
Subject: [PATCH v2 2/5] NFS: blocklayout pipe creation per network namespace context introduced

Posted by [Stanislav Kinsbursky](#) on Tue, 10 Jan 2012 13:04:24 GMT

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

This patch implements blocklayout pipe creation and registration per each existent network namespace.

This was achieved by registering NFS per-net operations, responsible for blocklayout pipe allocation/register and unregister/destruction instead of initialization and destruction of static "bl_device_pipe" pipe (this one was removed).

Note, than pointer to network blocklayout pipe is stored in per-net "nfs_net" structure, because allocating of one more per-net structure for blocklayout module looks redundant.

This patch also changes dev_remove() function prototype (and all it's callers, where it's required) by adding network namespace pointer parameter, which is used to discover proper blocklayout pipe for rpc_queue_upcall() call.

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

```
fs/nfs/blocklayout/blocklayout.c | 54 ++++++-----
fs/nfs/blocklayout/blocklayout.h | 3 +-
fs/nfs/blocklayout/blocklayoutdev.c | 5 +++
fs/nfs/blocklayout/blocklayoutdm.c | 7 +++--
fs/nfs/inode.c | 1 +
fs/nfs/netns.h | 1 +
6 files changed, 47 insertions(+), 24 deletions(-)
```

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

index 489f95c..ce76d05 100644

--- a/fs/nfs/blocklayout/blocklayout.c

+++ b/fs/nfs/blocklayout/blocklayout.c

@@ -46,7 +46,6 @@ MODULE_LICENSE("GPL");

MODULE_AUTHOR("Andy Adamson <andros@citi.umich.edu>");

MODULE_DESCRIPTION("The NFSv4.1 pNFS Block layout driver");

-struct rpc_pipe *bl_device_pipe;

wait_queue_head_t bl_wq;

static void print_page(struct page *page)

@@ -1011,6 +1010,37 @@ static void nfs4blocklayout_unregister_net(struct net *net,

}

}

+static int nfs4blocklayout_net_init(struct net *net)

+{

+ struct nfs_net *nn = net_generic(net, nfs_net_id);

```

+ struct dentry *dentry;
+
+ nn->bl_device_pipe = rpc_mkpipe_data(&bl_upcall_ops, 0);
+ if (IS_ERR(nn->bl_device_pipe))
+ return PTR_ERR(nn->bl_device_pipe);
+ dentry = nfs4blocklayout_register_net(net, nn->bl_device_pipe);
+ if (IS_ERR(dentry)) {
+ rpc_destroy_pipe_data(nn->bl_device_pipe);
+ return PTR_ERR(dentry);
+ }
+ nn->bl_device_pipe->dentry = dentry;
+ return 0;
+}
+
+static void nfs4blocklayout_net_exit(struct net *net)
+{
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+
+ nfs4blocklayout_unregister_net(net, nn->bl_device_pipe);
+ rpc_destroy_pipe_data(nn->bl_device_pipe);
+ nn->bl_device_pipe = NULL;
+}
+
+static struct pernet_operations nfs4blocklayout_net_ops = {
+ .init = nfs4blocklayout_net_init,
+ .exit = nfs4blocklayout_net_exit,
+};
+
+static int __init nfs4blocklayout_init(void)
+{
+ struct vfsmount *mnt;
@@ -1029,24 +1059,12 @@ static int __init nfs4blocklayout_init(void)
+ ret = PTR_ERR(mnt);
+ goto out_remove;
+ }
- bl_device_pipe = rpc_mkpipe_data(&bl_upcall_ops, 0);
- if (IS_ERR(bl_device_pipe)) {
- ret = PTR_ERR(bl_device_pipe);
- goto out_putrpc;
- }
- bl_device_pipe->dentry = nfs4blocklayout_register_net(&init_net,
- bl_device_pipe);
- if (IS_ERR(bl_device_pipe->dentry)) {
- ret = PTR_ERR(bl_device_pipe->dentry);
- goto out_destroy_pipe;
- }
+ ret = register_pernet_subsys(&nfs4blocklayout_net_ops);
+ if (ret)

```

```

+ goto out_remove;
out:
return ret;

-out_destroy_pipe:
- rpc_destroy_pipe_data(bl_device_pipe);
-out_putrpc:
- rpc_put_mount();
out_remove:
pnfs_unregister_layoutdriver(&blocklayout_type);
return ret;
@@ -1057,10 +1075,8 @@ static void __exit nfs4blocklayout_exit(void)
dprntk("%s: NFSv4 Block Layout Driver Unregistering...\n",
__func__);

+ unregister_pernet_subsys(&nfs4blocklayout_net_ops);
pnfs_unregister_layoutdriver(&blocklayout_type);
- nfs4blocklayout_unregister_net(&init_net, bl_device_pipe);
- rpc_destroy_pipe_data(bl_device_pipe);
- rpc_put_mount();
}

MODULE_ALIAS("nfs-layouttype4-3");
diff --git a/fs/nfs/blocklayout/blocklayout.h b/fs/nfs/blocklayout/blocklayout.h
index 046b513..10e0a62 100644
--- a/fs/nfs/blocklayout/blocklayout.h
+++ b/fs/nfs/blocklayout/blocklayout.h
@@ -37,6 +37,7 @@
#include <linux/sunrpc/rpc_pipe_fs.h>

#include "../pnfs.h"
+#include "../netns.h"

#define PAGE_CACHE_SECTORS (PAGE_CACHE_SIZE >> SECTOR_SHIFT)
#define PAGE_CACHE_SECTOR_SHIFT (PAGE_CACHE_SHIFT - SECTOR_SHIFT)
@@ -50,6 +51,7 @@ struct pnfs_block_dev {
struct list_head bm_node;
struct nfs4_deviceid bm_mdevid; /* associated devid */
struct block_device *bm_mdev; /* meta device itself */
+ struct net *net;
};

enum exstate4 {
@@ -159,7 +161,6 @@ struct bl_msg_hdr {
u16 totallen; /* length of entire message, including hdr itself */
};

-extern struct rpc_pipe *bl_device_pipe;

