
Subject: Re: [PATCH][RFC] dirty balancing for cgroups
Posted by [yamamoto](#) on Fri, 11 Jul 2008 04:06:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

hi,

> On Wed, 9 Jul 2008 15:00:34 +0900 (JST)
> yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:
>
> > hi,
> >
> > the following patch is a simple implementation of
> > dirty balancing for cgroups. any comments?
> >
> > it depends on the following fix:
> > <http://lkml.org/lkml/2008/7/8/428>
> >
>
> A few comments ;)

thanks for comments.

> - This looks simple but, could you merge this into memory resource controller ?

why?

> (if conflict, I'll queue on my stack.)
> - Do you have some number ? or How we can test this works well ?

i have no numbers yet.
it should work as well as the one for tasks. (ie. i don't know. :)

> - please CC to linux-mm.

ok, i will.

YAMAMOTO Takashi

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

```

> > diff --git a/include/linux/cgroup_subsys.h b/include/linux/cgroup_subsys.h
> > index 23c02e2..f5453cc 100644
> > --- a/include/linux/cgroup_subsys.h
> > +++ b/include/linux/cgroup_subsys.h
> > @@ -52,3 +52,9 @@ SUBSYS(memrlimit_cgroup)
> > #endif
> >
> > /* */
> > +
> > +#ifdef CONFIG_CGROUP_MEMDIRTYLIMIT_CTLR
> > +SUBSYS(memdirtylimit_cgroup)
> > +#endif
> > +
> > +/* */
> > diff --git a/include/linux/memdirtylimitcgroup.h b/include/linux/memdirtylimitcgroup.h
> > new file mode 100644
> > index 0000000..667d312
> > --- /dev/null
> > +++ b/include/linux/memdirtylimitcgroup.h
> > @@ -0,0 +1,47 @@
> > +
> > +/*
> > + * memdirtylimitcgroup.h COPYRIGHT FUJITSU LIMITED 2008
> > + *
> > + * Author: yamamoto@valinux.co.jp
> > + */
> > +
> > +struct task_struct;
> > +
> > +#if defined(CONFIG_CGROUP_MEMDIRTYLIMIT_CTLR)
> > +
> > +void memdirtylimitcgroup_dirty_inc(struct task_struct *);
> > +void memdirtylimitcgroup_dirty_limit(struct task_struct *, long *);
> > +void memdirtylimitcgroup_change_shift(int);
> > +void memdirtylimitcgroup_init(int);
> > +
> > +#else /* defined(CONFIG_CGROUP_MEMDIRTYLIMIT_CTLR) */
> > +
> > +static inline void
> > +memdirtylimitcgroup_dirty_inc(struct task_struct *t)
> > +{
> > +
> > + /* nothing */
> > +}
> > +
> > +static inline void
> > +memdirtylimitcgroup_dirty_limit(struct task_struct *t, long *dirty)
> > +{

```

```

>> +
>> + /* nothing */
>> +}
>> +
>> +static inline void
>> +memdirtylimitcgroup_change_shift(int shift)
>> +{
>> +
>> + /* nothing */
>> +}
>> +
>> +static inline void
>> +memdirtylimitcgroup_init(int shift)
>> +{
>> +
>> + /* nothing */
>> +}
>> +
>> +#endif /* defined(CONFIG_CGROUP_MEMDIRTYLIMIT_CTLR) */
>> diff --git a/init/Kconfig b/init/Kconfig
>> index 162d462..985bac8 100644
>> --- a/init/Kconfig
>> +++ b/init/Kconfig
>> @@ -418,6 +418,12 @@ config CGROUP_MEMRLIMIT_CTLR
>>     memory RSS and Page Cache control. Virtual address space control
>>     is provided by this controller.
>>
>> +config CGROUP_MEMDIRTYLIMIT_CTLR
>> + bool "Memory Dirty Limit Controller for Control Groups"
>> + depends on CGROUPS && RESOURCE_COUNTERS
>> + help
>> +   XXX TBD
>> +
>> config SYSFS_DEPRECATED
>> bool
>>
>> diff --git a/mm/Makefile b/mm/Makefile
>> index f54232d..8603d19 100644
>> --- a/mm/Makefile
>> +++ b/mm/Makefile
>> @@ -35,4 +35,5 @@ obj-$(CONFIG_SMP) += allocpercpu.o
>> obj-$(CONFIG_QUICKLIST) += quicklist.o
>> obj-$(CONFIG_CGROUP_MEM_RES_CTLR) += memcontrol.o
>> obj-$(CONFIG_CGROUP_MEMRLIMIT_CTLR) += memrlimitcgroup.o
>> +obj-$(CONFIG_CGROUP_MEMDIRTYLIMIT_CTLR) += memdirtylimitcgroup.o
>> obj-$(CONFIG_MMU_NOTIFIER) += mmu_notifier.o
>> diff --git a/mm/memdirtylimitcgroup.c b/mm/memdirtylimitcgroup.c
>> new file mode 100644

