
Subject: [PATCH 2/4] Block I/O tracking
Posted by [Hirokazu Takahashi](#) on Tue, 18 Mar 2008 09:29:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

This patch implements the bio cgroup on the memory cgroup.

Signed-off-by: Hirokazu Takahashi <taka@valinux.co.jp>

```
--- linux-2.6.25-rc5.pagecgroup2/include/linux/memcontrol.h 2008-03-18 12:45:14.000000000 +0900
+++ linux-2.6.25-rc5-mm1/include/linux/memcontrol.h 2008-03-18 12:55:59.000000000 +0900
@@ -54,6 +54,10 @@ struct page_cgroup {
    struct list_head lru; /* per cgroup LRU list */
    struct mem_cgroup *mem_cgroup;
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+#ifdef CONFIG_CGROUP_BIO
+    struct list_head blist; /* for bio_cgroup page list */
+    struct bio_cgroup *bio_cgroup;
#endif
    struct page *page;
    int ref_cnt; /* cached, mapped, migrating */
    int flags;
--- linux-2.6.25-rc5.pagecgroup2/include/linux/biocontrol.h 2008-03-17 22:06:49.000000000 +0900
+++ linux-2.6.25-rc5-mm1/include/linux/biocontrol.h 2008-03-18 14:19:53.000000000 +0900
@@ -0,0 +1,148 @@
+#include <linux/cgroup.h>
+#include <linux/mm.h>
+#include <linux/memcontrol.h>
+
#ifndef _LINUX_BIOCONTROL_H
#define _LINUX_BIOCONTROL_H
+
#endif CONFIG_CGROUP_BIO
+
+struct io_context;
+
+struct bio_cgroup {
+    struct cgroup_subsys_state css;
+    int id;
+    struct io_context *io_context; /* default io_context */
+/* struct radix_tree_root io_context_root; per device io_context */
+    spinlock_t page_list_lock;
+    struct list_head page_list;
+};
```

```

+static inline int bio_cgroup_disabled(void)
+{
+    return bio_cgroup_subsys.disabled;
+}
+
+extern void mm_init_bio_cgroup(struct mm_struct *, struct task_struct *);
+
+static inline void __bio_cgroup_add_page(struct page_cgroup *pc)
+{
+    struct bio_cgroup *biog = pc->bio_cgroup;
+    list_add(&pc->blist, &biog->page_list);
+}
+
+static inline void bio_cgroup_add_page(struct page_cgroup *pc)
+{
+    struct bio_cgroup *biog = pc->bio_cgroup;
+    unsigned long flags;
+    spin_lock_irqsave(&biog->page_list_lock, flags);
+    __bio_cgroup_add_page(pc);
+    spin_unlock_irqrestore(&biog->page_list_lock, flags);
+}
+
+static inline void __bio_cgroup_remove_page(struct page_cgroup *pc)
+{
+    list_del_init(&pc->blist);
+}
+
+static inline void bio_cgroup_remove_page(struct page_cgroup *pc)
+{
+    struct bio_cgroup *biog = pc->bio_cgroup;
+    unsigned long flags;
+    spin_lock_irqsave(&biog->page_list_lock, flags);
+    __bio_cgroup_remove_page(pc);
+    spin_unlock_irqrestore(&biog->page_list_lock, flags);
+}
+
+static inline void get_bio_cgroup(struct bio_cgroup *biog)
+{
+    css_get(&biog->css);
+}
+
+static inline void put_bio_cgroup(struct bio_cgroup *biog)
+{
+    css_put(&biog->css);
+}
+
+static inline void set_bio_cgroup(struct page_cgroup *pc,
+    struct bio_cgroup *biog)

