Subject: [PATCH 0/4] SUNRPC: service release in network namespace context
Posted by Stanislav Kinsbursky on Wed, 25 Jan 2012 13:47:00 GMT
View Forum Message <> Reply to Message

This patch set is the final step towards to making LockD network namespace
aware.
I can't prove, that this patch set is enough for NFSd (just haven't try), by
Lockd works fine and patches for it will be send soon.

The following series consists of:

Stanislav Kinsbursky (4):
    SUNRPC: clear svc pools lists helper introduced
    SUNRPC: clear svc transports lists helper introduced
    SUNRPC: service destruction in network namespace context
    SUNRPC: service shutdown function in network namespace context introduced

 include/linux/sunrpc/svcsock.h | 2 +-
 net/sunrpc/svc.c | 36 ++++++++++++++++++-----------
 net/sunrpc/svc_xprt.c | 45 ++-----------
 3 files changed, 56 insertions(+), 27 deletions(-)

Subject: [PATCH 1/4] SUNRPC: clear svc pools lists helper introduced
Posted by Stanislav Kinsbursky on Wed, 25 Jan 2012 13:47:09 GMT
View Forum Message <> Reply to Message

This patch moves removing of service transport from it's pools ready lists to
separated function. Also this clear is now done with list_for_each_entry_safe()
helper.
This is a precursor patch, which would be usefull with service shutdown in
network namespace context, introduced later in the series.

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

---
 net/sunrpc/svc_xprt.c | 19 +++++++++++++------
 1 files changed, 13 insertions(+), 6 deletions(-)

diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
index de1e1a8..50bf7c1 100644
--- a/net/sunrpc/svc_xprt.c
+++ b/net/sunrpc/svc_xprt.c
@@ -932,26 +932,33 @@ static void svc_close_list(struct list_head *xprt_list)
  }

```
 }

-void svc_close_all(struct svc_serv *serv)
+static void svc_clear_pools(struct svc_serv *serv)
 {
  struct svc_pool *pool;
  struct svc_xprt *xprt;
  struct svc_xprt *tmp;
  int i;

- svc_close_list(&serv->sv_tempsocks);
- svc_close_list(&serv->sv_permsocks);
-
  for (i = 0; i < serv->sv_nrpools; i++) {
   pool = &serv->sv_pools[i];

   spin_lock_bh(&pool->sp_lock);
-  while (!list_empty(&pool->sp_sockets)) {
-   xprt = list_first_entry(&pool->sp_sockets, struct svc_xprt, xpt_ready);
+  list_for_each_entry_safe(xprt, tmp, &pool->sp_sockets, xpt_ready) {
    list_del_init(&xprt->xpt_ready);
   }
   spin_unlock_bh(&pool->sp_lock);
  }
+}
+
+void svc_close_all(struct svc_serv *serv)
+{
+ struct svc_xprt *xprt;
+ struct svc_xprt *tmp;
+
+ svc_close_list(&serv->sv_tempsocks);
+ svc_close_list(&serv->sv_permsocks);
+
+ svc_clear_pools(serv);
  /*
   * At this point the sp_sockets lists will stay empty, since
   * svc_enqueue will not add new entries without taking the
```

Subject: [PATCH 2/4] SUNRPC: clear svc transports lists helper introduced
Posted by Stanislav Kinsbursky on Wed, 25 Jan 2012 13:47:17 GMT
View Forum Message <> Reply to Message

This patch moves service transports deletion from service sockets lists to
separated function.
This is a precursor patch, which would be usefull with service shutdown in
network namespace context, introduced later in the series.

