Subject: [PATCH v5 0/6] SUNRPC: rcbind clients virtualization
Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 08:52:01 GMT
View Forum Message <> Reply to Message

This patch-set was created in context of clone of git branch:
git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
head: devel

v5:
1) Rebased on up-to-date state ("devel" head).
2) Added two more patches from latter series ("SUNRPC: register services with per-net rpcbind"), because logically this series is the more suitable place for them.

v4:
1) Rebased of current repo state (i.e. all commits pulled before apply)

v3:
1) First two patches from previous version were squashed.

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

The following series consists of:

Stanislav Kinsbursky (6):
SUNRPC: move rpcbind internals to sunrpc part of network namespace context
SUNRPC: optimize net_ns dereferencing in rpcbind creation calls
SUNRPC: optimize net_ns dereferencing in rpcbind registering calls
SUNRPC: create rpcbind client in passed network namespace context
SUNRPC: register rpcbind programs in passed network namespase context
SUNRPC: parametrize local rpcbind clients creation with net ns

include/linux/sunrpc/clnt.h | 9 ++-
net/sunrpc/netns.h | 5 ++
net/sunrpc/rpcb_clnt.c | 121 ++++++++++++++++++++++++++----------
net/sunrpc/svc.c | 14 ++---
4 files changed, 82 insertions(+), 67 deletions(-)

Subject: [PATCH v5 1/6] SUNRPC: move rpcbind internals to sunrpc part of network namespace context
Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 08:52:10 GMT

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) required for registering svc services per network namespace.

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

```
---
 net/sunrpc/netns.h     |    5 ++++
 net/sunrpc/rpcb_clnt.c |   64 ++++++++++++++++++++++++++----------------------
 2 files changed, 40 insertions(+), 29 deletions(-)

diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index 0f3af34..1fdeb1b 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -15,6 +15,11 @@ struct sunrpc_net {

  struct list_head all_clients;
  spinlock_t rpc_client_lock;
+
+ 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;
diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index 8761bf8..7d32f19 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) {
    /*
@@ -204,14 +203,16 @@ 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);
 }

 /*
@@ -431,6 +432,7 @@ 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"),
@@ -440,7 +442,7 @@ 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);
 }
```

```
 /*
@@ -453,6 +455,7 @@ 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, GFP_KERNEL);

@@ -465,7 +468,7 @@ 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;
 }
@@ -480,6 +483,7 @@ 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, GFP_KERNEL);

@@ -492,7 +496,7 @@ 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;
 }
@@ -500,6 +504,7 @@ 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",
@@ -508,7 +513,7 @@ 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);
 }

 /**
@@ -566,8 +571,9 @@ 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 v5 3/6] SUNRPC: optimize net_ns dereferencing in rpcbind registering calls

Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 08:52:26 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 3df276a..371d229 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
@@ -451,14 +451,14 @@ 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, GFP_KERNEL);

@@ -479,14 +479,14 @@ 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, GFP_KERNEL);

@@ -504,10 +504,10 @@ 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",
@@ -580,13 +580,13 @@ 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 v5 5/6] SUNRPC: register rpcbind programs in passed network namespase context
Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 08:52:43 GMT
View Forum Message <> Reply to Message

Registering rpcbind program requires rpcbind clients, which are per network
namespace context.

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

```
---
 include/linux/sunrpc/clnt.h |    5 +++--
 net/sunrpc/rpcb_clnt.c      |    8 ++++++----
 net/sunrpc/svc.c            |   10 +++++++---
 3 files changed, 12 insertions(+), 11 deletions(-)

diff --git a/include/linux/sunrpc/clnt.h b/include/linux/sunrpc/clnt.h
index bfd6185..b0b3e57 100644
--- a/include/linux/sunrpc/clnt.h
+++ b/include/linux/sunrpc/clnt.h
@@ -138,8 +138,9 @@ void  rpc_task_release_client(struct rpc_task *);

 int  rpcb_create_local(void);
 void  rpcb_put_local(void);
-int  rpcb_register(u32, u32, int, unsigned short);
-int  rpcb_v4_register(const u32 program, const u32 version,
+int  rpcb_register(struct net *, u32, u32, int, unsigned short);
+int  rpcb_v4_register(struct net *net, const u32 program,
+     const u32 version,
      const struct sockaddr *address,
      const char *netid);
 void  rpcb_getport_async(struct rpc_task *);
diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index 0d76c0f..a5aa50d 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
@@ -425,7 +425,7 @@ static int rpcb_register_call(struct rpc_clnt *clnt, struct rpc_message
*msg)
  * IN6ADDR_ANY (ie available for all AF_INET and AF_INET6
  * addresses).
  */
-int rpcb_register(u32 prog, u32 vers, int prot, unsigned short port)
```

```
+int rpcb_register(struct net *net, u32 prog, u32 vers, int prot, unsigned short port)
 {
  struct rpcbind_args map = {
   .r_prog  = prog,
@@ -436,7 +436,7 @@ 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);
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

  dprintk("RPC:       %sregistering (%u, %u, %d, %u) with local "
    "rpcbind\n", (port ? "" : "un"),
@@ -563,7 +563,7 @@ static int rpcb_unregister_all_protofamilies(struct sunrpc_net *sn,
  * service on any IPv4 address, but not on IPv6.  The latter
  * advertises the service on all IPv4 and IPv6 addresses.
  */
-int rpcb_v4_register(const u32 program, const u32 version,
+int rpcb_v4_register(struct net *net, const u32 program, const u32 version,
        const struct sockaddr *address, const char *netid)
 {
  struct rpcbind_args map = {
@@ -575,7 +575,7 @@ 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);
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);

  if (sn->rpcb_local_clnt4 == NULL)
   return -EPROTONOSUPPORT;
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index 9d01d46..aa0041a 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -813,7 +813,7 @@ static int __svc_rpcb_register4(const u32 program, const u32 version,
   return -ENOPROTOOPT;
  }

