

---

Subject: [PATCH v2 1/6] SUNRPC: fix pipe->ops cleanup on pipe dentry unlink  
Posted by Stanislav Kinsbursky on Tue, 10 Jan 2012 12:12:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

This patch looks late due to GSS AUTH patches sent already. But it fixes a flaw in RPC PipeFS pipes handling.

I've added this patch in the series, because this series related to pipes. But it should be a part of previous series named "SUNRPC: cleanup PipeFS for network-namespace-aware users".

Pipe dentry can be created and destroyed many times during pipe life cycle. This actually means, that we can't set pipe->ops to NULL in rpc\_close\_pipes() and use this variable as a flag, indicating, that pipe's dentry is unlinking.

To follow this restriction, this patch replaces "pipe->ops = NULL" assignment and checks for NULL with "pipe->dentry = NULL" assignment and checks for NULL respectively.

This patch also removes check for non-NUL pipe->ops (or pipe->dentry) in rpc\_close\_pipes() because it always non-NUL now.

Signed-off-by: Stanislav Kinsbursky <[skinsbursky@parallels.com](mailto:skinsbursky@parallels.com)>

---

net/sunrpc/rpc\_pipe.c | 51 ++++++-----  
1 files changed, 20 insertions(+), 31 deletions(-)

```
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 8d74575..6415241 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -86,10 +86,6 @@ rpc_timeout_upcall_queue(struct work_struct *work)
 void (*destroy_msg)(struct rpc_pipe_msg *);

spin_lock(&pipe->lock);
- if (pipe->ops == NULL) {
- spin_unlock(&pipe->lock);
- return;
- }
destroy_msg = pipe->ops->destroy_msg;
if (pipe->nreaders == 0) {
list_splice_init(&pipe->pipe, &free_list);
@@ -135,8 +131,6 @@ rpc_queue_upcall(struct rpc_pipe *pipe, struct rpc_pipe_msg *msg)
int res = -EPIPE;

spin_lock(&pipe->lock);
- if (pipe->ops == NULL)
- goto out;
if (pipe->nreaders) {
list_add_tail(&msg->list, &pipe->pipe);
```

```

pipe->pipelen += msg->len;
@@ -150,7 +144,6 @@ rpc_queue_upcall(struct rpc_pipe *pipe, struct rpc_pipe_msg *msg)
    pipe->pipelen += msg->len;
    res = 0;
}
-out:
    spin_unlock(&pipe->lock);
    wake_up(&pipe->waitq);
    return res;
@@ -167,27 +160,23 @@ static void
rpc_close_pipes(struct inode *inode)
{
    struct rpc_pipe *pipe = RPC_I(inode)->pipe;
- const struct rpc_pipe_ops *ops;
    int need_release;
+ LIST_HEAD(free_list);

    mutex_lock(&inode->i_mutex);
- ops = pipe->ops;
- if (ops != NULL) {
-     LIST_HEAD(free_list);
-     spin_lock(&pipe->lock);
-     need_release = pipe->nreaders != 0 || pipe->nwriters != 0;
-     pipe->nreaders = 0;
-     list_splice_init(&pipe->in_upcall, &free_list);
-     list_splice_init(&pipe->pipe, &free_list);
-     pipe->pipelen = 0;
-     pipe->ops = NULL;
-     spin_unlock(&pipe->lock);
-     rpc_purge_list(pipe, &free_list, ops->destroy_msg, -EPIPE);
-     pipe->nwriters = 0;
-     if (need_release && ops->release_pipe)
-         ops->release_pipe(inode);
-     cancel_delayed_work_sync(&pipe->queue_timeout);
- }
+ spin_lock(&pipe->lock);
+ need_release = pipe->nreaders != 0 || pipe->nwriters != 0;
+ pipe->nreaders = 0;
+ list_splice_init(&pipe->in_upcall, &free_list);
+ list_splice_init(&pipe->pipe, &free_list);
+ pipe->pipelen = 0;
+ pipe->dentry = NULL;
+ spin_unlock(&pipe->lock);
+ rpc_purge_list(pipe, &free_list, pipe->ops->destroy_msg, -EPIPE);
+ pipe->nwriters = 0;
+ if (need_release && pipe->ops->release_pipe)
+     pipe->ops->release_pipe(inode);
+ cancel_delayed_work_sync(&pipe->queue_timeout);

```

```

rpc_inode_setowner(inode, NULL);
mutex_unlock(&inode->i_mutex);
}
@@ -224,7 +213,7 @@ rpc_pipe_open(struct inode *inode, struct file *filp)
int res = -ENXIO;

mutex_lock(&inode->i_mutex);
- if (pipe->ops == NULL)
+ if (pipe->dentry == NULL)
    goto out;
first_open = pipe->nreaders == 0 && pipe->nwriters == 0;
if (first_open && pipe->ops->open_pipe) {
@@ -250,7 +239,7 @@ rpc_pipe_release(struct inode *inode, struct file *filp)
int last_close;

mutex_lock(&inode->i_mutex);
- if (pipe->ops == NULL)
+ if (pipe->dentry == NULL)
    goto out;
msg = filp->private_data;
if (msg != NULL) {
@@ -291,7 +280,7 @@ rpc_pipe_read(struct file *filp, char __user *buf, size_t len, loff_t *offset)
int res = 0;

mutex_lock(&inode->i_mutex);
- if (pipe->ops == NULL) {
+ if (pipe->dentry == NULL) {
    res = -EPIPE;
    goto out_unlock;
}
@@ -334,7 +323,7 @@ rpc_pipe_write(struct file *filp, const char __user *buf, size_t len, loff_t *of

mutex_lock(&inode->i_mutex);
res = -EPIPE;
- if (pipe->ops != NULL)
+ if (pipe->dentry != NULL)
    res = pipe->ops->downcall(filp, buf, len);
mutex_unlock(&inode->i_mutex);
return res;
@@ -349,7 +338,7 @@ rpc_pipe_poll(struct file *filp, struct poll_table_struct *wait)
poll_wait(filp, &pipe->waitq, wait);

mask = POLLOUT | POLLWRNORM;
- if (pipe->ops == NULL)
+ if (pipe->dentry == NULL)
    mask |= POLLERR | POLLHUP;
if (filp->private_data || !list_empty(&pipe->pipe))
    mask |= POLLIN | POLLRDNORM;

```

```
@@ -366,7 +355,7 @@ @@ rpc_pipe_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
switch (cmd) {
case FIONREAD:
    spin_lock(&pipe->lock);
- if (pipe->ops == NULL) {
+ if (pipe->dentry == NULL) {
    spin_unlock(&pipe->lock);
    return -EPIPE;
}
```

---