Subject: [PATCH 08/10] memcg: Add CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT.
Posted by Suleiman Souhlal on Mon, 27 Feb 2012 22:58:51 GMT
View Forum Message <> Reply to Message

This config option dictates whether or not kernel memory in the
root cgroup should be accounted.

This may be useful in an environment where everything is supposed to be
in a cgroup and accounted for. Large amounts of kernel memory in the
root cgroup would indicate problems with memory isolation or accounting.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
---
 init/Kconfig | 8 ++++++++
 mm/memcontrol.c | 44 ++
 2 files changed, 52 insertions(+), 0 deletions(-)

diff --git a/init/Kconfig b/init/Kconfig
index 3f42cd6..a119270 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -714,6 +714,14 @@ config CGROUP_MEM_RES_CTLR_KMEM
    Memory Controller, which are page-based, and can be swapped. Users of
    the kmem extension can use it to guarantee that no group of processes
    will ever exhaust kernel resources alone.
+config CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+ bool "Root Cgroup Kernel Memory Accounting (EXPERIMENTAL)"
+ depends on CGROUP_MEM_RES_CTLR_KMEM
+ default n
+ help
+   Account for kernel memory used by the root cgroup. This may be useful
+   to know how much kernel memory isn't currently accounted to any
+   cgroup.

 config CGROUP_PERF
   bool "Enable perf_event per-cpu per-container group (cgroup) monitoring"
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 6a475ed..d4cdb8e 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -61,6 +61,10 @@ struct cgroup_subsys mem_cgroup_subsys __read_mostly;
 #define MEM_CGROUP_RECLAIM_RETRIES 5
 struct mem_cgroup *root_mem_cgroup __read_mostly;

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+atomic64_t pre_memcg_kmem_bytes; /* kmem usage before memcg is enabled */
+#endif

```
+
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_SWAP
 /* Turned on only when memory cgroup is enabled && really_do_swap_account = 1 */
 int do_swap_account __read_mostly;
@@ -5643,6 +5647,13 @@ memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, long
long delta)

  if (memcg)
   ret = res_counter_charge(&memcg->kmem_bytes, delta, &fail_res);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+ else if (root_mem_cgroup != NULL)
+  ret = res_counter_charge(&root_mem_cgroup->kmem_bytes, delta,
+      &fail_res);
+ else
+  atomic64_add(delta, &pre_memcg_kmem_bytes);
+#endif

  return ret;
 }
@@ -5668,6 +5679,12 @@ memcg_uncharge_kmem(struct mem_cgroup *memcg, long long
delta)

  if (memcg)
   res_counter_uncharge(&memcg->kmem_bytes, delta);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+ else if (root_mem_cgroup != NULL)
+  res_counter_uncharge(&root_mem_cgroup->kmem_bytes, delta);
+ else
+  atomic64_sub(delta, &pre_memcg_kmem_bytes);
+#endif

  if (memcg && !memcg->independent_kmem_limit)
   res_counter_uncharge(&memcg->res, delta);
@@ -5953,7 +5970,12 @@ memcg_slab_move(struct mem_cgroup *memcg)
   cachep = rcu_access_pointer(memcg->slabs[i]);
   if (cachep != NULL) {
    rcu_assign_pointer(memcg->slabs[i], NULL);
+
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+   cachep->memcg = root_mem_cgroup;
+#else
    cachep->memcg = NULL;
+#endif

    /* The space for this is already allocated */
    strcat((char *)cachep->name, "dead");
@@ -5991,6 +6013,15 @@ memcg_kmem_init(struct mem_cgroup *memcg, struct mem_cgroup
*parent)
```

```
  memcg_slab_init(memcg);

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+ if (memcg == root_mem_cgroup) {
+  long kmem_bytes;
+
+  kmem_bytes = atomic64_xchg(&pre_memcg_kmem_bytes, 0);
+  memcg->kmem_bytes.usage = kmem_bytes;
+ }
+#endif
+
  atomic64_set(&memcg->kmem_bypassed, 0);
  memcg->independent_kmem_limit = 0;
 }
@@ -6010,6 +6041,19 @@ memcg_kmem_move(struct mem_cgroup *memcg)
  spin_unlock_irqrestore(&memcg->kmem_bytes.lock, flags);
  if (!memcg->independent_kmem_limit)
   res_counter_uncharge(&memcg->res, kmem_bytes);
+
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT
+ {
+  struct res_counter *dummy;
+  int err;
+
+  /* Can't fail because it's the root cgroup */
+  err = res_counter_charge(&root_mem_cgroup->kmem_bytes,
+      kmem_bytes, &dummy);
+  err = res_counter_charge(&root_mem_cgroup->res, kmem_bytes,
+      &dummy);
+ }
+#endif
 }
 #else /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
 static void
--
1.7.7.3
```