Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Pavel Emelianov on Wed, 05 Mar 2008 08:33:38 GMT
View Forum Message <> Reply to Message

Daisuke Nishimura wrote:
> Hi.
>
> Even if limiting memory usage by cgroup memory subsystem
> or isolating memory by cpuset, swap space is shared, so
> resource isolation is not enough. If one group uses up all the
> swap space, it can affect other groups.
>
> I try making a patch of swap subsystem based on memory
> subsystem, which limits swap usage per cgroup.
> It can now charge and limit the swap usage.
>
> I implemented this feature as a new subsystem,
> not as a part of memory subsystem, because I don't want to
> make big change to memcontrol.c, and even if implemented
> as other subsystem, users can manage memory and swap on
> the same cgroup directory if mount them together.
>
> Basic idea of my implementation:
> - what will be charged ?
> the number of swap entries.

This is a very obscure thing "a swap entry" for the end user. People
would prefer accounting bytes.

> - when to charge/uncharge ?
> charge at get_swap_entry(), and uncharge at swap_entry_free().
>
> - to what group charge the swap entry ?
> To determine to what swap_cgroup (corresponding to mem_cgroup in
> memory subsystem) the swap entry should be charged,
> I added a pointer to mm_struct to page_cgroup(pc->pc_mm), and
> changed the argument of get_swap_entry() from (void) to
> (struct page *). As a result, get_swap_entry() can determine
> to what swap_cgroup it should charge the swap entry
> by referring to page->page_cgroup->mm_struct->swap_cgroup.
>
> - from what group uncharge the swap entry ?
> I added to swap_info_struct a member 'struct swap_cgroup **',
> array of pointer to which swap_cgroup the swap entry is
> charged.
>
> Todo:
> - rebase new kernel, and split into some patches.

>   - Merge with memory subsystem (if it would be better), or
>   remove dependency on CONFIG_CGROUP_MEM_CONT if possible
>   (needs to make page_cgroup more generic one).

Merge is a must IMHO. I can hardly imagine a situation in which
someone would need these two separately.

>   - More tests, cleanups, and feartures   :-)
>
>
> Any comments or discussions would be appreciated.
>
> Thanks,
> Daisuke Nishimura
>
>
> Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>
>
> ---
> diff -uprN linux-2.6.24-mm1/include/linux/cgroup_subsys.h
linux-2.6.24-mm1-swaplimit/include/linux/cgroup_subsys.h
> --- linux-2.6.24-mm1/include/linux/cgroup_subsys.h 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/include/linux/cgroup_subsys.h 2008-03-03
10:56:56.000000000 +0900
> @@ -42,3 +42,9 @@ SUBSYS(mem_cgroup)
> #endif
>
> /* */
> +
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> +SUBSYS(swap)
> +#endif
> +
> +/* */
> diff -uprN linux-2.6.24-mm1/include/linux/memcontrol.h
linux-2.6.24-mm1-swaplimit/include/linux/memcontrol.h
> --- linux-2.6.24-mm1/include/linux/memcontrol.h 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/include/linux/memcontrol.h 2008-03-03 10:56:56.000000000
+0900
> @@ -29,6 +29,21 @@ struct page;
> struct mm_struct;
>
> #ifdef CONFIG_CGROUP_MEM_CONT
> +/*
> + * A page_cgroup page is associated with every page descriptor. The
> + * page_cgroup helps us identify information about the cgroup
> + */
> +struct page_cgroup {

> + struct list_head lru; /* per cgroup LRU list */
> + struct page *page;
> + struct mem_cgroup *mem_cgroup;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct mm_struct *pc_mm;
> +#endif

Try not to add new entries here.