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

---
 net/sunrpc/svc_xprt.c | 19 ++++++++++++--------
 1 files changed, 11 insertions(+), 8 deletions(-)

diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
index 50bf7c1..493e70b 100644
--- a/net/sunrpc/svc_xprt.c
+++ b/net/sunrpc/svc_xprt.c
@@ -950,11 +950,19 @@ static void svc_clear_pools(struct svc_serv *serv)
  }
 }

-void svc_close_all(struct svc_serv *serv)
+static void svc_clear_list(struct list_head *xprt_list)
 {
  struct svc_xprt *xprt;
  struct svc_xprt *tmp;

+ list_for_each_entry_safe(xprt, tmp, xprt_list, xpt_list) {
+ svc_delete_xprt(xprt);
+ }
+ BUG_ON(!list_empty(xprt_list));
+}
+
+void svc_close_all(struct svc_serv *serv)
+{
  svc_close_list(&serv->sv_tempsocks);
  svc_close_list(&serv->sv_permsocks);

@@ -964,13 +972,8 @@ void svc_close_all(struct svc_serv *serv)
   * svc_enqueue will not add new entries without taking the
   * sp_lock and checking XPT_BUSY.
   */
- list_for_each_entry_safe(xprt, tmp, &serv->sv_tempsocks, xpt_list)
- svc_delete_xprt(xprt);
- list_for_each_entry_safe(xprt, tmp, &serv->sv_permsocks, xpt_list)
- svc_delete_xprt(xprt);
-
- BUG_ON(!list_empty(&serv->sv_permsocks));
- BUG_ON(!list_empty(&serv->sv_tempsocks));
+ svc_clear_list(&serv->sv_tempsocks);
+ svc_clear_list(&serv->sv_permsocks);
 }

 /*

Subject: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by Stanislav Kinsbursky on Wed, 25 Jan 2012 13:47:26 GMT
View Forum Message <> Reply to Message

This patch introduces network namespace filter for service destruction
function.
Nothing special here - just do exactly the same operations, but only for
tranports in passed networks namespace context.
BTW, BUG_ON() checks for empty service transports lists were returned into
svc_destroy() function. This is because of swithing generic svc_close_all() to
networks namespace dependable svc_close_net().

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

---
 include/linux/sunrpc/svcsock.h | 2 +-
 net/sunrpc/svc.c | 9 +++++++--
 net/sunrpc/svc_xprt.c | 27 +++++++++++++++++++----------
 3 files changed, 25 insertions(+), 13 deletions(-)

diff --git a/include/linux/sunrpc/svcsock.h b/include/linux/sunrpc/svcsock.h
index c84e974..cb4ac69 100644
--- a/include/linux/sunrpc/svcsock.h
+++ b/include/linux/sunrpc/svcsock.h
@@ -34,7 +34,7 @@ struct svc_sock {
 /*
  * Function prototypes.
  */
-void svc_close_all(struct svc_serv *);
+void svc_close_net(struct svc_serv *, struct net *);
 int svc_recv(struct svc_rqst *, long);
 int svc_send(struct svc_rqst *);
 void svc_drop(struct svc_rqst *);
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index a8b49a0..279bbd8 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -517,6 +517,8 @@ EXPORT_SYMBOL_GPL(svc_create_pooled);
 void
 svc_destroy(struct svc_serv *serv)
 {
+ struct net *net = current->nsproxy->net_ns;
+
  dprintk("svc: svc_destroy(%s, %d)\n",
     serv->sv_program->pg_name,
     serv->sv_nrthreads);
@@ -539,10 +541,13 @@ svc_destroy(struct svc_serv *serv)
   * caller is using--nfsd_mutex in the case of nfsd). So it's
   * safe to traverse those lists and shut everything down:

```
   */
- svc_close_all(serv);
+ svc_close_net(serv, net);
+
+ BUG_ON(!list_empty(&serv->sv_permsocks));
+ BUG_ON(!list_empty(&serv->sv_tempsocks));

  if (serv->sv_shutdown)
-  serv->sv_shutdown(serv, current->nsproxy->net_ns);
+  serv->sv_shutdown(serv, net);

  cache_clean_deferred(serv);

diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
index 493e70b..4bda09d 100644
--- a/net/sunrpc/svc_xprt.c
+++ b/net/sunrpc/svc_xprt.c
@@ -922,17 +922,19 @@ void svc_close_xprt(struct svc_xprt *xprt)
 }
 EXPORT_SYMBOL_GPL(svc_close_xprt);

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

  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);
  }
 }

-static void svc_clear_pools(struct svc_serv *serv)
+static void svc_clear_pools(struct svc_serv *serv, struct net *net)
 {
  struct svc_pool *pool;
  struct svc_xprt *xprt;
@@ -944,36 +946,41 @@ static void svc_clear_pools(struct svc_serv *serv)

   spin_lock_bh(&pool->sp_lock);
   list_for_each_entry_safe(xprt, tmp, &pool->sp_sockets, xpt_ready) {
+   if (xprt->xpt_net != net)
+    continue;
    list_del_init(&xprt->xpt_ready);
   }
   spin_unlock_bh(&pool->sp_lock);
```

```
  }
 }

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

  list_for_each_entry_safe(xprt, tmp, xprt_list, xpt_list) {
+  if (xprt->xpt_net != net)
+   continue;
   svc_delete_xprt(xprt);
  }
- BUG_ON(!list_empty(xprt_list));
+ list_for_each_entry(xprt, xprt_list, xpt_list)
+  BUG_ON(xprt->xpt_net == net);
 }

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

- svc_clear_pools(serv);
+ svc_clear_pools(serv, net);
  /*
   * At this point the sp_sockets lists will stay empty, since
   * svc_enqueue will not add new entries without taking the
   * sp_lock and checking XPT_BUSY.
   */
- svc_clear_list(&serv->sv_tempsocks);
- svc_clear_list(&serv->sv_permsocks);
+ svc_clear_list(&serv->sv_tempsocks, net);
+ svc_clear_list(&serv->sv_permsocks, net);
 }

 /*
```

