
Subject: [PATCH net-next 7/9] ipv4: pass current value of rt_genid into rt_hash
Posted by [den](#) on Fri, 04 Jul 2008 13:17:10 GMT

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

Basically, there is no difference to atomic_read internally or pass it as a parameter as rt_hash is inline.

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

net/ipv4/route.c | 28 ++++++-----
1 files changed, 17 insertions(+), 11 deletions(-)

diff --git a/net/ipv4/route.c b/net/ipv4/route.c

index 9725223..e4e37ed 100644

--- a/net/ipv4/route.c

+++ b/net/ipv4/route.c

@@ -256,11 +256,12 @@ static DEFINE_PER_CPU(struct rt_cache_stat, rt_cache_stat);

#define RT_CACHE_STAT_INC(field) \

(__raw_get_cpu_var(rt_cache_stat).field++)

-static inline unsigned int rt_hash(__be32 daddr, __be32 saddr, int idx)

+static inline unsigned int rt_hash(__be32 daddr, __be32 saddr, int idx,

+ int genid)

{

return jhash_3words((__force u32)(__be32)(daddr),
(__force u32)(__be32)(saddr),

- idx, atomic_read(&rt_genid))

+ idx, genid)

& rt_hash_mask;

}

@@ -1180,7 +1181,8 @@ 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]);

+ unsigned hash = rt_hash(daddr, skeys[i], ikeys[k],

+ atomic_read(&rt_genid));

rthp=&rt_hash_table[hash].chain;

@@ -1295,7 +1297,8 @@ static struct dst_entry *ipv4_negative_advice(struct dst_entry *dst)

} else if ((rt->rt_flags & RTCF_REDIRECTED) ||

rt->u.dst.expires) {

unsigned hash = rt_hash(rt->fl.fl4_dst, rt->fl.fl4_src,

- rt->fl.oif);

+ rt->fl.oif,

+ atomic_read(&rt_genid));

```

#if RT_CACHE_DEBUG >= 1
    printk(KERN_DEBUG "ipv4_negative_advice: redirect to "
        NIPQUAD_FMT "/%02x dropped\n",
@@ -1444,7 +1447,8 @@ 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]);
+         unsigned hash = rt_hash(daddr, skeys[i], ikeys[k],
+         atomic_read(&rt_genid));

        rcu_read_lock();
        for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
@@ -1709,7 +1713,7 @@ 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);
+ hash = rt_hash(daddr, saddr, dev->ifindex, atomic_read(&rt_genid));
    return rt_intern_hash(hash, rth, &skb->rtable);

e_nobufs:
@@ -1870,7 +1874,7 @@ static int ip_mkroute_input(struct sk_buff *skb,
    return err;

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

@@ -2026,7 +2030,7 @@ local_input:
    rth->rt_flags &= ~RTCF_LOCAL;
}
    rth->rt_type = res.type;
- hash = rt_hash(daddr, saddr, fl.iif);
+ hash = rt_hash(daddr, saddr, fl.iif, atomic_read(&rt_genid));
    err = rt_intern_hash(hash, rth, &skb->rtable);
    goto done;

@@ -2077,7 +2081,7 @@ 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);
+ hash = rt_hash(daddr, saddr, iif, atomic_read(&rt_genid));

```

```

rcu_read_lock();
for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
@@ -2266,7 +2270,8 @@ static int ip_mkroute_output(struct rtable **rp,
int err = __mkroute_output(&rth, res, fl, oldflp, dev_out, flags);
unsigned hash;
if (err == 0) {
- hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);
+ hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif,
+ atomic_read(&rt_genid));
err = rt_intern_hash(hash, rth, rp);
}

@@ -2478,7 +2483,8 @@ 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);
+ hash = rt_hash(flp->fl4_dst, flp->fl4_src, flp->oif,
+ atomic_read(&rt_genid));

rcu_read_lock_bh();
for (rth = rcu_dereference(rt_hash_table[hash].chain); rth;
--
1.5.3.rc5

```

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