Subject: Re: [PATCH -mm 4/5] swapcgroup (v3): modify vm_swap_full()
Posted by KAMEZAWA Hiroyuki on Fri, 04 Jul 2008 09:58:45 GMT
View Forum Message <> Reply to Message

On Fri, 4 Jul 2008 15:22:44 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> /* Swap 50% full? Release swapcache more aggressively.. */
> -#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
> +#define vm_swap_full(memcg) ((nr_swap_pages*2 < total_swap_pages) \
> + || swap_cgroup_vm_swap_full(memcg))
> +
>
Maybe nitpick but I like
==
vm_swap_full(page) ((nr_swap_pages *2 < total_swap_pages)
|| swap_cgroup_vm_swap_full_page(page))
==
rather than vm_swap_full(memcg)

And could you change this to inline funcition ?

> /* linux/mm/page_alloc.c */
> extern unsigned long totalram_pages;
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index d16d0a5..160fca1 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -1296,5 +1296,23 @@ void swap_cgroup_uncharge(struct swap_info_struct *si,
> css_put(&mem->css);
> }
> }
> +
> +int swap_cgroup_vm_swap_full(struct mem_cgroup *memcg)
> +{
> + u64 usage;
> + u64 limit;
> +
> + if (memcg)
> + css_get(&memcg->css);
> + else
> + return 0;
> +
> + usage = res_counter_read_u64(&memcg->swap_res, RES_USAGE);
> + limit = res_counter_read_u64(&memcg->swap_res, RES_LIMIT);
> +
> + css_put(&memcg->css);
> +

> + return usage * 2 > limit;
> +}
> #endif
>

How about this under above my changes ?
==
```
int memcg_swap_full(struct mem_cgroup *mem)
{
 if (!mem)
  return 0;
 usage = res_counter_read_u64(&memcg->swap_res, RES_USAGE);
 limit = res_counter_read_u64(&memcg->swap_res, RES_LIMIT);

 return (usage *2 > limit);
}

int swap_cgroup_vm_swap_full_page(struct page *page)
{
 struct page_cgroup *pc;
 int ret = 0;

 if (mem_cgroup_subsys.disabled)
  return 0;

 if (!PageSwapCache(page))
  return 0;

 entry.val = page_private(page);
 p = swap_info_get(entry);
 if (!p)
  goto out;

 mem = p->memcg[swp_offset(entry)];
 /* becasue we get swap_lock here, access to memcg is safe.*/
 ret = memcg_swap_full(mem);
 spin_unlock(&swap_lock);
out:
 return ret;
}
```
==

Thanks,
-Kame

> diff --git a/mm/memory.c b/mm/memory.c

> index 536d748..afcf737 100644
> --- a/mm/memory.c
> +++ b/mm/memory.c
> @@ -2230,7 +2230,9 @@ static int do_swap_page(struct mm_struct *mm, struct
vm_area_struct *vma,
> page_add_anon_rmap(page, vma, address);
>
> swap_free(entry);
> - if (vm_swap_full() || (vma->vm_flags & VM_LOCKED) || PageMlocked(page))
> + if (vm_swap_full(page_to_memcg(page))
> + || (vma->vm_flags & VM_LOCKED)
> + || PageMlocked(page))
> remove_exclusive_swap_page(page);
> unlock_page(page);
>
> diff --git a/mm/swapfile.c b/mm/swapfile.c
> index 57798c5..6a863bc 100644
> --- a/mm/swapfile.c
> +++ b/mm/swapfile.c
> @@ -28,6 +28,7 @@
> #include <linux/capability.h>
> #include <linux/syscalls.h>
> #include <linux/memcontrol.h>
> +#include <linux/cgroup.h>
>
> #include <asm/pgtable.h>
> #include <asm/tlbflush.h>
> @@ -447,7 +448,8 @@ void free_swap_and_cache(swp_entry_t entry)
> /* Only cache user (+us), or swap space full? Free it! */
> /* Also recheck PageSwapCache after page is locked (above) */
> if (PageSwapCache(page) && !PageWriteback(page) &&
> - (one_user || vm_swap_full())) {
> + (one_user
> + || vm_swap_full(page_to_memcg(page)))) {
> delete_from_swap_cache(page);
> SetPageDirty(page);
> }
> @@ -1889,3 +1891,37 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
> *offset = ++toff;
> return nr_pages? ++nr_pages: 0;
> }
> +
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> +/*
> + * returns mem_cgroup to which the swap cache page is charged as swap.
> + * should be called with the page locked.
> + */
> +struct mem_cgroup *page_to_memcg(struct page *page)

> +{
> + struct swap_info_struct *p;
> + struct mem_cgroup *mem = NULL;
> + swp_entry_t entry;
> +
> + BUG_ON(!PageLocked(page));
> +
> + if (mem_cgroup_subsys.disabled)
> + goto out;
> +
> + if (!PageSwapCache(page))
> + goto out;
> +
> + entry.val = page_private(page);
> + p = swap_info_get(entry);
> + if (!p)
> + goto out;
> +
> + mem = p->memcg[swp_offset(entry)];
> +
> + spin_unlock(&swap_lock);
> +
> +out:
> + return mem;
> +}

> +#endif
> +
> diff --git a/mm/vmscan.c b/mm/vmscan.c
> index 9a5e423..ac45993 100644
> --- a/mm/vmscan.c
> +++ b/mm/vmscan.c
> @@ -752,7 +752,7 @@ cull_mlocked:
>
> activate_locked:
>   /* Not a candidate for swapping, so reclaim swap space. */
> - if (PageSwapCache(page) && vm_swap_full())
> + if (PageSwapCache(page) && vm_swap_full(page_to_memcg(page)))
>   remove_exclusive_swap_page_ref(page);
>   VM_BUG_ON(PageActive(page));
>   SetPageActive(page);
> @@ -1317,7 +1317,7 @@ static void shrink_active_list(unsigned long nr_pages, struct zone
*zone,
>   __mod_zone_page_state(zone, NR_LRU_BASE + lru, pgmoved);
>   pgmoved = 0;
>   spin_unlock_irq(&zone->lru_lock);

> -   if (vm_swap_full())
> +   if (vm_swap_full(sc->mem_cgroup))
>   pagevec_swap_free(&pvec);
>   __pagevec_release(&pvec);
>   spin_lock_irq(&zone->lru_lock);
> @@ -1328,7 +1328,7 @@ static void shrink_active_list(unsigned long nr_pages, struct zone
*zone,
>   __count_zone_vm_events(PGREFILL, zone, pgscanned);
>   __count_vm_events(PGDEACTIVATE, pgdeactivate);
>   spin_unlock_irq(&zone->lru_lock);
> - if (vm_swap_full())
> + if (vm_swap_full(sc->mem_cgroup))
>   pagevec_swap_free(&pvec);
>
>   pagevec_release(&pvec);
>

_____

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