> + atomic_t ref_cnt; /* Helpful when pages move b/w */
> +   /* mapped and cached states   */
> + int flags;
> +};
>
> extern void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p);
> extern void mm_free_cgroup(struct mm_struct *mm);
> diff -uprN linux-2.6.24-mm1/include/linux/mm_types.h
linux-2.6.24-mm1-swaplimit/include/linux/mm_types.h
> --- linux-2.6.24-mm1/include/linux/mm_types.h 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/include/linux/mm_types.h 2008-03-03 10:56:56.000000000
+0900
> @@ -233,6 +233,9 @@ struct mm_struct {
> #ifdef CONFIG_CGROUP_MEM_CONT
>   struct mem_cgroup *mem_cgroup;
> #endif
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct swap_cgroup *swap_cgroup;
> +#endif
> };
>
> #endif /* _LINUX_MM_TYPES_H */
> diff -uprN linux-2.6.24-mm1/include/linux/swap.h
linux-2.6.24-mm1-swaplimit/include/linux/swap.h
> --- linux-2.6.24-mm1/include/linux/swap.h 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/include/linux/swap.h 2008-03-03 10:56:56.000000000 +0900
> @@ -7,6 +7,7 @@
> #include <linux/list.h>
> #include <linux/memcontrol.h>
> #include <linux/sched.h>
> +#include <linux/swap_limit.h>
>
> #include <asm/atomic.h>
> #include <asm/page.h>
> @@ -141,6 +142,9 @@ struct swap_info_struct {
>   struct swap_extent *curr_swap_extent;
>   unsigned old_block_size;
>   unsigned short * swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT

> + struct swap_cgroup **swap_cgroup;
> +#endif
>   unsigned int lowest_bit;
>   unsigned int highest_bit;
>   unsigned int cluster_next;
> @@ -239,7 +243,7 @@ extern struct page *swapin_readahead(swp
> extern long total_swap_pages;
> extern unsigned int nr_swapfiles;
> extern void si_swapinfo(struct sysinfo *);
> -extern swp_entry_t get_swap_page(void);
> +extern swp_entry_t get_swap_page(struct page *);
> extern swp_entry_t get_swap_page_of_type(int);
> extern int swap_duplicate(swp_entry_t);
> extern int valid_swaphandles(swp_entry_t, unsigned long *);
> @@ -342,7 +346,7 @@ static inline int remove_exclusive_swap_
>   return 0;
> }
>
> -static inline swp_entry_t get_swap_page(void)
> +static inline swp_entry_t get_swap_page(struct page *page)
> {
>   swp_entry_t entry;
>   entry.val = 0;
> diff -uprN linux-2.6.24-mm1/include/linux/swap_limit.h
linux-2.6.24-mm1-swaplimit/include/linux/swap_limit.h
> --- linux-2.6.24-mm1/include/linux/swap_limit.h 1970-01-01 09:00:00.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/include/linux/swap_limit.h 2008-03-03 10:56:56.000000000
+0900
> @@ -0,0 +1,65 @@
> +/*
> + * swap_limit.h
> + *
> + */
> +#ifndef _LINUX_SWAP_LIMIT_H
> +#define _LINUX_SWAP_LIMIT_H
> +
> +#include <linux/swap.h>
> +#include <linux/cgroup.h>
> +#include <linux/res_counter.h>
> +
> +struct swap_cgroup;
> +struct swap_info_struct;
> +
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> +struct swap_cgroup {
> + struct cgroup_subsys_state css;
> + struct res_counter res;
> +};

