Subject: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by den on Wed, 17 Oct 2007 11:10:37 GMT
View Forum Message <> Reply to Message

/proc/sys/net/route/flush should be accessible inside the net namespace.
Though, the complete opening of this file will result in a DoS or
significant entire host slowdown if a namespace process will continually
flush routes.

This patch introduces per/namespace route flush facility.

Each namespace wanted to flush a cache copies global generation count to
itself and starts the timer. The cache is dropped for a specific namespace
iff the namespace counter is greater or equal global ones.

So, in general, unwanted namespaces do nothing. They hold very old low
counter and they are unaffected by the requested cleanup.

Signed-of-by: Denis V. Lunev <den@openvz.org>

```
diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index 85abf14..b492ce8 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -143,6 +143,8 @@ struct net {

  /* iptable_filter.c */
  struct xt_table  *ip_packet_filter;
+
+ unsigned long  rt_flush_required;
 };

 extern struct net init_net;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index f9a59ff..413587c 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -91,6 +91,7 @@
 #include <linux/jhash.h>
 #include <linux/rcupdate.h>
 #include <linux/times.h>
+#include <linux/nsproxy.h>
 #include <net/net_namespace.h>
 #include <net/protocol.h>
 #include <net/ip.h>
@@ -642,6 +643,51 @@ static void rt_check_expire(unsigned long dummy)
  mod_timer(&rt_periodic_timer, jiffies + ip_rt_gc_interval);
 }
```

```
+
+static DEFINE_SPINLOCK(rt_flush_lock);
+
+#ifdef CONFIG_NET_NS
+static unsigned long rt_flush_gen;
+
+/* called under rt_flush_lock */
+static void rt_flush_required_set(struct net *net)
+{
+ /*
+  * If the global generation rt_flush_gen is equal to G, then
+  * the pass considering entries labelled by G is yet to come.
+  */
+ net->rt_flush_required = rt_flush_gen;
+}
+
+static unsigned long rt_flush_required_reset(void)
+{
+ unsigned long g;
+
+ spin_lock_bh(&rt_flush_lock);
+ g = rt_flush_gen++;
+ spin_unlock_bh(&rt_flush_lock);
+ return g;
+}
+
+static int rt_flush_required_check(struct net *net, unsigned long gen)
+{
+ /* can be checked without the lock */
+ return net->rt_flush_required >= gen;
+}
+
+#else
+
+static void rt_flush_required_reset(struct net *net)
+{
+}
+
+static unsigned long rt_flush_required_reset(void)
+{
+ return 0;
+}
+#endif
+
+
 /* This can run from both BH and non-BH contexts, the latter
  * in the case of a forced flush event.
  */
@@ -649,16 +695,46 @@ static void rt_run_flush(unsigned long dummy)
 {
  int i;
  struct rtable *rth, *next;
+ unsigned long gen;

  rt_deadline = 0;

  get_random_bytes(&rt_hash_rnd, 4);
+ gen = rt_flush_required_reset();

  for (i = rt_hash_mask; i >= 0; i--) {
+#ifdef CONFIG_NET_NS
+  struct rtable **prev, *p, *tail;
+
+  spin_lock_bh(rt_hash_lock_addr(i));
+  rth = rt_hash_table[i].chain;
+
+  /* defer releasing the head of the list after spin_unlock */
+  for (tail = rth; tail; tail = tail->u.dst.rt_next)
+   if (!rt_flush_required_check(tail->fl.fl_net, gen))
+    break;
+  if (rth != tail)
+   rt_hash_table[i].chain = tail;
+
+  /* call rt_free on entries after the tail requiring flush */
+  prev = &rt_hash_table[i].chain;
+  for (p = *prev; p; p = next) {
+   next = p->u.dst.rt_next;
+   if (!rt_flush_required_check(p->fl.fl_net, gen)) {
+    prev = &p->u.dst.rt_next;
+   } else {
+    *prev = next;
+    rt_free(p);
+   }
+  }
+#else
   spin_lock_bh(rt_hash_lock_addr(i));
   rth = rt_hash_table[i].chain;
   if (rth)
    rt_hash_table[i].chain = NULL;
+  tail = NULL;
+
+#endif
   spin_unlock_bh(rt_hash_lock_addr(i));

   for (; rth; rth = next) {
```

```
@@ -668,8 +744,6 @@ static void rt_run_flush(unsigned long dummy)
  }
 }

-static DEFINE_SPINLOCK(rt_flush_lock);
-
 void rt_cache_flush(int delay)
 {
  unsigned long now = jiffies;
@@ -697,6 +771,8 @@ void rt_cache_flush(int delay)
    delay = tmo;
  }

+ rt_flush_required_set(current->nsproxy->net_ns);
+
  if (delay <= 0) {
   spin_unlock_bh(&rt_flush_lock);
   rt_run_flush(0);
```

