Subject: [PATCH 24/33] memory controller accounting setup v7
Posted by Paul Menage on Mon, 17 Sep 2007 21:03:31 GMT
View Forum Message <> Reply to Message

From: Pavel Emelianov <xemul@openvz.org>
(container->cgroup renaming by Paul Menage <menage@google.com>)

Basic setup routines, the mm_struct has a pointer to the cgroup that
it belongs to and the the page has a page_cgroup associated with it.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
Signed-off-by: Paul Menage <menage@google.com>
---

include/linux/memcontrol.h | 36 ++++++++++++++++++++++
include/linux/mm_types.h | 6 +++
include/linux/sched.h | 1
kernel/fork.c | 11 ++++--
mm/memcontrol.c | 57 ++++++++++++++++++++++++++++++++++---
5 files changed, 104 insertions(+), 7 deletions(-)

diff -puN include/linux/memcontrol.h~memory-controller-accounting-setup-v7
include/linux/memcontrol.h
--- a/include/linux/memcontrol.h~memory-controller-accounting-setup-v7
+++ a/include/linux/memcontrol.h
@@ -3,6 +3,9 @@
* Copyright IBM Corporation, 2007
* Author Balbir Singh <balbir@linux.vnet.ibm.com>
*
+ * Copyright 2007 OpenVZ SWsoft Inc
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
@@ -17,5 +20,38 @@
#ifndef _LINUX_MEMCONTROL_H
#define _LINUX_MEMCONTROL_H

