Subject: [PATCH net-2.6.26 0/6][NETNS][SOCK]: Make "prot inuse" counters work per-net.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:04:27 GMT
View Forum Message <> Reply to Message

These counters show the number of sockets used for each protocol,
but currently they gather global statistics. This is useful to
have them work per-net.

This set adds this possibility, but hides it under CONFIG_NET_NS
option, so that not-virtualized kernel keeps using the optimized
version by Eric (and he's in Cc for that reason).

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

Subject: [PATCH net-2.6.26 1/6][NETNS][SOCK]: Add net parameter to sock_prot_inuse_(add|get).
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:07:00 GMT
View Forum Message <> Reply to Message

It will be used in netnsizated versions of these calls only.
The net passed to then is either get from a sock, or is available
int place, with two exceptions - the "sockstat" and "sockstat6"
proc files use init_net by now.

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

---
 include/net/sock.h | 8 +++++---
 include/net/udp.h | 2 +-
 net/ipv4/inet_hashtables.c | 8 ++++----
 net/ipv4/inet_timewait_sock.c | 2 +-
 net/ipv4/proc.c | 11 +++++++----
 net/ipv4/raw.c | 4 ++--
 net/ipv4/udp.c | 2 +-
 net/ipv6/inet6_hashtables.c | 4 ++--
 net/ipv6/ipv6_sockglue.c | 8 ++++----
 net/ipv6/proc.c | 8 ++++----
 10 files changed, 31 insertions(+), 26 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index 1c9d059..a57c58f 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -638,7 +638,8 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)
 # define DEFINE_PROTO_INUSE(NAME) DEFINE_PCOUNTER(NAME)
 # define REF_PROTO_INUSE(NAME) PCOUNTER_MEMBER_INITIALIZER(NAME, .inuse)

```
 /* Called with local bh disabled */
-static inline void sock_prot_inuse_add(struct proto *prot, int inc)
+static inline void sock_prot_inuse_add(struct net *net,
+  struct proto *prot, int inc)
 {
  pcounter_add(&prot->inuse, inc);
 }
@@ -646,7 +647,7 @@ static inline int sock_prot_inuse_init(struct proto *proto)
 {
  return pcounter_alloc(&proto->inuse);
 }
-static inline int sock_prot_inuse_get(struct proto *proto)
+static inline int sock_prot_inuse_get(struct net *net, struct proto *proto)
 {
  return pcounter_getval(&proto->inuse);
 }
@@ -657,7 +658,8 @@ static inline void sock_prot_inuse_free(struct proto *proto)
 #else
 # define DEFINE_PROTO_INUSE(NAME)
 # define REF_PROTO_INUSE(NAME)
-static void inline sock_prot_inuse_add(struct proto *prot, int inc)
+static void inline sock_prot_inuse_add(struct net *net,
+  struct proto *prot, int inc)
 {
 }
 static int inline sock_prot_inuse_init(struct proto *proto)
diff --git a/include/net/udp.h b/include/net/udp.h
index 635940d..fec52c1 100644
--- a/include/net/udp.h
+++ b/include/net/udp.h
@@ -115,7 +115,7 @@ static inline void udp_lib_unhash(struct sock *sk)
  write_lock_bh(&udp_hash_lock);
  if (sk_del_node_init(sk)) {
   inet_sk(sk)->num = 0;
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  }
  write_unlock_bh(&udp_hash_lock);
 }
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index 1b6ff51..32ca2f8 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -288,7 +288,7 @@ unique:
  sk->sk_hash = hash;
  BUG_TRAP(sk_unhashed(sk));
  __sk_add_node(sk, &head->chain);
- sock_prot_inuse_add(sk->sk_prot, 1);
```

```
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);

  if (twp) {
@@ -332,7 +332,7 @@ void __inet_hash_nolisten(struct sock *sk)

  write_lock(lock);
  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
 }
 EXPORT_SYMBOL_GPL(__inet_hash_nolisten);
@@ -354,7 +354,7 @@ static void __inet_hash(struct sock *sk)

  inet_listen_wlock(hashinfo);
  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
  wake_up(&hashinfo->lhash_wait);
 }
@@ -387,7 +387,7 @@ void inet_unhash(struct sock *sk)
  }

  if (__sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  write_unlock_bh(lock);
 out:
  if (sk->sk_state == TCP_LISTEN)
diff --git a/net/ipv4/inet_timewait_sock.c b/net/ipv4/inet_timewait_sock.c
index f12bc24..a741378 100644
--- a/net/ipv4/inet_timewait_sock.c
+++ b/net/ipv4/inet_timewait_sock.c
@@ -91,7 +91,7 @@ void __inet_twsk_hashdance(struct inet_timewait_sock *tw, struct sock *sk,

  /* Step 2: Remove SK from established hash. */
  if (__sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);

  /* Step 3: Hash TW into TIMEWAIT chain. */
  inet_twsk_add_node(tw, &ehead->twchain);
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index d63474c..8156c26 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
```

```
@@ -53,14 +53,17 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
 {
  socket_seq_show(seq);
  seq_printf(seq, "TCP: inuse %d orphan %d tw %d alloc %d mem %d\n",
-     sock_prot_inuse_get(&tcp_prot),
