auth->dentry[0]);
+ rpc_unlink(gss_auth->pipe[0]->dentry);
err_unlink_pipe_1:
- rpc_unlink(gss_auth->dentry[1]);
+ rpc_unlink(gss_auth->pipe[1]->dentry);
+err_destroy_pipe_0:
+ rpc_destroy_pipe_data(gss_auth->pipe[0]);
+err_destroy_pipe_1:
+ rpc_destroy_pipe_data(gss_auth->pipe[1]);
err_put_mech:
    gss_mech_put(gss_auth->mech);
err_free:
@@ -839,8 +855,10 @@ out_dec:
static void
gss_free(struct gss_auth *gss_auth)
{
- rpc_unlink(gss_auth->dentry[1]);
- rpc_unlink(gss_auth->dentry[0]);
+ rpc_unlink(gss_auth->pipe[0]->dentry);
+ rpc_unlink(gss_auth->pipe[1]->dentry);
+ rpc_destroy_pipe_data(gss_auth->pipe[0]);
+ rpc_destroy_pipe_data(gss_auth->pipe[1]);
    gss_mech_put(gss_auth->mech);

    kfree(gss_auth);
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index dc24af3..d3a0d26 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -207,7 +207,6 @@ rpc_i_callback(struct rcu_head *head)
{
    struct inode *inode = container_of(head, struct inode, i_rcu);
    INIT_LIST_HEAD(&inode->i_dentry);
- kfree(RPC_I(inode)->pipe);
    kmem_cache_free(rpc_inode_cachep, RPC_I(inode));
}

@@ -576,34 +575,44 @@ init_pipe(struct rpc_pipe *pipe)
    rpc_timeout_upcall_queue);
    pipe->ops = NULL;
    spin_lock_init(&pipe->lock);
+ pipe->dentry = NULL;
+}

+void rpc_destroy_pipe_data(struct rpc_pipe *pipe)
+{

```

```

+ kfree(pipe);
}
+EXPORT_SYMBOL_GPL(rpc_destroy_pipe_data);

-static int __rpc_mkpipe(struct inode *dir, struct dentry *dentry,
- umode_t mode,
- const struct file_operations *i_fop,
- void *private,
- const struct rpc_pipe_ops *ops,
- int flags)
+struct rpc_pipe *rpc_mkpipe_data(const struct rpc_pipe_ops *ops, int flags)
{
    struct rpc_pipe *pipe;
- struct rpc_inode *rpci;
- int err;

    pipe = kzalloc(sizeof(struct rpc_pipe), GFP_KERNEL);
    if (!pipe)
- return -ENOMEM;
+ return ERR_PTR(-ENOMEM);
    init_pipe(pipe);
+ pipe->ops = ops;
+ pipe->flags = flags;
+ return pipe;
+}
+EXPORT_SYMBOL_GPL(rpc_mkpipe_data);
+
+static int __rpc_mkpipe_dentry(struct inode *dir, struct dentry *dentry,
+ umode_t mode,
+ const struct file_operations *i_fop,
+ void *private,
+ struct rpc_pipe *pipe)
+{
+ struct rpc_inode *rpci;
+ int err;
+
    err = __rpc_create_common(dir, dentry, S_IFIFO | mode, i_fop, private);
- if (err) {
- kfree(pipe);
+ if (err)
    return err;
- }
    rpci = RPC_I(dentry->d_inode);
    rpci->private = private;
    rpci->pipe = pipe;
- rpci->pipe->flags = flags;
- rpci->pipe->ops = ops;
    fsnotify_create(dir, dentry);

```



```

    return 0;
}
@@ -820,9 +829,8 @@ static int rpc_rmdir_deppopulate(struct dentry *dentry,
 * The @private argument passed here will be available to all these methods
 * from the file pointer, via RPC_I(file->f_dentry->d_inode)->private.
 */
-struct dentry *rpc_mkpipe(struct dentry *parent, const char *name,
-    void *private, const struct rpc_pipe_ops *ops,
-    int flags)
+struct dentry *rpc_mkpipe_dentry(struct dentry *parent, const char *name,
+    void *private, struct rpc_pipe *pipe)
{
    struct dentry *dentry;
    struct inode *dir = parent->d_inode;
@@ -830,9 +838,9 @@ struct dentry *rpc_mkpipe(struct dentry *parent, const char *name,
    struct qstr q;
    int err;

- if (ops->upcall == NULL)
+ if (pipe->ops->upcall == NULL)
     umode &= ~S_IRUGO;
- if (ops->downcall == NULL)
+ if (pipe->ops->downcall == NULL)
     umode &= ~S_IWUGO;

    q.name = name;
@@ -843,8 +851,8 @@ struct dentry *rpc_mkpipe(struct dentry *parent, const char *name,
    dentry = __rpc_lookup_create_exclusive(parent, &q);
    if (IS_ERR(dentry))
        goto out;
- err = __rpc_mkpipe(dir, dentry, umode, &rpc_pipe_fops,
-    private, ops, flags);
+ err = __rpc_mkpipe_dentry(dir, dentry, umode, &rpc_pipe_fops,
+    private, pipe);
    if (err)
        goto out_err;
out:
@@ -857,7 +865,7 @@ out_err:
    err);
    goto out;
}
-EXPORT_SYMBOL_GPL(rpc_mkpipe);
+EXPORT_SYMBOL_GPL(rpc_mkpipe_dentry);

/**
 * rpc_unlink - remove a pipe

```
