Subject: [RFC PATCH 3/5] mgmcg VM overcommit interface
Posted by Andrea Righi on Mon, 09 Jun 2008 23:33:01 GMT
View Forum Message <> Reply to Message

Per-cgroup overcommit_memory and overcommit_ratio parameters can be shown
and/or manipulated using the files "memory.overcommit_memory" and
"memory.overcommit_ratio" in the cgroup filesystem.

The file "overcommit_as" can be used to retrieve the current committed space
and limit of each cgroup.

Signed-off-by: Andrea Righi <righi.andrea@gmail.com>
---
 mm/memcontrol.c | 116 ++-
 1 files changed, 115 insertions(+), 1 deletions(-)

```
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 4100e24..e3e34e9 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -36,6 +36,8 @@

 #include <asm/uaccess.h>

+#define K(x) ((x) << (PAGE_SHIFT - 10))
+
 struct cgroup_subsys mem_cgroup_subsys;
 static const int MEM_CGROUP_RECLAIM_RETRIES = 5;
 static struct kmem_cache *page_cgroup_cache;
@@ -47,7 +49,7 @@ enum mem_cgroup_stat_index {
   /*
    * For MEM_CONTAINER_TYPE_ALL, usage = pagecache + rss.
    */
- MEM_CGROUP_STAT_CACHE,     /* # of pages charged as cache */
+ MEM_CGROUP_STAT_CACHE,    /* # of pages charged as cache */
   MEM_CGROUP_STAT_RSS,    /* # of pages charged as rss */
   MEM_CGROUP_STAT_PGPGIN_COUNT, /* # of pages paged in */
   MEM_CGROUP_STAT_PGPGOUT_COUNT, /* # of pages paged out */
@@ -55,6 +57,11 @@ enum mem_cgroup_stat_index {
   MEM_CGROUP_STAT_NSTATS,
 };

+enum mem_cgroup_vmacct_index {
+ MEM_CGROUP_OVERCOMMIT_MEMORY,
+ MEM_CGROUP_OVERCOMMIT_RATIO,
+};
+
 struct mem_cgroup_stat_cpu {
```

```
  s64 count[MEM_CGROUP_STAT_NSTATS];
 } ____cacheline_aligned_in_smp;
@@ -995,6 +1002,97 @@ static ssize_t mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
     mem_cgroup_write_strategy);
 }

+static ssize_t mem_cgroup_committed_read(struct cgroup *cont,
+       struct cftype *cft,
+       struct file *file,
+       char __user *userbuf,
+       size_t nbytes,
+       loff_t *ppos)
+{
+ struct mem_cgroup *mem;
+ char *page;
+ unsigned long total, committed, allowed;
+ ssize_t count;
+ int ret;
+
+ page = (char *)__get_free_page(GFP_TEMPORARY);
+ if (unlikely(!page))
+  return -ENOMEM;
+
+ cgroup_lock();
+ if (cgroup_is_removed(cont)) {
+  cgroup_unlock();
+  ret = -ENODEV;
+  goto out;
+ }
+
+ mem = mem_cgroup_from_cont(cont);
+ committed = atomic_long_read(&mem->vmacct.vm_committed_space);
+
+ total = (long)(mem->res.limit >> PAGE_SHIFT) + 1L;
+ if (total > (totalram_pages - hugetlb_total_pages()))
+  allowed = ((totalram_pages - hugetlb_total_pages())
+    * mem->vmacct.overcommit_ratio / 100)
+    + total_swap_pages;
+ else
+  allowed = total * mem->vmacct.overcommit_ratio / 100
+    + total_swap_pages;
+ cgroup_unlock();
+
+ count = sprintf(page, "CommitLimit:  %8lu kB\n"
+         "Committed_AS: %8lu kB\n",
+         K(allowed), K(committed));
+ ret = simple_read_from_buffer(userbuf, nbytes, ppos, page, count);
+out:
```

```
+ free_page((unsigned long)page);
+ return ret;
+}
+
+static s64 mem_cgroup_vmacct_read_s64(struct cgroup *cont, struct cftype *cft)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ enum mem_cgroup_vmacct_index type = cft->private;
+
+ switch (type) {
+ case MEM_CGROUP_OVERCOMMIT_MEMORY:
+  return mem->vmacct.overcommit_memory;
+ case MEM_CGROUP_OVERCOMMIT_RATIO:
+  return mem->vmacct.overcommit_ratio;
+ default:
+  BUG();
+ }
+}
+
+static int mem_cgroup_vmacct_write_s64(struct cgroup *cont, struct cftype *cft,
+           s64 val)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ enum mem_cgroup_vmacct_index type = cft->private;
+ int ret = 0;
+
+ cgroup_lock();
+ if (cgroup_is_removed(cont)) {
+  cgroup_unlock();
+  return -ENODEV;
+ }
+
+ switch (type) {
+ case MEM_CGROUP_OVERCOMMIT_MEMORY:
+  mem->vmacct.overcommit_memory = (int)val;
+  break;
+ case MEM_CGROUP_OVERCOMMIT_RATIO:
+  mem->vmacct.overcommit_ratio = (int)val;
+  break;
+ default:
+  ret = -EINVAL;
+  break;
+ }
+ cgroup_unlock();
+
+ return ret;
+
+}
```

```
+
 static int mem_cgroup_reset(struct cgroup *cont, unsigned int event)
 {
  struct mem_cgroup *mem;
@@ -1086,6 +1184,22 @@ static struct cftype mem_cgroup_files[] = {
   .name = "stat",
   .read_map = mem_control_stat_show,
  },
+ {
+  .name = "overcommit_memory",
+  .private = MEM_CGROUP_OVERCOMMIT_MEMORY,
+  .read_s64 = mem_cgroup_vmacct_read_s64,
+  .write_s64 = mem_cgroup_vmacct_write_s64,
+ },
+ {
+  .name = "overcommit_ratio",
+  .private = MEM_CGROUP_OVERCOMMIT_RATIO,
+  .read_s64 = mem_cgroup_vmacct_read_s64,
+  .write_s64 = mem_cgroup_vmacct_write_s64,
+ },
+ {
+  .name = "overcommit_as",
+  .read = mem_cgroup_committed_read,
+ },
 };

 static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
--
1.5.4.3
```

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