
Subject: [PATCH][just for review] memory controller enhancements [0/5]

Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:48:20 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch set includes following functions for memory cgroup.

Based on 2.6.23-rc8-mm2 + My bugfix patch set.

- memory.force_empty ... uncharge all pages in cgroup.
- memory.stat ... status accounting in cgroup.

I merged YAMAMOTO-san's patch set for statistics to this set.

[1/5] ... force_empty patch

[2/5] ... remember page is charged as page-cache patch

[3/5] ... remember page is on which list patch

[4/5] ... memory cgroup statistics patch

[5/5] ... show statistics patch

Any comments are welcome.

Thanks,

-Kame

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: [PATCH][just for review] memory controller enhancements [1/5] force empty

Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:51:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch adds an interface "memory.force_empty".

Any write to this file will drop all charges in this cgroup if there is no task under.

```
%echo 1 > /...../memory.force_empty
```

will drop all charges of memory cgroup if cgroup's tasks is empty.

This is useful to invoke rmdir() against memory cgroup successfully.

Tested and worked well on x86_64/fake-NUMA system.

Changelog v2 -> v3:

- changed the name from force_reclaim to force_empty.

- fixed wrong usage of css_put() bug.

Changelog v1 -> v2:

- added a new interface force_reclaim.
- changes spin_lock to spin_lock_irqsave().

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

mm/memcontrol.c | 102 ++++++-----
1 file changed, 95 insertions(+), 7 deletions(-)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

```
=====
--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -469,6 +469,7 @@ void mem_cgroup_uncharge(struct page_cgr
    page = pc->page;
    /*
     * get page->cgroup and clear it under lock.
+ * force-empty can drop page->cgroup without checking refcnt.
     */
    if (clear_page_cgroup(page, pc) == pc) {
        mem = pc->mem_cgroup;
@@ -478,13 +479,6 @@ void mem_cgroup_uncharge(struct page_cgr
    list_del_init(&pc->lru);
    spin_unlock_irqrestore(&mem->lru_lock, flags);
    kfree(pc);
- } else {
- /*
- * Note:This will be removed when force-empty patch is
- * applied. just show warning here.
- */
- printk(KERN_ERR "Race in mem_cgroup_uncharge() ?");
- dump_stack();
    }
}
}
@@ -532,6 +526,70 @@ retry:
    return;
}

+/*
+ * This routine traverse page_cgroup in given list and drop them all.
+ * This routine ignores page_cgroup->ref_cnt.
+ * *And* this routine doesn't reclaim page itself, just removes page_cgroup.
+ */
```

```

+static void
+mem_cgroup_force_empty_list(struct mem_cgroup *mem, struct list_head *list)
+{
+ struct page_cgroup *pc;
+ struct page *page;
+ int count = SWAP_CLUSTER_MAX;
+ unsigned long flags;
+
+ spin_lock_irqsave(&mem->lru_lock, flags);
+
+ while (!list_empty(list)) {
+ pc = list_entry(list->prev, struct page_cgroup, lru);
+ page = pc->page;
+ /* Avoid race with charge */
+ atomic_set(&pc->ref_cnt, 0);
+ if (clear_page_cgroup(page, pc) == pc) {
+ css_put(&mem->css);
+ res_counter_uncharge(&mem->res, PAGE_SIZE);
+ list_del_init(&pc->lru);
+ kfree(pc);
+ } else
+ count = 1; /* being uncharged ? ...do relax */
+
+ if (--count == 0) {
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+ cond_resched();
+ spin_lock_irqsave(&mem->lru_lock, flags);
+ count = SWAP_CLUSTER_MAX;
+ }
+ }
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+}
+
+/*
+ *
+ *
+ */
+
+int mem_cgroup_force_empty(struct mem_cgroup *mem)
+{
+ int ret = -EBUSY;
+ css_get(&mem->css);
+ while (!list_empty(&mem->active_list) ||
+ !list_empty(&mem->inactive_list)) {
+ if (atomic_read(&mem->css.cgroup->count) > 0)
+ goto out;
+ /* drop all page_cgroup in active_list */
+ mem_cgroup_force_empty_list(mem, &mem->active_list);