> +
> +static inline struct swap_cgroup *swap_cgroup_from_cgrp(struct cgroup *cgrp)
> +{
> + return container_of(cgroup_subsys_state(cgrp, swap_subsys_id),
> + struct swap_cgroup,
> + css);
> +}
> +
> +static inline struct swap_cgroup *swap_cgroup_from_task(struct task_struct *p)
> +{
> + return container_of(task_subsys_state(p, swap_subsys_id),
> + struct swap_cgroup, css);
> +}
> +
> +extern int swap_cgroup_charge(struct page *page,
> + struct swap_info_struct *si,
> + unsigned long offset);
> +extern void swap_cgroup_uncharge(struct swap_info_struct *si,
> + unsigned long offset);
> +
> +#else /* CONFIG_CGROUP_SWAP_LIMIT */
> +static inline struct swap_cgroup *swap_cgroup_from_cgrp(struct cgroup *cgrp)
> +{
> + return NULL;
> +}
> +
> +static inline struct swap_cgroup *swap_cgroup_from_task(struct task_struct *p)
> +{
> + return NULL;
> +}
> +
> +static inline int swap_cgroup_charge(struct page *page,
> + struct swap_info_struct *si,
> + unsigned long offset)
> +{
> + return 0;
> +}
> +
> +static inline void swap_cgroup_uncharge(struct swap_info_struct *si,
> + unsigned long offset)
> +{
> +}
> +
> +#endif
> +
> +#endif
> diff -uprN linux-2.6.24-mm1/init/Kconfig linux-2.6.24-mm1-swaplimit/init/Kconfig
> --- linux-2.6.24-mm1/init/Kconfig 2008-02-04 14:34:24.000000000 +0900

> +++ linux-2.6.24-mm1-swaplimit/init/Kconfig 2008-03-03 10:56:56.000000000 +0900
> @@ -383,6 +383,12 @@ config CGROUP_MEM_CONT
>   Provides a memory controller that manages both page cache and
>   RSS memory.
>
> +config CGROUP_SWAP_LIMIT
> + bool "cgroup subsystem for swap"
> + depends on CGROUP_MEM_CONT && SWAP
> + help
> +   Provides a swap controller that manages and limits swap usage.
> +
> config PROC_PID_CPUSET
>   bool "Include legacy /proc/<pid>/cpuset file"
>   depends on CPUSETS
> diff -uprN linux-2.6.24-mm1/mm/Makefile linux-2.6.24-mm1-swaplimit/mm/Makefile
> --- linux-2.6.24-mm1/mm/Makefile 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/mm/Makefile 2008-03-03 10:56:56.000000000 +0900
> @@ -32,4 +32,5 @@ obj-$(CONFIG_MIGRATION) += migrate.o
> obj-$(CONFIG_SMP) += allocpercpu.o
> obj-$(CONFIG_QUICKLIST) += quicklist.o
> obj-$(CONFIG_CGROUP_MEM_CONT) += memcontrol.o
> +obj-$(CONFIG_CGROUP_SWAP_LIMIT) += swap_limit.o
>
> diff -uprN linux-2.6.24-mm1/mm/memcontrol.c linux-2.6.24-mm1-swaplimit/mm/memcontrol.c
> --- linux-2.6.24-mm1/mm/memcontrol.c 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/mm/memcontrol.c 2008-03-03 10:56:56.000000000 +0900
> @@ -19,6 +19,7 @@
>
> #include <linux/res_counter.h>
> #include <linux/memcontrol.h>
> +#include <linux/swap_limit.h>
> #include <linux/cgroup.h>
> #include <linux/mm.h>
> #include <linux/smp.h>
> @@ -146,18 +147,6 @@ struct mem_cgroup {
> #define PAGE_CGROUP_LOCK_BIT 0x0
> #define PAGE_CGROUP_LOCK   (1 << PAGE_CGROUP_LOCK_BIT)
>
> -/*
> - * A page_cgroup page is associated with every page descriptor. The
> - * page_cgroup helps us identify information about the cgroup
> - */
> -struct page_cgroup {
> - struct list_head lru; /* per cgroup LRU list */
> - struct page *page;
> - struct mem_cgroup *mem_cgroup;
> - atomic_t ref_cnt; /* Helpful when pages move b/w */
> -   /* mapped and cached states   */

