
Subject: [PATCH 11/13] BC: vmrss (preparations)
Posted by [dev](#) on Tue, 05 Sep 2006 15:28:05 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch does simple things:

- introduces an bc_magic field on beancounter to make sure union on struct page is correctly used in next patches
- adds nr_beancounters
- adds unused_privvmpages variable (counter of privvm pages which are not mapped into VM address space and thus potentially can be allocated later)

This is needed by vmrss accounting and is done to make patch reviewing simpler.

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

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

```
include/bc/beancounter.h | 13 ++++++++
include/bc/vmpages.h     |  2 ++
kernel/bc/beancounter.c  |  5 +++++
kernel/bc/kmem.c         |  1 +
kernel/bc/vmpages.c      | 44 ++++++++++++++++++++++++++++++++++++++-----
5 files changed, 61 insertions(+), 4 deletions(-)
```

```
--- ./include/bc/beancounter.h.bcvmrssprep 2006-09-05 13:17:50.000000000 +0400
```

```
+++ ./include/bc/beancounter.h 2006-09-05 13:44:33.000000000 +0400
```

```
@@ -45,6 +45,13 @@ struct bc_resource_parm {
#define BC_MAXVALUE LONG_MAX
```

```
/*
```

```
+ * This magic is used to distinguish user beancounter and pages beancounter
+ * in struct page. page_ub and page_bc are placed in union and MAGIC
+ * ensures us that we don't use pbc as ubc in bc_page_uncharge().
```

```
+ */
```

```
+#define BC_MAGIC          0x62756275UL
```

```
+
```

```
+/*
```

```
 * Resource management structures
 * Serialization issues:
 * beancounter list management is protected via bc_hash_lock
```

```
@@ -54,11 +61,13 @@ struct bc_resource_parm {
 */
```

```
struct beancounter {
+ unsigned long bc_magic;
```

```

atomic_t bc_refcount;
spinlock_t bc_lock;
bcid_t bc_id;
struct hlist_node hash;

+ unsigned long unused_privvmpages;
/* resources statistics and settings */
struct bc_resource_parm bc_parms[BC_RESOURCES];
};
@@ -74,6 +83,8 @@ enum bc_severity { BC_BARRIER, BC_LIMIT,

#ifdef CONFIG_BEANCOUNTERS

+extern unsigned int nr_beancounters = 1;
+
/*
 * These functions tune minheld and maxheld values for a given
 * resource when held value changes
@@ -137,6 +137,8 @@ extern const char *bc_rnames[];

#else /* CONFIG_BEANCOUNTERS */

+#define nr_beancounters 0
+
#define beancounter_findcreate(id, f) (NULL)
#define get_beancounter(bc) (NULL)
#define put_beancounter(bc) do { } while (0)
--- ./include/bc/vmpages.h.bcvmrssprep 2006-09-05 13:38:07.000000000 +0400
+++ ./include/bc/vmpages.h 2006-09-05 13:40:21.000000000 +0400
@@ -77,6 +77,8 @@ void bc_locked_shm_uncharge(struct shmем
    put_beancounter((info)->shm_bc); \
} while (0)

+void bc_update_privvmpages(struct beancounter *bc);
+
#else /* CONFIG_BEANCOUNTERS */

static inline int __must_check bc_memory_charge(struct mm_struct *mm,
--- ./kernel/bc/beancounter.c.bcvmrssprep 2006-09-05 13:17:50.000000000 +0400
+++ ./kernel/bc/beancounter.c 2006-09-05 13:44:53.000000000 +0400
@@ -19,6 +19,8 @@ static void init_beancounter_struct(stru

struct beancounter init_bc;

+unsigned int nr_beancounters;
+
const char *bc_rnames[] = {
    "kmemsize", /* 0 */

