Subject: [PATCH net-2.6.25 4/10][NETNS][FRAGS]: Make the mem counter per-namespace.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:59:41 GMT
View Forum Message <> Reply to Message

This is also simple, but introduces more changes, since
then mem counter is altered in more places.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
---
 include/net/inet_frag.h                 |    4 ++--
 include/net/ip.h                        |    2 +-
 include/net/ipv6.h                      |    2 +-
 net/ipv4/inet_fragment.c                |   21 +++++++++++----------
 net/ipv4/ip_fragment.c                  |   29 +++++++++++++++--------------
 net/ipv4/proc.c                         |    2 +-
 net/ipv6/netfilter/nf_conntrack_reasm.c |   14 +++++++-------
 net/ipv6/proc.c                         |    2 +-
 net/ipv6/reassembly.c                   |   28 ++++++++++++++-------------
 9 files changed, 54 insertions(+), 50 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index d36f3a6..6edce7b 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -3,6 +3,7 @@

 struct netns_frags {
  int   nqueues;
+ atomic_t  mem;
 };

 struct inet_frag_queue {
@@ -38,7 +39,6 @@ struct inet_frags {
  rwlock_t  lock;
  u32   rnd;
  int   qsize;
- atomic_t  mem;
  struct timer_list secret_timer;
  struct inet_frags_ctl *ctl;

@@ -60,7 +60,7 @@ void inet_frags_init_net(struct netns_frags *nf);
 void inet_frag_kill(struct inet_frag_queue *q, struct inet_frags *f);
 void inet_frag_destroy(struct inet_frag_queue *q,
     struct inet_frags *f, int *work);
-int inet_frag_evictor(struct inet_frags *f);
+int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f);
```

```
 struct inet_frag_queue *inet_frag_find(struct netns_frags *nf,
   struct inet_frags *f, void *key, unsigned int hash);

diff --git a/include/net/ip.h b/include/net/ip.h
index 9ea1bc5..d41ff83 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -329,7 +329,7 @@ enum ip_defrag_users
 };

 int ip_defrag(struct sk_buff *skb, u32 user);
-int ip_frag_mem(void);
+int ip_frag_mem(struct net *net);
 int ip_frag_nqueues(struct net *net);

 /*
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index da1c089..fa80ea4 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -246,7 +246,7 @@ struct ipv6_txoptions *ipv6_fixup_options(struct ipv6_txoptions
*opt_space,
 extern int ipv6_opt_accepted(struct sock *sk, struct sk_buff *skb);

 int ip6_frag_nqueues(struct net *net);
-int ip6_frag_mem(void);
+int ip6_frag_mem(struct net *net);

 #define IPV6_FRAG_TIMEOUT (60*HZ)  /* 60 seconds */

diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 4fec0b9..ad79ae0 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -63,8 +63,6 @@ void inet_frags_init(struct inet_frags *f)
  f->rnd = (u32) ((num_physpages ^ (num_physpages>>7)) ^
        (jiffies ^ (jiffies >> 6)));

- atomic_set(&f->mem, 0);
-
  setup_timer(&f->secret_timer, inet_frag_secret_rebuild,
    (unsigned long)f);
  f->secret_timer.expires = jiffies + f->ctl->secret_interval;
@@ -75,6 +73,7 @@ EXPORT_SYMBOL(inet_frags_init);
 void inet_frags_init_net(struct netns_frags *nf)
 {
  nf->nqueues = 0;
+ atomic_set(&nf->mem, 0);
```

```
 }
 EXPORT_SYMBOL(inet_frags_init_net);

@@ -107,13 +106,13 @@ void inet_frag_kill(struct inet_frag_queue *fq, struct inet_frags *f)

 EXPORT_SYMBOL(inet_frag_kill);

-static inline void frag_kfree_skb(struct inet_frags *f, struct sk_buff *skb,
-      int *work)
+static inline void frag_kfree_skb(struct netns_frags *nf, struct inet_frags *f,
+  struct sk_buff *skb, int *work)
 {
  if (work)
   *work -= skb->truesize;

