Subject: [PATCH 4/10] BC: context inheriting and changing
Posted by Kirill Korotaev on Thu, 05 Oct 2006 15:50:05 GMT
View Forum Message <> Reply to Message

Contains code responsible for setting BC on task,
it's inheriting and setting host context in interrupts.

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

```
 include/bc/task.h     |   51 ++++++++++++++++++++++++++++++++++++++++++++++++
 include/linux/sched.h |    5 +++
 kernel/bc/misc.c      |   70 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 kernel/fork.c         |    5 +++
 kernel/irq/handle.c   |    6 ++++
 kernel/softirq.c      |    5 +++
 6 files changed, 142 insertions(+)

--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./include/bc/task.h 2006-10-05 12:11:44.000000000 +0400
@@ -0,0 +1,51 @@
+/*
+ * include/bc/beancounter.h
+ *
+ * Copyright (C) 2006 OpenVZ SWsoft Inc
+ *
+ */
+
+#ifndef __BC_TASK_H__
+#define __BC_TASK_H__
+
+struct beancounter;
+struct task_struct;
+
+#ifdef CONFIG_BEANCOUNTERS
+#define get_exec_bc() (current->exec_bc)
+
+#define set_exec_bc(bc) ({   \
+  struct task_struct *t;  \
+  struct beancounter *old; \
+  t = current;   \
+  old = t->exec_bc;  \
+  t->exec_bc = bc;  \
+  old;    \
+ })
+
```

```
+#define reset_exec_bc(old, expected) do { \
+  struct task_struct *t;  \
+  t = current;   \
+  BUG_ON(t->exec_bc != expected);\
+  t->exec_bc = old;  \
+ } while (0)
+
+extern struct beancounter init_bc;
+
+void copy_beancounter(struct task_struct *tsk, struct task_struct *parent);
+void free_beancounter(struct task_struct *tsk);
+int bc_task_move(struct task_struct *tsk, struct beancounter *bc);
+#else
+static inline void copy_beancounter(struct task_struct *tsk,
+  struct task_struct *parent)
+{
+}
+
+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
--- ./include/linux/sched.h.bc_task 2006-10-05 11:42:43.000000000 +0400
+++ ./include/linux/sched.h 2006-10-05 12:11:44.000000000 +0400
@@ -77,6 +77,8 @@ struct sched_param {
 #include <linux/futex.h>
 #include <linux/rtmutex.h>

+#include <bc/task.h>
+
 #include <linux/time.h>
 #include <linux/param.h>
 #include <linux/resource.h>
@@ -1055,6 +1057,9 @@ struct task_struct {
 #ifdef CONFIG_TASK_DELAY_ACCT
  struct task_delay_info *delays;
 #endif
+#ifdef CONFIG_BEANCOUNTERS
+ struct beancounter *exec_bc;
+#endif
 };

 static inline pid_t process_group(struct task_struct *tsk)
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
```

```
+++ ./kernel/bc/misc.c 2006-10-05 12:11:44.000000000 +0400
@@ -0,0 +1,70 @@
+/*
+ * kernel/bc/misc.c
+ *
+ * Copyright (C) 2006 OpenVZ SWsoft Inc
+ *
+ */
+
+#include <linux/sched.h>
+#include <linux/stop_machine.h>
+
+#include <bc/beancounter.h>
+#include <bc/task.h>
+
+void copy_beancounter(struct task_struct *tsk, struct task_struct *parent)
+{
+ tsk->exec_bc = bc_get(parent->exec_bc);
+}
+
+void free_beancounter(struct task_struct *tsk)
+{
+ bc_put(tsk->exec_bc);
+ tsk->exec_bc = NULL;
+}
+
+struct set_bcid_data {
+ struct beancounter *bc;
+ struct mm_struct *mm;
+};
+
+static int do_set_bcid(void *data)
+{