> - int flags;
> -};
> #define PAGE_CGROUP_FLAG_CACHE (0x1) /* charged as cache */
> #define PAGE_CGROUP_FLAG_ACTIVE (0x2) /* page is active in this cgroup */
>
> @@ -254,15 +243,27 @@ struct mem_cgroup *mem_cgroup_from_task(
> void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p)
> {
>   struct mem_cgroup *mem;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct swap_cgroup *swap;
> +#endif
>
>   mem = mem_cgroup_from_task(p);
>   css_get(&mem->css);
>   mm->mem_cgroup = mem;
> +
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + swap = swap_cgroup_from_task(p);
> + css_get(&swap->css);
> + mm->swap_cgroup = swap;
> +#endif
> }
>
> void mm_free_cgroup(struct mm_struct *mm)
> {
>   css_put(&mm->mem_cgroup->css);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + css_put(&mm->swap_cgroup->css);
> +#endif
> }
>
> static inline int page_cgroup_locked(struct page *page)
> @@ -664,6 +665,10 @@ retry:
>   pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
>   if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
>   pc->flags |= PAGE_CGROUP_FLAG_CACHE;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + atomic_inc(&mm->mm_count);
> + pc->pc_mm = mm;
> +#endif

What kernel is this patch for? I cannot find this code in 2.6.25-rc3-mm1

>   if (!page || page_cgroup_assign_new_page_cgroup(page, pc)) {
>   /*
> @@ -673,6 +678,9 @@ retry:
>   */

