Subject: [PATCH v3, RESEND 10/16] nfs: per-rpc_pipefs dns cache
Posted by Kirill A. Shutemov on Tue, 08 Feb 2011 18:42:01 GMT
View Forum Message <> Reply to Message

Lazy initialization of dns cache: on first call nfs_dns_resolve_name().
Every rpc_pipefs has separate dns cache now.

Signed-off-by: Kirill A. Shutemov <kas@openvz.org>
Reviewed-by: Rob Landley <rlandley@parallels.com>

```
---
 fs/nfs/cache_lib.c     |   17 ++-----
 fs/nfs/cache_lib.h     |    3 +-
 fs/nfs/dns_resolve.c   |  137 ++++++++++++++++++++++++++++++++++++++-----------
 fs/nfs/dns_resolve.h   |   15 +-----
 fs/nfs/inode.c         |    9 +---
 fs/nfs/nfs4namespace.c |    4 +-
 6 files changed, 117 insertions(+), 68 deletions(-)

diff --git a/fs/nfs/cache_lib.c b/fs/nfs/cache_lib.c
index 0944d4e..9b99d9e 100644
--- a/fs/nfs/cache_lib.c
+++ b/fs/nfs/cache_lib.c
@@ -12,7 +12,6 @@
 #include <linux/namei.h>
 #include <linux/slab.h>
 #include <linux/sunrpc/cache.h>
-#include <linux/sunrpc/rpc_pipe_fs.h>

 #include "cache_lib.h"

@@ -111,25 +110,17 @@ int nfs_cache_wait_for_upcall(struct nfs_cache_defer_req *dreq)
  return 0;
 }

-int nfs_cache_register(struct cache_detail *cd)
+int nfs_cache_register(struct cache_detail *cd, struct vfsmount *rpcmount)
 {
  struct nameidata nd;
- struct vfsmount *mnt;
  int ret;

- mnt = mntget(init_rpc_pipefs);
- if (IS_ERR(mnt))
-  return PTR_ERR(mnt);
- ret = vfs_path_lookup(mnt->mnt_root, mnt, "/cache", 0, &nd);
+ ret = vfs_path_lookup(rpcmount->mnt_root, rpcmount, "/cache", 0, &nd);
  if (ret)
-  goto err;
```

```
- ret = sunrpc_cache_register_pipefs(mnt, nd.path.dentry,
+  return ret;
+ ret = sunrpc_cache_register_pipefs(rpcmount, nd.path.dentry,
    cd->name, 0600, cd);
  path_put(&nd.path);
- if (!ret)
-  return ret;
-err:
- mntput(mnt);
  return ret;
 }

diff --git a/fs/nfs/cache_lib.h b/fs/nfs/cache_lib.h
index 76f856e..1d4a0a5 100644
--- a/fs/nfs/cache_lib.h
+++ b/fs/nfs/cache_lib.h
@@ -23,5 +23,6 @@ extern struct nfs_cache_defer_req *nfs_cache_defer_req_alloc(void);
 extern void nfs_cache_defer_req_put(struct nfs_cache_defer_req *dreq);
 extern int nfs_cache_wait_for_upcall(struct nfs_cache_defer_req *dreq);

-extern int nfs_cache_register(struct cache_detail *cd);
+extern int nfs_cache_register(struct cache_detail *cd,
+  struct vfsmount *rpcmount);
 extern void nfs_cache_unregister(struct cache_detail *cd);
diff --git a/fs/nfs/dns_resolve.c b/fs/nfs/dns_resolve.c
index a6e711a..a832e64 100644
--- a/fs/nfs/dns_resolve.c
+++ b/fs/nfs/dns_resolve.c
@@ -12,7 +12,7 @@
 #include <linux/dns_resolver.h>

 ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
-  struct sockaddr *sa, size_t salen)
+  struct sockaddr *sa, size_t salen, struct vfsmount *rpcmount)
 {
  ssize_t ret;
  char *ip_addr = NULL;
@@ -37,9 +37,11 @@ ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
 #include <linux/socket.h>
 #include <linux/seq_file.h>
 #include <linux/inet.h>
+#include <linux/mount.h>
 #include <linux/sunrpc/clnt.h>
 #include <linux/sunrpc/cache.h>
 #include <linux/sunrpc/svcauth.h>
+#include <linux/sunrpc/rpc_pipe_fs.h>

 #include "dns_resolve.h"
```

```
 #include "cache_lib.h"
@@ -47,7 +49,13 @@ ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
 #define NFS_DNS_HASHBITS 4
