Subject: [PATCH 0/4] SUNRPC: rcbind clients virtualization
Posted by Stanislav Kinsbursky on Tue, 04 Oct 2011 16:59:58 GMT
View Forum Message <> Reply to Message

This patch-set virtualizes rpcbind clients per network namespace context. IOW,
each network namespace will have its own pair of rpcbind clients (if the would
be created by request).

Note:
1) this patch-set depends on "SUNRPC: make rpcbind clients allocated and
destroyed dynamically" patch-set which has been send earlier.
2) init_net pointer is still used instead of current->nsproxy->net_ns,
because I'm not sure yet about how to virtualize services.
I.e. NFS callback services will be per netns. NFSd service will be per netns
too from my pow. But Lockd can be per netns or one for all.
And also we have NFSd file system, which is not virtualized yet.

The following series consists of:

Stanislav Kinsbursky (4):
    SUNRPC: rpcbind clients internals virtualization
    SUNRPC: use virtualized rpcbind internals instead of static ones
    SUNRPC: optimize net_ns dereferencing in rpcbind creation calls
    SUNRPC: optimize net_ns dereferencing in rpcbind registering calls

 net/sunrpc/netns.h | 5 ++
 net/sunrpc/rpcb_clnt.c | 103 ++++++++++++++++++++++++++++++++----------------
 2 files changed, 61 insertions(+), 47 deletions(-)

--
Signature

Subject: [PATCH 2/4] SUNRPC: use virtualized rpcbind internals instead of static ones
Posted by Stanislav Kinsbursky on Tue, 04 Oct 2011 17:01:28 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>

```
 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_protofamilies(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_protofamilies(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)
```

Subject: [PATCH 3/4] SUNRPC: optimize net_ns dereferencing in rpcbind creation calls

Posted by Stanislav Kinsbursky on Tue, 04 Oct 2011 17:01:29 GMT

View Forum Message <> Reply to Message

Static rpcbind creation functions can be parametrized by network namespace pointer, calculated only once, instead of using init_net pointer (or taking it from current when virtualization will be comleted) in many places.

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

---
 net/sunrpc/rpcb_clnt.c | 35 ++++++++++++++++++++++----------
 1 files changed, 19 insertions(+), 16 deletions(-)

diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index 18eba8e..ef37c55 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
@@ -161,10 +161,10 @@ static void rpcb_map_release(void *data)
  kfree(map);
 }

-static int rpcb_get_local(void)
+static int rpcb_get_local(struct net *net)
 {
  int cnt;
- struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

  spin_lock(&sn->rpcb_clnt_lock);
  if (sn->rpcb_users)
@@ -202,9 +202,10 @@ void rpcb_put_local(void)
  return;
 }

-static void rpcb_set_local(struct rpc_clnt *clnt, struct rpc_clnt *clnt4)
+static void rpcb_set_local(struct net *net, struct rpc_clnt *clnt,
+   struct rpc_clnt *clnt4)
 {
- struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

  /* Protected by rpcb_create_local_mutex */
  sn->rpcb_local_clnt = clnt;
@@ -212,22 +213,23 @@ static void rpcb_set_local(struct rpc_clnt *clnt, struct rpc_clnt *clnt4)
  smp_wmb();
  sn->rpcb_users = 1;
  dprintk("RPC: created new rpcb local clients (rpcb_local_clnt: "
-   "%p, rpcb_local_clnt4: %p)\n", sn->rpcb_local_clnt,
-   sn->rpcb_local_clnt4);
+   "%p, rpcb_local_clnt4: %p) for net %p%s\n",
+   sn->rpcb_local_clnt, sn->rpcb_local_clnt4,
+   net, (net == &init_net) ? " (init_net)" : "");
 }

```
 /*
  * Returns zero on success, otherwise a negative errno value
  * is returned.
  */
-static int rpcb_create_local_unix(void)
+static int rpcb_create_local_unix(struct net *net)
 {
  static const struct sockaddr_un rpcb_localaddr_rpcbind = {
   .sun_family  = AF_LOCAL,
   .sun_path  = RPCBIND_SOCK_PATHNAME,
  };
  struct rpc_create_args args = {
-  .net  = &init_net,
+  .net  = net,
   .protocol = XPRT_TRANSPORT_LOCAL,
   .address = (struct sockaddr *)&rpcb_localaddr_rpcbind,
   .addrsize = sizeof(rpcb_localaddr_rpcbind),
@@ -260,7 +262,7 @@ static int rpcb_create_local_unix(void)
   clnt4 = NULL;
  }

- rpcb_set_local(clnt, clnt4);
+ rpcb_set_local(net, clnt, clnt4);

 out:
  return result;
@@ -270,7 +272,7 @@ out:
  * Returns zero on success, otherwise a negative errno value
  * is returned.
  */
-static int rpcb_create_local_net(void)
+static int rpcb_create_local_net(struct net *net)
 {
  static const struct sockaddr_in rpcb_inaddr_loopback = {
   .sin_family  = AF_INET,
@@ -278,7 +280,7 @@ static int rpcb_create_local_net(void)
   .sin_port  = htons(RPCBIND_PORT),
  };
  struct rpc_create_args args = {
-  .net  = &init_net,
+  .net  = net,
   .protocol = XPRT_TRANSPORT_TCP,
   .address = (struct sockaddr *)&rpcb_inaddr_loopback,
   .addrsize = sizeof(rpcb_inaddr_loopback),
@@ -312,7 +314,7 @@ static int rpcb_create_local_net(void)
   clnt4 = NULL;
  }
```

```
- rpcb_set_local(clnt, clnt4);
+ rpcb_set_local(net, clnt, clnt4);

 out:
  return result;
@@ -326,16 +328,17 @@ int rpcb_create_local(void)
 {
  static DEFINE_MUTEX(rpcb_create_local_mutex);
  int result = 0;
+ struct net *net = &init_net;

- if (rpcb_get_local())
+ if (rpcb_get_local(net))
   return result;

  mutex_lock(&rpcb_create_local_mutex);
- if (rpcb_get_local())
+ if (rpcb_get_local(net))
   goto out;

- if (rpcb_create_local_unix() != 0)
-  result = rpcb_create_local_net();
+ if (rpcb_create_local_unix(net) != 0)
+  result = rpcb_create_local_net(net);

 out:
  mutex_unlock(&rpcb_create_local_mutex);
```