+     sock_prot_inuse_get(&init_net, &tcp_prot),
      atomic_read(&tcp_orphan_count),
      tcp_death_row.tw_count, atomic_read(&tcp_sockets_allocated),
      atomic_read(&tcp_memory_allocated));
- seq_printf(seq, "UDP: inuse %d mem %d\n", sock_prot_inuse_get(&udp_prot),
+ seq_printf(seq, "UDP: inuse %d mem %d\n",
+     sock_prot_inuse_get(&init_net, &udp_prot),
      atomic_read(&udp_memory_allocated));
- 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, "UDPLITE: inuse %d\n",
+     sock_prot_inuse_get(&init_net, &udplite_prot));
+ seq_printf(seq, "RAW: inuse %d\n",
+     sock_prot_inuse_get(&init_net, &raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
    ip_frag_nqueues(&init_net), ip_frag_mem(&init_net));
  return 0;
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index d965f0a..247a88d 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -93,7 +93,7 @@ void raw_hash_sk(struct sock *sk)

  write_lock_bh(&h->lock);
  sk_add_node(sk, head);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock_bh(&h->lock);
 }
 EXPORT_SYMBOL_GPL(raw_hash_sk);
@@ -104,7 +104,7 @@ void raw_unhash_sk(struct sock *sk)

  write_lock_bh(&h->lock);
  if (sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  write_unlock_bh(&h->lock);
 }
 EXPORT_SYMBOL_GPL(raw_unhash_sk);
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 80007c7..979c95f 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
```

```
@@ -231,7 +231,7 @@ gotit:
  if (sk_unhashed(sk)) {
   head = &udptable[snum & (UDP_HTABLE_SIZE - 1)];
   sk_add_node(sk, head);
-  sock_prot_inuse_add(sk->sk_prot, 1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  }
  error = 0;
 fail:
diff --git a/net/ipv6/inet6_hashtables.c b/net/ipv6/inet6_hashtables.c
index 340c7d4..580014a 100644
--- a/net/ipv6/inet6_hashtables.c
+++ b/net/ipv6/inet6_hashtables.c
@@ -43,7 +43,7 @@ void __inet6_hash(struct sock *sk)
  }

  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
 }
 EXPORT_SYMBOL(__inet6_hash);
@@ -204,7 +204,7 @@ unique:
  BUG_TRAP(sk_unhashed(sk));
  __sk_add_node(sk, &head->chain);
  sk->sk_hash = hash;
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);

  if (twp != NULL) {
diff --git a/net/ipv6/ipv6_sockglue.c b/net/ipv6/ipv6_sockglue.c
index d3d93d7..3ffbfa5 100644
--- a/net/ipv6/ipv6_sockglue.c
+++ b/net/ipv6/ipv6_sockglue.c
@@ -157,8 +157,8 @@ static int do_ipv6_setsockopt(struct sock *sk, int level, int optname,
     struct inet_connection_sock *icsk = inet_csk(sk);

     local_bh_disable();
-    sock_prot_inuse_add(sk->sk_prot, -1);
-    sock_prot_inuse_add(&tcp_prot, 1);
+    sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
+    sock_prot_inuse_add(sock_net(sk), &tcp_prot, 1);
     local_bh_enable();
     sk->sk_prot = &tcp_prot;
     icsk->icsk_af_ops = &ipv4_specific;
@@ -171,8 +171,8 @@ static int do_ipv6_setsockopt(struct sock *sk, int level, int optname,
     if (sk->sk_protocol == IPPROTO_UDPLITE)
```

```
      prot = &udplite_prot;
     local_bh_disable();
-    sock_prot_inuse_add(sk->sk_prot, -1);
-    sock_prot_inuse_add(prot, 1);
+    sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
+    sock_prot_inuse_add(sock_net(sk), prot, 1);
     local_bh_enable();
     sk->sk_prot = prot;
     sk->sk_socket->ops = &inet_dgram_ops;
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 364dc33..4b9d5a9 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -36,13 +36,13 @@ static struct proc_dir_entry *proc_net_devsnmp6;
 static int sockstat6_seq_show(struct seq_file *seq, void *v)
 {
  seq_printf(seq, "TCP6: inuse %d\n",
-        sock_prot_inuse_get(&tcpv6_prot));
+        sock_prot_inuse_get(&init_net, &tcpv6_prot));
  seq_printf(seq, "UDP6: inuse %d\n",
-        sock_prot_inuse_get(&udpv6_prot));
+        sock_prot_inuse_get(&init_net, &udpv6_prot));
  seq_printf(seq, "UDPLITE6: inuse %d\n",
-   sock_prot_inuse_get(&udplitev6_prot));
+   sock_prot_inuse_get(&init_net, &udplitev6_prot));
  seq_printf(seq, "RAW6: inuse %d\n",
-        sock_prot_inuse_get(&rawv6_prot));
+        sock_prot_inuse_get(&init_net, &rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
         ip6_frag_nqueues(&init_net), ip6_frag_mem(&init_net));
  return 0;
--
1.5.3.4
```

Subject: [PATCH net-2.6.26 2/6][NETNS][SOCK]: Introduce per-net inuse counters.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:13:05 GMT
View Forum Message <> Reply to Message

This is probably the most controversial part of the set.

The counters are stored in a per-cpu array on a struct net. To
index in this array the prot->inuse is declared as int and used.

Numbers (indices) to protos are generated with the appropriate
enum. I though about using some existing IPPROTO_XXX numbers for
protocols but they were too large (IPPROTO_RAW is 255) and did
not differ for ipv4 and ipv6 (there's no IP6PROTO_RAW, etc).

