Subject: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:13:41 GMT
View Forum Message <> Reply to Message

Hi.

I updated my swapcgroup patch.

Major changes from previous version(*1):
- Rebased on 2.6.26-rc2-mm1 + KAMEZAWA-san's performance
  improvement patchset v4.
- Implemented as a add-on to memory cgroup.
  So, there is no need to add a new member to page_cgroup now.
- (NEW)Modified vm_swap_full() to calculate the rate of
  swap usage per cgroup.

Patchs:
- [1/4] add cgroup files
- [2/4] add member to swap_info_struct for cgroup
- [3/4] implement charge/uncharge
- [4/4] modify vm_swap_full for cgroup

ToDo:
- handle force_empty.
- make it possible for users to select if they use
  this feature or not, and avoid overhead for users
  not using this feature.
- move charges along with task move between cgroups.

*1
https://lists.linux-foundation.org/pipermail/containers/2008-March/010216.html

Thanks,
Daisuke Nishimura.

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

Subject: [PATCH 1/4] swapcgroup: add cgroup files
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:17:26 GMT
View Forum Message <> Reply to Message

This patch add cgroup files(and a member to struct mem_cgroup)
for swapcgroup.

The files to be added are:

- memory.swap_usage_in_bytes
- memory.swap_max_usage_in_bytes
- memory.swap_limit_in_bytes
- memory.swap_failcnt

The meaning of those files are the same as memory cgroup.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

---
 init/Kconfig | 7 +++++
 mm/memcontrol.c | 67 ++
 2 files changed, 74 insertions(+), 0 deletions(-)

```
diff --git a/init/Kconfig b/init/Kconfig
index 6135d07..aabbb83 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -407,6 +407,13 @@ config CGROUP_MEM_RES_CTLR
    This config option also selects MM_OWNER config option, which
    could in turn add some fork/exit overhead.

+config CGROUP_SWAP_RES_CTLR
+ bool "Swap Resource Controller for Control Groups"
+ depends on CGROUP_MEM_RES_CTLR && SWAP
+ help
+   Provides a swap resource controller that manages and limits swap usage.
+   Implemented as a add-on to Memory Resource Controller.
+
 config SYSFS_DEPRECATED
  bool

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index a96577f..a837215 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -141,6 +141,12 @@ struct mem_cgroup {
   * statistics.
   */
  struct mem_cgroup_stat stat;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ /*
+  * the counter to account for swap usage
+  */
+ struct res_counter swap_res;
```

```
+#endif
 };
 static struct mem_cgroup init_mem_cgroup;

@@ -960,6 +966,39 @@ static int mem_control_stat_show(struct cgroup *cont, struct cftype *cft,
  return 0;
 }

+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+static u64 swap_cgroup_read(struct cgroup *cont, struct cftype *cft)
+{
+ return res_counter_read_u64(&mem_cgroup_from_cont(cont)->swap_res,
+        cft->private);
+}
+
+static ssize_t swap_cgroup_write(struct cgroup *cont, struct cftype *cft,
+    struct file *file, const char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&mem_cgroup_from_cont(cont)->swap_res,
+    cft->private, userbuf, nbytes, ppos,
+    mem_cgroup_write_strategy);
+}
+
+static int swap_cgroup_reset(struct cgroup *cont, unsigned int event)
+{
+ struct mem_cgroup *mem;
+
+ mem = mem_cgroup_from_cont(cont);
+ switch (event) {
+ case RES_MAX_USAGE:
+  res_counter_reset_max(&mem->swap_res);
+  break;
+ case RES_FAILCNT:
+  res_counter_reset_failcnt(&mem->swap_res);
+  break;
+ }
+ return 0;
+}
+#endif
+
 static struct cftype mem_cgroup_files[] = {
  {
   .name = "usage_in_bytes",
@@ -992,6 +1031,31 @@ static struct cftype mem_cgroup_files[] = {
   .name = "stat",
   .read_map = mem_control_stat_show,
  },
```

```
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ {
+  .name = "swap_usage_in_bytes",
+  .private = RES_USAGE,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_max_usage_in_bytes",
+  .private = RES_MAX_USAGE,
+  .trigger = swap_cgroup_reset,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_limit_in_bytes",
+  .private = RES_LIMIT,
+  .write = swap_cgroup_write,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_failcnt",
+  .private = RES_FAILCNT,
+  .trigger = swap_cgroup_reset,
+  .read_u64 = swap_cgroup_read,
+ },
+#endif
 };

 static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
@@ -1069,6 +1133,9 @@ mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
  }

  res_counter_init(&mem->res);
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ res_counter_init(&mem->swap_res);
+#endif

  for_each_node_state(node, N_POSSIBLE)
   if (alloc_mem_cgroup_per_zone_info(mem, node))
```

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

Subject: [PATCH 2/4] swapcgroup: add member to swap_info_struct for cgroup

This patch add a member to swap_info_struct for cgroup.

This member, array of pointers to mem_cgroup, is used to
remember to which cgroup each swap entries are charged.

The memory for this array of pointers is allocated on swapon,
and freed on swapoff.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

---
 include/linux/swap.h | 3 +++
 mm/swapfile.c | 32 ++++++++++++++++++++++++++++++++
 2 files changed, 35 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/swap.h b/include/linux/swap.h
index de40f16..67de27b 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -141,6 +141,9 @@ struct swap_info_struct {
  struct swap_extent *curr_swap_extent;
  unsigned old_block_size;
  unsigned short * swap_map;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ struct mem_cgroup **memcg;
+#endif
  unsigned int lowest_bit;
  unsigned int highest_bit;
  unsigned int cluster_next;
diff --git a/mm/swapfile.c b/mm/swapfile.c
index d3caf3a..232bf20 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -1207,6 +1207,9 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
 {
  struct swap_info_struct * p = NULL;
  unsigned short *swap_map;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ struct mem_cgroup **memcg;
+#endif
  struct file *swap_file, *victim;
  struct address_space *mapping;
  struct inode *inode;
@@ -1309,10 +1312,17 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
  p->max = 0;
```

```
  swap_map = p->swap_map;
  p->swap_map = NULL;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ memcg = p->memcg;
+ p->memcg = NULL;
+#endif
  p->flags = 0;
  spin_unlock(&swap_lock);
  mutex_unlock(&swapon_mutex);
  vfree(swap_map);
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ vfree(memcg);
+#endif
  inode = mapping->host;
  if (S_ISBLK(inode->i_mode)) {
   struct block_device *bdev = I_BDEV(inode);
@@ -1456,6 +1466,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
  unsigned long maxpages = 1;
  int swapfilesize;
  unsigned short *swap_map;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ struct mem_cgroup **memcg;
+#endif
  struct page *page = NULL;
  struct inode *inode = NULL;
  int did_down = 0;
@@ -1479,6 +1492,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
  p->swap_file = NULL;
  p->old_block_size = 0;
  p->swap_map = NULL;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ p->memcg = NULL;
+#endif
  p->lowest_bit = 0;
  p->highest_bit = 0;
  p->cluster_nr = 0;
@@ -1651,6 +1667,15 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
     1 /* header page */;
   if (error)
    goto bad_swap;
+
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+  p->memcg = vmalloc(maxpages * sizeof(struct mem_cgroup *));
+  if (!p->memcg) {
+   error = -ENOMEM;
```

```
+   goto bad_swap;
+  }
+  memset(p->memcg, 0, maxpages * sizeof(struct mem_cgroup *));
+#endif
  }

  if (nr_good_pages) {
@@ -1710,11 +1735,18 @@ bad_swap_2:
  swap_map = p->swap_map;
  p->swap_file = NULL;
  p->swap_map = NULL;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ memcg = p->memcg;
+ p->memcg = NULL;
+#endif
  p->flags = 0;
  if (!(swap_flags & SWAP_FLAG_PREFER))
   ++least_priority;
  spin_unlock(&swap_lock);
  vfree(swap_map);
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ vfree(memcg);
+#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: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:20:05 GMT
View Forum Message <> Reply to Message

This patch implements charge and uncharge of swapcgroup.

