
Subject: [-mm PATCH 8/10] Memory controller add switch to control what type of pages to limit (v7)

Posted by [Balbir Singh](#) on Fri, 24 Aug 2007 15:21:15 GMT

[View Forum Message](#) <> [Reply to Message](#)

Choose if we want cached pages to be accounted or not. By default both are accounted for. A new set of tunables are added.

```
echo -n 1 > mem_control_type
```

switches the accounting to account for only mapped pages

```
echo -n 3 > mem_control_type
```

switches the behaviour back

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

```
include/linux/memcontrol.h | 9 ++++
mm/filemap.c                | 2
mm/memcontrol.c             | 92 +++++++++++++++++++++++++++++++++++++
mm/swap_state.c            | 2
4 files changed, 103 insertions(+), 2 deletions(-)
```

```
diff -puN include/linux/memcontrol.h~mem-control-choose-rss-vs-rss-and-pagecache
include/linux/memcontrol.h
```

```
linux-2.6.23-rc2-mm2/include/linux/memcontrol.h~mem-control-choose-rss-vs-rss-and-pagecache
2007-08-24 20:46:08.000000000 +0530
```

```
+++ linux-2.6.23-rc2-mm2-balbir/include/linux/memcontrol.h 2007-08-24 20:46:08.000000000
+0530
```

```
@@ -20,6 +20,8 @@
```

```
#ifndef _LINUX_MEMCONTROL_H
#define _LINUX_MEMCONTROL_H
```

```
+#include <linux/mm.h>
```

```
+
```

```
struct mem_container;
struct page_container;
```

```
@@ -40,6 +42,7 @@ extern unsigned long mem_container_isola
```

```
struct mem_container *mem_cont,
int active);
```

```
extern void mem_container_out_of_memory(struct mem_container *mem);
```

```
+extern int mem_container_cache_charge(struct page *page, struct mm_struct *mm);
```

```
static inline void mem_container_uncharge_page(struct page *page)
```

```

{
@@ -84,6 +87,12 @@ static inline void mem_container_move_li
{
}

+static inline int mem_container_cache_charge(struct page *page,
+ struct mm_struct *mm)
+{
+ return 0;
+}
+
#endif /* CONFIG_CONTAINER_MEM_CONT */

#endif /* _LINUX_MEMCONTROL_H */
diff -puN mm/filemap.c~mem-control-choose-rss-vs-rss-and-pagecache mm/filemap.c
---
linux-2.6.23-rc2-mm2/mm/filemap.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-08-24
20:46:08.000000000 +0530
+++ linux-2.6.23-rc2-mm2-balbir/mm/filemap.c 2007-08-24 20:46:08.000000000 +0530
@@ -445,7 +445,7 @@ int add_to_page_cache(struct page *page,

    if (error == 0) {

- error = mem_container_charge(page, current->mm);
+ error = mem_container_cache_charge(page, current->mm);
    if (error)
        goto out;

diff -puN mm/memcontrol.c~mem-control-choose-rss-vs-rss-and-pagecache mm/memcontrol.c
---
linux-2.6.23-rc2-mm2/mm/memcontrol.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-08-
24 20:46:08.000000000 +0530
+++ linux-2.6.23-rc2-mm2-balbir/mm/memcontrol.c 2007-08-24 20:46:08.000000000 +0530
@@ -28,6 +28,8 @@
#include <linux/spinlock.h>
#include <linux/fs.h>

+#include <asm/uaccess.h>
+
struct container_subsys mem_container_subsys;
static const int MEM_CONTAINER_RECLAIM_RETRIES = 5;

@@ -59,6 +61,7 @@ struct mem_container {
    * spin_lock to protect the per container LRU
    */
    spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
};