- atomic_sub(skb->truesize, &f->mem);
+ atomic_sub(skb->truesize, &nf->mem);
  if (f->skb_free)
   f->skb_free(skb);
  kfree_skb(skb);
@@ -123,22 +122,24 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,
      int *work)
 {
  struct sk_buff *fp;
+ struct netns_frags *nf;

  BUG_TRAP(q->last_in & COMPLETE);
  BUG_TRAP(del_timer(&q->timer) == 0);

  /* Release all fragment data. */
  fp = q->fragments;
+ nf = q->net;
  while (fp) {
   struct sk_buff *xp = fp->next;

-  frag_kfree_skb(f, fp, work);
+  frag_kfree_skb(nf, f, fp, work);
   fp = xp;
  }

  if (work)
   *work -= f->qsize;
- atomic_sub(f->qsize, &f->mem);
+ atomic_sub(f->qsize, &nf->mem);

  if (f->destructor)
   f->destructor(q);
@@ -147,12 +148,12 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,
```

```
 }
 EXPORT_SYMBOL(inet_frag_destroy);

-int inet_frag_evictor(struct inet_frags *f)
+int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f)
 {
  struct inet_frag_queue *q;
  int work, evicted = 0;

- work = atomic_read(&f->mem) - f->ctl->low_thresh;
+ work = atomic_read(&nf->mem) - f->ctl->low_thresh;
  while (work > 0) {
   read_lock(&f->lock);
   if (list_empty(&f->lru_list)) {
@@ -226,7 +227,7 @@ static struct inet_frag_queue *inet_frag_alloc(struct netns_frags *nf,
   return NULL;

  f->constructor(q, arg);
- atomic_add(f->qsize, &f->mem);
+ atomic_add(f->qsize, &nf->mem);
  setup_timer(&q->timer, f->frag_expire, (unsigned long)q);
  spin_lock_init(&q->lock);
  atomic_set(&q->refcnt, 1);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index cd8c830..4f01334 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -100,9 +100,9 @@ int ip_frag_nqueues(struct net *net)
  return net->ipv4.frags.nqueues;
 }

-int ip_frag_mem(void)
+int ip_frag_mem(struct net *net)
 {
- return atomic_read(&ip4_frags.mem);
+ return atomic_read(&net->ipv4.frags.mem);
 }

 static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
@@ -142,11 +142,12 @@ static int ip4_frag_match(struct inet_frag_queue *q, void *a)
 }

 /* Memory Tracking Functions. */
-static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
+static __inline__ void frag_kfree_skb(struct netns_frags *nf,
+  struct sk_buff *skb, int *work)
 {
  if (work)
```

```
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &ip4_frags.mem);
+ atomic_sub(skb->truesize, &nf->mem);
  kfree_skb(skb);
 }

@@ -192,11 +193,11 @@ static void ipq_kill(struct ipq *ipq)
 /* Memory limiting on fragments.  Evictor trashes the oldest
  * fragment queue until we are back under the threshold.
  */
-static void ip_evictor(void)
+static void ip_evictor(struct net *net)
 {
  int evicted;

- evicted = inet_frag_evictor(&ip4_frags);
+ evicted = inet_frag_evictor(&net->ipv4.frags, &ip4_frags);
  if (evicted)
   IP_ADD_STATS_BH(IPSTATS_MIB_REASMFAILS, evicted);
 }
@@ -294,7 +295,7 @@ static int ip_frag_reinit(struct ipq *qp)
  fp = qp->q.fragments;
  do {
   struct sk_buff *xp = fp->next;
-  frag_kfree_skb(fp, NULL);
+  frag_kfree_skb(qp->q.net, fp, NULL);
   fp = xp;
  } while (fp);

@@ -431,7 +432,7 @@ static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb)
     qp->q.fragments = next;

    qp->q.meat -= free_it->len;
-   frag_kfree_skb(free_it, NULL);
+   frag_kfree_skb(qp->q.net, free_it, NULL);
   }
  }

@@ -451,7 +452,7 @@ static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb)
  }
  qp->q.stamp = skb->tstamp;
  qp->q.meat += skb->len;
- atomic_add(skb->truesize, &ip4_frags.mem);
+ atomic_add(skb->truesize, &qp->q.net->mem);
  if (offset == 0)
   qp->q.last_in |= FIRST_IN;