- what will be charged ?
  charge the number of swap entries in 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 mem_cgroup the swap entry should be charged,

I changed the argument of get_swap_entry() from (void) to
(struct page *). As a result, get_swap_entry() can determine
to what mem_cgroup it should charge the swap entry
by referring to page->page_cgroup->mem_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.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

---
include/linux/memcontrol.h | 21 ++++++++++++++++++++
include/linux/swap.h | 4 +-
mm/memcontrol.c | 47 ++
mm/shmem.c | 2 +-
mm/swap_state.c | 2 +-
mm/swapfile.c | 14 ++++++++++++-
6 files changed, 85 insertions(+), 5 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index fdf3967..a7e6621 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -24,6 +24,7 @@ struct mem_cgroup;
struct page_cgroup;
struct page;
struct mm_struct;
+struct swap_info_struct;

#ifdef CONFIG_CGROUP_MEM_RES_CTLR

@@ -172,5 +173,25 @@ static inline long mem_cgroup_calc_reclaim_inactive(struct
mem_cgroup *mem,
}
#endif /* CONFIG_CGROUP_MEM_CONT */

+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+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_RES_CTLR */
+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 /* CONFIG_CGROUP_SWAP_RES_CTLR */
+
 #endif /* _LINUX_MEMCONTROL_H */

diff --git a/include/linux/swap.h b/include/linux/swap.h
index 67de27b..18887f0 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -241,7 +241,7 @@ extern struct page *swapin_readahead(swp_entry_t, gfp_t,
 /* linux/mm/swapfile.c */
 extern long total_swap_pages;
 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 +342,7 @@ static inline int remove_exclusive_swap_page(struct page *p)
  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 --git a/mm/memcontrol.c b/mm/memcontrol.c
index a837215..84e803d 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1220,3 +1220,50 @@ struct cgroup_subsys mem_cgroup_subsys = {
  .attach = mem_cgroup_move_task,
  .early_init = 0,
 };
+
+#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);
+
+ css_get(&mem->css);
+ 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;
```

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

---

Subject: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:22:24 GMT
View Forum Message <> Reply to Message

This patch modifies vm_swap_full() to calculate
the rate of swap usage per cgroup.

The purpose of this change is to free freeable swap caches
(that is, swap entries) per cgroup, so that swap_cgroup_charge()
fails less frequently.

I think I can free freeable swap caches more agressively
with one of Rik's splitlru patchset:

  [patch 04/14] free swap space on swap-in/activation

but, I should verify whether this change to vm_swap_full()
is valid.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

```
 include/linux/memcontrol.h |    3 +++
 include/linux/swap.h       |    6 +++++-
 mm/memcontrol.c            |   10 ++++++++++
 mm/memory.c                |    2 +-
```

```
 mm/swapfile.c            |   35 +++++++++++++++++++++++++++++++++++-
 5 files changed, 53 insertions(+), 3 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index a7e6621..256b298 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -179,6 +179,9 @@ extern int swap_cgroup_charge(struct page *page,
    unsigned long offset);
 extern void swap_cgroup_uncharge(struct swap_info_struct *si,
     unsigned long offset);
+extern int swap_cgroup_vm_swap_full(struct page *page);
+extern u64 swap_cgroup_read_usage(struct mem_cgroup *mem);
+extern u64 swap_cgroup_read_limit(struct mem_cgroup *mem);
 #else /* CONFIG_CGROUP_SWAP_RES_CTLR */
 static inline int swap_cgroup_charge(struct page *page,
    struct swap_info_struct *si,
diff --git a/include/linux/swap.h b/include/linux/swap.h
index 18887f0..ef156c9 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -159,8 +159,12 @@ struct swap_list_t {
  int next; /* swapfile to be used next */
 };

+#ifndef CONFIG_CGROUP_SWAP_RES_CTLR
 /* Swap 50% full? Release swapcache more aggressively.. */
-#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
+#define vm_swap_full(page) (nr_swap_pages*2 < total_swap_pages)
+#else
+#define vm_swap_full(page) swap_cgroup_vm_swap_full(page)
+#endif

 /* linux/mm/page_alloc.c */
 extern unsigned long totalram_pages;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 84e803d..58d72ca 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1265,5 +1265,15 @@ void swap_cgroup_uncharge(struct swap_info_struct *si,
   css_put(&mem->css);
  }
 }
+
+u64 swap_cgroup_read_usage(struct mem_cgroup *mem)
+{
+ return res_counter_read_u64(&mem->swap_res, RES_USAGE);
+}
```

```
+
+u64 swap_cgroup_read_limit(struct mem_cgroup *mem)
+{
+ return res_counter_read_u64(&mem->swap_res, RES_LIMIT);
+}
 #endif

diff --git a/mm/memory.c b/mm/memory.c
index df8f0e9..be2ff96 100644
--- a/mm/memory.c
+++ b/mm/memory.c
@@ -2175,7 +2175,7 @@ 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())
+ if (vm_swap_full(page))
   remove_exclusive_swap_page(page);
  unlock_page(page);

diff --git a/mm/swapfile.c b/mm/swapfile.c
index 682b71e..9256c2d 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -429,7 +429,7 @@ 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))) {
    delete_from_swap_cache(page);
    SetPageDirty(page);
   }
@@ -1892,3 +1892,36 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
  *offset = ++toff;
  return nr_pages? ++nr_pages: 0;
 }
+
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+int swap_cgroup_vm_swap_full(struct page *page)
+{
+ int ret;
+ struct swap_info_struct *p;
+ struct mem_cgroup *mem;
+ u64 usage;
+ u64 limit;
+ swp_entry_t entry;
```

```
+
+ VM_BUG_ON(!PageLocked(page));
+ VM_BUG_ON(!PageSwapCache(page));
+
+ ret = 0;
+ entry.val = page_private(page);
+ p = swap_info_get(entry);
+ if (!p)
+  goto out;
+
+ mem = p->memcg[swp_offset(entry)];
+ usage = swap_cgroup_read_usage(mem) / PAGE_SIZE;
+ limit = swap_cgroup_read_limit(mem) / PAGE_SIZE;
+ limit = (limit < total_swap_pages) ? limit : total_swap_pages;
+
+ ret = usage * 2 > limit;
+
+ spin_unlock(&swap_lock);
+
+out:
+ return ret;
+}
+#endif
```

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

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by yamamoto on Thu, 22 May 2008 06:45:07 GMT
View Forum Message <> Reply to Message

```
> @@ -1892,3 +1892,36 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
>   *offset = ++toff;
>   return nr_pages? ++nr_pages: 0;
>  }
> +
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> +int swap_cgroup_vm_swap_full(struct page *page)
> +{
> + int ret;
> + struct swap_info_struct *p;
> + struct mem_cgroup *mem;
> + u64 usage;
> + u64 limit;
```

