Subject: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 10:58:44 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.
tag: v3.1

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            |  151 ++++++++++++++++++++++++++++++++++++-----
 net/sunrpc/rpc_pipe.c        |   19 +++--
 net/sunrpc/sunrpc.h          |    2 +
 9 files changed, 218 insertions(+), 67 deletions(-)
```

--
Signature

Subject: [PATCH 1/6] SUNRPC: handle RPC client pipefs dentries by network namespace aware routines
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:02:56 GMT
View Forum Message <> Reply to Message

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 | 93 ++++++++++++++++++++++++++++++++----------------
 3 files changed, 62 insertions(+), 35 deletions(-)

diff --git a/fs/nfsd/nfs4callback.c b/fs/nfsd/nfs4callback.c
index 02eb4ed..1ac6f55 100644
--- a/fs/nfsd/nfs4callback.c
+++ b/fs/nfsd/nfs4callback.c
@@ -618,7 +618,7 @@ 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 c5347d2..008c755 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -93,52 +93,85 @@ static void rpc_unregister_client(struct rpc_clnt *clnt)
  spin_unlock(&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;
+
+ pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
+ if (pipefs_sb) {
+  __rpc_clnt_remove_pipedir(clnt);
+  rpc_put_sb_net(clnt->cl_xprt->xprt_net);
+ }
+ 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)
@@ -246,10 +279,7 @@ 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:
@@ -474,10 +504,7 @@ 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 5/6] SUNRPC: remove RPC pipefs mount point manipulations from RPC clients code
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:04:57 GMT
View Forum Message <> Reply to Message

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 |   15 +++++------------
 1 files changed, 3 insertions(+), 12 deletions(-)

diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index 23776a4..eb2595f 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -112,7 +112,6 @@ static void rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
   __rpc_clnt_remove_pipedir(clnt);
   rpc_put_sb_net(clnt->cl_xprt->xprt_net);
  }
- rpc_put_mount();
 }

 static struct dentry *rpc_setup_pipedir_sb(struct super_block *sb,
@@ -158,21 +157,13 @@ 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;
 }
```

Subject: [PATCH 6/6] SUNRPC: remove RPC PipeFS mount point reference from RPC client
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:04:57 GMT
View Forum Message <> Reply to Message

This is a cleanup patch. We don't need this reference anymore.

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

```
---
 fs/nfs/idmap.c                |    4 ++--
 include/linux/sunrpc/clnt.h   |    2 +-
 net/sunrpc/auth_gss/auth_gss.c |    8 ++++----
 net/sunrpc/clnt.c             |   21 ++++++++++-----------
 4 files changed, 17 insertions(+), 18 deletions(-)

diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index b09a7f1..60698a1 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -370,8 +370,8 @@ nfs_idmap_new(struct nfs_client *clp)
   return error;
  }

- if (clp->cl_rpcclient->cl_path.dentry)
-  pipe->dentry = rpc_mkpipe_dentry(clp->cl_rpcclient->cl_path.dentry,
+ 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);
diff --git a/include/linux/sunrpc/clnt.h b/include/linux/sunrpc/clnt.h
index db7bcaf..9fe39bc 100644
--- a/include/linux/sunrpc/clnt.h
+++ b/include/linux/sunrpc/clnt.h
@@ -56,7 +56,7 @@ struct rpc_clnt {

  int   cl_nodelen; /* nodename length */
```

```
  char    cl_nodename[UNX_MAXNODENAME];
- struct path  cl_path;
+ struct dentry *  cl_dentry;
  struct rpc_clnt * cl_parent; /* Points to parent of clones */
  struct rpc_rtt  cl_rtt_default;
  struct rpc_timeout cl_timeout_default;
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 66293dc..8673220 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -799,12 +799,12 @@ static int gss_pipes_dentries_create(struct rpc_auth *auth)
  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,
+ gss_auth->pipe[1]->dentry = rpc_mkpipe_dentry(clnt->cl_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->pipe[0]->dentry = rpc_mkpipe_dentry(clnt->cl_dentry,
             gss_auth->mech->gm_name,
             clnt, gss_auth->pipe[0]);
  if (IS_ERR(gss_auth->pipe[0]->dentry)) {
@@ -826,7 +826,7 @@ static void gss_pipes_dentries_destroy_net(struct rpc_clnt *clnt,

  sb = rpc_get_sb_net(net);
  if (sb) {
-  if (clnt->cl_path.dentry)
+  if (clnt->cl_dentry)
    gss_pipes_dentries_destroy(auth);
   rpc_put_sb_net(net);
  }
@@ -841,7 +841,7 @@ static int gss_pipes_dentries_create_net(struct rpc_clnt *clnt,

  sb = rpc_get_sb_net(net);
  if (sb) {
-  if (clnt->cl_path.dentry)
+  if (clnt->cl_dentry)
    err = gss_pipes_dentries_create(auth);
   rpc_put_sb_net(net);
  }
diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index eb2595f..3971aaa 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -95,12 +95,12 @@ 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_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);
+  rpc_remove_client_dir(clnt->cl_dentry);
   }
- clnt->cl_path.dentry = NULL;
+ clnt->cl_dentry = NULL;
 }

 static void rpc_clnt_remove_pipedir(struct rpc_clnt *clnt)
@@ -151,20 +151,19 @@ static int
 rpc_setup_pipedir(struct rpc_clnt *clnt, char *dir_name)
 {
  struct super_block *pipefs_sb;
- struct path path;
+ struct dentry *dentry;

- clnt->cl_path.mnt = ERR_PTR(-ENOENT);
- clnt->cl_path.dentry = NULL;
+ clnt->cl_dentry = NULL;
  if (dir_name == NULL)
   return 0;
  pipefs_sb = rpc_get_sb_net(clnt->cl_xprt->xprt_net);
  if (!pipefs_sb)
   return 0;
- path.dentry = rpc_setup_pipedir_sb(pipefs_sb, clnt, dir_name);
+ dentry = rpc_setup_pipedir_sb(pipefs_sb, clnt, dir_name);
  rpc_put_sb_net(clnt->cl_xprt->xprt_net);
- if (IS_ERR(path.dentry))
-  return PTR_ERR(path.dentry);
- clnt->cl_path = path;
+ if (IS_ERR(dentry))
+  return PTR_ERR(dentry);
+ clnt->cl_dentry = dentry;
  return 0;
 }

@@ -183,7 +182,7 @@ static int __rpc_pipefs_event(struct rpc_clnt *clnt, unsigned long event,
    BUG_ON(dentry == NULL);
    if (IS_ERR(dentry))
     return PTR_ERR(dentry);
-  clnt->cl_path.dentry = dentry;
+  clnt->cl_dentry = dentry;
```

```
   if (clnt->cl_auth->au_ops->pipes_create) {
    err = clnt->cl_auth->au_ops->pipes_create(clnt->cl_auth);
    if (err)
```

