

---

Subject: [PATCH v2 0/6] NFS: create clients and IDMAP pipes per network namespace

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:12:27 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) Rebased on tag v3.2
- 2) Added "nfs\_idmap\_init" and "nfs\_idmap\_quit" definitions for kernels built without CONFIG\_NFS\_V4 option set

This patch set depends on previous patch sets titled:

- 1) "SUNRPC: initial part of making pipefs work in net ns"
- 2) "SUNRPC: cleanup PipeFS for network-namespace-aware users"
- 3) "SUNRPC: make RPC clients use network-namespace-aware PipeFS routines"

Actually, this patch set consists of two tightly connected parts:

- 1) NFS clients creation per network namespace
- 2) IDMAP pipe dentries creation in owner client network namespace context

This patch set is the third part of making "PipeFS per network namespace".

The following series consists of:

---

Stanislav Kinsbursky (6):

- SUNRPC: fix pipe->ops cleanup on pipe dentry unlink
- NFS: make NFS client allocated per network namespace context
- NFS: pass NFS client owner network namespace to RPC client creation routine
- NFS: create callback transports in parent transport network namespace
- NFS: handle NFS idmap pipe PipeFS dentries by network namespace aware routines
- NFS: idmap PipeFS notifier introduced

```
fs/nfs/callback.c      | 10 +--
fs/nfs/client.c        | 22 +++++
fs/nfs/idmap.c         | 136 +++++
fs/nfs/internal.h      |  6 ++
fs/nfs/mount_clnt.c    |  4 +
fs/nfs/super.c         |  4 +
include/linux/nfs_fs_sb.h |  1
include/linux/nfs_idmap.h | 21 +---
include/linux/sunrpc/rpc_pipe_fs.h |  7 ++
net/sunrpc/clnt.c      |  1
```

net/sunrpc/rpc\_pipe.c | 67 ++++++++-----  
11 files changed, 217 insertions(+), 62 deletions(-)

--  
Signature

---

---

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-NULL pipe->ops (or pipe->dentry) in rpc\_close\_pipes() because it always non-NULL 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;
    }

```

---

Subject: [PATCH v2 2/6] NFS: make NFS client allocated per network namespace context

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:12:46 GMT

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

---

This patch adds new net variable to nfs\_client structure. This variable is set on NFS client creation and checked during matching NFS client search. Initially current->nsproxy->net\_ns is used as network namespace owner for new NFS client to create. This network namespace pointer is set during mount options parsing and thus can be passed from user-space utils in future if will be necessary.

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

---

```

fs/nfs/client.c      | 16 ++++++++
fs/nfs/internal.h    |  1 +
fs/nfs/super.c       |  3 +++
include/linux/nfs_fs_sb.h |  1 +
4 files changed, 18 insertions(+), 3 deletions(-)

```

diff --git a/fs/nfs/client.c b/fs/nfs/client.c

index 873bf00..4fb5cae 100644

--- a/fs/nfs/client.c

+++ b/fs/nfs/client.c

```

@@ -135,6 +135,7 @@ struct nfs_client_initdata {
    const struct nfs_rpc_ops *rpc_ops;
    int proto;
    u32 minorversion;
+ struct net *net;

```

```

};

/*
@@ -189,6 +190,7 @@ static struct nfs_client *nfs_alloc_client(const struct nfs_client_initdata
*cl_
    if (!IS_ERR(cred))
        clp->cl_machine_cred = cred;
    nfs_fscache_get_client_cookie(clp);
+ clp->net = cl_init->net;

    return clp;

@@ -476,6 +478,9 @@ static struct nfs_client *nfs_match_client(const struct nfs_client_initdata
*dat
    /* Match the full socket address */
    if (!nfs_sockaddr_cmp(sap, clap))
        continue;
+ /* Match network namespace */
+ if (clp->net != data->net)
+     continue;

    atomic_inc(&clp->cl_count);
    return clp;
@@ -826,6 +831,7 @@ static int nfs_init_server(struct nfs_server *server,
    .addrlen = data->nfs_server.addrlen,
    .rpc_ops = &nfs_v2_clientops,
    .proto = data->nfs_server.protocol,
+ .net = data->net,
};
struct rpc_timeout timeparms;
struct nfs_client *clp;
@@ -1387,7 +1393,7 @@ static int nfs4_set_client(struct nfs_server *server,
    const char *ip_addr,
    rpc_authflavor_t authflavour,
    int proto, const struct rpc_timeout *timeparms,
- u32 minorversion)
+ u32 minorversion, struct net *net)
{
    struct nfs_client_initdata cl_init = {
        .hostname = hostname,
@@ -1396,6 +1402,7 @@ static int nfs4_set_client(struct nfs_server *server,
    .rpc_ops = &nfs_v4_clientops,
    .proto = proto,
    .minorversion = minorversion,
+ .net = net,
};
struct nfs_client *clp;
int error;

```