```
> + swp_entry_t entry;
> +
> + VM_BUG_ON(!PageLocked(page));
> + VM_BUG_ON(!PageSwapCache(page));
> +
> + ret = 0;
> + entry.val = page_private(page);
> + p = swap_info_get(entry);
> + if (!p)
> +  goto out;
> +
> + mem = p->memcg[swp_offset(entry)];
> + usage = swap_cgroup_read_usage(mem) / PAGE_SIZE;
> + limit = swap_cgroup_read_limit(mem) / PAGE_SIZE;
> + limit = (limit < total_swap_pages) ? limit : total_swap_pages;
> +
> + ret = usage * 2 > limit;
> +
> + spin_unlock(&swap_lock);
> +
> +out:
> + return ret;
> +}
> +#endif
```

shouldn't it check the global usage (nr_swap_pages) as well?

YAMAMOTO Takashi

_____

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

---

Subject: Re: [PATCH 2/4] swapcgroup: add member to swap_info_struct for cgroup
Posted by KAMEZAWA Hiroyuki on Thu, 22 May 2008 07:23:12 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 15:18:51 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> This patch add a member to swap_info_struct for cgroup.
>
> This member, array of pointers to mem_cgroup, is used to
> remember to which cgroup each swap entries are charged.
>
> The memory for this array of pointers is allocated on swapon,
> and freed on swapoff.

>
Hi, in general, #ifdefs in the middle of functions are not good style.
I'd like to comment some hints.

>
> Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>
>
> ---
> include/linux/swap.h | 3 +++
> mm/swapfile.c | 32 ++++++++++++++++++++++++++++++++
> 2 files changed, 35 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/swap.h b/include/linux/swap.h
> index de40f16..67de27b 100644
> --- a/include/linux/swap.h
> +++ b/include/linux/swap.h
> @@ -141,6 +141,9 @@ struct swap_info_struct {
>   struct swap_extent *curr_swap_extent;
>   unsigned old_block_size;
>   unsigned short * swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + struct mem_cgroup **memcg;
> +#endif
>   unsigned int lowest_bit;
>   unsigned int highest_bit;
>   unsigned int cluster_next;
> diff --git a/mm/swapfile.c b/mm/swapfile.c
> index d3caf3a..232bf20 100644
> --- a/mm/swapfile.c
> +++ b/mm/swapfile.c
> @@ -1207,6 +1207,9 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
> {
>   struct swap_info_struct * p = NULL;
>   unsigned short *swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + struct mem_cgroup **memcg;
> +#endif
Remove #ifdef.

 struct mem_cgroup **memcg = NULL;

>   struct file *swap_file, *victim;
>   struct address_space *mapping;
>   struct inode *inode;
> @@ -1309,10 +1312,17 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
>   p->max = 0;
>   swap_map = p->swap_map;
>   p->swap_map = NULL;

> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + memcg = p->memcg;
> + p->memcg = NULL;
> +#endif

==
#ifdef CONFIG_CGROUP_SWAP_RES_CTR
  void swap_cgroup_init_memcg(p, memcg)
  {
    do something.
  }
#else
   void swap_cgroup_init_memcg(p, memcg)
  {
  }
#endif
==

>   p->flags = 0;
>   spin_unlock(&swap_lock);
>   mutex_unlock(&swapon_mutex);
>   vfree(swap_map);
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + vfree(memcg);
> +#endif
 if (memcg)
      vfree(memcg);

>   inode = mapping->host;
>   if (S_ISBLK(inode->i_mode)) {
>   struct block_device *bdev = I_BDEV(inode);
> @@ -1456,6 +1466,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
>   unsigned long maxpages = 1;
>   int swapfilesize;
>   unsigned short *swap_map;
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + struct mem_cgroup **memcg;
> +#endif
Remove #ifdefs

>   struct page *page = NULL;
>   struct inode *inode = NULL;
>   int did_down = 0;
> @@ -1479,6 +1492,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)

>   p->swap_file = NULL;
>   p->old_block_size = 0;
>   p->swap_map = NULL;
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + p->memcg = NULL;
> +#endif

void init_swap_ctlr_memcg(p);

>   p->lowest_bit = 0;
>   p->highest_bit = 0;
>   p->cluster_nr = 0;
> @@ -1651,6 +1667,15 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
>   1 /* header page */;
>   if (error)
>   goto bad_swap;
> +
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + p->memcg = vmalloc(maxpages * sizeof(struct mem_cgroup *));
> + if (!p->memcg) {
> +   error = -ENOMEM;
> +   goto bad_swap;
> + }
> + memset(p->memcg, 0, maxpages * sizeof(struct mem_cgroup *));
> +#endif
void alloc_swap_ctlr_memcg(p)

But this implies swapon will fail at memory shortage. Is it good ?

>   }
>
>   if (nr_good_pages) {
> @@ -1710,11 +1735,18 @@ bad_swap_2:
>   swap_map = p->swap_map;
>   p->swap_file = NULL;
>   p->swap_map = NULL;
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + memcg = p->memcg;
> + p->memcg = NULL;
> +#endif
>   p->flags = 0;
>   if (!(swap_flags & SWAP_FLAG_PREFER))
>   ++least_priority;
>   spin_unlock(&swap_lock);
>   vfree(swap_map);
> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> + vfree(memcg);

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

Thanks,
-Kame

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

---

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: email@kvack.org
>

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by KAMEZAWA Hiroyuki on Thu, 22 May 2008 07:36:09 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 15:22:24 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> + mem = p->memcg[swp_offset(entry)];
> + usage = swap_cgroup_read_usage(mem) / PAGE_SIZE;
> + limit = swap_cgroup_read_limit(mem) / PAGE_SIZE;
> + limit = (limit < total_swap_pages) ? limit : total_swap_pages;

mem can be NULL here.

Thanks,
-Kame

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by KAMEZAWA Hiroyuki on Thu, 22 May 2008 07:44:21 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 15:13:41 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> Hi.
>
> I updated my swapcgroup patch.
>
seems good in general.

> Major changes from previous version(*1):
> - Rebased on 2.6.26-rc2-mm1 + KAMEZAWA-san's performance
>   improvement patchset v4.
> - Implemented as a add-on to memory cgroup.
>   So, there is no need to add a new member to page_cgroup now.
> - (NEW)Modified vm_swap_full() to calculate the rate of
>   swap usage per cgroup.
>
> Patchs:
> - [1/4] add cgroup files
> - [2/4] add member to swap_info_struct for cgroup
> - [3/4] implement charge/uncharge
> - [4/4] modify vm_swap_full for cgroup
>
> ToDo:
> - handle force_empty.

Without this, we can do rmdir() against cgroup with swap. right ?

> - make it possible for users to select if they use
>   this feature or not, and avoid overhead for users
>   not using this feature.
> - move charges along with task move between cgroups.
>
I think memory-controller's anon pages should also do this....
But how do you think about shared entries ?

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by KOSAKI Motohiro on Thu, 22 May 2008 08:00:14 GMT
View Forum Message <> Reply to Message

Hi,

> +#ifndef CONFIG_CGROUP_SWAP_RES_CTLR
> /* Swap 50% full? Release swapcache more aggressively.. */
> -#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
> +#define vm_swap_full(page) (nr_swap_pages*2 < total_swap_pages)
> +#else
> +#define vm_swap_full(page) swap_cgroup_vm_swap_full(page)
> +#endif

I'd prefer #ifdef rather than #ifndef.

so...

#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
  your definition
#else
  original definition
#endif

and vm_swap_full() isn't page granularity operation.
this is memory(or swap) cgroup operation.

this argument is slightly odd.

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

Subject: Re: [PATCH 2/4] swapcgroup: add member to swap_info_struct for cgroup
Posted by Daisuke Nishimura on Thu, 22 May 2008 08:46:54 GMT
View Forum Message <> Reply to Message

Hi.

On 2008/05/22 16:23 +0900, KAMEZAWA Hiroyuki wrote:
> On Thu, 22 May 2008 15:18:51 +0900
> Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:
>
>> This patch add a member to swap_info_struct for cgroup.
>>
>> This member, array of pointers to mem_cgroup, is used to
>> remember to which cgroup each swap entries are charged.
>>
>> The memory for this array of pointers is allocated on swapon,
>> and freed on swapoff.
>>
> Hi, in general, #ifdefs in the middle of functions are not good style.
> I'd like to comment some hints.
>
I completely agree that it's not good style.

>> Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>
>>
>> ---
>> include/linux/swap.h | 3 +++
>> mm/swapfile.c | 32 ++++++++++++++++++++++++++++++++
>> 2 files changed, 35 insertions(+), 0 deletions(-)
>>
>> diff --git a/include/linux/swap.h b/include/linux/swap.h
>> index de40f16..67de27b 100644
>> --- a/include/linux/swap.h
>> +++ b/include/linux/swap.h
>> @@ -141,6 +141,9 @@ struct swap_info_struct {
>>   struct swap_extent *curr_swap_extent;
>>   unsigned old_block_size;
>>   unsigned short * swap_map;
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + struct mem_cgroup **memcg;
>> +#endif
>>   unsigned int lowest_bit;
>>   unsigned int highest_bit;
>>   unsigned int cluster_next;
>> diff --git a/mm/swapfile.c b/mm/swapfile.c
>> index d3caf3a..232bf20 100644
>> --- a/mm/swapfile.c
>> +++ b/mm/swapfile.c
>> @@ -1207,6 +1207,9 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
>> {
>>   struct swap_info_struct * p = NULL;
>>   unsigned short *swap_map;
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + struct mem_cgroup **memcg;

>> +#endif
> Remove #ifdef.
>
> struct mem_cgroup **memcg = NULL;
>
good idea.
I'll do it.

>>   struct file *swap_file, *victim;
>>   struct address_space *mapping;
>>   struct inode *inode;
>> @@ -1309,10 +1312,17 @@ asmlinkage long sys_swapoff(const char __user * specialfile)
>>   p->max = 0;
>>   swap_map = p->swap_map;
>>   p->swap_map = NULL;
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + memcg = p->memcg;
>> + p->memcg = NULL;
>> +#endif
>
>
> ==
> #ifdef CONFIG_CGROUP_SWAP_RES_CTR
>   void swap_cgroup_init_memcg(p, memcg)
>   {
>   do something.
>   }
> #else
>   void swap_cgroup_init_memcg(p, memcg)
>   {
>   }
> #endif
> ==
>
I think swap_cgroup_init_memcg should return old value
of p->memcg, and I would like to name it swap_cgroup_clear_memcg,
because it is called by sys_swapoff, so "clear" rather than "init"
would be better.

How about something like this?

struct mem_cgroup **swap_cgroup_clear_memcg(p, memcg)
{
 struct mem_cgroup **mem;

 mem = p->memcg;
 p->memcg = NULL;

```
 return mem;
}
```

and at sys_swapoff():

```
struct mem_cgroup **memcg;
 :
memcg = swap_cgroup_clear_memcg(p, memcg);
 :
if (memcg)
 vfree(memcg);

>>   p->flags = 0;
>>   spin_unlock(&swap_lock);
>>   mutex_unlock(&swapon_mutex);
>>   vfree(swap_map);
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + vfree(memcg);
>> +#endif
>  if (memcg)
>       vfree(memcg);
>
>
```

will do.

