Subject: [PATCH v3 0/7] SUNRPC: initial part of making pipefs work in net ns
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:39:55 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.

v3:
1) Rebased on tag "v3.2-rc7".

v2:
1) Rebased of current repo state (i.e. all commits were pulled before apply)

This patch-set implements pipefs superblock creation per network namespace context instead of using single one for all possible contexts. Also, it provides pipefs dentries creation and destruction helpers for kernel routines.

Additional description of the idea about how to make RPC pipefs work per network namespace context can be found in the letter titled "SUNRPC: "RPC pipefs per network namespace" preparations", which has been sent already to linux-nfs@vger.kernel.org.
The following series consists of:

Stanislav Kinsbursky (7):
SUNRPC: create RPC pipefs superblock per network namespace context
SUNRPC: hold current network namespace while pipefs superblock is active
SUNRPC: send notification events on pipefs sb creation and destruction
SUNRPC: pipefs dentry lookup helper introduced
SUNRPC: put pipefs superblock link on network namespace
SUNRPC: pipefs per-net operations helper introduced
SUNRPC: added debug messages to RPC pipefs

```
include/linux/sunrpc/rpc_pipe_fs.h |   14 ++++
net/sunrpc/netns.h                 |    3 +
net/sunrpc/rpc_pipe.c              |  113 ++++++++++++++++++++++++++++++++++++-
net/sunrpc/sunrpc_syms.c           |    1
4 files changed, 129 insertions(+), 2 deletions(-)
```

--
Signature

Subject: [PATCH v3 1/7] SUNRPC: create RPC pipefs superblock per network namespace context

Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:40:02 GMT
View Forum Message <> Reply to Message

This is the initial step of RPC pipefs virtualization. It changes nothing to
current pipefs behaviour except that mount of pipefs in other than init_net
network namespace context will provide only root tree. No other dentries will
be visible.

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

---
 net/sunrpc/rpc_pipe.c | 3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index e2f7b7f..bb8a40b 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -994,6 +994,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
 {
  struct inode *inode;
  struct dentry *root;
+ struct net *net = data;

  sb->s_blocksize = PAGE_CACHE_SIZE;
  sb->s_blocksize_bits = PAGE_CACHE_SHIFT;
@@ -1018,7 +1019,7 @@ static struct dentry *
 rpc_mount(struct file_system_type *fs_type,
   int flags, const char *dev_name, void *data)
 {
- return mount_single(fs_type, flags, data, rpc_fill_super);
+ return mount_ns(fs_type, flags, current->nsproxy->net_ns, rpc_fill_super);
 }

 static struct file_system_type rpc_pipe_fs_type = {

Subject: [PATCH v3 2/7] SUNRPC: hold current network namespace while pipefs superblock is active
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:03 GMT
View Forum Message <> Reply to Message

We want to be sure that network namespace is still alive while we have pipefs
mounted.
This will be required later, when RPC pipefs will be mounting only from
user-space context.

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

```
---
 net/sunrpc/rpc_pipe.c |   14 ++++++++++++-
 1 files changed, 13 insertions(+), 1 deletions(-)

diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index bb8a40b..ff41fef 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -27,6 +27,9 @@
 #include <linux/workqueue.h>
 #include <linux/sunrpc/rpc_pipe_fs.h>
 #include <linux/sunrpc/cache.h>
+#include <linux/nsproxy.h>
+
+#include "netns.h"

 static struct vfsmount *rpc_mnt __read_mostly;
 static int rpc_mount_count;
@@ -1012,6 +1015,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  }
  if (rpc_populate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF, NULL))
   return -ENOMEM;
+ sb->s_fs_info = get_net(net);
  return 0;
 }

@@ -1022,11 +1026,19 @@ rpc_mount(struct file_system_type *fs_type,
  return mount_ns(fs_type, flags, current->nsproxy->net_ns, rpc_fill_super);
 }

+void rpc_kill_sb(struct super_block *sb)
+{
+ struct net *net = sb->s_fs_info;
+
+ put_net(net);
+ kill_litter_super(sb);
+}
+
 static struct file_system_type rpc_pipe_fs_type = {
  .owner  = THIS_MODULE,
  .name  = "rpc_pipefs",
  .mount  = rpc_mount,
- .kill_sb = kill_litter_super,
+ .kill_sb = rpc_kill_sb,
 };

 static void
```

Subject: [PATCH v3 5/7] SUNRPC: put pipefs superblock link on network namespace
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:05 GMT
View Forum Message <> Reply to Message

We have modules (like, pNFS blocklayout module) which creates pipes on rpc_pipefs. Thus we need per-net operations for them. To make it possible we require appropriate super block. So we have to put sb link on network namespace context. Note, that it's not strongly required to create pipes in per-net operations. IOW, if pipefs wasn't mounted yet, that no sb link reference will present on network namespace and in this case we need just need to pass through pipe creation. Pipe dentry will be created during pipefs mount notification.

