
Subject: [PATCH v7 5/8] per-netns ipv4 sysctl_tcp_mem
Posted by [Glauber Costa](#) on Thu, 13 Oct 2011 13:09:39 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/sysctl_net_ipv4.c | 51 ++++++-----
net/ipv4/tcp.c           | 13 ++-----
5 files changed, 53 insertions(+), 21 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 ec57cf2..3609d87 100644
```

```
--- a/include/net/tcp.h
```

```
+++ b/include/net/tcp.h
```

```
@@ -232,7 +232,6 @@ 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 4e79171..f953b32 100644
```

```

--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -390,6 +390,7 @@ static struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont);
#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 @@ int tcp_init_cgroup(const struct proto *prot, 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/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 259f6d9..b1abebd 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);

@@ -334,7 +332,7 @@ EXPORT_SYMBOL(tcp_enter_memory_pressure_nocg);

```

```

long *tcp_sysctl_mem_nocg(const struct mem_cgroup *memcg)
{
- return sysctl_tcp_mem;
+ return init_net.ipv4.sysctl_tcp_mem;
}
EXPORT_SYMBOL(tcp_sysctl_mem_nocg);

@@ -3298,14 +3296,9 @@ 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;
--

```

1.7.6.4
