Subject: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:48:56 GMT
View Forum Message <> Reply to Message

One more step towards to "NFS in container".
With this patch set caches are allocated per network namespace.

The following series consists of:

Stanislav Kinsbursky (5):
SUNRPC: cache creation and destruction routines introduced
SUNRPC: create unix gid cache per network namespace
SUNRPC: create GSS auth cache per network namespace
SUNRPC: ip map cache per network namespace cleanup
SUNRPC: generic cache register routines removed

```
include/linux/sunrpc/cache.h       |    6 +
include/linux/sunrpc/svcauth_gss.h |    2
net/sunrpc/auth_gss/auth_gss.c     |   21 +++++
net/sunrpc/auth_gss/svcauth_gss.c  |  161 ++++++++++++++++++++++++++++-----------
net/sunrpc/cache.c                 |   34 ++++++--
net/sunrpc/netns.h                 |    3 +
net/sunrpc/sunrpc_syms.c           |   15 ++-
net/sunrpc/svcauth_unix.c          |  107 +++++++++++++++++----------
8 files changed, 240 insertions(+), 109 deletions(-)
```

Subject: [PATCH 1/5] SUNRPC: cache creation and destruction routines introduced
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:49:06 GMT
View Forum Message <> Reply to Message

This patch prepares infrastructure for network namespace aware cache detail allocation.
One note about adding network namespace link to cache structure. It's going to be used later in NFS DNS cache parsing routine (nfs_dns_parse for rpc_pton() call).

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

```
include/linux/sunrpc/cache.h |    4 ++++
net/sunrpc/cache.c           |   26 ++++++++++++++++++++++++
2 files changed, 30 insertions(+), 0 deletions(-)
```

diff --git a/include/linux/sunrpc/cache.h b/include/linux/sunrpc/cache.h

```
index 590a8ab..259381c 100644
--- a/include/linux/sunrpc/cache.h
+++ b/include/linux/sunrpc/cache.h
@@ -117,6 +117,7 @@ struct cache_detail {
   struct cache_detail_procfs procfs;
   struct cache_detail_pipefs pipefs;
  } u;
+ struct net  *net;
 };


@@ -202,6 +203,9 @@ extern int cache_register_net(struct cache_detail *cd, struct net *net);
 extern void cache_unregister(struct cache_detail *cd);
 extern void cache_unregister_net(struct cache_detail *cd, struct net *net);

+extern struct cache_detail *cache_create_net(struct cache_detail *tmpl, struct net *net);
+extern void cache_destroy_net(struct cache_detail *cd, struct net *net);
+
 extern void sunrpc_init_cache_detail(struct cache_detail *cd);
 extern void sunrpc_destroy_cache_detail(struct cache_detail *cd);
 extern int sunrpc_cache_register_pipefs(struct dentry *parent, const char *,
diff --git a/net/sunrpc/cache.c b/net/sunrpc/cache.c
index f128154..9ef5926 100644
--- a/net/sunrpc/cache.c
+++ b/net/sunrpc/cache.c
@@ -1662,6 +1662,32 @@ void cache_unregister(struct cache_detail *cd)
 }
 EXPORT_SYMBOL_GPL(cache_unregister);

+struct cache_detail *cache_create_net(struct cache_detail *tmpl, struct net *net)
+{
+ struct cache_detail *cd;
+
+ cd = kmemdup(tmpl, sizeof(struct cache_detail), GFP_KERNEL);
+ if (cd == NULL)
+  return ERR_PTR(-ENOMEM);
+
+ cd->hash_table = kzalloc(cd->hash_size * sizeof(struct cache_head *),
+     GFP_KERNEL);
+ if (cd->hash_table == NULL) {
+  kfree(cd);
+  return ERR_PTR(-ENOMEM);
+ }
+ cd->net = net;
+ return cd;
+}
+EXPORT_SYMBOL_GPL(cache_create_net);
+
```

```
+void cache_destroy_net(struct cache_detail *cd, struct net *net)
+{
+ kfree(cd->hash_table);
+ kfree(cd);
+}
+EXPORT_SYMBOL_GPL(cache_destroy_net);
+
 static ssize_t cache_read_pipefs(struct file *filp, char __user *buf,
     size_t count, loff_t *ppos)
 {
```

Subject: [PATCH 2/5] SUNRPC: create unix gid cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:49:14 GMT
View Forum Message <> Reply to Message

This patch makes unix_gid_cache cache detail allocated and registered per
network namespace context.
Thus with this patch unix_gid_cache contents for network namespace "X" are
controlled from proc file system mount for the same network namespace "X".

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

```
---
 net/sunrpc/netns.h        |    1 +
 net/sunrpc/sunrpc_syms.c  |   14 +++++++++++----
 net/sunrpc/svcauth_unix.c |   36 +++++++++++++++++++++++++++++++++++-----
 3 files changed, 42 insertions(+), 9 deletions(-)

diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index 1fdeb1b..309f88d 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -9,6 +9,7 @@ struct cache_detail;
 struct sunrpc_net {
  struct proc_dir_entry *proc_net_rpc;
  struct cache_detail *ip_map_cache;
+ struct cache_detail *unix_gid_cache;

  struct super_block *pipefs_sb;
  struct mutex pipefs_sb_lock;
diff --git a/net/sunrpc/sunrpc_syms.c b/net/sunrpc/sunrpc_syms.c
index b4217dc..38a72a1 100644
--- a/net/sunrpc/sunrpc_syms.c
+++ b/net/sunrpc/sunrpc_syms.c
@@ -26,6 +26,9 @@

 int sunrpc_net_id;
```

```
+extern int unix_gid_cache_create(struct net *net);
+extern int unix_gid_cache_destroy(struct net *net);
+
 static __net_init int sunrpc_init_net(struct net *net)
 {
  int err;
@@ -39,11 +42,17 @@ static __net_init int sunrpc_init_net(struct net *net)
  if (err)
   goto err_ipmap;

+ err = unix_gid_cache_create(net);
+ if (err)
+  goto err_unixgid;
+
  rpc_pipefs_init_net(net);
  INIT_LIST_HEAD(&sn->all_clients);
  spin_lock_init(&sn->rpc_client_lock);
  return 0;

+err_unixgid:
+ ip_map_cache_destroy(net);
 err_ipmap:
  rpc_proc_exit(net);
 err_proc:
@@ -52,6 +61,7 @@ err_proc:

 static __net_exit void sunrpc_exit_net(struct net *net)
 {
+ unix_gid_cache_destroy(net);
  ip_map_cache_destroy(net);
  rpc_proc_exit(net);
 }
@@ -63,8 +73,6 @@ static struct pernet_operations sunrpc_net_ops = {
  .size = sizeof(struct sunrpc_net),
 };

-extern struct cache_detail unix_gid_cache;
-
 static int __init
 init_sunrpc(void)
 {
@@ -86,7 +94,6 @@ init_sunrpc(void)
 #ifdef RPC_DEBUG
  rpc_register_sysctl();
 #endif
- cache_register(&unix_gid_cache);
  svc_init_xprt_sock(); /* svc sock transport */
```

```
  init_socket_xprt(); /* clnt sock transport */
  return 0;
@@ -109,7 +116,6 @@ cleanup_sunrpc(void)
  svc_cleanup_xprt_sock();
  unregister_rpc_pipefs();
  rpc_destroy_mempool();
- cache_unregister(&unix_gid_cache);
  unregister_pernet_subsys(&sunrpc_net_ops);
 #ifdef RPC_DEBUG
  rpc_unregister_sysctl();
diff --git a/net/sunrpc/svcauth_unix.c b/net/sunrpc/svcauth_unix.c
index 2f8c426..c753ae4 100644
--- a/net/sunrpc/svcauth_unix.c
+++ b/net/sunrpc/svcauth_unix.c
@@ -436,7 +436,6 @@ struct unix_gid {
  uid_t   uid;
  struct group_info *gi;
 };
-static struct cache_head *gid_table[GID_HASHMAX];

 static void unix_gid_put(struct kref *kref)
 {
@@ -495,7 +494,6 @@ static int unix_gid_upcall(struct cache_detail *cd, struct cache_head *h)
 }

 static struct unix_gid *unix_gid_lookup(uid_t uid);
-extern struct cache_detail unix_gid_cache;

 static int unix_gid_parse(struct cache_detail *cd,
    char *mesg, int mlen)
@@ -544,14 +542,14 @@ static int unix_gid_parse(struct cache_detail *cd,
   struct cache_head *ch;
   ug.h.flags = 0;
   ug.h.expiry_time = expiry;
-  ch = sunrpc_cache_update(&unix_gid_cache,
+  ch = sunrpc_cache_update(cd,
       &ug.h, &ugp->h,
       hash_long(uid, GID_HASHBITS));
   if (!ch)
    err = -ENOMEM;
   else {
    err = 0;
-   cache_put(ch, &unix_gid_cache);
+   cache_put(ch, cd);
   }
  } else
   err = -ENOMEM;
@@ -590,7 +588,6 @@ static int unix_gid_show(struct seq_file *m,
```

```
 struct cache_detail unix_gid_cache = {
  .owner  = THIS_MODULE,
  .hash_size = GID_HASHMAX,
- .hash_table = gid_table,
  .name  = "auth.unix.gid",
  .cache_put = unix_gid_put,
  .cache_upcall = unix_gid_upcall,
@@ -602,6 +599,35 @@ struct cache_detail unix_gid_cache = {
  .alloc  = unix_gid_alloc,
 };

+int unix_gid_cache_create(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;
+
+ cd = cache_create_net(&unix_gid_cache, net);
+ if (IS_ERR(cd))
+  return PTR_ERR(cd);
+ err = cache_register_net(cd, net);
+ if (err) {
+  cache_destroy_net(&unix_gid_cache, net);
+  return err;
+ }
+ sn->unix_gid_cache = cd;
+ return 0;
+}
+
+void unix_gid_cache_destroy(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd = sn->unix_gid_cache;
+
+ sn->unix_gid_cache = NULL;
+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
+}
+
 static struct unix_gid *unix_gid_lookup(uid_t uid)
 {
  struct unix_gid ug;
```

