Subject: [PATCH 8/13] BC: locked pages (core)
Posted by dev on Tue, 05 Sep 2006 15:24:18 GMT
View Forum Message <> Reply to Message

Introduce new resource BC_LOCKEDPAGES which stands for accounting
of mlock-ed user pages.

Locked pages are important to be accounted separately
as they are unreclaimable.

Pages are charged to mm_struct BC.

Signed-Off-By: Pavel Emelianov <xemul@sw.ru>
Signed-Off-By: Kirill Korotaev <dev@sw.ru>

---

 include/bc/beancounter.h | 3 -
 include/bc/vmpages.h | 95 ++
 include/linux/sched.h | 3 +
 include/linux/shmem_fs.h | 5 ++
 kernel/bc/Makefile | 1
 kernel/bc/beancounter.c | 2
 kernel/bc/vmpages.c | 75 ++++++++++++++++++++++++++++++++++++
 kernel/fork.c | 11 +++--
 mm/shmem.c | 4 +
 9 files changed, 195 insertions(+), 4 deletions(-)

--- ./include/bc/beancounter.h.bclockcore 2006-09-05 12:54:40.000000000 +0400
+++ ./include/bc/beancounter.h 2006-09-05 12:59:27.000000000 +0400
@@ -13,8 +13,9 @@
  */

