Subject: [PATCH 1/4] NFS: make nfs_client_list per net ns
Posted by Stanislav Kinsbursky on Mon, 23 Jan 2012 17:26:05 GMT
View Forum Message <> Reply to Message

This patch splits global list of NFS clients into per-net-ns array of lists.
This looks more strict and clearer.
BTW, this patch also makes "/proc/fs/nfsfs/servers" entry content depends on
/proc mount owner pid namespace. See below for details.

NOTE: few words about how was /proc/fs/nfsfs/ entries content show per network
namespace done. This is a little bit tricky and not the best is could be. But
it's cheap (proper fix for /proc conteinerization is a hard nut to crack).
The idea is simple: take proper network namespace from pid namespace
child reaper nsproxy of /proc/ mount creator.
This actually means, that if there are 2 containers with different net
namespace sharing pid namespace, then read of /proc/fs/nfsfs/ entries will
always return content, taken from net namespace of pid namespace creator task
(and thus second namespace set wil be unvisible).

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

---
 fs/nfs/client.c | 38 ++++++++++++++++++++++++++++++++++++++----------
 fs/nfs/idmap.c | 5 ++---
 fs/nfs/inode.c | 1 +
 fs/nfs/internal.h | 2 +-
 fs/nfs/netns.h | 1 +
 5 files changed, 32 insertions(+), 15 deletions(-)

diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 98af1cb..058eb9b 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -39,6 +39,8 @@
 #include <net/ipv6.h>
 #include <linux/nfs_xdr.h>
 #include <linux/sunrpc/bc_xprt.h>
+#include <linux/nsproxy.h>
+#include <linux/pid_namespace.h>