Subject: [PATCH 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:49:23 GMT

This patch makes GSS auth cache details allocated and registered per network namespace context.
Thus with this patch rsi_cache and rsc_cache contents for network namespace "X" are controlled from proc file system mount for the same network namespace "X".

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

```
---
 include/linux/sunrpc/svcauth_gss.h |    2
 net/sunrpc/auth_gss/auth_gss.c     |   21 +++++
 net/sunrpc/auth_gss/svcauth_gss.c  |  161 ++++++++++++++++++++++++++----------
 net/sunrpc/cache.c                 |    2
 net/sunrpc/netns.h                 |    2
 net/sunrpc/sunrpc_syms.c           |    1
 6 files changed, 141 insertions(+), 48 deletions(-)

diff --git a/include/linux/sunrpc/svcauth_gss.h b/include/linux/sunrpc/svcauth_gss.h
index 83bbee3..7c32daa 100644
--- a/include/linux/sunrpc/svcauth_gss.h
+++ b/include/linux/sunrpc/svcauth_gss.h
@@ -18,6 +18,8 @@

 int gss_svc_init(void);
 void gss_svc_shutdown(void);
+int gss_svc_init_net(struct net *net);
+void gss_svc_shutdown_net(struct net *net);
 int svcauth_gss_register_pseudoflavor(u32 pseudoflavor, char * name);
 u32 svcauth_gss_flavor(struct auth_domain *dom);
 char *svc_gss_principal(struct svc_rqst *);
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 9ef7fa3..5e02207 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -1662,6 +1662,21 @@ static const struct rpc_pipe_ops gss_upcall_ops_v1 = {
  .release_pipe = gss_pipe_release,
 };

+static __net_init int rpcsec_gss_init_net(struct net *net)
+{
+ return gss_svc_init_net(net);
+}
+
+static __net_exit void rpcsec_gss_exit_net(struct net *net)
+{
+ gss_svc_shutdown_net(net);
+}
+
```

```
+static struct pernet_operations rpcsec_gss_net_ops = {
+ .init = rpcsec_gss_init_net,
+ .exit = rpcsec_gss_exit_net,
+};
+
 /*
  * Initialize RPCSEC_GSS module
  */
@@ -1675,8 +1690,13 @@ static int __init init_rpcsec_gss(void)
  err = gss_svc_init();
  if (err)
   goto out_unregister;
+ err = register_pernet_subsys(&rpcsec_gss_net_ops);
+ if (err)
+  goto out_svc_exit;
  rpc_init_wait_queue(&pipe_version_rpc_waitqueue, "gss pipe version");
  return 0;
+out_svc_exit:
+ gss_svc_shutdown();
 out_unregister:
  rpcauth_unregister(&authgss_ops);
 out:
@@ -1685,6 +1705,7 @@ out:

 static void __exit exit_rpcsec_gss(void)
 {
+ unregister_pernet_subsys(&rpcsec_gss_net_ops);
  gss_svc_shutdown();
  rpcauth_unregister(&authgss_ops);
  rcu_barrier(); /* Wait for completion of call_rcu()'s */
diff --git a/net/sunrpc/auth_gss/svcauth_gss.c b/net/sunrpc/auth_gss/svcauth_gss.c
index 8d0f7d3..e544cd8 100644
--- a/net/sunrpc/auth_gss/svcauth_gss.c
+++ b/net/sunrpc/auth_gss/svcauth_gss.c
@@ -48,6 +48,8 @@
 #include <linux/sunrpc/svcauth_gss.h>
 #include <linux/sunrpc/cache.h>

+#include "../netns.h"
+
 #ifdef RPC_DEBUG
 # define RPCDBG_FACILITY RPCDBG_AUTH
 #endif
@@ -75,10 +77,8 @@ struct rsi {
  int   major_status, minor_status;
 };

-static struct cache_head *rsi_table[RSI_HASHMAX];
```

```
-static struct cache_detail rsi_cache;
-static struct rsi *rsi_update(struct rsi *new, struct rsi *old);
-static struct rsi *rsi_lookup(struct rsi *item);
+static struct rsi *rsi_update(struct cache_detail *cd, struct rsi *new, struct rsi *old);
+static struct rsi *rsi_lookup(struct cache_detail *cd, struct rsi *item);

 static void rsi_free(struct rsi *rsii)
 {
@@ -216,7 +216,7 @@ static int rsi_parse(struct cache_detail *cd,
  if (dup_to_netobj(&rsii.in_token, buf, len))
   goto out;

- rsip = rsi_lookup(&rsii);
+ rsip = rsi_lookup(cd, &rsii);
  if (!rsip)
   goto out;

@@ -258,12 +258,12 @@ static int rsi_parse(struct cache_detail *cd,
  if (dup_to_netobj(&rsii.out_token, buf, len))
   goto out;
  rsii.h.expiry_time = expiry;
- rsip = rsi_update(&rsii, rsip);
+ rsip = rsi_update(cd, &rsii, rsip);
  status = 0;
 out:
  rsi_free(&rsii);
  if (rsip)
-  cache_put(&rsip->h, &rsi_cache);
+  cache_put(&rsip->h, cd);
  else
   status = -ENOMEM;
  return status;
@@ -272,7 +272,6 @@ out:
 static struct cache_detail rsi_cache = {
  .owner  = THIS_MODULE,
  .hash_size = RSI_HASHMAX,
- .hash_table     = rsi_table,
  .name           = "auth.rpcsec.init",
  .cache_put      = rsi_put,
  .cache_upcall   = rsi_upcall,
@@ -283,24 +282,24 @@ static struct cache_detail rsi_cache = {
  .alloc  = rsi_alloc,
 };

-static struct rsi *rsi_lookup(struct rsi *item)
+static struct rsi *rsi_lookup(struct cache_detail *cd, struct rsi *item)
 {
  struct cache_head *ch;
```

```
  int hash = rsi_hash(item);

- ch = sunrpc_cache_lookup(&rsi_cache, &item->h, hash);
+ ch = sunrpc_cache_lookup(cd, &item->h, hash);
  if (ch)
   return container_of(ch, struct rsi, h);
  else
   return NULL;
 }

-static struct rsi *rsi_update(struct rsi *new, struct rsi *old)