```
>>   inode = mapping->host;
>>   if (S_ISBLK(inode->i_mode)) {
>>    struct block_device *bdev = I_BDEV(inode);
>> @@ -1456,6 +1466,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
>>   unsigned long maxpages = 1;
>>   int swapfilesize;
>>   unsigned short *swap_map;
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + struct mem_cgroup **memcg;
>> +#endif
> Remove #ifdefs
>
```

will do.

```
>>   struct page *page = NULL;
>>   struct inode *inode = NULL;
>>   int did_down = 0;
>> @@ -1479,6 +1492,9 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
>>   p->swap_file = NULL;
>>   p->old_block_size = 0;
>>   p->swap_map = NULL;
```

>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + p->memcg = NULL;
>> +#endif
>
> void init_swap_ctlr_memcg(p);
>
I would like to call this one swap_cgroup_init_memcg.

>>   p->lowest_bit = 0;
>>   p->highest_bit = 0;
>>   p->cluster_nr = 0;
>> @@ -1651,6 +1667,15 @@ asmlinkage long sys_swapon(const char __user * specialfile, int
swap_flags)
>>   1 /* header page */;
>>   if (error)
>>   goto bad_swap;
>> +
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + p->memcg = vmalloc(maxpages * sizeof(struct mem_cgroup *));
>> + if (!p->memcg) {
>> +   error = -ENOMEM;
>> +   goto bad_swap;
>> + }
>> + memset(p->memcg, 0, maxpages * sizeof(struct mem_cgroup *));
>> +#endif
> void alloc_swap_ctlr_memcg(p)
>
OK.
I'll implement swap_cgroup_alloc_memcg.

> But this implies swapon will fail at memory shortage. Is it good ?
>
Hum.
Would it be better to just disabling this feature?

>>   }
>>
>>   if (nr_good_pages) {
>> @@ -1710,11 +1735,18 @@ bad_swap_2:
>>   swap_map = p->swap_map;
>>   p->swap_file = NULL;
>>   p->swap_map = NULL;
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + memcg = p->memcg;
>> + p->memcg = NULL;
>> +#endif
>>   p->flags = 0;
>>   if (!(swap_flags & SWAP_FLAG_PREFER))

>>   ++least_priority;
>>   spin_unlock(&swap_lock);
>>   vfree(swap_map);
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> + vfree(memcg);
>> +#endif
>>   if (swap_file)
>>   filp_close(swap_file, NULL);
>> out:
>>
I'll handle these 2 #ifdefs as sys_swapoff.

>>
>
> Thanks,
> -Kame
>

Thank you for your advice!

Daisuke Nishimura.

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

---

Subject: Re: [PATCH 2/4] swapcgroup: add member to swap_info_struct for cgroup
Posted by KAMEZAWA Hiroyuki on Thu, 22 May 2008 09:33:35 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 17:46:54 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> > ==
> > #ifdef CONFIG_CGROUP_SWAP_RES_CTR
> >   void swap_cgroup_init_memcg(p, memcg)
> >   {
> >   do something.
> >   }
> > #else
> >   void swap_cgroup_init_memcg(p, memcg)
> >   {
> >   }
> > #endif
> > ==
> >

> I think swap_cgroup_init_memcg should return old value
> of p->memcg, and I would like to name it swap_cgroup_clear_memcg,
> because it is called by sys_swapoff, so "clear" rather than "init"
> would be better.
>
> How about something like this?
>
> struct mem_cgroup **swap_cgroup_clear_memcg(p, memcg)
> {
> struct mem_cgroup **mem;
>
> mem = p->memcg;
> p->memcg = NULL;
>
> return mem;
> }
>
> and at sys_swapoff():
>
> struct mem_cgroup **memcg;
> :
> memcg = swap_cgroup_clear_memcg(p, memcg);
> :
> if (memcg)
> vfree(memcg);
>
seems good.

> >> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
> >> + p->memcg = vmalloc(maxpages * sizeof(struct mem_cgroup *));
> >> + if (!p->memcg) {
> >> +   error = -ENOMEM;
> >> +   goto bad_swap;
> >> + }
> >> + memset(p->memcg, 0, maxpages * sizeof(struct mem_cgroup *));
> >> +#endif
> > void alloc_swap_ctlr_memcg(p)
> >
> OK.
> I'll implement swap_cgroup_alloc_memcg.
>
> > But this implies swapon will fail at memory shortage. Is it good ?
> >
> Hum.
> Would it be better to just disabling this feature?
>
I have no good idea. IMHO, adding printk() to show 'fatal status of

not-enough-memory-for-vmalloc' will be first step.

I believe vmalloc() tend not to fail on 64bit machine, but on i386,
vmalloc area is not enough.

Thanks,
-Kame

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by Daisuke Nishimura on Thu, 22 May 2008 12:22:05 GMT
View Forum Message <> Reply to Message

Hi,

On 2008/05/22 17:00 +0900, KOSAKI Motohiro wrote:
> Hi,
>
>> +#ifndef CONFIG_CGROUP_SWAP_RES_CTLR
>> /* Swap 50% full? Release swapcache more aggressively.. */
>> -#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
>> +#define vm_swap_full(page) (nr_swap_pages*2 < total_swap_pages)
>> +#else
>> +#define vm_swap_full(page) swap_cgroup_vm_swap_full(page)
>> +#endif
>
> I'd prefer #ifdef rather than #ifndef.
>
> so...
>
> #ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>   your definition
> #else
>   original definition
> #endif
>
OK.
I'll change it.

> and vm_swap_full() isn't page granularity operation.
> this is memory(or swap) cgroup operation.
>
> this argument is slightly odd.

>
But what callers of vm_swap_full() know is page,
not mem_cgroup.
I don't want to add to callers something like:

```
pc = get_page_cgroup(page);
mem = pc->mem_cgroup;
vm_swap_full(mem);
```

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by KOSAKI Motohiro on Thu, 22 May 2008 12:32:36 GMT
View Forum Message <> Reply to Message

>> I'd prefer #ifdef rather than #ifndef.
>>
>> so...
>>
>> #ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>>   your definition
>> #else
>>   original definition
>> #endif
>>
> OK.
> I'll change it.

Thanks.

>> and vm_swap_full() isn't page granularity operation.
>> this is memory(or swap) cgroup operation.
>>
>> this argument is slightly odd.
>>
> But what callers of vm_swap_full() know is page,
> not mem_cgroup.
> I don't want to add to callers something like:
>

> pc = get_page_cgroup(page);
> mem = pc->mem_cgroup;
> vm_swap_full(mem);

perhaps, I don't understand your intention exactly.
Why can't you make wrapper function?

e.g.
vm_swap_full(page_to_memcg(page))

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by Daisuke Nishimura on Thu, 22 May 2008 12:34:50 GMT
View Forum Message <> Reply to Message

Hi,

On 2008/05/22 15:45 +0900, YAMAMOTO Takashi wrote:
>> @@ -1892,3 +1892,36 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
>> *offset = ++toff;
>> return nr_pages? ++nr_pages: 0;
>> }
>> +
>> +#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
>> +int swap_cgroup_vm_swap_full(struct page *page)
>> +{
>> + int ret;
>> + struct swap_info_struct *p;
>> + struct mem_cgroup *mem;
>> + u64 usage;
>> + u64 limit;
>> + swp_entry_t entry;
>> +
>> + VM_BUG_ON(!PageLocked(page));
>> + VM_BUG_ON(!PageSwapCache(page));
>> +
>> + ret = 0;
>> + entry.val = page_private(page);
>> + p = swap_info_get(entry);
>> + if (!p)
>> + goto out;
>> +
>> + mem = p->memcg[swp_offset(entry)];
>> + usage = swap_cgroup_read_usage(mem) / PAGE_SIZE;

>> + limit = swap_cgroup_read_limit(mem) / PAGE_SIZE;
>> + limit = (limit < total_swap_pages) ? limit : total_swap_pages;
>> +
>> + ret = usage * 2 > limit;
>> +
>> + spin_unlock(&swap_lock);
>> +
>> +out:
>> + return ret;
>> +}
>> +#endif
>
> shouldn't it check the global usage (nr_swap_pages) as well?
>
> YAMAMOTO Takashi
>

I didn't check global usage because I didn't want
some group to be influenced by other groups.

But in above code, there would be some cases that
vm_swap_full() returns false even when more than
half of swap is used in global.

Thanks you for pointing it out.

How about something like this?