 #define BC_KMEMSIZE 0
+#define BC_LOCKEDPAGES 1

-#define BC_RESOURCES 1
+#define BC_RESOURCES 2

 struct bc_resource_parm {
  unsigned long barrier; /* A barrier over which resource allocations
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./include/bc/vmpages.h 2006-09-05 13:04:03.000000000 +0400
@@ -0,0 +1,95 @@
+/*
+ * include/bc/vmpages.h
+ *
+ * Copyright (C) 2006 OpenVZ. SWsoft Inc

```
+ *
+ */
+
+#ifndef __BC_VMPAGES_H_
+#define __BC_VMPAGES_H_
+
+#include <bc/beancounter.h>
+#include <bc/task.h>
+
+struct mm_struct;
+struct file;
+struct shmem_inode_info;
+
+#ifdef CONFIG_BEANCOUNTERS
+int __must_check bc_memory_charge(struct mm_struct *mm, unsigned long size,
+  unsigned long vm_flags, struct file *vm_file, int strict);
+void bc_memory_uncharge(struct mm_struct *mm, unsigned long size,
+  unsigned long vm_flags, struct file *vm_file);
+
+int __must_check bc_locked_charge(struct mm_struct *mm, unsigned long size);
+void bc_locked_uncharge(struct mm_struct *mm, unsigned long size);
+
+int __must_check bc_locked_shm_charge(struct shmem_inode_info *info,
+  unsigned long size);
+void bc_locked_shm_uncharge(struct shmem_inode_info *info,
+  unsigned long size);
+
+/*
+ * mm's beancounter should be the same as the exec one
+ * of taks using this mm. thus we have two cases of its
+ * initialisation:
+ *  1. new mm is done for fork-ed task
+ *  2. new mm is done for exec-ing task
+ */
+#define mm_init_bc(mm, t) do {     \
+  (mm)->mm_bc = get_beancounter((t)->task_bc.exec_bc); \
+ } while (0)
+#define mm_free_bc(mm)  do {     \
+  put_beancounter((mm)->mm_bc);    \
+ } while (0)
+
+#define shmi_init_bc(info) do {     \
+  (info)->shm_bc = get_beancounter(get_exec_bc()); \
+ } while (0)
+#define shmi_free_bc(info) do {     \
+  put_beancounter((info)->shm_bc);   \
+ } while (0)
+
```

```
+#else /* CONFIG_BEANCOUNTERS */
+
+static inline int __must_check bc_memory_charge(struct mm_struct *mm,
+  unsigned long size, unsigned long vm_flags,
+  struct file *vm_file, int strict)
+{
+ return 0;
+}
+
+static inline void bc_memory_uncharge(struct mm_struct *mm, unsigned long size,
+  unsigned long vm_flags, struct file *vm_file)
+{
+}
+
+static inline int __must_check bc_locked_charge(struct mm_struct *mm,
+  unsigned long size)
+{
+ return 0;
+}
+
+static inline void bc_locked_uncharge(struct mm_struct *mm, unsigned long size)
+{
+}
+
+static inline int __must_check bc_locked_shm_charge(struct shmem_inode_info *i,
+  unsigned long size)
+{
+ return 0;
+}
+
+static inline void bc_locked_shm_uncharge(struct shmem_inode_info *i,
+  unsigned long size)
+{
+}
+
+#define mm_init_bc(mm, t) do { } while (0)
+#define mm_free_bc(mm)  do { } while (0)
+#define shmi_init_bc(info) do { } while (0)
+#define shmi_free_bc(info) do { } while (0)
+
+#endif /* CONFIG_BEANCOUNTERS */
+#endif
+
--- ./include/linux/sched.h.bclockcore 2006-09-05 12:54:21.000000000 +0400
+++ ./include/linux/sched.h 2006-09-05 12:59:27.000000000 +0400
@@ -358,6 +358,9 @@ struct mm_struct {
  /* aio bits */
  rwlock_t  ioctx_list_lock;
```

```
  struct kioctx  *ioctx_list;
+#ifdef CONFIG_BEANCOUNTERS
+ struct beancounter *mm_bc;
+#endif
 };

 struct sighand_struct {
--- ./include/linux/shmem_fs.h.bclockcore 2006-04-21 11:59:36.000000000 +0400
+++ ./include/linux/shmem_fs.h 2006-09-05 12:59:27.000000000 +0400
@@ -8,6 +8,8 @@

 #define SHMEM_NR_DIRECT 16

+struct beancounter;
+
 struct shmem_inode_info {
  spinlock_t  lock;
  unsigned long  flags;
@@ -19,6 +21,9 @@ struct shmem_inode_info {
  swp_entry_t  i_direct[SHMEM_NR_DIRECT]; /* first blocks */
  struct list_head swaplist; /* chain of maybes on swap */
  struct inode  vfs_inode;
+#ifdef CONFIG_BEANCOUNTERS
+ struct beancounter *shm_bc;
+#endif
 };

 struct shmem_sb_info {
--- ./kernel/bc/Makefile.bclockcore 2006-09-05 12:54:50.000000000 +0400
+++ ./kernel/bc/Makefile 2006-09-05 12:59:37.000000000 +0400
@@ -8,3 +8,4 @@ obj-y += beancounter.o
 obj-y += misc.o
 obj-y += sys.o
 obj-y += kmem.o
+obj-y += vmpages.o
--- ./kernel/bc/beancounter.c.bclockcore 2006-09-05 12:55:13.000000000 +0400
+++ ./kernel/bc/beancounter.c 2006-09-05 12:59:45.000000000 +0400
@@ -21,6 +21,7 @@ struct beancounter init_bc;

 const char *bc_rnames[] = {
  "kmemsize", /* 0 */
+ "lockedpages",
 };

 #define BC_HASH_BITS  8
@@ -232,6 +233,7 @@ static void init_beancounter_syslimits(s
  int k;
```

```
  bc->bc_parms[BC_KMEMSIZE].limit = 32 * 1024 * 1024;
+ bc->bc_parms[BC_LOCKEDPAGES].limit = 8;

  for (k = 0; k < BC_RESOURCES; k++)
   bc->bc_parms[k].barrier = bc->bc_parms[k].limit;
--- /dev/null 2006-07-18 14:52:43.075228448 +0400
+++ ./kernel/bc/vmpages.c 2006-09-05 12:59:27.000000000 +0400
@@ -0,0 +1,75 @@
+/*
+ *  kernel/bc/vmpages.c
+ *
+ *  Copyright (C) 2006 OpenVZ. SWsoft Inc
+ *
+ */
+
+#include <linux/sched.h>
+#include <linux/mm.h>
+#include <linux/shmem_fs.h>
+
+#include <bc/beancounter.h>
+#include <bc/vmpages.h>
+
+#include <asm/page.h>
+
+int bc_memory_charge(struct mm_struct *mm, unsigned long size,
+  unsigned long vm_flags, struct file *vm_file, int strict)
+{
+ struct beancounter *bc;
+
+ bc = mm->mm_bc;
+ size >>= PAGE_SHIFT;
+
+ if (vm_flags & VM_LOCKED)
+  if (bc_charge(bc, BC_LOCKEDPAGES, size, strict))
+   return -ENOMEM;
+ return 0;
+}
+
+void bc_memory_uncharge(struct mm_struct *mm, unsigned long size,
+  unsigned long vm_flags, struct file *vm_file)
+{
+ struct beancounter *bc;
+
+ bc = mm->mm_bc;
+ size >>= PAGE_SHIFT;
+
+ if (vm_flags & VM_LOCKED)
+  bc_uncharge(bc, BC_LOCKEDPAGES, size);
```

```
+}
+
+static inline int locked_charge(struct beancounter *bc,
+  unsigned long size)
+{
+ size >>= PAGE_SHIFT;
+ return bc_charge(bc, BC_LOCKEDPAGES, size, BC_BARRIER);
+}
+
+static inline void locked_uncharge(struct beancounter *bc,
+  unsigned long size)
+{
+ size >>= PAGE_SHIFT;
+ bc_uncharge(bc, BC_LOCKEDPAGES, size);
+}
+
+int bc_locked_charge(struct mm_struct *mm, unsigned long size)
+{
+ return locked_charge(mm->mm_bc, size);
+}
+
+void bc_locked_uncharge(struct mm_struct *mm, unsigned long size)
+{
+ locked_uncharge(mm->mm_bc, size);
+}
+
+int bc_locked_shm_charge(struct shmem_inode_info *info, unsigned long size)
+{
+ return locked_charge(info->shm_bc, size);
+}
+
+void bc_locked_shm_uncharge(struct shmem_inode_info *info, unsigned long size)
+{
+ locked_uncharge(info->shm_bc, size);
+}
--- ./kernel/fork.c.bclockcore 2006-09-05 12:58:17.000000000 +0400
+++ ./kernel/fork.c 2006-09-05 12:59:59.000000000 +0400
@@ -49,6 +49,7 @@
 #include <linux/taskstats_kern.h>

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

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -322,7 +323,8 @@ 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);
@@ -339,6 +341,7 @@ static struct mm_struct * mm_init(struct
  mm->cached_hole_size = ~0UL;

  if (likely(!mm_alloc_pgd(mm))) {
+  mm_init_bc(mm, tsk);
   mm->def_flags = 0;
   return mm;
  }
@@ -356,7 +359,7 @@ 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;
 }
@@ -371,6 +374,7 @@ void fastcall __mmdrop(struct mm_struct
  BUG_ON(mm == &init_mm);
  mm_free_pgd(mm);
  destroy_context(mm);
+ mm_free_bc(mm);
  free_mm(mm);
 }

@@ -477,7 +481,7 @@ static struct mm_struct *dup_mm(struct t

  memcpy(mm, oldmm, sizeof(*mm));

- if (!mm_init(mm))
+ if (!mm_init(mm, tsk))
   goto fail_nomem;

  if (init_new_context(tsk, mm))
@@ -504,6 +508,7 @@ fail_nocontext:
   * because it calls destroy_context()
   */
  mm_free_pgd(mm);
+ mm_free_bc(mm);
  free_mm(mm);
  return NULL;
```

```
 }
--- ./mm/shmem.c.bclockcore 2006-09-05 12:58:17.000000000 +0400
+++ ./mm/shmem.c 2006-09-05 12:59:27.000000000 +0400
@@ -47,6 +47,8 @@
 #include <linux/migrate.h>
 #include <linux/highmem.h>

+#include <bc/vmpages.h>
+
 #include <asm/uaccess.h>
 #include <asm/div64.h>
 #include <asm/pgtable.h>
@@ -698,6 +700,7 @@ static void shmem_delete_inode(struct in
   sbinfo->free_inodes++;
   spin_unlock(&sbinfo->stat_lock);
  }
+ shmi_free_bc(info);
  clear_inode(inode);
 }

@@ -1359,6 +1362,7 @@ shmem_get_inode(struct super_block *sb,
   info = SHMEM_I(inode);
   memset(info, 0, (char *)inode - (char *)info);
   spin_lock_init(&info->lock);
+  shmi_init_bc(info);
   INIT_LIST_HEAD(&info->swaplist);

   switch (mode & S_IFMT) {
```