>   res_counter_uncharge(&mem->res, PAGE_SIZE);
>   css_put(&mem->css);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + mmdrop(mm);
> +#endif
>   kfree(pc);
>   if (!page)
>   goto done;
> @@ -744,6 +752,9 @@ void mem_cgroup_uncharge(struct page_cgr
>   if (clear_page_cgroup(page, pc) == pc) {
>   mem = pc->mem_cgroup;
>   css_put(&mem->css);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> +   mmdrop(pc->pc_mm);
> +#endif
>   res_counter_uncharge(&mem->res, PAGE_SIZE);
>   spin_lock_irqsave(&mz->lru_lock, flags);
>   __mem_cgroup_remove_list(pc);
> @@ -859,6 +870,9 @@ retry:
>   atomic_set(&pc->ref_cnt, 0);
>   if (clear_page_cgroup(page, pc) == pc) {
>   css_put(&mem->css);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> +   mmdrop(pc->pc_mm);
> +#endif
>   res_counter_uncharge(&mem->res, PAGE_SIZE);
>   __mem_cgroup_remove_list(pc);
>   kfree(pc);
> diff -uprN linux-2.6.24-mm1/mm/shmem.c linux-2.6.24-mm1-swaplimit/mm/shmem.c
> --- linux-2.6.24-mm1/mm/shmem.c 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/mm/shmem.c 2008-03-03 10:56:56.000000000 +0900
> @@ -1024,7 +1024,7 @@ static int shmem_writepage(struct page *
>   * 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 -uprN linux-2.6.24-mm1/mm/swap_limit.c linux-2.6.24-mm1-swaplimit/mm/swap_limit.c
> --- linux-2.6.24-mm1/mm/swap_limit.c 1970-01-01 09:00:00.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/mm/swap_limit.c 2008-03-05 14:39:23.000000000 +0900
> @@ -0,0 +1,194 @@
> +/*
> + * swap_limit.c - SWAP controller (based on memcontrol.c)
> + *
> + */

```
> +
> +#include <linux/err.h>
> +#include <linux/fs.h>
> +#include <linux/types.h>
> +#include <linux/sched.h>
> +#include <linux/mm.h>
> +#include <linux/swap.h>
> +#include <linux/rcupdate.h>
> +#include <linux/cgroup.h>
> +#include <linux/res_counter.h>
> +#include <linux/memcontrol.h>
> +#include <linux/swap_limit.h>
> +
> +static struct swap_cgroup init_swap_cgroup;
> +
> +int swap_cgroup_charge(struct page *page,
> +   struct swap_info_struct *si,
> +   unsigned long offset)
> +{
> + int ret;
> + struct page_cgroup *pc;
> + struct mm_struct *mm;
> + struct swap_cgroup *swap;
> +
> + BUG_ON(!page);
> +
> + /*
> +  * Pages to be swapped out should have been charged by memory cgroup,
> +  * but very rarely, pc would be NULL (pc is not reliable without lock,
> +  * so I should fix here).
> +  * In such cases, we charge the init_mm now.
> +  */
> + pc = page_get_page_cgroup(page);
> + if (WARN_ON(!pc))
> +  mm = &init_mm;
> + else
> +  mm = pc->pc_mm;
> + BUG_ON(!mm);
> +
> + rcu_read_lock();
> + swap = rcu_dereference(mm->swap_cgroup);
> + rcu_read_unlock();
> + BUG_ON(!swap);
> +
> + ret = res_counter_charge(&swap->res, PAGE_SIZE);
> + if (!ret) {
> +  css_get(&swap->css);
> +  si->swap_cgroup[offset] = swap;
```

```
> + }
> +
> + return ret;
> +}
> +
> +void swap_cgroup_uncharge(struct swap_info_struct *si, unsigned long offset)
> +{
> + struct swap_cgroup *swap = si->swap_cgroup[offset];
> +
> + /*
> +  * "swap" 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 (swap) {
> +  res_counter_uncharge(&swap->res, PAGE_SIZE);
> +  si->swap_cgroup[offset] = NULL;
> +  css_put(&swap->css);
> + }
> +}
> +
> +static struct cgroup_subsys_state *swap_cgroup_create(struct cgroup_subsys *ss,
> +            struct cgroup *cgrp)
> +{
> + struct swap_cgroup *swap;
> +
> + if (unlikely((cgrp->parent) == NULL)) {
> +  swap = &init_swap_cgroup;
> +  init_mm.swap_cgroup = swap;
> + } else
> +  swap = kzalloc(sizeof(struct swap_cgroup), GFP_KERNEL);
> +
> + if (swap == NULL)
> +  return ERR_PTR(-ENOMEM);
> +
> + res_counter_init(&swap->res);
> +
> + return &swap->css;
> +}
> +
> +static void swap_cgroup_destroy(struct cgroup_subsys *ss, struct cgroup *cgrp)
> +{
> + kfree(swap_cgroup_from_cgrp(cgrp));
> +}
> +
> +static ssize_t swap_cgroup_read(struct cgroup *cgrp,
```

```
> +    struct cftype *cft, struct file *file,
> +    char __user *userbuf, size_t nbytes,
> +    loff_t *ppos)
> +{
> + return res_counter_read(&swap_cgroup_from_cgrp(cgrp)->res,
> +    cft->private, userbuf, nbytes, ppos,
> +    NULL);
> +}
> +
> +static int swap_cgroup_write_strategy(char *buf, unsigned long long *tmp)
> +{
> + *tmp = memparse(buf, &buf);
> + if (*buf != '\0')
> +  return -EINVAL;
> +
> + /*
> +  * Round up the value to the closest page size
> +  */
> + *tmp = ((*tmp + PAGE_SIZE - 1) >> PAGE_SHIFT) << PAGE_SHIFT;
> + return 0;
> +}
> +
> +static ssize_t swap_cgroup_write(struct cgroup *cgrp, struct cftype *cft,
> +     struct file *file, const char __user *userbuf,
> +     size_t nbytes, loff_t *ppos)
> +{
> + return res_counter_write(&swap_cgroup_from_cgrp(cgrp)->res,
> +     cft->private, userbuf, nbytes, ppos,
> +     swap_cgroup_write_strategy);
> +}
> +
> +static struct cftype swap_files[] = {
> + {
> +  .name = "usage_in_bytes",
> +  .private = RES_USAGE,
> +  .read = swap_cgroup_read,
> + },
> + {
> +  .name = "limit_in_bytes",
> +  .private = RES_LIMIT,
> +  .write = swap_cgroup_write,
> +  .read = swap_cgroup_read,
> + },
> + {
> +  .name = "failcnt",
> +  .private = RES_FAILCNT,
> +  .read = swap_cgroup_read,
> + },
```

```
> +};