```

```

+ /* drop all page_cgroup in inactive_list */
+ mem_cgroup_force_empty_list(mem, &mem->inactive_list);
+ }
+ ret = 0;
+out:
+ css_put(&mem->css);
+ return ret;
+}
+
+
+
+ int mem_cgroup_write_strategy(char *buf, unsigned long long *tmp)
+ {
+     *tmp = memparse(buf, &buf);
@@ -617,6 +675,31 @@ static ssize_t mem_control_type_read(str
+     ppos, buf, s - buf);
+ }
+
+
+static ssize_t mem_force_empty_write(struct cgroup *cont,
+ struct cftype *cft, struct file *file,
+ const char __user *userbuf,
+ size_t nbytes, loff_t *ppos)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ int ret;
+ ret = mem_cgroup_force_empty(mem);
+ if (!ret)
+     ret = nbytes;
+ return ret;
+}
+
+static ssize_t mem_force_empty_read(struct cgroup *cont,
+ struct cftype *cft,
+ struct file *file, char __user *userbuf,
+ size_t nbytes, loff_t *ppos)
+{
+ char buf[2] = "0";
+ return simple_read_from_buffer((void __user *)userbuf, nbytes,
+ ppos, buf, strlen(buf));
+}
+
+
+
+ static struct cftype mem_cgroup_files[] = {
+     {
+         .name = "usage_in_bytes",
@@ -639,6 +722,11 @@ static struct cftype mem_cgroup_files[]
+         .write = mem_control_type_write,

```

```

    .read = mem_control_type_read,
    },
+ {
+ .name = "force_empty",
+ .write = mem_force_empty_write,
+ .read = mem_force_empty_read,
+ },
};

static struct mem_cgroup init_mem_cgroup;

```

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

Subject: [PATCH][just for review] memory controller enhancements [2/5] remember page cache
Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:51:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Add PCGF_PAGECACHE flag to page_cgroup to remember "this page is charged as page-cache."
This is very useful for implementing precise accounting in memory cgroup.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

```

mm/memcontrol.c | 18 ++++++
1 file changed, 15 insertions(+), 3 deletions(-)

```

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

```

=====
--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -83,6 +83,8 @@ struct page_cgroup {
    struct mem_cgroup *mem_cgroup;
    atomic_t ref_cnt; /* Helpful when pages move b/w */
    /* mapped and cached states */
+ int flags;
+#define PCGF_PAGECACHE (0x1) /* charged as page-cache */
};

enum {
@@ -305,8 +307,8 @@ unsigned long mem_cgroup_isolate_pages(u
 * 0 if the charge was successful

```

```

* < 0 if the cgroup is over its limit
*/
-int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
-   gfp_t gfp_mask)
+static int mem_cgroup_charge_common(struct page *page, struct mm_struct *mm,
+   gfp_t gfp_mask, int is_cache)
{
    struct mem_cgroup *mem;
    struct page_cgroup *pc;
@@ -406,6 +408,10 @@ noreclaim:
    atomic_set(&pc->ref_cnt, 1);
    pc->mem_cgroup = mem;
    pc->page = page;
+ if (is_cache)
+ pc->flags = PCGF_PAGECACHE;
+ else
+ pc->flags = 0;
    if (page_cgroup_assign_new_page_cgroup(page, pc)) {
/*
    * an another charge is added to this page already.
@@ -431,6 +437,12 @@ err:
    return -ENOMEM;
}

+int mem_cgroup_charge(struct page *page, struct mm_struct *mm,
+   gfp_t gfp_mask)
+{
+ return mem_cgroup_charge_common(page, mm, gfp_mask, 0);
+}
+
/*
* See if the cached pages should be charged at all?
*/
@@ -443,7 +455,7 @@ int mem_cgroup_cache_charge(struct page

    mem = rcu_dereference(mm->mem_cgroup);
    if (mem->control_type == MEM_CGROUP_TYPE_ALL)
- return mem_cgroup_charge(page, mm, gfp_mask);
+ return mem_cgroup_charge_common(page, mm, gfp_mask, 1);
    else
        return 0;
}

```

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

Subject: [PATCH][just for review] memory controller enhancements [3/5] remember on which list patch

Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:52:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

