
Subject: [RFC][for -mm] memory cgroup enhancements take3 [0/9] intro
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:12:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi, this is updated set of enhancements for memory cgroup in my box.
I'd like to post some of these when the next -mm is shipped.
(but rebase and more test should be done.)

Any comments are welcome.

Changes from previous sets:

- added comments/explanation.
- dropped failcnt patch
- modified per-node numastat to zonestat
- and small changes for good looking.

Thanks,
-Kame

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa handl
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:14:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

Because NODE_DATA(node)->node_zonelists[] is guaranteed to contain
all necessary zones, it is not necessary to use for_each_online_node.

And this for_each_online_node() makes reclaim routine start always
from node 0. This is bad. This patch will make reclaim code start
from caller's node.

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

mm/vmscan.c | 8 +++-----
1 file changed, 3 insertions(+), 5 deletions(-)

Index: devel-2.6.23-mm1/mm/vmscan.c

=====

--- devel-2.6.23-mm1.orig/mm/vmscan.c
+++ devel-2.6.23-mm1/mm/vmscan.c

```

@@ -1375,15 +1375,13 @@ unsigned long try_to_free_mem_cgroup_pag
    .mem_cgroup = mem_cont,
    .isolate_pages = mem_cgroup_isolate_pages,
};
- int node;
+ int node = numa_node_id();
    struct zone **zones;
    int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);

- for_each_online_node(node) {
-     zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
-     if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
+ zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
+ if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
    return 1;
- }
    return 0;
}
#endif

```

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [2/9] add
force_empty

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:15:49 GMT

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

no changes from previous post.

==

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 v4 -> v5:

- added comments to mem_cgroup_force_empty()
- made mem_force_empty_read return -EINVAL

- cleanup mem_cgroup_force_empty_list()
- removed SWAP_CLUSTER_MAX

Changelog v3 -> v4:

- adjusted to 2.6.23-mm1
- fixed typo
- changes buf[2]="0" to static const

Changelog v2 -> v3:

- changed the name from force_reclaim to force_empty.

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 | 110 ++++++-----
1 file changed, 103 insertions(+), 7 deletions(-)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -480,6 +480,7 @@ 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;

@@ -489,13 +490,6 @@ 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();

}

}

}

```

@@ -543,6 +537,76 @@ 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.
+ */
+#define FORCE_UNCHARGE_BATCH (128)
+static void
+mem_cgroup_force_empty_list(struct mem_cgroup *mem, struct list_head *list)
+{
+ struct page_cgroup *pc;
+ struct page *page;
+ int count;
+ unsigned long flags;
+
+retry:
+ count = FORCE_UNCHARGE_BATCH;
+ spin_lock_irqsave(&mem->lru_lock, flags);
+
+ while (--count && !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 /* being uncharged ? ...do relax */
+   break;
+ }
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+ if (!list_empty(list)) {
+  cond_resched();
+  goto retry;
+ }
+ return;
+}
+
+/*
+ * make mem_cgroup's charge to be 0 if there is no task.
+ * This enables deleting this mem_cgroup.
+ */
+

```

```

+int mem_cgroup_force_empty(struct mem_cgroup *mem)
+{
+ int ret = -EBUSY;
+ css_get(&mem->css);
+ /*
+  * page reclaim code (kswapd etc..) will move pages between
+  * active_list <-> inactive_list while we don't take a lock.
+  * So, we have to do loop here until all lists are empty.
+  */
+ 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);
+@@ -628,6 +692,33 @@ 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;
+}
+
+
+/*

```

+ * Note: This should be removed if cgroup supports write-only file.

+ */

+

+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)

+{

+ return -EINVAL;

+}

+

+

static struct cftype mem_cgroup_files[] = {

{

.name = "usage_in_bytes",

@@ -650,6 +741,11 @@ 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: [RFC][for -mm] memory cgroup enhancements take3 [3/9] remember page cache

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:16:55 GMT

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

Add flag to page_cgroup to remember "this page is charged as page-cache."

This is very useful for implementing precise accounting in memory cgroup.

