
Subject: Re: [PATCH 6/7] containers (V7): BeanCounters over generic process containers

Posted by [xemul](#) on Tue, 13 Feb 2007 09:49:36 GMT

[View Forum Message](#) <> [Reply to Message](#)

Paul Menage wrote:

> On 2/13/07, Pavel Emelianov <xemul@sw.ru> wrote:

>>

>> I have implementation that moves arbitrary task :)

>

> Is that the one that calls stop_machine() in order to move a task

> around? That seemed a little heavyweight ...

Nope :) I've rewritten it completely.

>> May be we can do context (container-on-task) handling lockless?

>

> What did you have in mind?

The example patch is attached. Fits 2.6.20-rc6-mm3.

>> > I thought that we solved that problem by having a tmp_bc field in the

>> > task_struct that would take precedence over the main bc if it was

>> > non-null?

>>

>> Of course, but I'm commenting this patchset which doesn't have

>> this facility.

>

> OK, I can add the concept in to the example too.

>

> Paul

>

--- ./kernel/bc/misc.c.bcctx 2007-01-31 13:56:45.000000000 +0300

+++ ./kernel/bc/misc.c 2007-01-31 14:20:32.000000000 +0300

@@ -0,0 +1,63 @@

+/*

+ * kernel/bc/misc.c

+ *

+ * Copyright (C) 2007 OpenVZ SWsoft Inc

+ *

+ */

+

+#include <linux/sched.h>

+#include <linux/stop_machine.h>

+#include <linux/module.h>

+

```

+ #include <bc/beancounter.h>
+ #include <bc/task.h>
+ #include <bc/misc.h>
+
+ static DEFINE_MUTEX(task_move_mutex);
+
+ int copy_beancounter(struct task_struct *tsk, struct task_struct *parent)
+ {
+     struct beancounter *bc;
+
+     bc = parent->exec_bc;
+     tsk->exec_bc = bc_get(bc);
+     BUG_ON(tsk->tmp_exec_bc != NULL);
+     return 0;
+ }
+
+ void free_beancounter(struct task_struct *tsk)
+ {
+     struct beancounter *bc;
+
+     BUG_ON(tsk->tmp_exec_bc != NULL);
+     bc = tsk->exec_bc;
+     bc_put(bc);
+ }
+
+ int bc_task_move(int pid, struct beancounter *bc)
+ {
+     struct task_struct *tsk;
+     struct beancounter *old_bc;
+
+     read_lock(&tasklist_lock);
+     tsk = find_task_by_pid(pid);
+     if (tsk)
+         get_task_struct(tsk);
+     read_unlock(&tasklist_lock);
+     if (tsk == NULL)
+         return -ESRCH;
+
+     mutex_lock(&task_move_mutex);
+     old_bc = tsk->exec_bc;
+
+     bc_get(bc);
+     rcu_assign_pointer(tsk->exec_bc, bc);
+
+     /* wait for all users if any get this beancounter */
+     synchronize_rcu();
+     mutex_unlock(&task_move_mutex);
+     bc_put(old_bc);

```

```

+
+ return err;
+}
+EXPORT_SYMBOL(bc_task_move);
--- ./kernel/fork.c.bcctx 2007-01-31 13:35:21.000000000 +0300
+++ ./kernel/fork.c 2007-01-31 13:56:45.000000000 +0300
@@ -51,6 +51,8 @@
#include <linux/random.h>
#include <linux/user_namespace.h>

+#include <bc/task.h>
+
#include <asm/pgtable.h>
#include <asm/pgalloc.h>
#include <asm/uaccess.h>
@@ -105,12 +107,18 @@ struct kmem_cache *vm_area_cachep;
/* SLAB cache for mm_struct structures (tsk->mm) */
static struct kmem_cache *mm_cachep;

-void free_task(struct task_struct *tsk)
+static void __free_task(struct task_struct *tsk)
{
    free_thread_info(tsk->thread_info);
    rt_mutex_debug_task_free(tsk);
    free_task_struct(tsk);
}
+
+void free_task(struct task_struct *tsk)
+{
+    free_beancounter(tsk);
+    __free_task(tsk);
+}
EXPORT_SYMBOL(free_task);

void __put_task_struct(struct task_struct *tsk)
@@ -999,6 +1007,10 @@ static struct task_struct *copy_process(

    rt_mutex_init_task(p);

+    retval = copy_beancounter(p, current);
+    if (retval < 0)
+        goto bad_fork_bc;
+
#ifdef CONFIG_TRACE_IRQFLAGS
    DEBUG_LOCKS_WARN_ON(!p->hardirqs_enabled);
    DEBUG_LOCKS_WARN_ON(!p->softirqs_enabled);
@@ -1321,7 +1333,9 @@ bad_fork_cleanup_count:
    atomic_dec(&p->user->processes);