> +
> +static int swap_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cgrp)
> +{
> + return cgroup_add_files(cgrp, ss, swap_files, ARRAY_SIZE(swap_files));
> +}
> +
> +static void swap_cgroup_move_task(struct cgroup_subsys *ss,
> +       struct cgroup *cgrp,
> +       struct cgroup *old_cgrp,
> +       struct task_struct *p)
> +{
> + struct mm_struct *mm;
> + struct swap_cgroup *swap, *old_swap;
> +
> + mm = get_task_mm(p);
> + if (mm == NULL)
> +  return;
> +
> + swap = swap_cgroup_from_cgrp(cgrp);
> + old_swap = swap_cgroup_from_cgrp(old_cgrp);
> +
> + if (swap == old_swap)
> +  goto out;
> +
> + if (p->tgid != p->pid)
> +  goto out;
> +
> + css_get(&swap->css);
> + rcu_assign_pointer(mm->swap_cgroup, swap);
> + css_put(&old_swap->css);
> +
> +out:
> + mmput(mm);
> + return;
> +}
> +
> +struct cgroup_subsys swap_subsys = {
> + .name = "swap",
> + .create = swap_cgroup_create,
> + .destroy = swap_cgroup_destroy,
> + .populate = swap_cgroup_populate,
> + .subsys_id = swap_subsys_id,
> + .attach = swap_cgroup_move_task,
> + .early_init = 0,
> +};
> diff -uprN linux-2.6.24-mm1/mm/swap_state.c linux-2.6.24-mm1-swaplimit/mm/swap_state.c
> --- linux-2.6.24-mm1/mm/swap_state.c 2008-02-04 14:34:24.000000000 +0900
```

```
> +++ linux-2.6.24-mm1-swaplimit/mm/swap_state.c 2008-03-03 10:56:56.000000000 +0900
> @@ -128,7 +128,7 @@ int add_to_swap(struct page * page, gfp_
>   BUG_ON(!PageUptodate(page));
>
>   for (;;) {
> -  entry = get_swap_page();
> +  entry = get_swap_page(page);
>    if (!entry.val)
>     return 0;
>
> diff -uprN linux-2.6.24-mm1/mm/swapfile.c linux-2.6.24-mm1-swaplimit/mm/swapfile.c
> --- linux-2.6.24-mm1/mm/swapfile.c 2008-02-04 14:34:24.000000000 +0900
> +++ linux-2.6.24-mm1-swaplimit/mm/swapfile.c 2008-03-03 10:56:56.000000000 +0900
> @@ -28,6 +28,7 @@
>  #include <linux/capability.h>
>  #include <linux/syscalls.h>
>  #include <linux/memcontrol.h>
> +#include <linux/swap_limit.h>
>
>  #include <asm/pgtable.h>
>  #include <asm/tlbflush.h>
> @@ -172,7 +173,10 @@ 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 +205,16 @@ 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 now.
> +    */
> +   if (swap_cgroup_charge(page, si, offset)) {
> +    /* should free this entry */

:) Please, don't create comments, that duplicate the next line.

> +    swap_entry_free(si, offset);
> +
```

```
> +    goto noswap;
> +   }
>     spin_unlock(&swap_lock);
>     return swp_entry(type, offset);
>    }
> @@ -285,6 +299,7 @@ static int swap_entry_free(struct swap_i
>      swap_list.next = p - swap_info;
>     nr_swap_pages++;
>     p->inuse_pages--;
> +   swap_cgroup_uncharge(p, offset);
>    }
>   }
>   return count;
> @@ -1207,6 +1222,9 @@ asmlinkage long sys_swapoff(const char _
>  {
>   struct swap_info_struct * p = NULL;
>   unsigned short *swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct swap_cgroup **swap_cgroup;
> +#endif
>   struct file *swap_file, *victim;
>   struct address_space *mapping;
>   struct inode *inode;
> @@ -1309,10 +1327,17 @@ asmlinkage long sys_swapoff(const char _
>   p->max = 0;
>   swap_map = p->swap_map;
>   p->swap_map = NULL;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + swap_cgroup = p->swap_cgroup;
> + p->swap_cgroup = NULL;
> +#endif
>   p->flags = 0;
>   spin_unlock(&swap_lock);
>   mutex_unlock(&swapon_mutex);
>   vfree(swap_map);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + vfree(swap_cgroup);
> +#endif
>   inode = mapping->host;
>   if (S_ISBLK(inode->i_mode)) {
>    struct block_device *bdev = I_BDEV(inode);
> @@ -1460,6 +1485,9 @@ asmlinkage long sys_swapon(const char __
>   unsigned long maxpages = 1;
>   int swapfilesize;
>   unsigned short *swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct swap_cgroup **swap_cgroup;
> +#endif
```

```
>   struct page *page = NULL;
>   struct inode *inode = NULL;
>   int did_down = 0;
> @@ -1483,6 +1511,9 @@ asmlinkage long sys_swapon(const char __
>   p->swap_file = NULL;
>   p->old_block_size = 0;
>   p->swap_map = NULL;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + p->swap_cgroup = NULL;
> +#endif
>   p->lowest_bit = 0;
>   p->highest_bit = 0;
>   p->cluster_nr = 0;
> @@ -1647,6 +1678,15 @@ asmlinkage long sys_swapon(const char __
>      1 /* header page */;
>    if (error)
>     goto bad_swap;
> +
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> +  p->swap_cgroup = vmalloc(maxpages * sizeof(*swap_cgroup));
> +  if (!(p->swap_cgroup)) {
> +   error = -ENOMEM;
> +   goto bad_swap;
> +  }
> +  memset(p->swap_cgroup, 0, maxpages * sizeof(*swap_cgroup));
> +#endif
>   }
>
>   if (nr_good_pages) {
> @@ -1704,13 +1744,22 @@ bad_swap:
>  bad_swap_2:
>   spin_lock(&swap_lock);
>   swap_map = p->swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + swap_cgroup = p->swap_cgroup;
> +#endif
>   p->swap_file = NULL;
>   p->swap_map = NULL;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + p->swap_cgroup = NULL;
> +#endif
>   p->flags = 0;
>   if (!(swap_flags & SWAP_FLAG_PREFER))
>    ++least_priority;
>   spin_unlock(&swap_lock);
>   vfree(swap_map);
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + vfree(swap_cgroup);
```

