Subject: [PATCH v5 05/10] per-netns ipv4 sysctl_tcp_mem
Posted by Glauber Costa on Mon, 07 Nov 2011 15:26:30 GMT
View Forum Message <> Reply to Message

This patch allows each namespace to independently set up
its levels for tcp memory pressure thresholds. This patch
alone does not buy much: we need to make this values
per group of process somehow. This is achieved in the
patches that follows in this patchset.

Signed-off-by: Glauber Costa <glommer@parallels.com>
Reviewed-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: David S. Miller <davem@davemloft.net>
CC: Eric W. Biederman <ebiederm@xmission.com>
---
 include/net/netns/ipv4.h | 1 +
 include/net/tcp.h | 1 -
 mm/memcontrol.c | 8 ++++--
 net/ipv4/af_inet.c | 2 +
 net/ipv4/sysctl_net_ipv4.c | 51 ++++++++++++++++++++++++++++++++++++------
 net/ipv4/tcp.c | 11 +-------
 net/ipv4/tcp_ipv4.c | 1 -
 net/ipv6/af_inet6.c | 2 +
 net/ipv6/tcp_ipv6.c | 1 -
 9 files changed, 56 insertions(+), 22 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index d786b4f..bbd023a 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -55,6 +55,7 @@ struct netns_ipv4 {
  int current_rt_cache_rebuild_count;

  unsigned int sysctl_ping_group_range[2];
+ long sysctl_tcp_mem[3];