Subject: [PATCH 4/6] SUNRPC: remove RPC client pipefs dentries after unregister
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:05:00 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 61c76ae..23776a4 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -572,7 +572,6 @@ 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;
@@ -581,6 +580,7 @@ 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 3/6] SUNRPC: subscribe RPC clients to pipefs notifications
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:05:03 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 | 70 ++
 net/sunrpc/rpc_pipe.c | 19 +++++++----
 net/sunrpc/sunrpc.h | 2 +
 5 files changed, 93 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 248acd0..66293dc 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -784,8 +784,10 @@ 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)
@@ -1628,7 +1630,9 @@ 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 008c755..61c76ae 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -95,8 +95,11 @@ 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;
 }

@@ -174,6 +177,71 @@ 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;
+
+ spin_lock(&rpc_client_lock);
+ list_for_each_entry(clnt, &all_clients, cl_clients) {
+  if (clnt->cl_xprt->xprt_net != sb->s_fs_info)
+   continue;
+  error = __rpc_pipefs_event(clnt, event, sb);
+  if (error)
+   break;
+ }
+ spin_unlock(&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 8e59580..1ea0dcf 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -918,7 +918,7 @@ 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)
 {
@@ -1169,17 +1169,24 @@ 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 2/6] SUNRPC: handle GSS AUTH pipes by network namespace aware routines
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 11:14:01 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 2b25a7b..248acd0 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -779,6 +779,73 @@ 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)
@@ -834,31 +901,16 @@ 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:
@@ -875,8 +927,7 @@ 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: Re: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by bfields on Wed, 23 Nov 2011 16:27:11 GMT
View Forum Message <> Reply to Message

On Wed, Nov 23, 2011 at 02:51:10PM +0300, Stanislav Kinsbursky wrote:
> This patch set was created in context of clone of git
> branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
> tag: v3.1
>
> 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"

Do you have a git tree set up with all of these applied?

(If I want to take a quick look it would personally be easier for me to
fetch your git tree than to get those patches out of old email.)

--b.

>
> 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 | 151 ++++++++++++++++++++++++++++++++++++++---------
> net/sunrpc/rpc_pipe.c | 19 +++--
> net/sunrpc/sunrpc.h | 2 +
> 9 files changed, 218 insertions(+), 67 deletions(-)
>
> --
> Signature

Subject: Re: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by bfields on Wed, 23 Nov 2011 16:36:32 GMT
View Forum Message <> Reply to Message

On Wed, Nov 23, 2011 at 02:51:10PM +0300, Stanislav Kinsbursky wrote:
> This patch set was created in context of clone of git
> branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
> tag: v3.1
>
> 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.