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

```
---
 net/sunrpc/netns.h    |    2 ++
 net/sunrpc/rpc_pipe.c |    4 ++++
 2 files changed, 6 insertions(+), 0 deletions(-)

diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index d013bf2..b384252 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -9,6 +9,8 @@ struct cache_detail;
 struct sunrpc_net {
  struct proc_dir_entry *proc_net_rpc;
  struct cache_detail *ip_map_cache;
+
+ struct super_block *pipefs_sb;
 };

 extern int sunrpc_net_id;
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 1de840d..831e81d 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -1030,6 +1030,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  struct inode *inode;
  struct dentry *root;
  struct net *net = data;
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
  int err;

  sb->s_blocksize = PAGE_CACHE_SIZE;
@@ -1054,6 +1055,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  if (err)
   goto err_depopulate;
  sb->s_fs_info = get_net(net);
```

```
+ sn->pipefs_sb = sb;
  return 0;

 err_depopulate:
@@ -1074,7 +1076,9 @@ rpc_mount(struct file_system_type *fs_type,
 void rpc_kill_sb(struct super_block *sb)
 {
  struct net *net = sb->s_fs_info;
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

+ sn->pipefs_sb = NULL;
  put_net(net);
  blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
        RPC_PIPEFS_UMOUNT,
```

Subject: [PATCH v3 3/7] SUNRPC: send notification events on pipefs sb creation and destruction

Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:05 GMT

View Forum Message <> Reply to Message

They will be used to notify subscribers about pipefs superblock creation and destruction.
Subcribers will have to create their dentries on passed superblock on mount event and destroy otherwise.

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

```
---
 include/linux/sunrpc/rpc_pipe_fs.h |    8 ++++++++
 net/sunrpc/rpc_pipe.c              |   32 ++++++++++++++++++++++++++++++++
 2 files changed, 40 insertions(+), 0 deletions(-)

diff --git a/include/linux/sunrpc/rpc_pipe_fs.h b/include/linux/sunrpc/rpc_pipe_fs.h
index 08aae01..733ef50 100644
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -43,6 +43,14 @@ RPC_I(struct inode *inode)
  return container_of(inode, struct rpc_inode, vfs_inode);
 }

+extern int rpc_pipefs_notifier_register(struct notifier_block *);
+extern void rpc_pipefs_notifier_unregister(struct notifier_block *);
+
+enum {
+ RPC_PIPEFS_MOUNT,
+ RPC_PIPEFS_UMOUNT,
+};
```

```
+
 extern ssize_t rpc_pipe_generic_upcall(struct file *, struct rpc_pipe_msg *,
           char __user *, size_t);
 extern int rpc_queue_upcall(struct inode *, struct rpc_pipe_msg *);
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index ff41fef..07fb7dd 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -28,8 +28,10 @@
 #include <linux/sunrpc/rpc_pipe_fs.h>
 #include <linux/sunrpc/cache.h>
 #include <linux/nsproxy.h>
+#include <linux/notifier.h>

 #include "netns.h"
+#include "sunrpc.h"

 static struct vfsmount *rpc_mnt __read_mostly;
 static int rpc_mount_count;
@@ -41,6 +43,20 @@ static struct kmem_cache *rpc_inode_cachep __read_mostly;

 #define RPC_UPCALL_TIMEOUT (30*HZ)

+static BLOCKING_NOTIFIER_HEAD(rpc_pipefs_notifier_list);
+
+int rpc_pipefs_notifier_register(struct notifier_block *nb)
+{
+ return  blocking_notifier_chain_cond_register(&rpc_pipefs_notifi er_list, nb);
+}
+EXPORT_SYMBOL_GPL(rpc_pipefs_notifier_register);
+
+void rpc_pipefs_notifier_unregister(struct notifier_block *nb)
+{
+  blocking_notifier_chain_unregister(&rpc_pipefs_notifier_ list, nb);
+}
+EXPORT_SYMBOL_GPL(rpc_pipefs_notifier_unregister);
+
 static void rpc_purge_list(struct rpc_inode *rpci, struct list_head *head,
   void (*destroy_msg)(struct rpc_pipe_msg *), int err)
 {
@@ -998,6 +1014,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  struct inode *inode;
  struct dentry *root;
  struct net *net = data;
+ int err;

  sb->s_blocksize = PAGE_CACHE_SIZE;
  sb->s_blocksize_bits = PAGE_CACHE_SHIFT;
```

```
@@ -1015,8 +1032,20 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  }
  if (rpc_populate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF, NULL))
   return -ENOMEM;
+ err = blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
+       RPC_PIPEFS_MOUNT,
+       sb);
+ if (err)
+  goto err_depopulate;
  sb->s_fs_info = get_net(net);
  return 0;
+
+err_depopulate:
+ blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
+       RPC_PIPEFS_UMOUNT,
+       sb);
+ __rpc_depopulate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF);
+ return err;
 }

 static struct dentry *
@@ -1031,6 +1060,9 @@ void rpc_kill_sb(struct super_block *sb)
  struct net *net = sb->s_fs_info;

  put_net(net);
+ blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
+       RPC_PIPEFS_UMOUNT,
+       sb);
  kill_litter_super(sb);
 }
```