Subject: [PATCH 4/4] SUNRPC: service shutdown function in network namespace context introduced
Posted by Stanislav Kinsbursky on Wed, 25 Jan 2012 13:47:34 GMT
View Forum Message <> Reply to Message

This function is enough for releasing resources, allocated for network
namespace context, in case of sharing service between them.
IOW, each service "user" (LockD, NFSd, etc), which wants to share service
between network namespaces, have to release related resources by the function,
introduced in this patch, instead of performing service shutdown (of course in
case the service is shared already to the moment of release).

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

---
 net/sunrpc/svc.c | 33 ++++++++++++++++++++++++++----------
 1 files changed, 20 insertions(+), 13 deletions(-)

diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index 279bbd8..3adfd26 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -510,6 +510,24 @@ svc_create_pooled(struct svc_program *prog, unsigned int bufsize,
 }
 EXPORT_SYMBOL_GPL(svc_create_pooled);

+void svc_shutdown_net(struct svc_serv *serv, struct net *net)
+{
+ /*
+ * The set of xprts (contained in the sv_tempsocks and
+ * sv_permsocks lists) is now constant, since it is modified
+ * only by accepting new sockets (done by service threads in
+ * svc_recv) or aging old ones (done by sv_temptimer), or
+ * configuration changes (excluded by whatever locking the
+ * caller is using--nfsd_mutex in the case of nfsd). So it's
+ * safe to traverse those lists and shut everything down:
+ */
+ svc_close_net(serv, net);
+
+ if (serv->sv_shutdown)
+ serv->sv_shutdown(serv, net);
+}
+EXPORT_SYMBOL_GPL(svc_shutdown_net);
+
 /*
  * Destroy an RPC service. Should be called with appropriate locking to
  * protect the sv_nrthreads, sv_permsocks and sv_tempsocks.
@@ -532,23 +550,12 @@ svc_destroy(struct svc_serv *serv)
   printk("svc_destroy: no threads for serv=%p!\n", serv);

  del_timer_sync(&serv->sv_temptimer);
- /*
- * The set of xprts (contained in the sv_tempsocks and

```
-  * sv_permsocks lists) is now constant, since it is modified
-  * only by accepting new sockets (done by service threads in
-  * svc_recv) or aging old ones (done by sv_temptimer), or
-  * configuration changes (excluded by whatever locking the
-  * caller is using--nfsd_mutex in the case of nfsd).  So it's
-  * safe to traverse those lists and shut everything down:
-  */
- svc_close_net(serv, net);
+
+ svc_shutdown_net(serv, net);

   BUG_ON(!list_empty(&serv->sv_permsocks));
   BUG_ON(!list_empty(&serv->sv_tempsocks));

- if (serv->sv_shutdown)
-  serv->sv_shutdown(serv, net);
-
   cache_clean_deferred(serv);

   if (svc_serv_is_pooled(serv))
```

