

UPDATED: Re: [NFS] Re: [PATCH v2] reduce NFS stack usage

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-09/7738.html>

From: Steve Dickson (*SteveD_at_redhat.com*)

Date: 09/30/03

Date: Tue, 30 Sep 2003 11:31:47 -0400
To: Trond Myklebust <trond.myklebust@fys.uio.no>

While doing some testing with this patch I found a hole in one of the return paths...

```
diff -u linux-2.4/fs/nfs/dir.c linux-2.4/fs/nfs/dir.c
--- linux-2.4/fs/nfs/dir.c 2003-09-30 10:34:46.000000000 -0400
+++ linux-2.4/fs/nfs/dir.c 2003-09-29 20:36:46.000000000 -0400
@@ -572,7 +572,6 @@
     nfs_renew_times(dentry);
 out_valid:
     unlock_kernel();
+ nfs_dirop_free(fhandle, fattr);
     return 1;
 out_bad:
     NFS_CACHEINV(dir);
```

Attached is the updated patch...

SteveD.

```
--- linux-2.4/fs/nfs/inode.c.orig 2003-09-29 13:39:47.000000000 -0400
+++ linux-2.4/fs/nfs/inode.c 2003-09-29 20:15:12.000000000 -0400
@@ -1133,6 +1133,8 @@ extern int nfs_init_readpagecache(void);
extern int nfs_destroy_readpagecache(void);
extern int nfs_init_writepagecache(void);
extern int nfs_destroy_writepagecache(void);
+extern int nfs_init_diroppagecache(void);
+extern void nfs_destroy_diroppagecache(void);

/*
 * Initialize NFS
@@ -1153,6 +1155,10 @@ static int __init init_nfs_fs(void)
```

```

    if (err)
        return err;

+ err = nfs_init_diroppagecache();
+ if (err)
+ return err;
+
+ #ifdef CONFIG_PROC_FS
+     rpc_proc_register(&nfs_rpcstat);
+ #endif
@@ -1161,6 +1167,7 @@ static int __init init_nfs_fs(void)

static void __exit exit_nfs_fs(void)
{
+ nfs_destroy_diroppagecache();
+     nfs_destroy_writepagecache();
+     nfs_destroy_readpagecache();
+     nfs_destroy_nfspagecache();
--- linux-2.4/fs/nfs/dir.c.orig 2003-09-29 13:39:47.000000000 -0400
+++ linux-2.4/fs/nfs/dir.c 2003-09-30 10:34:46.000000000 -0400
@@ -69,6 +69,51 @@ struct inode_operations nfs_dir_inode_op
+     revalidate: nfs_revalidate,
+     setattr: nfs_notify_change,
+ };
+static kmem_cache_t *nfs_ddata_cachep;
+
+int nfs_init_diroppagecache(void)
+{
+ nfs_ddata_cachep = kmem_cache_create("nfs_diropt_data",
+ sizeof(struct nfs_fh) + sizeof(struct nfs_fattr),
+ 0, SLAB_HWCACHE_ALIGN,
+ NULL, NULL);
+ if (nfs_ddata_cachep == NULL)
+ return -ENOMEM;
+
+ return 0;
+}
+
+void nfs_destroy_diroppagecache(void)
+{
+ if (kmem_cache_destroy(nfs_ddata_cachep))
+ printk(KERN_INFO "nfs_diropt_data: not all structures were freed\n");
+}
+static __inline__ int
+nfs_diropt_alloc(struct nfs_fh **fh, struct nfs_fattr **fattr)
+{
+ void *mem;
+ static int len = sizeof(struct nfs_fh) + sizeof(struct nfs_fattr);
+
+ mem = kmem_cache_alloc(nfs_ddata_cachep, GFP_NOFS);
+ if (mem == NULL)

```

```

+ return -ENOMEM;
+
+ memset(mem, 0, len);
+ *fh = (struct nfs_fh *)mem;
+ *fattr = (struct nfs_fattr *) (mem + sizeof(struct nfs_fh));
+
+ return 0;
+}
+static __inline__ void
+nfs_dirop_free(struct nfs_fh *fh, struct nfs_fattr *fattr)
+{
+ void *mem = ((void *)fh < (void *)fattr ?
+ (void *)fh : (void *)fattr);
+
+ kmem_cache_free(nfs_ddata_cachep, mem);
+
+ return;
+}

typedef u32 * (*decode_dirent_t)(u32 *, struct nfs_entry *, int);
typedef struct {
@@ -352,12 +397,16 @@ static int nfs_readdir(struct file *filp
    nfs_readdir_descriptor_t my_desc,
        *desc = &my_desc;
    struct nfs_entry my_entry;
+ struct nfs_fh *fh;
+ struct nfs_fattr *fattr;
    long res;

    res = nfs_revalidate(dentry);
    if (res < 0)
        return res;

+ if (nfs_dirop_alloc(&fh, &fattr))
+ return -ENOMEM;
    /*
     * filp->f_pos points to the file offset in the page cache.
     * but if the cache has meanwhile been zapped, we need to
@@ -366,6 +415,8 @@ static int nfs_readdir(struct file *filp
    */
    memset(desc, 0, sizeof(*desc));
    memset(&my_entry, 0, sizeof(my_entry));
+ my_entry.fh = fh;
+ my_entry.fattr = fattr;

    desc->file = filp;
    desc->target = filp->f_pos;
@@ -393,6 +444,8 @@ static int nfs_readdir(struct file *filp
        break;
    }
}