+static struct rsi *rsi_update(struct cache_detail *cd, struct rsi *new, struct rsi *old)
 {
  struct cache_head *ch;
  int hash = rsi_hash(new);

- ch = sunrpc_cache_update(&rsi_cache, &new->h,
+ ch = sunrpc_cache_update(cd, &new->h,
      &old->h, hash);
  if (ch)
   return container_of(ch, struct rsi, h);
@@ -339,10 +338,8 @@ struct rsc {
  char   *client_name;
 };

-static struct cache_head *rsc_table[RSC_HASHMAX];
-static struct cache_detail rsc_cache;
-static struct rsc *rsc_update(struct rsc *new, struct rsc *old);
-static struct rsc *rsc_lookup(struct rsc *item);
+static struct rsc *rsc_update(struct cache_detail *cd, struct rsc *new, struct rsc *old);
+static struct rsc *rsc_lookup(struct cache_detail *cd, struct rsc *item);

 static void rsc_free(struct rsc *rsci)
 {
@@ -444,7 +441,7 @@ static int rsc_parse(struct cache_detail *cd,
  if (expiry == 0)
   goto out;

- rscp = rsc_lookup(&rsci);
+ rscp = rsc_lookup(cd, &rsci);
  if (!rscp)
   goto out;

@@ -506,13 +503,13 @@ static int rsc_parse(struct cache_detail *cd,

  }
  rsci.h.expiry_time = expiry;
- rscp = rsc_update(&rsci, rscp);
```

```
+ rscp = rsc_update(cd, &rsci, rscp);
  status = 0;
 out:
  gss_mech_put(gm);
  rsc_free(&rsci);
  if (rscp)
-  cache_put(&rscp->h, &rsc_cache);
+  cache_put(&rscp->h, cd);
  else
   status = -ENOMEM;
  return status;
@@ -521,7 +518,6 @@ out:
 static struct cache_detail rsc_cache = {
  .owner  = THIS_MODULE,
  .hash_size = RSC_HASHMAX,
- .hash_table = rsc_table,
  .name  = "auth.rpcsec.context",
  .cache_put = rsc_put,
  .cache_parse = rsc_parse,
@@ -531,24 +527,24 @@ static struct cache_detail rsc_cache = {
  .alloc  = rsc_alloc,
 };

-static struct rsc *rsc_lookup(struct rsc *item)
+static struct rsc *rsc_lookup(struct cache_detail *cd, struct rsc *item)
 {
  struct cache_head *ch;
  int hash = rsc_hash(item);

- ch = sunrpc_cache_lookup(&rsc_cache, &item->h, hash);
+ ch = sunrpc_cache_lookup(cd, &item->h, hash);
  if (ch)
   return container_of(ch, struct rsc, h);
  else
   return NULL;
 }

-static struct rsc *rsc_update(struct rsc *new, struct rsc *old)
+static struct rsc *rsc_update(struct cache_detail *cd, struct rsc *new, struct rsc *old)
 {
  struct cache_head *ch;
  int hash = rsc_hash(new);

- ch = sunrpc_cache_update(&rsc_cache, &new->h,
+ ch = sunrpc_cache_update(cd, &new->h,
      &old->h, hash);
  if (ch)
   return container_of(ch, struct rsc, h);
```

```
@@ -558,7 +554,7 @@ static struct rsc *rsc_update(struct rsc *new, struct rsc *old)


 static struct rsc *
-gss_svc_searchbyctx(struct xdr_netobj *handle)
+gss_svc_searchbyctx(struct cache_detail *cd, struct xdr_netobj *handle)
 {
  struct rsc rsci;
  struct rsc *found;
@@ -566,11 +562,11 @@ gss_svc_searchbyctx(struct xdr_netobj *handle)
  memset(&rsci, 0, sizeof(rsci));
  if (dup_to_netobj(&rsci.handle, handle->data, handle->len))
   return NULL;
- found = rsc_lookup(&rsci);
+ found = rsc_lookup(cd, &rsci);
  rsc_free(&rsci);
  if (!found)
   return NULL;
- if (cache_check(&rsc_cache, &found->h, NULL))
+ if (cache_check(cd, &found->h, NULL))
   return NULL;
  return found;
 }
@@ -968,20 +964,20 @@ svcauth_gss_set_client(struct svc_rqst *rqstp)
 }

 static inline int
-gss_write_init_verf(struct svc_rqst *rqstp, struct rsi *rsip)
+gss_write_init_verf(struct cache_detail *cd, struct svc_rqst *rqstp, struct rsi *rsip)
 {
  struct rsc *rsci;
  int        rc;

  if (rsip->major_status != GSS_S_COMPLETE)
   return gss_write_null_verf(rqstp);
- rsci = gss_svc_searchbyctx(&rsip->out_handle);
+ rsci = gss_svc_searchbyctx(cd, &rsip->out_handle);
  if (rsci == NULL) {
   rsip->major_status = GSS_S_NO_CONTEXT;
   return gss_write_null_verf(rqstp);
  }
  rc = gss_write_verf(rqstp, rsci->mechctx, GSS_SEQ_WIN);
- cache_put(&rsci->h, &rsc_cache);
+ cache_put(&rsci->h, cd);
  return rc;
 }

@@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
```

```
  struct xdr_netobj tmpobj;
  struct rsi *rsip, rsikey;
  int ret;
+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

  /* Read the verifier; should be NULL: */
  *authp = rpc_autherr_badverf;
@@ -1028,17 +1025,17 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
  }

  /* Perform upcall, or find upcall result: */
- rsip = rsi_lookup(&rsikey);
+ rsip = rsi_lookup(sn->rsi_cache, &rsikey);
  rsi_free(&rsikey);
  if (!rsip)
   return SVC_CLOSE;
- if (cache_check(&rsi_cache, &rsip->h, &rqstp->rq_chandle) < 0)
+ if (cache_check(sn->rsi_cache, &rsip->h, &rqstp->rq_chandle) < 0)
   /* No upcall result: */
   return SVC_CLOSE;

  ret = SVC_CLOSE;
  /* Got an answer to the upcall; use it: */
- if (gss_write_init_verf(rqstp, rsip))
+ if (gss_write_init_verf(sn->rsc_cache, rqstp, rsip))
   goto out;
  if (resv->iov_len + 4 > PAGE_SIZE)
   goto out;
@@ -1055,7 +1052,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,

  ret = SVC_COMPLETE;
 out:
- cache_put(&rsip->h, &rsi_cache);