 #include <asm/system.h>

@@ -49,11 +51,11 @@
 #include "internal.h"
 #include "fscache.h"
 #include "pnfs.h"
+#include "netns.h"

```
 #define NFSDBG_FACILITY  NFSDBG_CLIENT

 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
@@ -464,8 +466,9 @@ static struct nfs_client *nfs_match_client(const struct nfs_client_initdata
*dat
 {
  struct nfs_client *clp;
  const struct sockaddr *sap = data->addr;
+ struct nfs_net *nn = net_generic(data->net, nfs_net_id);

- list_for_each_entry(clp, &nfs_client_list, cl_share_link) {
+ list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
         const struct sockaddr *clap = (struct sockaddr *)&clp->cl_addr;
   /* Don't match clients that failed to initialise properly */
   if (clp->cl_cons_state < 0)
@@ -483,9 +486,6 @@ 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;
@@ -506,6 +506,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
 {
  struct nfs_client *clp, *new = NULL;
  int error;
+ struct nfs_net *nn = net_generic(cl_init->net, nfs_net_id);

  dprintk("--> nfs_get_client(%s,v%u)\n",
   cl_init->hostname ?: "", cl_init->rpc_ops->version);
@@ -531,7 +532,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
  /* install a new client and return with it unready */
 install_client:
  clp = new;
- list_add(&clp->cl_share_link, &nfs_client_list);
+ list_add(&clp->cl_share_link, &nn->nfs_client_list);
  spin_unlock(&nfs_client_lock);

  error = cl_init->rpc_ops->init_client(clp, timeparms, ip_addr,
@@ -1227,9 +1228,10 @@ nfs4_find_client_sessionid(const struct sockaddr *addr,
```

```
       struct nfs4_sessionid *sid)
 {
  struct nfs_client *clp;
+ struct nfs_net *nn = net_generic(&init_net, nfs_net_id);

  spin_lock(&nfs_client_lock);
- list_for_each_entry(clp, &nfs_client_list, cl_share_link) {
+ list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
    if (nfs4_cb_match_client(addr, clp, 1) == false)
     continue;

@@ -1757,6 +1759,13 @@ out_free_server:
  return ERR_PTR(error);
 }

+void nfs_clients_init(struct net *net)
+{
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+
+ INIT_LIST_HEAD(&nn->nfs_client_list);
+}
+
 #ifdef CONFIG_PROC_FS
 static struct proc_dir_entry *proc_fs_nfs;

@@ -1810,13 +1819,15 @@ static int nfs_server_list_open(struct inode *inode, struct file *file)
 {
  struct seq_file *m;
  int ret;
+ struct pid_namespace *pid_ns = file->f_dentry->d_sb->s_fs_info;
+ struct net *net = pid_ns->child_reaper->nsproxy->net_ns;

  ret = seq_open(file, &nfs_server_list_ops);
  if (ret < 0)
   return ret;

  m = file->private_data;
- m->private = PDE(inode)->data;
+ m->private = net;

  return 0;
 }
@@ -1826,9 +1837,11 @@ static int nfs_server_list_open(struct inode *inode, struct file *file)
  */
 static void *nfs_server_list_start(struct seq_file *m, loff_t *_pos)
 {
+ struct nfs_net *nn = net_generic(m->private, nfs_net_id);
+
```

```
  /* lock the list against modification */
  spin_lock(&nfs_client_lock);
- return seq_list_start_head(&nfs_client_list, *_pos);
+ return seq_list_start_head(&nn->nfs_client_list, *_pos);
 }

 /*
@@ -1836,7 +1849,9 @@ static void *nfs_server_list_start(struct seq_file *m, loff_t *_pos)
  */
 static void *nfs_server_list_next(struct seq_file *p, void *v, loff_t *pos)
 {
- return seq_list_next(v, &nfs_client_list, pos);
+ struct nfs_net *nn = net_generic(p->private, nfs_net_id);
+
+ return seq_list_next(v, &nn->nfs_client_list, pos);
 }

 /*
@@ -1853,9 +1868,10 @@ static void nfs_server_list_stop(struct seq_file *p, void *v)
 static int nfs_server_list_show(struct seq_file *m, void *v)
 {
  struct nfs_client *clp;
+ struct nfs_net *nn = net_generic(m->private, nfs_net_id);

  /* display header on line 1 */
- if (v == &nfs_client_list) {
+ if (v == &nn->nfs_client_list) {
   seq_puts(m, "NV SERVER   PORT USE HOSTNAME\n");
   return 0;
  }
diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index ff084d2..92deaf8 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -571,13 +571,12 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long
event,
        void *ptr)
 {
  struct super_block *sb = ptr;
+ struct nfs_net *nn = net_generic(sb->s_fs_info, nfs_net_id);
  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;
+ list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
```

```
   if (clp->rpc_ops != &nfs_v4_clientops)
    continue;
   error = __rpc_pipefs_event(clp, event, sb);
diff --git a/fs/nfs/inode.c b/fs/nfs/inode.c
index d2c760e..e8f81e5 100644
--- a/fs/nfs/inode.c
+++ b/fs/nfs/inode.c
@@ -1558,6 +1558,7 @@ EXPORT_SYMBOL_GPL(nfs_net_id);

 static int nfs_net_init(struct net *net)
 {
+ nfs_clients_init(net);
  return nfs_dns_resolver_cache_init(net);
 }

diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
index cdb121d..a9ae806 100644
--- a/fs/nfs/internal.h
+++ b/fs/nfs/internal.h
@@ -146,6 +146,7 @@ extern void nfs_umount(const struct nfs_mount_request *info);

 /* client.c */
 extern const struct rpc_program nfs_program;
+extern void nfs_clients_init(struct net *net);

 extern void nfs_cleanup_cb_ident_idr(void);
 extern void nfs_put_client(struct nfs_client *);
@@ -183,7 +184,6 @@ 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 */
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index 39ae4ca..feb33c3 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -7,6 +7,7 @@
 struct nfs_net {
  struct cache_detail *nfs_dns_resolve;
  struct rpc_pipe *bl_device_pipe;
+ struct list_head nfs_client_list;
 };

 extern int nfs_net_id;
```