```

```

+ nfs_dirop_free(fh, fattr);
+
+   if (desc->error < 0)
+       return desc->error;
+   if (res < 0)
@@ -476,8 +529,11 @@ static int nfs_lookup_revalidate(struct
+   struct inode *dir;
+   struct inode *inode;
+   int error;
- struct nfs_fh fhandle;
- struct nfs_fattr fattr;
+ struct nfs_fh *fhandle;
+ struct nfs_fattr *fattr;
+
+ if (nfs_dirop_alloc(&fhandle, &fattr))
+ return 0;

+   lock_kernel();
+   dir = dentry->d_parent->d_inode;
@@ -505,17 +561,18 @@ static int nfs_lookup_revalidate(struct
+   if (NFS_STALE(inode))
+       goto out_bad;

- error = NFS_PROTO(dir)->lookup(dir, &dentry->d_name, &fhandle, &fattr);
+ error = NFS_PROTO(dir)->lookup(dir, &dentry->d_name, fhandle, fattr);
+   if (error)
+       goto out_bad;
- if (memcmp(NFS_FH(inode), &fhandle, sizeof(struct nfs_fh))!= 0)
+ if (memcmp(NFS_FH(inode), fhandle, sizeof(struct nfs_fh))!= 0)
+       goto out_bad;
- if ((error = nfs_refresh_inode(inode, &fattr)) != 0)
+ if ((error = nfs_refresh_inode(inode, fattr)) != 0)
+       goto out_bad;

+   nfs_renew_times(dentry);
+ out_valid:
+   unlock_kernel();
+ nfs_dirop_free(fhandle, fattr);
+   return 1;
+ out_bad:
+   NFS_CACHEINV(dir);
@@ -529,6 +586,7 @@ static int nfs_lookup_revalidate(struct
+   }
+   d_drop(dentry);
+   unlock_kernel();
+ nfs_dirop_free(fhandle, fattr);
+   return 0;
+ }

@@ -575,8 +633,8 @@ static struct dentry *nfs_lookup(struct
+ {

```

```

    struct inode *inode;
    int error;
- struct nfs_fh fhandle;
- struct nfs_fattr fattr;
+ struct nfs_fh *fhandle;
+ struct nfs_fattr *fattr;

    dfprintk(VFS, "NFS: lookup(%s/%s)\n",
             dentry->d_parent->d_name.name, dentry->d_name.name);
@@ -586,15 +644,17 @@ static struct dentry *nfs_lookup(struct
    goto out;

    error = -ENOMEM;
+ if (nfs_dirop_alloc(&fhandle, &fattr))
+ goto out;
    dentry->d_op = &nfs_dentry_operations;

- error = NFS_PROTO(dir)->lookup(dir, &dentry->d_name, &fhandle, &fattr);
+ error = NFS_PROTO(dir)->lookup(dir, &dentry->d_name, fhandle, fattr);
    inode = NULL;
    if (error == -ENOENT)
        goto no_entry;
    if (!error) {
        error = -EACCES;
- inode = nfs_fhget(dentry, &fhandle, &fattr);
+ inode = nfs_fhget(dentry, fhandle, fattr);
        if (inode) {
            no_entry:
                d_add(dentry, inode);
@@ -603,6 +663,7 @@ static struct dentry *nfs_lookup(struct
            nfs_renew_times(dentry);
        }
    out:
+ nfs_dirop_free(fhandle, fattr);
    return ERR_PTR(error);
}

@@ -642,13 +703,16 @@ out_err:
static int nfs_create(struct inode *dir, struct dentry *dentry, int mode)
{
    struct iattr attr;
- struct nfs_fattr fattr;
- struct nfs_fh fhandle;
+ struct nfs_fattr *fattr;
+ struct nfs_fh *fhandle;
    int error;

    dfprintk(VFS, "NFS: create(%x/%ld, %s\n",
             dir->i_dev, dir->i_ino, dentry->d_name.name);

+ if (nfs_dirop_alloc(&fhandle, &fattr))