```
  :
usage = swap_cgroup_read_usage(mem); //no need to align to number of page
limit = swap_cgroup_read_limit(mem); //no need to align to number of page
ret = (usage * 2 > limit) || (nr_swap_pages * 2 < total_swap_pages)
  :
```

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Thu, 22 May 2008 21:27:54 GMT
View Forum Message <> Reply to Message

Daisuke Nishimura wrote:

> Hi.
>
> I updated my swapcgroup patch.
>
> Major changes from previous version(*1):
> - Rebased on 2.6.26-rc2-mm1 + KAMEZAWA-san's performance
> improvement patchset v4.
> - Implemented as a add-on to memory cgroup.
> So, there is no need to add a new member to page_cgroup now.
> - (NEW)Modified vm_swap_full() to calculate the rate of
> swap usage per cgroup.
>
> Patchs:
> - [1/4] add cgroup files
> - [2/4] add member to swap_info_struct for cgroup
> - [3/4] implement charge/uncharge
> - [4/4] modify vm_swap_full for cgroup
>
> ToDo:
> - handle force_empty.
> - make it possible for users to select if they use
> this feature or not, and avoid overhead for users
> not using this feature.
> - move charges along with task move between cgroups.
>

Thanks for looking into this. Yamamoto-San is also looking into a swap
controller. Is there a consensus on the approach?

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Fri, 23 May 2008 02:10:29 GMT
View Forum Message <> Reply to Message

On 2008/05/22 16:44 +0900, KAMEZAWA Hiroyuki wrote:
> On Thu, 22 May 2008 15:13:41 +0900
> Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:
>

>> Hi.
>>
>> I updated my swapcgroup patch.
>>
> seems good in general.
>
>
Thanks :-)

>> ToDo:
>> - handle force_empty.
>
> Without this, we can do rmdir() against cgroup with swap. right ?
>
You are right.

There are some cases that cgroup dir cannot be removed
because there remains some swap usage
even when no tasks remain in the dir.
In such cases, the only way to remove the dir is currently
to do swapoff.

So, I think this is the most important todo.

>> - make it possible for users to select if they use
>>   this feature or not, and avoid overhead for users
>>   not using this feature.
>> - move charges along with task move between cgroups.
>>
> I think memory-controller's anon pages should also do this....

> But how do you think about shared entries ?
>
Yes.
This is a big problem. I don't have any practical idea yet,
but at least I think it should be avoided for some shared
entry to be charged to different groups.

Thanks,
Daisuke Nishimura.

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

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Rik van Riel on Fri, 23 May 2008 02:26:55 GMT
View Forum Message <> Reply to Message

On Thu, 22 May 2008 15:13:41 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> I updated my swapcgroup patch.

I do not understand why this is useful.

With the other cgroup resource controllers, once a process group reaches its limit, it is limited or punished in some way. For example, when it goes over its RSS limit, memory is taken away.

However, once a cgroup reaches its swap limit, it is rewarded, by allowing more of its pages to stay resident in RAM, instead of having them swapped out.

This, in turn, will cause the VM to evict pages from other, better behaving groups. In short, the cgroup that has "misbehaved" by reaching its limit causes other cgroups to get punished.

Even worse is that a cgroup has NO CONTROL over how much of its memory is kept in RAM and how much is swapped out. This kind of decision is made on a system-wide basis by the kernel, dependent on what other processes in the system are doing. There also is no easy way for a cgroup to reduce its swap use, unlike with other resources.

In what scenario would you use a resource controller that rewards a group for reaching its limit?

How can the cgroup swap space controller help sysadmins achieve performance or fairness goals on a system?

--
All rights reversed.

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

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Fri, 23 May 2008 02:42:50 GMT

(sorry, I sent the previous mail before completing it)

On 2008/05/23 11:10 +0900, Daisuke Nishimura wrote:
> On 2008/05/22 16:44 +0900, KAMEZAWA Hiroyuki wrote:
(snip)
>>> - make it possible for users to select if they use
>>> this feature or not, and avoid overhead for users
>>> not using this feature.
>>> - move charges along with task move between cgroups.
>>>
>> I think memory-controller's anon pages should also do this....
>
I want it too.

Not only it's usefull for users IMHO,
but alsoI need it to charge a swap to the group which the task
belongs to at the point of swapout.

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by KAMEZAWA Hiroyuki on Fri, 23 May 2008 03:08:44 GMT

On Thu, 22 May 2008 22:26:55 -0400
Rik van Riel <riel@redhat.com> wrote:

> Even worse is that a cgroup has NO CONTROL over how much
> of its memory is kept in RAM and how much is swapped out.
Could you explain "NO CONTROL" ? cgroup has LRU....
'how mucch memory should be swapped out from memory' is well controlled
in the VM besides LRU logic ?

> This kind of decision is made on a system-wide basis by
> the kernel, dependent on what other processes in the system
> are doing. There also is no easy way for a cgroup to reduce
> its swap use, unlike with other resources.
>

> In what scenario would you use a resource controller that
> rewards a group for reaching its limit?
>
> How can the cgroup swap space controller help sysadmins
> achieve performance or fairness goals on a system?
>
Perforamnce is not the first goal of this swap controller, I think.
This is for resouce isolation/overcommiting.

1. Some _crazy_ people considers swap as very-slow-memory resource ;)
   I don't think so but I know there are tons of people....

2. Resource Isolation.
   When a cgroup has memory limitation, it can create tons of swap.
   For example, limit a cgroup's memory to be 128M and malloc 3G bytes.
   2.8Gbytes of swap will be used _easily_. A process can use up all swap.
   In that case, other process can't use swap.

IIRC, a man shown his motivation to controll swap in OLS2007/BOF as following.
==
Consider following system. (and there is no swap controller.)
Memory 4G. Swap 1G. with 2 cgroups A, B.

state 1) swap is not used.
  A....memory limit to be 1G no swap usage memory_usage=0M
  B....memory limit to be 1G no swap usage memory_usage=0M

state 2) Run a big program on A.
  A....memory limit to be 1G and try to use 1.7G. uses 700MBytes of swap.
       memory_usage=1G swap_usage=700M
  B....memory_usage=0M

state 3) A some of programs ends in 'A'
  A....memory_usage=500M swap_usage=700M
  B....memory_usage=0M.

state 4) Run a big program on B.
  A...memory_usage=500M swap_usage=700M.
  B...memory_usage=1G   swap_usage=300M

Group B can only use 1.3G because of unfair swap use of group A.
But users think why A uses 700M of swap with 500M of free memory....

If we don't have limitation to swap, we'll have to innovate a way to move swap
to memory in some reasonable logic.

Thanks,
-Kame

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

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Rik van Riel on Fri, 23 May 2008 03:32:07 GMT
View Forum Message <> Reply to Message

On Fri, 23 May 2008 12:10:27 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
> On Thu, 22 May 2008 22:26:55 -0400
> Rik van Riel <riel@redhat.com> wrote:
>
> > Even worse is that a cgroup has NO CONTROL over how much
> > of its memory is kept in RAM and how much is swapped out.
> Could you explain "NO CONTROL" ? cgroup has LRU....
> 'how mucch memory should be swapped out from memory' is well controlled
> in the VM besides LRU logic ?

The kernel controls what is swapped out. The userland
processes in the cgroup can do nothing to reduce their
swap usage.

> Consider following system. (and there is no swap controller.)
> Memory 4G. Swap 1G. with 2 cgroups A, B.
>
> state 1) swap is not used.
> A....memory limit to be 1G no swap usage memory_usage=0M
> B....memory limit to be 1G no swap usage memory_usage=0M
>
> state 2) Run a big program on A.
> A....memory limit to be 1G and try to use 1.7G. uses 700MBytes of swap.
> memory_usage=1G swap_usage=700M
> B....memory_usage=0M
>
> state 3) A some of programs ends in 'A'
> A....memory_usage=500M swap_usage=700M
> B....memory_usage=0M.
>
> state 4) Run a big program on B.
> A...memory_usage=500M swap_usage=700M.
> B...memory_usage=1G swap_usage=300M
>

> Group B can only use 1.3G because of unfair swap use of group A.
> But users think why A uses 700M of swap with 500M of free memory....
>
> If we don't have limitation to swap, we'll have to innovate a way to move swap