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Wed, 17 Oct 2007 11:46:23 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> /proc/sys/net/route/flush should be accessible inside the net namespace.
> Though, the complete opening of this file will result in a DoS or
> significant entire host slowdown if a namespace process will continually
> flush routes.
>
> This patch introduces per/namespace route flush facility.
>
> Each namespace wanted to flush a cache copies global generation count to
> itself and starts the timer. The cache is dropped for a specific namespace
> iff the namespace counter is greater or equal global ones.
>
> So, in general, unwanted namespaces do nothing. They hold very old low
> counter and they are unaffected by the requested cleanup.
>
> Signed-of-by: Denis V. Lunev <den@openvz.org>

That's right and that will happen when manipulating ip addresses of the
network devices too. But I am not confortable with your patchset. It
touches the routing flush function too hardly and it uses
current->nsproxy->net_ns.

IMHO we should have two flush functions. One taking a network namespace

parameter and one without the network namespace parameter. The first one
is called when a write to /proc/sys/net/ipv4/route/flush is done (we
must use the network namespace of the writer) or when a interface
address is changed or shutdown|up. The last one is called by the timer,
so we have a global timer flushing the routing cache for all the namespaces.

```
> diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
> index 85abf14..b492ce8 100644
> --- a/include/net/net_namespace.h
> +++ b/include/net/net_namespace.h
> @@ -143,6 +143,8 @@ struct net {
>
>   /* iptable_filter.c */
>   struct xt_table  *ip_packet_filter;
> +
> + unsigned long  rt_flush_required;
>  };
>
>  extern struct net init_net;
> diff --git a/net/ipv4/route.c b/net/ipv4/route.c
> index f9a59ff..413587c 100644
> --- a/net/ipv4/route.c
> +++ b/net/ipv4/route.c
> @@ -91,6 +91,7 @@
>  #include <linux/jhash.h>
>  #include <linux/rcupdate.h>
>  #include <linux/times.h>
> +#include <linux/nsproxy.h>
>  #include <net/net_namespace.h>
>  #include <net/protocol.h>
>  #include <net/ip.h>
> @@ -642,6 +643,51 @@ static void rt_check_expire(unsigned long dummy)
>   mod_timer(&rt_periodic_timer, jiffies + ip_rt_gc_interval);
>  }
>
> +
> +static DEFINE_SPINLOCK(rt_flush_lock);
> +
> +#ifdef CONFIG_NET_NS
> +static unsigned long rt_flush_gen;
> +
> +/* called under rt_flush_lock */
> +static void rt_flush_required_set(struct net *net)
> +{
> + /*
> +  * If the global generation rt_flush_gen is equal to G, then
> +  * the pass considering entries labelled by G is yet to come.
> +  */
> + net->rt_flush_required = rt_flush_gen;
> +}
> +
> +static unsigned long rt_flush_required_reset(void)
> +{
> + unsigned long g;
> +
> + spin_lock_bh(&rt_flush_lock);
> + g = rt_flush_gen++;
> + spin_unlock_bh(&rt_flush_lock);
> + return g;
> +}
> +
> +static int rt_flush_required_check(struct net *net, unsigned long gen)
> +{
> + /* can be checked without the lock */
> + return net->rt_flush_required >= gen;
> +}
> +
> +#else
> +
> +static void rt_flush_required_reset(struct net *net)
> +{
> +}
> +
> +static unsigned long rt_flush_required_reset(void)
> +{
> + return 0;
> +}
> +#endif
> +
> +
>  /* This can run from both BH and non-BH contexts, the latter
>   * in the case of a forced flush event.
>   */
> @@ -649,16 +695,46 @@ static void rt_run_flush(unsigned long dummy)
>  {
>   int i;
>   struct rtable *rth, *next;
> + unsigned long gen;
>
>   rt_deadline = 0;
>
>   get_random_bytes(&rt_hash_rnd, 4);
> + gen = rt_flush_required_reset();
>
>   for (i = rt_hash_mask; i >= 0; i--) {
```

```
> +#ifdef CONFIG_NET_NS
> +  struct rtable **prev, *p, *tail;
> +
> +  spin_lock_bh(rt_hash_lock_addr(i));
> +  rth = rt_hash_table[i].chain;
> +
> +  /* defer releasing the head of the list after spin_unlock */
> +  for (tail = rth; tail; tail = tail->u.dst.rt_next)
> +   if (!rt_flush_required_check(tail->fl.fl_net, gen))
> +    break;
> +  if (rth != tail)
> +   rt_hash_table[i].chain = tail;
> +
> +  /* call rt_free on entries after the tail requiring flush */
> +  prev = &rt_hash_table[i].chain;
> +  for (p = *prev; p; p = next) {
> +   next = p->u.dst.rt_next;
> +   if (!rt_flush_required_check(p->fl.fl_net, gen)) {
> +    prev = &p->u.dst.rt_next;
> +   } else {
> +    *prev = next;
> +    rt_free(p);
> +   }
> +  }
> +#else
>    spin_lock_bh(rt_hash_lock_addr(i));
>    rth = rt_hash_table[i].chain;
>    if (rth)
>     rt_hash_table[i].chain = NULL;
> +  tail = NULL;
> +
> +#endif
>    spin_unlock_bh(rt_hash_lock_addr(i));
>
>    for (; rth; rth = next) {
> @@ -668,8 +744,6 @@ static void rt_run_flush(unsigned long dummy)
>   }
>  }
>
> -static DEFINE_SPINLOCK(rt_flush_lock);
> -
>  void rt_cache_flush(int delay)
>  {
>   unsigned long now = jiffies;
> @@ -697,6 +771,8 @@ void rt_cache_flush(int delay)
>     delay = tmo;
>   }
>
```

```
> + rt_flush_required_set(current->nsproxy->net_ns);
> +
>   if (delay <= 0) {
>    spin_unlock_bh(&rt_flush_lock);
>    rt_run_flush(0);
```