Subject: Re: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by bfields on Thu, 26 Jan 2012 21:14:50 GMT
View Forum Message <> Reply to Message

On Wed, Jan 25, 2012 at 05:47:26PM +0400, Stanislav Kinsbursky wrote:
> This patch introduces network namespace filter for service destruction
> function.
> Nothing special here - just do exactly the same operations, but only for
> tranports in passed networks namespace context.
> BTW, BUG_ON() checks for empty service transports lists were returned into
> svc_destroy() function. This is because of swithing generic svc_close_all() to
> networks namespace dependable svc_close_net().
>
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
>
> ---
> include/linux/sunrpc/svcsock.h | 2 +-
> net/sunrpc/svc.c | 9 +++++++--
> net/sunrpc/svc_xprt.c | 27 +++++++++++++++++++----------
> 3 files changed, 25 insertions(+), 13 deletions(-)
>
> diff --git a/include/linux/sunrpc/svcsock.h b/include/linux/sunrpc/svcsock.h
> index c84e974..cb4ac69 100644
> --- a/include/linux/sunrpc/svcsock.h
> +++ b/include/linux/sunrpc/svcsock.h

```
> @@ -34,7 +34,7 @@ struct svc_sock {
>  /*
>   * Function prototypes.
>   */
> -void  svc_close_all(struct svc_serv *);
> +void  svc_close_net(struct svc_serv *, struct net *);
>  int  svc_recv(struct svc_rqst *, long);
>  int  svc_send(struct svc_rqst *);
>  void  svc_drop(struct svc_rqst *);
> diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
> index a8b49a0..279bbd8 100644
> --- a/net/sunrpc/svc.c
> +++ b/net/sunrpc/svc.c
> @@ -517,6 +517,8 @@ EXPORT_SYMBOL_GPL(svc_create_pooled);
>  void
>  svc_destroy(struct svc_serv *serv)
>  {
> + struct net *net = current->nsproxy->net_ns;
> +
>   dprintk("svc: svc_destroy(%s, %d)\n",
>       serv->sv_program->pg_name,
>       serv->sv_nrthreads);
> @@ -539,10 +541,13 @@ svc_destroy(struct svc_serv *serv)
>    * caller is using--nfsd_mutex in the case of nfsd).  So it's
>    * safe to traverse those lists and shut everything down:
>    */
> - svc_close_all(serv);
> + svc_close_net(serv, net);
> +
> + BUG_ON(!list_empty(&serv->sv_permsocks));
> + BUG_ON(!list_empty(&serv->sv_tempsocks));
```

I'm confused--what guarantees this is true, at this point?

There are two ways I could imagine containerizing svc_serv: either we create a new one for each namespace, or we share a single global one between them.

If the former, then something that takes a "serv" argument shouldn't also need a "net" argument--the serv should already know which namespace it belongs to.

If the latter, then these lists could have sockets from multiple namespaces, and they aren't guaranteed to be empty here.

?