The sock_prot_inuse_(add|get) now use the net argument to
get the counter, but this all hides under CONFIG_NET_NS.

The sock_prot_inuse_(init|fini) are no-ops. DEFINE_PROTO_INUSE
is empty and REF_PROTO_INUSE assigns an index to a proto.

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

---
 include/net/net_namespace.h | 3 ++
 include/net/sock.h | 35 ++++++++++++++++++++++++++++++
 net/core/sock.c | 52 ++
 3 files changed, 90 insertions(+), 0 deletions(-)

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index f8f3d1a..8a37be1 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -18,6 +18,7 @@ struct proc_dir_entry;
 struct net_device;
 struct sock;
 struct ctl_table_header;
+struct net_prot_inuse;

 struct net {
  atomic_t count; /* To decided when the network
@@ -50,6 +51,8 @@ struct net {
  struct ctl_table_header *sysctl_core_hdr;
  int   sysctl_somaxconn;

+ struct net_prot_inuse *inuse;
+
  struct netns_packet packet;
  struct netns_unix unx;
  struct netns_ipv4 ipv4;
diff --git a/include/net/sock.h b/include/net/sock.h
index a57c58f..84a672c 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -562,8 +562,12 @@ struct proto {

  /* Keeping track of sockets in use */
 #ifdef CONFIG_PROC_FS
+#ifdef CONFIG_NET_NS
+ unsigned int inuse;
+#else
  struct pcounter inuse;

```
 #endif
+#endif

  /* Memory pressure */
  void   (*enter_memory_pressure)(void);
@@ -635,6 +639,36 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)


 #ifdef CONFIG_PROC_FS
+#ifdef CONFIG_NET_NS
+enum {
+ NET_INUSE_dccp_v4,
+ NET_INUSE_dccp_v6,
+ NET_INUSE_raw,
+ NET_INUSE_tcp,
+ NET_INUSE_udp,
+ NET_INUSE_udplite,
+ NET_INUSE_rawv6,
+ NET_INUSE_tcpv6,
+ NET_INUSE_udpv6,
+ NET_INUSE_udplitev6,
+ NET_INUSE_sctp,
+ NET_INUSE_sctpv6,
+ NET_INUSE_NR,
+};
+
+# define DEFINE_PROTO_INUSE(NAME)
+# define REF_PROTO_INUSE(NAME) .inuse = NET_INUSE_##NAME,
+
+extern void sock_prot_inuse_add(struct net *net, struct proto *prot, int inc);
+static inline int sock_prot_inuse_init(struct proto *proto)
+{
+ return 0;
+}
+extern int sock_prot_inuse_get(struct net *net, struct proto *proto);
+static inline void sock_prot_inuse_free(struct proto *proto)
+{
+}
+#else /* ! CONFIG_NET_NS */
 # define DEFINE_PROTO_INUSE(NAME) DEFINE_PCOUNTER(NAME)
 # define REF_PROTO_INUSE(NAME) PCOUNTER_MEMBER_INITIALIZER(NAME, .inuse)
 /* Called with local bh disabled */
@@ -655,6 +689,7 @@ static inline void sock_prot_inuse_free(struct proto *proto)
 {
  pcounter_free(&proto->inuse);
 }
+#endif /* CONFIG_NET_NS */
 #else
```

```
 # define DEFINE_PROTO_INUSE(NAME)
 # define REF_PROTO_INUSE(NAME)
diff --git a/net/core/sock.c b/net/core/sock.c
index 3ee9506..743f628 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -2056,6 +2056,58 @@ void proto_unregister(struct proto *prot)
 EXPORT_SYMBOL(proto_unregister);

 #ifdef CONFIG_PROC_FS
+#ifdef CONFIG_NET_NS
+struct net_prot_inuse {
+ int val[NET_INUSE_NR];
+};
+
+void sock_prot_inuse_add(struct net *net, struct proto *prot, int val)
+{
+ per_cpu_ptr(net->inuse, get_cpu())->val[prot->inuse] += val;
+ put_cpu();
+}
+EXPORT_SYMBOL_GPL(sock_prot_inuse_add);
+
+int sock_prot_inuse_get(struct net *net, struct proto *prot)
+{
+ int cpu, idx, val;
+
+ idx = prot->inuse;
+ val = 0;
+ for_each_online_cpu(cpu)
+  val += per_cpu_ptr(net->inuse, cpu)->val[idx];
+
+ return val;
+}
+EXPORT_SYMBOL_GPL(sock_prot_inuse_get);
+
+static int sock_inuse_init_net(struct net *net)
+{
+ net->inuse = alloc_percpu(struct net_prot_inuse);
+ return net->inuse ? 0 : -ENOMEM;
+}
+
+static void sock_inuse_exit_net(struct net *net)
+{
+ free_percpu(net->inuse);
+}
+
+static struct pernet_operations net_inuse_ops = {
+ .init = sock_inuse_init_net,
```

```
+ .exit = sock_inuse_exit_net,
+};
+
+static __init int net_inuse_init(void)
+{
+ if (register_pernet_subsys(&net_inuse_ops))
+  panic("Cannot initialize net inuse counters");
+
+ return 0;
+}
+
+core_initcall(net_inuse_init);
+#endif
+
 static void *proto_seq_start(struct seq_file *seq, loff_t *pos)
  __acquires(proto_list_lock)
 {
--
1.5.3.4
```

Subject: [PATCH net-2.6.26 3/6][NETNS][SOCK]: Swap unhash and net change for icmp and tcp service sockets.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:16:32 GMT
View Forum Message <> Reply to Message

This is tricky.