Changelog v2 -> v3

- added enum for mem_cgroup_charge_type_common(...charge_type)

- renamed #define PCGF_XXX_XXX to PAGE_CGROUP_FLAG_XXX

Changelog v1 -> v2

- moved #define to out-side of struct definition

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

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

mm/memcontrol.c | 24 ++++++++-----

1 file changed, 21 insertions(+), 3 deletions(-)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -83,7 +83,9 @@ 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 PAGE_CGROUP_FLAG_CACHE (0x1) /* charged as cache */

enum {

MEM_CGROUP_TYPE_UNSPEC = 0,

@@ -93,6 +95,11 @@ enum {

MEM_CGROUP_TYPE_MAX,

};

+ enum charge_type {

+ MEM_CGROUP_CHARGE_TYPE_CACHE = 0,

+ MEM_CGROUP_CHARGE_TYPE_MAPPED = 0,

+};

+

static struct mem_cgroup init_mem_cgroup;

static inline

@@ -315,8 +322,8 @@ 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, enum charge_type ctype)

{

struct mem_cgroup *mem;

struct page_cgroup *pc;

@@ -418,6 +425,9 @@ noreclaim:

atomic_set(&pc->ref_cnt, 1);

pc->mem_cgroup = mem;

```

pc->page = page;
+ pc->flags = 0;
+ if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
+ pc->flags |= PAGE_CGROUP_FLAG_CACHE;
  if (page_cgroup_assign_new_page_cgroup(page, pc)) {
/*
 * an another charge is added to this page already.
@@ -442,6 +452,13 @@ 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,
+ MEM_CGROUP_CHARGE_TYPE_MAPPED);
+}
+
/*
 * See if the cached pages should be charged at all?
 */
@@ -454,7 +471,8 @@ 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,
+ MEM_CGROUP_CHARGE_TYPE_CACHE);
    else
    return 0;
  }

```

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [4/9] remember active
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:17:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

Against 2.6.23-mm1.

Changelov v2->v3

- renamed #define PCGF_ACTIVE to PAGE_CGROUP_FLAG_ACTIVE.

Changelog v1->v2

- moved #define to out-side of struct definition

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

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

mm/memcontrol.c | 10 ++++++---

1 file changed, 7 insertions(+), 3 deletions(-)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -86,6 +86,7 @@ struct page_cgroup {

int flags;

};

#define PAGE_CGROUP_FLAG_CACHE (0x1) /* charged as cache */

+#define PAGE_CGROUP_FLAG_ACTIVE (0x2) /* page is active in this cgroup */

enum {

MEM_CGROUP_TYPE_UNSPEC = 0,

@@ -213,10 +214,13 @@ 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 |= PAGE_CGROUP_FLAG_ACTIVE;

list_move(&pc->lru, &pc->mem_cgroup->active_list);

- else

+ } else {

+ pc->flags &= ~PAGE_CGROUP_FLAG_ACTIVE;

list_move(&pc->lru, &pc->mem_cgroup->inactive_list);

+ }

}

int task_in_mem_cgroup(struct task_struct *task, const struct mem_cgroup *mem)

@@ -425,7 +429,7 @@ noreclaim:

atomic_set(&pc->ref_cnt, 1);

pc->mem_cgroup = mem;

pc->page = page;

- pc->flags = 0;

+ pc->flags = PAGE_CGROUP_FLAG_ACTIVE;

if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)

pc->flags |= PAGE_CGROUP_FLAG_CACHE;

if (page_cgroup_assign_new_page_cgroup(page, pc)) {

Subject: [RFC][for -mm] memory cgroup enhancements take3 [5/9] per-cpu status accounting

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:18:33 GMT

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

Add statistics account infrastructure for memory controller.

Changelov v2 -> v3

- adjusted to rename of #define PAGE_CGROUP_FLAG....
- dropped ACTIVE/INACTIVE counter.
They should be accounted against per zone. Then, using pcp counter, we need array of NR_CPU * MAX_NUMNODES * NR_ZONES against all stats. This is too big for statistics *per-memory-cgroup*.
ACTIVE/INACTIVE counter is added as per-zone statistics later.

