

---

Subject: [PATCH v2 2/4] SUNRPC: use virtualized rpcbind internals instead of static ones

Posted by [Stanislav Kinsbursky](#) on Tue, 25 Oct 2011 13:59:41 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

This patch makes rpcbind logic works in network namespace context. IOW each network namespace will have it's own unique rpcbind internals (clients and friends) which is required for registering svc services per network namespace.

Signed-off-by: Stanislav Kinsbursky <[skinsbursky@parallels.com](mailto:skinsbursky@parallels.com)>

---

net/sunrpc/rpcb\_clnt.c | 64 ++++++-----  
1 files changed, 35 insertions(+), 29 deletions(-)

diff --git a/net/sunrpc/rpcb\_clnt.c b/net/sunrpc/rpcb\_clnt.c

index c2054ae..18eba8e 100644

--- a/net/sunrpc/rpcb\_clnt.c

+++ b/net/sunrpc/rpcb\_clnt.c

@@ -23,12 +23,15 @@

#include <linux/errno.h>

#include <linux/mutex.h>

#include <linux/slab.h>

+#include <linux/nsproxy.h>

#include <net/ipv6.h>

#include <linux/sunrpc/clnt.h>

#include <linux/sunrpc/sched.h>

#include <linux/sunrpc/xprtsock.h>

+#include "netns.h"

+

#ifdef RPC\_DEBUG

# define RPCDBG\_FACILITY RPCDBG\_BIND

#endif

@@ -111,12 +114,6 @@ static void rpcb\_getport\_done(struct rpc\_task \*, void \*);

static void rpcb\_map\_release(void \*data);

static struct rpc\_program rpcb\_program;

-static struct rpc\_clnt \* rpcb\_local\_clnt;

-static struct rpc\_clnt \* rpcb\_local\_clnt4;

-

-DEFINE\_SPINLOCK(rpcb\_clnt\_lock);

-unsigned int rpcb\_users;

-

struct rpcbind\_args {  
 struct rpc\_xprt \* r\_xprt;

```

@@ -167,29 +164,31 @@ static void rpcb_map_release(void *data)
static int rpcb_get_local(void)
{
    int cnt;
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

- spin_lock(&rpcb_clnt_lock);
- if (rpcb_users)
-   rpcb_users++;
- cnt = rpcb_users;
- spin_unlock(&rpcb_clnt_lock);
+ spin_lock(&sn->rpcb_clnt_lock);
+ if (sn->rpcb_users)
+   sn->rpcb_users++;
+ cnt = sn->rpcb_users;
+ spin_unlock(&sn->rpcb_clnt_lock);

    return cnt;
}

void rpcb_put_local(void)
{
- struct rpc_clnt *clnt = rpcb_local_clnt;
- struct rpc_clnt *clnt4 = rpcb_local_clnt4;
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);
+ struct rpc_clnt *clnt = sn->rpcb_local_clnt;
+ struct rpc_clnt *clnt4 = sn->rpcb_local_clnt4;
    int shutdown;

- spin_lock(&rpcb_clnt_lock);
- if (--rpcb_users == 0) {
-   rpcb_local_clnt = NULL;
-   rpcb_local_clnt4 = NULL;
+ spin_lock(&sn->rpcb_clnt_lock);
+ if (--sn->rpcb_users == 0) {
+   sn->rpcb_local_clnt = NULL;
+   sn->rpcb_local_clnt4 = NULL;
    }
- shutdown = !rpcb_users;
- spin_unlock(&rpcb_clnt_lock);
+ shutdown = !sn->rpcb_users;
+ spin_unlock(&sn->rpcb_clnt_lock);

    if (shutdown) {
        /*
@@ -205,14 +204,16 @@ void rpcb_put_local(void)

static void rpcb_set_local(struct rpc_clnt *clnt, struct rpc_clnt *clnt4)