I'm not following very well. (My fault, I haven't been paying
attention.) Could you summarize the itended behavior of pipefs after
all this is done?

So there's a separate superblock (and separate dentries) for each
namespace?

What decides which clients are visible in which network namespaces?

--b.

>
> 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 | 151 ++---------
> net/sunrpc/rpc_pipe.c | 19 +++--
> net/sunrpc/sunrpc.h | 2 +
> 9 files changed, 218 insertions(+), 67 deletions(-)
>
> --
> Signature

Subject: Re: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 17:18:41 GMT
View Forum Message <> Reply to Message

> On Wed, Nov 23, 2011 at 02:51:10PM +0300, Stanislav Kinsbursky wrote:
>> This patch set was created in context of clone of git
>> branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
>> tag: v3.1
>>
>> 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"
>
> Do you have a git tree set up with all of these applied?
>

I have, but no external access to it yet.

> (If I want to take a quick look it would personally be easier for me to
> fetch your git tree than to get those patches out of old email.)
>

Will check tomorrow, is it possible to organize it.

> --b.
>

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by Stanislav Kinsbursky on Wed, 23 Nov 2011 17:58:25 GMT
View Forum Message <> Reply to Message

> On Wed, Nov 23, 2011 at 02:51:10PM +0300, Stanislav Kinsbursky wrote:
>> This patch set was created in context of clone of git
>> branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
>> tag: v3.1
>>
>> 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.
>
> I'm not following very well. (My fault, I haven't been paying
> attention.) Could you summarize the itended behavior of pipefs after
> all this is done?
>

> So there's a separate superblock (and separate dentries) for each
> namespace?
>

Yes, you right.
So, here is a brief summary of what will be at the end:

1) PipeFS superblock will be per net ns.

2) Superblock holds net ns, which is taken from current. Struct net will have link ot pipefs superblock (it can be NULL, if PipeFS wasn't mounted yet or already unmounted).

3) Notifications will be send on superblock creation and destruction.

4) All kernel mount point creation and destruction calls (rpc_get_mount() and rpc_put_mount()) will be removed. I.e. this superblock will be created only from user-space.

5) Kernel pipes and dentries will be created or destroyed:
1. During per-net operations (only for static NFS stuff: dns_resolve cache, pnfs blocklayout and idmap pipes).
2. On notification events (all directories, files and pipes in proper callbacks). Notification subscribers:
a. rpc clients (responsible for client dentries and gss pipes creation),
b. nfs clients (responsible nfs idmap pipes),
c. nfs dns_resolve cache,
d. pnfs blocklayout pipes,

6) PipeFS dentries creation logic:
a) All directories and files creators - will try to create them as usual. But if fail - then no problem here. I.e. dentries will be created on PipeFS mount notification call.
b) Pipes creators - will create new structure rpc_pipe (all pipe stuff from rpc inode) nad then try to create pipe denties. If fail - then, again, no problem. Dentries will be be created on PipeFS mount notification call.

Almost all (exept 5.2.b - forgot about it during debasing and resending) is done and ready to send.

> What decides which clients are visible in which network namespaces?
>

Clients dentries will be created in proper superblock from the beginning. I.e. rpc clients transports have struct net reference. NFS clients will have such reference too. Struct net will have reference to pipefs superblock.
Currently, for dentry creation all we need is parent dentry. Some of creators

(like GSS pipes) takes parent dentry from associated struct (like rpc_clnt). For
others parent dentry can be found by simple d_lookup() starting from sb->root
(reminder: sb can be taken from net).

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 0/6] SUNRPC: make RPC clients use network-namespace-aware PipeFS routines
Posted by Stanislav Kinsbursky on Thu, 24 Nov 2011 08:45:30 GMT
View Forum Message <> Reply to Message

> On Wed, Nov 23, 2011 at 02:51:10PM +0300, Stanislav Kinsbursky wrote:
>> This patch set was created in context of clone of git
>> branch: git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
>> tag: v3.1
>>
>> 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"
>
> Do you have a git tree set up with all of these applied?
>

Hello, Bruce.
You can clone my current working branch from:

git://github.com/skinsbursky/nfs-per-net-ns.git

I'll update it after any significant changes.

There is also web interface:

https://github.com/skinsbursky/nfs-per-net-ns/tree/rpcpipefs _per_netns

--
Best regards,
Stanislav Kinsbursky
