Subject: [PATCH net-next 8/9] netns: place rt_genid into struct net
Posted by den on Fri, 04 Jul 2008 13:17:11 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/netns/ipv4.h |    1 +
 net/ipv4/route.c         |   76 +++++++++++++++++++++++++---------------------
 2 files changed, 44 insertions(+), 33 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index 356617f..a6ed838 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -48,5 +48,6 @@ struct netns_ipv4 {
  int sysctl_icmp_errors_use_inbound_ifaddr;

  struct timer_list rt_secret_timer;
+ atomic_t rt_genid;
 };
 #endif
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index e4e37ed..67c3ed7 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -250,7 +250,6 @@ static inline void rt_hash_lock_init(void)
 static struct rt_hash_bucket  *rt_hash_table __read_mostly;
 static unsigned   rt_hash_mask __read_mostly;
 static unsigned int  rt_hash_log  __read_mostly;
-static atomic_t   rt_genid __read_mostly;

 static DEFINE_PER_CPU(struct rt_cache_stat, rt_cache_stat);
 #define RT_CACHE_STAT_INC(field) \
@@ -265,6 +264,11 @@ static inline unsigned int rt_hash(__be32 daddr, __be32 saddr, int idx,
   & rt_hash_mask;
 }

+static inline int rt_genid(struct net *net)
+{
+ return atomic_read(&net->ipv4.rt_genid);
+}
+
 #ifdef CONFIG_PROC_FS
 struct rt_cache_iter_state {
  struct seq_net_private p;
@@ -334,7 +338,7 @@ static void *rt_cache_seq_start(struct seq_file *seq, loff_t *pos)
  struct rt_cache_iter_state *st = seq->private;
  if (*pos)
```

```
   return rt_cache_get_idx(seq, *pos - 1);
- st->genid = atomic_read(&rt_genid);
+ st->genid = rt_genid(seq_file_net(seq));
  return SEQ_START_TOKEN;
 }

@@ -681,6 +685,11 @@ static inline int compare_netns(struct rtable *rt1, struct rtable *rt2)
  return dev_net(rt1->u.dst.dev) == dev_net(rt2->u.dst.dev);
 }

+static inline int rt_is_expired(struct rtable *rth)
+{
+ return rth->rt_genid != rt_genid(dev_net(rth->u.dst.dev));
+}
+
 /*
  * Perform a full scan of hash table and free all entries.
  * Can be called by a softirq or a process.
@@ -736,7 +745,7 @@ static void rt_check_expire(void)
    continue;
   spin_lock_bh(rt_hash_lock_addr(i));
   while ((rth = *rthp) != NULL) {
-   if (rth->rt_genid != atomic_read(&rt_genid)) {
+   if (rt_is_expired(rth)) {
     *rthp = rth->u.dst.rt_next;
     rt_free(rth);
     continue;
@@ -784,7 +793,7 @@ static void rt_cache_invalidate(struct net *net)
  unsigned char shuffle;

  get_random_bytes(&shuffle, sizeof(shuffle));
- atomic_add(shuffle + 1U, &rt_genid);
+ atomic_add(shuffle + 1U, &net->ipv4.rt_genid);
 }

 /*
@@ -881,7 +890,7 @@ static int rt_garbage_collect(struct dst_ops *ops)
    rthp = &rt_hash_table[k].chain;
    spin_lock_bh(rt_hash_lock_addr(k));
    while ((rth = *rthp) != NULL) {
-    if (rth->rt_genid == atomic_read(&rt_genid) &&
+    if (!rt_is_expired(rth) &&
      !rt_may_expire(rth, tmo, expire)) {
      tmo >>= 1;
      rthp = &rth->u.dst.rt_next;
@@ -963,7 +972,7 @@ restart:

  spin_lock_bh(rt_hash_lock_addr(hash));
```

```
  while ((rth = *rthp) != NULL) {
-  if (rth->rt_genid != atomic_read(&rt_genid)) {
+  if (rt_is_expired(rth)) {
    *rthp = rth->u.dst.rt_next;
    rt_free(rth);
    continue;
@@ -1139,7 +1148,7 @@ static void rt_del(unsigned hash, struct rtable *rt)
  spin_lock_bh(rt_hash_lock_addr(hash));
  ip_rt_put(rt);
  while ((aux = *rthp) != NULL) {
-  if (aux == rt || (aux->rt_genid != atomic_read(&rt_genid))) {
+  if (aux == rt || rt_is_expired(aux)) {
    *rthp = aux->u.dst.rt_next;
    rt_free(aux);
    continue;
@@ -1182,7 +1191,7 @@ void ip_rt_redirect(__be32 old_gw, __be32 daddr, __be32 new_gw,
  for (i = 0; i < 2; i++) {
   for (k = 0; k < 2; k++) {
    unsigned hash = rt_hash(daddr, skeys[i], ikeys[k],
-       atomic_read(&rt_genid));
+       rt_genid(net));

    rthp=&rt_hash_table[hash].chain;

@@ -1194,7 +1203,7 @@ void ip_rt_redirect(__be32 old_gw, __be32 daddr, __be32 new_gw,
         rth->fl.fl4_src != skeys[i] ||
         rth->fl.oif != ikeys[k] ||
         rth->fl.iif != 0 ||
-        rth->rt_genid != atomic_read(&rt_genid) ||
+        rt_is_expired(rth) ||
         !net_eq(dev_net(rth->u.dst.dev), net)) {
      rthp = &rth->u.dst.rt_next;
      continue;
@@ -1233,7 +1242,7 @@ void ip_rt_redirect(__be32 old_gw, __be32 daddr, __be32 new_gw,
     rt->u.dst.neighbour = NULL;
     rt->u.dst.hh  = NULL;
     rt->u.dst.xfrm  = NULL;
-    rt->rt_genid  = atomic_read(&rt_genid);
+    rt->rt_genid  = rt_genid(net);
     rt->rt_flags  |= RTCF_REDIRECTED;

     /* Gateway is different ... */
@@ -1298,7 +1307,7 @@ static struct dst_entry *ipv4_negative_advice(struct dst_entry *dst)
       rt->u.dst.expires) {
    unsigned hash = rt_hash(rt->fl.fl4_dst, rt->fl.fl4_src,
       rt->fl.oif,
-       atomic_read(&rt_genid));
+       rt_genid(dev_net(dst->dev)));
```

```
 #if RT_CACHE_DEBUG >= 1
    printk(KERN_DEBUG "ipv4_negative_advice: redirect to "
        NIPQUAD_FMT "/%02x dropped\n",
@@ -1448,7 +1457,7 @@ unsigned short ip_rt_frag_needed(struct net *net, struct iphdr *iph,
  for (k = 0; k < 2; k++) {
   for (i = 0; i < 2; i++) {
    unsigned hash = rt_hash(daddr, skeys[i], ikeys[k],
-       atomic_read(&rt_genid));
+       rt_genid(net));

    rcu_read_lock();
    for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
@@ -1463,7 +1472,7 @@ unsigned short ip_rt_frag_needed(struct net *net, struct iphdr *iph,
         rth->fl.iif != 0 ||
         dst_metric_locked(&rth->u.dst, RTAX_MTU) ||
         !net_eq(dev_net(rth->u.dst.dev), net) ||
-        rth->rt_genid != atomic_read(&rt_genid))
+        !rt_is_expired(rth))
      continue;

     if (new_mtu < 68 || new_mtu >= old_mtu) {
@@ -1698,7 +1707,7 @@ static int ip_route_input_mc(struct sk_buff *skb, __be32 daddr,
__be32 saddr,
  rth->fl.oif = 0;
  rth->rt_gateway = daddr;
  rth->rt_spec_dst= spec_dst;
- rth->rt_genid = atomic_read(&rt_genid);
+ rth->rt_genid = rt_genid(dev_net(dev));
  rth->rt_flags = RTCF_MULTICAST;
  rth->rt_type = RTN_MULTICAST;
  if (our) {
@@ -1713,7 +1722,7 @@ static int ip_route_input_mc(struct sk_buff *skb, __be32 daddr,
__be32 saddr,
  RT_CACHE_STAT_INC(in_slow_mc);