```

```

+{
+ pc->bio_cgroup = biog;
+}
+
+static inline void clear_bio_cgroup(struct page_cgroup *pc)
+{
+ struct bio_cgroup *biog = pc->bio_cgroup;
+ pc->bio_cgroup = NULL;
+ put_bio_cgroup(biog);
+}
+
+/* This could be called in an RCU-protected section. */
+static inline struct bio_cgroup *mm_get_bio_cgroup(struct mm_struct *mm)
+{
+ struct bio_cgroup *biog = rcu_dereference(mm->bio_cgroup);
+ get_bio_cgroup(biog);
+ return biog;
+}
+
+static inline void mm_free_bio_cgroup(struct mm_struct *mm)
+{
+ put_bio_cgroup(mm->bio_cgroup);
+}
+
+extern int get_bio_cgroup_id(struct page *page);
+
+#else /* CONFIG_CGROUP_BIO */
+
+struct bio_cgroup;
+
+static inline int bio_cgroup_disabled(void)
+{
+     return 1;
+}
+
+static inline void mm_init_bio_cgroup(struct mm_struct *mm, struct task_struct *p)
+{
+}
+
+static inline void bio_cgroup_add_page(struct page_cgroup *pc)
+{
+}
+
+static inline void bio_cgroup_remove_page(struct page_cgroup *pc)
+{
+}
+
+static inline void get_bio_cgroup(struct bio_cgroup *biog)

```

```

+{
+}
+
+static inline void put_bio_cgroup(struct bio_cgroup *biog)
+{
+}
+
+static inline void set_bio_cgroup(struct page_cgroup *pc,
+    struct bio_cgroup *biog)
+{
+}
+
+static inline void clear_bio_cgroup(struct page_cgroup *pc)
+{
+}
+
+static inline struct bio_cgroup *mm_get_bio_cgroup(struct mm_struct *mm)
+{
+    return NULL;
+}
+
+static inline void mm_free_bio_cgroup(struct mm_struct *mm)
+{
+}
+
+static inline int get_bio_cgroup_id(struct page *page)
+{
+    return 0;
+}
#endif /* CONFIG_CGROUP_BIO */
+
#endif /* _LINUX_BIOCONTROL_H */
--- linux-2.6.25-rc5.pagecgroup2/include/linux/cgroup_subsys.h 2008-03-18 12:45:14.000000000
+0900
+++ linux-2.6.25-rc5-mm1/include/linux/cgroup_subsys.h 2008-03-18 12:55:59.000000000 +0900
@@ -42,3 +42,9 @@ SUBSYS(mem_cgroup)
#endif

/*
+
+#ifdef CONFIG_CGROUP_BIO
+SUBSYS(bio_cgroup)
#endif
+
+*/
--- linux-2.6.25-rc5.pagecgroup2/include/linux/mm_types.h 2008-03-18 12:45:14.000000000
+0900
+++ linux-2.6.25-rc5-mm1/include/linux/mm_types.h 2008-03-18 12:55:59.000000000 +0900

```

```

@@ -230,6 +230,9 @@ struct mm_struct {
#endif CONFIG_CGROUP_MEM_RES_CTLR
    struct mem_cgroup *mem_cgroup;
#endif
+ifdef CONFIG_CGROUP_BIO
+ struct bio_cgroup *bio_cgroup;
+endif

#ifndef CONFIG_PROC_FS
/* store ref to file /proc/<pid>/exe symlink points to */
--- linux-2.6.25-rc5.pagecgroup2/init/Kconfig 2008-03-18 12:45:14.000000000 +0900
+++ linux-2.6.25-rc5-mm1/init/Kconfig 2008-03-18 12:55:59.000000000 +0900
@@ -379,9 +379,15 @@ config CGROUP_MEM_RES_CTLR
    Only enable when you're ok with these trade offs and really
    sure you need the memory resource controller.

+config CGROUP_BIO
+    bool "Block I/O cgroup subsystem"
+    depends on CGROUPS
+    help
+        Provides a Resource Controller that manages Block I/O.
+
+config CGROUP_PAGE
+    def_bool y
+    depends on CGROUP_MEM_RES_CTLR
+    depends on CGROUP_MEM_RES_CTLR || CGROUP_BIO

config SYSFS_DEPRECATED
    bool
--- linux-2.6.25-rc5.pagecgroup2/mm/Makefile 2008-03-18 12:45:14.000000000 +0900
+++ linux-2.6.25-rc5-mm1/mm/Makefile 2008-03-18 12:55:59.000000000 +0900
@@ -33,4 +33,5 @@ obj-$(CONFIG_MIGRATION) += migrate.o
obj-$(CONFIG_SMP) += allocpercpu.o
obj-$(CONFIG_QUICKLIST) += quicklist.o
obj-$(CONFIG_CGROUP_PAGE) += memcontrol.o
+obj-$(CONFIG_CGROUP_BIO) += biocontrol.o

--- linux-2.6.25-rc5.pagecgroup2/mm/biocontrol.c 2008-03-17 22:06:49.000000000 +0900
+++ linux-2.6.25-rc5-mm1/mm/biocontrol.c 2008-03-18 12:55:59.000000000 +0900
@@ -0,0 +1,229 @@
+/* biocontrol.c - Block I/O Controller
+ *
+ * Copyright VA Linux Systems Japan, 2008
+ * Author Hirokazu Takahashi <taka@valinux.co.jp>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2 of the License, or