> to memory in some reasonable logic.

OK, I see the use case.

In the above example, it would be possible for cgroup A
to have only 800MB of anonymous memory total, in addition
to 400MB of page cache. The page cache could push the
anonymous memory into swap, indirectly penalizing how much
memory cgroup B can use.

Of course, it could be argued that the system should just
be run with enough swap space, but that is another story :)

--
All rights reversed.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Fri, 23 May 2008 03:59:23 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Thu, 22 May 2008 22:26:55 -0400
> Rik van Riel <riel@redhat.com> wrote:
>
>> Even worse is that a cgroup has NO CONTROL over how much
>> of its memory is kept in RAM and how much is swapped out.

We used to have a control on the swap cache pages as well, but their
implementation needed more thought

> Could you explain "NO CONTROL" ? cgroup has LRU....
> 'how mucch memory should be swapped out from memory' is well controlled
> in the VM besides LRU logic ?
>
>> This kind of decision is made on a system-wide basis by
>> the kernel, dependent on what other processes in the system
>> are doing. There also is no easy way for a cgroup to reduce
>> its swap use, unlike with other resources.
>>

One option is to limit the virtual address space usage of the cgroup to ensure
that swap usage of a cgroup will *not* exceed the specified limit. Along with a
good swap controller, it should provide good control over the cgroup's memory usage.

>
>> In what scenario would you use a resource controller that
>> rewards a group for reaching its limit?
>>
>> How can the cgroup swap space controller help sysadmins
>> achieve performance or fairness goals on a system?
>>
> Perforamnce is not the first goal of this swap controller, I think.
> This is for resouce isolation/overcommiting.
>
> 1. Some _crazy_ people considers swap as very-slow-memory resource ;)
>   I don't think so but I know there are tons of people....
>
> 2. Resource Isolation.
>   When a cgroup has memory limitation, it can create tons of swap.
>   For example, limit a cgroup's memory to be 128M and malloc 3G bytes.
>   2.8Gbytes of swap will be used _easily_. A process can use up all swap.
>   In that case, other process can't use swap.
>
> IIRC, a man shown his motivation to controll swap in OLS2007/BOF as following.
> ==
> Consider following system. (and there is no swap controller.)
> Memory 4G. Swap 1G. with 2 cgroups A, B.
>
> state 1) swap is not used.
>   A....memory limit to be 1G no swap usage memory_usage=0M
>   B....memory limit to be 1G no swap usage memory_usage=0M
>
> state 2) Run a big program on A.
>   A....memory limit to be 1G and try to use 1.7G. uses 700MBytes of swap.
>   memory_usage=1G swap_usage=700M
>   B....memory_usage=0M
>
> state 3) A some of programs ends in 'A'
>   A....memory_usage=500M swap_usage=700M
>   B....memory_usage=0M.
>
> state 4) Run a big program on B.
>   A...memory_usage=500M swap_usage=700M.
>   B...memory_usage=1G   swap_usage=300M
>
> Group B can only use 1.3G because of unfair swap use of group A.
> But users think why A uses 700M of swap with 500M of free memory....

>
> If we don't have limitation to swap, we'll have to innovate a way to move swap
> to memory in some reasonable logic.
>
> Thanks,
> -Kame

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Fri, 23 May 2008 04:27:23 GMT
View Forum Message <> Reply to Message

Hi.

On 2008/05/23 6:27 +0900, Balbir Singh wrote:
> Daisuke Nishimura wrote:
>> Hi.
>>
>> I updated my swapcgroup patch.
>>
>> Major changes from previous version(*1):
>> - Rebased on 2.6.26-rc2-mm1 + KAMEZAWA-san's performance
>>   improvement patchset v4.
>> - Implemented as a add-on to memory cgroup.
>>   So, there is no need to add a new member to page_cgroup now.
>> - (NEW)Modified vm_swap_full() to calculate the rate of
>>   swap usage per cgroup.
>>
>> Patchs:
>> - [1/4] add cgroup files
>> - [2/4] add member to swap_info_struct for cgroup
>> - [3/4] implement charge/uncharge
>> - [4/4] modify vm_swap_full for cgroup
>>
>> ToDo:
>> - handle force_empty.
>> - make it possible for users to select if they use
>>   this feature or not, and avoid overhead for users

>> not using this feature.
>> - move charges along with task move between cgroups.
>>
>
> Thanks for looking into this. Yamamoto-San is also looking into a swap
> controller. Is there a consensus on the approach?
>
Not yet, but I think we should have some consensus each other
before going further.

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by KOSAKI Motohiro on Fri, 23 May 2008 04:30:27 GMT
View Forum Message <> Reply to Message

> One option is to limit the virtual address space usage of the cgroup to ensure
> that swap usage of a cgroup will *not* exceed the specified limit. Along with a
> good swap controller, it should provide good control over the cgroup's memory usage.

unfortunately, it doesn't works in real world.
IMHO you said as old good age.

because, Some JavaVM consume crazy large virtual address space.
it often consume >10x than phycal memory consumption.

yes, that behaviour is crazy. but it is used widely.
thus, We shouldn't assume virtual address space limitation.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Fri, 23 May 2008 04:51:04 GMT
View Forum Message <> Reply to Message

KOSAKI Motohiro wrote:
>> One option is to limit the virtual address space usage of the cgroup to ensure
>> that swap usage of a cgroup will *not* exceed the specified limit. Along with a
>> good swap controller, it should provide good control over the cgroup's memory usage.
>
> unfortunately, it doesn't works in real world.
> IMHO you said as old good age.
>
> because, Some JavaVM consume crazy large virtual address space.
> it often consume >10x than phycal memory consumption.
>

Have you seen any real world example of this? The overcommit feature of Linux.
We usually by default limit the overcommit to 1.5 times total memory (IIRC).
Yes, one can override that value, you get the same flexibility with the virtual
address space controller.

I thought java was particular about it with it's heap management options and policy.

> yes, that behaviour is crazy. but it is used widely.
> thus, We shouldn't assume virtual address space limitation.

It's useful in many cases to limit the virtual address space - to allow
applications to deal with memory failure, rather than

1. OOM the application later
2. Allow uncontrolled swapping (swap controller would help here)

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by KAMEZAWA Hiroyuki on Fri, 23 May 2008 05:19:37 GMT
View Forum Message <> Reply to Message

On Fri, 23 May 2008 10:21:04 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KOSAKI Motohiro wrote:
> >> One option is to limit the virtual address space usage of the cgroup to ensure

> >> that swap usage of a cgroup will *not* exceed the specified limit. Along with a
> >> good swap controller, it should provide good control over the cgroup's memory usage.
> >
> > unfortunately, it doesn't works in real world.
> > IMHO you said as old good age.
> >
> > because, Some JavaVM consume crazy large virtual address space.
> > it often consume >10x than phycal memory consumption.
> >
>
> Have you seen any real world example of this?
I have no objection to that virual-address-space limitation can work well on
well-controlled-system. But there are more complicated systems in chaos.

One example I know was that a team for the system tried to count all vm space
for setting vm.overcommit_memory to be proper value. The just found they can't
do it on a server with tens of applications after a month.

One of difficult problem is that a system administrator can't assume the total
size of virtual address space of proprietary applications/library.
An application designer can estimate "the virutal address usage of an application
is between XXM to XXXXM. but admin can't esitmate the total.

In above case, the most problematic user of virual adddress space was pthreads.
Default stack size of pthreads on ia64 was 10M bytes (--; And almost all application
doesn't answer how small they can set its stack size to. It's crazy to set this value
per applications. Then, "stack" of 2000 threads requires 20G bytes of virtual
address space on 12G system ;) They failed to use overcommit.

Thanks,
-Kame

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by David Singleton on Fri, 23 May 2008 05:29:31 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> KOSAKI Motohiro wrote:
>>> One option is to limit the virtual address space usage of the cgroup to ensure
>>> that swap usage of a cgroup will *not* exceed the specified limit. Along with a