--b.

```
>
>   if (serv->sv_shutdown)
> -  serv->sv_shutdown(serv, current->nsproxy->net_ns);
> +  serv->sv_shutdown(serv, net);
>
>   cache_clean_deferred(serv);
>
> diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
> index 493e70b..4bda09d 100644
> --- a/net/sunrpc/svc_xprt.c
> +++ b/net/sunrpc/svc_xprt.c
> @@ -922,17 +922,19 @@ void svc_close_xprt(struct svc_xprt *xprt)
>  }
>  EXPORT_SYMBOL_GPL(svc_close_xprt);
>
> -static void svc_close_list(struct list_head *xprt_list)
> +static void svc_close_list(struct list_head *xprt_list, struct net *net)
>  {
>   struct svc_xprt *xprt;
>
>   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);
>   }
>  }
>
> -static void svc_clear_pools(struct svc_serv *serv)
> +static void svc_clear_pools(struct svc_serv *serv, struct net *net)
>  {
>   struct svc_pool *pool;
>   struct svc_xprt *xprt;
> @@ -944,36 +946,41 @@ static void svc_clear_pools(struct svc_serv *serv)
>
>    spin_lock_bh(&pool->sp_lock);
>    list_for_each_entry_safe(xprt, tmp, &pool->sp_sockets, xpt_ready) {
> +   if (xprt->xpt_net != net)
> +    continue;
>     list_del_init(&xprt->xpt_ready);
>    }
>    spin_unlock_bh(&pool->sp_lock);
>   }
>  }
>
> -static void svc_clear_list(struct list_head *xprt_list)
> +static void svc_clear_list(struct list_head *xprt_list, struct net *net)
```

```
>  {
>   struct svc_xprt *xprt;
>   struct svc_xprt *tmp;
>
>   list_for_each_entry_safe(xprt, tmp, xprt_list, xpt_list) {
> +  if (xprt->xpt_net != net)
> +   continue;
>    svc_delete_xprt(xprt);
>   }
> - BUG_ON(!list_empty(xprt_list));
> + list_for_each_entry(xprt, xprt_list, xpt_list)
> +  BUG_ON(xprt->xpt_net == net);
>  }
>
> -void svc_close_all(struct svc_serv *serv)
> +void svc_close_net(struct svc_serv *serv, struct net *net)
>  {
> - svc_close_list(&serv->sv_tempsocks);
> - svc_close_list(&serv->sv_permsocks);
> + svc_close_list(&serv->sv_tempsocks, net);
> + svc_close_list(&serv->sv_permsocks, net);
>
> - svc_clear_pools(serv);
> + svc_clear_pools(serv, net);
>   /*
>    * At this point the sp_sockets lists will stay empty, since
>    * svc_enqueue will not add new entries without taking the
>    * sp_lock and checking XPT_BUSY.
>    */
> - svc_clear_list(&serv->sv_tempsocks);
> - svc_clear_list(&serv->sv_permsocks);
> + svc_clear_list(&serv->sv_tempsocks, net);
> + svc_clear_list(&serv->sv_permsocks, net);
>  }
>
>  /*
>
```

Subject: Re: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by Stanislav Kinsbursky on Fri, 27 Jan 2012 09:08:06 GMT
View Forum Message <> Reply to Message

> On Wed, Jan 25, 2012 at 05:47:26PM +0400, Stanislav Kinsbursky wrote:
>> This patch introduces network namespace filter for service destruction
>> function.

>> Nothing special here - just do exactly the same operations, but only for
>> tranports in passed networks namespace context.
>> BTW, BUG_ON() checks for empty service transports lists were returned into
>> svc_destroy() function. This is because of swithing generic svc_close_all() to
>> networks namespace dependable svc_close_net().
>>
>> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
>>
>> ---
>> include/linux/sunrpc/svcsock.h | 2 +-
>> net/sunrpc/svc.c | 9 +++++++--
>> net/sunrpc/svc_xprt.c | 27 +++++++++++++++++++----------
>> 3 files changed, 25 insertions(+), 13 deletions(-)
>>
>> diff --git a/include/linux/sunrpc/svcsock.h b/include/linux/sunrpc/svcsock.h
>> index c84e974..cb4ac69 100644
>> --- a/include/linux/sunrpc/svcsock.h
>> +++ b/include/linux/sunrpc/svcsock.h
>> @@ -34,7 +34,7 @@ struct svc_sock {
>> /*
>>   * Function prototypes.
>>   */
>> -void svc_close_all(struct svc_serv *);
>> +void svc_close_net(struct svc_serv *, struct net *);
>> int svc_recv(struct svc_rqst *, long);
>> int svc_send(struct svc_rqst *);
>> void svc_drop(struct svc_rqst *);
>> diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
>> index a8b49a0..279bbd8 100644
>> --- a/net/sunrpc/svc.c
>> +++ b/net/sunrpc/svc.c
>> @@ -517,6 +517,8 @@ EXPORT_SYMBOL_GPL(svc_create_pooled);
>> void
>> svc_destroy(struct svc_serv *serv)
>> {
>> + struct net *net = current->nsproxy->net_ns;
>> +
>>   dprintk("svc: svc_destroy(%s, %d)\n",
>> serv->sv_program->pg_name,
>> serv->sv_nrthreads);
>> @@ -539,10 +541,13 @@ svc_destroy(struct svc_serv *serv)
>>   * caller is using--nfsd_mutex in the case of nfsd). So it's
>>   * safe to traverse those lists and shut everything down:
>>   */
>> - svc_close_all(serv);
>> + svc_close_net(serv, net);
>> +
>> + BUG_ON(!list_empty(&serv->sv_permsocks));