+ cache_put(&rsip->h, sn->rsi_cache);
  return ret;
 }

@@ -1079,6 +1076,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
  __be32  *rpcstart;
  __be32  *reject_stat = resv->iov_base + resv->iov_len;
  int  ret;
+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

  dprintk("RPC:       svcauth_gss: argv->iov_len = %zd\n",
    argv->iov_len);
@@ -1129,7 +1127,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
  case RPC_GSS_PROC_DESTROY:
   /* Look up the context, and check the verifier: */
```

```
   *authp = rpcsec_gsserr_credproblem;
-  rsci = gss_svc_searchbyctx(&gc->gc_ctx);
+  rsci = gss_svc_searchbyctx(sn->rsc_cache, &gc->gc_ctx);
   if (!rsci)
    goto auth_err;
   switch (gss_verify_header(rqstp, rsci, rpcstart, gc, authp)) {
@@ -1209,7 +1207,7 @@ drop:
  ret = SVC_DROP;
 out:
  if (rsci)
-  cache_put(&rsci->h, &rsc_cache);
+  cache_put(&rsci->h, sn->rsc_cache);
  return ret;
 }

@@ -1362,6 +1360,7 @@ svcauth_gss_release(struct svc_rqst *rqstp)
  struct rpc_gss_wire_cred *gc = &gsd->clcred;
  struct xdr_buf *resbuf = &rqstp->rq_res;
  int stat = -EINVAL;
+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

  if (gc->gc_proc != RPC_GSS_PROC_DATA)
   goto out;
@@ -1404,7 +1403,7 @@ out_err:
   put_group_info(rqstp->rq_cred.cr_group_info);
  rqstp->rq_cred.cr_group_info = NULL;
  if (gsd->rsci)
-  cache_put(&gsd->rsci->h, &rsc_cache);
+  cache_put(&gsd->rsci->h, sn->rsc_cache);
  gsd->rsci = NULL;

  return stat;
@@ -1429,30 +1428,96 @@ static struct auth_ops svcauthops_gss = {
  .set_client = svcauth_gss_set_client,
 };

+static int rsi_cache_create_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;
+
+ cd = cache_create_net(&rsi_cache, net);
+ if (IS_ERR(cd))
+  return PTR_ERR(cd);
+ err = cache_register_net(cd, net);
+ if (err) {
+  cache_destroy_net(cd, net);
```

```
+  return err;
+ }
+ sn->rsi_cache = cd;
+ return 0;
+}
+
+static void rsi_cache_destroy_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd = sn->rsi_cache;
+
+ sn->rsi_cache = NULL;
+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
+}
+
+static int rsc_cache_create_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;
+
+ cd = cache_create_net(&rsc_cache, net);
+ if (IS_ERR(cd))
+  return PTR_ERR(cd);
+ err = cache_register_net(cd, net);
+ if (err) {
+  cache_destroy_net(cd, net);
+  return err;
+ }
+ sn->rsc_cache = cd;
+ return 0;
+}
+
+static void rsc_cache_destroy_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd = sn->rsc_cache;
+
+ sn->rsc_cache = NULL;
+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
+}
+
 int
-gss_svc_init(void)
```

```
+gss_svc_init_net(struct net *net)
 {
- int rv = svc_auth_register(RPC_AUTH_GSS, &svcauthops_gss);
+ int rv;
+
+ rv = rsc_cache_create_net(net);
  if (rv)
   return rv;
- rv = cache_register(&rsc_cache);
+ rv = rsi_cache_create_net(net);
  if (rv)
   goto out1;
- rv = cache_register(&rsi_cache);
- if (rv)
-  goto out2;
  return 0;
-out2:
- cache_unregister(&rsc_cache);
 out1:
- svc_auth_unregister(RPC_AUTH_GSS);
+ rsc_cache_destroy_net(net);
  return rv;
 }

 void
+gss_svc_shutdown_net(struct net *net)
+{
+ rsi_cache_destroy_net(net);
+ rsc_cache_destroy_net(net);
+}
+
+int
+gss_svc_init(void)
+{
+ return svc_auth_register(RPC_AUTH_GSS, &svcauthops_gss);
+}
+
+void
 gss_svc_shutdown(void)
 {
- cache_unregister(&rsc_cache);
- cache_unregister(&rsi_cache);
  svc_auth_unregister(RPC_AUTH_GSS);
 }
diff --git a/net/sunrpc/cache.c b/net/sunrpc/cache.c
index 9ef5926..a450b8a 100644
--- a/net/sunrpc/cache.c
+++ b/net/sunrpc/cache.c
```

```
@@ -1643,6 +1643,7 @@ int cache_register_net(struct cache_detail *cd, struct net *net)
   sunrpc_destroy_cache_detail(cd);
  return ret;
 }
+EXPORT_SYMBOL_GPL(cache_register_net);

 int cache_register(struct cache_detail *cd)
 {
@@ -1655,6 +1656,7 @@ void cache_unregister_net(struct cache_detail *cd, struct net *net)
  remove_cache_proc_entries(cd, net);
  sunrpc_destroy_cache_detail(cd);
 }
+EXPORT_SYMBOL_GPL(cache_unregister_net);

 void cache_unregister(struct cache_detail *cd)
 {
diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index 309f88d..ce7bd44 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -10,6 +10,8 @@ struct sunrpc_net {
  struct proc_dir_entry *proc_net_rpc;
  struct cache_detail *ip_map_cache;
  struct cache_detail *unix_gid_cache;
+ struct cache_detail *rsc_cache;
+ struct cache_detail *rsi_cache;

  struct super_block *pipefs_sb;
  struct mutex pipefs_sb_lock;
diff --git a/net/sunrpc/sunrpc_syms.c b/net/sunrpc/sunrpc_syms.c
index 38a72a1..d16ac08 100644
--- a/net/sunrpc/sunrpc_syms.c
+++ b/net/sunrpc/sunrpc_syms.c
@@ -25,6 +25,7 @@
 #include "netns.h"

 int sunrpc_net_id;
+EXPORT_SYMBOL_GPL(sunrpc_net_id);

 extern int unix_gid_cache_create(struct net *net);
 extern int unix_gid_cache_destroy(struct net *net);
```