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by den on Wed, 17 Oct 2007 12:49:49 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>> /proc/sys/net/route/flush should be accessible inside the net namespace.
>> Though, the complete opening of this file will result in a DoS or
>> significant entire host slowdown if a namespace process will continually
>> flush routes.
>>
>> This patch introduces per/namespace route flush facility.
>>
>> Each namespace wanted to flush a cache copies global generation count to
>> itself and starts the timer. The cache is dropped for a specific
>> namespace
>> iff the namespace counter is greater or equal global ones.
>>
>> So, in general, unwanted namespaces do nothing. They hold very old low
>> counter and they are unaffected by the requested cleanup.
>>
>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>
> That's right and that will happen when manipulating ip addresses of the
> network devices too. But I am not confortable with your patchset. It
> touches the routing flush function too hardly and it uses
> current->nsproxy->net_ns.
>
> IMHO we should have two flush functions. One taking a network namespace
> parameter and one without the network namespace parameter. The first one
> is called when a write to /proc/sys/net/ipv4/route/flush is done (we
> must use the network namespace of the writer) or when a interface
> address is changed or shutdown|up. The last one is called by the timer,
> so we have a global timer flushing the routing cache for all the
> namespaces.

we can't :(The unfortunate thing is that the actual cleanup is called
indirectly and asynchronously. The user _schedule_ the garbage collector
to run NOW and we are moving over a large routing cache. Really large.

The idea to iterate over the list of each namespace to flush is bad. We
are in atomic context. The list is protected by the mutex.

The idea of several timers (per namespace) is also bad. You will iterate
over large cache several times.

No other acceptable way here for me :(.

As for "the trigger" - rt_cache_flush, looks like you are right. We
should pass namespace as a parameter. This should be done as a separate
patch.

Regards,
 Den

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Wed, 17 Oct 2007 13:40:56 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> /proc/sys/net/route/flush should be accessible inside the net namespace.
>>> Though, the complete opening of this file will result in a DoS or
>>> significant entire host slowdown if a namespace process will continually
>>> flush routes.
>>>
>>> This patch introduces per/namespace route flush facility.
>>>
>>> Each namespace wanted to flush a cache copies global generation count to
>>> itself and starts the timer. The cache is dropped for a specific
>>> namespace
>>> iff the namespace counter is greater or equal global ones.
>>>
>>> So, in general, unwanted namespaces do nothing. They hold very old low
>>> counter and they are unaffected by the requested cleanup.
>>>
>>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>>
>> That's right and that will happen when manipulating ip addresses of
>> the network devices too. But I am not confortable with your patchset.
>> It touches the routing flush function too hardly and it uses
>> current->nsproxy->net_ns.
>>
>> IMHO we should have two flush functions. One taking a network
>> namespace parameter and one without the network namespace parameter.

