Subject: [PATCH 08/11] SUNRPC: use per-net xs tunables instead of static ones
Posted by Stanislav Kinsbursky on Wed, 14 Dec 2011 10:47:51 GMT
View Forum Message <> Reply to Message

This patch replaces references to static tunables with per-net ones.

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

---
 net/sunrpc/xprtsock.c | 42 ++++++++++++++++++++++++++----------------
 1 files changed, 26 insertions(+), 16 deletions(-)

```
diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
index fe4b04b..49219d2 100644
--- a/net/sunrpc/xprtsock.c
+++ b/net/sunrpc/xprtsock.c
@@ -1458,6 +1458,7 @@ static void xs_sock_mark_closed(struct rpc_xprt *xprt)
 static void xs_tcp_state_change(struct sock *sk)
 {
  struct rpc_xprt *xprt;
+ struct sunrpc_net *sn = net_generic(sk->sk_net, sunrpc_net_id);

  read_lock_bh(&sk->sk_callback_lock);
  if (!(xprt = xprt_from_sock(sk)))
@@ -1496,7 +1497,7 @@ static void xs_tcp_state_change(struct sock *sk)
   clear_bit(XPRT_CONNECTED, &xprt->state);
   clear_bit(XPRT_CLOSE_WAIT, &xprt->state);
   smp_mb__after_clear_bit();
-  xs_tcp_schedule_linger_timeout(xprt, xs_tcp_fin_timeout);
+  xs_tcp_schedule_linger_timeout(xprt, sn->xs_tcp_fin_timeout);
   break;
  case TCP_CLOSE_WAIT:
   /* The server initiated a shutdown of the socket */
@@ -1512,7 +1513,7 @@ static void xs_tcp_state_change(struct sock *sk)
   break;
  case TCP_LAST_ACK:
   set_bit(XPRT_CLOSING, &xprt->state);
-  xs_tcp_schedule_linger_timeout(xprt, xs_tcp_fin_timeout);
+  xs_tcp_schedule_linger_timeout(xprt, sn->xs_tcp_fin_timeout);
   smp_mb__before_clear_bit();
   clear_bit(XPRT_CONNECTED, &xprt->state);
   smp_mb__after_clear_bit();
@@ -1652,11 +1653,13 @@ static void xs_udp_timer(struct rpc_task *task)
  xprt_adjust_cwnd(task, -ETIMEDOUT);
 }

-static unsigned short xs_get_random_port(void)
+static unsigned short xs_get_random_port(struct net *net)
```

```
 {
- unsigned short range = xprt_max_resvport - xprt_min_resvport;
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+
+ unsigned short range = sn->xprt_max_resvport - sn->xprt_min_resvport;
  unsigned short rand = (unsigned short) net_random() % range;
- return rand + xprt_min_resvport;
+ return rand + sn->xprt_min_resvport;
 }

 /**
@@ -1678,18 +1681,21 @@ static unsigned short xs_get_srcport(struct sock_xprt *transport)
  unsigned short port = transport->srcport;

  if (port == 0 && transport->xprt.resvport)
-  port = xs_get_random_port();
+  port = xs_get_random_port(transport->xprt.xprt_net);
  return port;
 }

 static unsigned short xs_next_srcport(struct sock_xprt *transport, unsigned short port)
 {
+ struct sunrpc_net *sn = net_generic(transport->xprt.xprt_net,
+         sunrpc_net_id);
+
  if (transport->srcport != 0)
   transport->srcport = 0;
  if (!transport->xprt.resvport)
   return 0;
- if (port <= xprt_min_resvport || port > xprt_max_resvport)
-  return xprt_max_resvport;
+ if (port <= sn->xprt_min_resvport || port > sn->xprt_max_resvport)
+  return sn->xprt_max_resvport;
  return --port;
 }
 static int xs_bind(struct sock_xprt *transport, struct socket *sock)
@@ -2541,9 +2547,10 @@ static struct rpc_xprt *xs_setup_local(struct xprt_create *args)
  struct sock_xprt *transport;
  struct rpc_xprt *xprt;
  struct rpc_xprt *ret;
+ struct sunrpc_net *sn = net_generic(args->net, sunrpc_net_id);

- xprt = xs_setup_xprt(args, xprt_tcp_slot_table_entries,
-   xprt_max_tcp_slot_table_entries);
+ xprt = xs_setup_xprt(args, sn->xprt_tcp_slot_table_entries,
+   sn->xprt_max_tcp_slot_table_entries);
  if (IS_ERR(xprt))
   return xprt;
```

```
  transport = container_of(xprt, struct sock_xprt, xprt);
@@ -2606,9 +2613,10 @@ static struct rpc_xprt *xs_setup_udp(struct xprt_create *args)
  struct rpc_xprt *xprt;
  struct sock_xprt *transport;
  struct rpc_xprt *ret;
+ struct sunrpc_net *sn = net_generic(args->net, sunrpc_net_id);

- xprt = xs_setup_xprt(args, xprt_udp_slot_table_entries,
-    xprt_udp_slot_table_entries);
+ xprt = xs_setup_xprt(args, sn->xprt_udp_slot_table_entries,
+    sn->xprt_udp_slot_table_entries);
  if (IS_ERR(xprt))
   return xprt;
  transport = container_of(xprt, struct sock_xprt, xprt);
@@ -2683,9 +2691,10 @@ static struct rpc_xprt *xs_setup_tcp(struct xprt_create *args)
  struct rpc_xprt *xprt;
  struct sock_xprt *transport;
  struct rpc_xprt *ret;
+ struct sunrpc_net *sn = net_generic(args->net, sunrpc_net_id);

- xprt = xs_setup_xprt(args, xprt_tcp_slot_table_entries,
-    xprt_max_tcp_slot_table_entries);
+ xprt = xs_setup_xprt(args, sn->xprt_tcp_slot_table_entries,
+    sn->xprt_max_tcp_slot_table_entries);
  if (IS_ERR(xprt))
   return xprt;
  transport = container_of(xprt, struct sock_xprt, xprt);
@@ -2754,6 +2763,7 @@ static struct rpc_xprt *xs_setup_bc_tcp(struct xprt_create *args)
  struct sock_xprt *transport;
  struct svc_sock *bc_sock;
  struct rpc_xprt *ret;
+ struct sunrpc_net *sn = net_generic(args->net, sunrpc_net_id);

  if (args->bc_xprt->xpt_bc_xprt) {
   /*
@@ -2764,8 +2774,8 @@ static struct rpc_xprt *xs_setup_bc_tcp(struct xprt_create *args)
    */
    return args->bc_xprt->xpt_bc_xprt;
  }
- xprt = xs_setup_xprt(args, xprt_tcp_slot_table_entries,
-    xprt_tcp_slot_table_entries);
+ xprt = xs_setup_xprt(args, sn->xprt_tcp_slot_table_entries,
+    sn->xprt_tcp_slot_table_entries);
  if (IS_ERR(xprt))
   return xprt;
  transport = container_of(xprt, struct sock_xprt, xprt);
```