> +#endif
> if (swap_file)
> filp_close(swap_file, NULL);
> out:
>
>
>

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Daisuke Nishimura on Wed, 05 Mar 2008 08:51:22 GMT
View Forum Message <> Reply to Message

Hi.

>> @@ -664,6 +665,10 @@ retry:
>> pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
>> if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
>> pc->flags |= PAGE_CGROUP_FLAG_CACHE;
>> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
>> + atomic_inc(&mm->mm_count);
>> + pc->pc_mm = mm;
>> +#endif
>
> What kernel is this patch for? I cannot find this code in 2.6.25-rc3-mm1
>
For linux-2.6.24-mm1.

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Hugh Dickins on Wed, 05 Mar 2008 14:07:07 GMT
View Forum Message <> Reply to Message

On Wed, 5 Mar 2008, Pavel Emelyanov wrote:

> Daisuke Nishimura wrote:
> >
> > Todo:
> > - rebase new kernel, and split into some patches.
> > - Merge with memory subsystem (if it would be better), or
> > remove dependency on CONFIG_CGROUP_MEM_CONT if possible
> > (needs to make page_cgroup more generic one).
>
> Merge is a must IMHO. I can hardly imagine a situation in which
> someone would need these two separately.

Strongly agree. Nobody's interested in swap as such: it's just
secondary memory, where RAM is primary memory. People want to
control memory as the sum of the two; and I expect they may also
want to control primary memory (all that the current memcg does)
within that. I wonder if such nesting of limits fits easily
into cgroups or will be problematic.