```

```

+ * (at your option) any later version.
+
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ * GNU General Public License for more details.
+ */
+
+#include <linux/module.h>
+#include <linux/cgroup.h>
+#include <linux/mm.h>
+#include <linux/smp.h>
+#include <linux/bit_spinlock.h>
+#include <linux/idr.h>
+#include <linux/err.h>
+#include <linux/biocontrol.h>
+
+/* return corresponding bio_cgroup object of a cgroup */
+static inline struct bio_cgroup *cgroup_bio(struct cgroup *cgrp)
+{
+    return container_of(cgroup_subsys_state(cgrp, bio_cgroup_subsys_id),
+        struct bio_cgroup, css);
+}
+
+static inline struct bio_cgroup *bio_cgroup_from_task(struct task_struct *p)
+{
+    return container_of(task_subsys_state(p, bio_cgroup_subsys_id),
+        struct bio_cgroup, css);
+}
+
+void mm_init_bio_cgroup(struct mm_struct *mm, struct task_struct *p)
+{
+    struct bio_cgroup *biog;
+
+    biog = bio_cgroup_from_task(p);
+    get_bio_cgroup(biog);
+    mm->bio_cgroup = biog;
+}
+
+static struct idr bio_cgroup_id;
+static DEFINE_SPINLOCK(bio_cgroup_idr_lock);
+
+static struct cgroup_subsys_state *
+bio_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)
+{
+    struct bio_cgroup *biog;
+    int error;
+

```

```

+ biog = kzalloc(sizeof(*biog), GFP_KERNEL);
+ if (!biog)
+   return ERR_PTR(-ENOMEM);
+ if (!cgrp->parent) {
+   init_mm.bio_cgroup = biog;
+   idr_init(&bio_cgroup_id);
+   biog->id = 0;
+ } else {
+retry:
+   if (unlikely(!idr_pre_get(&bio_cgroup_id, GFP_KERNEL))) {
+     error = -EAGAIN;
+     goto out;
+   }
+   spin_lock_irq(&bio_cgroup_idr_lock);
+   error = idr_get_new_above(&bio_cgroup_id, (void *)biog, 1, &biog->id);
+   spin_unlock_irq(&bio_cgroup_idr_lock);
+   if (error == -EAGAIN)
+     goto retry;
+   else if (error)
+     goto out;
+ }
+
+ INIT_LIST_HEAD(&biog->page_list);
+ spin_lock_init(&biog->page_list_lock);
+
+ /* Bind the cgroup to bio_cgroup object we just created */
+ biog->css.cgroup = cgrp;
+
+ return &biog->css;
+out:
+ kfree(biog);
+ return ERR_PTR(error);
+}
+
+#define FORCE_UNCHARGE_BATCH (128)
+static void bio_cgroup_force_empty(struct bio_cgroup *biog)
+{
+ struct page_cgroup *pc;
+ struct page *page;
+ int count = FORCE_UNCHARGE_BATCH;
+ struct list_head *list = &biog->page_list;
+ unsigned long flags;
+
+ spin_lock_irqsave(&biog->page_list_lock, flags);
+ while (!list_empty(list)) {
+   pc = list_entry(list->prev, struct page_cgroup, blist);
+   page = pc->page;
+   get_page(page);

