Subject: [PATCH 6/6] NFS: idmap PipeFS notifier introduced
Posted by Stanislav Kinsbursky on Mon, 28 Nov 2011 13:38:49 GMT
View Forum Message <> Reply to Message

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 | 13 +-----
 include/linux/sunrpc/rpc_pipe_fs.h | 7 +++
 net/sunrpc/clnt.c | 1
 net/sunrpc/rpc_pipe.c | 16 ++++++++
 7 files changed, 107 insertions(+), 13 deletions(-)

diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 58eebb5..4398587 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 0680f63..d8fed37 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

@@ -449,6 +450,80 @@ 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 451acb5..00e8fa5 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..238c814 100644
--- a/include/linux/nfs_idmap.h
+++ b/include/linux/nfs_idmap.h
@@ -67,11 +67,11 @@ struct idmap_msg {
 struct nfs_client;
 struct nfs_server;

-#ifdef CONFIG_NFS_USE_NEW_IDMAPPER
-
 int nfs_idmap_init(void);
 void nfs_idmap_quit(void);

+#ifdef CONFIG_NFS_USE_NEW_IDMAPPER
+
 static inline int nfs_idmap_new(struct nfs_client *clp)
 {
  return 0;
@@ -83,15 +83,6 @@ static inline void nfs_idmap_delete(struct nfs_client *clp)

 #else /* CONFIG_NFS_USE_NEW_IDMAPPER not set */

-static inline int nfs_idmap_init(void)
-{
- return 0;
-}
-
-static inline void nfs_idmap_quit(void)
-{
-}
-
 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 c7237fa..e4f210d 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -234,6 +234,7 @@ 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 b80d7bd..e194e32 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -597,6 +597,22 @@ 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;
```