The problem is that these sockets are created in init_net not to pin
the new one and then are moved to it with sk_change_net. _After_ this
the socket is unhashed.

The problem is that unhash will decrement the inuse counter for the
socket in a new net, but no in the one it was incremented (init_net).
This will lead to RAW inuse counter increment each time a new net is
created and will be negative for the new net.

Swap unhashing with net changing so that RAW inuse is decremented in
the proper namespace.

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

---
 net/ipv4/icmp.c | 2 +-
 net/ipv6/icmp.c | 2 +-
 net/ipv6/mcast.c | 2 +-
 net/ipv6/ndisc.c | 2 +-
 4 files changed, 4 insertions(+), 4 deletions(-)

```
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 3697e05..74eb856 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -1160,7 +1160,6 @@ int __net_init icmp_sk_init(struct net *net)
     goto fail;

   net->ipv4.icmp_sk[i] = sk = sock->sk;
-  sk_change_net(sk, net);

   sk->sk_allocation = GFP_ATOMIC;

@@ -1179,6 +1178,7 @@ int __net_init icmp_sk_init(struct net *net)
    * packets.
    */
   sk->sk_prot->unhash(sk);
+  sk_change_net(sk, net);
  }

  /* Control parameters for ECHO replies. */
diff --git a/net/ipv6/icmp.c b/net/ipv6/icmp.c
index 63309d1..fec5d55 100644
--- a/net/ipv6/icmp.c
+++ b/net/ipv6/icmp.c
@@ -820,7 +820,6 @@ static int __net_init icmpv6_sk_init(struct net *net)
   }

   net->ipv6.icmp_sk[i] = sk = sock->sk;
-  sk_change_net(sk, net);

   sk->sk_allocation = GFP_ATOMIC;
   /*
@@ -839,6 +838,7 @@ static int __net_init icmpv6_sk_init(struct net *net)
    (2 * ((64 * 1024) + sizeof(struct sk_buff)));

   sk->sk_prot->unhash(sk);
+  sk_change_net(sk, net);
  }
  return 0;

diff --git a/net/ipv6/mcast.c b/net/ipv6/mcast.c
index d810cff..8fbfffc 100644
--- a/net/ipv6/mcast.c
+++ b/net/ipv6/mcast.c
@@ -2686,9 +2686,9 @@ static int igmp6_net_init(struct net *net)
  }
```

```
  net->ipv6.igmp_sk = sk = sock->sk;
- sk_change_net(sk, net);
  sk->sk_allocation = GFP_ATOMIC;
  sk->sk_prot->unhash(sk);
+ sk_change_net(sk, net);

  np = inet6_sk(sk);
  np->hop_limit = 1;
diff --git a/net/ipv6/ndisc.c b/net/ipv6/ndisc.c
index b4d8e33..a13ec57 100644
--- a/net/ipv6/ndisc.c
+++ b/net/ipv6/ndisc.c
@@ -1731,7 +1731,6 @@ static int ndisc_net_init(struct net *net)
  }

  net->ipv6.ndisc_sk = sk = sock->sk;
- sk_change_net(sk, net);

  np = inet6_sk(sk);
  sk->sk_allocation = GFP_ATOMIC;
@@ -1739,6 +1738,7 @@ static int ndisc_net_init(struct net *net)
  /* Do not loopback ndisc messages */
  np->mc_loop = 0;
  sk->sk_prot->unhash(sk);
+ sk_change_net(sk, net);

  return 0;
 }
--
1.5.3.4
```

Subject: [PATCH net-2.6.26 4/6][NETNS][SOCK]: Create sockstat and sockstat6 files in net.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:17:46 GMT
View Forum Message <> Reply to Message

This in nothing but register two new pernet ops (for ipv4 and ipv6)
and create the files in init callbacks.