Changelog v1 -> v2

- Removed Charge/Uncharge counter
- reflected comments.
 - changes __move_lists() args.
 - changes __mem_cgroup_stat_add() name, comment and added VM_BUGON

Changes from original:

- divided into 2 patch (account 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().

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

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

mm/memcontrol.c | 83

++++
1 file changed, 83 insertions(+)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -35,6 +35,58 @@ 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 */
+
+ 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).
+ * MUST be called under preempt_disable().
+ */
+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();
+ #ifdef CONFIG_PREEMPT
+ VM_BUG_ON(preempt_count() == 0);
+ #endif
+ 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 +115,10 @@ struct mem_cgroup {
+ */
+ spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
+ /*
+ * statistics.
+ */
+ struct mem_cgroup_stat stat;
+ };

+ /*
@@ -101,6 +157,28 @@ enum charge_type {
+ MEM_CGROUP_CHARGE_TYPE_MAPPED = 0,
+ };

+ /*
+ * 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, int charge)
+ {
+ int val = (charge)? 1 : -1;
+ struct mem_cgroup_stat *stat = &mem->stat;
+ preempt_disable();
+
+ if (flags & PAGE_CGROUP_FLAG_CACHE)
+ __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_PAGECACHE, val);
+ else
+ __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_RSS, val);
+
+ preempt_enable();
+ }
+
+
+
+
+ static struct mem_cgroup init_mem_cgroup;

+ static inline

```

```

@@ -444,6 +522,9 @@ 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);
@@ -511,6 +592,7 @@ 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);
}
}
@@ -586,6 +668,7 @@ retry:
    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 /* being uncharged ? ...do relax */
    break;

```

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [7/9] add
memory.stat file
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:19:44 GMT
[View Forum Message](#) <> [Reply to Message](#)

Show accounted information of memory cgroup by memory.stat file

Changelog v1->v2
- dropped Charge/Uncharge entry.

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

mm/memcontrol.c | 50 +++++
1 file changed, 50 insertions(+)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/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>

@@ -824,6 +825,51 @@ 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, },

+};

+

+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",
@@ -851,6 +897,10 @@ 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: [RFC][for -mm] memory cgroup enhancements take3 [7/9] add
pre_destroy
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:20:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

My main purpose of this patch is for memory controller..

This patch adds a handler "pre_destroy" to cgroup_subsys.
It is called before cgroup_rmdir() checks all subsys's refcnt.

I think this is useful for subsyses which have some extra refs
even if there are no tasks in cgroup. By adding pre_destroy(),
the kernel keeps the rule "destroy() against subsystem is called only

when refcnt=0." and allows css's ref to be used by other objects than tasks.

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

include/linux/cgroup.h | 1 +
kernel/cgroup.c | 7 +++++++
2 files changed, 8 insertions(+)

Index: devel-2.6.23-mm1/include/linux/cgroup.h

```
=====
--- devel-2.6.23-mm1.orig/include/linux/cgroup.h
+++ devel-2.6.23-mm1/include/linux/cgroup.h
@@ -233,6 +233,7 @@ int cgroup_is_descendant(const struct cg
struct cgroup_subsys {
    struct cgroup_subsys_state *(*create)(struct cgroup_subsys *ss,
        struct cgroup *cont);
+ void (*pre_destroy)(struct cgroup_subsys *ss, struct cgroup *cont);
    void (*destroy)(struct cgroup_subsys *ss, struct cgroup *cont);
    int (*can_attach)(struct cgroup_subsys *ss,
        struct cgroup *cont, struct task_struct *tsk);
```

Index: devel-2.6.23-mm1/kernel/cgroup.c

