Subject: [PATCH 0/6] NFS: create DNS resolver cache per network namespace
Posted by Stanislav Kinsbursky on Fri, 25 Nov 2011 13:18:27 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.
tag: v3.1

This patch set depends on previous patch sets titled:
1) "SUNRPC: initial part of making pipefs work in net ns"
2) "SUNPRC: cleanup PipeFS for network-namespace-aware users"

Actually, this patch set consists of two tightly connected parts:
1) DNS resolver cache per network namespcae implementation itself
2) DNS resolver cache dentries creation by PipeFS network namespace aware
routines.

Thus this patch set is the second part of "PipeFS per network namespace"
story.

The following series consists of:

Stanislav Kinsbursky (6):
      SUNRPC: split cache creation and PipeFS registration
      NFS: split cache creation and PipeFS registration
      NFS: handle NFS caches dentries by network namespace aware routines
      NFS: DNS resolver cache per network namespace context introduced
      NFS: DNS resolver PipeFS notifier introduced
      NFS: remove RPC PipeFS mount point references from NFS cache routines

```
 fs/nfs/cache_lib.c           |   61 ++++++++++++++------
 fs/nfs/cache_lib.h           |   10 +++
 fs/nfs/dns_resolve.c         |  125 ++++++++++++++++++++++++++++++++++---------
 fs/nfs/dns_resolve.h         |   14 ++++-
 fs/nfs/inode.c               |   33 ++++++++++-
 fs/nfs/netns.h               |   13 ++++
 fs/nfs/nfs4namespace.c       |    8 ++-
 include/linux/sunrpc/cache.h |    2 +
 net/sunrpc/cache.c           |   12 ++--
 9 files changed, 218 insertions(+), 60 deletions(-)
 create mode 100644 fs/nfs/netns.h
```

--
Signature

Subject: [PATCH 5/6] NFS: DNS resolver PipeFS notifier introduced
Posted by Stanislav Kinsbursky on Fri, 25 Nov 2011 13:19:22 GMT
View Forum Message <> Reply to Message

This patch subscribes DNS resolver caches to RPC pipefs notifications. Notifier
is registering on NFS module load. This notifier callback is responsible for
creation/destruction of PipeFS DNS resolver cache directory.
Note that no locking required in notifier callback because PipeFS superblock
pointer is passed as an argument from it's creation or destruction routine and
thus we can be sure about it's validity.

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

---
 fs/nfs/cache_lib.c | 4 ++--
 fs/nfs/cache_lib.h | 4 ++++
 fs/nfs/dns_resolve.c | 38 ++++++++++++++++++++++++++++++++++++++-
 3 files changed, 43 insertions(+), 3 deletions(-)

diff --git a/fs/nfs/cache_lib.c b/fs/nfs/cache_lib.c
index 5dd017b..5905a31 100644
--- a/fs/nfs/cache_lib.c
+++ b/fs/nfs/cache_lib.c
@@ -112,7 +112,7 @@ int nfs_cache_wait_for_upcall(struct nfs_cache_defer_req *dreq)
  return 0;
 }

-static int nfs_cache_register_sb(struct super_block *sb, struct cache_detail *cd)
+int nfs_cache_register_sb(struct super_block *sb, struct cache_detail *cd)
 {
  int ret;
  struct dentry *dir;
@@ -147,7 +147,7 @@ err:
  return ret;
 }

-static void nfs_cache_unregister_sb(struct super_block *sb, struct cache_detail *cd)
+void nfs_cache_unregister_sb(struct super_block *sb, struct cache_detail *cd)
 {
  if (cd->u.pipefs.dir)
   sunrpc_cache_unregister_pipefs(cd);
diff --git a/fs/nfs/cache_lib.h b/fs/nfs/cache_lib.h
index e0a6cc4..317db95 100644
--- a/fs/nfs/cache_lib.h
+++ b/fs/nfs/cache_lib.h
@@ -27,3 +27,7 @@ extern void nfs_cache_init(struct cache_detail *cd);
 extern void nfs_cache_destroy(struct cache_detail *cd);
 extern int nfs_cache_register_net(struct net *net, struct cache_detail *cd);
 extern void nfs_cache_unregister_net(struct net *net, struct cache_detail *cd);

```
+extern int nfs_cache_register_sb(struct super_block *sb,
+      struct cache_detail *cd);
+extern void nfs_cache_unregister_sb(struct super_block *sb,
+         struct cache_detail *cd);
diff --git a/fs/nfs/dns_resolve.c b/fs/nfs/dns_resolve.c
index 9aea78a..200eb67 100644
--- a/fs/nfs/dns_resolve.c
+++ b/fs/nfs/dns_resolve.c
@@ -40,6 +40,7 @@ ssize_t nfs_dns_resolve_name(struct net *net, char *name, size_t namelen,
 #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"
@@ -400,12 +401,47 @@ void nfs_dns_resolver_cache_destroy(struct net *net)
  kfree(cd);
 }

+static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
+       void *ptr)
+{
+ struct super_block *sb = ptr;
+ struct net *net = sb->s_fs_info;
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+ struct cache_detail *cd = nn->nfs_dns_resolve;
+ int ret = 0;
+
+ if (cd == NULL)
+  return 0;
+
+ if (!try_module_get(THIS_MODULE))
+  return 0;
+
+ switch (event) {
+ case RPC_PIPEFS_MOUNT:
+  ret = nfs_cache_register_sb(sb, cd);
+  break;
+ case RPC_PIPEFS_UMOUNT:
+  nfs_cache_unregister_sb(sb, cd);
+  break;
+ default:
+  ret = -ENOTSUPP;
+  break;
+ }
+ module_put(THIS_MODULE);
+ return ret;
```

```
+}
+
+static struct notifier_block nfs_dns_resolver_block = {
+ .notifier_call = rpc_pipefs_event,
+};
+
 int nfs_dns_resolver_init(void)
 {
- return 0;
+ return rpc_pipefs_notifier_register(&nfs_dns_resolver_block);
 }

 void nfs_dns_resolver_destroy(void)
 {
+  rpc_pipefs_notifier_unregister(&nfs_dns_resolver_block);
 }
 #endif
```