>> + BUG_ON(!list_empty(&serv->sv_tempsocks));
>
> I'm confused--what guarantees this is true, at this point?
>

Hi, Bruce.
I'm confused with your question. IOW, this must be true, because this code is executed only in case of last service thread is exiting, doesn't it?

> There are two ways I could imagine containerizing svc_serv: either we
> create a new one for each namespace, or we share a single global one
> between them.
>

This is done for the second one.

> If the former, then something that takes a "serv" argument shouldn't
> also need a "net" argument--the serv should already know which namespace
> it belongs to.
>
> If the latter, then these lists could have sockets from multiple
> namespaces, and they aren't guaranteed to be empty here.
>
> ?
>

I'll explain it on Lockd example (this code is done already - I just haven't sent it yet).
Lockd is still only one thread and can handle lock requests from different network namespaces:
1) Introduced per-net lockd users counter and resources.
2) nlmsvc_users counter become global one. I.e. it's equal to sum of all per-net lockd users counters.
3) For each lockd_up() call global and current net lockd users counters are increased by one.
3) On lockd_up() call: if nlmsvc_users if equal to 0, then lockd thread is started.
4) On lockd_up() call: if current network context lockd users counter equal to 0, then resources for Lockd service are allocated in current network context.
5) On lockd_down() call: if current network context lockd users counter equal to 0, then resources for Lockd service are released in current network context (svc_shutdown_net() introduced in this series).
6) On lockd_down() call: if nlmsvc_users if equal to 0, then lockd thread is stopped and svc_destroy is called. And herewe can expect, that no service transports left.

I've just realized, that probably it's possible to implement some more generic helpers in SUNRPC code to make the code looks clearer.
I would appreciate for any advices how to do so.

--
Best regards,
Stanislav Kinsbursky

Subject: Re: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by bfields on Fri, 27 Jan 2012 14:33:35 GMT
View Forum Message <> Reply to Message

On Fri, Jan 27, 2012 at 01:08:06PM +0400, Stanislav Kinsbursky wrote:

> >On Wed, Jan 25, 2012 at 05:47:26PM +0400, Stanislav Kinsbursky wrote:
> >>This patch introduces network namespace filter for service destruction
> >>function.
> >>Nothing special here - just do exactly the same operations, but only for
> >>tranports in passed networks namespace context.
> >>BTW, BUG_ON() checks for empty service transports lists were returned into
> >>svc_destroy() function. This is because of swithing generic svc_close_all() to
> >>networks namespace dependable svc_close_net().
> >>
> >>Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
> >>
> >>---
> >> include/linux/sunrpc/svcsock.h | 2 +-
> >> net/sunrpc/svc.c | 9 +++++++--
> >> net/sunrpc/svc_xprt.c | 27 +++++++++++++++++++----------
> >> 3 files changed, 25 insertions(+), 13 deletions(-)
> >>
> >>diff --git a/include/linux/sunrpc/svcsock.h b/include/linux/sunrpc/svcsock.h
> >>index c84e974..cb4ac69 100644
> >>--- a/include/linux/sunrpc/svcsock.h
> >>+++ b/include/linux/sunrpc/svcsock.h
> >>@@ -34,7 +34,7 @@ struct svc_sock {
> >> /*
> >>   * Function prototypes.
> >>   */
> >>-void svc_close_all(struct svc_serv *);
> >>+void svc_close_net(struct svc_serv *, struct net *);
> >> int svc_recv(struct svc_rqst *, long);
> >> int svc_send(struct svc_rqst *);
> >> void svc_drop(struct svc_rqst *);
> >>diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
> >>index a8b49a0..279bbd8 100644
> >>--- a/net/sunrpc/svc.c
> >>+++ b/net/sunrpc/svc.c
> >>@@ -517,6 +517,8 @@ EXPORT_SYMBOL_GPL(svc_create_pooled);