```

```

+ spin_unlock_irqrestore(&biog->page_list_lock, flags);
+ mem_cgroup_uncharge_page(page);
+ put_page(page);
+ if (--count <= 0) {
+ count = FORCE_UNCHARGE_BATCH;
+ cond_resched();
+ }
+ spin_lock_irqsave(&biog->page_list_lock, flags);
+ }
+ spin_unlock_irqrestore(&biog->page_list_lock, flags);
+ return;
+}
+
+static void bio_cgroup_pre_destroy(struct cgroup_subsys *ss, struct cgroup *cgrp)
+{
+ struct bio_cgroup *biog = cgroup_bio(cgrp);
+ bio_cgroup_force_empty(biog);
+}
+
+static void bio_cgroup_destroy(struct cgroup_subsys *ss, struct cgroup *cgrp)
+{
+ struct bio_cgroup *biog = cgroup_bio(cgrp);
+
+ spin_lock_irq(&bio_cgroup_idr_lock);
+ idr_remove(&bio_cgroup_id, biog->id);
+ spin_unlock_irq(&bio_cgroup_idr_lock);
+
+ kfree(biog);
+}
+
+struct bio_cgroup *find_bio_cgroup(int id)
+{
+ struct bio_cgroup *biog;
+ spin_lock_irq(&bio_cgroup_idr_lock);
+ biog = (struct bio_cgroup *)
+ idr_find(&bio_cgroup_id, id);
+ spin_unlock_irq(&bio_cgroup_idr_lock);
+ get_bio_cgroup(biog);
+ return biog;
+}
+
+int get_bio_cgroup_id(struct page *page)
+{
+ struct page_cgroup *pc;
+ int id = 0;
+ lock_page_cgroup(page);
+ pc = page_get_page_cgroup(page);
+ if (pc)

```

```

+ id = pc->bio_cgroup->id;
+ unlock_page_cgroup(page);
+ return id;
+}
+EXPORT_SYMBOL(get_bio_cgroup_id);
+
+static u64 bio_id_read(struct cgroup *cgrp, struct cftype *cft)
+{
+ struct bio_cgroup *biog = cgroup_bio(cgrp);
+
+ return (u64) biog->id;
+}
+
+
+static struct cftype bio_files[] = {
+ {
+ .name = "id",
+ .read_u64 = bio_id_read,
+ },
+};
+
+
+static int bio_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ if (bio_cgroup_disabled())
+ return 0;
+ return cgroup_add_files(cont, ss, bio_files, ARRAY_SIZE(bio_files));
+}
+
+
+static void bio_cgroup_move_task(struct cgroup_subsys *ss,
+ struct cgroup *cont,
+ struct cgroup *old_cont,
+ struct task_struct *p)
+{
+ struct mm_struct *mm;
+ struct bio_cgroup *biog, *old_biog;
+
+ if (bio_cgroup_disabled())
+ return;
+
+ mm = get_task_mm(p);
+ if (mm == NULL)
+ return;
+
+ biog = cgroup_bio(cont);
+ old_biog = cgroup_bio(old_cont);
+
+ if (biog == old_biog)
+ goto out;

```

```

+
+ /*
+ * Only thread group leaders are allowed to migrate, the mm_struct is
+ * in effect owned by the leader
+ */
+ if (p->tgid != p->pid)
+ goto out;
+
+ get_bio_cgroup(biog);
+ rcu_assign_pointer(mm->bio_cgroup, biog);
+ put_bio_cgroup(old_biog);
+
+out:
+ mmput(mm);
+ return;
+}
+
+
+struct cgroup_subsys bio_cgroup_subsys = {
+ .name      = "bio",
+ .subsys_id = bio_cgroup_subsys_id,
+ .create    = bio_cgroup_create,
+ .destroy   = bio_cgroup_destroy,
+ .pre_destroy = bio_cgroup_pre_destroy,
+// .can_attach = bio_cgroup_can_attach,
+ .populate  = bio_cgroup_populate,
+ .attach    = bio_cgroup_move_task,
+ .early_init = 0,
+};
--- linux-2.6.25-rc5.pagecgroup2/mm/memcontrol.c 2008-03-18 12:45:14.000000000 +0900
+++ linux-2.6.25-rc5-mm1/mm/memcontrol.c 2008-03-18 14:19:04.000000000 +0900
@@ -20,6 +20,7 @@
#include <linux/res_counter.h>
#include <linux/memcontrol.h>
#include <linux/cgroup.h>
+#include <linux/biocontrol.h>
#include <linux/mm.h>
#include <linux/smp.h>
#include <linux/page-flags.h>
@@ -784,6 +785,7 @@ void mm_init_cgroup(struct mm_struct *mm
css_get(&mem->css);
mm->mem_cgroup = mem;
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mm_init_bio_cgroup(mm, p);
}

void mm_free_cgroup(struct mm_struct *mm)
@@ -791,6 +793,7 @@ void mm_free_cgroup(struct mm_struct *mm