Subject: [PATCH 4/5] SUNRPC: ip map cache per network namespace cleanup
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:49:31 GMT
View Forum Message <> Reply to Message

This patch converts ip_map_cache per network namespace implemenetation to the

same view, as other caches done in the series.
Besides generalization, code becomes shorter with this patch.

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

```
---
 net/sunrpc/svcauth_unix.c |   71 +++++++++++++++++++++----------------------------
 1 files changed, 30 insertions(+), 41 deletions(-)

diff --git a/net/sunrpc/svcauth_unix.c b/net/sunrpc/svcauth_unix.c
index c753ae4..0a50788 100644
--- a/net/sunrpc/svcauth_unix.c
+++ b/net/sunrpc/svcauth_unix.c
@@ -211,7 +211,7 @@ static int ip_map_parse(struct cache_detail *cd,
  len = qword_get(&mesg, buf, mlen);
  if (len <= 0) return -EINVAL;

- if (rpc_pton(&init_net, buf, len, &address.sa, sizeof(address)) == 0)
+ if (rpc_pton(cd->net, buf, len, &address.sa, sizeof(address)) == 0)
   return -EINVAL;
  switch (address.sa.sa_family) {
  case AF_INET:
@@ -875,56 +875,45 @@ struct auth_ops svcauth_unix = {
  .set_client = svcauth_unix_set_client,
 };

+struct cache_detail ip_map_cache = {
+ .owner  = THIS_MODULE,
+ .hash_size = IP_HASHMAX,
+ .name  = "auth.unix.ip",
+ .cache_put = ip_map_put,
+ .cache_upcall = ip_map_upcall,
+ .cache_parse = ip_map_parse,
+ .cache_show = ip_map_show,
+ .match  = ip_map_match,
+ .init  = ip_map_init,
+ .update  = update,
+ .alloc  = ip_map_alloc,
+};
+
 int ip_map_cache_create(struct net *net)
 {
- int err = -ENOMEM;
- struct cache_detail *cd;
- struct cache_head **tbl;
  struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;
```

```
- cd = kzalloc(sizeof(struct cache_detail), GFP_KERNEL);
- if (cd == NULL)
-  goto err_cd;
-
- tbl = kzalloc(IP_HASHMAX * sizeof(struct cache_head *), GFP_KERNEL);
- if (tbl == NULL)
-  goto err_tbl;
-
- cd->owner = THIS_MODULE,
- cd->hash_size = IP_HASHMAX,
- cd->hash_table = tbl,
- cd->name = "auth.unix.ip",
- cd->cache_put = ip_map_put,
- cd->cache_upcall = ip_map_upcall,
- cd->cache_parse = ip_map_parse,
- cd->cache_show = ip_map_show,
- cd->match = ip_map_match,
- cd->init = ip_map_init,
- cd->update = update,
- cd->alloc = ip_map_alloc,
-
+ cd = cache_create_net(&ip_map_cache, net);
+ if (IS_ERR(cd))
+  return PTR_ERR(cd);
  err = cache_register_net(cd, net);
- if (err)
-  goto err_reg;
-
+ if (err) {
+  cache_destroy_net(cd, net);
+  return err;
+ }
  sn->ip_map_cache = cd;
  return 0;
-
-err_reg:
- kfree(tbl);
-err_tbl:
- kfree(cd);
-err_cd:
- return err;
 }

 void ip_map_cache_destroy(struct net *net)
 {
- struct sunrpc_net *sn;
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
```

```
+ struct cache_detail *cd = sn->ip_map_cache;

- sn = net_generic(net, sunrpc_net_id);
- cache_purge(sn->ip_map_cache);
- cache_unregister_net(sn->ip_map_cache, net);
- kfree(sn->ip_map_cache->hash_table);
- kfree(sn->ip_map_cache);
+ sn->ip_map_cache = NULL;
+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
 }
```

Subject: [PATCH 5/5] SUNRPC: generic cache register routines removed
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 14:49:39 GMT
View Forum Message <> Reply to Message

All cache users now uses network-namespace-aware routines, so generic ones
are obsolete.

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

```
---
 include/linux/sunrpc/cache.h |    2 --
 net/sunrpc/cache.c           |   12 ------------
 2 files changed, 0 insertions(+), 14 deletions(-)

diff --git a/include/linux/sunrpc/cache.h b/include/linux/sunrpc/cache.h
index 259381c..f5fd616 100644
--- a/include/linux/sunrpc/cache.h
+++ b/include/linux/sunrpc/cache.h
@@ -198,9 +198,7 @@ extern void cache_flush(void);
 extern void cache_purge(struct cache_detail *detail);
 #define NEVER (0x7FFFFFFF)
 extern void __init cache_initialize(void);
-extern int cache_register(struct cache_detail *cd);
 extern int cache_register_net(struct cache_detail *cd, struct net *net);
-extern void cache_unregister(struct cache_detail *cd);
 extern void cache_unregister_net(struct cache_detail *cd, struct net *net);

 extern struct cache_detail *cache_create_net(struct cache_detail *tmpl, struct net *net);
diff --git a/net/sunrpc/cache.c b/net/sunrpc/cache.c
index a450b8a..f21ece0 100644
--- a/net/sunrpc/cache.c
+++ b/net/sunrpc/cache.c
@@ -1645,12 +1645,6 @@ int cache_register_net(struct cache_detail *cd, struct net *net)
 }
```

```
 EXPORT_SYMBOL_GPL(cache_register_net);

-int cache_register(struct cache_detail *cd)
-{
- return cache_register_net(cd, &init_net);
-}
-EXPORT_SYMBOL_GPL(cache_register);
-
 void cache_unregister_net(struct cache_detail *cd, struct net *net)
 {
  remove_cache_proc_entries(cd, net);
@@ -1658,12 +1652,6 @@ void cache_unregister_net(struct cache_detail *cd, struct net *net)
 }
 EXPORT_SYMBOL_GPL(cache_unregister_net);

-void cache_unregister(struct cache_detail *cd)
-{
- cache_unregister_net(cd, &init_net);
-}
-EXPORT_SYMBOL_GPL(cache_unregister);
-
 struct cache_detail *cache_create_net(struct cache_detail *tmpl, struct net *net)
 {
  struct cache_detail *cd;
```

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Myklebust, Trond on Thu, 19 Jan 2012 15:19:52 GMT
View Forum Message <> Reply to Message

On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
> One more step towards to "NFS in container".
> With this patch set caches are allocated per network namespace.
>
> The following series consists of:
>
> ---
>
> Stanislav Kinsbursky (5):
> SUNRPC: cache creation and destruction routines introduced
> SUNRPC: create unix gid cache per network namespace
> SUNRPC: create GSS auth cache per network namespace
> SUNRPC: ip map cache per network namespace cleanup
> SUNRPC: generic cache register routines removed

The patches look good, and I've applied them for now in my 'devel'
branch so we can test them, but I'd like to get an Ack/Nack from Bruce
before committing to merging them.

--
Trond Myklebust
Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com
www.netapp.com

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 15:31:56 GMT
View Forum Message <> Reply to Message

19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
>> One more step towards to "NFS in container".
>> With this patch set caches are allocated per network namespace.
>>
>> The following series consists of:
>>
>> ---
>>
>> Stanislav Kinsbursky (5):
>> SUNRPC: cache creation and destruction routines introduced
>> SUNRPC: create unix gid cache per network namespace
>> SUNRPC: create GSS auth cache per network namespace
>> SUNRPC: ip map cache per network namespace cleanup
>> SUNRPC: generic cache register routines removed
>
> The patches look good, and I've applied them for now in my 'devel'
> branch so we can test them, but I'd like to get an Ack/Nack from Bruce
> before committing to merging them.
>

Cool, thanks.
BTW, these patches depends on patch "NFSd: use network-namespace-aware cache registering routines", which Bruce took already (and it's even in linux-next already), but not in your devel tree for some reason.

