
Subject: [PATCH v6 04/10] Account tcp memory as kernel memory

Posted by [Glauber Costa](#) on Fri, 25 Nov 2011 17:38:10 GMT

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

Now that we account and control tcp memory buffers memory for pressure controlling purposes, display this information as part of the normal memcg files and other usages.

Signed-off-by: Glauber Costa <glommer@parallels.com>

CC: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

CC: Eric W. Biederman <ebiederm@xmission.com>

```
include/linux/memcontrol.h | 3 ++
include/net/sock.h         | 3 ++
include/net/tcp_memcg.h    | 17 ++++++++
mm/memcontrol.c            | 39 ++++++++
net/core/sock.c            | 42 ++++++++
net/ipv4/Makefile          | 1 +
net/ipv4/tcp_ipv4.c        | 8 +++++-
net/ipv4/tcp_memcg.c       | 64 ++++++++
net/ipv6/tcp_ipv6.c        | 4 +++
9 files changed, 174 insertions(+), 7 deletions(-)
create mode 100644 include/net/tcp_memcg.h
create mode 100644 net/ipv4/tcp_memcg.c
```

```
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
```

```
index 6644d90..1aff2f6 100644
```

```
--- a/include/linux/memcontrol.h
```

```
+++ b/include/linux/memcontrol.h
```

```
@@ -85,6 +85,9 @@ extern struct mem_cgroup *try_get_mem_cgroup_from_page(struct page
*page);
```

```
extern struct mem_cgroup *mem_cgroup_from_task(struct task_struct *p);
```

```
extern struct mem_cgroup *try_get_mem_cgroup_from_mm(struct mm_struct *mm);
```

```
+extern struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont);
```

```
+extern struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *mem);
```

```
+
```

```
static inline
```

```
int mm_match_cgroup(const struct mm_struct *mm, const struct mem_cgroup *cgroup)
```

```
{
```

```
diff --git a/include/net/sock.h b/include/net/sock.h
```

```
index d802761..da38de2 100644
```

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

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

```
@@ -65,6 +65,9 @@
```

```
#include <net/dst.h>
```

```
#include <net/checksum.h>
```

```

+int sockets_populate(struct cgroup *cgrp, struct cgroup_subsys *ss);
+void sockets_destroy(struct cgroup *cgrp, struct cgroup_subsys *ss);
+
+/*
+ * This structure really needs to be cleaned up.
+ * Most of it is for TCP, and not used by any of
diff --git a/include/net/tcp_memcg.h b/include/net/tcp_memcg.h
new file mode 100644
index 0000000..5f5e158
--- /dev/null
+++ b/include/net/tcp_memcg.h
@@ -0,0 +1,17 @@
+#ifndef _TCP_MEMCG_H
+#define _TCP_MEMCG_H
+
+struct tcp_memcontrol {
+ struct cg_proto cg_proto;
+ /* per-cgroup tcp memory pressure knobs */
+ struct res_counter tcp_memory_allocated;
+ struct percpu_counter tcp_sockets_allocated;
+ /* those two are read-mostly, leave them at the end */
+ long tcp_prot_mem[3];
+ int tcp_memory_pressure;
+};
+
+struct cg_proto *tcp_proto_cgroup(struct mem_cgroup *memcg);
+int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss);
+void tcp_destroy_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss);
+#endif /* _TCP_MEMCG_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 5f29194..2df5d3c 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -49,6 +49,8 @@
#include <linux/cpu.h>
#include <linux/oom.h>
#include "internal.h"
+#include <net/sock.h>
+#include <net/tcp_memcg.h>

#include <asm/uaccess.h>

@@ -294,6 +296,10 @@ struct mem_cgroup {
 */
struct mem_cgroup_stat_cpu nocpu_base;
spinlock_t pcp_counter_lock;
+
+#ifdef CONFIG_INET

```

```

+ struct tcp_memcontrol tcp_mem;
+ #endif
+ };

/* Stuffs for move charges at task migration. */
@@ -385,6 +391,7 @@ static inline bool mem_cgroup_is_root(struct mem_cgroup *mem)
# ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
# ifdef CONFIG_INET
# include <net/sock.h>
+ #include <net/ip.h>

void sock_update_memcg(struct sock *sk)
{
@@ -406,13 +413,21 @@ void sock_update_memcg(struct sock *sk)
}
}

+ struct cg_proto *tcp_proto_cgroup(struct mem_cgroup *memcg)
+ {
+ if (!memcg || mem_cgroup_is_root(memcg))
+ return NULL;
+
+ return &memcg->tcp_mem.cg_proto;
+ }
+ EXPORT_SYMBOL(tcp_proto_cgroup);
+
# endif /* CONFIG_INET */
# endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */

static void mem_cgroup_get(struct mem_cgroup *mem);
static void mem_cgroup_put(struct mem_cgroup *mem);
- static struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *mem);
static void drain_all_stock_async(struct mem_cgroup *mem);

static struct mem_cgroup_per_zone *
@@ -787,7 +802,7 @@ static void memcg_check_events(struct mem_cgroup *mem, struct page
*page)
}
}

- static struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont)
+ struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont)
{
return container_of(cgroup_subsys_state(cont,
mem_cgroup_subsys_id), struct mem_cgroup,
@@ -4828,14 +4843,28 @@ static int register_kmem_files(struct cgroup *cont, struct
cgroup_subsys *ss)

