Subject: [PATCH 4/13] BC: context handling
Posted by dev on Thu, 09 Nov 2006 16:51:08 GMT
View Forum Message <> Reply to Message

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

Task references beancounter by exec_bc pointer:
exec_bc: current context. All resources are
charged to this beancounter.

Signed-off-by: Pavel Emelianov <xemul@sw.ru>
Signed-off-by: Kirill Korotaev <dev@sw.ru>

```
 arch/i386/kernel/irq.c  |    5 ++
 fs/proc/array.c         |    6 ++
 include/bc/task.h       |   53 ++++++++++++++++++++++++++
 include/linux/sched.h   |    5 ++
 kernel/bc/beancounter.c |    2
 kernel/bc/misc.c        |  100 ++++++++++++++++++++++++++++++++++++++++++++++++
 kernel/fork.c           |   18 +++++++-
 kernel/irq/handle.c     |    6 ++
 kernel/softirq.c        |    5 ++
 9 files changed, 198 insertions(+), 2 deletions(-)

--- ./arch/i386/kernel/irq.c.bcctx 2006-11-09 11:29:09.000000000 +0300
+++ ./arch/i386/kernel/irq.c 2006-11-09 11:32:24.000000000 +0300
@@ -21,6 +21,8 @@
 #include <asm/apic.h>
 #include <asm/uaccess.h>

+#include <bc/task.h>
+
 DEFINE_PER_CPU(irq_cpustat_t, irq_stat) ____cacheline_internodealigned_in_smp;
 EXPORT_PER_CPU_SYMBOL(irq_stat);

@@ -75,6 +77,7 @@ fastcall unsigned int do_IRQ(struct pt_r
  union irq_ctx *curctx, *irqctx;
  u32 *isp;
 #endif
+ struct beancounter *bc;

  if (unlikely((unsigned)irq >= NR_IRQS)) {
   printk(KERN_EMERG "%s: cannot handle IRQ %d\n",
@@ -99,6 +102,7 @@ fastcall unsigned int do_IRQ(struct pt_r
  }
```

```
 #endif

+ bc = set_exec_bc(&init_bc);
 #ifdef CONFIG_4KSTACKS

  curctx = (union irq_ctx *) current_thread_info();
@@ -139,6 +143,7 @@ fastcall unsigned int do_IRQ(struct pt_r
 #endif
   desc->handle_irq(irq, desc);

+ reset_exec_bc(bc, &init_bc);
  irq_exit();
  set_irq_regs(old_regs);
  return 1;
--- ./fs/proc/array.c.bcctx 2006-11-09 11:29:11.000000000 +0300
+++ ./fs/proc/array.c 2006-11-09 11:32:24.000000000 +0300
@@ -76,6 +76,8 @@
 #include <linux/rcupdate.h>
 #include <linux/delayacct.h>

+#include <bc/beancounter.h>
+
 #include <asm/uaccess.h>
 #include <asm/pgtable.h>
 #include <asm/io.h>
@@ -180,6 +182,10 @@ static inline char * task_state(struct t
   p->uid, p->euid, p->suid, p->fsuid,
   p->gid, p->egid, p->sgid, p->fsgid);

+#ifdef CONFIG_BEANCOUNTERS
+ buffer += sprintf(buffer, "Bcid:\t%d\n", p->exec_bc->bc_id);
+#endif
+
  task_lock(p);
  if (p->files)
   fdt = files_fdtable(p->files);
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./include/bc/task.h 2006-11-09 11:32:24.000000000 +0300
@@ -0,0 +1,53 @@
+/*
+ * include/bc/task.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;
+
+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, int whole);
+#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
--- ./include/linux/sched.h.bcctx 2006-11-09 11:29:12.000000000 +0300
+++ ./include/linux/sched.h 2006-11-09 11:32:24.000000000 +0300
@@ -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>
@@ -1061,6 +1063,9 @@ 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)
--- ./kernel/bc/beancounter.c.bcctx 2006-11-09 11:30:04.000000000 +0300
+++ ./kernel/bc/beancounter.c 2006-11-09 11:32:24.000000000 +0300
@@ -235,6 +235,8 @@ void __init bc_init_early(void)

  spin_lock_init(&bc_hash_lock);
  hlist_add_head(&init_bc.bc_hash, &bc_hash[hash_long(0, BC_HASH_BITS)]);
+
+ current->exec_bc = bc_get(&init_bc);
 }

 int __init bc_init_late(void)
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./kernel/bc/misc.c 2006-11-09 11:32:24.000000000 +0300
@@ -0,0 +1,100 @@
+/*
+ * kernel/bc/misc.c
+ *
+ * Copyright (C) 2006 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>
+
+int copy_beancounter(struct task_struct *tsk, struct task_struct *parent)
+{
```

```
+ struct beancounter *bc;
+
+ bc = parent->exec_bc;
+ tsk->exec_bc = bc_get(bc);
+ return 0;
+}
+
+void free_beancounter(struct task_struct *tsk)
+{
+ struct beancounter *bc;
+
+ bc = tsk->exec_bc;
+ bc_put(bc);
+}
+
+struct set_bcid_data {
+ struct beancounter *bc;
+ struct mm_struct *mm;
+ struct task_struct *tsk;
+ int whole;
+};
+
+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;
+
+ if (!d->whole) {
+  p = d->tsk;
+  bc_put(p->exec_bc);
+  p->exec_bc = bc_get(d->bc);
+  return 0;
+ }
+
+ 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(int pid, struct beancounter *bc, int whole)
+{
+ int err;
+ struct set_bcid_data data;
+ struct task_struct *tsk;
+ struct mm_struct *mm;
+
+ 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;
+
+ mm = get_task_mm(tsk);
+ if (mm == NULL)
+  return -EINVAL;
+
+ data.bc = bc;
+ data.mm = mm;
+ data.tsk = tsk;
+ data.whole = whole;
+
+ down_write(&mm->mmap_sem);
+ err = stop_machine_run(do_set_bcid, &data, NR_CPUS);
+ up_write(&mm->mmap_sem);
+
+ mmput(mm);
+ put_task_struct(tsk);
+ return err;
+}
+EXPORT_SYMBOL(bc_task_move);
--- ./kernel/fork.c.bcctx 2006-11-09 11:29:12.000000000 +0300
+++ ./kernel/fork.c 2006-11-09 11:32:24.000000000 +0300
@@ -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>
@@ -103,12 +105,18 @@ kmem_cache_t *vm_area_cachep;
 /* SLAB cache for mm_struct structures (tsk->mm) */
```

```
 static kmem_cache_t *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)
@@ -985,6 +993,10 @@ 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);
@@ -1298,7 +1310,9 @@ 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/irq/handle.c.bcctx 2006-11-09 11:29:12.000000000 +0300
+++ ./kernel/irq/handle.c 2006-11-09 11:32:24.000000000 +0300
@@ -16,6 +16,8 @@
 #include <linux/interrupt.h>
 #include <linux/kernel_stat.h>

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

```
 #include "internals.h"

 /**
@@ -169,6 +171,7 @@ 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)) {
@@ -225,6 +228,8 @@ 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;

@@ -239,6 +244,7 @@ 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.bcctx 2006-11-09 11:29:12.000000000 +0300
+++ ./kernel/softirq.c 2006-11-09 11:32:24.000000000 +0300
@@ -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();
```