Hugh

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Pavel Emelianov on Wed, 05 Mar 2008 14:14:12 GMT
View Forum Message <> Reply to Message

Hugh Dickins wrote:
> On Wed, 5 Mar 2008, Pavel Emelyanov wrote:
>> Daisuke Nishimura wrote:
>>> Todo:
>>> - rebase new kernel, and split into some patches.
>>> - Merge with memory subsystem (if it would be better), or
>>> remove dependency on CONFIG_CGROUP_MEM_CONT if possible
>>> (needs to make page_cgroup more generic one).
>> Merge is a must IMHO. I can hardly imagine a situation in which
>> someone would need these two separately.
>
> Strongly agree. Nobody's interested in swap as such: it's just
> secondary memory, where RAM is primary memory. People want to
> control memory as the sum of the two; and I expect they may also
> want to control primary memory (all that the current memcg does)
> within that. I wonder if such nesting of limits fits easily
> into cgroups or will be problematic.

This nesting would affect the res_couter abstraction, not the

cgroup infrastructure. Current design of resource counters doesn't
allow for such thing, but the extension is a couple-of-lines patch :)

> Hugh
>

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by KAMEZAWA Hiroyuki on Thu, 06 Mar 2008 00:33:15 GMT
View Forum Message <> Reply to Message

On Wed, 05 Mar 2008 17:14:12 +0300
Pavel Emelyanov <xemul@openvz.org> wrote:
> > Strongly agree. Nobody's interested in swap as such: it's just
> > secondary memory, where RAM is primary memory. People want to
> > control memory as the sum of the two; and I expect they may also
> > want to control primary memory (all that the current memcg does)
> > within that. I wonder if such nesting of limits fits easily
> > into cgroups or will be problematic.
>
> This nesting would affect the res_couter abstraction, not the
> cgroup infrastructure. Current design of resource counters doesn't
> allow for such thing, but the extension is a couple-of-lines patch :)
>
IMHO, keeping res_counter simple is better.

Is this kind of new entry in mem_cgroup not good ?
==
struct mem_cgroup {
 ...
 struct res_counter memory_limit.
 struct res_counter swap_limit.
 ..
}

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem

Posted by Paul Menage on Thu, 06 Mar 2008 00:35:43 GMT
View Forum Message <> Reply to Message

On Wed, Mar 5, 2008 at 4:33 PM, KAMEZAWA Hiroyuki
<kamezawa.hiroyu@jp.fujitsu.com> wrote:
> Is this kind of new entry in mem_cgroup not good ?
> ==
> struct mem_cgroup {
> ...
> struct res_counter memory_limit.
> struct res_counter swap_limit.
> ..

I agree with this - main memory and swap memory are rather different
kinds of resources, with very different performance characteristics.
It should be possible to control them completely independently (e.g.
this job gets 100M of main memory, and doesn't swap at all).

Paul

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

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Pavel Emelianov on Thu, 06 Mar 2008 08:20:17 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Wed, 05 Mar 2008 17:14:12 +0300
> Pavel Emelyanov <xemul@openvz.org> wrote:
>>> Strongly agree. Nobody's interested in swap as such: it's just
>>> secondary memory, where RAM is primary memory. People want to
>>> control memory as the sum of the two; and I expect they may also
>>> want to control primary memory (all that the current memcg does)
>>> within that. I wonder if such nesting of limits fits easily
>>> into cgroups or will be problematic.
>> This nesting would affect the res_couter abstraction, not the
>> cgroup infrastructure. Current design of resource counters doesn't
>> allow for such thing, but the extension is a couple-of-lines patch :)
>>
> IMHO, keeping res_counter simple is better.
>
> Is this kind of new entry in mem_cgroup not good ?
> ==
> struct mem_cgroup {
> ...

> struct res_counter memory_limit.
> struct res_counter swap_limit.
> ..
> }

I meant the same thing actually. By "nesting would affect" I meant, that we might want to make res_counters hierarchical.

That would kill two birds with one stone - we will make a true hierarchical memory accounting and let charging of two counters with one call.

>

_____

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