```

```

"lockedpages",
@@ -88,6 +90,7 @@ retry:

out_install:
    hlist_add_head(&new_bc->hash, slot);
+ nr_beancounters++;
    spin_unlock_irqrestore(&bc_hash_lock, flags);
out:
    return new_bc;
@@ -110,6 +113,7 @@ void put_beancounter(struct beancounter
    bc->bc_parms[i].held, bc_rnames[i]);

    hlist_del(&bc->hash);
+ nr_beancounters--;
    spin_unlock_irqrestore(&bc_hash_lock, flags);

    kmem_cache_free(bc_cachep, bc);
@@ -214,6 +218,7 @@ EXPORT_SYMBOL_GPL(bc_uncharge);

static void init_beancounter_struct(struct beancounter *bc, bcid_t id)
{
+ bc->bc_magic = BC_MAGIC;
    atomic_set(&bc->bc_refcount, 1);
    spin_lock_init(&bc->bc_lock);
    bc->bc_id = id;
--- ./kernel/bc/kmem.c.bcvmrssprep 2006-09-05 12:54:40.000000000 +0400
+++ ./kernel/bc/kmem.c 2006-09-05 13:40:21.000000000 +0400
@@ -79,6 +79,7 @@ void bc_page_uncharge(struct page *page,
    if (bc == NULL)
        return;

+ BUG_ON(bc->bc_magic != BC_MAGIC);
    bc_uncharge(bc, BC_KMEMSIZE, PAGE_SIZE << order);
    put_beancounter(bc);
    page_bc(page) = NULL;
--- ./kernel/bc/vmpages.c.bcvmrssprep 2006-09-05 13:28:16.000000000 +0400
+++ ./kernel/bc/vmpages.c 2006-09-05 13:45:34.000000000 +0400
@@ -14,6 +14,34 @@

#include <asm/page.h>

+void bc_update_privvmpages(struct beancounter *bc)
+{
+ bc->bc_parms[BC_PRIVVMPAGES].held = bc->unused_privvmpages;
+ bc_adjust_minheld(bc, BC_PRIVVMPAGES);
+ bc_adjust_maxheld(bc, BC_PRIVVMPAGES);
+}
+

```

```

+static inline int privvm_charge(struct beancounter *bc, unsigned long sz,
+ int strict)
+{
+ if (bc_charge_locked(bc, BC_PRIVVMPAGES, sz, strict))
+ return -ENOMEM;
+
+ bc->unused_privvmpages += sz;
+ return 0;
+}
+
+static inline void privvm_uncharge(struct beancounter *bc, unsigned long sz)
+{
+ if (unlikely(bc->unused_privvmpages < sz)) {
+ printk("BC: overuncharging %d unused pages: val %lu held %lu\n",
+ bc->bc_id, sz, bc->unused_privvmpages);
+ sz = bc->unused_privvmpages;
+ }
+ bc->unused_privvmpages -= sz;
+ bc_update_privvmpages(bc);
+}
+
+int bc_memory_charge(struct mm_struct *mm, unsigned long size,
+ unsigned long vm_flags, struct file *vm_file, int strict)
+{
@@ -28,7 +56,7 @@ int bc_memory_charge(struct mm_struct *m
+ if (bc_charge_locked(bc, BC_LOCKEDPAGES, size, strict))
+ goto err_locked;
+ if (BC_VM_PRIVATE(vm_flags, vm_file))
- if (bc_charge_locked(bc, BC_PRIVVMPAGES, size, strict))
+ if (privvm_charge(bc, size, strict))
+ goto err_privvm;
+ spin_unlock_irqrestore(&bc->bc_lock, flags);
+ return 0;
@@ -53,7 +81,7 @@ void bc_memory_uncharge(struct mm_struct
+ if (vm_flags & VM_LOCKED)
+ bc_uncharge_locked(bc, BC_LOCKEDPAGES, size);
+ if (BC_VM_PRIVATE(vm_flags, vm_file))
- bc_uncharge_locked(bc, BC_PRIVVMPAGES, size);
+ privvm_uncharge(bc, size);
+ spin_unlock_irqrestore(&bc->bc_lock, flags);
+ }

@@ -73,18 +101,26 @@ int bc_privvm_recharge(unsigned long vm_

int bc_privvm_charge(struct mm_struct *mm, unsigned long size)
{
+ int ret;
+ struct beancounter *bc;

```

+ unsigned long flags;

```
    bc = mm->mm_bc;
- bc_charge(bc, BC_PRIVVMPAGES, size >> PAGE_SHIFT);
+ spin_lock_irqsave(&bc->bc_lock, flags);
+ ret = privvm_charge(bc, size >> PAGE_SHIFT, BC_BARRIER);
+ spin_unlock_irqrestore(&bc->bc_lock, flags);
+ return ret;
}
```

```
void bc_privvm_uncharge(struct mm_struct *mm, unsigned long size)
{
    struct beancounter *bc;
+ unsigned long flags;
```

```
    bc = mm->mm_bc;
- bc_uncharge(bc, BC_PRIVVMPAGES, size >> PAGE_SHIFT);
+ spin_lock_irqsave(&bc->bc_lock, flags);
+ privvm_uncharge(bc, size >> PAGE_SHIFT);
+ spin_unlock_irqrestore(&bc->bc_lock, flags);
}
```

```
static inline int locked_charge(struct beancounter *bc,
```