--
Best regards,
Stanislav Kinsbursky

Subject: RE: [PATCH 0/5] SUNRPC: make caches network namespace aware

Posted by Myklebust, Trond on Thu, 19 Jan 2012 15:34:02 GMT
View Forum Message <> Reply to Message

> -----Original Message-----
> From: Stanislav Kinsbursky [mailto:skinsbursky@parallels.com]
> Sent: Thursday, January 19, 2012 10:32 AM
> To: Myklebust, Trond
> Cc: Dr James Bruce Fields; linux-nfs@vger.kernel.org; Pavel Emelianov;
> neilb@suse.de; netdev@vger.kernel.org; linux-kernel@vger.kernel.org;
> James Bottomley; bfields@fieldses.org; davem@davemloft.net;
> devel@openvz.org
> Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
>
> 19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
> > On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
> >> One more step towards to "NFS in container".
> >> With this patch set caches are allocated per network namespace.
> >>
> >> The following series consists of:
> >>
> >> ---
> >>
> >> Stanislav Kinsbursky (5):
> >> SUNRPC: cache creation and destruction routines introduced
> >> SUNRPC: create unix gid cache per network namespace
> >> SUNRPC: create GSS auth cache per network namespace
> >> SUNRPC: ip map cache per network namespace cleanup
> >> SUNRPC: generic cache register routines removed
> >
> > The patches look good, and I've applied them for now in my 'devel'
> > branch so we can test them, but I'd like to get an Ack/Nack from Bruce
> > before committing to merging them.
> >
>
> Cool, thanks.
> BTW, these patches depends on patch "NFSd: use network-namespace-
> aware cache registering routines", which Bruce took already (and it's even in
> linux-next already), but not in your devel tree for some reason.

In that case, would it make more sense to carry them in Bruce's tree?

--
Trond Myklebust
Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com
www.netapp.com

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 15:43:22 GMT
View Forum Message <> Reply to Message

19.01.2012 19:34, Myklebust, Trond Ð¿Ð¸ÑˆÐµÑ‚:
>> -----Original Message-----
>> From: Stanislav Kinsbursky [mailto:skinsbursky@parallels.com]
>> Sent: Thursday, January 19, 2012 10:32 AM
>> To: Myklebust, Trond
>> Cc: Dr James Bruce Fields; linux-nfs@vger.kernel.org; Pavel Emelianov;
>> neilb@suse.de; netdev@vger.kernel.org; linux-kernel@vger.kernel.org;
>> James Bottomley; bfields@fieldses.org; davem@davemloft.net;
>> devel@openvz.org
>> Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
>>
>> 19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
>>> On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
>>>> One more step towards to "NFS in container".
>>>> With this patch set caches are allocated per network namespace.
>>>>
>>>> The following series consists of:
>>>>
>>>> ---
>>>>
>>>> Stanislav Kinsbursky (5):
>>>> SUNRPC: cache creation and destruction routines introduced
>>>> SUNRPC: create unix gid cache per network namespace
>>>> SUNRPC: create GSS auth cache per network namespace
>>>> SUNRPC: ip map cache per network namespace cleanup
>>>> SUNRPC: generic cache register routines removed
>>>
>>> The patches look good, and I've applied them for now in my 'devel'
>>> branch so we can test them, but I'd like to get an Ack/Nack from Bruce
>>> before committing to merging them.
>>>
>>
>> Cool, thanks.
>> BTW, these patches depends on patch "NFSd: use network-namespace-
>> aware cache registering routines", which Bruce took already (and it's even in
>> linux-next already), but not in your devel tree for some reason.
>
> In that case, would it make more sense to carry them in Bruce's tree?
>

It's up to you, of course. But, frankly, I don't think so.
To be more precise, only the last patch ("SUNRPC: generic cache register
routines removed") depends on the patch, which is in linux-next already. IOW,
those patch removed NFSd dependence on cache_register() and cache_unregister()
routines and was commited to linux-next at Wed, 7 Dec 2011.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by bfields on Thu, 19 Jan 2012 15:47:31 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 07:31:56PM +0400, Stanislav Kinsbursky wrote:
> 19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
> >On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
> >>One more step towards to "NFS in container".
> >>With this patch set caches are allocated per network namespace.
> >>
> >>The following series consists of:
> >>
> >>---
> >>
> >>Stanislav Kinsbursky (5):
> >> SUNRPC: cache creation and destruction routines introduced
> >> SUNRPC: create unix gid cache per network namespace
> >> SUNRPC: create GSS auth cache per network namespace
> >> SUNRPC: ip map cache per network namespace cleanup
> >> SUNRPC: generic cache register routines removed
> >
> >The patches look good, and I've applied them for now in my 'devel'
> >branch so we can test them, but I'd like to get an Ack/Nack from Bruce
> >before committing to merging them.
> >
>
> Cool, thanks.
> BTW, these patches depends on patch "NFSd: use
> network-namespace-aware cache registering routines", which Bruce
> took already (and it's even in linux-next already), but not in your
> devel tree for some reason.

And Linus has pulled those, so if Trond bases his tree on -rc1, he'll
have "NFSd: use network-namespace-aware cache registering routines".

You know best what you're planning to submit over the next cycle, and
what the dependencies are, and which side (client or server) it's likely
to touch more. So, your choice, I think. If necessary Trond and I can
probably do a merge between us at some point.

Either way, I'll go take a look at them now....--b.

--b.

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 16:01:33 GMT
View Forum Message <> Reply to Message

19.01.2012 19:47, bfields@fieldses.org Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, Jan 19, 2012 at 07:31:56PM +0400, Stanislav Kinsbursky wrote:
>> 19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
>>> On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
>>>> One more step towards to "NFS in container".
>>>> With this patch set caches are allocated per network namespace.
>>>>
>>>> The following series consists of:
>>>>
>>>> ---
>>>>
>>>> Stanislav Kinsbursky (5):
>>>> SUNRPC: cache creation and destruction routines introduced
>>>> SUNRPC: create unix gid cache per network namespace
>>>> SUNRPC: create GSS auth cache per network namespace
>>>> SUNRPC: ip map cache per network namespace cleanup
>>>> SUNRPC: generic cache register routines removed
>>>
>>> The patches look good, and I've applied them for now in my 'devel'
>>> branch so we can test them, but I'd like to get an Ack/Nack from Bruce
>>> before committing to merging them.
>>>
>>
>> Cool, thanks.
>> BTW, these patches depends on patch "NFSd: use
>> network-namespace-aware cache registering routines", which Bruce
>> took already (and it's even in linux-next already), but not in your
>> devel tree for some reason.
>
> And Linus has pulled those, so if Trond bases his tree on -rc1, he'll
> have "NFSd: use network-namespace-aware cache registering routines".
>
> You know best what you're planning to submit over the next cycle, and
> what the dependencies are, and which side (client or server) it's likely
> to touch more. So, your choice, I think. If necessary Trond and I can
> probably do a merge between us at some point.
>

Actually, I'm not going to patch NFSd in this cycle. Some more NFS and Lockd
patches will follow soon.

So, probably, Trond's tree suits better.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 2/5] SUNRPC: create unix gid cache per network namespace
Posted by bfields on Thu, 19 Jan 2012 16:24:31 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 06:49:14PM +0400, Stanislav Kinsbursky wrote:
> This patch makes unix_gid_cache cache detail allocated and registered per
> network namespace context.
> Thus with this patch unix_gid_cache contents for network namespace "X" are
> controlled from proc file system mount for the same network namespace "X".

I'm confused--this seems half-done. I see that you're creating the gid
caches dynamically, but there's still this static definition:

```
>  struct cache_detail unix_gid_cache = {
>   .owner  = THIS_MODULE,
>   .hash_size = GID_HASHMAX,
> - .hash_table = gid_table,
>   .name  = "auth.unix.gid",
>   .cache_put = unix_gid_put,
>   .cache_upcall = unix_gid_upcall,
```

And it looks like unix_gid_lookup is still using this static
unix_gid_cache. Maybe I'm missing something....

--b.