```

@@ -1447,6 +1454,7 @@ struct nfs_client *nfs4_set_ds_client(struct nfs_client* mds_clp,
    .rpc_ops = &nfs_v4_clientops,
    .proto = ds_proto,
    .minorversion = mds_clp->cl_minorversion,
+   .net = mds_clp->net,
};
struct rpc_timeout ds_timeout = {
    .to_initval = 15 * HZ,
@@ -1573,7 +1581,8 @@ static int nfs4_init_server(struct nfs_server *server,
    data->auth_flavors[0],
    data->nfs_server.protocol,
    &timeparms,
-   data->minorversion);
+   data->minorversion,
+   data->net);
    if (error < 0)
        goto error;

@@ -1670,7 +1679,8 @@ struct nfs_server *nfs4_create_referral_server(struct nfs_clone_mount
*data,
    data->authflavor,
    parent_server->client->cl_xprt->prot,
    parent_server->client->cl_timeout,
-   parent_client->cl_mvops->minor_version);
+   parent_client->cl_mvops->minor_version,
+   parent_client->net);
    if (error < 0)
        goto error;

diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
index 3f4d957..c60fa0b 100644
--- a/fs/nfs/internal.h
+++ b/fs/nfs/internal.h
@@ -123,6 +123,7 @@ struct nfs_parsed_mount_data {
    } nfs_server;

    struct security_mnt_opts lsm_opts;
+   struct net *net;
};

/* mount_clnt.c */
diff --git a/fs/nfs/super.c b/fs/nfs/super.c
index b31bdf4..7495039 100644
--- a/fs/nfs/super.c
+++ b/fs/nfs/super.c
@@ -53,6 +53,7 @@
#include <linux/nfs_xdr.h>
#include <linux/magic.h>

```

```

#include <linux/parser.h>
#include <linux/nsproxy.h>

#include <asm/system.h>
#include <asm/uaccess.h>
@@ -1094,6 +1095,8 @@ static int nfs_parse_mount_options(char *raw,

    free_secdata(secdata);

+ mnt->net = current->nsproxy->net_ns;
+
    while ((p = strsep(&raw, ",")) != NULL) {
        substring_t args[MAX_OPT_ARGS];
        unsigned long option;
diff --git a/include/linux/nfs_fs_sb.h b/include/linux/nfs_fs_sb.h
index b5479df..71fd605 100644
--- a/include/linux/nfs_fs_sb.h
+++ b/include/linux/nfs_fs_sb.h
@@ -85,6 +85,7 @@ struct nfs_client {
    #endif

    struct server_scope *server_scope; /* from exchange_id */
+ struct net *net;
};

/*

```

---

Subject: [PATCH v2 3/6] NFS: pass NFS client owner network namespace to RPC client creation routine

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:12:54 GMT

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

---

This patch replaces static "init\_net" with nfs\_client->net pointer in RPC client creation calls.

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

```

---
fs/nfs/client.c    | 2 +-
fs/nfs/internal.h  | 1 +
fs/nfs/mount_clnt.c | 4 +---
fs/nfs/super.c     | 1 +
4 files changed, 5 insertions(+), 3 deletions(-)

```

```

diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 4fb5cae..18153c2 100644
--- a/fs/nfs/client.c

```