```

```

#endif CONFIG_CGROUP_MEM_RES_CTLR
css_put(&mm->mem_cgroup->css);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mm_free_bio_cgroup(mm);
}

static inline int page_cgroup_locked(struct page *page)
@@ -830,8 +833,9 @@ static int mem_cgroup_charge_common(stru
    unsigned long nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
    struct mem_cgroup_per_zone *mz;
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */

-
- if (mem_cgroup_disabled())
+ struct bio_cgroup *biog;
+
+ if (mem_cgroup_disabled() && bio_cgroup_disabled())
    return 0;

/*
@@ -879,6 +883,7 @@ retry:
 */
css_get(&mem->css);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ biog = mm_get_bio_cgroup(mm);
rcu_read_unlock();

#endif CONFIG_CGROUP_MEM_RES_CTLR
@@ -911,6 +916,7 @@ retry:
#endif CONFIG_CGROUP_MEM_RES_CTLR
pc->mem_cgroup = mem;
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ set_bio_cgroup(pc, biog);
pc->page = page;
pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
@@ -928,6 +934,7 @@ retry:
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    css_put(&mem->css);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ clear_bio_cgroup(pc);
kfree(pc);
goto retry;
}
@@ -939,6 +946,7 @@ retry:
__mem_cgroup_add_list(pc);
spin_unlock_irqrestore(&mz->lru_lock, flags);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ bio_cgroup_add_page(pc);

```

```

unlock_page_cgroup(page);
done:
@@ -947,6 +955,7 @@ out:
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
css_put(&mem->css);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ put_bio_cgroup(biog);
kfree(pc);
err:
return -ENOMEM;
@@ -978,7 +987,7 @@ void mem_cgroup_uncharge_page(struct pag
struct mem_cgroup_per_zone *mz;
unsigned long flags;

- if (mem_cgroup_disabled())
+ if (mem_cgroup_disabled() && bio_cgroup_disabled())
return;

/*
@@ -999,6 +1008,7 @@ void mem_cgroup_uncharge_page(struct pag
__mem_cgroup_remove_list(pc);
spin_unlock_irqrestore(&mz->lru_lock, flags);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ bio_cgroup_remove_page(pc);

page_assign_page_cgroup(page, NULL);
unlock_page_cgroup(page);
@@ -1008,6 +1018,7 @@ void mem_cgroup_uncharge_page(struct pag
res_counter_uncharge(&mem->res, PAGE_SIZE);
css_put(&mem->css);
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ clear_bio_cgroup(pc);

kfree(pc);
return;
@@ -1025,7 +1036,7 @@ int mem_cgroup_prepare_migration(struct
{
struct page_cgroup *pc;

- if (mem_cgroup_disabled())
+ if (mem_cgroup_disabled() && bio_cgroup_disabled())
return 0;

lock_page_cgroup(page);
@@ -1065,6 +1076,7 @@ void mem_cgroup_page_migration(struct pa
__mem_cgroup_remove_list(pc);
spin_unlock_irqrestore(&mz->lru_lock, flags);

```

```
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */  
+ bio_cgroup_remove_page(pc);  
  
    page_assign_page_cgroup(page, NULL);  
    unlock_page_cgroup(page);  
@@ -1079,6 +1091,7 @@ void mem_cgroup_page_migration(struct pa  
    __mem_cgroup_add_list(pc);  
    spin_unlock_irqrestore(&mz->lru_lock, flags);  
#endif /* CONFIG_CGROUP_MEM_RES_CTLR */  
+ bio_cgroup_add_page(pc);  
  
    unlock_page_cgroup(newpage);  
}
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