>>> good swap controller, it should provide good control over the cgroup's memory usage.
>> unfortunately, it doesn't works in real world.
>> IMHO you said as old good age.
>>
>> because, Some JavaVM consume crazy large virtual address space.
>> it often consume >10x than phycal memory consumption.
>>
>
> Have you seen any real world example of this?

At the unsophisticated end, there are lots of (Fortran) HPC applications
with very large static array declarations but only "use" a small fraction
of that. Those users know they only need a small fraction and are happy
to volunteer small physical memory limits that we (admins/queuing
systems) can apply.

At the sophisticated end, the use of numerous large memory maps in
parallel HPC applications to gain visibility into other processes is
growing. We have processes with VSZ > 400GB just because they have
4GB maps into 127 other processes. Their physical page use is of
the order 2GB.

Imposing virtual address space limits on these applications is
meaningless.

The overcommit feature of Linux.
> We usually by default limit the overcommit to 1.5 times total memory (IIRC).
> Yes, one can override that value, you get the same flexibility with the virtual
> address space controller.
>
> I thought java was particular about it with it's heap management options and policy.
>
>> yes, that behaviour is crazy. but it is used widely.
>> thus, We shouldn't assume virtual address space limitation.
>
> It's useful in many cases to limit the virtual address space - to allow
> applications to deal with memory failure, rather than
>
> 1. OOM the application later
> 2. Allow uncontrolled swapping (swap controller would help here)
>

David

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by KOSAKI Motohiro on Fri, 23 May 2008 06:00:42 GMT
View Forum Message <> Reply to Message

> > Have you seen any real world example of this?
>
> At the unsophisticated end, there are lots of (Fortran) HPC applications
> with very large static array declarations but only "use" a small fraction
> of that. Those users know they only need a small fraction and are happy
> to volunteer small physical memory limits that we (admins/queuing
> systems) can apply.
>
> At the sophisticated end, the use of numerous large memory maps in
> parallel HPC applications to gain visibility into other processes is
> growing. We have processes with VSZ > 400GB just because they have
> 4GB maps into 127 other processes. Their physical page use is of
> the order 2GB.

Ah, agreed.
Fujitsu HPC user said similar things ago.

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

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Fri, 23 May 2008 06:45:15 GMT
View Forum Message <> Reply to Message

KOSAKI Motohiro wrote:
>>> Have you seen any real world example of this?
>> At the unsophisticated end, there are lots of (Fortran) HPC applications
>> with very large static array declarations but only "use" a small fraction
>> of that. Those users know they only need a small fraction and are happy
>> to volunteer small physical memory limits that we (admins/queuing
>> systems) can apply.
>>
>> At the sophisticated end, the use of numerous large memory maps in
>> parallel HPC applications to gain visibility into other processes is
>> growing. We have processes with VSZ > 400GB just because they have

>> 4GB maps into 127 other processes. Their physical page use is of
>> the order 2GB.
>
> Ah, agreed.
> Fujitsu HPC user said similar things ago.

OK, so this use case is HPC specific. I am not against the swap controller, but overcommit can lead to problems if not controlled - such as OOM kill. The virtual address space limit helps applications fail gracefully rather than swap out excessively or OOM.

I suspect there'll be applications that swing both ways.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by Daisuke Nishimura on Fri, 23 May 2008 11:52:29 GMT
View Forum Message <> Reply to Message

On 2008/05/22 16:37 +0900, KAMEZAWA Hiroyuki wrote:
> 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() ?
I'm sorry, but I can't understand this situation.
memory controller is initialized at kernel initialization,
so aren't processes created after it is initialized?

> How about returning 'failure' in this case ? I think returning 'failure' here
> is not so bad.
>
>
Which of below do you mean by 'failure'?

A. make it fail to get swap entry, so the page cannot be swapped out.
B. don't charge this swap entry to any cgroup, but the page
   would be swapped out.

I don't want to do B, because I don't want to make such
not-charged-to-anywhere entries.
And I've seen several times this condition(!pc) becomes true,
so I charged to init_mem_cgroup.

BTW, I noticed that almost all of functions I added by this patch set
should check "mem_cgroup_subsys.disabled" first because it depend on
memory cgroup.

>> +
>> + css_get(&mem->css);
>
> move this css_get() before unlock_page_cgroup() is safer.
>
OK, thanks.

Daisuke Nishimura.

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by Daisuke Nishimura on Fri, 23 May 2008 12:26:47 GMT
View Forum Message <> Reply to Message

On 2008/05/22 21:32 +0900, KOSAKI Motohiro wrote:

> perhaps, I don't understand your intention exactly.
> Why can't you make wrapper function?
>
> e.g.
> vm_swap_full(page_to_memcg(page))
>
OK.
I'll try it.

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

---

Subject: Re: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by yamamoto on Sun, 25 May 2008 23:35:32 GMT
View Forum Message <> Reply to Message

> How about something like this?
>
> :
> usage = swap_cgroup_read_usage(mem); //no need to align to number of page
> limit = swap_cgroup_read_limit(mem); //no need to align to number of page
> ret = (usage * 2 > limit) || (nr_swap_pages * 2 < total_swap_pages)
> :

it seems reasonable to me.

YAMAMOTO Takashi

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

---

Subject: Re: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by KAMEZAWA Hiroyuki on Mon, 26 May 2008 00:55:43 GMT
View Forum Message <> Reply to Message

On Fri, 23 May 2008 20:52:29 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> On 2008/05/22 16:37 +0900, KAMEZAWA Hiroyuki wrote:
> > 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() ?
> I'm sorry, but I can't understand this situation.
> memory controller is initialized at kernel initialization,
> so aren't processes created after it is initialized?
>
I think add_to_page_cache() may be called before late_init..I'll check again.
(Because I saw some panics related to it, but I noticed this is _swap_ controller
 ...)

> > How about returning 'failure' in this case ? I think returning 'failure' here
> > is not so bad.
> >
> >
> Which of below do you mean by 'failure'?
>
> A. make it fail to get swap entry, so the page cannot be swapped out.
> B. don't charge this swap entry to any cgroup, but the page
>   would be swapped out.
means A.

>
> I don't want to do B, because I don't want to make such
> not-charged-to-anywhere entries.
> And I've seen several times this condition(!pc) becomes true,
> so I charged to init_mem_cgroup.
>
>
> BTW, I noticed that almost all of functions I added by this patch set
> should check "mem_cgroup_subsys.disabled" first because it depend on

> memory cgroup.
>
Ah, yes, please.

Thanks,
-Kame

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by yamamoto on Tue, 27 May 2008 07:31:18 GMT
View Forum Message <> Reply to Message

hi,

> > Thanks for looking into this. Yamamoto-San is also looking into a swap
> > controller. Is there a consensus on the approach?
> >
> Not yet, but I think we should have some consensus each other
> before going further.
>
>
> Thanks,
> Daisuke Nishimura.

while nishimura-san's one still seems to have a lot of todo,
it seems good enough as a start point to me.
so i'd like to withdraw mine.

nishimura-san, is it ok for you?

YAMAMOTO Takashi

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Tue, 27 May 2008 07:42:04 GMT
View Forum Message <> Reply to Message

YAMAMOTO Takashi wrote:

> hi,
>
>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>> controller. Is there a consensus on the approach?
>>>
>> Not yet, but I think we should have some consensus each other
>> before going further.
>>
>>
>> Thanks,
>> Daisuke Nishimura.
>
> while nishimura-san's one still seems to have a lot of todo,
> it seems good enough as a start point to me.
> so i'd like to withdraw mine.
>
> nishimura-san, is it ok for you?
>

I would suggest that me merge the good parts from both into the swap controller.
Having said that I'll let the two of you decide on what the good aspects of both
are. I cannot see any immediate overlap, but there might be some w.r.t.