Subject: Re: [PATCH 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by bfields on Thu, 19 Jan 2012 16:31:49 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 06:49:23PM +0400, Stanislav Kinsbursky wrote:
> @@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
> struct xdr_netobj tmpobj;
> struct rsi *rsip, rsikey;
> int ret;
> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

OK, so you're getting the network namespace out of the rqstp, and, then
passing it down, makes sense. And:

> @@ -1079,6 +1076,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
> __be32 *rpcstart;
> __be32 *reject_stat = resv->iov_base + resv->iov_len;
> int ret;
> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

... same for the gss cache. Looks good.

How do you plan to test this?

--b.

Subject: Re: [PATCH 4/5] SUNRPC: ip map cache per network namespace cleanup
Posted by bfields on Thu, 19 Jan 2012 16:37:22 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 06:49:31PM +0400, Stanislav Kinsbursky wrote:
> This patch converts ip_map_cache per network namespace implemenetation to the
> same view, as other caches done in the series.
> Besides generalization, code becomes shorter with this patch.

Again, isnt' it ip_map_cache, and not one of the dynamically created
cache details, that's being passed to the lookup routines?

I may just be confused....

--b.

>
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
>
> ---
> net/sunrpc/svcauth_unix.c | 71 +++++++++++++++++++++----------------------
> 1 files changed, 30 insertions(+), 41 deletions(-)
>
> diff --git a/net/sunrpc/svcauth_unix.c b/net/sunrpc/svcauth_unix.c
> index c753ae4..0a50788 100644
> --- a/net/sunrpc/svcauth_unix.c
> +++ b/net/sunrpc/svcauth_unix.c
> @@ -211,7 +211,7 @@ static int ip_map_parse(struct cache_detail *cd,
> len = qword_get(&mesg, buf, mlen);
> if (len <= 0) return -EINVAL;
>
> - if (rpc_pton(&init_net, buf, len, &address.sa, sizeof(address)) == 0)

```
> + if (rpc_pton(cd->net, buf, len, &address.sa, sizeof(address)) == 0)
>    return -EINVAL;
>   switch (address.sa.sa_family) {
>   case AF_INET:
> @@ -875,56 +875,45 @@ struct auth_ops svcauth_unix = {
>   .set_client = svcauth_unix_set_client,
>  };
>
> +struct cache_detail ip_map_cache = {
> + .owner  = THIS_MODULE,
> + .hash_size = IP_HASHMAX,
> + .name  = "auth.unix.ip",
> + .cache_put = ip_map_put,
> + .cache_upcall = ip_map_upcall,
> + .cache_parse = ip_map_parse,
> + .cache_show = ip_map_show,
> + .match  = ip_map_match,
> + .init  = ip_map_init,
> + .update  = update,
> + .alloc  = ip_map_alloc,
> +};
> +
>  int ip_map_cache_create(struct net *net)
>  {
> - int err = -ENOMEM;
> - struct cache_detail *cd;
> - struct cache_head **tbl;
>   struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
> + struct cache_detail *cd;
> + int err;
>
> - cd = kzalloc(sizeof(struct cache_detail), GFP_KERNEL);
> - if (cd == NULL)
> -  goto err_cd;
> -
> - tbl = kzalloc(IP_HASHMAX * sizeof(struct cache_head *), GFP_KERNEL);
> - if (tbl == NULL)
> -  goto err_tbl;
> -
> - cd->owner = THIS_MODULE,
> - cd->hash_size = IP_HASHMAX,
> - cd->hash_table = tbl,
> - cd->name = "auth.unix.ip",
> - cd->cache_put = ip_map_put,
> - cd->cache_upcall = ip_map_upcall,
> - cd->cache_parse = ip_map_parse,
> - cd->cache_show = ip_map_show,
> - cd->match = ip_map_match,
```

```
> - cd->init = ip_map_init,
> - cd->update = update,
> - cd->alloc = ip_map_alloc,
> -
> + cd = cache_create_net(&ip_map_cache, net);
> + if (IS_ERR(cd))
> +  return PTR_ERR(cd);
>   err = cache_register_net(cd, net);
> - if (err)
> -  goto err_reg;
> -
> + if (err) {
> +  cache_destroy_net(cd, net);
> +  return err;
> + }
>   sn->ip_map_cache = cd;
>   return 0;
> -
> -err_reg:
> - kfree(tbl);
> -err_tbl:
> - kfree(cd);
> -err_cd:
> - return err;
>  }
>
>  void ip_map_cache_destroy(struct net *net)
>  {
> - struct sunrpc_net *sn;
> + struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
> + struct cache_detail *cd = sn->ip_map_cache;
>
> - sn = net_generic(net, sunrpc_net_id);
> - cache_purge(sn->ip_map_cache);
> - cache_unregister_net(sn->ip_map_cache, net);
> - kfree(sn->ip_map_cache->hash_table);
> - kfree(sn->ip_map_cache);
> + sn->ip_map_cache = NULL;
> + cache_purge(cd);
> + cache_unregister_net(cd, net);
> + cache_destroy_net(cd, net);
>  }
>
```

Subject: Re: [PATCH 2/5] SUNRPC: create unix gid cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:00:23 GMT

View Forum Message <> Reply to Message

19.01.2012 20:24, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, Jan 19, 2012 at 06:49:14PM +0400, Stanislav Kinsbursky wrote:
>> This patch makes unix_gid_cache cache detail allocated and registered per
>> network namespace context.
>> Thus with this patch unix_gid_cache contents for network namespace "X" are
>> controlled from proc file system mount for the same network namespace "X".
>
> I'm confused--this seems half-done. I see that you're creating the gid
> caches dynamically, but there's still this static definition:
>

This static definition is used as a template now.
Have a look at the first path of the series: cache creation routine duplicates
incomming cache_detail and initializes net and hash_table fields in freshly
created copy.
This copy is the new per-net cache_detail instance.

>> struct cache_detail unix_gid_cache = {
>> .owner = THIS_MODULE,
>> .hash_size = GID_HASHMAX,
>> - .hash_table = gid_table,
>> .name = "auth.unix.gid",
>> .cache_put = unix_gid_put,
>> .cache_upcall = unix_gid_upcall,
>
> And it looks like unix_gid_lookup is still using this static
> unix_gid_cache. Maybe I'm missing something....
>

Yep, thanks for the catch.
My fault, I've missed this usage.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:04:40 GMT
View Forum Message <> Reply to Message

19.01.2012 20:31, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, Jan 19, 2012 at 06:49:23PM +0400, Stanislav Kinsbursky wrote:
>> @@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,

>> struct xdr_netobj tmpobj;
>> struct rsi *rsip, rsikey;
>> int ret;
>> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
>
> OK, so you're getting the network namespace out of the rqstp, and, then
> passing it down, makes sense. And:
>
>> @@ -1079,6 +1076,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
>> __be32 *rpcstart;
>> __be32 *reject_stat = resv->iov_base + resv->iov_len;
>> int ret;
>> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
>
> ... same for the gss cache. Looks good.
>
> How do you plan to test this?
>

Do you mean something special or in general?
Currently I validate all my chages in container by using simple test environment.
I would be appreciate for any hints to tests, than can help.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 4/5] SUNRPC: ip map cache per network namespace cleanup
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:10:08 GMT
View Forum Message <> Reply to Message

19.01.2012 20:37, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, Jan 19, 2012 at 06:49:31PM +0400, Stanislav Kinsbursky wrote:
>> This patch converts ip_map_cache per network namespace implemenetation to the
>> same view, as other caches done in the series.
>> Besides generalization, code becomes shorter with this patch.
>
> Again, isnt' it ip_map_cache, and not one of the dynamically created
> cache details, that's being passed to the lookup routines?
>
> I may just be confused....
>

Nope. ip_map_cache was made per network namespace a long time ago.
But it was done in other way: no static templates - just allocation and
initialization.

