Subject: [PATCH 2/2] NFS: replace global bl_wq with per-net one
Posted by Stanislav Kinsbursky on Sun, 11 Mar 2012 14:20:31 GMT
View Forum Message <> Reply to Message

This queue is used for sleeping in kernel and it have to be per-net since we
don't want to wake any other waiters except in out network nemespace.
BTW, move wq to per-net data is easy. But some way to handle upcall timeouts
have to be provided. On message destroy in case of timeout, tasks, waiting for
message to be delivered, should be awakened. Thus, some data required to
located the right wait queue. Chosen solution replaces rpc_pipe_msg object with
new introduced bl_pipe_msg object, containing rpc_pipe_msg and proper wq.

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

---
 fs/nfs/blocklayout/blocklayout.c | 4 +---
 fs/nfs/blocklayout/blocklayout.h | 7 +++++--
 fs/nfs/blocklayout/blocklayoutdev.c | 32 ++++++++++++++++++++++----------
 fs/nfs/blocklayout/blocklayoutdm.c | 26 ++++++++++++++++++++------
 fs/nfs/netns.h | 1 +
 5 files changed, 39 insertions(+), 31 deletions(-)

diff --git a/fs/nfs/blocklayout/blocklayout.c b/fs/nfs/blocklayout/blocklayout.c
index 783ebd5..6150134 100644
--- a/fs/nfs/blocklayout/blocklayout.c
+++ b/fs/nfs/blocklayout/blocklayout.c
@@ -46,8 +46,6 @@ MODULE_LICENSE("GPL");
 MODULE_AUTHOR("Andy Adamson <andros@citi.umich.edu>");
 MODULE_DESCRIPTION("The NFSv4.1 pNFS Block layout driver");

-wait_queue_head_t bl_wq;
-
 static void print_page(struct page *page)
 {
  dprintk("PRINTPAGE page %p\n", page);
@@ -1117,6 +1115,7 @@ static int nfs4blocklayout_net_init(struct net *net)
  struct nfs_net *nn = net_generic(net, nfs_net_id);
  struct dentry *dentry;

+ init_waitqueue_head(&nn->bl_wq);
  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);
@@ -1153,7 +1152,6 @@ static int __init nfs4blocklayout_init(void)
  if (ret)
   goto out;

