Subject: [PATCH v5 07/10] Display current tcp memory allocation in kmem cgroup
Posted by Glauber Costa on Mon, 07 Nov 2011 15:26:32 GMT
View Forum Message <> Reply to Message

This patch introduces kmem.tcp.usage_in_bytes file, living in the
kmem_cgroup filesystem. It is a simple read-only file that displays the
amount of kernel memory currently consumed by the 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 +
 mm/memcontrol.c | 14 ++++++++++---
 2 files changed, 12 insertions(+), 3 deletions(-)

diff --git a/Documentation/cgroups/memory.txt b/Documentation/cgroups/memory.txt
index c1db134..00f1a88 100644
--- a/Documentation/cgroups/memory.txt
+++ b/Documentation/cgroups/memory.txt
@@ -79,6 +79,7 @@ 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
+ memory.kmem.tcp.usage_in_bytes # show current tcp buf memory allocation

 1. History

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index ee122a6..51b5a55 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -528,6 +528,11 @@ static struct cftype tcp_files[] = {
   .read_u64 = mem_cgroup_read,
   .private = MEMFILE_PRIVATE(_KMEM_TCP, RES_LIMIT),
  },
+ {
+ .name = "kmem.tcp.usage_in_bytes",
+ .read_u64 = mem_cgroup_read,
+ .private = MEMFILE_PRIVATE(_KMEM_TCP, RES_USAGE),
+ },
 };

 static void tcp_create_cgroup(struct mem_cgroup *cg, struct cgroup_subsys *ss)
@@ -4126,9 +4131,12 @@ static u64 mem_cgroup_read(struct cgroup *cont, struct cftype *cft)
 #if defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM) && defined(CONFIG_INET)
  case _KMEM_TCP:

```
   /* Be explicit: tcp root does not have a res_counter */
-  if (mem_cgroup_is_root(mem))
-   val = RESOURCE_MAX;
-  else
+  if (mem_cgroup_is_root(mem)) {
+   if (name == RES_USAGE)
+    val = atomic_long_read(&tcp_memory_allocated) << PAGE_SHIFT;
+   else
+    val = RESOURCE_MAX;
+  } else
    val = res_counter_read_u64(&mem->tcp.tcp_memory_allocated, name);
   break;
 #endif
--
1.7.6.4
```