@@ -534,12 +535,12 @@ static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
```

```
   head->len -= clone->len;
   clone->csum = 0;
   clone->ip_summed = head->ip_summed;
-  atomic_add(clone->truesize, &ip4_frags.mem);
+  atomic_add(clone->truesize, &qp->q.net->mem);
  }

  skb_shinfo(head)->frag_list = head->next;
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &ip4_frags.mem);
+ atomic_sub(head->truesize, &qp->q.net->mem);

  for (fp=head->next; fp; fp = fp->next) {
   head->data_len += fp->len;
@@ -549,7 +550,7 @@ static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
   else if (head->ip_summed == CHECKSUM_COMPLETE)
    head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &ip4_frags.mem);
+  atomic_sub(fp->truesize, &qp->q.net->mem);
  }

  head->next = NULL;
@@ -588,8 +589,8 @@ int ip_defrag(struct sk_buff *skb, u32 user)

  net = skb->dev->nd_net;
  /* Start by cleaning up the memory. */
- if (atomic_read(&ip4_frags.mem) > ip4_frags_ctl.high_thresh)
-  ip_evictor();
+ if (atomic_read(&net->ipv4.frags.mem) > ip4_frags_ctl.high_thresh)
+  ip_evictor(net);

  /* Lookup (or create) queue header */
  if ((qp = ip_find(net, ip_hdr(skb), user)) != NULL) {
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index bae3280..d63474c 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -62,7 +62,7 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "UDPLITE: inuse %d\n", sock_prot_inuse_get(&udplite_prot));
  seq_printf(seq, "RAW: inuse %d\n", sock_prot_inuse_get(&raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
-   ip_frag_nqueues(&init_net), ip_frag_mem());
+   ip_frag_nqueues(&init_net), ip_frag_mem(&init_net));
  return 0;
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
```

```
index 0b9d009..cb826be 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -155,7 +155,7 @@ static inline void frag_kfree_skb(struct sk_buff *skb, unsigned int *work)
 {
  if (work)
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &nf_frags.mem);
+ atomic_sub(skb->truesize, &nf_init_frags.mem);
  nf_skb_free(skb);
  kfree_skb(skb);
 }
@@ -177,7 +177,7 @@ static __inline__ void fq_kill(struct nf_ct_frag6_queue *fq)

 static void nf_ct_frag6_evictor(void)
 {
- inet_frag_evictor(&nf_frags);
+ inet_frag_evictor(&nf_init_frags, &nf_frags);
 }

 static void nf_ct_frag6_expire(unsigned long data)
@@ -382,7 +382,7 @@ static int nf_ct_frag6_queue(struct nf_ct_frag6_queue *fq, struct sk_buff
*skb,
  skb->dev = NULL;
  fq->q.stamp = skb->tstamp;
  fq->q.meat += skb->len;
- atomic_add(skb->truesize, &nf_frags.mem);
+ atomic_add(skb->truesize, &nf_init_frags.mem);

  /* The first fragment.
   * nhoffset is obtained from the first fragment, of course.
@@ -459,7 +459,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
   clone->ip_summed = head->ip_summed;

   NFCT_FRAG6_CB(clone)->orig = NULL;
-  atomic_add(clone->truesize, &nf_frags.mem);
+  atomic_add(clone->truesize, &nf_init_frags.mem);
  }

  /* We have to remove fragment header from datagram and to relocate
@@ -473,7 +473,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
  skb_shinfo(head)->frag_list = head->next;
  skb_reset_transport_header(head);
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &nf_frags.mem);
+ atomic_sub(head->truesize, &nf_init_frags.mem);

  for (fp=head->next; fp; fp = fp->next) {
```

```
   head->data_len += fp->len;
@@ -483,7 +483,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
   else if (head->ip_summed == CHECKSUM_COMPLETE)
    head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &nf_frags.mem);
+  atomic_sub(fp->truesize, &nf_init_frags.mem);
  }

  head->next = NULL;
@@ -633,7 +633,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
   goto ret_orig;
  }