 #define NFS_DNS_HASHTBL_SIZE (1 << NFS_DNS_HASHBITS)

-static struct cache_head *nfs_dns_table[NFS_DNS_HASHTBL_SIZE];
+static DEFINE_SPINLOCK(nfs_dns_resolve_lock);
+static LIST_HEAD(nfs_dns_resolve_list);
+
+struct nfs_dns_resolve_list {
+ struct list_head list;
+ struct cache_detail *cd;
+};

 struct nfs_dns_ent {
  struct cache_head h;
@@ -259,21 +267,6 @@ out:
  return ret;
 }

-static struct cache_detail nfs_dns_resolve = {
- .owner = THIS_MODULE,
- .hash_size = NFS_DNS_HASHTBL_SIZE,
- .hash_table = nfs_dns_table,
- .name = "dns_resolve",
- .cache_put = nfs_dns_ent_put,
- .cache_upcall = nfs_dns_upcall,
- .cache_parse = nfs_dns_parse,
- .cache_show = nfs_dns_show,
- .match = nfs_dns_match,
- .init = nfs_dns_ent_init,
- .update = nfs_dns_ent_update,
- .alloc = nfs_dns_ent_alloc,
-};
-
 static int do_cache_lookup(struct cache_detail *cd,
   struct nfs_dns_ent *key,
   struct nfs_dns_ent **item,
@@ -336,37 +329,119 @@ out:
  return ret;
 }

+static struct cache_detail *nfs_alloc_dns_resolve(void)
+{
+ struct cache_detail *dns_resolve;
+ struct cache_head **hash_table;
+
+ dns_resolve = kmalloc(sizeof(*dns_resolve), GFP_KERNEL);
```

```
+ if (!dns_resolve)
+  return NULL;
+
+ hash_table = kmalloc(sizeof(*hash_table) * NFS_DNS_HASHTBL_SIZE,
+   GFP_KERNEL);
+ if (!hash_table) {
+  kfree(dns_resolve);
+  return NULL;
+ }
+
+ dns_resolve->owner = THIS_MODULE;
+ dns_resolve->hash_size = NFS_DNS_HASHTBL_SIZE;
+ dns_resolve->hash_table = hash_table;
+ dns_resolve->name = "dns_resolve";
+ dns_resolve->cache_put = nfs_dns_ent_put;
+ dns_resolve->cache_upcall = nfs_dns_upcall;
+ dns_resolve->cache_parse = nfs_dns_parse;
+ dns_resolve->cache_show = nfs_dns_show;
+ dns_resolve->match = nfs_dns_match;
+ dns_resolve->init = nfs_dns_ent_init;
+ dns_resolve->update = nfs_dns_ent_update;
+ dns_resolve->alloc = nfs_dns_ent_alloc;
+
+ return dns_resolve;
+}
+
+static void nfs_free_dns_resolve(struct cache_detail *dns_resolve)
+{
+ kfree(dns_resolve->hash_table);
+ kfree(dns_resolve);
+}
+
+static struct cache_detail *nfs_get_dns_resolve(struct vfsmount *rpcmount)
+{
+ struct nfs_dns_resolve_list *dns_resolve;
+ int error = 0;
+
+ spin_lock(&nfs_dns_resolve_lock);
+ list_for_each_entry(dns_resolve, &nfs_dns_resolve_list, list) {
+  if (dns_resolve->cd->u.pipefs.mnt->mnt_sb != rpcmount->mnt_sb)
+   continue;
+
+  spin_unlock(&nfs_dns_resolve_lock);
+  return dns_resolve->cd;
+ }
+
+ dns_resolve = kmalloc(sizeof(*dns_resolve), GFP_KERNEL);
+ if (dns_resolve)
```

```
+  dns_resolve->cd = nfs_alloc_dns_resolve();
+ if (!dns_resolve || !dns_resolve->cd) {
+  error = -ENOMEM;
+  goto err;
+ }
+
+ error = nfs_cache_register(dns_resolve->cd, rpcmount);
+ if (error)
+  goto err;
+
+ INIT_LIST_HEAD(&dns_resolve->list);
+ list_add(&dns_resolve->list, &nfs_dns_resolve_list);
+ spin_unlock(&nfs_dns_resolve_lock);
+
+ return dns_resolve->cd;
+err:
+ spin_unlock(&nfs_dns_resolve_lock);
+ if (dns_resolve)
+  kfree(dns_resolve->cd);
+ kfree(dns_resolve);
+ return dns_resolve->cd;
+}
+
+static void nfs_dns_resolver_destroy(void *data)
+{
+ struct nfs_dns_resolve_list *dns_resolve = data;
+
+ spin_lock(&nfs_dns_resolve_lock);
+ nfs_cache_unregister(dns_resolve->cd);
+ nfs_free_dns_resolve(dns_resolve->cd);
+ list_del(&dns_resolve->list);
+ kfree(dns_resolve);
+ spin_unlock(&nfs_dns_resolve_lock);
+}
+
 ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
-  struct sockaddr *sa, size_t salen)
+  struct sockaddr *sa, size_t salen, struct vfsmount *rpcmount)
 {
   struct nfs_dns_ent key = {
    .hostname = name,
    .namelen = namelen,
   };
+ struct cache_detail *dns_resolve;
   struct nfs_dns_ent *item = NULL;
   ssize_t ret;

- ret = do_cache_lookup_wait(&nfs_dns_resolve, &key, &item);
```

```
+ dns_resolve = nfs_get_dns_resolve(rpcmount);
+ ret = do_cache_lookup_wait(dns_resolve, &key, &item);
  if (ret == 0) {
   if (salen >= item->addrlen) {
    memcpy(sa, &item->addr, item->addrlen);
    ret = item->addrlen;
   } else
    ret = -EOVERFLOW;
-  cache_put(&item->h, &nfs_dns_resolve);
+  cache_put(&item->h, dns_resolve);
+  rpc_pipefs_add_destroy_cb(rpcmount->mnt_sb,
+    nfs_dns_resolver_destroy, dns_resolve);
  } else if (ret == -ENOENT)
   ret = -ESRCH;
  return ret;
 }
-
-int nfs_dns_resolver_init(void)
-{
- return nfs_cache_register(&nfs_dns_resolve);
-}
-
-void nfs_dns_resolver_destroy(void)
-{
- nfs_cache_unregister(&nfs_dns_resolve);
-}
-
 #endif
diff --git a/fs/nfs/dns_resolve.h b/fs/nfs/dns_resolve.h
index 199bb55..74ade60 100644
--- a/fs/nfs/dns_resolve.h
+++ b/fs/nfs/dns_resolve.h
@@ -7,20 +7,7 @@
 #define NFS_DNS_HOSTNAME_MAXLEN (128)


-#ifdef CONFIG_NFS_USE_KERNEL_DNS
-static inline int nfs_dns_resolver_init(void)
-{
- return 0;
-}
-
-static inline void nfs_dns_resolver_destroy(void)
-{}
-#else
-extern int nfs_dns_resolver_init(void);
-extern void nfs_dns_resolver_destroy(void);
-#endif
```

```
-
 extern ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
-  struct sockaddr *sa, size_t salen);
+  struct sockaddr *sa, size_t salen, struct vfsmount *rpcmount);

 #endif
diff --git a/fs/nfs/inode.c b/fs/nfs/inode.c
index 1cc600e..1c3be51 100644
--- a/fs/nfs/inode.c
+++ b/fs/nfs/inode.c
@@ -1543,10 +1543,6 @@ static int __init init_nfs_fs(void)

  err = nfs_idmap_init();
  if (err < 0)
-  goto out9;
-
- err = nfs_dns_resolver_init();
- if (err < 0)
   goto out8;

  err = nfs_fscache_register();
@@ -1607,10 +1603,8 @@ out5:
 out6:
  nfs_fscache_unregister();
 out7:
- nfs_dns_resolver_destroy();
-out8:
  nfs_idmap_quit();
-out9:
+out8:
  return err;
 }

@@ -1622,7 +1616,6 @@ static void __exit exit_nfs_fs(void)
  nfs_destroy_inodecache();
  nfs_destroy_nfspagecache();
  nfs_fscache_unregister();
- nfs_dns_resolver_destroy();
  nfs_idmap_quit();
 #ifdef CONFIG_PROC_FS
  rpc_proc_unregister("nfs");
diff --git a/fs/nfs/nfs4namespace.c b/fs/nfs/nfs4namespace.c
index 3c2a172..7a61fdb 100644
--- a/fs/nfs/nfs4namespace.c
+++ b/fs/nfs/nfs4namespace.c
@@ -14,6 +14,7 @@
 #include <linux/slab.h>
 #include <linux/string.h>
```

```
 #include <linux/sunrpc/clnt.h>
+#include <linux/sunrpc/rpc_pipe_fs.h>
 #include <linux/vfs.h>
 #include <linux/inet.h>
 #include "internal.h"
@@ -104,7 +105,8 @@ static size_t nfs_parse_server_name(char *string, size_t len,

  ret = rpc_pton(string, len, sa, salen);
  if (ret == 0) {
-  ret = nfs_dns_resolve_name(string, len, sa, salen);
+  ret = nfs_dns_resolve_name(string, len, sa, salen,
+    init_rpc_pipefs);
   if (ret < 0)
    ret = 0;
  }
--
1.7.4
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs