
Subject: [PATCH 3/8] Add container pointer on mm_struct
Posted by [Pavel Emelianov](#) on Wed, 30 May 2007 15:24:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

Naturally mm_struct determines the resource consumer in memory accounting. So each mm_struct should have a pointer on container it belongs to. When a new task is created its mm_struct is assigned to the container this task belongs to.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff -upr linux-2.6.22-rc2-mm1.orig/include/linux/sched.h
linux-2.6.22-rc2-mm1-0/include/linux/sched.h
--- linux-2.6.22-rc2-mm1.orig/include/linux/sched.h 2007-05-30 16:13:08.000000000 +0400
+++ linux-2.6.22-rc2-mm1-0/include/linux/sched.h 2007-05-30 16:13:09.000000000 +0400
@@ -390,6 +390,9 @@ struct mm_struct {
    /* aio bits */
    rwlock_t ioctx_list_lock;
    struct kiocb *ioctx_list;
+ #ifdef CONFIG_RSS_CONTAINER
+ struct rss_container *rss_container;
+ #endif
};

struct sighand_struct {
diff -upr linux-2.6.22-rc2-mm1.orig/kernel/fork.c linux-2.6.22-rc2-mm1-0/kernel/fork.c
--- linux-2.6.22-rc2-mm1.orig/kernel/fork.c 2007-05-30 16:13:08.000000000 +0400
+++ linux-2.6.22-rc2-mm1-0/kernel/fork.c 2007-05-30 16:13:09.000000000 +0400
@@ -57,6 +57,8 @@
#include <asm/cacheflush.h>
#include <asm/tlbflush.h>

+ #include <linux/rss_container.h>
+
/*
 * Protected counters by write_lock_irq(&tasklist_lock)
 */
@@ -329,7 +331,7 @@ static inline void mm_free_pgd(struct mm

#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 *tsk)
{
    atomic_set(&mm->mm_users, 1);
    atomic_set(&mm->mm_count, 1);
```

```

@@ -344,11 +346,14 @@ static struct mm_struct * mm_init(struct
    mm->iocctx_list = NULL;
    mm->free_area_cache = TASK_UNMAPPED_BASE;
    mm->cached_hole_size = ~0UL;
+ mm_init_container(mm, tsk);

    if (likely(!mm_alloc_pgd(mm))) {
        mm->def_flags = 0;
        return mm;
    }
+
+ mm_free_container(mm);
    free_mm(mm);
    return NULL;
}
@@ -363,7 +368,7 @@ 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;
}
@@ -377,6 +382,7 @@ void fastcall __mmdrop(struct mm_struct
{
    BUG_ON(mm == &init_mm);
    mm_free_pgd(mm);
+ mm_free_container(mm);
    destroy_context(mm);
    free_mm(mm);
}
@@ -497,7 +503,7 @@ 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))
@@ -524,6 +530,7 @@ fail_nocontext:
    * because it calls destroy_context()
    */
    mm_free_pgd(mm);
+ mm_free_container(mm);
    free_mm(mm);
    return NULL;

```

}