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

```
---
 net/ipv4/proc.c |   27 ++++++++++++++++++++++-----
 net/ipv6/proc.c |   28 ++++++++++++++++++++++-----
 2 files changed, 45 insertions(+), 10 deletions(-)

diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
```

```
index 8156c26..24ae23b 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -426,25 +426,42 @@ static const struct file_operations netstat_seq_fops = {
  .release = single_release,
 };

+static __net_init int ip_proc_init_net(struct net *net)
+{
+ if (!proc_net_fops_create(net, "sockstat", S_IRUGO, &sockstat_seq_fops))
+  return -ENOMEM;
+ return 0;
+}
+
+static __net_exit void ip_proc_exit_net(struct net *net)
+{
+ proc_net_remove(net, "sockstat");
+}
+
+static __net_initdata struct pernet_operations ip_proc_ops = {
+ .init = ip_proc_init_net,
+ .exit = ip_proc_exit_net,
+};
+
 int __init ip_misc_proc_init(void)
 {
  int rc = 0;

+ if (register_pernet_subsys(&ip_proc_ops))
+  goto out_pernet;
+
  if (!proc_net_fops_create(&init_net, "netstat", S_IRUGO, &netstat_seq_fops))
   goto out_netstat;

  if (!proc_net_fops_create(&init_net, "snmp", S_IRUGO, &snmp_seq_fops))
   goto out_snmp;
-
- if (!proc_net_fops_create(&init_net, "sockstat", S_IRUGO, &sockstat_seq_fops))
-  goto out_sockstat;
 out:
  return rc;
-out_sockstat:
- proc_net_remove(&init_net, "snmp");
 out_snmp:
  proc_net_remove(&init_net, "netstat");
 out_netstat:
+ unregister_pernet_subsys(&ip_proc_ops);
+out_pernet:
```

```
  rc = -ENOMEM;
  goto out;
 }
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 4b9d5a9..6a34794 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -243,27 +243,44 @@ int snmp6_unregister_dev(struct inet6_dev *idev)
  return 0;
 }

+static int ipv6_proc_init_net(struct net *net)
+{
+ if (!proc_net_fops_create(net, "sockstat6", S_IRUGO, &sockstat6_seq_fops))
+  return -ENOMEM;
+ return 0;
+}
+
+static void ipv6_proc_exit_net(struct net *net)
+{
+ proc_net_remove(net, "sockstat6");
+}
+
+static struct pernet_operations ipv6_proc_ops = {
+ .init = ipv6_proc_init_net,
+ .exit = ipv6_proc_exit_net,
+};
+
 int __init ipv6_misc_proc_init(void)
 {
  int rc = 0;

+ if (register_pernet_subsys(&ipv6_proc_ops))
+  goto proc_net_fail;
+
  if (!proc_net_fops_create(&init_net, "snmp6", S_IRUGO, &snmp6_seq_fops))
   goto proc_snmp6_fail;

  proc_net_devsnmp6 = proc_mkdir("dev_snmp6", init_net.proc_net);
  if (!proc_net_devsnmp6)
   goto proc_dev_snmp6_fail;
-
- if (!proc_net_fops_create(&init_net, "sockstat6", S_IRUGO, &sockstat6_seq_fops))
-  goto proc_sockstat6_fail;
 out:
  return rc;

-proc_sockstat6_fail:
```

```
- proc_net_remove(&init_net, "dev_snmp6");
 proc_dev_snmp6_fail:
  proc_net_remove(&init_net, "snmp6");
 proc_snmp6_fail:
+ unregister_pernet_subsys(&ipv6_proc_ops);
+proc_net_fail:
  rc = -ENOMEM;
  goto out;
 }
@@ -273,5 +290,6 @@ void ipv6_misc_proc_exit(void)
  proc_net_remove(&init_net, "sockstat6");
  proc_net_remove(&init_net, "dev_snmp6");
  proc_net_remove(&init_net, "snmp6");
+ unregister_pernet_subsys(&ipv6_proc_ops);
 }

--
1.5.3.4
```

Subject: [PATCH net-2.6.26 5/6][NETNS][IPV4]: Use proper net in sockstat file.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:19:45 GMT
View Forum Message <> Reply to Message

Now show the per-net inuse values in sockstat file.

Besides, now we can see per-net fragments statistics in the
same file, since this stats is already per-net.

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

```
---
 net/ipv4/proc.c |   41 ++++++++++++++++++++++++++++++++++-------
 1 files changed, 34 insertions(+), 7 deletions(-)

diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index 24ae23b..552169b 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -51,27 +51,54 @@
  */
 static int sockstat_seq_show(struct seq_file *seq, void *v)
 {
+ struct net *net = seq->private;
+
  socket_seq_show(seq);
  seq_printf(seq, "TCP: inuse %d orphan %d tw %d alloc %d mem %d\n",
-     sock_prot_inuse_get(&init_net, &tcp_prot),
```

```
+     sock_prot_inuse_get(net, &tcp_prot),
      atomic_read(&tcp_orphan_count),
      tcp_death_row.tw_count, atomic_read(&tcp_sockets_allocated),
      atomic_read(&tcp_memory_allocated));
  seq_printf(seq, "UDP: inuse %d mem %d\n",
-     sock_prot_inuse_get(&init_net, &udp_prot),
+     sock_prot_inuse_get(net, &udp_prot),
      atomic_read(&udp_memory_allocated));
  seq_printf(seq, "UDPLITE: inuse %d\n",
-     sock_prot_inuse_get(&init_net, &udplite_prot));
+     sock_prot_inuse_get(net, &udplite_prot));
  seq_printf(seq, "RAW: inuse %d\n",
-     sock_prot_inuse_get(&init_net, &raw_prot));
+     sock_prot_inuse_get(net, &raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
-   ip_frag_nqueues(&init_net), ip_frag_mem(&init_net));
+   ip_frag_nqueues(net), ip_frag_mem(net));
  return 0;
 }

 static int sockstat_seq_open(struct inode *inode, struct file *file)
 {
- return single_open(file, sockstat_seq_show, NULL);
+ int err;
+ struct net *net;
+
+ err = -ENXIO;
+ net = get_proc_net(inode);
+ if (net == NULL)
+  goto err_net;
+
+ err = single_open(file, sockstat_seq_show, net);
+ if (err < 0)
+  goto err_open;
+
+ return 0;
+
+err_open:
+ put_net(net);
+err_net:
+ return err;
+}
+
+static int sockstat_seq_release(struct inode *inode, struct file *file)
+{
+ struct net *net = ((struct seq_file *)file->private_data)->private;
+
+ put_net(net);
```

```
+ return single_release(inode, file);
 }

 static const struct file_operations sockstat_seq_fops = {
@@ -79,7 +106,7 @@ static const struct file_operations sockstat_seq_fops = {
  .open  = sockstat_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
- .release = single_release,
+ .release = sockstat_seq_release,
 };

 /* snmp items */
--
1.5.3.4
```

Subject: [PATCH net-2.6.26 6/6][NETNS][IPV6]: Use proper net in sockstat6 file.
Posted by Pavel Emelianov on Thu, 27 Mar 2008 08:21:14 GMT
View Forum Message <> Reply to Message

Do with sockstat file what we've already done for sockstat. Same good
side effect - ipv6 reassembling stats are now shown per-net.