- if (atomic_read(&nf_frags.mem) > nf_frags_ctl.high_thresh)
+ if (atomic_read(&nf_init_frags.mem) > nf_frags_ctl.high_thresh)
   nf_ct_frag6_evictor();

  fq = fq_find(fhdr->identification, &hdr->saddr, &hdr->daddr);
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 0b55785..c51cf34 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -44,7 +44,7 @@ static int sockstat6_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "RAW6: inuse %d\n",
         sock_prot_inuse_get(&rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
-         ip6_frag_nqueues(&init_net), ip6_frag_mem());
+         ip6_frag_nqueues(&init_net), ip6_frag_mem(&init_net));
  return 0;
 }

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 77a8740..241b2cc 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -89,9 +89,9 @@ int ip6_frag_nqueues(struct net *net)
  return net->ipv6.frags.nqueues;
 }

-int ip6_frag_mem(void)
+int ip6_frag_mem(struct net *net)
 {
- return atomic_read(&ip6_frags.mem);
+ return atomic_read(&net->ipv6.frags.mem);
 }

 static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
```

```
@@ -149,11 +149,12 @@ int ip6_frag_match(struct inet_frag_queue *q, void *a)
 EXPORT_SYMBOL(ip6_frag_match);

 /* Memory Tracking Functions. */
-static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
+static inline void frag_kfree_skb(struct netns_frags *nf,
+  struct sk_buff *skb, int *work)
 {
  if (work)
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &ip6_frags.mem);
+ atomic_sub(skb->truesize, &nf->mem);
  kfree_skb(skb);
 }

@@ -183,11 +184,11 @@ static __inline__ void fq_kill(struct frag_queue *fq)
  inet_frag_kill(&fq->q, &ip6_frags);
 }

-static void ip6_evictor(struct inet6_dev *idev)
+static void ip6_evictor(struct net *net, struct inet6_dev *idev)
 {
  int evicted;

- evicted = inet_frag_evictor(&ip6_frags);
+ evicted = inet_frag_evictor(&net->ipv6.frags, &ip6_frags);
  if (evicted)
   IP6_ADD_STATS_BH(idev, IPSTATS_MIB_REASMFAILS, evicted);
 }
@@ -389,7 +390,7 @@ static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
     fq->q.fragments = next;

    fq->q.meat -= free_it->len;
-   frag_kfree_skb(free_it, NULL);
+   frag_kfree_skb(fq->q.net, free_it, NULL);
   }
  }

@@ -409,7 +410,7 @@ static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
  }
  fq->q.stamp = skb->tstamp;
  fq->q.meat += skb->len;
- atomic_add(skb->truesize, &ip6_frags.mem);
+ atomic_add(skb->truesize, &fq->q.net->mem);

  /* The first fragment.
   * nhoffset is obtained from the first fragment, of course.
@@ -503,7 +504,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
```

```
   head->len -= clone->len;
   clone->csum = 0;
   clone->ip_summed = head->ip_summed;
-  atomic_add(clone->truesize, &ip6_frags.mem);
+  atomic_add(clone->truesize, &fq->q.net->mem);
  }

  /* We have to remove fragment header from datagram and to relocate
@@ -518,7 +519,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
  skb_shinfo(head)->frag_list = head->next;
  skb_reset_transport_header(head);
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &ip6_frags.mem);
+ atomic_sub(head->truesize, &fq->q.net->mem);

  for (fp=head->next; fp; fp = fp->next) {
   head->data_len += fp->len;
@@ -528,7 +529,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
    else if (head->ip_summed == CHECKSUM_COMPLETE)
     head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &ip6_frags.mem);
+  atomic_sub(fp->truesize, &fq->q.net->mem);
  }

  head->next = NULL;
@@ -600,8 +601,9 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
  }

  net = skb->dev->nd_net;
- if (atomic_read(&ip6_frags.mem) > init_net.ipv6.sysctl.frags.high_thresh)
-  ip6_evictor(ip6_dst_idev(skb->dst));
+ if (atomic_read(&net->ipv6.frags.mem) >
+    init_net.ipv6.sysctl.frags.high_thresh)
+  ip6_evictor(net, ip6_dst_idev(skb->dst));

  if ((fq = fq_find(net, fhdr->identification, &hdr->saddr, &hdr->daddr,
      ip6_dst_idev(skb->dst))) != NULL) {
--
1.5.3.4
```