+struct mem_cgroup;
+struct page_cgroup;
+
+#ifdef CONFIG_CGROUP_MEM_CONT
+
+extern void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p);
+extern void mm_free_cgroup(struct mm_struct *mm);
+extern void page_assign_page_cgroup(struct page *page,

```
+     struct page_cgroup *pc);
+extern struct page_cgroup *page_get_page_cgroup(struct page *page);
+
+#else /* CONFIG_CGROUP_MEM_CONT */
+static inline void mm_init_cgroup(struct mm_struct *mm,
+     struct task_struct *p)
+{
+}
+
+static inline void mm_free_cgroup(struct mm_struct *mm)
+{
+}
+
+static inline void page_assign_page_cgroup(struct page *page,
+      struct page_cgroup *pc)
+{
+}
+
+static inline struct page_cgroup *page_get_page_cgroup(struct page *page)
+{
+ return NULL;
+}
+
+#endif /* CONFIG_CGROUP_MEM_CONT */
+
 #endif /* _LINUX_MEMCONTROL_H */

diff -puN include/linux/mm_types.h~memory-controller-accounting-setup-v7
include/linux/mm_types.h
--- a/include/linux/mm_types.h~memory-controller-accounting-setup-v7
+++ a/include/linux/mm_types.h
@@ -83,6 +83,9 @@ struct page {
  void *virtual;   /* Kernel virtual address (NULL if
        not kmapped, ie. highmem) */
 #endif /* WANT_PAGE_VIRTUAL */
+#ifdef CONFIG_CGROUP_MEM_CONT
+ unsigned long page_cgroup;
+#endif
 };

 /*
@@ -214,6 +217,9 @@ struct mm_struct {
  /* aio bits */
  rwlock_t  ioctx_list_lock;
  struct kioctx  *ioctx_list;
+#ifdef CONFIG_CGROUP_MEM_CONT
+ struct mem_cgroup *mem_cgroup;
+#endif
```

```
 };

 #endif /* _LINUX_MM_TYPES_H */
diff -puN include/linux/sched.h~memory-controller-accounting-setup-v7 include/linux/sched.h
--- a/include/linux/sched.h~memory-controller-accounting-setup-v7
+++ a/include/linux/sched.h
@@ -88,6 +88,7 @@ struct sched_param {

 #include <asm/processor.h>

+struct mem_cgroup;
 struct exec_domain;
 struct futex_pi_state;
 struct bio;
diff -puN kernel/fork.c~memory-controller-accounting-setup-v7 kernel/fork.c
--- a/kernel/fork.c~memory-controller-accounting-setup-v7
+++ a/kernel/fork.c
@@ -51,6 +51,7 @@
 #include <linux/random.h>
 #include <linux/tty.h>
 #include <linux/proc_fs.h>
+#include <linux/memcontrol.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -329,7 +330,7 @@ __cacheline_aligned_in_smp DEFINE_SPINLO

 #include <linux/init_task.h>

-static struct mm_struct * mm_init(struct mm_struct * mm)
+static struct mm_struct * mm_init(struct mm_struct * mm, struct task_struct *p)
 {
  atomic_set(&mm->mm_users, 1);
  atomic_set(&mm->mm_count, 1);
@@ -346,11 +347,14 @@ static struct mm_struct * mm_init(struct
  mm->ioctx_list = NULL;
  mm->free_area_cache = TASK_UNMAPPED_BASE;
  mm->cached_hole_size = ~0UL;
+ mm_init_cgroup(mm, p);

  if (likely(!mm_alloc_pgd(mm))) {
   mm->def_flags = 0;
   return mm;
  }
+
+ mm_free_cgroup(mm);
  free_mm(mm);
  return NULL;
```

```
 }
@@ -365,7 +369,7 @@ struct mm_struct * mm_alloc(void)
  mm = allocate_mm();
  if (mm) {
   memset(mm, 0, sizeof(*mm));
-  mm = mm_init(mm);
+  mm = mm_init(mm, current);
  }
  return mm;
 }
@@ -379,6 +383,7 @@ void fastcall __mmdrop(struct mm_struct
 {
  BUG_ON(mm == &init_mm);
  mm_free_pgd(mm);
+ mm_free_cgroup(mm);
  destroy_context(mm);
  free_mm(mm);
 }
@@ -499,7 +504,7 @@ static struct mm_struct *dup_mm(struct t
  mm->token_priority = 0;
  mm->last_interval = 0;

- if (!mm_init(mm))
+ if (!mm_init(mm, tsk))
   goto fail_nomem;

  if (init_new_context(tsk, mm))
diff -puN mm/memcontrol.c~memory-controller-accounting-setup-v7 mm/memcontrol.c
--- a/mm/memcontrol.c~memory-controller-accounting-setup-v7
+++ a/mm/memcontrol.c
@@ -3,6 +3,9 @@
  * Copyright IBM Corporation, 2007
  * Author Balbir Singh <balbir@linux.vnet.ibm.com>
  *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
  * This program is free software; you can redistribute it and/or modify
  * it under the terms of the GNU General Public License as published by
  * the Free Software Foundation; either version 2 of the License, or
@@ -17,6 +20,7 @@
 #include <linux/res_counter.h>
 #include <linux/memcontrol.h>
 #include <linux/cgroup.h>
+#include <linux/mm.h>

 struct cgroup_subsys mem_cgroup_subsys;
```

```
@@ -35,6 +39,13 @@ struct mem_cgroup {
   * the counter to account for memory usage
   */
  struct res_counter res;
+ /*
+  * Per cgroup active and inactive list, similar to the
+  * per zone LRU lists.
+  * TODO: Consider making these lists per zone
+  */
+ struct list_head active_list;
+ struct list_head inactive_list;
 };

 /*
@@ -56,6 +67,37 @@ struct mem_cgroup *mem_cgroup_from
     css);
 }

+static inline
+struct mem_cgroup *mem_cgroup_from_task(struct task_struct *p)
+{
+ return container_of(task_subsys_state(p, mem_cgroup_subsys_id),
+     struct mem_cgroup, css);
+}
+
+void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p)
+{
+ struct mem_cgroup *mem;
+
+ mem = mem_cgroup_from_task(p);
+ css_get(&mem->css);
+ mm->mem_cgroup = mem;
+}
+
+void mm_free_cgroup(struct mm_struct *mm)
+{
+ css_put(&mm->mem_cgroup->css);
+}
+
+void page_assign_page_cgroup(struct page *page, struct page_cgroup *pc)
+{
+ page->page_cgroup = (unsigned long)pc;
+}
+
+struct page_cgroup *page_get_page_cgroup(struct page *page)
+{
+ return page->page_cgroup;
+}
```

```
+
 static ssize_t mem_cgroup_read(struct cgroup *cont, struct cftype *cft,
    struct file *file, char __user *userbuf, size_t nbytes,
    loff_t *ppos)
@@ -91,14 +133,21 @@ static struct cftype mem_cgroup_files
  },
 };

+static struct mem_cgroup init_mem_cgroup;
+
 static struct cgroup_subsys_state *
 mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
 {
  struct mem_cgroup *mem;

- mem = kzalloc(sizeof(struct mem_cgroup), GFP_KERNEL);
- if (!mem)
-  return -ENOMEM;
+ if (unlikely((cont->parent) == NULL)) {
+  mem = &init_mem_cgroup;
+  init_mm.mem_cgroup = mem;
+ } else
+  mem = kzalloc(sizeof(struct mem_cgroup), GFP_KERNEL);
+
+ if (mem == NULL)
+  return NULL;

  res_counter_init(&mem->res);
  return &mem->css;
@@ -123,5 +172,5 @@ struct cgroup_subsys mem_cgroup_su
  .create = mem_cgroup_create,
  .destroy = mem_cgroup_destroy,
  .populate = mem_cgroup_populate,
- .early_init = 0,
+ .early_init = 1,
 };
_

--
```