Subject: [PATCH 4/6] NFS: DNS resolver cache per network namespace context introduced

Posted by Stanislav Kinsbursky on Fri, 25 Nov 2011 13:19:24 GMT

View Forum Message <> Reply to Message

This patch implements DNS resolver cache creation and registration for each alive network namespace context.
This was done by registering NFS per-net operations, responsible for DNS cache allocation/register and unregister/destructioning instead of initialization and destruction of static "nfs_dns_resolve" cache detail (this one was removed).
Pointer to network dns resolver cache is stored in new per-net "nfs_net" structure.
This patch also changes nfs_dns_resolve_name() function prototype (and it's calls) by adding network pointer parameter, which is used to get proper DNS resolver cache pointer for do_cache_lookup_wait() call.

Note: empty nfs_dns_resolver_init() and nfs_dns_resolver_destroy() functions will be used in next patch in the series.

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

```
 fs/nfs/dns_resolve.c   |   96 ++++++++++++++++++++++++++++++++++++++----------
 fs/nfs/dns_resolve.h   |   14 ++++++-
 fs/nfs/inode.c         |   33 ++++++++++++++--
 fs/nfs/netns.h         |   13 +++++++
 fs/nfs/nfs4namespace.c |    8 +++-
 5 files changed, 123 insertions(+), 41 deletions(-)
 create mode 100644 fs/nfs/netns.h
```

```
diff --git a/fs/nfs/dns_resolve.c b/fs/nfs/dns_resolve.c
index 3cbf4b8..9aea78a 100644
--- a/fs/nfs/dns_resolve.c
+++ b/fs/nfs/dns_resolve.c
@@ -11,7 +11,7 @@
 #include <linux/sunrpc/clnt.h>
 #include <linux/dns_resolver.h>

-ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
+ssize_t nfs_dns_resolve_name(struct net *net, char *name, size_t namelen,
   struct sockaddr *sa, size_t salen)
 {
  ssize_t ret;
@@ -43,12 +43,11 @@ ssize_t nfs_dns_resolve_name(char *name, size_t namelen,

 #include "dns_resolve.h"
 #include "cache_lib.h"
+#include "netns.h"

 #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];
-
 struct nfs_dns_ent {
  struct cache_head h;

@@ -259,21 +258,6 @@ 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,
@@ -336,8 +320,8 @@ out:
  return ret;
 }

-ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
-  struct sockaddr *sa, size_t salen)
+ssize_t nfs_dns_resolve_name(struct net *net, char *name,
+  size_t namelen, struct sockaddr *sa, size_t salen)
 {
  struct nfs_dns_ent key = {
   .hostname = name,
@@ -345,37 +329,83 @@ ssize_t nfs_dns_resolve_name(char *name, size_t namelen,
  };
  struct nfs_dns_ent *item = NULL;
  ssize_t ret;
+ struct nfs_net *nn = net_generic(net, nfs_net_id);

- ret = do_cache_lookup_wait(&nfs_dns_resolve, &key, &item);
+ ret = do_cache_lookup_wait(nn->nfs_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, nn->nfs_dns_resolve);
  } else if (ret == -ENOENT)
   ret = -ESRCH;
  return ret;
 }

-int nfs_dns_resolver_init(void)
+int nfs_dns_resolver_cache_init(struct net *net)
 {
- int err;
+ int err = -ENOMEM;
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+ struct cache_detail *cd;
+ struct cache_head **tbl;
+
+ cd = kzalloc(sizeof(struct cache_detail), GFP_KERNEL);
+ if (cd == NULL)
+  goto err_cd;
+
+ tbl = kzalloc(NFS_DNS_HASHTBL_SIZE * sizeof(struct cache_head *),
```

```
+   GFP_KERNEL);
+ if (tbl == NULL)
+  goto err_tbl;
+
+ cd->owner = THIS_MODULE,
+ cd->hash_size = NFS_DNS_HASHTBL_SIZE,
+ cd->hash_table = tbl,
+ cd->name = "dns_resolve",
+ cd->cache_put = nfs_dns_ent_put,
+ cd->cache_upcall = nfs_dns_upcall,
+ cd->cache_parse = nfs_dns_parse,
+ cd->cache_show = nfs_dns_show,
+ cd->match = nfs_dns_match,
+ cd->init = nfs_dns_ent_init,
+ cd->update = nfs_dns_ent_update,
+ cd->alloc = nfs_dns_ent_alloc,
+
+ nfs_cache_init(cd);
+ err = nfs_cache_register_net(net, cd);
+ if (err)
+  goto err_reg;
+ nn->nfs_dns_resolve = cd;
+ return 0;

- nfs_cache_init(&nfs_dns_resolve);
- err = nfs_cache_register_net(&init_net, &nfs_dns_resolve);
- if (err) {
-  nfs_cache_destroy(&nfs_dns_resolve);
-  return err;
- }
+err_reg:
+ nfs_cache_destroy(cd);
+ kfree(cd->hash_table);
+err_tbl:
+ kfree(cd);
+err_cd:
+ return err;
+}
+