```

+++ b/fs/nfs/client.c
@@ -642,7 +642,7 @@ static int nfs_create_rpc_client(struct nfs_client *clp,
{
    struct rpc_clnt *clnt = NULL;
    struct rpc_create_args args = {
- .net = &init_net,
+ .net = clp->net,
    .protocol = clp->cl_proto,
    .address = (struct sockaddr *)&clp->cl_addr,
    .addrsz = clp->cl_addrlen,
diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
index c60fa0b..e07627d 100644
--- a/fs/nfs/internal.h
+++ b/fs/nfs/internal.h
@@ -138,6 +138,7 @@ struct nfs_mount_request {
    int noresvport;
    unsigned int *auth_flav_len;
    rpc_authflavor_t *auth_flavs;
+ struct net *net;
};

extern int nfs_mount(struct nfs_mount_request *info);
diff --git a/fs/nfs/mount_clnt.c b/fs/nfs/mount_clnt.c
index d4c2d6b..4fbc3a8 100644
--- a/fs/nfs/mount_clnt.c
+++ b/fs/nfs/mount_clnt.c
@@ -153,7 +153,7 @@ int nfs_mount(struct nfs_mount_request *info)
{
    .rpc_resp = &result,
};
    struct rpc_create_args args = {
- .net = &init_net,
+ .net = info->net,
    .protocol = info->protocol,
    .address = info->sap,
    .addrsz = info->salen,
@@ -225,7 +225,7 @@ void nfs_umount(const struct nfs_mount_request *info)
{
    .to_retries = 2,
};
    struct rpc_create_args args = {
- .net = &init_net,
+ .net = info->net,
    .protocol = IPPROTO_UDP,
    .address = info->sap,
    .addrsz = info->salen,
diff --git a/fs/nfs/super.c b/fs/nfs/super.c
index 7495039..9395136 100644
--- a/fs/nfs/super.c
+++ b/fs/nfs/super.c

```

```
@@ -1612,6 +1612,7 @@ static int nfs_try_mount(struct nfs_parsed_mount_data *args,
    .nresvport = args->flags & NFS_MOUNT_NORESVPORT,
    .auth_flav_len = &server_authlist_len,
    .auth_flavs = server_authlist,
+ .net = args->net,
};
int status;
```

---

Subject: [PATCH v2 4/6] NFS: create callback transports in parent transport network namespace

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:13:03 GMT

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

---

This patch replaces static "init\_net" references with parent transport xprt\_net reference. Thus callback transports will be created in the same network namespace as respective NFS mount point was created.

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

---

```
fs/nfs/callback.c | 10 +++++-----
1 files changed, 5 insertions(+), 5 deletions(-)
```

```
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 516f337..d81040a 100644
```

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

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

```
@@ -102,11 +102,11 @@ nfs4_callback_svc(void *vrqstp)
```

```
 * Prepare to bring up the NFSv4 callback service
 */
```

```
struct svc_rqst *
```

```
-nfs4_callback_up(struct svc_serv *serv)
```

```
+nfs4_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
```

```
{
    int ret;
```

```
- ret = svc_create_xprt(serv, "tcp", &init_net, PF_INET,
```

```
+ ret = svc_create_xprt(serv, "tcp", xprt->xprt_net, PF_INET,
    nfs_callback_set_tcpport, SVC_SOCKET_ANONYMOUS);
    if (ret <= 0)
```

```
    goto out_err;
```

```
@@ -114,7 +114,7 @@ nfs4_callback_up(struct svc_serv *serv)
```

```
    dprintk("NFS: Callback listener port = %u (af %u)\n",
    nfs_callback_tcpport, PF_INET);
```

```
- ret = svc_create_xprt(serv, "tcp", &init_net, PF_INET6,
```

```
+ ret = svc_create_xprt(serv, "tcp", xprt->xprt_net, PF_INET6,
```

```

    nfs_callback_set_tcpport, SVC_SOCKET_ANONYMOUS);
if (ret > 0) {
    nfs_callback_tcpport6 = ret;
@@ -183,7 +183,7 @@ nfs41_callback_up(struct svc_serv *serv, struct rpc_xprt *xprt)
    * fore channel connection.
    * Returns the input port (0) and sets the svc_serv bc_xprt on success
    */
- ret = svc_create_xprt(serv, "tcp-bc", &init_net, PF_INET, 0,
+ ret = svc_create_xprt(serv, "tcp-bc", xprt->xprt_net, PF_INET, 0,
    SVC_SOCKET_ANONYMOUS);
if (ret < 0) {
    rqstp = ERR_PTR(ret);
@@ -269,7 +269,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
    serv, xprt, &rqstp, &callback_svc);
if (!minorversion_setup) {
    /* v4.0 callback setup */
- rqstp = nfs4_callback_up(serv);
+ rqstp = nfs4_callback_up(serv, xprt);
    callback_svc = nfs4_callback_svc;
}

```

---

Subject: [PATCH v2 5/6] NFS: handle NFS idmap pipe PipeFS dentries by network namespace aware routines

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:13:11 GMT

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

---

This patch makes NFS idmap pipes dentries allocated and destroyed in network namespace context by PipeFS network namespace aware routines.

Network namespace context is obtained from nfs\_client structure.

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

---