```

```

    free_uid(p->user);
bad_fork_free:
- free_task(p);
+ free_beancounter(p);
+bad_fork_bc:
+ __free_task(p);
fork_out:
    return ERR_PTR(retval);
}
--- ./kernel/softirq.c.bcctx 2007-01-31 13:35:21.000000000 +0300
+++ ./kernel/softirq.c 2007-01-31 14:22:44.000000000 +0300
@@ -19,6 +19,8 @@
#include <linux/smp.h>
#include <linux/tick.h>

+#include <bc/task.h>
+
#include <asm/irq.h>
/*
- No shared variables, all the data are CPU local.
@@ -210,6 +212,7 @@ asmlinkage void __do_softirq(void)
    __u32 pending;
    int max_restart = MAX_SOFTIRQ_RESTART;
    int cpu;
+ struct beancounter *bc;

    pending = local_softirq_pending();
    account_system_vtime(current);
@@ -226,6 +229,7 @@ restart:

    h = softirq_vec;

+ bc = set_exec_bc(&init_bc);
do {
    if (pending & 1) {
        h->action(h);
@@ -234,6 +238,7 @@ restart:
        h++;
        pending >>= 1;
    } while (pending);
+ reset_exec_bc(bc, &init_bc);

    local_irq_disable();

--- ./include/linux/sched.h.bcctx 2007-01-31 13:35:21.000000000 +0300
+++ ./include/linux/sched.h 2007-01-31 14:06:28.000000000 +0300
@@ -1082,6 +1082,10 @@ struct task_struct {
#ifdef CONFIG_FAULT_INJECTION

```

```

    int make_it_fail;
#endif
#ifdef CONFIG_BEANCOUNTERS
+ struct beancounter *exec_bc;
+ struct beancounter *tmp_exec_bc;
#endif
};

static inline pid_t process_group(struct task_struct *tsk)
--- ./include/bc/task.h.bcctx 2007-01-31 13:56:45.000000000 +0300
+++ ./include/bc/task.h 2007-01-31 14:19:33.000000000 +0300
@@ -0,0 +1,68 @@
+/*
+ * include/bc/task.h
+ *
+ * Copyright (C) 2007 OpenVZ SWsoft Inc
+ *
+ */
+
+#ifndef __BC_TASK_H__
+#define __BC_TASK_H__
+
+struct beancounter;
+struct task_struct;
+
+#ifdef CONFIG_BEANCOUNTERS
+extern struct beancounter init_bc;
+
+/*
+ * Caller must be in rcu_read safe section
+ */
+static inline struct beancounter *get_exec_bc(void)
+{
+ struct task_struct *tsk;
+
+ if (in_irq())
+ return &init_bc;
+
+ tsk = current;
+ if (tsk->tmp_exec_bc != NULL)
+ return tsk->tmp_exec_bc;
+
+ return rcu_dereference(tsk->exec_bc);
+}
+
+#define set_exec_bc(bc) ({ \
+ struct task_struct *t; \
+ struct beancounter *old; \

```

```

+ t = current; \
+ old = t->tmp_exec_bc; \
+ t->tmp_exec_bc = bc; \
+ old; \
+ })
+
+#define reset_exec_bc(old, expected) do { \
+ struct task_struct *t; \
+ t = current; \
+ BUG_ON(t->tmp_exec_bc != expected); \
+ t->tmp_exec_bc = old; \
+ } while (0)
+
+int __must_check copy_beancounter(struct task_struct *tsk,
+ struct task_struct *parent);
+void free_beancounter(struct task_struct *tsk);
+int bc_task_move(int pid, struct beancounter *bc);
+
+#else
+static inline int __must_check copy_beancounter(struct task_struct *tsk,
+ struct task_struct *parent)
+{
+ return 0;
+}
+
+static inline void free_beancounter(struct task_struct *tsk)
+{
+}
+
+#define set_exec_bc(bc) (NULL)
+#define reset_exec_bc(bc, exp) do { } while (0)
+#endif
+#endif

```
