Subject: [RFC][-mm PATCH 8/8] Add switch to control what type of pages to limit (v3)
Posted by Balbir Singh on Fri, 20 Jul 2007 08:25:28 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 2 > mem_control_type

switches the behaviour back

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

 include/linux/memcontrol.h | 9 +++
 mm/filemap.c | 2
 mm/memcontrol.c | 129 ++++++++++++++++++++++++++++++++++++-------
 mm/swap_state.c | 2
 4 files changed, 122 insertions(+), 20 deletions(-)

diff -puN mm/memcontrol.c~mem-control-choose-rss-vs-rss-and-pagecache mm/memcontrol.c
--- linux-2.6.22-rc6/mm/memcontrol.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-07-20 13:13:26.000000000 +0530
+++ linux-2.6.22-rc6-balbir/mm/memcontrol.c 2007-07-20 13:13:26.000000000 +0530
@@ -27,6 +27,8 @@
 #include <linux/swap.h>
 #include <linux/spinlock.h>

+#include <asm/uaccess.h>
+
 struct container_subsys mem_container_subsys;

 /*
@@ -57,6 +59,7 @@ struct mem_container {
   * spin_lock to protect the per container LRU
   */
  spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
 };

 /*
@@ -70,6 +73,14 @@ struct meta_page {

```
  atomic_t ref_cnt;
 };

+enum {
+ MEM_CONTAINER_TYPE_UNSPEC = 0,
+ MEM_CONTAINER_TYPE_MAPPED,
+ 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)
@@ -307,6 +318,22 @@ 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.
  */
@@ -317,7 +344,9 @@ void mem_container_uncharge(struct meta_
  unsigned long flags;

  /*
-  * This can happen for PAGE_ZERO
+  * This can happen for PAGE_ZERO. This can also handle cases when
+  * a page is not charged at all and we are switching between
+  * handling the control_type.
   */
  if (!mp)
   return;
```

```
@@ -356,26 +385,59 @@ static ssize_t mem_container_write(struc
     cft->private, userbuf, nbytes, ppos);
 }

-static struct cftype mem_container_usage = {
- .name = "mem_usage",
- .private = RES_USAGE,
- .read = mem_container_read,
-};
+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;

-static struct cftype mem_container_limit = {
- .name = "mem_limit",
- .private = RES_LIMIT,
- .write = mem_container_write,
- .read = mem_container_read,
-};
+ mem = mem_container_from_cont(cont);
+ buf = kmalloc(nbytes + 1, GFP_KERNEL);
+ ret = -ENOMEM;
+ if (buf == NULL)
+  goto out;

-static struct cftype mem_container_failcnt = {
- .name = "mem_failcnt",
- .private = RES_FAILCNT,
- .read = mem_container_read,
-};
+ 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 struct mem_container init_mem_container;
+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 int mem_container_create(struct container_subsys *ss,
     struct container *cont)
@@ -398,9 +460,36 @@ static int mem_container_create(struct c
  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 0;
 }

+static struct cftype mem_container_usage = {
+ .name = "mem_usage",
+ .private = RES_USAGE,
+ .read = mem_container_read,
+};
+
+static struct cftype mem_container_limit = {
+ .name = "mem_limit",
+ .private = RES_LIMIT,
+ .write = mem_container_write,
+ .read = mem_container_read,
```

```
+};
+
+static struct cftype mem_container_control_type = {
+ .name = "mem_control_type",
+ .write = mem_control_type_write,
+ .read = mem_control_type_read,
+};
+
+static struct cftype mem_container_failcnt = {
+ .name = "mem_failcnt",
+ .private = RES_FAILCNT,
+ .read = mem_container_read,
+};
+
+
 static void mem_container_destroy(struct container_subsys *ss,
     struct container *cont)
 {
@@ -424,6 +513,10 @@ static int mem_container_populate(struct
  if (rc < 0)
   goto err;

+ rc = container_add_file(cont, &mem_container_control_type);
+ if (rc < 0)
+  goto err;
+
 err:
  return rc;
 }
diff -puN include/linux/memcontrol.h~mem-control-choose-rss-vs-rss-and-pagecache
include/linux/memcontrol.h
---
linux-2.6.22-rc6/include/linux/memcontrol.h~mem-control-choose-rss-vs-rss-and-pagecache 2007-
07-20 13:13:26.000000000 +0530
+++ linux-2.6.22-rc6-balbir/include/linux/memcontrol.h 2007-07-20 13:13:26.000000000 +0530
@@ -20,6 +20,8 @@
 #ifndef _LINUX_MEMCONTROL_H
 #define _LINUX_MEMCONTROL_H

+#include <linux/mm.h>
+
 struct mem_container;
 struct meta_page;

@@ -39,6 +41,7 @@ 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);

 #else /* CONFIG_CONTAINER_MEM_CONT */
 static inline void mm_init_container(struct mm_struct *mm,
@@ -73,6 +76,12 @@ 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/swap_state.c~mem-control-choose-rss-vs-rss-and-pagecache mm/swap_state.c
--- linux-2.6.22-rc6/mm/swap_state.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-07-20
13:13:26.000000000 +0530
+++ linux-2.6.22-rc6-balbir/mm/swap_state.c 2007-07-20 13:13:26.000000000 +0530
@@ -81,7 +81,7 @@ 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;

diff -puN mm/filemap.c~mem-control-choose-rss-vs-rss-and-pagecache mm/filemap.c
--- linux-2.6.22-rc6/mm/filemap.c~mem-control-choose-rss-vs-rss-and-pagecache 2007-07-20
13:13:26.000000000 +0530
+++ linux-2.6.22-rc6-balbir/mm/filemap.c 2007-07-20 13:13:26.000000000 +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;


_

--
 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