  in_dev_put(in_dev);
- hash = rt_hash(daddr, saddr, dev->ifindex, atomic_read(&rt_genid));
+ hash = rt_hash(daddr, saddr, dev->ifindex, rt_genid(dev_net(dev)));
  return rt_intern_hash(hash, rth, &skb->rtable);

 e_nobufs:
@@ -1839,7 +1848,7 @@ static int __mkroute_input(struct sk_buff *skb,

  rth->u.dst.input = ip_forward;
  rth->u.dst.output = ip_output;
- rth->rt_genid = atomic_read(&rt_genid);
+ rth->rt_genid = rt_genid(dev_net(rth->u.dst.dev));
```

```
  rt_set_nexthop(rth, res, itag);

@@ -1874,7 +1883,8 @@ static int ip_mkroute_input(struct sk_buff *skb,
   return err;

  /* put it into the cache */
- hash = rt_hash(daddr, saddr, fl->iif, atomic_read(&rt_genid));
+ hash = rt_hash(daddr, saddr, fl->iif,
+         rt_genid(dev_net(rth->u.dst.dev)));
  return rt_intern_hash(hash, rth, &skb->rtable);
 }

@@ -2000,7 +2010,7 @@ local_input:
   goto e_nobufs;

  rth->u.dst.output= ip_rt_bug;
- rth->rt_genid = atomic_read(&rt_genid);
+ rth->rt_genid = rt_genid(net);

  atomic_set(&rth->u.dst.__refcnt, 1);
  rth->u.dst.flags= DST_HOST;
@@ -2030,7 +2040,7 @@ local_input:
   rth->rt_flags  &= ~RTCF_LOCAL;
  }
  rth->rt_type = res.type;
- hash = rt_hash(daddr, saddr, fl.iif, atomic_read(&rt_genid));
+ hash = rt_hash(daddr, saddr, fl.iif, rt_genid(net));
  err = rt_intern_hash(hash, rth, &skb->rtable);
  goto done;

@@ -2081,7 +2091,7 @@ int ip_route_input(struct sk_buff *skb, __be32 daddr, __be32 saddr,

  net = dev_net(dev);
  tos &= IPTOS_RT_MASK;
- hash = rt_hash(daddr, saddr, iif, atomic_read(&rt_genid));
+ hash = rt_hash(daddr, saddr, iif, rt_genid(net));

  rcu_read_lock();
  for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
@@ -2093,7 +2103,7 @@ int ip_route_input(struct sk_buff *skb, __be32 daddr, __be32 saddr,
        (rth->fl.fl4_tos ^ tos)) == 0 &&
       rth->fl.mark == skb->mark &&
       net_eq(dev_net(rth->u.dst.dev), net) &&
-      rth->rt_genid == atomic_read(&rt_genid)) {
+      !rt_is_expired(rth)) {
    dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(in_hit);
    rcu_read_unlock();
```

```
@@ -2221,7 +2231,7 @@ static int __mkroute_output(struct rtable **result,
  rth->rt_spec_dst= fl->fl4_src;

  rth->u.dst.output=ip_output;
- rth->rt_genid = atomic_read(&rt_genid);
+ rth->rt_genid = rt_genid(dev_net(dev_out));

  RT_CACHE_STAT_INC(out_slow_tot);

@@ -2271,7 +2281,7 @@ static int ip_mkroute_output(struct rtable **rp,
  unsigned hash;
  if (err == 0) {
   hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif,
-          atomic_read(&rt_genid));
+          rt_genid(dev_net(dev_out)));
   err = rt_intern_hash(hash, rth, rp);
  }