Remember page_cgroup is on active_list or not in page_cgroup->flags.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

mm/memcontrol.c | 12 ++++++-----
1 file changed, 8 insertions(+), 4 deletions(-)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

```
=====
--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -85,6 +85,7 @@ struct page_cgroup {
    /* mapped and cached states */
    int flags;
    #define PCGF_PAGECACHE (0x1) /* charged as page-cache */
    #define PCGF_ACTIVE (0x2) /* this is on cgroup's active list */
};

enum {
@@ -208,10 +209,13 @@ clear_page_cgroup(struct page *page, str

static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
{
- if (active)
+ if (active) {
+ pc->flags |= PCGF_ACTIVE;
    list_move(&pc->lru, &pc->mem_cgroup->active_list);
- else
+ } else {
+ pc->flags &= ~PCGF_ACTIVE;
    list_move(&pc->lru, &pc->mem_cgroup->inactive_list);
+ }
}

/*
@@ -409,9 +413,9 @@ noreclaim:
    pc->mem_cgroup = mem;
    pc->page = page;
    if (is_cache)
- pc->flags = PCGF_PAGECACHE;
+ pc->flags = PCGF_PAGECACHE | PCGF_ACTIVE;
    else
- pc->flags = 0;
```

```
+ pc->flags = PCGF_ACTIVE;
  if (page_cgroup_assign_new_page_cgroup(page, pc)) {
/*
 * an another charge is added to this page already.
```

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

Subject: [PATCH][just for review] memory controller enhancements [4/5] statistics for memory cgroup
Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:54:22 GMT
[View Forum Message](#) <> [Reply to Message](#)

Add statistics account infrastructure for memory controller.

Changes from original:

- divided into 2 patch (this one and show info)
- changed from u64 to s64
- added mem_cgroup_stat_add() and batched statistics modification logic.
- removed stat init code because mem_cgroup is allocated by kzalloc().

Problem:

- charge/uncharge count can be overflow. But they are unnecessary ?

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

mm/memcontrol.c | 120

+++++
1 file changed, 116 insertions(+), 4 deletions(-)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

=====

--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c

+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c

@@ -35,6 +35,63 @@ struct cgroup_subsys mem_cgroup_subsys;
static const int MEM_CGROUP_RECLAIM_RETRIES = 5;

/*

+ * Statistics for memory cgroup.

+ */

+enum mem_cgroup_stat_index {

+ /*

+ * For MEM_CONTAINER_TYPE_ALL, usage = pagecache + rss.

+ */


```

+ MEM_CGROUP_STAT_PAGECACHE, /* # of pages charged as cache */
+ MEM_CGROUP_STAT_RSS, /* # of pages charged as rss */
+
+ /*
+ * usage = charge - uncharge.
+ */
+ MEM_CGROUP_STAT_CHARGE, /* # of pages charged */
+ MEM_CGROUP_STAT_UNCHARGE, /* # of pages uncharged */
+
+ MEM_CGROUP_STAT_ACTIVE, /* # of pages in active list */
+ MEM_CGROUP_STAT_INACTIVE, /* # of pages on inactive list */
+
+ MEM_CGROUP_STAT_NSTATS,
+};
+
+struct mem_cgroup_stat_cpu {
+ s64 count[MEM_CGROUP_STAT_NSTATS];
+} ____cacheline_aligned_in_smp;
+
+struct mem_cgroup_stat {
+ struct mem_cgroup_stat_cpu cpustat[NR_CPUS];
+};
+
+/*
+ * For batching....mem_cgroup_charge_statistics()(see below).
+ */
+static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
+ enum mem_cgroup_stat_index idx, int val)
+{
+ int cpu = smp_processor_id();
+ stat->cpustat[cpu].count[idx] += val;
+}
+
+static inline void mem_cgroup_stat_inc(struct mem_cgroup_stat *stat,
+ enum mem_cgroup_stat_index idx)
+{
+ preempt_disable();
+ mem_cgroup_stat_add(stat, idx, 1);
+ preempt_enable();
+}
+
+static inline void mem_cgroup_stat_dec(struct mem_cgroup_stat *stat,
+ enum mem_cgroup_stat_index idx)
+{
+ preempt_disable();
+ mem_cgroup_stat_add(stat, idx, -1);
+ preempt_enable();
+}

```

```

+
+
+/*
+ * The memory controller data structure. The memory controller controls both
+ * page cache and RSS per cgroup. We would eventually like to provide
+ * statistics based on the statistics developed by Rik Van Riel for clock-pro,
@@ -63,6 +120,10 @@ struct mem_cgroup {
+ */
+ spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
+ /*
+ * statistics.
+ */
+ struct mem_cgroup_stat stat;
+ };

+ /*
@@ -96,6 +157,37 @@ enum {
+ MEM_CGROUP_TYPE_MAX,
+ };

+ /*
+ * Batched statistics modification.
+ * We have to modify several values at charge/uncharge..
+ */
+ static inline void
+ mem_cgroup_charge_statistics(struct mem_cgroup *mem, int flags, bool charge)
+ {
+ int val = (charge)? 1 : -1;
+ preempt_disable();
+
+ if (flags & PCGF_PAGECACHE)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_PAGECACHE, val);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_RSS, val);
+
+ if (flags & PCGF_ACTIVE)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_ACTIVE, val);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_INACTIVE, val);
+
+ if (charge)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_CHARGE, 1);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_UNCHARGE, 1);
+
+ preempt_enable();
+ }

```

```

+
+
+
+
static struct mem_cgroup init_mem_cgroup;

static inline
@@ -207,8 +299,23 @@ clear_page_cgroup(struct page *page, str
}

-static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
+static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active,
+ struct mem_cgroup *mem)
{
+ int moved = 0;
+
+ if (active && (pc->flags & PCGF_ACTIVE) == 0)
+ moved = 1; /* Move from inactive to active */
+ else if (!active && (pc->flags & PCGF_ACTIVE))
+ moved = -1; /* Move from active to inactive */
+
+ if (moved) {
+ struct mem_cgroup_stat *stat = &mem->stat;
+ preempt_disable();
+ mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_ACTIVE, moved);
+ mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_INACTIVE, -moved);
+ preempt_enable();
+ }
+ if (active) {
+ pc->flags |= PCGF_ACTIVE;
+ list_move(&pc->lru, &pc->mem_cgroup->active_list);
@@ -230,7 +337,7 @@ void mem_cgroup_move_lists(struct page_c
mem = pc->mem_cgroup;

spin_lock(&mem->lru_lock);
- __mem_cgroup_move_lists(pc, active);
+ __mem_cgroup_move_lists(pc, active, mem);
spin_unlock(&mem->lru_lock);
}

@@ -267,12 +374,12 @@ unsigned long mem_cgroup_isolate_pages(u
}

if (PageActive(page) && !active) {
- __mem_cgroup_move_lists(pc, true);
+ __mem_cgroup_move_lists(pc, true, pc->mem_cgroup);
scan--;
}

```

```

    continue;
}
if (!PageActive(page) && active) {
- __mem_cgroup_move_lists(pc, false);
+ __mem_cgroup_move_lists(pc, false, pc->mem_cgroup);
    scan--;
    continue;
}
@@ -428,6 +535,9 @@ noreclaim:
    goto retry;
}

+ /* Update statistics vector */
+ mem_cgroup_charge_statistics(mem, pc->flags, true);
+
    spin_lock_irqsave(&mem->lru_lock, flags);
    list_add(&pc->lru, &mem->active_list);
    spin_unlock_irqrestore(&mem->lru_lock, flags);
@@ -494,6 +604,7 @@ void mem_cgroup_uncharge(struct page_cgr
    spin_lock_irqsave(&mem->lru_lock, flags);
    list_del_init(&pc->lru);
    spin_unlock_irqrestore(&mem->lru_lock, flags);
+ mem_cgroup_charge_statistics(mem, pc->flags, false);
    kfree(pc);
}
}
@@ -566,6 +677,7 @@ mem_cgroup_force_empty_list(struct mem_c
    css_put(&mem->css);
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    list_del_init(&pc->lru);
+ mem_cgroup_charge_statistics(mem, pc->flags, false);
    kfree(pc);
} else
    count = 1; /* being uncharged ? ...do relax */

```

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

Subject: [PATCH][just for review] memory controller enhancements [5/5]
memory.stat file
Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:54:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

Show accounted information of memory cgroup by memory.stat file

Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>
Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

mm/memcontrol.c | 54 ++
1 file changed, 54 insertions(+)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

=====

--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c

+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c

@@ -28,6 +28,7 @@

#include <linux/swap.h>

#include <linux/spinlock.h>

#include <linux/fs.h>

+#include <linux/seq_file.h>

#include <asm/uaccess.h>

@@ -828,6 +829,55 @@ static ssize_t mem_force_empty_read(stru
{

+static const struct mem_cgroup_stat_desc {

+ const char *msg;

+ u64 unit;

+} mem_cgroup_stat_desc[] = {

+ [MEM_CGROUP_STAT_PAGECACHE] = { "page_cache", PAGE_SIZE, },

+ [MEM_CGROUP_STAT_RSS] = { "rss", PAGE_SIZE, },

+ [MEM_CGROUP_STAT_CHARGE] = { "charge", PAGE_SIZE, },

+ [MEM_CGROUP_STAT_UNCHARGE] = { "uncharge", PAGE_SIZE, },

+ [MEM_CGROUP_STAT_ACTIVE] = { "active", PAGE_SIZE, },

+ [MEM_CGROUP_STAT_INACTIVE] = { "inactive", PAGE_SIZE, },

+};