- init_waitqueue_head(&bl_wq);

```
  ret = rpc_pipefs_notifier_register(&nfs4blocklayout_block);
  if (ret)
   goto out_remove;
diff --git a/fs/nfs/blocklayout/blocklayout.h b/fs/nfs/blocklayout/blocklayout.h
index 58ac861..0335069 100644
--- a/fs/nfs/blocklayout/blocklayout.h
+++ b/fs/nfs/blocklayout/blocklayout.h
@@ -153,13 +153,16 @@ BLK_LSEG2EXT(struct pnfs_layout_segment *lseg)
  return BLK_LO2EXT(lseg->pls_layout);
 }

+struct bl_pipe_msg {
+ struct rpc_pipe_msg msg;
+ wait_queue_head_t *bl_wq;
+};
+
 struct bl_msg_hdr {
  u8  type;
  u16 totallen; /* length of entire message, including hdr itself */
 };

-extern wait_queue_head_t bl_wq;
-
 #define BL_DEVICE_UMOUNT               0x0 /* Umount--delete devices */
 #define BL_DEVICE_MOUNT                0x1 /* Mount--create devices*/
 #define BL_DEVICE_REQUEST_INIT         0x0 /* Start request */
diff --git a/fs/nfs/blocklayout/blocklayoutdev.c b/fs/nfs/blocklayout/blocklayoutdev.c
index 1d58642..a5c88a5 100644
--- a/fs/nfs/blocklayout/blocklayoutdev.c
+++ b/fs/nfs/blocklayout/blocklayoutdev.c
@@ -91,16 +91,18 @@ ssize_t bl_pipe_downcall(struct file *filp, const char __user *src,
  if (copy_from_user(&nn->bl_mount_reply, src, mlen) != 0)
   return -EFAULT;

- wake_up(&bl_wq);
+ wake_up(&nn->bl_wq);

  return mlen;
 }

 void bl_pipe_destroy_msg(struct rpc_pipe_msg *msg)
 {
+ struct bl_pipe_msg *bl_pipe_msg = container_of(msg, struct bl_pipe_msg, msg);
+
  if (msg->errno >= 0)
   return;
- wake_up(&bl_wq);
+ wake_up(bl_pipe_msg->bl_wq);
```

```
 }

 /*
@@ -112,7 +114,8 @@ nfs4_blk_decode_device(struct nfs_server *server,
 {
  struct pnfs_block_dev *rv;
  struct block_device *bd = NULL;
- struct rpc_pipe_msg msg;
+ struct bl_pipe_msg bl_pipe_msg;
+ struct rpc_pipe_msg *msg = &bl_pipe_msg.msg;
  struct bl_msg_hdr bl_msg = {
   .type = BL_DEVICE_MOUNT,
   .totallen = dev->mincount,
@@ -128,15 +131,16 @@ nfs4_blk_decode_device(struct nfs_server *server,
  dprintk("%s: deviceid: %s, mincount: %d\n", __func__, dev->dev_id.data,
   dev->mincount);

- memset(&msg, 0, sizeof(msg));
- msg.data = kzalloc(sizeof(bl_msg) + dev->mincount, GFP_NOFS);
- if (!msg.data) {
+ bl_pipe_msg.bl_wq = &nn->bl_wq;
+ memset(msg, 0, sizeof(*msg));
+ msg->data = kzalloc(sizeof(bl_msg) + dev->mincount, GFP_NOFS);
+ if (!msg->data) {
   rv = ERR_PTR(-ENOMEM);
   goto out;
  }

- memcpy(msg.data, &bl_msg, sizeof(bl_msg));
- dataptr = (uint8_t *) msg.data;
+ memcpy(msg->data, &bl_msg, sizeof(bl_msg));
+ dataptr = (uint8_t *) msg->data;
  len = dev->mincount;
  offset = sizeof(bl_msg);
  for (i = 0; len > 0; i++) {
@@ -145,13 +149,13 @@ nfs4_blk_decode_device(struct nfs_server *server,
   len -= PAGE_CACHE_SIZE;
   offset += PAGE_CACHE_SIZE;
  }
- msg.len = sizeof(bl_msg) + dev->mincount;
+ msg->len = sizeof(bl_msg) + dev->mincount;

  dprintk("%s CALLING USERSPACE DAEMON\n", __func__);
- add_wait_queue(&bl_wq, &wq);
- rc = rpc_queue_upcall(nn->bl_device_pipe, &msg);
+ add_wait_queue(&nn->bl_wq, &wq);
+ rc = rpc_queue_upcall(nn->bl_device_pipe, msg);
  if (rc < 0) {
```

```
-  remove_wait_queue(&bl_wq, &wq);
+  remove_wait_queue(&nn->bl_wq, &wq);
   rv = ERR_PTR(rc);
   goto out;
  }
@@ -159,7 +163,7 @@ nfs4_blk_decode_device(struct nfs_server *server,
  set_current_state(TASK_UNINTERRUPTIBLE);
  schedule();
  __set_current_state(TASK_RUNNING);
- remove_wait_queue(&bl_wq, &wq);
+ remove_wait_queue(&nn->bl_wq, &wq);

  if (reply->status != BL_DEVICE_REQUEST_PROC) {
   dprintk("%s failed to open device: %d\n",
@@ -191,7 +195,7 @@ nfs4_blk_decode_device(struct nfs_server *server,
   bd->bd_block_size);

 out:
- kfree(msg.data);
+ kfree(msg->data);
  return rv;
 }

diff --git a/fs/nfs/blocklayout/blocklayoutdm.c b/fs/nfs/blocklayout/blocklayoutdm.c
index a0f588f..30fc22a 100644
--- a/fs/nfs/blocklayout/blocklayoutdm.c
+++ b/fs/nfs/blocklayout/blocklayoutdm.c
@@ -40,7 +40,8 @@

 static void dev_remove(struct net *net, dev_t dev)
 {
- struct rpc_pipe_msg msg;
+ struct bl_pipe_msg bl_pipe_msg;
+ struct rpc_pipe_msg *msg = &bl_pipe_msg.msg;
  struct bl_dev_msg bl_umount_request;
  struct bl_msg_hdr bl_msg = {
   .type = BL_DEVICE_UMOUNT,
@@ -52,33 +53,34 @@ static void dev_remove(struct net *net, dev_t dev)

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

- memset(&msg, 0, sizeof(msg));
- msg.data = kzalloc(1 + sizeof(bl_umount_request), GFP_NOFS);
- if (!msg.data)
+ bl_pipe_msg.bl_wq = &nn->bl_wq;
+ memset(&msg, 0, sizeof(*msg));
+ msg->data = kzalloc(1 + sizeof(bl_umount_request), GFP_NOFS);
+ if (!msg->data)
```

```
   goto out;

  memset(&bl_umount_request, 0, sizeof(bl_umount_request));
  bl_umount_request.major = MAJOR(dev);
  bl_umount_request.minor = MINOR(dev);

- memcpy(msg.data, &bl_msg, sizeof(bl_msg));
- dataptr = (uint8_t *) msg.data;
+ memcpy(msg->data, &bl_msg, sizeof(bl_msg));
+ dataptr = (uint8_t *) msg->data;
  memcpy(&dataptr[sizeof(bl_msg)], &bl_umount_request, sizeof(bl_umount_request));
- msg.len = sizeof(bl_msg) + bl_msg.totallen;
+ msg->len = sizeof(bl_msg) + bl_msg.totallen;

- add_wait_queue(&bl_wq, &wq);
- if (rpc_queue_upcall(nn->bl_device_pipe, &msg) < 0) {
-  remove_wait_queue(&bl_wq, &wq);
+ add_wait_queue(&nn->bl_wq, &wq);
+ if (rpc_queue_upcall(nn->bl_device_pipe, msg) < 0) {
+  remove_wait_queue(&nn->bl_wq, &wq);
   goto out;
  }

  set_current_state(TASK_UNINTERRUPTIBLE);
  schedule();
  __set_current_state(TASK_RUNNING);
- remove_wait_queue(&bl_wq, &wq);
+ remove_wait_queue(&nn->bl_wq, &wq);

 out:
- kfree(msg.data);
+ kfree(msg->data);
 }

 /*
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index 73425f5..aa14ec3 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -13,6 +13,7 @@ struct nfs_net {
  struct cache_detail *nfs_dns_resolve;
  struct rpc_pipe *bl_device_pipe;
  struct bl_dev_msg bl_mount_reply;
+ wait_queue_head_t bl_wq;
  struct list_head nfs_client_list;
  struct list_head nfs_volume_list;
 #ifdef CONFIG_NFS_V4
```