Subject: [PATCH 4/4] get rid of populate for memcg
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:56 GMT
View Forum Message <> Reply to Message

The last man standing justifying the need for populate() is the
sock memcg initialization functions. By adopting the convention
that prot->init_cgroup is always called (when existing) with
NULL arguments, we are able to comunicate to the underlying protocol
that it needs to perform global initialization.

We can use that to register cftype files, and then follow the cgroup
rework that gets rid of populate().

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Tejun Heo <tj@kernel.org>
---
 include/net/sock.h | 5 +++++
 include/net/tcp_memcontrol.h | 4 ++--
 mm/memcontrol.c | 12 +++++---------
 net/core/sock.c | 5 +++++
 net/ipv4/tcp_memcontrol.c | 8 +++++++-
 5 files changed, 22 insertions(+), 12 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index e476277..3daf7cd94 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -866,6 +866,11 @@ struct proto {
   * protocols that implement it, from cgroups populate function.
   * This function has to setup any files the protocol want to
   * appear in the kmem cgroup filesystem.
+ *
+ * init_cgroup() will be called upon protocol registering, with
+ * all arguments being NULL. The protocol should act accordingly,
+ * by doing global initialization in case any memcg is already
+ * initialized, like registering its files.
   */
  int   (*init_cgroup)(struct mem_cgroup *memcg,
            struct cgroup_subsys *ss);
diff --git a/include/net/tcp_memcontrol.h b/include/net/tcp_memcontrol.h
index 48410ff..7df18bc 100644
--- a/include/net/tcp_memcontrol.h
+++ b/include/net/tcp_memcontrol.h
@@ -12,8 +12,8 @@ struct tcp_memcontrol {
 };

 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);
+int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys *ss);
+void tcp_destroy_cgroup(struct mem_cgroup *memcg);
 unsigned long long tcp_max_memory(const struct mem_cgroup *memcg);
 void tcp_prot_mem(struct mem_cgroup *memcg, long val, int idx);
 #endif /* _TCP_MEMCG_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index bd91a46..87a1e21 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4929,7 +4929,9 @@ mem_cgroup_create(struct cgroup *cont)
  atomic_set(&memcg->refcnt, 1);
  memcg->move_charge_at_immigrate = 0;
  mutex_init(&memcg->thresholds_lock);
- return &memcg->css;
+
+ if (!register_kmem_files(memcg, &mem_cgroup_subsys))
+  return &memcg->css;
 free_out:
  __mem_cgroup_free(memcg);
  return ERR_PTR(error);
@@ -4951,13 +4953,6 @@ static void mem_cgroup_destroy(struct cgroup *cont)
  mem_cgroup_put(memcg);
 }

-static int mem_cgroup_populate(struct cgroup_subsys *ss,
-    struct cgroup *cont)
-{
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
- return register_kmem_files(memcg, ss);
-}
-
 #ifdef CONFIG_MMU
 /* Handlers for move charge at task migration. */
 #define PRECHARGE_COUNT_AT_ONCE 256
@@ -5465,7 +5460,6 @@ struct cgroup_subsys mem_cgroup_subsys = {
  .create = mem_cgroup_create,
  .pre_destroy = mem_cgroup_pre_destroy,
  .destroy = mem_cgroup_destroy,
- .populate = mem_cgroup_populate,
  .can_attach = mem_cgroup_can_attach,
  .cancel_attach = mem_cgroup_cancel_attach,
  .attach = mem_cgroup_move_task,
diff --git a/net/core/sock.c b/net/core/sock.c
index 443a404..1d60538 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -2484,6 +2484,11 @@ int proto_register(struct proto *prot, int alloc_slab)
```

```
  }
 }

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (prot->init_cgroup)
+  prot->init_cgroup(NULL, NULL);
+#endif
+
  mutex_lock(&proto_list_mutex);
  list_add(&prot->node, &proto_list);
  assign_proto_idx(prot);
diff --git a/net/ipv4/tcp_memcontrol.c b/net/ipv4/tcp_memcontrol.c
index 94f32ce..b8cdf50 100644
--- a/net/ipv4/tcp_memcontrol.c
+++ b/net/ipv4/tcp_memcontrol.c
@@ -37,7 +37,6 @@ static struct cftype tcp_files[] = {
  },
  { } /* terminate */
 };
-CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files);

 static inline struct tcp_memcontrol *tcp_from_cgproto(struct cg_proto *cg_proto)
 {
@@ -64,6 +63,13 @@ int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys
*ss)
  struct mem_cgroup *parent;
  struct net *net = current->nsproxy->net_ns;

+ if (!memcg && !ss)
+  return mem_cgroup_register_cftype(tcp_files);
+
+ if (WARN_ON(!memcg || !ss)) {
+  return -1;
+ }
+
  cg_proto = tcp_prot.proto_cgroup(memcg);
  if (!cg_proto)
   return 0;
--
1.7.7.6
```