+void nfs_dns_resolver_cache_destroy(struct net *net)
+{
+ struct nfs_net *nn = net_generic(net, nfs_net_id);
+ struct cache_detail *cd = nn->nfs_dns_resolve;
+
+ nfs_cache_unregister_net(net, cd);
+ nfs_cache_destroy(cd);
+ kfree(cd->hash_table);
+ kfree(cd);
```

```
+}
+
+int nfs_dns_resolver_init(void)
+{
  return 0;
 }

 void nfs_dns_resolver_destroy(void)
 {
- nfs_cache_unregister_net(&init_net, &nfs_dns_resolve);
- nfs_cache_destroy(&nfs_dns_resolve);
 }
-
 #endif
diff --git a/fs/nfs/dns_resolve.h b/fs/nfs/dns_resolve.h
index 199bb55..2e4f596 100644
--- a/fs/nfs/dns_resolve.h
+++ b/fs/nfs/dns_resolve.h
@@ -15,12 +15,22 @@ static inline int nfs_dns_resolver_init(void)

 static inline void nfs_dns_resolver_destroy(void)
 {}
+
+static inline int nfs_dns_resolver_cache_init(struct net *net)
+{
+ return 0;
+}
+
+static inline void nfs_dns_resolver_cache_destroy(struct net *net)
+{}
 #else
 extern int nfs_dns_resolver_init(void);
 extern void nfs_dns_resolver_destroy(void);
+extern int nfs_dns_resolver_cache_init(struct net *net);
+extern void nfs_dns_resolver_cache_destroy(struct net *net);
 #endif

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

 #endif
diff --git a/fs/nfs/inode.c b/fs/nfs/inode.c
index fe12037..84d8506 100644
--- a/fs/nfs/inode.c
+++ b/fs/nfs/inode.c
@@ -50,6 +50,7 @@
```

```
 #include "fscache.h"
 #include "dns_resolve.h"
 #include "pnfs.h"
+#include "netns.h"

 #define NFSDBG_FACILITY  NFSDBG_VFS

@@ -1550,6 +1551,25 @@ static void nfsiod_stop(void)
  destroy_workqueue(wq);
 }

+int nfs_net_id;
+
+static int nfs_net_init(struct net *net)
+{
+ return nfs_dns_resolver_cache_init(net);
+}
+
+static void nfs_net_exit(struct net *net)
+{
+ nfs_dns_resolver_cache_destroy(net);
+}
+
+static struct pernet_operations nfs_net_ops = {
+ .init = nfs_net_init,
+ .exit = nfs_net_exit,
+ .id   = &nfs_net_id,
+ .size = sizeof(struct nfs_net),
+};
+
 /*
  * Initialize NFS
  */
@@ -1559,10 +1579,14 @@ static int __init init_nfs_fs(void)

  err = nfs_idmap_init();
  if (err < 0)
-  goto out9;
+  goto out10;

  err = nfs_dns_resolver_init();
  if (err < 0)
+  goto out9;
+
+ err = register_pernet_subsys(&nfs_net_ops);
+ if (err < 0)
   goto out8;
```

```
  err = nfs_fscache_register();
@@ -1623,10 +1647,12 @@ out5:
 out6:
  nfs_fscache_unregister();
 out7:
- nfs_dns_resolver_destroy();
+ unregister_pernet_subsys(&nfs_net_ops);
 out8:
- nfs_idmap_quit();
+ nfs_dns_resolver_destroy();
 out9:
+ nfs_idmap_quit();
+out10:
  return err;
 }

@@ -1638,6 +1664,7 @@ static void __exit exit_nfs_fs(void)
  nfs_destroy_inodecache();
  nfs_destroy_nfspagecache();
  nfs_fscache_unregister();
+ unregister_pernet_subsys(&nfs_net_ops);
  nfs_dns_resolver_destroy();
  nfs_idmap_quit();
 #ifdef CONFIG_PROC_FS
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
new file mode 100644
index 0000000..8c1f130
--- /dev/null
+++ b/fs/nfs/netns.h
@@ -0,0 +1,13 @@
+#ifndef __NFS_NETNS_H__
+#define __NFS_NETNS_H__
+
+#include <net/net_namespace.h>
+#include <net/netns/generic.h>
+
+struct nfs_net {
+ struct cache_detail *nfs_dns_resolve;
+};
+
+extern int nfs_net_id;
+
+#endif
diff --git a/fs/nfs/nfs4namespace.c b/fs/nfs/nfs4namespace.c
index bb80c49..919a369 100644
--- a/fs/nfs/nfs4namespace.c
+++ b/fs/nfs/nfs4namespace.c
@@ -94,13 +94,14 @@ static int nfs4_validate_fspath(struct dentry *dentry,
```

```
 }

 static size_t nfs_parse_server_name(char *string, size_t len,
-  struct sockaddr *sa, size_t salen)
+  struct sockaddr *sa, size_t salen, struct nfs_server *server)
 {
  ssize_t ret;

  ret = rpc_pton(string, len, sa, salen);
  if (ret == 0) {
-  ret = nfs_dns_resolve_name(string, len, sa, salen);
+  ret = nfs_dns_resolve_name(server->client->cl_xprt->xprt_net,
+         string, len, sa, salen);
   if (ret < 0)
    ret = 0;
  }
@@ -137,7 +138,8 @@ static struct vfsmount *try_location(struct nfs_clone_mount *mountdata,
    continue;

   mountdata->addrlen = nfs_parse_server_name(buf->data, buf->len,
-    mountdata->addr, addr_bufsize);
+    mountdata->addr, addr_bufsize,
+    NFS_SB(mountdata->sb));
   if (mountdata->addrlen == 0)
    continue;
```