But static template looks clearer to me. So I've used this approach for other caches. And decided to update ip_map_cache management in the same way.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:14:15 GMT
View Forum Message <> Reply to Message

Sorry, this patch set has some bugs.
I'll resend fixed version soon.

--
Best regards,
Stanislav Kinsbursky

Subject: RE: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by Myklebust, Trond on Thu, 19 Jan 2012 17:34:11 GMT
View Forum Message <> Reply to Message

> -----Original Message-----
> From: bfields@fieldses.org [mailto:bfields@fieldses.org]
> Sent: Thursday, January 19, 2012 10:48 AM
> To: Stanislav Kinsbursky
> Cc: Myklebust, Trond; Dr James Bruce Fields; linux-nfs@vger.kernel.org;
> Pavel Emelianov; neilb@suse.de; netdev@vger.kernel.org; linux-
> kernel@vger.kernel.org; James Bottomley; davem@davemloft.net;
> devel@openvz.org
> Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
>
> On Thu, Jan 19, 2012 at 07:31:56PM +0400, Stanislav Kinsbursky wrote:
> > 19.01.2012 19:19, Trond Myklebust Ð¿Ð¸ÑˆÐµÑ‚:
> > >On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
> > >>One more step towards to "NFS in container".
> > >>With this patch set caches are allocated per network namespace.
> > >>
> > >>The following series consists of:
> > >>
> > >>---
> > >>
> > >>Stanislav Kinsbursky (5):
> > >> SUNRPC: cache creation and destruction routines introduced
> > >> SUNRPC: create unix gid cache per network namespace

> > >> SUNRPC: create GSS auth cache per network namespace
> > >> SUNRPC: ip map cache per network namespace cleanup
> > >> SUNRPC: generic cache register routines removed
> > >
> > >The patches look good, and I've applied them for now in my 'devel'
> > >branch so we can test them, but I'd like to get an Ack/Nack from
> > >Bruce before committing to merging them.
> > >
> >
> > Cool, thanks.
> > BTW, these patches depends on patch "NFSd: use network-namespace-
> aware
> > cache registering routines", which Bruce took already (and it's even
> > in linux-next already), but not in your devel tree for some reason.
>
> And Linus has pulled those, so if Trond bases his tree on -rc1, he'll have
> "NFSd: use network-namespace-aware cache registering routines".

OK. In that case I should have them...

Subject: Re: [PATCH 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by bfields on Thu, 19 Jan 2012 17:40:02 GMT

View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 09:04:40PM +0400, Stanislav Kinsbursky wrote:
> 19.01.2012 20:31, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> >On Thu, Jan 19, 2012 at 06:49:23PM +0400, Stanislav Kinsbursky wrote:
> >>@@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
> >> struct xdr_netobj tmpobj;
> >> struct rsi *rsip, rsikey;
> >> int ret;
> >>+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
> >
> >OK, so you're getting the network namespace out of the rqstp, and, then
> >passing it down, makes sense. And:
> >
> >>@@ -1079,6 +1076,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
> >> __be32 *rpcstart;
> >> __be32 *reject_stat = resv->iov_base + resv->iov_len;
> >> int ret;
> >>+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
> >
> >... same for the gss cache. Looks good.
> >
> >How do you plan to test this?
> >

>
> Do you mean something special or in general?
> Currently I validate all my chages in container by using simple test environment.
> I would be appreciate for any hints to tests, than can help.

The server needs to be tested after these changes, and we need to make
sure the caches affected still work.

(I suspect the gid cache code will oops if it's used after these
patches, since the table is left NULl?)

And then we should also test in a container environment, with different
instances of mountd and rpc.svcidmapd running in each container, to
verify that the right thing happens.

--b.

Subject: Re: [PATCH 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:51:59 GMT
View Forum Message <> Reply to Message

19.01.2012 21:40, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> On Thu, Jan 19, 2012 at 09:04:40PM +0400, Stanislav Kinsbursky wrote:
>> 19.01.2012 20:31, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
>>> On Thu, Jan 19, 2012 at 06:49:23PM +0400, Stanislav Kinsbursky wrote:
>>>> @@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
>>>> struct xdr_netobj tmpobj;
>>>> struct rsi *rsip, rsikey;
>>>> int ret;
>>>> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
>>>
>>> OK, so you're getting the network namespace out of the rqstp, and, then
>>> passing it down, makes sense. And:
>>>
>>>> @@ -1079,6 +1076,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
>>>> __be32 *rpcstart;
>>>> __be32 *reject_stat = resv->iov_base + resv->iov_len;
>>>> int ret;
>>>> + struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);
>>>
>>> ... same for the gss cache. Looks good.
>>>
>>> How do you plan to test this?
>>>
>>
>> Do you mean something special or in general?

>> Currently I validate all my chages in container by using simple test environment.
>> I would be appreciate for any hints to tests, than can help.
>
> The server needs to be tested after these changes, and we need to make
> sure the caches affected still work.
>
> (I suspect the gid cache code will oops if it's used after these
> patches, since the table is left NULl?)
>
> And then we should also test in a container environment, with different
> instances of mountd and rpc.svcidmapd running in each container, to
> verify that the right thing happens.
>

Ok. I'll test server with 2-nd version of these patches tomorrow and reply with
the results.
Thanks, Bruce.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 4/5] SUNRPC: ip map cache per network namespace cleanup
Posted by bfields on Thu, 19 Jan 2012 22:21:01 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 09:10:08PM +0400, Stanislav Kinsbursky wrote:
> 19.01.2012 20:37, J. Bruce Fields Ð¿Ð¸ÑˆÐµÑ‚:
> >On Thu, Jan 19, 2012 at 06:49:31PM +0400, Stanislav Kinsbursky wrote:
> >>This patch converts ip_map_cache per network namespace implemenetation to the
> >>same view, as other caches done in the series.
> >>Besides generalization, code becomes shorter with this patch.
> >
> >Again, isnt' it ip_map_cache, and not one of the dynamically created
> >cache details, that's being passed to the lookup routines?
> >
> >I may just be confused....
> >
>
> Nope. ip_map_cache was made per network namespace a long time ago.
> But it was done in other way: no static templates - just allocation
> and initialization.
> But static template looks clearer to me. So I've used this approach
> for other caches. And decided to update ip_map_cache management in
> the same way.

Going and reading the code.... OK, I see what you mean, makes sense.
Thanks!

--b.

Subject: Re: [PATCH 0/5] SUNRPC: make caches network namespace aware
Posted by bfields on Thu, 19 Jan 2012 22:40:02 GMT
View Forum Message <> Reply to Message

On Thu, Jan 19, 2012 at 10:19:52AM -0500, Trond Myklebust wrote:
> On Thu, 2012-01-19 at 18:48 +0400, Stanislav Kinsbursky wrote:
> > One more step towards to "NFS in container".
> > With this patch set caches are allocated per network namespace.
> >
> > The following series consists of:
> >
> > ---
> >
> > Stanislav Kinsbursky (5):
> > SUNRPC: cache creation and destruction routines introduced
> > SUNRPC: create unix gid cache per network namespace
> > SUNRPC: create GSS auth cache per network namespace
> > SUNRPC: ip map cache per network namespace cleanup
> > SUNRPC: generic cache register routines removed
>
> The patches look good, and I've applied them for now in my 'devel'
> branch so we can test them, but I'd like to get an Ack/Nack from Bruce
> before committing to merging them.

They look good to me. I'd like to hear how they've been tested--that
done, they'll have my ack.

--b.