Subject: Re: [PATCH v3] SUNRPC: protect service sockets lists during per-net shutdown
Posted by bfields on Tue, 21 Aug 2012 19:06:46 GMT
View Forum Message <> Reply to Message

On Thu, Aug 16, 2012 at 03:29:03PM -0400, J. Bruce Fields wrote:
> Looking back at this:
>
> - adding the sv_lock looks like the right thing to do anyway
> independent of containers, because svc_age_temp_xprts may
> still be running.

This is what I've been testing with.

Or alternatively if you'd rather strip out the other stuff from your
patch I could take that instead.

--b.

commit 719f8bcc883e7992615f4d5625922e24995e2d98
Author: J. Bruce Fields <bfields@redhat.com>
Date: Mon Aug 13 17:03:00 2012 -0400

svcrpc: fix xpt_list traversal locking on shutdown

Server threads are not running at this point, but svc_age_temp_xprts
still may be, so we need this locking.

Signed-off-by: J. Bruce Fields <bfields@redhat.com>

```
diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
index bac973a..e1810b9 100644
--- a/net/sunrpc/svc_xprt.c
+++ b/net/sunrpc/svc_xprt.c
@@ -917,16 +917,18 @@ void svc_close_xprt(struct svc_xprt *xprt)
 }
 EXPORT_SYMBOL_GPL(svc_close_xprt);

-static void svc_close_list(struct list_head *xprt_list, struct net *net)
+static void svc_close_list(struct svc_serv *serv, struct list_head *xprt_list, struct net *net)
 {
  struct svc_xprt *xprt;

+ spin_lock(&serv->sv_lock);
  list_for_each_entry(xprt, xprt_list, xpt_list) {
   if (xprt->xpt_net != net)
    continue;
   set_bit(XPT_CLOSE, &xprt->xpt_flags);
```

```
   set_bit(XPT_BUSY, &xprt->xpt_flags);
  }
+ spin_unlock(&serv->sv_lock);
 }

 static void svc_clear_pools(struct svc_serv *serv, struct net *net)
@@ -949,24 +951,28 @@ static void svc_clear_pools(struct svc_serv *serv, struct net *net)
  }
 }

-static void svc_clear_list(struct list_head *xprt_list, struct net *net)
+static void svc_clear_list(struct svc_serv *serv, struct list_head *xprt_list, struct net *net)
 {
  struct svc_xprt *xprt;
  struct svc_xprt *tmp;
+ LIST_HEAD(victims);

+ spin_lock(&serv->sv_lock);
  list_for_each_entry_safe(xprt, tmp, xprt_list, xpt_list) {
   if (xprt->xpt_net != net)
    continue;
-  svc_delete_xprt(xprt);
+  list_move(&xprt->xpt_list, &victims);
  }
- list_for_each_entry(xprt, xprt_list, xpt_list)
-  BUG_ON(xprt->xpt_net == net);
+ spin_unlock(&serv->sv_lock);
+
+ list_for_each_entry_safe(xprt, tmp, &victims, xpt_list)
+  svc_delete_xprt(xprt);
 }

 void svc_close_net(struct svc_serv *serv, struct net *net)
 {
- svc_close_list(&serv->sv_tempsocks, net);
- svc_close_list(&serv->sv_permsocks, net);
+ svc_close_list(serv, &serv->sv_tempsocks, net);
+ svc_close_list(serv, &serv->sv_permsocks, net);

  svc_clear_pools(serv, net);
  /*
@@ -974,8 +980,8 @@ void svc_close_net(struct svc_serv *serv, struct net *net)
   * svc_xprt_enqueue will not add new entries without taking the
   * sp_lock and checking XPT_BUSY.
   */
- svc_clear_list(&serv->sv_tempsocks, net);
- svc_clear_list(&serv->sv_permsocks, net);
+ svc_clear_list(serv, &serv->sv_tempsocks, net);
```

```
+ svc_clear_list(serv, &serv->sv_permsocks, net);
 }

 /*
```