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

```
---
 net/ipv6/proc.c |   41 +++++++++++++++++++++++++++++++++++++----
 1 files changed, 34 insertions(+), 7 deletions(-)

diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 6a34794..783b30c 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -35,16 +35,18 @@ static struct proc_dir_entry *proc_net_devsnmp6;

 static int sockstat6_seq_show(struct seq_file *seq, void *v)
 {
+ struct net *net = seq->private;
+
  seq_printf(seq, "TCP6: inuse %d\n",
-        sock_prot_inuse_get(&init_net, &tcpv6_prot));
+        sock_prot_inuse_get(net, &tcpv6_prot));
  seq_printf(seq, "UDP6: inuse %d\n",
-        sock_prot_inuse_get(&init_net, &udpv6_prot));
+        sock_prot_inuse_get(net, &udpv6_prot));
  seq_printf(seq, "UDPLITE6: inuse %d\n",
-   sock_prot_inuse_get(&init_net, &udplitev6_prot));
```

```
+   sock_prot_inuse_get(net, &udplitev6_prot));
  seq_printf(seq, "RAW6: inuse %d\n",
-         sock_prot_inuse_get(&init_net, &rawv6_prot));
+         sock_prot_inuse_get(net, &rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
-         ip6_frag_nqueues(&init_net), ip6_frag_mem(&init_net));
+         ip6_frag_nqueues(net), ip6_frag_mem(net));
  return 0;
 }

@@ -183,7 +185,32 @@ static int snmp6_seq_show(struct seq_file *seq, void *v)

 static int sockstat6_seq_open(struct inode *inode, struct file *file)
 {
- return single_open(file, sockstat6_seq_show, NULL);
+ int err;
+ struct net *net;
+
+ err = -ENXIO;
+ net = get_proc_net(inode);
+ if (net == NULL)
+  goto err_net;
+
+ err = single_open(file, sockstat6_seq_show, net);
+ if (err < 0)
+  goto err_open;
+
+ return 0;
+
+err_open:
+ put_net(net);
+err_net:
+ return err;
+}
+
+static int sockstat6_seq_release(struct inode *inode, struct file *file)
+{
+ struct net *net = ((struct seq_file *)file->private_data)->private;
+
+ put_net(net);
+ return single_release(inode, file);
 }

 static const struct file_operations sockstat6_seq_fops = {
@@ -191,7 +218,7 @@ static const struct file_operations sockstat6_seq_fops = {
  .open  = sockstat6_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
```

```
- .release = single_release,
+ .release = sockstat6_seq_release,
 };

 static int snmp6_seq_open(struct inode *inode, struct file *file)
--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.26 2/6][NETNS][SOCK]: Introduce per-net inuse counters.
Posted by Eric Dumazet on Thu, 27 Mar 2008 20:21:02 GMT
View Forum Message <> Reply to Message

> This is probably the most controversial part of the set.
>
> The counters are stored in a per-cpu array on a struct net. To
> index in this array the prot->inuse is declared as int and used.
>
> Numbers (indices) to protos are generated with the appropriate
> enum. I though about using some existing IPPROTO_XXX numbers for
> protocols but they were too large (IPPROTO_RAW is 255) and did
> not differ for ipv4 and ipv6 (there's no IP6PROTO_RAW, etc).
>
> The sock_prot_inuse_(add|get) now use the net argument to
> get the counter, but this all hides under CONFIG_NET_NS.
>
> The sock_prot_inuse_(init|fini) are no-ops. DEFINE_PROTO_INUSE
> is empty and REF_PROTO_INUSE assigns an index to a proto.
>
>

Given that :

1) pcounter should really go away from kernel, since Andrew disagree
with the implementation.

2) the need to enumerate all protocols in your enum, it seems ... ugly :)

3) alloc_percpu(struct net_prot_inuse) per net is nice because we dont
waste memory (if we had to use percpu_counters for each proto for example)

I suggest to :

1) not use pcounter anymore

2) change 'inuse' field to 'inuse_idx' or 'prot_num' that is

automatically allocated at proto_register time, instead statically at
compile time.

Just provide a big enough NET_INUSE_NR (might depend on IPV6 present or