```

```

    ret = cgroup_add_files(cont, ss, kmem_cgroup_files,
        ARRAY_SIZE(kmem_cgroup_files));
+
+ if (!ret)
+   ret = sockets_populate(cont, ss);
+
    return ret;
};

+static void kmem_cgroup_destroy(struct cgroup_subsys *ss,
+   struct cgroup *cont)
+{
+   sockets_destroy(cont, ss);
+}
+else
static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
{
    return 0;
}
+
+static void kmem_cgroup_destroy(struct cgroup_subsys *ss,
+   struct cgroup *cont)
+{
+}
+
#endif

static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
@@ -4954,7 +4983,7 @@ static void mem_cgroup_put(struct mem_cgroup *mem)
/*
 * Returns the parent mem_cgroup in memcgroup hierarchy with hierarchy enabled.
 */
-static struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *mem)
+struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *mem)
{
    if (!mem->res.parent)
        return NULL;
@@ -5037,6 +5066,7 @@ mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
    res_counter_init(&mem->res, &parent->res);
    res_counter_init(&mem->memsw, &parent->memsw);
    res_counter_init(&mem->kmem, &parent->kmem);
+
+
/*
 * We increment refcnt of the parent to ensure that we can
 * safely access it on res_counter_charge/uncharge.
@@ -5053,6 +5083,7 @@ mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
    mem->last_scanned_node = MAX_NUMNODES;
    INIT_LIST_HEAD(&mem->oom_notify);

```

```
+
  if (parent)
    mem->swappiness = mem_cgroup_swappiness(parent);
    atomic_set(&mem->refcnt, 1);
@@ -5078,6 +5109,8 @@ static void mem_cgroup_destroy(struct cgroup_subsys *ss,
{
  struct mem_cgroup *mem = mem_cgroup_from_cont(cont);

+ kmem_cgroup_destroy(ss, cont);
+
  mem_cgroup_put(mem);
}
```

```
diff --git a/net/core/sock.c b/net/core/sock.c
index 8382c80..399e566 100644
```

```
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -135,6 +135,45 @@
#include <net/tcp.h>
#endif
```

```
+static DEFINE_RWLOCK(proto_list_lock);
+static LIST_HEAD(proto_list);
+
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+int sockets_populate(struct cgroup *cgrp, struct cgroup_subsys *ss)
+{
+    struct proto *proto;
+    int ret = 0;
+
+    read_lock(&proto_list_lock);
+    list_for_each_entry(proto, &proto_list, node) {
+        if (proto->init_cgroup)
+            ret = proto->init_cgroup(cgrp, ss);
+        if (ret)
+            goto out;
+    }
+
+    read_unlock(&proto_list_lock);
+    return ret;
+out:
+    list_for_each_entry_continue_reverse(proto, &proto_list, node)
+        if (proto->destroy_cgroup)
+            proto->destroy_cgroup(cgrp, ss);
+    read_unlock(&proto_list_lock);
+    return ret;
+}
```

```
+
+void sockets_destroy(struct cgroup *cgrp, struct cgroup_subsys *ss)
+{
+ struct proto *proto;
+
+ read_lock(&proto_list_lock);
+ list_for_each_entry_reverse(proto, &proto_list, node)
+ if (proto->destroy_cgroup)
+ proto->destroy_cgroup(cgrp, ss);
+ read_unlock(&proto_list_lock);
+}
+
+
+/*
+ * Each address family might have different locking rules, so we have
+ * one slock key per address family:
+@@ -2259,9 +2298,6 @@ void sk_common_release(struct sock *sk)
+}
+EXPORT_SYMBOL(sk_common_release);
+
+
+static DEFINE_RWLOCK(proto_list_lock);
+static LIST_HEAD(proto_list);
+
+
+#ifdef CONFIG_PROC_FS
+#define PROTO_INUSE_NR 64 /* should be enough for the first time */
+struct prot_inuse {
+diff --git a/net/ipv4/Makefile b/net/ipv4/Makefile
+index f2dc69c..393e0af 100644
+--- a/net/ipv4/Makefile
++++ b/net/ipv4/Makefile
+@@ -47,6 +47,7 @@ obj-$(CONFIG_TCP_CONG_SCALABLE) += tcp_scalable.o
+ obj-$(CONFIG_TCP_CONG_LP) += tcp_lp.o
+ obj-$(CONFIG_TCP_CONG_YEAH) += tcp_yeah.o
+ obj-$(CONFIG_TCP_CONG_ILLINOIS) += tcp_illinois.o
++obj-$(CONFIG_CGROUP_MEM_RES_CTLR_KMEM) += tcp_memcg.o
+ obj-$(CONFIG_NETLABEL) += cipso_ipv4.o
+
+
+ obj-$(CONFIG_XFRM) += xfrm4_policy.o xfrm4_state.o xfrm4_input.o \
+diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
+index f124a4b..c517d04 100644
+--- a/net/ipv4/tcp_ipv4.c
++++ b/net/ipv4/tcp_ipv4.c
+@@ -73,6 +73,7 @@
+#include <net/xfrm.h>
+#include <net/netdma.h>
+#include <net/secure_seq.h>
++#include <net/tcp_memcg.h>
```

```

#include <linux/inet.h>
#include <linux/ipv6.h>
@@ -1917,6 +1918,7 @@ static int tcp_v4_init_sock(struct sock *sk)
    sk_sockets_allocated_inc(sk);
    local_bh_enable();

+ sock_update_memcg(sk);
    return 0;
}

@@ -2629,10 +2631,14 @@ struct proto tcp_prot = {
    .compat_setsockopt = compat_tcp_setsockopt,
    .compat_getsockopt = compat_tcp_getsockopt,
#ifdef
+ #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ .init_cgroup = tcp_init_cgroup,
+ .destroy_cgroup = tcp_destroy_cgroup,
+ .proto_cgroup = tcp_proto_cgroup,
+ #endif
};
EXPORT_SYMBOL(tcp_prot);

-
static int __net_init tcp_sk_init(struct net *net)
{
    return inet_ctl_sock_create(&net->ipv4.tcp_sock,
diff --git a/net/ipv4/tcp_memcg.c b/net/ipv4/tcp_memcg.c
new file mode 100644
index 0000000..cc91918
--- /dev/null
+++ b/net/ipv4/tcp_memcg.c
@@ -0,0 +1,64 @@
#include <net/tcp.h>
#include <net/tcp_memcg.h>
#include <net/sock.h>
#include <linux/memcontrol.h>
+
+static inline struct tcp_memcontrol *tcp_from_cgproto(struct cg_proto *cg_proto)
+{
+ return container_of(cg_proto, struct tcp_memcontrol, cg_proto);
+}
+
+int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
+{
+ /*
+  * The root cgroup does not use res_counters, but rather,
+  * rely on the data already collected by the network
+  * subsystem

