Subject: [PATCH 13/13] BC: numfiles accounting
Posted by dev on Thu, 09 Nov 2006 17:03:02 GMT
View Forum Message <> Reply to Message

Very simple beancounter for accounting and limiting
container number of opened files.

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

```
 fs/file_table.c    |   12 ++++++++++-
 include/bc/misc.h  |   27 ++++++++++++++++++++++++++
 include/linux/fs.h |    3 +++
 kernel/bc/misc.c   |   41 ++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 82 insertions(+), 1 deletion(-)

--- ./fs/file_table.c.bcnumfiles 2006-11-09 11:29:11.000000000 +0300
+++ ./fs/file_table.c 2006-11-09 11:35:34.000000000 +0300
@@ -23,6 +23,8 @@
 #include <linux/sysctl.h>
 #include <linux/percpu_counter.h>

+#include <bc/misc.h>
+
 #include <asm/atomic.h>

 /* sysctl tunables... */
@@ -44,6 +46,7 @@ static inline void file_free_rcu(struct
 static inline void file_free(struct file *f)
 {
  percpu_counter_dec(&nr_files);
+ bc_file_uncharge(f);
  call_rcu(&f->f_u.fu_rcuhead, file_free_rcu);
 }

@@ -108,8 +111,11 @@ struct file *get_empty_filp(void)
  if (f == NULL)
   goto fail;

- percpu_counter_inc(&nr_files);
  memset(f, 0, sizeof(*f));
+ if (bc_file_charge(f))
+  goto fail_charge;
+
+ percpu_counter_inc(&nr_files);
  if (security_file_alloc(f))
```

```
   goto fail_sec;

@@ -136,6 +142,10 @@ fail_sec:
  file_free(f);
 fail:
  return NULL;
+
+fail_charge:
+ kmem_cache_free(filp_cachep, f);
+ return NULL;
 }

 EXPORT_SYMBOL(get_empty_filp);
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./include/bc/misc.h 2006-11-09 11:34:42.000000000 +0300
@@ -0,0 +1,27 @@
+/*
+ * include/bc/misc.h
+ *
+ * Copyright (C) 2006 OpenVZ SWsoft Inc
+ *
+ */
+
+#ifndef __BC_MISC_H__
+#define __BC_MISC_H__
+
+struct file;
+
+#ifdef CONFIG_BEANCOUNTERS
+int __must_check bc_file_charge(struct file *);
+void bc_file_uncharge(struct file *);
+#else
+static inline int __must_check bc_file_charge(struct file *f)
+{
+ return 0;
+}
+
+static inline void bc_file_uncharge(struct file *f)
+{
+}
+#endif
+
+#endif
--- ./include/linux/fs.h.bcnumfiles 2006-11-09 11:29:12.000000000 +0300
+++ ./include/linux/fs.h 2006-11-09 11:34:42.000000000 +0300
@@ -802,6 +802,9 @@ struct file {
  struct kevent_storage st;
 #endif
```

```
  struct address_space *f_mapping;
+#ifdef CONFIG_BEANCOUNTERS
+ struct beancounter *f_bc;
+#endif
 };
 extern spinlock_t files_lock;
 #define file_list_lock() spin_lock(&files_lock);
--- ./kernel/bc/misc.c.bcnumfiles 2006-11-09 11:34:31.000000000 +0300
+++ ./kernel/bc/misc.c 2006-11-09 11:34:42.000000000 +0300
@@ -7,6 +7,7 @@
 #include <linux/sched.h>
 #include <linux/stop_machine.h>
 #include <linux/module.h>
+#include <linux/fs.h>

 #include <bc/beancounter.h>
 #include <bc/task.h>
@@ -95,6 +96,45 @@ int bc_task_move(struct task_struct *tsk
 }
 EXPORT_SYMBOL(bc_task_move);

+int bc_file_charge(struct file *file)
+{
+ int sev;
+ struct beancounter *bc;
+
+ bc = get_exec_bc();
+ sev = (capable(CAP_SYS_ADMIN) ? BC_LIMIT : BC_BARRIER);
+
+ if (bc_charge(bc, BC_NUMFILES, 1, sev))
+  return -EMFILE;
+
+ file->f_bc = bc_get(bc);
+ return 0;
+}
+
+void bc_file_uncharge(struct file *file)
+{
+ struct beancounter *bc;
+
+ bc = file->f_bc;
+ bc_uncharge(bc, BC_NUMFILES, 1);
+ bc_put(bc);
+}
+
+#define BC_NUMFILES_BARRIER 256
+#define BC_NUMFILES_LIMIT 512
+
```

```
+static int bc_files_init(struct beancounter *bc, int i)
+{
+ bc_init_resource(&bc->bc_parms[BC_NUMFILES],
+   BC_NUMFILES_BARRIER, BC_NUMFILES_LIMIT);
+ return 0;
+}
+
+static struct bc_resource bc_files_resource = {
+ .bcr_name = "numfiles",
+ .bcr_init = bc_files_init,
+};
+
 #define BC_NUMTASKS_BARRIER 128
 #define BC_NUMTASKS_LIMIT 128

@@ -113,6 +153,7 @@ static struct bc_resource bc_task_resour
 static int __init bc_misc_init_resource(void)
 {
  bc_register_resource(BC_NUMTASKS, &bc_task_resource);
+ bc_register_resource(BC_NUMFILES, &bc_files_resource);
  return 0;
 }
```