not, static or module) to take into account all possible protocols.

struct net_prot_inuse {
 int val[NET_INUSE_NR];
};

> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
> include/net/net_namespace.h | 3 ++
> include/net/sock.h | 35 ++++++++++++++++++++++++++++++++++++
> net/core/sock.c | 52 ++
> 3 files changed, 90 insertions(+), 0 deletions(-)
>
> diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
> index f8f3d1a..8a37be1 100644
> --- a/include/net/net_namespace.h
> +++ b/include/net/net_namespace.h
> @@ -18,6 +18,7 @@ struct proc_dir_entry;
> struct net_device;
> struct sock;
> struct ctl_table_header;
> +struct net_prot_inuse;
>
> struct net {
>   atomic_t count; /* To decided when the network
> @@ -50,6 +51,8 @@ struct net {
>   struct ctl_table_header *sysctl_core_hdr;
>   int   sysctl_somaxconn;
>
> + struct net_prot_inuse *inuse;
> +
>   struct netns_packet packet;
>   struct netns_unix unx;
>   struct netns_ipv4 ipv4;
> diff --git a/include/net/sock.h b/include/net/sock.h
> index a57c58f..84a672c 100644
> --- a/include/net/sock.h
> +++ b/include/net/sock.h
> @@ -562,8 +562,12 @@ struct proto {
>
>   /* Keeping track of sockets in use */
> #ifdef CONFIG_PROC_FS

> +#ifdef CONFIG_NET_NS
> + unsigned int inuse;
> +#else
>   struct pcounter inuse;
> #endif
> +#endif
>
>   /* Memory pressure */
>   void   (*enter_memory_pressure)(void);
> @@ -635,6 +639,36 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)
>
>
> #ifdef CONFIG_PROC_FS
> +#ifdef CONFIG_NET_NS
> +enum {
> + NET_INUSE_dccp_v4,
> + NET_INUSE_dccp_v6,
> + NET_INUSE_raw,
> + NET_INUSE_tcp,
> + NET_INUSE_udp,
> + NET_INUSE_udplite,
> + NET_INUSE_rawv6,
> + NET_INUSE_tcpv6,
> + NET_INUSE_udpv6,
> + NET_INUSE_udplitev6,
> + NET_INUSE_sctp,
> + NET_INUSE_sctpv6,
> + NET_INUSE_NR,
> +};
> +
> +# define DEFINE_PROTO_INUSE(NAME)
> +# define REF_PROTO_INUSE(NAME) .inuse = NET_INUSE_##NAME,
> +
> +extern void sock_prot_inuse_add(struct net *net, struct proto *prot, int inc);
> +static inline int sock_prot_inuse_init(struct proto *proto)
> +{
> + return 0;
> +}
> +extern int sock_prot_inuse_get(struct net *net, struct proto *proto);
> +static inline void sock_prot_inuse_free(struct proto *proto)
> +{
> +}
> +#else /* ! CONFIG_NET_NS */
> # define DEFINE_PROTO_INUSE(NAME) DEFINE_PCOUNTER(NAME)
> # define REF_PROTO_INUSE(NAME) PCOUNTER_MEMBER_INITIALIZER(NAME, .inuse)
> /* Called with local bh disabled */
> @@ -655,6 +689,7 @@ static inline void sock_prot_inuse_free(struct proto *proto)
> {

> pcounter_free(&proto->inuse);
> }
> +#endif /* CONFIG_NET_NS */
> #else
> # define DEFINE_PROTO_INUSE(NAME)
> # define REF_PROTO_INUSE(NAME)
> diff --git a/net/core/sock.c b/net/core/sock.c
> index 3ee9506..743f628 100644
> --- a/net/core/sock.c
> +++ b/net/core/sock.c
> @@ -2056,6 +2056,58 @@ void proto_unregister(struct proto *prot)
> EXPORT_SYMBOL(proto_unregister);
>
> #ifdef CONFIG_PROC_FS
> +#ifdef CONFIG_NET_NS
> +struct net_prot_inuse {
> + int val[NET_INUSE_NR];
> +};
> +
> +void sock_prot_inuse_add(struct net *net, struct proto *prot, int val)
> +{
> + per_cpu_ptr(net->inuse, get_cpu())->val[prot->inuse] += val;
> + put_cpu();
> +}
> +EXPORT_SYMBOL_GPL(sock_prot_inuse_add);
> +
> +int sock_prot_inuse_get(struct net *net, struct proto *prot)
> +{
> + int cpu, idx, val;
> +
> + idx = prot->inuse;
> + val = 0;
> + for_each_online_cpu(cpu)
> + val += per_cpu_ptr(net->inuse, cpu)->val[idx];
> +
> + return val;
> +}
> +EXPORT_SYMBOL_GPL(sock_prot_inuse_get);
> +
> +static int sock_inuse_init_net(struct net *net)
> +{
> + net->inuse = alloc_percpu(struct net_prot_inuse);
> + return net->inuse ? 0 : -ENOMEM;
> +}
> +
> +static void sock_inuse_exit_net(struct net *net)
> +{
> + free_percpu(net->inuse);