```

```

{
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);
+
  /* Protected by rpcb_create_local_mutex */
- rpcb_local_clnt = clnt;
- rpcb_local_clnt4 = clnt4;
+ sn->rpcb_local_clnt = clnt;
+ sn->rpcb_local_clnt4 = clnt4;
  smp_wmb();
- rpcb_users = 1;
+ sn->rpcb_users = 1;
  dprintk("RPC:      created new rpcb local clients (rpcb_local_clnt: "
-   "%p, rpcb_local_clnt4: %p)\n", rpcb_local_clnt,
-   rpcb_local_clnt4);
+   "%p, rpcb_local_clnt4: %p)\n", sn->rpcb_local_clnt,
+   sn->rpcb_local_clnt4);
}

/*
@@ -432,6 +433,7 @@ int rpcb_register(u32 prog, u32 vers, int prot, unsigned short port)
  struct rpc_message msg = {
    .rpc_argp = &map,
  };
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

  dprintk("RPC:      %sregistering (%u, %u, %d, %u) with local "
    "rpcbind\n", (port ? "" : "un"),
@@ -441,7 +443,7 @@ int rpcb_register(u32 prog, u32 vers, int prot, unsigned short port)
  if (port)
    msg.rpc_proc = &rpcb_procedures2[RPCBPROC_SET];

- return rpcb_register_call(rpcb_local_clnt, &msg);
+ return rpcb_register_call(sn->rpcb_local_clnt, &msg);
}

/*
@@ -454,6 +456,7 @@ static int rpcb_register_inet4(const struct sockaddr *sap,
  struct rpcbind_args *map = msg->rpc_argp;
  unsigned short port = ntohs(sin->sin_port);
  int result;
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

  map->r_addr = rpc_sockaddr2uaddr(sap);

@@ -466,7 +469,7 @@ static int rpcb_register_inet4(const struct sockaddr *sap,
  if (port)
    msg->rpc_proc = &rpcb_procedures4[RPCBPROC_SET];

```

```

- result = rpcb_register_call(rpcb_local_clnt4, msg);
+ result = rpcb_register_call(sn->rpcb_local_clnt4, msg);
  kfree(map->r_addr);
  return result;
}
@@ -481,6 +484,7 @@ static int rpcb_register_inet6(const struct sockaddr *sap,
  struct rpcbind_args *map = msg->rpc_argp;
  unsigned short port = ntohs(sin6->sin6_port);
  int result;
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

  map->r_addr = rpc_sockaddr2uaddr(sap);

@@ -493,7 +497,7 @@ static int rpcb_register_inet6(const struct sockaddr *sap,
  if (port)
    msg->rpc_proc = &rpcb_procedures4[RPCBPROC_SET];

- result = rpcb_register_call(rpcb_local_clnt4, msg);
+ result = rpcb_register_call(sn->rpcb_local_clnt4, msg);
  kfree(map->r_addr);
  return result;
}
@@ -501,6 +505,7 @@ static int rpcb_register_inet6(const struct sockaddr *sap,
static int rpcb_unregister_all_protocols(struct rpc_message *msg)
{
  struct rpcbind_args *map = msg->rpc_argp;
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

  dprintk("RPC:   unregistering [%u, %u, '%s'] with "
    "local rpcbind\n",
@@ -509,7 +514,7 @@ static int rpcb_unregister_all_protocols(struct rpc_message *msg)
  map->r_addr = "";
  msg->rpc_proc = &rpcb_procedures4[RPCBPROC_UNSET];

- return rpcb_register_call(rpcb_local_clnt4, msg);
+ return rpcb_register_call(sn->rpcb_local_clnt4, msg);
}

/**
@@ -567,8 +572,9 @@ int rpcb_v4_register(const u32 program, const u32 version,
  struct rpc_message msg = {
    .rpc_argp = &map,
  };
+ struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);

- if (rpcb_local_clnt4 == NULL)
+ if (sn->rpcb_local_clnt4 == NULL)
  return -EPROTONOSUPPORT;

```

if (address == NULL)

---