Subject: [PATCH v3 4/7] SUNRPC: pipefs dentry lookup helper introduced
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:06 GMT
View Forum Message <> Reply to Message

In all places, where pipefs dentries are created, only directory inode is
actually required to create new dentry. And all this directories has root
pipefs dentry as their parent. So we actually don't need this pipefs mount
point at all if some pipefs lookup method will be provided.
IOW, all we really need is just superblock and simple lookup method to find
root's child dentry with appropriate name. And this patch introduces this
method.
Note, that no locking implemented in rpc_d_lookup_sb(). So it can be used only
in case of assurance, that pipefs superblock still exist. IOW, we can use this
method only in pipefs mount-umount notification subscribers callbacks.

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

```
---
 include/linux/sunrpc/rpc_pipe_fs.h |    3 +++
 net/sunrpc/rpc_pipe.c              |   16 ++++++++++++++++
 2 files changed, 19 insertions(+), 0 deletions(-)

diff --git a/include/linux/sunrpc/rpc_pipe_fs.h b/include/linux/sunrpc/rpc_pipe_fs.h
index 733ef50..4585985 100644
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -51,6 +51,9 @@ enum {
  RPC_PIPEFS_UMOUNT,
 };

+extern struct dentry *rpc_d_lookup_sb(const struct super_block *sb,
+          const unsigned char *dir_name);
+
 extern ssize_t rpc_pipe_generic_upcall(struct file *, struct rpc_pipe_msg *,
           char __user *, size_t);
 extern int rpc_queue_upcall(struct inode *, struct rpc_pipe_msg *);
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 07fb7dd..1de840d 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -1008,6 +1008,22 @@ static const struct rpc_filelist files[] = {
  },
 };

+/*
+ * This call can be used only in RPC pipefs mount notification hooks.
+ */
+struct dentry *rpc_d_lookup_sb(const struct super_block *sb,
+          const unsigned char *dir_name)
+{
+ struct qstr dir = {
+  .name = dir_name,
+  .len = strlen(dir_name),
+  .hash = full_name_hash(dir_name, strlen(dir_name)),
+ };
+
+ return d_lookup(sb->s_root, &dir);
+}
+EXPORT_SYMBOL_GPL(rpc_d_lookup_sb);
+
 static int
 rpc_fill_super(struct super_block *sb, void *data, int silent)
 {
```

Subject: [PATCH v3 6/7] SUNRPC: pipefs per-net operations helper introduced
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:09 GMT
View Forum Message <> Reply to Message

During per-net pipes creation and destruction we have to make sure, that pipefs
sb exists for the whole creation/destruction cycle. This is done by using
special mutex which controls pipefs sb reference on network namespace context.
Helper consists of two parts: first of them (rpc_get_dentry_net) searches for
dentry with specified name and returns with mutex taken on success. When pipe
creation or destructions is completed, caller should release this mutex by
rpc_put_dentry_net call.