> >> void
> >> svc_destroy(struct svc_serv *serv)
> >> {
> >>+ struct net *net = current->nsproxy->net_ns;
> >>+
> >>   dprintk("svc: svc_destroy(%s, %d)\n",
> >>   serv->sv_program->pg_name,
> >>   serv->sv_nrthreads);
> >>@@ -539,10 +541,13 @@ svc_destroy(struct svc_serv *serv)
> >>   * caller is using--nfsd_mutex in the case of nfsd). So it's
> >>   * safe to traverse those lists and shut everything down:
> >>   */
> >>- svc_close_all(serv);
> >>+ svc_close_net(serv, net);
> >>+
> >>+ BUG_ON(!list_empty(&serv->sv_permsocks));
> >>+ BUG_ON(!list_empty(&serv->sv_tempsocks));
> >
> >I'm confused--what guarantees this is true, at this point?
> >
>
> Hi, Bruce.
> I'm confused with your question. IOW, this must be true, because
> this code is executed only in case of last service thread is
> exiting, doesn't it?
>
> >There are two ways I could imagine containerizing svc_serv: either we
> >create a new one for each namespace, or we share a single global one
> >between them.
> >
>
> This is done for the second one.
>
> >If the former, then something that takes a "serv" argument shouldn't
> >also need a "net" argument--the serv should already know which namespace
> >it belongs to.
> >
> >If the latter, then these lists could have sockets from multiple
> >namespaces, and they aren't guaranteed to be empty here.
> >
> >?
> >
>
> I'll explain it on Lockd example (this code is done already - I just
> haven't sent it yet).
> Lockd is still only one thread and can handle lock requests from
> different network namespaces:
> 1) Introduced per-net lockd users counter and resources.

> 2) nlmsvc_users counter become global one. I.e. it's equal to sum of
> all per-net lockd users counters.
> 3) For each lockd_up() call global and current net lockd users
> counters are increased by one.
> 3) On lockd_up() call: if nlmsvc_users if equal to 0, then lockd thread is started.
> 4) On lockd_up() call: if current network context lockd users
> counter equal to 0, then resources for Lockd service are allocated
> in current network context.
> 5) On lockd_down() call: if current network context lockd users
> counter equal to 0, then resources for Lockd service are released in
> current network context (svc_shutdown_net() introduced in this
> series).
> 6) On lockd_down() call: if nlmsvc_users if equal to 0, then lockd
> thread is stopped and svc_destroy is called. And herewe can expect,
> that no service transports left.

OK, so at this point svc_close_net(serv, current->nsproxy->net_ns) is
enough to clear out sv_permsocks and sv_temp_socks because we know that
the only sockets left are in that namespace. Got it.

> I've just realized, that probably it's possible to implement some
> more generic helpers in SUNRPC code to make the code looks clearer.
> I would appreciate for any advices how to do so.

I'm not sure. The one thing that might have helped mere here would be
a comment to explain what's going on, maybe something like:

```
 svc_close_net(serv, net);
+ /*
+  * The last user is gone, so the only sockets left belonged its
+  * network namespace:
+  */
 BUG_ON(!list_empty(&serv->sv_permsocks));
 BUG_ON(!list_empty(&serv->sv_tempsocks));
```

--b.

Subject: Re: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by Stanislav Kinsbursky on Fri, 27 Jan 2012 15:21:18 GMT
View Forum Message <> Reply to Message

> OK, so at this point svc_close_net(serv, current->nsproxy->net_ns) is
> enough to clear out sv_permsocks and sv_temp_socks because we know that
> the only sockets left are in that namespace. Got it.
>

BTW, is this approach looks suitable for NFSd service?

> I'm not sure. The one thing that might have helped mere here would be
> a comment to explain what's going on, maybe something like:
>
> svc_close_net(serv, net);
> + /*
> + * The last user is gone, so the only sockets left belonged its
> + * network namespace:
> + */
> BUG_ON(!list_empty(&serv->sv_permsocks));
> BUG_ON(!list_empty(&serv->sv_tempsocks));

Ok, can do so.

--
Best regards,
Stanislav Kinsbursky