

---

Subject: [PATCH v6 6/8] tcp buffer limitation: per-cgroup limit  
Posted by [Glauber Costa](#) on Mon, 10 Oct 2011 10:24:26 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

This patch uses the "tcp\_max\_mem" field of the kmem\_cgroup to effectively control the amount of kernel memory pinned by a cgroup.

We have to make sure that none of the memory pressure thresholds specified in the namespace are bigger than the current cgroup.

Signed-off-by: Glauber Costa <glommer@parallels.com>  
CC: David S. Miller <davem@davemloft.net>  
CC: Hiroyouki Kamezawa <kamezawa.hiroyu@jp.fujitsu.com>  
CC: Eric W. Biederman <ebiederm@xmission.com>

---

```
Documentation/cgroups/memory.txt | 1 +
include/linux/memcontrol.h       | 10 +++++
mm/memcontrol.c                  | 79 +++++++++++++++++++++++++++++++++++++-----
net/ipv4/sysctl_net_ipv4.c       | 20 ++++++++
4 files changed, 102 insertions(+), 8 deletions(-)
```

```
diff --git a/Documentation/cgroups/memory.txt b/Documentation/cgroups/memory.txt
index bf00cd2..c1db134 100644
--- a/Documentation/cgroups/memory.txt
+++ b/Documentation/cgroups/memory.txt
@@ -78,6 +78,7 @@ @@ Brief summary of control files.
```

```
memory.independent_kmem_limit # select whether or not kernel memory limits are
    independent of user limits
+ memory.kmem.tcp.limit_in_bytes # set/show hard limit for tcp buf memory
```

## 1. History

```
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index a27dad9..e0ccec5 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -397,6 +397,9 @@ @@ int tcp_init_cgroup(const struct proto *prot, struct cgroup *cgrp,
    struct cgroup_subsys *ss);
void tcp_destroy_cgroup(const struct proto *prot, struct cgroup *cgrp,
    struct cgroup_subsys *ss);
+
+unsigned long long tcp_max_memory(const struct mem_cgroup *memcg);
+void tcp_prot_mem(struct mem_cgroup *memcg, long val, int idx);
#else
/* memcontrol includes sockets.h, that includes memcontrol.h ... */
static inline void memcg_sockets_allocated_dec(struct mem_cgroup *memcg,
@@ -413,6 +416,13 @@ @@ static inline void sock_update_memcg(struct sock *sk)
```

```

static inline void sock_release_memcg(struct sock *sk)
{
}
+static inline unsigned long long tcp_max_memory(const struct mem_cgroup *memcg)
+{
+ return -1ULL;
+}
+static inline void tcp_prot_mem(struct mem_cgroup *memcg, long val, int idx)
+{
+}
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
#endif /* CONFIG_INET */
#endif /* _LINUX_MEMCONTROL_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 651badb..60e0ec0 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -365,6 +365,7 @@ enum mem_type {
    _MEMSWAP,
    _OOM_TYPE,
    _KMEM,
+ _KMEM_TCP,
};

#define MEMFILE_PRIVATE(x, val) (((x) << 16) | (val))
@@ -385,6 +386,11 @@ enum mem_type {

static struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *memcg);
static struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont);
+static inline bool mem_cgroup_is_root(struct mem_cgroup *memcg)
+{
+ return (memcg == root_mem_cgroup);
+}
+
/* Writing them here to avoid exposing memcg's inner layout */
#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
#ifdef CONFIG_INET
@@ -510,6 +516,35 @@ struct percpu_counter *sockets_allocated_tcp(const struct mem_cgroup
*memcg)
}
EXPORT_SYMBOL(sockets_allocated_tcp);

+static void tcp_update_limit(struct mem_cgroup *memcg, u64 val)
+{
+ struct net *net = current->nsproxy->net_ns;
+ int i;
+
+ val >>= PAGE_SHIFT;

```

```

+
+ for (i = 0; i < 3; i++)
+ memcg->tcp.tcp_prot_mem[i] = min_t(long, val,
+   net->ipv4.sysctl_tcp_mem[i]);
+}
+
+static int mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
+   const char *buffer);
+
+static u64 mem_cgroup_read(struct cgroup *cont, struct cftype *cft);
+/*
+ * We need those things internally in pages, so don't reuse
+ * mem_cgroup_{read,write}
+ */
+static struct cftype tcp_files[] = {
+ {
+   .name = "kmem.tcp.limit_in_bytes",
+   .write_string = mem_cgroup_write,
+   .read_u64 = mem_cgroup_read,
+   .private = MEMFILE_PRIVATE(_KMEM_TCP, RES_LIMIT),
+ },
+};
+
+static void tcp_create_cgroup(struct mem_cgroup *cg, struct cgroup_subsys *ss)
+{
+   struct res_counter *parent_res_counter = NULL;
@@ -527,6 +562,7 @@ 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 mem_cgroup *parent = parent_mem_cgroup(memcg);
+   struct net *net = current->nsproxy->net_ns;
+   /*
+    * We need to initialize it at populate, not create time.
+@@ -537,7 +573,20 @@ int tcp_init_cgroup(const struct proto *prot, struct cgroup *cgrp,
+   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;
+   /* Let root cgroup unlimited. All others, respect parent's if needed */
+   if (parent && !parent->use_hierarchy) {
+   unsigned long limit;
+   int ret;
+   limit = nr_free_buffer_pages() / 8;
+   limit = max(limit, 128UL);
+   ret = res_counter_set_limit(&memcg->tcp.memory_allocated,
+   limit * 2);
+   if (ret)