```
> +}
> +
> +static struct pernet_operations net_inuse_ops = {
> + .init = sock_inuse_init_net,
> + .exit = sock_inuse_exit_net,
> +};
> +
> +static __init int net_inuse_init(void)
> +{
> + if (register_pernet_subsys(&net_inuse_ops))
> +  panic("Cannot initialize net inuse counters");
> +
> + return 0;
> +}
> +
> +core_initcall(net_inuse_init);
> +#endif
> +
>  static void *proto_seq_start(struct seq_file *seq, loff_t *pos)
>   __acquires(proto_list_lock)
>  {
>
```

Subject: Re: [PATCH net-2.6.26 2/6][NETNS][SOCK]: Introduce per-net inuse counters.
Posted by davem on Fri, 28 Mar 2008 00:38:23 GMT
View Forum Message <> Reply to Message

From: Eric Dumazet <dada1@cosmosbay.com>
Date: Thu, 27 Mar 2008 21:21:02 +0100

> Given that :
>
> 1) pcounter should really go away from kernel, since Andrew disagree
> with the implementation.
>
> 2) the need to enumerate all protocols in your enum, it seems ... ugly :)
>
> 3) alloc_percpu(struct net_prot_inuse) per net is nice because we dont
> waste memory (if we had to use percpu_counters for each proto for example)
>
> I suggest to :
>
> 1) not use pcounter anymore
>
> 2) change 'inuse' field to 'inuse_idx' or 'prot_num' that is
> automatically allocated at proto_register time, instead statically at

> compile time.
>
> Just provide a big enough NET_INUSE_NR (might depend on IPV6 present or
> not, static or module) to take into account all possible protocols.
>
> struct net_prot_inuse {
> int val[NET_INUSE_NR];
> };

I agree mostly, and I'll thus hold off on these patches
until the details are hashed out.

Subject: Re: [PATCH net-2.6.26 2/6][NETNS][SOCK]: Introduce per-net inuse counters.
Posted by Pavel Emelianov on Fri, 28 Mar 2008 07:18:47 GMT
View Forum Message <> Reply to Message

Eric Dumazet wrote:

>> This is probably the most controversial part of the set.
>>
>> The counters are stored in a per-cpu array on a struct net. To
>> index in this array the prot->inuse is declared as int and used.
>>
>> Numbers (indices) to protos are generated with the appropriate
>> enum. I though about using some existing IPPROTO_XXX numbers for
>> protocols but they were too large (IPPROTO_RAW is 255) and did
>> not differ for ipv4 and ipv6 (there's no IP6PROTO_RAW, etc).
>>
>> The sock_prot_inuse_(add|get) now use the net argument to
>> get the counter, but this all hides under CONFIG_NET_NS.
>>
>> The sock_prot_inuse_(init|fini) are no-ops. DEFINE_PROTO_INUSE
>> is empty and REF_PROTO_INUSE assigns an index to a proto.
>>
>>
>
> Given that :
>
> 1) pcounter should really go away from kernel, since Andrew disagree
> with the implementation.

Does this and ... (below)

> 2) the need to enumerate all protocols in your enum, it seems ... ugly :)

Yup :(

> 3) alloc_percpu(struct net_prot_inuse) per net is nice because we dont
> waste memory (if we had to use percpu_counters for each proto for example)
>
> I suggest to :
>
> 1) not use pcounter anymore

... this mean that I can rework the inuse accounting in order not
to use pcounters at all even with CONFIG_NET_NS=n? :)

> 2) change 'inuse' field to 'inuse_idx' or 'prot_num' that is
> automatically allocated at proto_register time, instead statically at
> compile time.

Hm... I like this approach. Will do.

> Just provide a big enough NET_INUSE_NR (might depend on IPV6 present or
> not, static or module) to take into account all possible protocols.

Well, I though about this, but wasn't sure whether such heuristics
would be accepted.

> struct net_prot_inuse {
> int val[NET_INUSE_NR];
> };
>
>

Subject: Re: [PATCH net-2.6.26 2/6][NETNS][SOCK]: Introduce per-net inuse counters.
Posted by Eric Dumazet on Fri, 28 Mar 2008 07:36:15 GMT
View Forum Message <> Reply to Message

> Eric Dumazet wrote:
>

>>
>>> This is probably the most controversial part of the set.
>>>
>>> The counters are stored in a per-cpu array on a struct net. To
>>> index in this array the prot->inuse is declared as int and used.
>>>
>>> Numbers (indices) to protos are generated with the appropriate
>>> enum. I though about using some existing IPPROTO_XXX numbers for
>>> protocols but they were too large (IPPROTO_RAW is 255) and did

>>> not differ for ipv4 and ipv6 (there's no IP6PROTO_RAW, etc).
>>>
>>> The sock_prot_inuse_(add|get) now use the net argument to
>>> get the counter, but this all hides under CONFIG_NET_NS.
>>>
>>> The sock_prot_inuse_(init|fini) are no-ops. DEFINE_PROTO_INUSE
>>> is empty and REF_PROTO_INUSE assigns an index to a proto.
>>>
>>>
>>>
>> Given that :
>>
>> 1) pcounter should really go away from kernel, since Andrew disagree
>> with the implementation.
>>
>
> Does this and ... (below)
>
>
>> 2) the need to enumerate all protocols in your enum, it seems ... ugly :)
>>
>
> Yup :(
>
>
>> 3) alloc_percpu(struct net_prot_inuse) per net is nice because we dont
>> waste memory (if we had to use percpu_counters for each proto for example)
>>
>> I suggest to :
>>
>> 1) not use pcounter anymore
>>
>
> ... this mean that I can rework the inuse accounting in order not
> to use pcounters at all even with CONFIG_NET_NS=n? :)
>
>
Absolutely

I had to do it eventually but my paid work is currently taking me 10-12
hours per day, so please be my guest :)

reference : http://kerneltrap.org/mailarchive/linux-kernel/2008/2/16/873754

>> 2) change 'inuse' field to 'inuse_idx' or 'prot_num' that is
>> automatically allocated at proto_register time, instead statically at
>> compile time.
>>

>
> Hm... I like this approach. Will do.
>
>
>> Just provide a big enough NET_INUSE_NR (might depend on IPV6 present or
>> not, static or module) to take into account all possible protocols.
>>
>
> Well, I though about this, but wasn't sure whether such heuristics
> would be accepted.
>
>
>> struct net_prot_inuse {
>> int val[NET_INUSE_NR];
>> };
>>
>>
>>
>
>
>

Thank you