```

fs/nfs/idmap.c | 61 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-----
1 files changed, 53 insertions(+), 8 deletions(-)

```

```

diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c

```

```

index 9c701b5..2bab772 100644

```

```

--- a/fs/nfs/idmap.c

```

```

+++ b/fs/nfs/idmap.c

```

```

@@ -348,6 +348,56 @@ static const struct rpc_pipe_ops idmap_upcall_ops = {
    .destroy_msg = idmap_pipe_destroy_msg,
};

```

```

+static void __nfs_idmap_unregister(struct rpc_pipe *pipe)

```

```

+{

```

```

+ if (pipe->dentry)

```

```

+ rpc_unlink(pipe->dentry);
+}
+
+static int __nfs_idmap_register(struct dentry *dir,
+    struct idmap *idmap,
+    struct rpc_pipe *pipe)
+{
+ struct dentry *dentry;
+
+ dentry = rpc_mkpipe_dentry(dir, "idmap", idmap, pipe);
+ if (IS_ERR(dentry))
+ return PTR_ERR(dentry);
+ pipe->dentry = dentry;
+ return 0;
+}
+
+static void nfs_idmap_unregister(struct nfs_client *clp,
+    struct rpc_pipe *pipe)
+{
+ struct net *net = clp->net;
+ struct super_block *pipefs_sb;
+
+ pipefs_sb = rpc_get_sb_net(net);
+ if (pipefs_sb) {
+ __nfs_idmap_unregister(pipe);
+ rpc_put_sb_net(net);
+ }
+}
+
+static int nfs_idmap_register(struct nfs_client *clp,
+    struct idmap *idmap,
+    struct rpc_pipe *pipe)
+{
+ struct net *net = clp->net;
+ struct super_block *pipefs_sb;
+ int err = 0;
+
+ pipefs_sb = rpc_get_sb_net(net);
+ if (pipefs_sb) {
+ if (clp->cl_rpcclient->cl_dentry)
+ err = __nfs_idmap_register(clp->cl_rpcclient->cl_dentry,
+     idmap, pipe);
+ rpc_put_sb_net(net);
+ }
+ return err;
+}
+
+int

```

```

nfs_idmap_new(struct nfs_client *clp)
{
@@ -367,12 +417,8 @@ nfs_idmap_new(struct nfs_client *clp)
    kfree(idmap);
    return error;
}
-
- if (clp->cl_rpcclient->cl_dentry)
- pipe->dentry = rpc_mkpipe_dentry(clp->cl_rpcclient->cl_dentry,
- "idmap", idmap, pipe);
- if (IS_ERR(pipe->dentry)) {
- error = PTR_ERR(pipe->dentry);
+ error = nfs_idmap_register(clp, idmap, pipe);
+ if (error) {
    rpc_destroy_pipe_data(pipe);
    kfree(idmap);
    return error;
@@ -395,8 +441,7 @@ nfs_idmap_delete(struct nfs_client *clp)

    if (!idmap)
        return;
- if (idmap->idmap_pipe->dentry)
- rpc_unlink(idmap->idmap_pipe->dentry);
+ nfs_idmap_unregister(clp, idmap->idmap_pipe);
    rpc_destroy_pipe_data(idmap->idmap_pipe);
    clp->cl_idmap = NULL;
    kfree(idmap);

```

---

Subject: [PATCH v2 6/6] NFS: idmap PipeFS notifier introduced  
 Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 12:13:19 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

v2:

1) Added "nfs\_idmap\_init" and "nfs\_idmap\_quit" definitions for kernels built without CONFIG\_NFS\_V4 option set.

This patch subscribes NFS clients to RPC pipefs notifications. Idmap notifier is registering on NFS module load. This notifier callback is responsible for creation/destruction of PipeFS idmap pipe dentry for NFS4 clients.

Since ipdmap pipe is created in rpc client pipefs directory, we have make sure, that this directory has been created already. IOW RPC client notifier callback has been called already. To achive this, PipeFS notifier priorities has been introduced (RPC clients notifier priority is greater than NFS idmap one). But this approach gives another problem: unlink for RPC client directory will be called before NFS idmap pipe unlink on UMOUNT event and will fail, because directory is not empty.

The solution, introduced in this patch, is to try to remove client directory once again after idmap pipe was unlinked. This looks like ugly hack, so probably it should be replaced in some more elegant way.

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>