@@ -2483,8 +2493,7 @@ int __ip_route_output_key(struct net *net, struct rtable **rp,
  unsigned hash;
  struct rtable *rth;

- hash = rt_hash(flp->fl4_dst, flp->fl4_src, flp->oif,
-        atomic_read(&rt_genid));
+ hash = rt_hash(flp->fl4_dst, flp->fl4_src, flp->oif, rt_genid(net));

  rcu_read_lock_bh();
  for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
@@ -2497,7 +2506,7 @@ int __ip_route_output_key(struct net *net, struct rtable **rp,
       !((rth->fl.fl4_tos ^ flp->fl4_tos) &
         (IPTOS_RT_MASK | RTO_ONLINK)) &&
       net_eq(dev_net(rth->u.dst.dev), net) &&
-      rth->rt_genid == atomic_read(&rt_genid)) {
+      !rt_is_expired(rth)) {
    dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(out_hit);
    rcu_read_unlock_bh();
@@ -2528,7 +2537,7 @@ static struct dst_ops ipv4_dst_blackhole_ops = {
 };


-static int ipv4_dst_blackhole(struct rtable **rp, struct flowi *flp)
+static int ipv4_dst_blackhole(struct net *net, struct rtable **rp, struct flowi *flp)
 {
  struct rtable *ort = *rp;
  struct rtable *rt = (struct rtable *)
@@ -2552,7 +2561,7 @@ static int ipv4_dst_blackhole(struct rtable **rp, struct flowi *flp)
   rt->idev = ort->idev;
```

```
   if (rt->idev)
    in_dev_hold(rt->idev);
-  rt->rt_genid = atomic_read(&rt_genid);
+  rt->rt_genid = rt_genid(net);
   rt->rt_flags = ort->rt_flags;
   rt->rt_type = ort->rt_type;
   rt->rt_dst = ort->rt_dst;
@@ -2588,7 +2597,7 @@ int ip_route_output_flow(struct net *net, struct rtable **rp, struct flowi
*flp,
   err = __xfrm_lookup((struct dst_entry **)rp, flp, sk,
        flags ? XFRM_LOOKUP_WAIT : 0);
   if (err == -EREMOTE)
-   err = ipv4_dst_blackhole(rp, flp);
+   err = ipv4_dst_blackhole(net, rp, flp);

   return err;
  }
@@ -2807,7 +2816,7 @@ int ip_rt_dump(struct sk_buff *skb,  struct netlink_callback *cb)
        rt = rcu_dereference(rt->u.dst.rt_next), idx++) {
    if (!net_eq(dev_net(rt->u.dst.dev), net) || idx < s_idx)
     continue;
-   if (rt->rt_genid != atomic_read(&rt_genid))
+   if (rt_is_expired(rt))
     continue;
    skb->dst = dst_clone(&rt->u.dst);
    if (rt_fill_info(skb, NETLINK_CB(cb->skb).pid,
@@ -3081,6 +3090,10 @@ static __net_initdata struct pernet_operations sysctl_route_ops = {

 static __net_init int rt_secret_timer_init(struct net *net)
 {
+ atomic_set(&net->ipv4.rt_genid,
+   (int) ((num_physpages ^ (num_physpages>>8)) ^
+   (jiffies ^ (jiffies >> 7))));
+
  net->ipv4.rt_secret_timer.function = rt_secret_rebuild;
  net->ipv4.rt_secret_timer.data = (unsigned long)net;
  init_timer_deferrable(&net->ipv4.rt_secret_timer);
@@ -3121,9 +3134,6 @@ int __init ip_rt_init(void)
 {
  int rc = 0;

- atomic_set(&rt_genid, (int) ((num_physpages ^ (num_physpages>>8)) ^
-        (jiffies ^ (jiffies >> 7))));
-
 #ifdef CONFIG_NET_CLS_ROUTE
  ip_rt_acct = __alloc_percpu(256 * sizeof(struct ip_rt_acct));
  if (!ip_rt_acct)
--
```

1.5.3.rc5

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers