
Subject: [PATCH 33/33] memory-controller-make-charging-gfp-mask-aware
Posted by [Paul Menage](#) on Mon, 17 Sep 2007 21:03:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

Nick Piggin pointed out that swap cache and page cache addition routines could be called from non GFP_KERNEL contexts. This patch makes the charging routine aware of the gfp context. Charging might fail if the cgroup is over it's limit, in which case a suitable error is returned.

This patch was tested on a Powerpc box. I am still looking at being able to test the path, through which allocations happen in non GFP_KERNEL contexts.

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

```
include/linux/memcontrol.h | 12 ++++++-----
include/linux/swap.h       | 3 +-
mm/filemap.c               | 2 +-
mm/memcontrol.c            | 24 ++++++-----
mm/memory.c                | 10 +++++-----
mm/migrate.c               | 2 +-
mm/swap_state.c            | 2 +-
mm/swapfile.c              | 2 +-
mm/vmscan.c                | 5 +++--
9 files changed, 39 insertions(+), 23 deletions(-)
```

```
diff -puN include/linux/memcontrol.h~memory-controller-make-charging-gfp-mask-aware
include/linux/memcontrol.h
--- a/include/linux/memcontrol.h~memory-controller-make-charging-gfp-mask-aware
+++ a/include/linux/memcontrol.h
@@ -32,7 +32,8 @@ extern void mm_free_cgroup(struct mm_
extern void page_assign_page_cgroup(struct page *page,
    struct page_cgroup *pc);
extern struct page_cgroup *page_get_page_cgroup(struct page *page);
-extern int mem_cgroup_charge(struct page *page, struct mm_struct *mm);
+extern int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
+    gfp_t gfp_mask);
extern void mem_cgroup_uncharge(struct page_cgroup *pc);
extern void mem_cgroup_move_lists(struct page_cgroup *pc, bool active);
extern unsigned long mem_cgroup_isolate_pages(unsigned long nr_to_scan,
@@ -42,7 +43,8 @@ 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);
+extern int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm,
+   gfp_t gfp_mask);
extern struct mem_cgroup *mm_cgroup(struct mm_struct *mm);

static inline void mem_cgroup_uncharge_page(struct page *page)
@@ -70,7 +72,8 @@ static inline struct page_cgroup *pag
    return NULL;
}

-static inline int mem_cgroup_charge(struct page *page, struct mm_struct *mm)
+static inline int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
+   gfp_t gfp_mask)
{
    return 0;
}
@@ -89,7 +92,8 @@ static inline void mem_cgroup_move_li
}

static inline int mem_cgroup_cache_charge(struct page *page,
-   struct mm_struct *mm)
+   struct mm_struct *mm,
+   gfp_t gfp_mask)
{
    return 0;
}
diff -puN include/linux/swap.h~memory-controller-make-charging-gfp-mask-aware
include/linux/swap.h
--- a/include/linux/swap.h~memory-controller-make-charging-gfp-mask-aware
+++ a/include/linux/swap.h
@@ -191,7 +191,8 @@ extern void swap_setup(void);
/* linux/mm/vmscan.c */
extern unsigned long try_to_free_pages(struct zone **zones, int order,
    gfp_t gfp_mask);
-extern unsigned long try_to_free_mem_cgroup_pages(struct mem_cgroup *mem);
+extern unsigned long try_to_free_mem_cgroup_pages(struct mem_cgroup *mem,
+   gfp_t gfp_mask);
extern int __isolate_lru_page(struct page *page, int mode);
extern unsigned long shrink_all_memory(unsigned long nr_pages);
extern int vm_swappiness;
diff -puN mm/filemap.c~memory-controller-make-charging-gfp-mask-aware mm/filemap.c
--- a/mm/filemap.c~memory-controller-make-charging-gfp-mask-aware
+++ a/mm/filemap.c
@@ -444,7 +444,7 @@ int add_to_page_cache(struct page *page,

    if (error == 0) {

```

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

diff -puN mm/memcontrol.c~memory-controller-make-charging-gfp-mask-aware mm/memcontrol.c

--- a/mm/memcontrol.c~memory-controller-make-charging-gfp-mask-aware

+++ a/mm/memcontrol.c

@@ -261,7 +261,8 @@ unsigned long mem_cgroup_isolate_page

* 0 if the charge was successful

* < 0 if the cgroup is over its limit

*/

```
-int mem_cgroup_charge(struct page *page, struct mm_struct *mm)
```

```
+int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
```

```
+ gfp_t gfp_mask)
```

```
{
```

```
  struct mem_cgroup *mem;
```

```
  struct page_cgroup *pc, *race_pc;
```

@@ -287,7 +288,7 @@ int mem_cgroup_charge(struct page *pa

```
  unlock_page_cgroup(page);
```

```
- pc = kzalloc(sizeof(struct page_cgroup), GFP_KERNEL);
```

```
+ pc = kzalloc(sizeof(struct page_cgroup), gfp_mask);
```

```
  if (pc == NULL)
```

```
    goto err;
```

@@ -314,7 +315,14 @@ int mem_cgroup_charge(struct page *pa

* the cgroup limit.

*/

```
  while (res_counter_charge(&mem->res, PAGE_SIZE)) {
```

```
- if (try_to_free_mem_cgroup_pages(mem))
```

```
+ bool is_atomic = gfp_mask & GFP_ATOMIC;
```

```
+ /*
```

```
+ * We cannot reclaim under GFP_ATOMIC, fail the charge
```

```
+ */
```

```
+ if (is_atomic)
```

```
+   goto noreclaim;
```

```
+
```

```
+ if (try_to_free_mem_cgroup_pages(mem, gfp_mask))
```

```
  continue;
```

```
/*
```

@@ -338,9 +346,10 @@ int mem_cgroup_charge(struct page *pa

```
  congestion_wait(WRITE, HZ/10);
```

```
  continue;
```

```
}
```

```
-
```

```

+noreclaim:
    css_put(&mem->css);
- mem_cgroup_out_of_memory(mem);
+ if (!is_atomic)
+ mem_cgroup_out_of_memory(mem);
    goto free_pc;
}

@@ -381,7 +390,8 @@ err:
/*
 * See if the cached pages should be charged at all?
 */
-int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm)
+int mem_cgroup_cache_charge(struct page *page, struct mm_struct *mm,
+ gfp_t gfp_mask)
{
    struct mem_cgroup *mem;
    if (!mm)
@@ -389,7 +399,7 @@ int mem_cgroup_cache_charge(struct pa

    mem = rcu_dereference(mm->mem_cgroup);
    if (mem->control_type == MEM_CGROUP_TYPE_ALL)
- return mem_cgroup_charge(page, mm);
+ return mem_cgroup_charge(page, mm, gfp_mask);
    else
        return 0;
}
diff -puN mm/memory.c~memory-controller-make-charging-gfp-mask-aware mm/memory.c
--- a/mm/memory.c~memory-controller-make-charging-gfp-mask-aware
+++ a/mm/memory.c
@@ -1137,7 +1137,7 @@ static int insert_page(struct mm_struct
    pte_t *pte;
    spinlock_t *ptl;

- retval = mem_cgroup_charge(page, mm);
+ retval = mem_cgroup_charge(page, mm, GFP_KERNEL);
    if (retval)
        goto out;

@@ -1638,7 +1638,7 @@ gotten:
    goto oom;
    cow_user_page(new_page, old_page, address, vma);

- if (mem_cgroup_charge(new_page, mm))
+ if (mem_cgroup_charge(new_page, mm, GFP_KERNEL))
    goto oom_free_new;

/*

```

```

@@ -2101,7 +2101,7 @@ static int do_swap_page(struct mm_struct
}

delayacct_clear_flag(DELAYACCT_PF_SWAPIN);
- if (mem_cgroup_charge(page, mm)) {
+ if (mem_cgroup_charge(page, mm, GFP_KERNEL)) {
    ret = VM_FAULT_OOM;
    goto out;
}
@@ -2185,7 +2185,7 @@ static int do_anonymous_page(struct mm_s
if (!page)
    goto oom;

- if (mem_cgroup_charge(page, mm))
+ if (mem_cgroup_charge(page, mm, GFP_KERNEL))
    goto oom_free_page;

    entry = mk_pte(page, vma->vm_page_prot);
@@ -2320,7 +2320,7 @@ static int __do_fault(struct mm_struct *

}

- if (mem_cgroup_charge(page, mm)) {
+ if (mem_cgroup_charge(page, mm, GFP_KERNEL)) {
    ret = VM_FAULT_OOM;
    goto out;
}
diff -puN mm/migrate.c~memory-controller-make-charging-gfp-mask-aware mm/migrate.c
--- a/mm/migrate.c~memory-controller-make-charging-gfp-mask-aware
+++ a/mm/migrate.c
@@ -158,7 +158,7 @@ static void remove_migration_pte(struct
    return;
}

- if (mem_cgroup_charge(new, mm)) {
+ if (mem_cgroup_charge(new, mm, GFP_KERNEL)) {
    pte_unmap(pte);
    return;
}
diff -puN mm/swapfile.c~memory-controller-make-charging-gfp-mask-aware mm/swapfile.c
--- a/mm/swapfile.c~memory-controller-make-charging-gfp-mask-aware
+++ a/mm/swapfile.c
@@ -510,7 +510,7 @@ unsigned int count_swap_pages(int type,
static int unuse_pte(struct vm_area_struct *vma, pte_t *pte,
    unsigned long addr, swp_entry_t entry, struct page *page)
{
- if (mem_cgroup_charge(page, vma->vm_mm))
+ if (mem_cgroup_charge(page, vma->vm_mm, GFP_KERNEL))

```

```
return -ENOMEM;
```

```
inc_mm_counter(vma->vm_mm, anon_rss);
diff -puN mm/swap_state.c~memory-controller-make-charging-gfp-mask-aware mm/swap_state.c
--- a/mm/swap_state.c~memory-controller-make-charging-gfp-mask-aware
+++ 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_cache_charge(page, current->mm);
+ error = mem_cgroup_cache_charge(page, current->mm, gfp_mask);
    if (error)
        goto out;

diff -puN mm/vmscan.c~memory-controller-make-charging-gfp-mask-aware mm/vmscan.c
--- a/mm/vmscan.c~memory-controller-make-charging-gfp-mask-aware
+++ a/mm/vmscan.c
@@ -1350,10 +1350,11 @@ unsigned long try_to_free_pages(struct z
#define ZONE_USERPAGES ZONE_NORMAL
#endif

-unsigned long try_to_free_mem_cgroup_pages(struct mem_cgroup *mem_cont)
+unsigned long try_to_free_mem_cgroup_pages(struct mem_cgroup *mem_cont,
+      gfp_t gfp_mask)
{
    struct scan_control sc = {
- .gfp_mask = GFP_KERNEL,
+ .gfp_mask = gfp_mask,
    .may_writepage = !laptop_mode,
    .may_swap = 1,
    .swap_cluster_max = SWAP_CLUSTER_MAX,

--
```

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