```
=====
--- devel-2.6.23-mm1.orig/kernel/cgroup.c
+++ devel-2.6.23-mm1/kernel/cgroup.c
@@ -2158,6 +2158,13 @@ static int cgroup_rmdir(struct inode *un
    parent = cont->parent;
    root = cont->root;
    sb = root->sb;
+ /*
+  * Notify subsys that rmdir() request comes.
+  */
+ for_each_subsys(root, ss) {
+ if ((cont->subsys[ss->subsys_id]) && ss->pre_destroy)
+ ss->pre_destroy(ss, cont);
+ }

    if (cgroup_has_css_refs(cont)) {
        mutex_unlock(&cgroup_mutex);
```

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [8/9] implicit

force_empty at rmdir

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:21:20 GMT

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

This patch adds pre_destroy handler for mem_cgroup and try to make mem_cgroup empty at rmdir().

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

mm/memcontrol.c | 8 +++++++
1 file changed, 8 insertions(+)

Index: devel-2.6.23-mm1/mm/memcontrol.c

```
=====
--- devel-2.6.23-mm1.orig/mm/memcontrol.c
+++ devel-2.6.23-mm1/mm/memcontrol.c
@@ -927,6 +927,13 @@ mem_cgroup_create(struct cgroup_subsys *
    return &mem->css;
}

+static void mem_cgroup_pre_destroy(struct cgroup_subsys *ss,
+    struct cgroup *cont)
+{
+    struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+    mem_cgroup_force_empty(mem);
+}
+
static void mem_cgroup_destroy(struct cgroup_subsys *ss,
    struct cgroup *cont)
{
@@ -978,6 +985,7 @@ struct cgroup_subsys mem_cgroup_subsys =
    .name = "memory",
    .subsys_id = mem_cgroup_subsys_id,
    .create = mem_cgroup_create,
+    .pre_destroy = mem_cgroup_pre_destroy,
    .destroy = mem_cgroup_destroy,
    .populate = mem_cgroup_populate,
    .attach = mem_cgroup_move_task,
```

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: [RFC][for -mm] memory cgroup enhancements take3 [9/9] per zone stat

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:23:32 GMT

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

Add per-zone x per-cpu counter for accounting # of active/inactive
This array can be big if MAX_NUMNODE is very large, so we need
some cares.

This "active/inactive" information should be maintained per zone
(can be used for page reclaim code later, I think).

Memory cgroup total active/inactive information is shown in memory.stat
file.

This patch changes early_init from 1 to 0 for using kmalloc/vmalloc at boot.

Changelog v1 -> v2:

- changed from per-node to per-zone.
- just count active/inactive

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

mm/memcontrol.c | 171

+++++
1 file changed, 165 insertions(+), 6 deletions(-)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -29,6 +29,7 @@

#include <linux/spinlock.h>

#include <linux/fs.h>

#include <linux/seq_file.h>

+#include <linux/vmalloc.h>

#include <asm/uaccess.h>

@@ -56,6 +57,31 @@ struct mem_cgroup_stat {
 struct mem_cgroup_stat_cpu cpustat[NR_CPUS];
};

+
+/*
+ * Per-zone statistics.
+ * Please be carefull. The array can be very big on environments which has
+ * very big MAX_NUMNODES . Adding new stat member to this will eat much memory.
+ * Only Active/Inactive may be suitable.
+ */
+enum mem_cgroup_zonestat_index {
+ MEM_CGROUP_ZONESTAT_ACTIVE,
+ MEM_CGROUP_ZONESTAT_INACTIVE,


```

+ int index = page_cgroup_to_zonestat_index(pc);
  preempt_disable();

  if (flags & PAGE_CGROUP_FLAG_CACHE)
@@ -174,6 +243,13 @@ mem_cgroup_charge_statistics(struct mem_
    else
      __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_RSS, val);

+ if (flags & PAGE_CGROUP_FLAG_ACTIVE)
+ __mem_cgroup_zonestat_add(zonestat, MEM_CGROUP_ZONESTAT_ACTIVE,
+   val, index);
+ else
+ __mem_cgroup_zonestat_add(zonestat,
+   MEM_CGROUP_ZONESTAT_INACTIVE,
+   val, index);
  preempt_enable();
}