```

```

+ return -ENOMEM;
+
    attr.ia_mode = mode;
    attr.ia_valid = ATTR_MODE;

@@ -660,11 +724,13 @@ static int nfs_create(struct inode *dir,
    */
    nfs_zap_caches(dir);
    error = NFS_PROTO(dir)->create(dir, &dentry->d_name,
- &attr, 0, &fhandle, &fattr);
+ &attr, 0, fhandle, fattr);
    if (!error)
- error = nfs_instantiate(dentry, &fhandle, &fattr);
+ error = nfs_instantiate(dentry, fhandle, fattr);
    else
        d_drop(dentry);
+
+ nfs_dirop_free(fhandle, fattr);
    return error;
}

@@ -674,23 +740,28 @@ static int nfs_create(struct inode *dir,
static int nfs_mknod(struct inode *dir, struct dentry *dentry, int mode, int rdev)
{
    struct iattr attr;
- struct nfs_fattr fattr;
- struct nfs_fh fhandle;
+ struct nfs_fattr *fattr;
+ struct nfs_fh *fhandle;
    int error;

    dfprintk(VFS, "NFS: mknod(%x/%ld, %s\n",
        dir->i_dev, dir->i_ino, dentry->d_name.name);

+ if (nfs_dirop_alloc(&fhandle, &fattr))
+ return -ENOMEM;
+
    attr.ia_mode = mode;
    attr.ia_valid = ATTR_MODE;

    nfs_zap_caches(dir);
    error = NFS_PROTO(dir)->mknod(dir, &dentry->d_name, &attr, rdev,
- &fhandle, &fattr);
+ fhandle, fattr);
    if (!error)
- error = nfs_instantiate(dentry, &fhandle, &fattr);
+ error = nfs_instantiate(dentry, fhandle, fattr);
    else
        d_drop(dentry);
+
+ nfs_dirop_free(fhandle, fattr);

```

```

    return error;
}

@@ -700,13 +771,16 @@ static int nfs_mknod(struct inode *dir,
static int nfs_mkdir(struct inode *dir, struct dentry *dentry, int mode)
{
    struct iattr attr;
- struct nfs_fattr fattr;
- struct nfs_fh fhandle;
+ struct nfs_fattr *fattr;
+ struct nfs_fh *fhandle;
    int error;

    dfprintk(VFS, "NFS: mkdir(%x/%ld, %s\n",
              dir->i_dev, dir->i_ino, dentry->d_name.name);

+ if (nfs_dirop_alloc(&fhandle, &fattr))
+ return -ENOMEM;
+
    attr.ia_valid = ATTR_MODE;
    attr.ia_mode = mode | S_IFDIR;

@@ -720,12 +794,14 @@ static int nfs_mkdir(struct inode *dir,
    d_drop(dentry);
#endif
    nfs_zap_caches(dir);
- error = NFS_PROTO(dir)->mkdir(dir, &dentry->d_name, &attr, &fhandle,
- &fattr);
+ error = NFS_PROTO(dir)->mkdir(dir, &dentry->d_name, &attr, fhandle,
+ fattr);
    if (!error)
- error = nfs_instantiate(dentry, &fhandle, &fattr);
+ error = nfs_instantiate(dentry, fhandle, fattr);
    else
        d_drop(dentry);
+
+ nfs_dirop_free(fhandle, fattr);
    return error;
}

--- linux-2.4/fs/nfs/nfs3xdr.c.orig 2003-09-29 13:39:47.000000000 -0400
+++ linux-2.4/fs/nfs/nfs3xdr.c 2003-09-29 20:37:09.000000000 -0400
@@ -616,10 +616,10 @@ nfs3_decode_dirent(u32 *p, struct nfs_en
    p = xdr_decode_hyper(p, &entry->cookie);

    if (plus) {
- p = xdr_decode_post_op_attr(p, &entry->fattr);
+ p = xdr_decode_post_op_attr(p, entry->fattr);
        /* In fact, a post_op_fh3: */
        if (*p++) {
- p = xdr_decode_fhandle(p, &entry->fh);

```

Linux-Kernel: UPDATED: Re: [NFS] Re: [PATCH v2] reduce NFS stack usage

```
+ p = xdr_decode_fhandle(p, entry->fh);
    /* Ugh --- server reply was truncated */
    if (p == NULL) {
        dprintk("NFS: FH truncated\n");
@@ -629,7 +629,7 @@ nfs3_decode_dirent(u32 *p, struct nfs_en
    } else {
        /* If we don't get a file handle, the attrs
         * aren't worth a lot. */
- entry->fattr.valid = 0;
+ entry->fattr->valid = 0;
    }
}

--- linux-2.4/include/linux/nfs_xdr.h.orig 2003-09-29 13:40:08.000000000 -0400
+++ linux-2.4/include/linux/nfs_xdr.h 2003-09-29 20:18:20.000000000 -0400
@@ -109,8 +109,8 @@ struct nfs_entry {
    const char * name;
    unsigned int len;
    int eof;
- struct nfs_fh fh;
- struct nfs_fattr fattr;
+ struct nfs_fh *fh;
+ struct nfs_fattr *fattr;
};

/*
```

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>