  atomic_t rt_genid;
  atomic_t dev_addr_genid;
diff --git a/include/net/tcp.h b/include/net/tcp.h
index 7301ca8..c34b823 100644
--- a/include/net/tcp.h
+++ b/include/net/tcp.h
@@ -230,7 +230,6 @@ extern int sysctl_tcp_fack;
 extern int sysctl_tcp_reordering;
 extern int sysctl_tcp_ecn;
 extern int sysctl_tcp_dsack;
-extern long sysctl_tcp_mem[3];
 extern int sysctl_tcp_wmem[3];

```
 extern int sysctl_tcp_rmem[3];
 extern int sysctl_tcp_app_win;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index f14d7d2..63360f8 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -395,6 +395,7 @@ static inline bool mem_cgroup_is_root(struct mem_cgroup *mem)
 #ifdef CONFIG_INET
 #include <net/sock.h>
 #include <net/ip.h>
+#include <linux/nsproxy.h>

 void sock_update_memcg(struct sock *sk)
 {
@@ -526,14 +527,15 @@ static void tcp_create_cgroup(struct mem_cgroup *cg, struct
cgroup_subsys *ss)
 int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
 {
  struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
+ struct net *net = current->nsproxy->net_ns;
  /*
   * We need to initialize it at populate, not create time.
   * This is because net sysctl tables are not up until much
   * later
   */
- memcg->tcp.tcp_prot_mem[0] = sysctl_tcp_mem[0];
- memcg->tcp.tcp_prot_mem[1] = sysctl_tcp_mem[1];
- memcg->tcp.tcp_prot_mem[2] = sysctl_tcp_mem[2];
+ memcg->tcp.tcp_prot_mem[0] = net->ipv4.sysctl_tcp_mem[0];
+ memcg->tcp.tcp_prot_mem[1] = net->ipv4.sysctl_tcp_mem[1];
+ memcg->tcp.tcp_prot_mem[2] = net->ipv4.sysctl_tcp_mem[2];

  return 0;
 }
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index da19147..73be7da 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1674,6 +1674,8 @@ static int __init inet_init(void)
  ip_static_sysctl_init();
 #endif

+ tcp_prot.sysctl_mem = init_net.ipv4.sysctl_tcp_mem;
+
  /*
   * Add all the base protocols.
   */
diff --git a/net/ipv4/sysctl_net_ipv4.c b/net/ipv4/sysctl_net_ipv4.c
```

```
index 69fd720..bbd67ab 100644
--- a/net/ipv4/sysctl_net_ipv4.c
+++ b/net/ipv4/sysctl_net_ipv4.c
@@ -14,6 +14,7 @@
 #include <linux/init.h>
 #include <linux/slab.h>
 #include <linux/nsproxy.h>
+#include <linux/swap.h>
 #include <net/snmp.h>
 #include <net/icmp.h>
 #include <net/ip.h>
@@ -174,6 +175,36 @@ static int proc_allowed_congestion_control(ctl_table *ctl,
  return ret;
 }

+static int ipv4_tcp_mem(ctl_table *ctl, int write,
+       void __user *buffer, size_t *lenp,
+       loff_t *ppos)
+{
+ int ret;
+ unsigned long vec[3];
+ struct net *net = current->nsproxy->net_ns;
+
+ ctl_table tmp = {
+  .data = &vec,
+  .maxlen = sizeof(vec),
+  .mode = ctl->mode,
+ };
+
+ if (!write) {
+  ctl->data = &net->ipv4.sysctl_tcp_mem;
+  return proc_doulongvec_minmax(ctl, write, buffer, lenp, ppos);
+ }
+
+ ret = proc_doulongvec_minmax(&tmp, write, buffer, lenp, ppos);
+ if (ret)
+  return ret;
+
+ net->ipv4.sysctl_tcp_mem[0] = vec[0];
+ net->ipv4.sysctl_tcp_mem[1] = vec[1];
+ net->ipv4.sysctl_tcp_mem[2] = vec[2];
+
+ return 0;
+}
+
 static struct ctl_table ipv4_table[] = {
  {
   .procname = "tcp_timestamps",
```

```
@@ -433,13 +464,6 @@ static struct ctl_table ipv4_table[] = {
   .proc_handler = proc_dointvec
  },
  {
-  .procname = "tcp_mem",
-  .data  = &sysctl_tcp_mem,
-  .maxlen  = sizeof(sysctl_tcp_mem),
-  .mode  = 0644,
-  .proc_handler = proc_doulongvec_minmax
- },
- {
   .procname = "tcp_wmem",
   .data  = &sysctl_tcp_wmem,
   .maxlen  = sizeof(sysctl_tcp_wmem),
@@ -721,6 +745,12 @@ static struct ctl_table ipv4_net_table[] = {
   .mode  = 0644,
   .proc_handler = ipv4_ping_group_range,
  },
+ {
+  .procname = "tcp_mem",
+  .maxlen  = sizeof(init_net.ipv4.sysctl_tcp_mem),
+  .mode  = 0644,
+  .proc_handler = ipv4_tcp_mem,
+ },
  { }
 };

@@ -734,6 +764,7 @@ EXPORT_SYMBOL_GPL(net_ipv4_ctl_path);
 static __net_init int ipv4_sysctl_init_net(struct net *net)
 {
  struct ctl_table *table;
+ unsigned long limit;

  table = ipv4_net_table;
  if (!net_eq(net, &init_net)) {
@@ -769,6 +800,12 @@ static __net_init int ipv4_sysctl_init_net(struct net *net)

  net->ipv4.sysctl_rt_cache_rebuild_count = 4;

+ limit = nr_free_buffer_pages() / 8;
+ limit = max(limit, 128UL);
+ net->ipv4.sysctl_tcp_mem[0] = limit / 4 * 3;
+ net->ipv4.sysctl_tcp_mem[1] = limit;
+ net->ipv4.sysctl_tcp_mem[2] = net->ipv4.sysctl_tcp_mem[0] * 2;
+
  net->ipv4.ipv4_hdr = register_net_sysctl_table(net,
    net_ipv4_ctl_path, table);
  if (net->ipv4.ipv4_hdr == NULL)
```

```
diff --git a/net/ipv4/tcp.c b/net/ipv4/tcp.c
index 34f5db1..5f618d1 100644
--- a/net/ipv4/tcp.c
+++ b/net/ipv4/tcp.c
@@ -282,11 +282,9 @@ int sysctl_tcp_fin_timeout __read_mostly = TCP_FIN_TIMEOUT;
 struct percpu_counter tcp_orphan_count;
 EXPORT_SYMBOL_GPL(tcp_orphan_count);

-long sysctl_tcp_mem[3] __read_mostly;
 int sysctl_tcp_wmem[3] __read_mostly;
 int sysctl_tcp_rmem[3] __read_mostly;

-EXPORT_SYMBOL(sysctl_tcp_mem);
 EXPORT_SYMBOL(sysctl_tcp_rmem);
 EXPORT_SYMBOL(sysctl_tcp_wmem);

@@ -3272,14 +3270,9 @@ void __init tcp_init(void)
  sysctl_tcp_max_orphans = cnt / 2;
  sysctl_max_syn_backlog = max(128, cnt / 256);

- limit = nr_free_buffer_pages() / 8;
- limit = max(limit, 128UL);
- sysctl_tcp_mem[0] = limit / 4 * 3;
- sysctl_tcp_mem[1] = limit;
- sysctl_tcp_mem[2] = sysctl_tcp_mem[0] * 2;
-
  /* Set per-socket limits to no more than 1/128 the pressure threshold */
- limit = ((unsigned long)sysctl_tcp_mem[1]) << (PAGE_SHIFT - 7);
+ limit = ((unsigned long)init_net.ipv4.sysctl_tcp_mem[1])
+  << (PAGE_SHIFT - 7);
  max_share = min(4UL*1024*1024, limit);

  sysctl_tcp_wmem[0] = SK_MEM_QUANTUM;
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 54f6b96..dd1bab7 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -2616,7 +2616,6 @@ struct proto tcp_prot = {
  .orphan_count  = &tcp_orphan_count,
  .memory_allocated = &tcp_memory_allocated,
  .memory_pressure = &tcp_memory_pressure,
- .sysctl_mem  = sysctl_tcp_mem,
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
diff --git a/net/ipv6/af_inet6.c b/net/ipv6/af_inet6.c
index 51672f8..69a6da3 100644
--- a/net/ipv6/af_inet6.c
```

```
+++ b/net/ipv6/af_inet6.c
@@ -1118,6 +1118,8 @@ static int __init inet6_init(void)
  if (err)
   goto static_sysctl_fail;
 #endif
+ tcpv6_prot.sysctl_mem = init_net.ipv4.sysctl_tcp_mem;
+
  /*
   * ipngwg API draft makes clear that the correct semantics
   * for TCP and UDP is to consider one TCP and UDP instance
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 3c13142..52f8b64 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -2208,7 +2208,6 @@ struct proto tcpv6_prot = {
  .memory_allocated = &tcp_memory_allocated,
  .memory_pressure = &tcp_memory_pressure,
  .orphan_count  = &tcp_orphan_count,
- .sysctl_mem  = sysctl_tcp_mem,
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
--
1.7.6.4
```
