Subject: [PATCH 2/4] pass struct mem_cgroup instead of struct cgroup to socket memcg
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:54 GMT
View Forum Message <> Reply to Message

The only reason cgroup was used, was to be consistent with the populate() interface. Now that we're getting rid of it, not only we no longer need it, but we also *can't* call it this way.

Since we will no longer rely on populate(), this will be called from create(). During create, the association between struct mem_cgroup and struct cgroup does not yet exist, since cgroup internals hasn't yet initialized its bookkeeping. This means we would not be able to draw the memcg pointer from the cgroup pointer in these functions, which is highly undesirable.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Tejun Heo <tj@kernel.org>
---
 include/net/sock.h | 12 ++++++------
 mm/memcontrol.c | 24 ++++++++++----------------
 net/core/sock.c | 10 +++++-----
 net/ipv4/tcp_memcontrol.c | 9 ++++----
 4 files changed, 24 insertions(+), 31 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index 705d1ad..e476277 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -67,16 +67,16 @@
 struct cgroup;
 struct cgroup_subsys;
 #ifdef CONFIG_NET
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss);
-void mem_cgroup_sockets_destroy(struct cgroup *cgrp);
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss);
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg);
 #else
 static inline
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  return 0;
 }
 static inline
-void mem_cgroup_sockets_destroy(struct cgroup *cgrp)
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg)
 {

```
 }
 #endif
@@ -867,9 +867,9 @@ struct proto {
   * This function has to setup any files the protocol want to
   * appear in the kmem cgroup filesystem.
   */
- int   (*init_cgroup)(struct cgroup *cgrp,
+ int   (*init_cgroup)(struct mem_cgroup *memcg,
             struct cgroup_subsys *ss);
- void   (*destroy_cgroup)(struct cgroup *cgrp);
+ void   (*destroy_cgroup)(struct mem_cgroup *memcg);
  struct cg_proto  *(*proto_cgroup)(struct mem_cgroup *memcg);
 #endif
 };
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index f2221ce..f7c6727 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4568,29 +4568,22 @@ static int mem_control_numa_stat_open(struct inode *unused,
struct file *file)
 #endif /* CONFIG_NUMA */

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
-static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
+static int register_kmem_files(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
- /*
-  * Part of this would be better living in a separate allocation
-  * function, leaving us with just the cgroup tree population work.
-  * We, however, depend on state such as network's proto_list that
-  * is only initialized after cgroup creation. I found the less
-  * cumbersome way to deal with it to defer it all to populate time
-  */
- return mem_cgroup_sockets_init(cont, ss);
+ return mem_cgroup_sockets_init(memcg, ss);
 };

-static void kmem_cgroup_destroy(struct cgroup *cont)
+static void kmem_cgroup_destroy(struct mem_cgroup *memcg)
 {
- mem_cgroup_sockets_destroy(cont);
+ mem_cgroup_sockets_destroy(memcg);
 }
 #else
-static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
+static int register_kmem_files(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  return 0;
```

```
 }

-static void kmem_cgroup_destroy(struct cgroup *cont)
+static void kmem_cgroup_destroy(struct mem_cgroup *memcg)
 {
 }
 #endif
@@ -4947,7 +4940,7 @@ static void mem_cgroup_destroy(struct cgroup *cont)
 {
  struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);

- kmem_cgroup_destroy(cont);
+ kmem_cgroup_destroy(memcg);

  mem_cgroup_put(memcg);
 }
@@ -4955,7 +4948,8 @@ static void mem_cgroup_destroy(struct cgroup *cont)
 static int mem_cgroup_populate(struct cgroup_subsys *ss,
     struct cgroup *cont)
 {
- return register_kmem_files(cont, ss);
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
+ return register_kmem_files(memcg, ss);
 }

 #ifdef CONFIG_MMU
diff --git a/net/core/sock.c b/net/core/sock.c
index 688037c..443a404 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -141,7 +141,7 @@ static DEFINE_MUTEX(proto_list_mutex);
 static LIST_HEAD(proto_list);

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  struct proto *proto;
  int ret = 0;
@@ -149,7 +149,7 @@ int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys
*ss)
  mutex_lock(&proto_list_mutex);
  list_for_each_entry(proto, &proto_list, node) {
   if (proto->init_cgroup) {
-   ret = proto->init_cgroup(cgrp, ss);
+   ret = proto->init_cgroup(memcg, ss);
    if (ret)
     goto out;
```

```
   }
@@ -160,19 +160,19 @@ int mem_cgroup_sockets_init(struct cgroup *cgrp, struct
cgroup_subsys *ss)
 out:
  list_for_each_entry_continue_reverse(proto, &proto_list, node)
   if (proto->destroy_cgroup)
-   proto->destroy_cgroup(cgrp);
+   proto->destroy_cgroup(memcg);
  mutex_unlock(&proto_list_mutex);
  return ret;
 }

-void mem_cgroup_sockets_destroy(struct cgroup *cgrp)
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg)
 {
  struct proto *proto;

  mutex_lock(&proto_list_mutex);
  list_for_each_entry_reverse(proto, &proto_list, node)
   if (proto->destroy_cgroup)
-   proto->destroy_cgroup(cgrp);
+   proto->destroy_cgroup(memcg);
  mutex_unlock(&proto_list_mutex);
 }
 #endif
diff --git a/net/ipv4/tcp_memcontrol.c b/net/ipv4/tcp_memcontrol.c
index 21f48f3..94f32ce 100644
--- a/net/ipv4/tcp_memcontrol.c
+++ b/net/ipv4/tcp_memcontrol.c
@@ -51,7 +51,7 @@ static void memcg_tcp_enter_memory_pressure(struct sock *sk)
 }
 EXPORT_SYMBOL(memcg_tcp_enter_memory_pressure);

-int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  /*
   * The root cgroup does not use res_counters, but rather,
@@ -61,8 +61,7 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
  struct res_counter *res_parent = NULL;
  struct cg_proto *cg_proto, *parent_cg;
  struct tcp_memcontrol *tcp;
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
- struct mem_cgroup *parent = parent_mem_cgroup(memcg);
+ struct mem_cgroup *parent;
  struct net *net = current->nsproxy->net_ns;

  cg_proto = tcp_prot.proto_cgroup(memcg);
```

```
@@ -76,6 +75,7 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
  tcp->tcp_prot_mem[2] = net->ipv4.sysctl_tcp_mem[2];
  tcp->tcp_memory_pressure = 0;

+ parent = parent_mem_cgroup(memcg);
  parent_cg = tcp_prot.proto_cgroup(parent);
  if (parent_cg)
   res_parent = parent_cg->memory_allocated;
@@ -94,9 +94,8 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
 }
 EXPORT_SYMBOL(tcp_init_cgroup);

-void tcp_destroy_cgroup(struct cgroup *cgrp)
+void tcp_destroy_cgroup(struct mem_cgroup *memcg)
 {
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
  struct cg_proto *cg_proto;
  struct tcp_memcontrol *tcp;
  u64 val;
--
1.7.7.6
```