```

```

+ return ret;
+ }
+
+ return cgroup_add_files(cgrp, ss, tcp_files,
+ ARRAY_SIZE(tcp_files));
+ }
EXPORT_SYMBOL(tcp_init_cgroup);

@@ -549,7 +598,18 @@ void tcp_destroy_cgroup(const struct proto *prot, struct cgroup *cgrp,
    percpu_counter_destroy(&memcg->tcp.tcp_sockets_allocated);
+ }
EXPORT_SYMBOL(tcp_destroy_cgroup);
+
+ unsigned long long tcp_max_memory(const struct mem_cgroup *memcg)
+ {
+ return res_counter_read_u64(&CONSTCG(memcg)->tcp.tcp_memory_allocated,
+ RES_LIMIT);
+ }
#undef CONSTCG
+
+ void tcp_prot_mem(struct mem_cgroup *memcg, long val, int idx)
+ {
+ memcg->tcp.tcp_prot_mem[idx] = val;
+ }
#endif /* CONFIG_INET */
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */

@@ -1048,12 +1108,6 @@ static struct mem_cgroup *mem_cgroup_get_next(struct
mem_cgroup *iter,
#define for_each_mem_cgroup_all(iter) \
    for_each_mem_cgroup_tree_cond(iter, NULL, true)

-
- static inline bool mem_cgroup_is_root(struct mem_cgroup *mem)
- {
- return (mem == root_mem_cgroup);
- }
-
void mem_cgroup_count_vm_event(struct mm_struct *mm, enum vm_event_item idx)
{
    struct mem_cgroup *mem;
@@ -4071,7 +4125,9 @@ static u64 mem_cgroup_read(struct cgroup *cont, struct cftype *cft)
    case _KMEM:
        val = res_counter_read_u64(&mem->kmem, name);
        break;
-
+ case _KMEM_TCP:
+ val = res_counter_read_u64(&mem->tcp.tcp_memory_allocated, name);

```

```

+ break;
default:
    BUG();
    break;
@@ -4104,6 +4160,13 @@ static int mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
    break;
    if (type == _MEM)
        ret = mem_cgroup_resize_limit(memcg, val);
+ #if defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM) && defined(CONFIG_INET)
+ else if (type == _KMEM_TCP) {
+     ret = res_counter_set_limit(&memcg->tcp.tcp_memory_allocated,
+         val);
+     tcp_update_limit(memcg, val);
+ }
+ #endif
    else
        ret = mem_cgroup_resize_mems_w_limit(memcg, val);
    break;
diff --git a/net/ipv4/sysctl_net_ipv4.c b/net/ipv4/sysctl_net_ipv4.c
index bbd67ab..cdc35f6 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/memcontrol.h>
#include <linux/swap.h>
#include <net/snmp.h>
#include <net/icmp.h>
@@ -182,6 +183,10 @@ static int ipv4_tcp_mem(ctl_table *ctl, int write,
    int ret;
    unsigned long vec[3];
    struct net *net = current->nsproxy->net_ns;
+ #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ int i;
+ struct mem_cgroup *cg;
+ #endif

    ctl_table tmp = {
        .data = &vec,
@@ -198,6 +203,21 @@ static int ipv4_tcp_mem(ctl_table *ctl, int write,
    if (ret)
        return ret;

+ #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ rcu_read_lock();
+ cg = mem_cgroup_from_task(current);

```

```
+ for (i = 0; i < 3; i++)
+ if (vec[i] > tcp_max_memory(cg)) {
+   rcu_read_unlock();
+   return -EINVAL;
+ }
+
+ tcp_prot_mem(cg, vec[0], 0);
+ tcp_prot_mem(cg, vec[1], 1);
+ tcp_prot_mem(cg, vec[2], 2);
+ rcu_read_unlock();
+#endif
+
+ net->ipv4.sysctl_tcp_mem[0] = vec[0];
+ net->ipv4.sysctl_tcp_mem[1] = vec[1];
+ net->ipv4.sysctl_tcp_mem[2] = vec[2];
--
```

1.7.6.4

---