
Subject: [PATCH net-2.6.25 14/19] [NETNS] Place fib tables into netns.

Posted by [den](#) on Wed, 09 Jan 2008 18:01:05 GMT

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

The preparatory work has been done. All we need is to substitute fib_table_hash with net->ipv4.fib_table_hash. Netns context is available when required.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>

Acked-by: Daniel Lezcano <dlezcano@fr.ibm.com>

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

```
include/net/ip_fib.h      | 6 ++---
include/net/netns/ipv4.h | 2 ++
net/ipv4/fib_frontend.c   | 36 ++++++-----
3 files changed, 28 insertions(+), 16 deletions(-)
```

```
diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
```

```
index 1faad1f..ddb4c20 100644
```

```
--- a/include/net/ip_fib.h
```

```
+++ b/include/net/ip_fib.h
```

```
@@ -120,8 +120,6 @@ struct fib_result_nl {
    int      err;
};
```

```
-extern struct hlist_head fib_table_hash[];
```

```
-
#ifdef CONFIG_IP_ROUTE_MULTIPATH

#define FIB_RES_NH(res) ((res).fi->fib_nh[(res).nh_sel])
@@ -169,9 +167,9 @@ static inline struct fib_table *fib_get_table(struct net *net, u32 id)
{
    struct hlist_head *ptr;

- ptr = id == RT_TABLE_LOCAL ?
- &fib_table_hash[TABLE_LOCAL_INDEX] :
- &fib_table_hash[TABLE_MAIN_INDEX];
+ ptr = id == RT_TABLE_LOCAL ?
+ &net->ipv4.fib_table_hash[TABLE_LOCAL_INDEX] :
+ &net->ipv4.fib_table_hash[TABLE_MAIN_INDEX];
    return hlist_entry(ptr->first, struct fib_table, tb_hlist);
}
```

```
diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
```

```
index 299f8c6..629ec6c 100644
```

```
--- a/include/net/netns/ipv4.h
```

```
+++ b/include/net/netns/ipv4.h
```

```
@@ -8,6 +8,7 @@
```

```

struct ctl_table_header;
struct ipv4_devconf;
struct fib_rules_ops;
+struct hlist_head;

struct netns_ipv4 {
#ifdef CONFIG_SYSCTL
@@ -16,5 +17,6 @@ struct netns_ipv4 {
#ifdef CONFIG_IP_MULTIPLE_TABLES
    struct fib_rules_ops *rules_ops;
#endif
+ struct hlist_head *fib_table_hash;
};
#endif
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 08414eb..2e150ae 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -50,7 +50,6 @@
#define FFprint(a...) printk(KERN_DEBUG a)

static struct sock *fibnl;
-struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];

#ifdef CONFIG_IP_MULTIPLE_TABLES

@@ -67,9 +66,9 @@ static int __net_init fib4_rules_init(struct net *net)
    goto fail;

    hlist_add_head_rcu(&local_table->tb_hlist,
-    &fib_table_hash[TABLE_LOCAL_INDEX]);
+    &net->ipv4.fib_table_hash[TABLE_LOCAL_INDEX]);
    hlist_add_head_rcu(&main_table->tb_hlist,
-    &fib_table_hash[TABLE_MAIN_INDEX]);
+    &net->ipv4.fib_table_hash[TABLE_MAIN_INDEX]);
    return 0;

fail:
@@ -92,7 +91,7 @@ struct fib_table *fib_new_table(struct net *net, u32 id)
    if (!tb)
        return NULL;
    h = id & (FIB_TABLE_HASHSZ - 1);
-    hlist_add_head_rcu(&tb->tb_hlist, &fib_table_hash[h]);
+    hlist_add_head_rcu(&tb->tb_hlist, &net->ipv4.fib_table_hash[h]);
    return tb;
}

@@ -100,13 +99,16 @@ struct fib_table *fib_get_table(struct net *net, u32 id)

```

```

{
    struct fib_table *tb;
    struct hlist_node *node;
+ struct hlist_head *head;
    unsigned int h;

    if (id == 0)
        id = RT_TABLE_MAIN;
    h = id & (FIB_TABLE_HASHSZ - 1);
+
    rcu_read_lock();
- hlist_for_each_entry_rcu(tb, node, &fib_table_hash[h], tb_hlist) {
+ head = &net->ipv4.fib_table_hash[h];
+ hlist_for_each_entry_rcu(tb, node, head, tb_hlist) {
    if (tb->tb_id == id) {
        rcu_read_unlock();
        return tb;
@@ -117,15 +119,17 @@ struct fib_table *fib_get_table(struct net *net, u32 id)
    }
#endif /* CONFIG_IP_MULTIPLE_TABLES */

-static void fib_flush(void)
+static void fib_flush(struct net *net)
{
    int flushed = 0;
    struct fib_table *tb;
    struct hlist_node *node;
+ struct hlist_head *head;
    unsigned int h;

    for (h = 0; h < FIB_TABLE_HASHSZ; h++) {
- hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist)
+ head = &net->ipv4.fib_table_hash[h];
+ hlist_for_each_entry(tb, node, head, tb_hlist)
        flushed += tb->tb_flush(tb);
    }

@@ -620,6 +624,7 @@ static int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
    unsigned int e = 0, s_e;
    struct fib_table *tb;
    struct hlist_node *node;
+ struct hlist_head *head;
    int dumped = 0;

    if (net != &init_net)
@@ -634,7 +639,8 @@ static int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)

    for (h = s_h; h < FIB_TABLE_HASHSZ; h++, s_e = 0) {

```

```

    e = 0;
- hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist) {
+ head = &net->ipv4.fib_table_hash[h];
+ hlist_for_each_entry(tb, node, head, tb_hlist) {
    if (e < s_e)
        goto next;
    if (dumped)
@@ -797,7 +803,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
    for stray nexthop entries, then ignite fib_flush.
    */
    if (fib_sync_down(ifa->ifa_local, NULL, 0))
- fib_flush();
+ fib_flush(&init_net);
}
}
#undef LOCAL_OK
@@ -877,7 +883,7 @@ static void nl_fib_lookup_exit(struct net *net)
static void fib_disable_ip(struct net_device *dev, int force)
{
    if (fib_sync_down(0, dev, force))
- fib_flush();
+ fib_flush(&init_net);
    rt_cache_flush(0);
    arp_ifdown(dev);
}
@@ -958,8 +964,13 @@ static int __net_init ip_fib_net_init(struct net *net)
{
    unsigned int i;

+ net->ipv4.fib_table_hash = kzalloc(
+ sizeof(struct hlist_head)*FIB_TABLE_HASHSZ, GFP_KERNEL);
+ if (net->ipv4.fib_table_hash == NULL)
+ return -ENOMEM;
+
    for (i = 0; i < FIB_TABLE_HASHSZ; i++)
- INIT_HLIST_HEAD(&fib_table_hash[i]);
+ INIT_HLIST_HEAD(&net->ipv4.fib_table_hash[i]);

    return fib4_rules_init(net);
}
@@ -977,13 +988,14 @@ static void __net_exit ip_fib_net_exit(struct net *net)
    struct hlist_head *head;
    struct hlist_node *node, *tmp;

- head = &fib_table_hash[i];
+ head = &net->ipv4.fib_table_hash[i];
    hlist_for_each_entry_safe(tb, node, tmp, head, tb_hlist) {
        hlist_del(node);

```

```
tb->tb_flush(tb);
kfree(tb);
}
}
+ kfree(net->ipv4.fib_table_hash);
}

static int __net_init fib_net_init(struct net *net)
--
1.5.3.rc5
```