```
---
fs/nfs/client.c          |  4 +-
fs/nfs/idmap.c           | 75 ++++++
fs/nfs/internal.h        |  4 ++
include/linux/nfs_idmap.h | 21 +++++
include/linux/sunrpc/rpc_pipe_fs.h |  7 +++
net/sunrpc/clnt.c        |  1
net/sunrpc/rpc_pipe.c     | 16 ++++++
7 files changed, 116 insertions(+), 12 deletions(-)
```

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

```
index 18153c2..a3612db 100644
```

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

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

```
@ @ -52,8 +52,8 @ @
```

```
#define NFSDBG_FACILITY NFSDBG_CLIENT
```

```
-static DEFINE_SPINLOCK(nfs_client_lock);
```

```
-static LIST_HEAD(nfs_client_list);
```

```
+DEFINE_SPINLOCK(nfs_client_lock);
```

```
+LIST_HEAD(nfs_client_list);
```

```
static LIST_HEAD(nfs_volume_list);
```

```
static DECLARE_WAIT_QUEUE_HEAD(nfs_client_active_wq);
```

```
#ifdef CONFIG_NFS_V4
```

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

```
index 2bab772..75627aa 100644
```

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

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

```
@ @ -294,6 +294,7 @ @ int nfs_map_gid_to_group(const struct nfs_server *server, __u32 gid,
char *buf,
```

```
#include <linux/nfs_fs.h>
```

```
#include "nfs4_fs.h"
```

```
+#include "internal.h"
```

```
#define IDMAP_HASH_SZ      128
```

```

@@ -447,6 +448,80 @@ nfs_idmap_delete(struct nfs_client *clp)
    kfree(idmap);
}

```

```

+static int __rpc_pipefs_event(struct nfs_client *clp, unsigned long event,
+    struct super_block *sb)
+{
+    int err = 0;
+
+    switch (event) {
+    case RPC_PIPEFS_MOUNT:
+        BUG_ON(clp->cl_rpcclient->cl_dentry == NULL);
+        err = __nfs_idmap_register(clp->cl_rpcclient->cl_dentry,
+            clp->cl_idmap,
+            clp->cl_idmap->idmap_pipe);
+        break;
+    case RPC_PIPEFS_UMOUNT:
+        if (clp->cl_idmap->idmap_pipe) {
+            struct dentry *parent;
+
+            parent = clp->cl_idmap->idmap_pipe->dentry->d_parent;
+            __nfs_idmap_unregister(clp->cl_idmap->idmap_pipe);
+            /*
+             * Note: This is a dirty hack. SUNRPC hook has been
+             * called already but simple_rmdir() call for the
+             * directory returned with error because of idmap pipe
+             * inside. Thus now we have to remove this directory
+             * here.
+             */
+            if (rpc_rmdir(parent))
+                printk(KERN_ERR "%s: failed to remove clnt dir!\n", __func__);
+        }
+        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 nfs_client *clp;
+    int error = 0;
+
+    spin_lock(&nfs_client_lock);

```

```

+ list_for_each_entry(clp, &nfs_client_list, cl_share_link) {
+ if (clp->net != sb->s_fs_info)
+ continue;
+ if (clp->rpc_ops != &nfs_v4_clientops)
+ continue;
+ error = __rpc_pipefs_event(clp, event, sb);
+ if (error)
+ break;
+ }
+ spin_unlock(&nfs_client_lock);
+ return error;
+}
+
+#define PIPEFS_NFS_PRIO 1
+
+static struct notifier_block nfs_idmap_block = {
+ .notifier_call = rpc_pipefs_event,
+ .priority = SUNRPC_PIPEFS_NFS_PRIO,
+};
+
+int nfs_idmap_init(void)
+{
+ return rpc_pipefs_notifier_register(&nfs_idmap_block);
+}
+
+void nfs_idmap_quit(void)
+{
+ rpc_pipefs_notifier_unregister(&nfs_idmap_block);
+}
+
+/*
+ * Helper routines for manipulating the hashtable
+ */
diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
index e07627d..a4b518e 100644
--- a/fs/nfs/internal.h
+++ b/fs/nfs/internal.h
@@ -182,6 +182,10 @@ static inline void nfs_fs_proc_exit(void)
{
}
#endif
#ifdef CONFIG_NFS_V4
+extern spinlock_t nfs_client_lock;
+extern struct list_head nfs_client_list;
#endif