```

```

+ */
+ struct res_counter *res_parent = NULL;
+ struct cg_proto *cg_proto;
+ struct tcp_memcontrol *tcp;
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
+ struct mem_cgroup *parent = parent_mem_cgroup(memcg);
+
+ cg_proto = tcp_prot.proto_cgroup(memcg);
+ if (!cg_proto)
+ return 0;
+
+ tcp = tcp_from_cgproto(cg_proto);
+ cg_proto->parent = tcp_prot.proto_cgroup(parent);
+
+ tcp->tcp_prot_mem[0] = sysctl_tcp_mem[0];
+ tcp->tcp_prot_mem[1] = sysctl_tcp_mem[1];
+ tcp->tcp_prot_mem[2] = sysctl_tcp_mem[2];
+ tcp->tcp_memory_pressure = 0;
+
+ if (cg_proto->parent)
+ res_parent = cg_proto->parent->memory_allocated;
+
+ res_counter_init(&tcp->tcp_memory_allocated, res_parent);
+ percpu_counter_init(&tcp->tcp_sockets_allocated, 0);
+
+ cg_proto->memory_pressure = &tcp->tcp_memory_pressure;
+ cg_proto->sysctl_mem = tcp->tcp_prot_mem;
+ cg_proto->memory_allocated = &tcp->tcp_memory_allocated;
+ cg_proto->sockets_allocated = &tcp->tcp_sockets_allocated;
+
+ return 0;
+}
+EXPORT_SYMBOL(tcp_init_cgroup);
+
+void tcp_destroy_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
+{
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
+ struct cg_proto *cg_proto;
+ struct tcp_memcontrol *tcp;
+
+ cg_proto = tcp_prot.proto_cgroup(memcg);
+ if (!cg_proto)
+ return;
+
+ tcp = tcp_from_cgproto(cg_proto);
+ percpu_counter_destroy(&tcp->tcp_sockets_allocated);
+}
+EXPORT_SYMBOL(tcp_destroy_cgroup);

```



```
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 3a08fcd..d57d7a7 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -62,6 +62,7 @@
#include <net/netdma.h>
#include <net/inet_common.h>
#include <net/secure_seq.h>
+#include <net/tcp_memcg.h>

#include <asm/uaccess.h>

@@ -2222,6 +2223,9 @@ struct proto tcpv6_prot = {
    .compat_setsockopt = compat_tcp_setsockopt,
    .compat_getsockopt = compat_tcp_getsockopt,
    #endif
+    #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+    .proto_cgroup = tcp_proto_cgroup,
+    #endif
};

static const struct inet6_protocol tcpv6_protocol = {
--
1.7.6.4
```