```

```

/*
@@ -81,6 +84,15 @@ struct page_container {
    /* mapped and cached states */
};

+enum {
+ MEM_CONTAINER_TYPE_UNSPEC = 0,
+ MEM_CONTAINER_TYPE_MAPPED,
+ MEM_CONTAINER_TYPE_CACHED,
+ MEM_CONTAINER_TYPE_ALL,
+ MEM_CONTAINER_TYPE_MAX,
+} mem_control_type;
+
+static struct mem_container init_mem_container;

static inline
struct mem_container *mem_container_from_cont(struct container *cont)
@@ -361,6 +373,22 @@ err:
}

/*
+ * See if the cached pages should be charged at all?
+ */
+int mem_container_cache_charge(struct page *page, struct mm_struct *mm)
+{
+ struct mem_container *mem;
+ if (!mm)
+ mm = &init_mm;
+
+ mem = rcu_dereference(mm->mem_container);
+ if (mem->control_type == MEM_CONTAINER_TYPE_ALL)
+ return mem_container_charge(page, mm);
+ else
+ return 0;
+}
+
+/*
+ * Uncharging is always a welcome operation, we never complain, simply
+ * uncharge.
+ */
@@ -370,6 +398,10 @@ void mem_container_uncharge(struct page_
    struct page *page;
    unsigned long flags;

+ /*
+ * This can handle cases when a page is not charged at all and we
+ * are switching between handling the control_type.

```

```

+ */
+ if (!pc)
+   return;

@@ -405,6 +437,60 @@ static ssize_t mem_container_write(struct
+   cft->private, userbuf, nbytes, ppos);
+ }

+static ssize_t mem_control_type_write(struct container *cont,
+   struct cftype *cft, struct file *file,
+   const char __user *userbuf,
+   size_t nbytes, loff_t *pos)
+{
+   int ret;
+   char *buf, *end;
+   unsigned long tmp;
+   struct mem_container *mem;
+
+   mem = mem_container_from_cont(cont);
+   buf = kmalloc(nbytes + 1, GFP_KERNEL);
+   ret = -ENOMEM;
+   if (buf == NULL)
+     goto out;
+
+   buf[nbytes] = 0;
+   ret = -EFAULT;
+   if (copy_from_user(buf, userbuf, nbytes))
+     goto out_free;
+
+   ret = -EINVAL;
+   tmp = simple_strtoul(buf, &end, 10);
+   if (*end != '\0')
+     goto out_free;
+
+   if (tmp <= MEM_CONTAINER_TYPE_UNSPEC || tmp >= MEM_CONTAINER_TYPE_MAX)
+     goto out_free;
+
+   mem->control_type = tmp;
+   ret = nbytes;
+out_free:
+   kfree(buf);
+out:
+   return ret;
+}
+
+static ssize_t mem_control_type_read(struct container *cont,
+   struct cftype *cft,
+   struct file *file, char __user *userbuf,

```

```

+ size_t nbytes, loff_t *ppos)
+{
+ unsigned long val;
+ char buf[64], *s;
+ struct mem_container *mem;
+
+ mem = mem_container_from_cont(cont);
+ s = buf;
+ val = mem->control_type;
+ s += sprintf(s, "%lu\n", val);
+ return simple_read_from_buffer((void __user *)userbuf, nbytes,
+ ppos, buf, s - buf);
+}
+
static struct cftype mem_container_files[] = {
{
.name = "usage",
@@ -422,6 +508,11 @@ static struct cftype mem_container_files
.private = RES_FAILCNT,
.read = mem_container_read,
},
+ {
+ .name = "control_type",
+ .write = mem_control_type_write,
+ .read = mem_control_type_read,
+ },
};

static struct mem_container init_mem_container;
@@ -444,6 +535,7 @@ mem_container_create(struct container_su
INIT_LIST_HEAD(&mem->active_list);
INIT_LIST_HEAD(&mem->inactive_list);
spin_lock_init(&mem->lru_lock);
+ mem->control_type = MEM_CONTAINER_TYPE_ALL;
return &mem->css;
}

```

diff -puN mm/swap_state.c~mem-control-choose-rss-vs-rss-and-pagecache mm/swap_state.c

linux-2.6.23-rc2-mm2/mm/swap_state.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-08-24 20:46:08.000000000 +0530

+++ linux-2.6.23-rc2-mm2-balbir/mm/swap_state.c 2007-08-24 20:46:08.000000000 +0530

```

@@ -82,7 +82,7 @@ static int __add_to_swap_cache(struct pa
error = radix_tree_preload(gfp_mask);
if (!error) {

```

```

- error = mem_container_charge(page, current->mm);
+ error = mem_container_cache_charge(page, current->mm);

```

```
if (error)
goto out;
```

—

--

Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