@@ -293,6 +369,23 @@ clear_page_cgroup(struct page *page, str

static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
{
+ int direction = 0;
+
+ if (active && !(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
+   direction = 1; /*from inactive to active */
+ if (!active && (pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
+   direction = -1;
+
+ if (direction) {
+   struct mem_cgroup_zonestat *zstat = &pc->mem_cgroup->zonestat;
+   int index = page_cgroup_to_zonestat_index(pc);
+   preempt_disable();
+   __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_ACTIVE,
+     direction, index);
+   __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_INACTIVE,
+     direction, index);
+ }
+
  if (active) {
    pc->flags |= PAGE_CGROUP_FLAG_ACTIVE;
    list_move(&pc->lru, &pc->mem_cgroup->active_list);
@@ -509,6 +602,8 @@ noreclaim:
  pc->mem_cgroup = mem;
  pc->page = page;
  pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
+ pc->nid = page_to_nid(page);
+ pc->zone_id = page_zonenum(page);

```

```

if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
    pc->flags |= PAGE_CGROUP_FLAG_CACHE;
if (page_cgroup_assign_new_page_cgroup(page, pc)) {
@@ -524,7 +619,7 @@ noreclaim:
}

/* Update statistics vector */
- mem_cgroup_charge_statistics(mem, pc->flags, true);
+ mem_cgroup_charge_statistics(pc, true);

spin_lock_irqsave(&mem->lru_lock, flags);
list_add(&pc->lru, &mem->active_list);
@@ -591,9 +686,10 @@ void mem_cgroup_uncharge(struct page_cgr
    css_put(&mem->css);
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    spin_lock_irqsave(&mem->lru_lock, flags);
+ /* mem is valid while doing this. */
+ mem_cgroup_charge_statistics(pc, false);
    list_del_init(&pc->lru);
    spin_unlock_irqrestore(&mem->lru_lock, flags);
- mem_cgroup_charge_statistics(mem, pc->flags, false);
    kfree(pc);
}
}
@@ -669,7 +765,7 @@ retry:
    css_put(&mem->css);
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    list_del_init(&pc->lru);
- mem_cgroup_charge_statistics(mem, pc->flags, false);
+ mem_cgroup_charge_statistics(pc, false);
    kfree(pc);
} else /* being uncharged ? ...do relax */
    break;
@@ -833,11 +929,20 @@ static const struct mem_cgroup_stat_desc
[MEM_CGROUP_STAT_RSS] = { "rss", PAGE_SIZE, },
};

+static const struct mem_cgroup_zstat_desc {
+ const char *msg;
+ u64 unit;
+} mem_cgroup_zstat_desc[] = {
+ [MEM_CGROUP_ZONESTAT_ACTIVE] = {"active", PAGE_SIZE},
+ [MEM_CGROUP_ZONESTAT_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;
+ struct mem_cgroup_zonestat *zstat = &mem_cont->zonestat;
    int i;

```

```

    for (i = 0; i < ARRAY_SIZE(stat->cpustat[0].count); i++) {
@@ -850,6 +955,16 @@ static int mem_control_stat_show(struct
    val += mem_cgroup_stat_desc[i].unit;
    seq_printf(m, "%s %lld\n", mem_cgroup_stat_desc[i].msg, val);
    }
+ for (i = 0; i < MEM_CGROUP_ZONESTAT_NUM; i++) {
+ int nid, z;
+ s64 val = 0;
+ for_each_node_state(nid, N_POSSIBLE)
+ for (z = 0; z < MAX_NR_ZONES; z++)
+ val += mem_cgroup_count_zonestat(zstat, nid,
+ z, i);
+ val += mem_cgroup_zstat_desc[i].unit;
+ seq_printf(m, "%s %lld\n", mem_cgroup_zstat_desc[i].msg, val);
+ }
    return 0;
}

```

```

@@ -905,10 +1020,33 @@ static struct cftype mem_cgroup_files[]

```

```

static struct mem_cgroup init_mem_cgroup;