>> The first one is called when a write to /proc/sys/net/ipv4/route/flush
>> is done (we must use the network namespace of the writer) or when a
>> interface address is changed or shutdown|up. The last one is called by
>> the timer, so we have a global timer flushing the routing cache for
>> all the namespaces.
>
> we can't :(The unfortunate thing is that the actual cleanup is called
> indirectly and asynchronously. The user _schedule_ the garbage collector
> to run NOW and we are moving over a large routing cache. Really large.
>
> The idea to iterate over the list of each namespace to flush is bad. We
> are in atomic context. The list is protected by the mutex.
>
> The idea of several timers (per namespace) is also bad. You will iterate
> over large cache several times.
>
> No other acceptable way here for me :(.
>
> As for "the trigger" - rt_cache_flush, looks like you are right. We
> should pass namespace as a parameter. This should be done as a separate
> patch.

If we change:

```
 rt_cache_flush(struct net *net, int delay);
```

and inside we call rt_cache_flush((unsigned long)net);

And then we check in the rt_run_flush function,

```
 struct net *net = (struct net *)dummy;

 ...
 for (i = rt_hash_mask; i >= 0; i--) {
  ...
  if (dummy && rth->fl.fl_net != net)
   continue
  ...
 ...
```

So when rt_run_flush is called synchronously, the netns is specified in
dummy and only the routes belonging to netns are flushed. Otherwise when
it is called by the timer, netns is not set so all routes are flushed.

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup

Posted by den on Wed, 17 Oct 2007 14:08:21 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:
>>> Denis V. Lunev wrote:
>>>> /proc/sys/net/route/flush should be accessible inside the net
>>>> namespace.
>>>> Though, the complete opening of this file will result in a DoS or
>>>> significant entire host slowdown if a namespace process will
>>>> continually
>>>> flush routes.
>>>>
>>>> This patch introduces per/namespace route flush facility.
>>>>
>>>> Each namespace wanted to flush a cache copies global generation
>>>> count to
>>>> itself and starts the timer. The cache is dropped for a specific
>>>> namespace
>>>> iff the namespace counter is greater or equal global ones.
>>>>
>>>> So, in general, unwanted namespaces do nothing. They hold very old low
>>>> counter and they are unaffected by the requested cleanup.
>>>>
>>>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>>>
>>> That's right and that will happen when manipulating ip addresses of
>>> the network devices too. But I am not confortable with your patchset.
>>> It touches the routing flush function too hardly and it uses
>>> current->nsproxy->net_ns.
>>>
>>> IMHO we should have two flush functions. One taking a network
>>> namespace parameter and one without the network namespace parameter.
>>> The first one is called when a write to
>>> /proc/sys/net/ipv4/route/flush is done (we must use the network
>>> namespace of the writer) or when a interface address is changed or
>>> shutdown|up. The last one is called by the timer, so we have a global
>>> timer flushing the routing cache for all the namespaces.
>>
>> we can't :(The unfortunate thing is that the actual cleanup is called
>> indirectly and asynchronously. The user _schedule_ the garbage
>> collector to run NOW and we are moving over a large routing cache.
>> Really large.
>>
>> The idea to iterate over the list of each namespace to flush is bad.
>> We are in atomic context. The list is protected by the mutex.
>>
>> The idea of several timers (per namespace) is also bad. You will

```
>> iterate over large cache several times.
>>
>> No other acceptable way here for me :(.
>>
>> As for "the trigger" - rt_cache_flush, looks like you are right. We
>> should pass namespace as a parameter. This should be done as a
>> separate patch.
>
> If we change:
>
>     rt_cache_flush(struct net *net, int delay);
>
> and inside we call rt_cache_flush((unsigned long)net);
>
> And then we check in the rt_run_flush function,
>
>     struct net *net = (struct net *)dummy;
>
>     ...
>     for (i = rt_hash_mask; i >= 0; i--) {
>         ...
>         if (dummy && rth->fl.fl_net != net)
>             continue
>         ...
>     ...
>
> So when rt_run_flush is called synchronously, the netns is specified in
> dummy and only the routes belonging to netns are flushed. Otherwise when
> it is called by the timer, netns is not set so all routes are flushed.
>
```

this does not look good for me. The size of this cache for 4GB host is
2*10^6 entries for IPv4 with a 131072 chains. The conventional
mainstream kernel wants to merge the routing cache cleanup requests from
the different sources if they are delayed (default).

The main idea for this patch is to protect all other namespaces from the
current one. This cache is an important resource. You proposal will work
for the forced synchronous cleanups. Though, there are some requests
with results in the delayed rt_run_flush via [mod/add]_timer

How should we handle them?