+ struct set_bcid_data *d;
+ struct mm_struct *mm;
+ struct task_struct *g, *p;
+
+ d = (struct set_bcid_data *)data;
+ mm = d->mm;
+
+ do_each_thread (g, p) {
+  if (p->mm == mm) {
+   bc_put(p->exec_bc);
+   p->exec_bc = bc_get(d->bc);
+  }
+ } while_each_thread (g, p);
+
+ bc_put(mm->mm_bc);
```

```
+ mm->mm_bc = bc_get(d->bc);
+ return 0;
+}
+
+int bc_task_move(struct task_struct *tsk, struct beancounter *bc)
+{
+ int err;
+ struct set_bcid_data data;
+ struct mm_struct *mm;
+
+ mm = get_task_mm(tsk);
+ if (mm == NULL)
+  return -EINVAL;
+
+ data.bc = bc;
+ data.mm = mm;
+
+ down_write(&mm->mmap_sem);
+ err = stop_machine_run(do_set_bcid, &data, NR_CPUS);
+ up_write(&mm->mmap_sem);
+
+ mmput(mm);
+ return err;
+}
--- ./kernel/fork.c.bc_task 2006-10-05 11:42:43.000000000 +0400
+++ ./kernel/fork.c 2006-10-05 12:11:44.000000000 +0400
@@ -49,6 +49,8 @@
 #include <linux/taskstats_kern.h>
 #include <linux/random.h>

+#include <bc/task.h>
+
 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
 #include <asm/uaccess.h>
@@ -105,6 +107,7 @@ static kmem_cache_t *mm_cachep;

 void free_task(struct task_struct *tsk)
 {
+ free_beancounter(tsk);
  free_thread_info(tsk->thread_info);
  rt_mutex_debug_task_free(tsk);
  free_task_struct(tsk);
@@ -984,6 +987,8 @@ static struct task_struct *copy_process(
  if (!p)
   goto fork_out;

+ copy_beancounter(p, current);
```

```
+
 #ifdef CONFIG_TRACE_IRQFLAGS
  DEBUG_LOCKS_WARN_ON(!p->hardirqs_enabled);
  DEBUG_LOCKS_WARN_ON(!p->softirqs_enabled);
--- ./kernel/irq/handle.c.bc_task 2006-10-05 11:42:43.000000000 +0400
+++ ./kernel/irq/handle.c 2006-10-05 12:11:44.000000000 +0400
@@ -16,6 +16,8 @@
 #include <linux/interrupt.h>
 #include <linux/kernel_stat.h>

+#include <bc/task.h>
+
 #include "internals.h"

 /**
@@ -172,6 +174,7 @@ fastcall unsigned int __do_IRQ(unsigned
  struct irq_desc *desc = irq_desc + irq;
  struct irqaction *action;
  unsigned int status;
+ struct beancounter *bc;

  kstat_this_cpu.irqs[irq]++;
  if (CHECK_IRQ_PER_CPU(desc->status)) {
@@ -228,6 +231,8 @@ fastcall unsigned int __do_IRQ(unsigned
   * useful for irq hardware that does not mask cleanly in an
   * SMP environment.
   */
+
+ bc = set_exec_bc(&init_bc);
  for (;;) {
   irqreturn_t action_ret;

@@ -242,6 +247,7 @@ fastcall unsigned int __do_IRQ(unsigned
    break;
   desc->status &= ~IRQ_PENDING;
  }
+ reset_exec_bc(bc, &init_bc);
  desc->status &= ~IRQ_INPROGRESS;

 out:
--- ./kernel/softirq.c.bc_task 2006-10-05 11:42:43.000000000 +0400
+++ ./kernel/softirq.c 2006-10-05 12:11:44.000000000 +0400
@@ -18,6 +18,8 @@
 #include <linux/rcupdate.h>
 #include <linux/smp.h>

+#include <bc/task.h>
+
```

```
 #include <asm/irq.h>
 /*
    - No shared variables, all the data are CPU local.
@@ -209,6 +211,7 @@ 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);
@@ -225,6 +228,7 @@ restart:

  h = softirq_vec;

+ bc = set_exec_bc(&init_bc);
  do {
   if (pending & 1) {
    h->action(h);
@@ -233,6 +237,7 @@ restart:
   h++;
   pending >>= 1;
  } while (pending);
+ reset_exec_bc(bc, &init_bc);

  local_irq_disable();
```