+static struct mem_cgroup_zonestat_cpu *
+__alloc_mem_cgroup_zonestat(int nid)
+{
+ struct mem_cgroup_zonestat_cpu *mczc;
+ if (sizeof(*mczc) < PAGE_SIZE)
+ mczc = kmalloc_node(sizeof(*mczc), GFP_KERNEL, nid);
+ else
+ mczc = vmalloc_node(sizeof(*mczc), nid);
+ return mczc;
+}
+
+static void __free_mem_cgroup_zonestat(struct mem_cgroup_zonestat_cpu *mczc)
+{
+ if (!mczc)
+ return;
+ if (sizeof(*mczc) < PAGE_SIZE)
+ kfree(mczc);
+ else
+ vfree(mczc);
+ return;
+}

```

```

+
static struct cgroup_subsys_state *
mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
{
    struct mem_cgroup *mem;
+ int cpu;

    if (unlikely((cont->parent) == NULL)) {
        mem = &init_mem_cgroup;
@@ -924,7 +1062,23 @@ mem_cgroup_create(struct cgroup_subsys *
    INIT_LIST_HEAD(&mem->inactive_list);
    spin_lock_init(&mem->lru_lock);
    mem->control_type = MEM_CGROUP_TYPE_ALL;
+
+ for_each_possible_cpu(cpu) {
+     int nid = cpu_to_node(cpu);
+     struct mem_cgroup_zonestat_cpu *mczc;
+     mczc = __alloc_mem_cgroup_zonestat(nid);
+     if (!mczc)
+         goto free_err;
+     memset(mczc, sizeof(*mczc), 0);
+     mem->zonestat.cpustat[cpu] = mczc;
+ }
    return &mem->css;
+free_err:
+ for_each_possible_cpu(cpu)
+     __free_mem_cgroup_zonestat(mem->zonestat.cpustat[cpu]);
+ if (mem != &init_mem_cgroup)
+     kfree(mem);
+ return NULL;
}

static void mem_cgroup_pre_destroy(struct cgroup_subsys *ss,
@@ -937,7 +1091,12 @@ static void mem_cgroup_pre_destroy(struc
static void mem_cgroup_destroy(struct cgroup_subsys *ss,
    struct cgroup *cont)
{
- kfree(mem_cgroup_from_cont(cont));
+ int cpu;
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ for_each_possible_cpu(cpu)
+     __free_mem_cgroup_zonestat(mem->zonestat.cpustat[cpu]);
+
+ kfree(mem);
}

static int mem_cgroup_populate(struct cgroup_subsys *ss,
@@ -989,5 +1148,5 @@ struct cgroup_subsys mem_cgroup_subsys =

```

```
.destroy = mem_cgroup_destroy,  
.populate = mem_cgroup_populate,  
.attach = mem_cgroup_move_task,  
- .early_init = 1,  
+ .early_init = 0,  
};
```

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa h
Posted by [Balbir Singh](#) on Tue, 30 Oct 2007 12:05:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

KAMEZAWA Hiroyuki wrote:

```
> Because NODE_DATA(node)->node_zonelist[] is guaranteed to contain  
> all necessary zones, it is not necessary to use for_each_online_node.  
>  
> And this for_each_online_node() makes reclaim routine start always  
> from node 0. This is bad. This patch will make reclaim code start  
> from caller's node.  
>  
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>  
>
```

Hi, KAMEZAWA-San,

I remember acking a similar series of patches. Could you please cc
linux-mm and Andrew and request for this series to be picked up.
I'll try and re-ack the patches as I read through them.

--

Thanks,
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: [RFC][for -mm] memory cgroup enhancements take3 [3/9] remember

page cache

Posted by [yamamoto](#) on Tue, 30 Oct 2007 12:22:52 GMT

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

```
> @@ -93,6 +95,11 @@ enum {
> MEM_CGROUP_TYPE_MAX,
> };
>
> +enum charge_type {
> + MEM_CGROUP_CHARGE_TYPE_CACHE = 0,
> + MEM_CGROUP_CHARGE_TYPE_MAPPED = 0,
> +};
> +
```

should be different values. :-)

YAMAMOTO Takashi

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [9/9] per zone stat

Posted by [yamamoto](#) on Tue, 30 Oct 2007 12:32:59 GMT

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