+

+static int mem_control_stat_show(struct seq_file *m, void *arg)

+{

+ struct cgroup *cont = m->private;

+ struct mem_cgroup *mem_cont = mem_cgroup_from_cont(cont);

+ struct mem_cgroup_stat *stat = &mem_cont->stat;

+ int i;

+

+ for (i = 0; i < ARRAY_SIZE(stat->cpustat[0].count); i++) {

+ unsigned int cpu;

+ s64 val;

+

+ val = 0;

+ for (cpu = 0; cpu < NR_CPUS; cpu++)

+ val += stat->cpustat[cpu].count[i];

```

+ val *= mem_cgroup_stat_desc[i].unit;
+ seq_printf(m, "%s %lld\n", mem_cgroup_stat_desc[i].msg, val);
+ }
+ return 0;
+}
+
+static const struct file_operations mem_control_stat_file_operations = {
+ .read = seq_read,
+ .llseek = seq_lseek,
+ .release = single_release,
+};
+
+static int mem_control_stat_open(struct inode *unused, struct file *file)
+{
+ /* XXX __d_cont */
+ struct cgroup *cont = file->f_dentry->d_parent->d_fsdata;
+
+ file->f_op = &mem_control_stat_file_operations;
+ return single_open(file, mem_control_stat_show, cont);
+}
+
+
+
+static struct cftype mem_cgroup_files[] = {
+ {
+ .name = "usage_in_bytes",
@@ -855,6 +905,10 @@ static struct cftype mem_cgroup_files[]
+ .write = mem_force_empty_write,
+ .read = mem_force_empty_read,
+ },
+ {
+ .name = "stat",
+ .open = mem_control_stat_open,
+ },
+ };

static struct mem_cgroup init_mem_cgroup;

```

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

Subject: Re: [PATCH][just for review] memory controller enhancements [4/5]
statistics for memory cgroup
Posted by [yamamoto](#) on Mon, 15 Oct 2007 06:37:01 GMT

> - changed from u64 to s64

why?

```
> +/*
> + * For batching....mem_cgroup_charge_statistics()(see below).
> + */
> +static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
> +      enum mem_cgroup_stat_index idx, int val)
> +{
> + int cpu = smp_processor_id();
> + stat->cpustat[cpu].count[idx] += val;
> +}
```

i think the function name should be something which implies batching.

```
> @@ -207,8 +299,23 @@ clear_page_cgroup(struct page *page, str
> }
>
>
> -static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
> +static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active,
> +      struct mem_cgroup *mem)
> {
```

can mem be different from pc->mem_cgroup here?

YAMAMOTO Takashi

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

Subject: Re: [PATCH][just for review] memory controller enhancements [4/5]
statistics for memory cgroup

Posted by [KAMEZAWA Hiroyuki](#) on Mon, 15 Oct 2007 06:46:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon, 15 Oct 2007 15:37:01 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

```
> > - changed from u64 to s64
>
> why?
>
```

```
>> +/*
>> + * For batching....mem_cgroup_charge_statistics()(see below).
>> + */
>> +static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
>> +      enum mem_cgroup_stat_index idx, int val)
>> +{
>> + int cpu = smp_processor_id();
>> + stat->cpustat[cpu].count[idx] += val;
>> +}
>
> i think the function name should be something which implies batching.
>
Hm, How about this ?
==
mem_cgroup_stat_add_atomic()
==
and add this
==
VM_BUG_ON(preempt_count() == 0)
==

>> @@ -207,8 +299,23 @@ clear_page_cgroup(struct page *page, str
>> }
>>
>>
>> -static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
>> +static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active,
>> +      struct mem_cgroup *mem)
>> {
>
> can mem be different from pc->mem_cgroup here?
>
```

Ah, always pc->mem_cgroup. ok, I'll remove that.

Thanks,
-Kame

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

Subject: Re: [PATCH][just for review] memory controller enhancements [4/5]
statistics for memory cgroup
Posted by [yamamoto](#) on Mon, 15 Oct 2007 22:38:23 GMT

```
> > > +/*
> > > + * For batching....mem_cgroup_charge_statistics()(see below).
> > > + */
> > > +static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
> > > +      enum mem_cgroup_stat_index idx, int val)
> > > +{
> > > + int cpu = smp_processor_id();
> > > + stat->cpustat[cpu].count[idx] += val;
> > > +}
> >
> > i think the function name should be something which implies batching.
> >
> Hm, How about this ?
> ==
> mem_cgroup_stat_add_atomic()
> ==
> and add this
> ==
> VM_BUG_ON(preempt_count() == 0)
> ==
```

_atomic sounds like a different thing to me. _nonpreemptible?