Subject: [PATCH 4/4] SUNRPC: optimize net_ns dereferencing in rpcbind registering calls

Posted by Stanislav Kinsbursky on Tue, 04 Oct 2011 17:01:31 GMT

View Forum Message <> Reply to Message

Static rpcbind registering functions can be parametrized by network namespace pointer, calculated only once, instead of using init_net pointer (or taking it from current when virtualization will be comleted) in many places.

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

```
---
 net/sunrpc/rpcb_clnt.c |   18 +++++++++---------
 1 files changed, 9 insertions(+), 9 deletions(-)

diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index ef37c55..92ed74f 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
```

```
@@ -452,14 +452,14 @@ int rpcb_register(u32 prog, u32 vers, int prot, unsigned short port)
 /*
  * Fill in AF_INET family-specific arguments to register
  */
-static int rpcb_register_inet4(const struct sockaddr *sap,
+static int rpcb_register_inet4(struct sunrpc_net *sn,
+          const struct sockaddr *sap,
           struct rpc_message *msg)
 {
  const struct sockaddr_in *sin = (const struct sockaddr_in *)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);

@@ -480,14 +480,14 @@ static int rpcb_register_inet4(const struct sockaddr *sap,
 /*
  * Fill in AF_INET6 family-specific arguments to register
  */
-static int rpcb_register_inet6(const struct sockaddr *sap,
+static int rpcb_register_inet6(struct sunrpc_net *sn,
+          const struct sockaddr *sap,
           struct rpc_message *msg)
 {
  const struct sockaddr_in6 *sin6 = (const struct sockaddr_in6 *)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);

@@ -505,10 +505,10 @@ static int rpcb_register_inet6(const struct sockaddr *sap,
  return result;
 }

-static int rpcb_unregister_all_protofamilies(struct rpc_message *msg)
+static int rpcb_unregister_all_protofamilies(struct sunrpc_net *sn,
+          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",
@@ -581,13 +581,13 @@ int rpcb_v4_register(const u32 program, const u32 version,
```

```
  return -EPROTONOSUPPORT;

 if (address == NULL)
-  return rpcb_unregister_all_protofamilies(&msg);
+  return rpcb_unregister_all_protofamilies(sn, &msg);

 switch (address->sa_family) {
 case AF_INET:
-  return rpcb_register_inet4(address, &msg);
+  return rpcb_register_inet4(sn, address, &msg);
 case AF_INET6:
-  return rpcb_register_inet6(address, &msg);
+  return rpcb_register_inet6(sn, address, &msg);
 }

 return -EAFNOSUPPORT;
```

Subject: [PATCH 1/4] SUNRPC: rpcbind clients internals virtualization
Posted by Stanislav Kinsbursky on Tue, 04 Oct 2011 17:01:31 GMT
View Forum Message <> Reply to Message

This patch moves static rpcbind internals to sunrpc part of network namespace
context. This will allow to create rcpbind clients per network namespace.

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

```
---
 net/sunrpc/netns.h |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index d013bf2..83eede3 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -9,6 +9,11 @@ struct cache_detail;
 struct sunrpc_net {
  struct proc_dir_entry *proc_net_rpc;
  struct cache_detail *ip_map_cache;
+
+ struct rpc_clnt *rpcb_local_clnt;
+ struct rpc_clnt *rpcb_local_clnt4;
+ spinlock_t rpcb_clnt_lock;
+ unsigned int rpcb_users;
 };

 extern int sunrpc_net_id;
```