```
> +
> +/*
> + * Per-zone statistics.
> + * Please be carefull. The array can be very big on envrionments whic has
> + * very big MAX_NUMNODES . Adding new stat member to this will eat much memory.
> + * Only Active/Inactive may be sutiable.
```

s/whic/&h/

s/sutiable/suitable/

```
> +static inline void __mem_cgroup_zonesta_dec(struct mem_cgroup_zonestat *zstat,
```

s/zonesta/&t/

```
> @@ -293,6 +369,23 @@ clear_page_cgroup(struct page *page, str
>
> static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
> {
> + int direction = 0;
> +
```

```

> + if (active && !(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
> + direction = 1; /*from inactive to active */
> + if (!active && (pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
> + direction = -1;
> +
> + if (direction) {
> + struct mem_cgroup_zonestat *zstat = &pc->mem_cgroup->zonestat;
> + int index = page_cgroup_to_zonestat_index(pc);
> + preempt_disable();
> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_ACTIVE,
> + direction, index);
> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_INACTIVE,
> + direction, index);

```

dec?

YAMAMOTO Takashi

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa h

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 12:33:38 GMT

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

On Tue, 30 Oct 2007 17:35:01 +0530

Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KAMEZAWA Hiroyuki wrote:

> > Because NODE_DATA(node)->node_zonelist[] is guaranteed to contain

> > all necessary zones, it is not necessary to use for_each_online_node.

> >

> > And this for_each_online_node() makes reclaim routine start always

> > from node 0. This is bad. This patch will make reclaim code start

> > from caller's node.

> >

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

> >

>

> Hi, KAMEZAWA-San,

>

> I remember acking a similar series of patches. Could you please cc

> linux-mm and Andrew and request for this series to be picked up.

> I'll try and re-ack the patches as I read through them.

ok. I'll try tomorrow. If I have to rebase, I'll do.

Thanks,
-Kame

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [3/9] remember page cache

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 12:34:03 GMT

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

On Tue, 30 Oct 2007 21:22:52 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

```
> > @@ -93,6 +95,11 @@ enum {
> > MEM_CGROUP_TYPE_MAX,
> > };
> >
> > +enum charge_type {
> > + MEM_CGROUP_CHARGE_TYPE_CACHE = 0,
> > + MEM_CGROUP_CHARGE_TYPE_MAPPED = 0,
> > +};
> > +
>
> should be different values. :-)
>
Ugh...thanks..
```

-Kame

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [9/9] per zone stat

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 12:36:15 GMT

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

On Tue, 30 Oct 2007 21:32:59 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

```

> > +
> > +/*
> > + * Per-zone statistics.
> > + * Please be careful. The array can be very big on environments which has
> > + * very big MAX_NUMNODES . Adding new stat member to this will eat much memory.
> > + * Only Active/Inactive may be suitable.
>
> s/whic/&h/
> s/suitable/suitable/
>
> > +static inline void __mem_cgroup_zonestat_dec(struct mem_cgroup_zonestat *zstat,
>
> s/zonestat/&t/
>
thanks...maybe patch refresh was wrong.

```

```

> > + if (active && !(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
> > + direction = 1; /*from inactive to active */
> > + if (!active && (pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
> > + direction = -1;
> > +
> > + if (direction) {
> > + struct mem_cgroup_zonestat *zstat = &pc->mem_cgroup->zonestat;
> > + int index = page_cgroup_to_zonestat_index(pc);
> > + preempt_disable();
> > + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_ACTIVE,
> > + direction, index);
> > + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_INACTIVE,
> > + direction, index);
>
> dec?
>
direction(add value) is 1 or -1 here. Hmm, this is maybe confusing.
ok, I'll clean up this.

```

Thanks,
-Kame

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [9/9] per zone stat
Posted by [yamamoto](#) on Tue, 30 Oct 2007 12:42:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