Regards,
Den

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Wed, 17 Oct 2007 14:46:34 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Denis V. Lunev wrote:
>>>>> /proc/sys/net/route/flush should be accessible inside the net
>>>>> namespace.
>>>>> Though, the complete opening of this file will result in a DoS or
>>>>> significant entire host slowdown if a namespace process will
>>>>> continually
>>>>> flush routes.
>>>>>
>>>>> This patch introduces per/namespace route flush facility.
>>>>>
>>>>> Each namespace wanted to flush a cache copies global generation
>>>>> count to
>>>>> itself and starts the timer. The cache is dropped for a specific
>>>>> namespace
>>>>> iff the namespace counter is greater or equal global ones.
>>>>>
>>>>> So, in general, unwanted namespaces do nothing. They hold very old low
>>>>> counter and they are unaffected by the requested cleanup.
>>>>>
>>>>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>>>>
>>>> That's right and that will happen when manipulating ip addresses of
>>>> the network devices too. But I am not confortable with your
>>>> patchset. It touches the routing flush function too hardly and it
>>>> uses current->nsproxy->net_ns.
>>>>
>>>> IMHO we should have two flush functions. One taking a network
>>>> namespace parameter and one without the network namespace parameter.
>>>> The first one is called when a write to
>>>> /proc/sys/net/ipv4/route/flush is done (we must use the network
>>>> namespace of the writer) or when a interface address is changed or
>>>> shutdown|up. The last one is called by the timer, so we have a
>>>> global timer flushing the routing cache for all the namespaces.
>>>
>>> we can't :(The unfortunate thing is that the actual cleanup is
>>> called indirectly and asynchronously. The user _schedule_ the garbage
>>> collector to run NOW and we are moving over a large routing cache.
>>> Really large.
>>>
>>> The idea to iterate over the list of each namespace to flush is bad.

>>> We are in atomic context. The list is protected by the mutex.
>>>
>>> The idea of several timers (per namespace) is also bad. You will
>>> iterate over large cache several times.
>>>
>>> No other acceptable way here for me :(.
>>>
>>> As for "the trigger" - rt_cache_flush, looks like you are right. We
>>> should pass namespace as a parameter. This should be done as a
>>> separate patch.
>>
>> If we change:
>>

```
>>     rt_cache_flush(struct net *net, int delay);
```

>>
>> and inside we call rt_cache_flush((unsigned long)net);
>>
>> And then we check in the rt_run_flush function,
>>

```
>>     struct net *net = (struct net *)dummy;
>>
>>     ...
>>     for (i = rt_hash_mask; i >= 0; i--) {
>>         ...
>>         if (dummy && rth->fl.fl_net != net)
>>             continue
>>         ...
>>     ...
>>
```

>> So when rt_run_flush is called synchronously, the netns is specified
>> in dummy and only the routes belonging to netns are flushed. Otherwise
>> when it is called by the timer, netns is not set so all routes are
>> flushed.
>>
>
> this does not look good for me. The size of this cache for 4GB host is
> 2*10^6 entries for IPv4 with a 131072 chains. The conventional
> mainstream kernel wants to merge the routing cache cleanup requests from
> the different sources if they are delayed (default).
>
> The main idea for this patch is to protect all other namespaces from the
> current one. This cache is an important resource. You proposal will work
> for the forced synchronous cleanups. Though, there are some requests
> with results in the delayed rt_run_flush via [mod/add]_timer
>
> How should we handle them?

IHMO we should let the timer to remove all routes.

The problem you raised is someone in a namespace can flush routes for
another namespaces and mess the network traffic for them.
This case is resolved with the synchronous call to rt_cache_flush and
netns parameter.

The second point here is we can have a lot of routes and the timer will
flush them all, with or without namespaces.
IMHO I don't think we should handle this case ... for now.

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Wed, 17 Oct 2007 15:05:36 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Denis V. Lunev wrote:
>>>>> /proc/sys/net/route/flush should be accessible inside the net
>>>>> namespace.
>>>>> Though, the complete opening of this file will result in a DoS or
>>>>> significant entire host slowdown if a namespace process will
>>>>> continually
>>>>> flush routes.
>>>>>
>>>>> This patch introduces per/namespace route flush facility.
>>>>>
>>>>> Each namespace wanted to flush a cache copies global generation
>>>>> count to
>>>>> itself and starts the timer. The cache is dropped for a specific
>>>>> namespace
>>>>> iff the namespace counter is greater or equal global ones.
>>>>>
>>>>> So, in general, unwanted namespaces do nothing. They hold very old low
>>>>> counter and they are unaffected by the requested cleanup.
>>>>>
>>>>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>>>>
>>>> That's right and that will happen when manipulating ip addresses of
>>>> the network devices too. But I am not confortable with your
>>>> patchset. It touches the routing flush function too hardly and it
>>>> uses current->nsproxy->net_ns.
>>>>
>>>> IMHO we should have two flush functions. One taking a network
>>>> namespace parameter and one without the network namespace parameter.