YAMAMOTO Takashi

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

Subject: Re: [PATCH][just for review] memory controller enhancements [1/5] force empty

Posted by [yamamoto](#) on Tue, 16 Oct 2007 00:15:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

```
> Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c
> =====
> --- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
> +++ devel-2.6.23-rc8-mm2/mm/memcontrol.c
> @@ -469,6 +469,7 @@ void mem_cgroup_uncharge(struct page_cgr
> page = pc->page;
> /*
> * get page->cgroup and clear it under lock.
> + * force-empty can drop page->cgroup without checking refcnt.
```

force_empty

```
> + char buf[2] = "0";
```

it should be static const unless you want a runtime assignment.

YAMAMOTO Takashi

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: Re: [PATCH][just for review] memory controller enhancements [4/5]
statistics for memory cgroup

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 16 Oct 2007 03:03:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Tue, 16 Oct 2007 07:38:23 +0900 (JST)

yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

```
>>>> +/*
>>>> + * For batching....mem_cgroup_charge_statistics()(see below).
>>>> + */
>>>> +static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
>>>> +          enum mem_cgroup_stat_index idx, int val)
>>>> +{
>>>> + int cpu = smp_processor_id();
>>>> + stat->cpustat[cpu].count[idx] += val;
>>>> +}
>>>
>>> i think the function name should be something which implies batching.
>>>
>> Hm, How about this ?
>> ==
>> mem_cgroup_stat_add_atomic()
>> ==
>> and add this
>> ==
>> VM_BUG_ON(preempt_count() == 0)
>> ==
>
> _atomic sounds like a different thing to me. _nonpreemptible?
>
> Hmm, ok.
```

Thanks,
-Kame

Subject: Re: [PATCH][just for review] memory controller enhancements [4/5]
statistics for memory cgroup
Posted by [Balbir Singh](#) on Tue, 16 Oct 2007 03:55:43 GMT
[View Forum Message](#) <> [Reply to Message](#)

KAMEZAWA Hiroyuki wrote:

> On Tue, 16 Oct 2007 07:38:23 +0900 (JST)

> yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

>

>>>>> +/*

>>>>> + * For batching....mem_cgroup_charge_statistics()(see below).

>>>>> + */

>>>>> +static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,

>>>>> + enum mem_cgroup_stat_index idx, int val)

>>>>> +{

>>>>> + int cpu = smp_processor_id();

>>>>> + stat->cpustat[cpu].count[idx] += val;

>>>>> +}

>>>> i think the function name should be something which implies batching.

>>>>

>>> Hm, How about this ?

>>> ==

>>> mem_cgroup_stat_add_atomic()

>>> ==

>>> and add this

>>> ==

>>> VM_BUG_ON(preempt_count() == 0)

>>> ==

>> _atomic sounds like a different thing to me. _nonpreemptible?

>>

> Hmm, ok.

>

> Thanks,

> -Kame

>

How about we call it __mem_cgroup_add_stat() and add a comment stating that the routine should be called with pre-emption disabled?

We can also add the VM_BUG_ON() suggested.

--

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][just for review] memory controller enhancements [4/5]
statistics for memory cgroup
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 16 Oct 2007 04:33:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 16 Oct 2007 09:25:43 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> How about we call it `__mem_cgroup_add_stat()` and add a comment stating
> that the routine should be called with pre-emption disabled?

>
> We can also add the `VM_BUG_ON()` suggested.

>
Ah, nice suggestion.
I'll think again.

Thanks,
-Kame

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

Subject: Re: [PATCH][just for review] memory controller enhancements [1/5] force
empty
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 16 Oct 2007 06:32:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 16 Oct 2007 09:15:49 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

> > Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c
> > =====
> > --- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c
> > +++ devel-2.6.23-rc8-mm2/mm/memcontrol.c

```
> > @@ -469,6 +469,7 @@ void mem_cgroup_uncharge(struct page_cgr
> > page = pc->page;
> > /*
> > * get page->cgroup and clear it under lock.
> > + * force-empty can drop page->cgroup without checking refcnt.
>
> force_empty
>
> + char buf[2] = "0";
>
> it should be static const unless you want a runtime assignment.
>
> Sure, thank you for pointing out.
```

Regards,
-Kame

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