Subject: [PATCH 6/6] NFS: remove RPC PipeFS mount point references from NFS cache routines
Posted by Stanislav Kinsbursky on Fri, 25 Nov 2011 13:19:28 GMT
View Forum Message <> Reply to Message

This is a cleanup patch. We don't need this reference anymore, because DNS resolver cache now creates it's dentries in per-net operations and on PipeFS mount/umount notification.
Note that nfs_cache_register_net() now returns 0 instead of -ENOENT in case of PiepFS superblock absence. This is ok, Dns resolver cache will be regestered on PipeFS mount event.

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

```
---
 fs/nfs/cache_lib.c |   19 ++++----------------
 1 files changed, 4 insertions(+), 15 deletions(-)

diff --git a/fs/nfs/cache_lib.c b/fs/nfs/cache_lib.c
index 5905a31..dded263 100644
--- a/fs/nfs/cache_lib.c
```

```
+++ b/fs/nfs/cache_lib.c
@@ -126,24 +126,14 @@ int nfs_cache_register_sb(struct super_block *sb, struct cache_detail
*cd)

 int nfs_cache_register_net(struct net *net, struct cache_detail *cd)
 {
- struct vfsmount *mnt;
  struct super_block *pipefs_sb;
- int ret;
+ int ret = 0;

- mnt = rpc_get_mount();
- if (IS_ERR(mnt))
-  return PTR_ERR(mnt);
  pipefs_sb = rpc_get_sb_net(net);
- if (!pipefs_sb) {
-  ret = -ENOENT;
-  goto err;
+ if (pipefs_sb) {
+  ret = nfs_cache_register_sb(pipefs_sb, cd);
+  rpc_put_sb_net(net);
  }
- ret = nfs_cache_register_sb(pipefs_sb, cd);
- rpc_put_sb_net(net);
- if (!ret)
-  return ret;
-err:
- rpc_put_mount();
  return ret;
 }

@@ -162,7 +152,6 @@ void nfs_cache_unregister_net(struct net *net, struct cache_detail *cd)
   nfs_cache_unregister_sb(pipefs_sb, cd);
   rpc_put_sb_net(net);
  }
- rpc_put_mount();
 }

 void nfs_cache_init(struct cache_detail *cd)
```