```

>>> + if (active && !(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
>>> + direction = 1; /*from inactive to active */
>>> + if (!active && (pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
>>> + direction = -1;
>>> +
>>> + if (direction) {
>>> + struct mem_cgroup_zonestat *zstat = &pc->mem_cgroup->zonestat;
>>> + int index = page_cgroup_to_zonestat_index(pc);
>>> + preempt_disable();
>>> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_ACTIVE,
>>> + direction, index);
>>> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_INACTIVE,
>>> + direction, index);
>>
>> dec?
>>
> direction(add value) is 1 or -1 here. Hmm, this is maybe confusing.
> ok, I'll clean up this.

```

adding the same value to both of active and inactive seems wrong.
i think you want to subtract 'direction' from inactive here.

YAMAMOTO Takashi

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [9/9] per zone stat

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 12:44:08 GMT

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

On Tue, 30 Oct 2007 21:42:41 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

```

>>>> + if (active && !(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
>>>> + direction = 1; /*from inactive to active */
>>>> + if (!active && (pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
>>>> + direction = -1;
>>>> +
>>>> + if (direction) {
>>>> + struct mem_cgroup_zonestat *zstat = &pc->mem_cgroup->zonestat;
>>>> + int index = page_cgroup_to_zonestat_index(pc);
>>>> + preempt_disable();
>>>> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_ACTIVE,
>>>> + direction, index);

```

```
>>> + __mem_cgroup_zonestat_add(zstat, MEM_CGROUP_ZONESTAT_INACTIVE,  
>>> + direction, index);  
>>>  
>>> dec?  
>>>  
>> direction(add value) is 1 or -1 here. Hmm, this is maybe confusing.  
>> ok, I'll clean up this.  
>  
> adding the same value to both of active and inactive seems wrong.  
> i think you want to subtract 'direction' from inactive here.  
>
```

Ahh, ok, thanks. will fix.

-Kame

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa h
Posted by [Dave Hansen](#) on Tue, 30 Oct 2007 17:31:34 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 2007-10-30 at 20:14 +0900, KAMEZAWA Hiroyuki wrote:

```
>  
> -   for_each_online_node(node) {  
> -       zones =  
> NODE_DATA(node)->node_zonelists[target_zone].zones;  
> -       if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))  
> +       zones = NODE_DATA(node)->node_zonelists[target_zone].zones;  
> +       if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))  
>           return 1;  
> -   }  
>   return 0;  
> }  
> #endif
```

Am I reading the diff wrong, or is that return indented too far?

-- Dave

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

Subject: Re: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa h
Posted by [Balbir Singh](#) on Tue, 30 Oct 2007 18:28:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen wrote:

```
> On Tue, 2007-10-30 at 20:14 +0900, KAMEZAWA Hiroyuki wrote:
>> -   for_each_online_node(node) {
>> -       zones =
>> NODE_DATA(node)->node_zonelists[target_zone].zones;
>> -       if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>> +       zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
>> +       if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>>             return 1;
>> -   }
>>     return 0;
>> }
>> #endif
>
> Am I reading the diff wrong, or is that return indented too far?
>
> -- Dave
```

Good catch! It is indeed indented one additional tab away from where it should be

--

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: [RFC][for -mm] memory cgroup enhancements take3 [1/9] fix
try_to_free_mem_cgroup_pages() numa h
Posted by [KAMEZAWA Hiroyuki](#) on Wed, 31 Oct 2007 05:36:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 30 Oct 2007 23:58:00 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

```
> Dave Hansen wrote:
> > On Tue, 2007-10-30 at 20:14 +0900, KAMEZAWA Hiroyuki wrote:
> >> -   for_each_online_node(node) {
```

```
>>> -         zones =
>>> NODE_DATA(node)->node_zonelists[target_zone].zones;
>>> -         if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>>> +         zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
>>> +         if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>>>             return 1;
>>> -     }
>>>     return 0;
>>> }
>>> #endif
>>
>> Am I reading the diff wrong, or is that return indented too far?
>>
>> -- Dave
>
> Good catch! It is indeed indented one additional tab away from where it
> should be.
```

Thanks, I'll check my tree.

-Kame

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