- error = rpcb_v4_register(program, version,
+ error = rpcb_v4_register(&init_net, program, version,
      (const struct sockaddr *)&sin, netid);

  /*
@@ -821,7 +821,7 @@ static int __svc_rpcb_register4(const u32 program, const u32 version,
   * registration request with the legacy rpcbind v2 protocol.
   */
  if (error == -EPROTONOSUPPORT)
```

```
-  error = rpcb_register(program, version, protocol, port);
+  error = rpcb_register(&init_net, program, version, protocol, port);

  return error;
 }
@@ -860,7 +860,7 @@ static int __svc_rpcb_register6(const u32 program, const u32 version,
   return -ENOPROTOOPT;
  }

- error = rpcb_v4_register(program, version,
+ error = rpcb_v4_register(&init_net, program, version,
      (const struct sockaddr *)&sin6, netid);

  /*
@@ -963,14 +963,14 @@ static void __svc_unregister(const u32 program, const u32 version,
 {
  int error;

- error = rpcb_v4_register(program, version, NULL, "");
+ error = rpcb_v4_register(&init_net, program, version, NULL, "");

  /*
   * User space didn't support rpcbind v4, so retry this
   * request with the legacy rpcbind v2 protocol.
   */
  if (error == -EPROTONOSUPPORT)
-  error = rpcb_register(program, version, 0, 0);
+  error = rpcb_register(&init_net, program, version, 0, 0);

  dprintk("svc: %s(%sv%u), error %d\n",
    __func__, progname, version, error);
```

Subject: [PATCH v5 6/6] SUNRPC: parametrize local rpcbind clients creation with net ns

Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 08:52:51 GMT

View Forum Message <> Reply to Message

These client are per network namespace and thus can be created for different
network namespaces.

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

```
---
 include/linux/sunrpc/clnt.h |    4 ++--
 net/sunrpc/rpcb_clnt.c      |    7 +++----
 net/sunrpc/svc.c            |    4 ++--
 3 files changed, 7 insertions(+), 8 deletions(-)
```

```
diff --git a/include/linux/sunrpc/clnt.h b/include/linux/sunrpc/clnt.h
index b0b3e57..e891a8a 100644
--- a/include/linux/sunrpc/clnt.h
+++ b/include/linux/sunrpc/clnt.h
@@ -136,8 +136,8 @@ void  rpc_shutdown_client(struct rpc_clnt *);
 void  rpc_release_client(struct rpc_clnt *);
 void  rpc_task_release_client(struct rpc_task *);

-int  rpcb_create_local(void);
-void  rpcb_put_local(void);
+int  rpcb_create_local(struct net *);
+void  rpcb_put_local(struct net *);
 int  rpcb_register(struct net *, u32, u32, int, unsigned short);
 int  rpcb_v4_register(struct net *net, const u32 program,
      const u32 version,
diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index a5aa50d..6d6a84f 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
@@ -175,9 +175,9 @@ static int rpcb_get_local(struct net *net)
  return cnt;
 }

-void rpcb_put_local(void)
+void rpcb_put_local(struct net *net)
 {
- struct sunrpc_net *sn = net_generic(&init_net, sunrpc_net_id);
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
  struct rpc_clnt *clnt = sn->rpcb_local_clnt;
  struct rpc_clnt *clnt4 = sn->rpcb_local_clnt4;
  int shutdown;
@@ -323,11 +323,10 @@ out:
  * Returns zero on success, otherwise a negative errno value
  * is returned.
  */
-int rpcb_create_local(void)
+int rpcb_create_local(struct net *net)
 {
  static DEFINE_MUTEX(rpcb_create_local_mutex);
  int result = 0;
- struct net *net = &init_net;

  if (rpcb_get_local(net))
   return result;
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index aa0041a..8f6c255 100644
--- a/net/sunrpc/svc.c
```

```
+++ b/net/sunrpc/svc.c
@@ -370,7 +370,7 @@ static int svc_rpcb_setup(struct svc_serv *serv)
 {
  int err;

- err = rpcb_create_local();
+ err = rpcb_create_local(&init_net);
  if (err)
   return err;

@@ -382,7 +382,7 @@ static int svc_rpcb_setup(struct svc_serv *serv)
 void svc_rpcb_cleanup(struct svc_serv *serv)
 {
  svc_unregister(serv);
- rpcb_put_local();
+ rpcb_put_local(&init_net);
 }
 EXPORT_SYMBOL_GPL(svc_rpcb_cleanup);
```

Subject: Re: [PATCH v5 0/6] SUNRPC: rcbind clients virtualization
Posted by Stanislav Kinsbursky on Fri, 13 Jan 2012 10:03:39 GMT
View Forum Message <> Reply to Message

> This patch-set was created in context of clone of git branch:
> git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git.
> head: devel
>
> v5:
> 1) Rebased on up-to-date state ("devel" head).
> 2) Added two more patches from latter series ("SUNRPC: register services with
> per-net rpcbind"), because logically this series is the more suitable place for
> them.

Three patches from "SUNRPC: register services with per-net rpcbind" series:
a) [1/7] "SUNRPC: create rpcbind client in passed network namespace context"
b) [2/7] "SUNRPC: register rpcbind programs in passed network namespase context"
c) [4/7] "SUNRPC: parametrize local rpcbind clients creation with net ns"

--
Best regards,
Stanislav Kinsbursky