>>>> The first one is called when a write to
>>>> /proc/sys/net/ipv4/route/flush is done (we must use the network
>>>> namespace of the writer) or when a interface address is changed or
>>>> shutdown|up. The last one is called by the timer, so we have a
>>>> global timer flushing the routing cache for all the namespaces.
>>>
>>> we can't :(The unfortunate thing is that the actual cleanup is
>>> called indirectly and asynchronously. The user _schedule_ the garbage
>>> collector to run NOW and we are moving over a large routing cache.
>>> Really large.
>>>
>>> The idea to iterate over the list of each namespace to flush is bad.
>>> We are in atomic context. The list is protected by the mutex.

Oh, by the way, I forgot something important you spotted with the list
protected by the mutex.

When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
network namespaces list for the garbage collecting, but we are in an
interrupt handler, so I can not use rtnl_lock.

Why is not possible to protect the list with a simple spinlock ? so we
can call spin_lock_bh when we are in interrupt handler.

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by dlunev on Wed, 17 Oct 2007 17:54:33 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Oh, by the way, I forgot something important you spotted with the list
> protected by the mutex.
>
> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
> network namespaces list for the garbage collecting, but we are in an
> interrupt handler, so I can not use rtnl_lock.
where exactly....

all interesting places are called under rtnl already...

> Why is not possible to protect the list with a simple spinlock ? so we
> can call spin_lock_bh when we are in interrupt handler.

see my old patch with a locking rules :)

By the way, I have forgotten to mention, that the original patch works
from 2004 for OpenVz :)

Regards,
 Den

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by ebiederm on Wed, 17 Oct 2007 18:50:35 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <dlunev@gmail.com> writes:

> Daniel Lezcano wrote:
>> Oh, by the way, I forgot something important you spotted with the list
>> protected by the mutex.
>>
>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the network
>> namespaces list for the garbage collecting, but we are in an interrupt
>> handler, so I can not use rtnl_lock.
> where exactly....
>
> all interesting places are called under rtnl already...
>
>> Why is not possible to protect the list with a simple spinlock ? so we can
>> call spin_lock_bh when we are in interrupt handler.
>
> see my old patch with a locking rules :)

Actually at this point given that your rtnl_unlock() cleanup made it
in things are much simpler and if we really need to we can add a
spinlock protecting things.

> By the way, I have forgotten to mention, that the original patch works from 2004
> for OpenVz :)

Probably. It is a different optimization and maintenance point
however. So while it can inspire things it isn't an apples to apples
comparison.

Eric

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Benjamin Thery on Thu, 18 Oct 2007 07:18:48 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Oh, by the way, I forgot something important you spotted with the list
>> protected by the mutex.
>>
>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>> network namespaces list for the garbage collecting, but we are in an
>> interrupt handler, so I can not use rtnl_lock.
> where exactly....

Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().

fib6_clean_all() is called by fib6_run_gc() handler of the
ip6_fib_timer.
If we don't want to have one such timer per net namespace, in
fib6_clean_all() we have to go through all net to clean their own
fib_table_hash (using for_each_net() protected by rtnl_lock).

> all interesting places are called under rtnl already...
>
>> Why is not possible to protect the list with a simple spinlock ? so we
>> can call spin_lock_bh when we are in interrupt handler.
>
> see my old patch with a locking rules :)
>
> By the way, I have forgotten to mention, that the original patch works
> from 2004 for OpenVz :)
>
> Regards,
>   Den
>

--
B e n j a m i n   T h e r y - BULL/DT/Open Software R&D

   http://www.bull.com

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by den on Thu, 18 Oct 2007 09:53:04 GMT
View Forum Message <> Reply to Message

Benjamin Thery wrote:
> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:

>>> Oh, by the way, I forgot something important you spotted with the list
>>> protected by the mutex.
>>>
>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>> network namespaces list for the garbage collecting, but we are in an
>>> interrupt handler, so I can not use rtnl_lock.
>> where exactly....
>
> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>
> fib6_clean_all() is called by fib6_run_gc() handler of the
> ip6_fib_timer.
> If we don't want to have one such timer per net namespace, in
> fib6_clean_all() we have to go through all net to clean their own
> fib_table_hash (using for_each_net() protected by rtnl_lock).

The locking in mainstream is different with NETNS49.

I have proposed to add a dev_base_lock to protect namespace list for
atomic context to be in sync with rtnl usage.

May be we should introduce an additional one. I also do not see other
way for that place. May be you can copy my approach with generations for
IPv6 code. I'll send a new version in a minute.