infrastructure used.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Tue, 27 May 2008 08:30:08 GMT
View Forum Message <> Reply to Message

On 2008/05/27 16:42 +0900, Balbir Singh wrote:
> YAMAMOTO Takashi wrote:
>> hi,
>>
>>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>>> controller. Is there a consensus on the approach?
>>>>
>>> Not yet, but I think we should have some consensus each other
>>> before going further.

>>>
>>>
>>> Thanks,
>>> Daisuke Nishimura.
>> while nishimura-san's one still seems to have a lot of todo,
>> it seems good enough as a start point to me.
>> so i'd like to withdraw mine.
>>
>> nishimura-san, is it ok for you?
>>
Of cource.
I'll work hard to make it better.

>
> I would suggest that me merge the good parts from both into the swap controller.
> Having said that I'll let the two of you decide on what the good aspects of both
> are. I cannot see any immediate overlap, but there might be some w.r.t.
> infrastructure used.
>
Well, you mean you'll make another patch based on yamamoto-san's
and mine?

Basically, I think it's difficult to merge
because we charge different objects.

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Tue, 27 May 2008 13:18:05 GMT
View Forum Message <> Reply to Message

Daisuke Nishimura wrote:
> On 2008/05/27 16:42 +0900, Balbir Singh wrote:
>> YAMAMOTO Takashi wrote:
>>> hi,
>>>
>>>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>>>> controller. Is there a consensus on the approach?
>>>>>
>>>> Not yet, but I think we should have some consensus each other
>>>> before going further.

>>>>
>>>>
>>>> Thanks,
>>>> Daisuke Nishimura.
>>> while nishimura-san's one still seems to have a lot of todo,
>>> it seems good enough as a start point to me.
>>> so i'd like to withdraw mine.
>>>
>>> nishimura-san, is it ok for you?
>>>
> Of cource.
> I'll work hard to make it better.
>
>> I would suggest that me merge the good parts from both into the swap controller.
>> Having said that I'll let the two of you decide on what the good aspects of both
>> are. I cannot see any immediate overlap, but there might be some w.r.t.
>> infrastructure used.
>>
> Well, you mean you'll make another patch based on yamamoto-san's
> and mine?
>
> Basically, I think it's difficult to merge
> because we charge different objects.
>

Yes, I know - that's why I said infrastructure, to see the grouping and common
data structure aspects. I'll try and review the code.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 3/4] swapcgroup: implement charge/uncharge
Posted by KAMEZAWA Hiroyuki on Tue, 27 May 2008 13:42:03 GMT
View Forum Message <> Reply to Message

On Mon, 26 May 2008 09:57:06 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> On Fri, 23 May 2008 20:52:29 +0900
> Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

>
> > On 2008/05/22 16:37 +0900, KAMEZAWA Hiroyuki wrote:
> > > 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() ?
> > I'm sorry, but I can't understand this situation.
> > memory controller is initialized at kernel initialization,
> > so aren't processes created after it is initialized?
> >
> I think add_to_page_cache() may be called before late_init..I'll check again.
> (Because I saw some panics related to it, but I noticed this is _swap_ controller
> ...)

Now, force_empty() will create a page which is used but
page->page_cgroup is NULL page. I'm now writing a workaround (1/4 in my newest set)
but it's better to check page->page_cgroup is NULL or not.

Thanks,
-Kame

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

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Tue, 27 May 2008 13:42:10 GMT
View Forum Message <> Reply to Message

On 2008/05/27 22:18 +0900, Balbir Singh wrote:
> Daisuke Nishimura wrote:
>> On 2008/05/27 16:42 +0900, Balbir Singh wrote:
>>> YAMAMOTO Takashi wrote:
>>>> hi,
>>>>
>>>>>>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>>>>>>> controller. Is there a consensus on the approach?
>>>>>>>>
>>>>>>> Not yet, but I think we should have some consensus each other
>>>>>>> before going further.
>>>>>>>
>>>>>>>
>>>>>>> Thanks,
>>>>>>> Daisuke Nishimura.
>>>> while nishimura-san's one still seems to have a lot of todo,
>>>> it seems good enough as a start point to me.
>>>> so i'd like to withdraw mine.
>>>>
>>>> nishimura-san, is it ok for you?
>>>>
>> Of cource.
>> I'll work hard to make it better.
>>
>>> I would suggest that me merge the good parts from both into the swap controller.
>>> Having said that I'll let the two of you decide on what the good aspects of both
>>> are. I cannot see any immediate overlap, but there might be some w.r.t.
>>> infrastructure used.
>>>
>> Well, you mean you'll make another patch based on yamamoto-san's
>> and mine?
>>
>> Basically, I think it's difficult to merge
>> because we charge different objects.
>>
>
> Yes, I know - that's why I said infrastructure, to see the grouping and common
> data structure aspects. I'll try and review the code.
>
OK.

I'll wait for your patch.

Thanks,

Daisuke Nishimura.

_____

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Balbir Singh on Tue, 27 May 2008 13:46:40 GMT
View Forum Message <> Reply to Message

Daisuke Nishimura wrote:
> On 2008/05/27 22:18 +0900, Balbir Singh wrote:
>> Daisuke Nishimura wrote:
>>> On 2008/05/27 16:42 +0900, Balbir Singh wrote:
>>>> YAMAMOTO Takashi wrote:
>>>>> hi,
>>>>>
>>>>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>>>>> controller. Is there a consensus on the approach?
>>>>>>
>>>>> Not yet, but I think we should have some consensus each other
>>>>> before going further.
>>>>>
>>>>>
>>>>> Thanks,
>>>>> Daisuke Nishimura.
>>>>> while nishimura-san's one still seems to have a lot of todo,
>>>>> it seems good enough as a start point to me.
>>>>> so i'd like to withdraw mine.
>>>>>
>>>>> nishimura-san, is it ok for you?
>>>>>
>>> Of cource.
>>> I'll work hard to make it better.
>>>
>>>> I would suggest that me merge the good parts from both into the swap controller.
>>>> Having said that I'll let the two of you decide on what the good aspects of both
>>>> are. I cannot see any immediate overlap, but there might be some w.r.t.
>>>> infrastructure used.
>>>>
>>> Well, you mean you'll make another patch based on yamamoto-san's
>>> and mine?
>>>
>>> Basically, I think it's difficult to merge
>>> because we charge different objects.
>>>
>> Yes, I know - that's why I said infrastructure, to see the grouping and common

>> data structure aspects. I'll try and review the code.
>>
> OK.
>
> I'll wait for your patch.

Hi, Daisuke-San

I am not sending out any patch (sorry for the confusion, if I caused it). I am
going to review the swapcgroup patchset.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH 0/4] swapcgroup(v2)
Posted by Daisuke Nishimura on Tue, 27 May 2008 14:00:43 GMT
View Forum Message <> Reply to Message

On 2008/05/27 22:46 +0900, Balbir Singh wrote:
> Daisuke Nishimura wrote:
>> On 2008/05/27 22:18 +0900, Balbir Singh wrote:
>>> Daisuke Nishimura wrote:
>>>> On 2008/05/27 16:42 +0900, Balbir Singh wrote:
>>>>> YAMAMOTO Takashi wrote:
>>>>>> hi,
>>>>>>
>>>>>>> Thanks for looking into this. Yamamoto-San is also looking into a swap
>>>>>>> controller. Is there a consensus on the approach?
>>>>>>>
>>>>>>> Not yet, but I think we should have some consensus each other
>>>>>>> before going further.
>>>>>>>
>>>>>>>
>>>>>>> Thanks,
>>>>>>> Daisuke Nishimura.
>>>>>> while nishimura-san's one still seems to have a lot of todo,
>>>>>> it seems good enough as a start point to me.
>>>>>> so i'd like to withdraw mine.
>>>>>>
>>>>>> nishimura-san, is it ok for you?

>>>>>>
>>>> Of cource.
>>>> I'll work hard to make it better.
>>>>
>>>>> I would suggest that me merge the good parts from both into the swap controller.
>>>>> Having said that I'll let the two of you decide on what the good aspects of both
>>>>> are. I cannot see any immediate overlap, but there might be some w.r.t.
>>>>> infrastructure used.
>>>>>
>>>> Well, you mean you'll make another patch based on yamamoto-san's
>>>> and mine?
>>>>
>>>> Basically, I think it's difficult to merge
>>>> because we charge different objects.
>>>>
>>> Yes, I know - that's why I said infrastructure, to see the grouping and common
>>> data structure aspects. I'll try and review the code.
>>>
>> OK.
>>
>> I'll wait for your patch.
>
> Hi, Daisuke-San
>
> I am not sending out any patch (sorry for the confusion, if I caused it). I am
> going to review the swapcgroup patchset.
>

No problem :-)

I would very appreciate it if you review swap cgroup patches.

I'll update my patches based on your and others' comments.

Thanks,
Daisuke Nishimura.

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