/* nfs4namespace.c */
#ifdef CONFIG_NFS_V4

```



```

diff --git a/include/linux/nfs_idmap.h b/include/linux/nfs_idmap.h
index ae7d6a3..436a1d7 100644
--- a/include/linux/nfs_idmap.h
+++ b/include/linux/nfs_idmap.h
@@ -67,31 +67,32 @@ struct idmap_msg {
    struct nfs_client;
    struct nfs_server;

-#ifdef CONFIG_NFS_USE_NEW_IDMAPPER
-
+#ifdef CONFIG_NFS_V4
    int nfs_idmap_init(void);
    void nfs_idmap_quit(void);
-
-static inline int nfs_idmap_new(struct nfs_client *clp)
+#else
+static inline int nfs_idmap_init(void)
+{
+    return 0;
+}

-static inline void nfs_idmap_delete(struct nfs_client *clp)
-{
-}
+static inline void nfs_idmap_quit(void)
+{}
+#endif

-#else /* CONFIG_NFS_USE_NEW_IDMAPPER not set */
+#ifdef CONFIG_NFS_USE_NEW_IDMAPPER

-static inline int nfs_idmap_init(void)
+static inline int nfs_idmap_new(struct nfs_client *clp)
+{
+    return 0;
+}

-static inline void nfs_idmap_quit(void)
+static inline void nfs_idmap_delete(struct nfs_client *clp)
+{
+}

+#else /* CONFIG_NFS_USE_NEW_IDMAPPER not set */
+
+int nfs_idmap_new(struct nfs_client *);
+void nfs_idmap_delete(struct nfs_client *);

```

diff --git a/include/linux/sunrpc/rpc\_pipe\_fs.h b/include/linux/sunrpc/rpc\_pipe\_fs.h

```

index 0808ed2..7eb0160 100644
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -49,6 +49,11 @@ RPC_I(struct inode *inode)
    return container_of(inode, struct rpc_inode, vfs_inode);
}

+enum {
+ SUNRPC_PIPEFS_NFS_PRIO,
+ SUNRPC_PIPEFS_RPC_PRIO,
+};
+
extern int rpc_pipefs_notifier_register(struct notifier_block *);
extern void rpc_pipefs_notifier_unregister(struct notifier_block *);

@@ -78,6 +83,8 @@ extern struct dentry *rpc_create_cache_dir(struct dentry *,
    struct cache_detail *);
extern void rpc_remove_cache_dir(struct dentry *);

+extern int rpc_rmdir(struct dentry *dentry);
+
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 *,
diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index 1087e4a..e4b20e7 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -236,6 +236,7 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,

static struct notifier_block rpc_clients_block = {
    .notifier_call = rpc_pipefs_event,
+ .priority = SUNRPC_PIPEFS_RPC_PRIO,
};

int rpc_clients_notifier_register(void)
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 6415241..cd565d7 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -617,6 +617,22 @@ static int __rpc_rmdir(struct inode *dir, struct dentry *dentry)
    return ret;
}

+int rpc_rmdir(struct dentry *dentry)
+{
+ struct dentry *parent;
+ struct inode *dir;

```

```
+ int error;
+
+ parent = dget_parent(dentry);
+ dir = parent->d_inode;
+ mutex_lock_nested(&dir->i_mutex, I_MUTEX_PARENT);
+ error = __rpc_rmdir(dir, dentry);
+ mutex_unlock(&dir->i_mutex);
+ dput(parent);
+ return error;
+}
+EXPORT_SYMBOL_GPL(rpc_rmdir);
+
static int __rpc_unlink(struct inode *dir, struct dentry *dentry)
{
    int ret;
```

---