```

```
extern wait_queue_head_t bl_wq;
```

```
#define BL_DEVICE_UMOUNT          0x0 /* Umount--delete devices */  
diff --git a/fs/nfs/blocklayout/blocklayoutdev.c b/fs/nfs/blocklayout/blocklayoutdev.c  
index 949b624..94ed978 100644
```

```
--- a/fs/nfs/blocklayout/blocklayoutdev.c  
+++ b/fs/nfs/blocklayout/blocklayoutdev.c  
@@ -120,6 +120,8 @@ nfs4_blk_decode_device(struct nfs_server *server,  
    DECLARE_WAITQUEUE(wq, current);  
    struct bl_dev_msg *reply = &bl_mount_reply;  
    int offset, len, i, rc;  
+ struct net *net = server->nfs_client->net;  
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
```

```
    dprintk("%s CREATING PIPEFS MESSAGE\n", __func__);  
    dprintk("%s: deviceid: %s, mincount: %d\n", __func__, dev->dev_id.data,  
@@ -146,7 +148,7 @@ nfs4_blk_decode_device(struct nfs_server *server,
```

```
    dprintk("%s CALLING USERSPACE DAEMON\n", __func__);  
    add_wait_queue(&bl_wq, &wq);  
- rc = rpc_queue_upcall(bl_device_pipe, &msg);  
+ rc = rpc_queue_upcall(nn->bl_device_pipe, &msg);  
    if (rc < 0) {  
        remove_wait_queue(&bl_wq, &wq);  
        rv = ERR_PTR(rc);  
@@ -181,6 +183,7 @@ nfs4_blk_decode_device(struct nfs_server *server,
```

```
    rv->bm_mdev = bd;  
    memcpy(&rv->bm_mdevid, &dev->dev_id, sizeof(struct nfs4_deviceid));  
+ rv->net = net;  
    dprintk("%s Created device %s with bd_block_size %u\n",  
        __func__,  
        bd->bd_disk->disk_name,  
diff --git a/fs/nfs/blocklayout/blocklayoutdm.c b/fs/nfs/blocklayout/blocklayoutdm.c  
index 631f254..970490f 100644
```

```
--- a/fs/nfs/blocklayout/blocklayoutdm.c  
+++ b/fs/nfs/blocklayout/blocklayoutdm.c  
@@ -38,7 +38,7 @@
```

```
#define NFSDBG_FACILITY      NFSDBG_PNFS_LD
```

```
-static void dev_remove(dev_t dev)  
+static void dev_remove(struct net *net, dev_t dev)  
{  
    struct rpc_pipe_msg msg;  
    struct bl_dev_msg bl_umount_request;  
@@ -48,6 +48,7 @@ static void dev_remove(dev_t dev)  
};
```

```

uint8_t *dataptr;
DECLARE_WAITQUEUE(wq, current);
+ struct nfs_net *nn = net_generic(net, nfs_net_id);

dprintk("Entering %s\n", __func__);

@@ -66,7 +67,7 @@ static void dev_remove(dev_t dev)
    msg.len = sizeof(bl_msg) + bl_msg.totallen;

    add_wait_queue(&bl_wq, &wq);
- if (rpc_queue_upcall(bl_device_pipe, &msg) < 0) {
+ if (rpc_queue_upcall(nn->bl_device_pipe, &msg) < 0) {
    remove_wait_queue(&bl_wq, &wq);
    goto out;
}
@@ -93,7 +94,7 @@ static void nfs4_blk_metadev_release(struct pnfs_block_dev *bdev)
    printk(KERN_ERR "%s nfs4_blkdev_put returns %d\n",
        __func__, rv);

- dev_remove(bdev->bm_mdev->bd_dev);
+ dev_remove(bdev->net, bdev->bm_mdev->bd_dev);
}

void bl_free_block_dev(struct pnfs_block_dev *bdev)
diff --git a/fs/nfs/inode.c b/fs/nfs/inode.c
index 9590da3..60cf71a 100644
--- a/fs/nfs/inode.c
+++ b/fs/nfs/inode.c
@@ -1552,6 +1552,7 @@ static void nfsiod_stop(void)
}

int nfs_net_id;
+EXPORT_SYMBOL_GPL(nfs_net_id);

static int nfs_net_init(struct net *net)
{
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index 8c1f130..39ae4ca 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -6,6 +6,7 @@

struct nfs_net {
    struct cache_detail *nfs_dns_resolve;
+ struct rpc_pipe *bl_device_pipe;
};

extern int nfs_net_id;

```
