Subject: [PATCH 30/33] memory controller add switch to control what type of pages to limit v7
Posted by Paul Menage on Mon, 17 Sep 2007 21:03:37 GMT
View Forum Message <> Reply to Message

From: Balbir Singh <balbir@linux.vnet.ibm.com>
(container->cgroup renaming by Paul Menage <menage@google.com>)

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>
Signed-off-by: Paul Menage <menage@google.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~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v
7 include/linux/memcontrol.h
---
a/include/linux/memcontrol.h~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-
v7
+++ a/include/linux/memcontrol.h
@@ -20,6 +20,8 @@
#ifndef _LINUX_MEMCONTROL_H
#define _LINUX_MEMCONTROL_H

+#include <linux/mm.h>
+
struct mem_cgroup;
struct page_cgroup;

@@ -40,6 +42,7 @@ extern unsigned long mem_cgroup_isola
struct mem_cgroup *mem_cont,
int active);

```
 extern void mem_cgroup_out_of_memory(struct mem_cgroup *mem);
+extern int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm);

 static inline void mem_cgroup_uncharge_page(struct page *page)
 {
@@ -84,6 +87,12 @@ static inline void mem_cgroup_move_li
 {
 }

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

 #endif /* _LINUX_MEMCONTROL_H */
diff -puN mm/filemap.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
mm/filemap.c
--- a/mm/filemap.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
+++ a/mm/filemap.c
@@ -444,7 +444,7 @@ int add_to_page_cache(struct page *page,

  if (error == 0) {

-  error = mem_cgroup_charge(page, current->mm);
+  error = mem_cgroup_cache_charge(page, current->mm);
    if (error)
     goto out;

diff -puN
mm/memcontrol.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
mm/memcontrol.c
--- a/mm/memcontrol.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
+++ a/mm/memcontrol.c
@@ -28,6 +28,8 @@
 #include <linux/spinlock.h>
 #include <linux/fs.h>

+#include <asm/uaccess.h>
+
 struct cgroup_subsys mem_cgroup_subsys;
 static const int MEM_CGROUP_RECLAIM_RETRIES = 5;

@@ -59,6 +61,7 @@ struct mem_cgroup {
    * spin_lock to protect the per cgroup LRU
    */
```

```
  spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
 };

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

+enum {
+ MEM_CGROUP_TYPE_UNSPEC = 0,
+ MEM_CGROUP_TYPE_MAPPED,
+ MEM_CGROUP_TYPE_CACHED,
+ MEM_CGROUP_TYPE_ALL,
+ MEM_CGROUP_TYPE_MAX,
+} mem_control_type;
+
+static struct mem_cgroup init_mem_cgroup;

 static inline
 struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont)
@@ -361,6 +373,22 @@ err:
 }

 /*
+ * See if the cached pages should be charged at all?
+ */
+int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm)
+{
+ struct mem_cgroup *mem;
+ if (!mm)
+   mm = &init_mm;
+
+ mem = rcu_dereference(mm->mem_cgroup);
+ if (mem->control_type == MEM_CGROUP_TYPE_ALL)
+   return mem_cgroup_charge(page, mm);
+ else
+   return 0;
+}
+
+/*
  * Uncharging is always a welcome operation, we never complain, simply
  * uncharge.
  */
@@ -370,6 +398,10 @@ void mem_cgroup_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_cgroup_write(struc
     cft->private, userbuf, nbytes, ppos);
 }

+static ssize_t mem_control_type_write(struct cgroup *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_cgroup *mem;
+
+ mem = mem_cgroup_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_CGROUP_TYPE_UNSPEC || tmp >= MEM_CGROUP_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 cgroup *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_cgroup *mem;
+
+ mem = mem_cgroup_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_cgroup_files[] = {
  {
   .name = "usage",
@@ -422,6 +508,11 @@ static struct cftype mem_cgroup_files
   .private = RES_FAILCNT,
   .read = mem_cgroup_read,
  },
+ {
+  .name = "control_type",
+  .write = mem_control_type_write,
+  .read = mem_control_type_read,
+ },
 };

 static struct mem_cgroup init_mem_cgroup;
@@ -444,6 +535,7 @@ mem_cgroup_create(struct cgroup_su
  INIT_LIST_HEAD(&mem->active_list);
  INIT_LIST_HEAD(&mem->inactive_list);
  spin_lock_init(&mem->lru_lock);
+ mem->control_type = MEM_CGROUP_TYPE_ALL;
  return &mem->css;
 }

diff -puN
mm/swap_state.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
mm/swap_state.c
--- a/mm/swap_state.c~memory-controller-add-switch-to-control-what-type-of-pages-to-limit-v7
+++ a/mm/swap_state.c
@@ -81,7 +81,7 @@ static int __add_to_swap_cache(struct pa
  error = radix_tree_preload(gfp_mask);
  if (!error) {
```

```
-  error = mem_cgroup_charge(page, current->mm);
+  error = mem_cgroup_cache_charge(page, current->mm);
   if (error)
    goto out;
```

–

--

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