Subject: Re: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by KAMEZAWA Hiroyuki on Thu, 22 May 2008 07:35:25 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 15:20:05 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> +int swap_cgroup_charge(struct page *page,
> + struct swap_info_struct *si,
> + unsigned long offset)
> +{
> + int ret;
> + struct page_cgroup *pc;
> + struct mem_cgroup *mem;
> +
> + lock_page_cgroup(page);
> + pc = page_get_page_cgroup(page);
> + if (unlikely(!pc))
> + mem = &init_mem_cgroup;
> + else
> + mem = pc->mem_cgroup;
> + unlock_page_cgroup(page);

If !pc, the page is used before memory controller is available. But is it
good to be charged to init_mem_cgroup() ?
How about returning 'failure' in this case ? I think returning 'failure' here
is not so bad.

> +
> + css_get(&mem->css);

move this css_get() before unlock_page_cgroup() is safer.

> + ret = res_counter_charge(&mem->swap_res, PAGE_SIZE);
> + if (!ret)
> + si->memcg[offset] = mem;
> + else
> + css_put(&mem->css);
> +
> + return ret;
> +}
> +
> +void swap_cgroup_uncharge(struct swap_info_struct *si,
> + unsigned long offset)
> +{
> + struct mem_cgroup *mem = si->memcg[offset];

```
> +
> + /* "mem" would be NULL:
> +  * 1. when get_swap_page() failed at charging swap_cgroup,
> +  *    and called swap_entry_free().
> +  * 2. when this swap entry had been assigned by
> +  *    get_swap_page_of_type() (via SWSUSP?).
> +  */
> + if (mem) {
> +  res_counter_uncharge(&mem->swap_res, PAGE_SIZE);
> +  si->memcg[offset] = NULL;
> +  css_put(&mem->css);
> + }
> +}
> +#endif
> +
> diff --git a/mm/shmem.c b/mm/shmem.c
> index 95b056d..69f8909 100644
> --- a/mm/shmem.c
> +++ b/mm/shmem.c
> @@ -1029,7 +1029,7 @@ static int shmem_writepage(struct page *page, struct
writeback_control *wbc)
>     * want to check if there's a redundant swappage to be discarded.
>     */
>    if (wbc->for_reclaim)
> -  swap = get_swap_page();
> +  swap = get_swap_page(page);
>    else
>     swap.val = 0;
>
> diff --git a/mm/swap_state.c b/mm/swap_state.c
> index 676e191..a78d617 100644
> --- a/mm/swap_state.c
> +++ b/mm/swap_state.c
> @@ -130,7 +130,7 @@ int add_to_swap(struct page * page, gfp_t gfp_mask)
>    BUG_ON(!PageUptodate(page));
>
>    for (;;) {
> -  entry = get_swap_page();
> +  entry = get_swap_page(page);
>     if (!entry.val)
>      return 0;
>
> diff --git a/mm/swapfile.c b/mm/swapfile.c
> index 232bf20..682b71e 100644
> --- a/mm/swapfile.c
> +++ b/mm/swapfile.c
> @@ -172,7 +172,10 @@ no_page:
>    return 0;
```

```
>  }
>
> -swp_entry_t get_swap_page(void)
> +/* get_swap_page() calls this */
> +static int swap_entry_free(struct swap_info_struct *, unsigned long);
> +
> +swp_entry_t get_swap_page(struct page *page)
>  {
>   struct swap_info_struct *si;
>   pgoff_t offset;
> @@ -201,6 +204,14 @@ swp_entry_t get_swap_page(void)
>    swap_list.next = next;
>    offset = scan_swap_map(si);
>    if (offset) {
> +   /*
> +    * This should be the first use of this swap entry.
> +    * So, charge this swap entry here.
> +    */
> +   if (swap_cgroup_charge(page, si, offset)) {
> +    swap_entry_free(si, offset);
> +    goto noswap;
> +   }
>     spin_unlock(&swap_lock);
>     return swp_entry(type, offset);
>    }
> @@ -285,6 +296,7 @@ static int swap_entry_free(struct swap_info_struct *p, unsigned long
offset)
>      swap_list.next = p - swap_info;
>     nr_swap_pages++;
>     p->inuse_pages--;
> +   swap_cgroup_uncharge(p, offset);
>    }
>   }
>   return count;
>
>
> --
> To unsubscribe, send a message with 'unsubscribe linux-mm' in
> the body to majordomo@kvack.org.  For more info on Linux MM,
> see: http://www.linux-mm.org/ .
> Don't email: <a href=mailto:"dont@kvack.org"> email@kvack.org </a>
>
```