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

---
 include/linux/sunrpc/rpc_pipe_fs.h | 3 +++
 net/sunrpc/netns.h | 1 +
 net/sunrpc/rpc_pipe.c | 36 ++++++++++++++++++++++++++++++++++++
 net/sunrpc/sunrpc_syms.c | 1 +
 4 files changed, 41 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/sunrpc/rpc_pipe_fs.h b/include/linux/sunrpc/rpc_pipe_fs.h
index 4585985..f32490c 100644
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -53,6 +53,9 @@ enum {

 extern struct dentry *rpc_d_lookup_sb(const struct super_block *sb,
                const unsigned char *dir_name);
+extern void rpc_pipefs_init_net(struct net *net);
+extern struct super_block *rpc_get_sb_net(const struct net *net);
+extern void rpc_put_sb_net(const struct net *net);

 extern ssize_t rpc_pipe_generic_upcall(struct file *, struct rpc_pipe_msg *,
                char __user *, size_t);
diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index b384252..11d2f48 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -11,6 +11,7 @@ struct sunrpc_net {
  struct cache_detail *ip_map_cache;

  struct super_block *pipefs_sb;
+ struct mutex pipefs_sb_lock;
 };

 extern int sunrpc_net_id;
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 831e81d..e40ec55 100644
```

```
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -1024,6 +1024,40 @@ struct dentry *rpc_d_lookup_sb(const struct super_block *sb,
 }
 EXPORT_SYMBOL_GPL(rpc_d_lookup_sb);

+void rpc_pipefs_init_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+
+ mutex_init(&sn->pipefs_sb_lock);
+}
+
+/*
+ * This call will be used for per network namespace operations calls.
+ * Note: Function will be returned with pipefs_sb_lock taken if superblock was
+ * found. This lock have to be released by rpc_put_sb_net() when all operations
+ * will be completed.
+ */
+struct super_block *rpc_get_sb_net(const struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+
+ mutex_lock(&sn->pipefs_sb_lock);
+ if (sn->pipefs_sb)
+  return sn->pipefs_sb;
+ mutex_unlock(&sn->pipefs_sb_lock);
+ return NULL;
+}
+EXPORT_SYMBOL_GPL(rpc_get_sb_net);
+
+void rpc_put_sb_net(const struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+
+ BUG_ON(sn->pipefs_sb == NULL);
+ mutex_unlock(&sn->pipefs_sb_lock);
+}
+EXPORT_SYMBOL_GPL(rpc_put_sb_net);
+
 static int
 rpc_fill_super(struct super_block *sb, void *data, int silent)
 {
@@ -1078,7 +1112,9 @@ void rpc_kill_sb(struct super_block *sb)
  struct net *net = sb->s_fs_info;
  struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

+ mutex_lock(&sn->pipefs_sb_lock);
```

```
  sn->pipefs_sb = NULL;
+ mutex_unlock(&sn->pipefs_sb_lock);
  put_net(net);
  blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
        RPC_PIPEFS_UMOUNT,
diff --git a/net/sunrpc/sunrpc_syms.c b/net/sunrpc/sunrpc_syms.c
index 8ec9778..7086d11 100644
--- a/net/sunrpc/sunrpc_syms.c
+++ b/net/sunrpc/sunrpc_syms.c
@@ -38,6 +38,7 @@ static __net_init int sunrpc_init_net(struct net *net)
  if (err)
   goto err_ipmap;

+ rpc_pipefs_init_net(net);
  return 0;

 err_ipmap:
```

Subject: [PATCH v3 7/7] SUNRPC: added debug messages to RPC pipefs
Posted by Stanislav Kinsbursky on Mon, 26 Dec 2011 11:41:10 GMT
View Forum Message <> Reply to Message

This patch adds debug messages for notification events.

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

```
---
 net/sunrpc/rpc_pipe.c |    8 ++++++++
 1 files changed, 8 insertions(+), 0 deletions(-)

diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index e40ec55..0cafd59 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -33,6 +33,10 @@
 #include "netns.h"
 #include "sunrpc.h"

+#define RPCDBG_FACILITY RPCDBG_DEBUG
+
+#define NET_NAME(net) ((net == &init_net) ? " (init_net)" : "")
+
 static struct vfsmount *rpc_mnt __read_mostly;
 static int rpc_mount_count;

@@ -1083,6 +1087,8 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  }
```

```
  if (rpc_populate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF, NULL))
   return -ENOMEM;
+ dprintk("RPC: sending pipefs MOUNT notification for net %p%s\n", net,
+         NET_NAME(net));
  err = blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
         RPC_PIPEFS_MOUNT,
         sb);
@@ -1116,6 +1122,8 @@ void rpc_kill_sb(struct super_block *sb)
  sn->pipefs_sb = NULL;
  mutex_unlock(&sn->pipefs_sb_lock);
  put_net(net);
+ dprintk("RPC: sending pipefs UMOUNT notification for net %p%s\n", net,
+         NET_NAME(net));
  blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
         RPC_PIPEFS_UMOUNT,
         sb);
```