Regards,
 Den

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by den on Thu, 18 Oct 2007 14:51:23 GMT
View Forum Message <> Reply to Message

Benjamin Thery wrote:
> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:
>>> Oh, by the way, I forgot something important you spotted with the list
>>> protected by the mutex.
>>>
>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>> network namespaces list for the garbage collecting, but we are in an
>>> interrupt handler, so I can not use rtnl_lock.
>> where exactly....
>
> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>
> fib6_clean_all() is called by fib6_run_gc() handler of the

> ip6_fib_timer.
> If we don't want to have one such timer per net namespace, in
> fib6_clean_all() we have to go through all net to clean their own
> fib_table_hash (using for_each_net() protected by rtnl_lock).

after careful thinking, one timer per/namespace looks better for me :)

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Benjamin Thery on Thu, 18 Oct 2007 16:29:49 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Benjamin Thery wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Oh, by the way, I forgot something important you spotted with the list
>>>> protected by the mutex.
>>>>
>>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>>> network namespaces list for the garbage collecting, but we are in an
>>>> interrupt handler, so I can not use rtnl_lock.
>>> where exactly....
>>
>> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>>
>> fib6_clean_all() is called by fib6_run_gc() handler of the
>> ip6_fib_timer. If we don't want to have one such timer per net
>> namespace, in fib6_clean_all() we have to go through all net to clean
>> their own
>> fib_table_hash (using for_each_net() protected by rtnl_lock).
>
> after careful thinking, one timer per/namespace looks better for me :)

Why? :)
Then you'll have to find a way pass the target net to fib6_run_gc()
(the timer handler). current->nsproxy->net_ns won't work :)
One timer for all looked simpler to me.

Can there be an impact on performance if we have several GC timers
for the several netns running?

Benjamin

--
B e n j a m i n   T h e r y - BULL/DT/Open Software R&D

http://www.bull.com

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup

Posted by dlunev on Thu, 18 Oct 2007 18:59:50 GMT

View Forum Message <> Reply to Message

Benjamin Thery wrote:
> Denis V. Lunev wrote:
>> Benjamin Thery wrote:
>>> Denis V. Lunev wrote:
>>>> Daniel Lezcano wrote:
>>>>> Oh, by the way, I forgot something important you spotted with the list
>>>>> protected by the mutex.
>>>>>
>>>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>>>> network namespaces list for the garbage collecting, but we are in an
>>>>> interrupt handler, so I can not use rtnl_lock.
>>>> where exactly....
>>> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>>>
>>> fib6_clean_all() is called by fib6_run_gc() handler of the
>>> ip6_fib_timer. If we don't want to have one such timer per net
>>> namespace, in fib6_clean_all() we have to go through all net to clean
>>> their own
>>> fib_table_hash (using for_each_net() protected by rtnl_lock).
>> after careful thinking, one timer per/namespace looks better for me :)
>
> Why? :)

This is
- scalable: no long-long iteration over 1000 namespace
- easy in respect to a patch I just sent. We should not allow to drop
cache from all namespaces from one
- and this approach was good for Alexey Kuznetsov. We have talked about
this discussing OpenVZ implementation

> Then you'll have to find a way pass the target net to fib6_run_gc()
> (the timer handler). current->nsproxy->net_ns won't work :)
> One timer for all looked simpler to me.
>
> Can there be an impact on performance if we have several GC timers
> for the several netns running?

the amount of job is not greater. Isn't it?

Regards,

Den

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by dlunev on Thu, 18 Oct 2007 19:03:35 GMT
View Forum Message <> Reply to Message

Benjamin Thery wrote:
> Denis V. Lunev wrote:
>> Benjamin Thery wrote:
>>> Denis V. Lunev wrote:
>>>> Daniel Lezcano wrote:
>>>>> Oh, by the way, I forgot something important you spotted with the list
>>>>> protected by the mutex.
>>>>>
>>>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>>>> network namespaces list for the garbage collecting, but we are in an
>>>>> interrupt handler, so I can not use rtnl_lock.
>>>> where exactly....
>>> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>>>
>>> fib6_clean_all() is called by fib6_run_gc() handler of the
>>> ip6_fib_timer. If we don't want to have one such timer per net
>>> namespace, in fib6_clean_all() we have to go through all net to clean
>>> their own
>>> fib_table_hash (using for_each_net() protected by rtnl_lock).
>> after careful thinking, one timer per/namespace looks better for me :)
>
> Why? :)
> Then you'll have to find a way pass the target net to fib6_run_gc()
> (the timer handler). current->nsproxy->net_ns won't work :)

sorry for spam :) forget to mention that the namespace can be passed via
timer data :) yes - additional structure will be required

By the way, why do you, guys, so dislike our approach :) In that case
there will be no problem to get context from currrent. Yes. In soft
interrupt also :))

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Fri, 19 Oct 2007 07:39:06 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:

> Benjamin Thery wrote:
>> Denis V. Lunev wrote:
>>> Benjamin Thery wrote:
>>>> Denis V. Lunev wrote:
>>>>> Daniel Lezcano wrote:
>>>>>> Oh, by the way, I forgot something important you spotted with the
>>>>>> list
>>>>>> protected by the mutex.
>>>>>>
>>>>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>>>>> network namespaces list for the garbage collecting, but we are in an
>>>>>> interrupt handler, so I can not use rtnl_lock.
>>>>> where exactly....
>>>> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>>>>
>>>> fib6_clean_all() is called by fib6_run_gc() handler of the
>>>> ip6_fib_timer. If we don't want to have one such timer per net
>>>> namespace, in fib6_clean_all() we have to go through all net to clean
>>>> their own
>>>> fib_table_hash (using for_each_net() protected by rtnl_lock).
>>> after careful thinking, one timer per/namespace looks better for me :)
>>
>> Why? :)
>> Then you'll have to find a way pass the target net to fib6_run_gc()
>> (the timer handler). current->nsproxy->net_ns won't work :)
>
> sorry for spam :) forget to mention that the namespace can be passed via
> timer data :) yes - additional structure will be required
>
> By the way, why do you, guys, so dislike our approach :) In that case
> there will be no problem to get context from currrent. Yes. In soft
> interrupt also :))

the approach of the patchset is : we don't use current->nsproxy->net_ns.

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Fri, 19 Oct 2007 07:39:24 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Benjamin Thery wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Oh, by the way, I forgot something important you spotted with the list
>>>> protected by the mutex.
>>>>

>>>> When looking at ipv6/fib_hash.c with Benjamin, we need to browse the
>>>> network namespaces list for the garbage collecting, but we are in an
>>>> interrupt handler, so I can not use rtnl_lock.
>>> where exactly....
>>
>> Actually, it is in net/ipv6/ip6_fib.c, in fib6_clean_all().
>>
>> fib6_clean_all() is called by fib6_run_gc() handler of the
>> ip6_fib_timer. If we don't want to have one such timer per net
>> namespace, in fib6_clean_all() we have to go through all net to clean
>> their own
>> fib_table_hash (using for_each_net() protected by rtnl_lock).
>
> after careful thinking, one timer per/namespace looks better for me :)

So you prefer to have multiple timers flushing a portion of a shared
routing cache ?

It is not impossible I already tryed that for ipv6. it will just touch
much more files.

If we want, at all costs, to have the routing cache flushed per
namespace, I am much more favorable to have a timer per namespace than
using the algorithm you sent previously.

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by den on Fri, 19 Oct 2007 08:51:35 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> So you prefer to have multiple timers flushing a portion of a shared
> routing cache ?
>
> It is not impossible I already tryed that for ipv6. it will just touch
> much more files.
>
> If we want, at all costs, to have the routing cache flushed per
> namespace, I am much more favorable to have a timer per namespace than
> using the algorithm you sent previously.

as do I. But the IPv4 and IPv6 situation is much different. IPv6 cache
is separated while IPv4 one is common.

So, the situation is not uniform, at least :(

By the way, we can move routing cache into the FIB tree like one for IPv4 :)

Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by ebiederm on Fri, 19 Oct 2007 19:03:31 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

> Daniel Lezcano wrote:
>> So you prefer to have multiple timers flushing a portion of a shared routing
>> cache ?
>>
>> It is not impossible I already tryed that for ipv6. it will just touch much
>> more files.
>>
>> If we want, at all costs, to have the routing cache flushed per namespace, I
>> am much more favorable to have a timer per namespace than using the algorithm
>> you sent previously.
>
> as do I. But the IPv4 and IPv6 situation is much different. IPv6 cache is
> separated while IPv4 one is common.
>
> So, the situation is not uniform, at least :(
>
> By the way, we can move routing cache into the FIB tree like one for IPv4 :)

I'm trying to think through this.

- We have two cases garbage collection and flushing the cache.

- The routing cache is just a cache we loose nothing except
  performance if it is flushed. At least that is my memory.

- Especially for ipv4 the routing cache is a shared resource because
  of a shared hash table, therefore we want to do the garbage
  collection globally. Especially since we should be able to
  make better decisions that way.

- We appear to use a timer beneath flush to keep flushing operations
  from happening to frequently, and thus killing everyones
  performance.

- We also flush the timer when it is time to change secrets.

So as long as we are really only making the user space flush
operation per namespace I don't have a real problem with it,
as that timer should not run periodically and the whole reason
for the existence of the timer is to ensure people don't kill
other peoples performance by flushing to often.

I do still think things like the periodic secret rebuild and
the garbage collection should continue to be global across
the entire cache.

Eric
