Subject: [PATCH v2 4/5] SUNRPC: ip map cache per network namespace cleanup
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:42:45 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 a6eef38..bcd574f 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:
@@ -876,56 +876,45 @@ struct auth_ops svcauth_unix = {
  .set_client = svcauth_unix_set_client,
 };

+static struct cache_detail ip_map_cache_template = {
+ .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_template, 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);
 }
```