```

```

> > index 00000000..b70b33d
> > --- /dev/null
> > +++ b/mm/memdirtylimitcgroup.c
> > @@ -0,0 +1,179 @@
> > +
> > +/*
> > + * memdirtylimitcgroup.c COPYRIGHT FUJITSU LIMITED 2008
> > + *
> > + * Author: yamamoto@valinux.co.jp
> > + */
> > +
> > +#include <linux/err.h>
> > +#include <linux/cgroup.h>
> > +#include <linux/rcupdate.h>
> > +#include <linux/slab.h>
> > +#include <linux/memdirtylimitcgroup.h>
> > +
> > +#include <asm/div64.h>
> > +
> > +static struct prop_descriptor vm_cgroup_dirties;
> > +
> > +struct memdirtylimit_cgroup {
> > + struct cgroup_subsys_state dlcg_css;
> > + spinlock_t dlcg_lock;
> > + struct prop_local_single dlcg_dirties;
> > +};
> > +
> > +static struct cgroup_subsys_state *
> > +task_to_css(struct task_struct *task)
> > +{
> > +
> > + return task_subsys_state(task, memdirtylimit_cgroup_subsys_id);
> > +}
> > +
> > +static struct memdirtylimit_cgroup *
> > +css_to_dlcg(struct cgroup_subsys_state *css)
> > +{
> > +
> > + return container_of(css, struct memdirtylimit_cgroup, dlcg_css);
> > +}
> > +
> > +static struct cgroup_subsys_state *
> > +cg_to_css(struct cgroup *cg)
> > +{
> > +
> > + return cgroup_subsys_state(cg, memdirtylimit_cgroup_subsys_id);
> > +}
> > +

```

```

>> +static struct memdirtylimit_cgroup *
>> +cg_to_dlcg(struct cgroup *cg)
>> +{
>> +
>> + return css_to_dlcg(cg_to_css(cg));
>> +}
>> +
>> +/* ----- */
>> +
>> +static void
>> +getfraction(struct memdirtylimit_cgroup *dlcg, long *numeratorp,
>> + long *denominatorp)
>> +{
>> +
>> + spin_lock(&dlcg->dlcg_lock);
>> + prop_fraction_single(&vm_cgroup_dirties, &dlcg->dlcg_dirties,
>> + numeratorp, denominatorp);
>> + spin_unlock(&dlcg->dlcg_lock);
>> +}
>> +
>> +/* ----- */
>> +
>> +void
>> +memdirtylimitcgroup_dirty_inc(struct task_struct *t)
>> +{
>> + struct memdirtylimit_cgroup *dlcg;
>> +
>> + rcu_read_lock();
>> + dlcg = css_to_dlcg(task_to_css(t));
>> + spin_lock(&dlcg->dlcg_lock);
>> + prop_inc_single(&vm_cgroup_dirties, &dlcg->dlcg_dirties);
>> + spin_unlock(&dlcg->dlcg_lock);
>> + rcu_read_unlock();
>> +}
>> +
>> +void
>> +memdirtylimitcgroup_dirty_limit(struct task_struct *t, long *dirtyp)
>> +{
>> + struct memdirtylimit_cgroup *dlcg;
>> + unsigned long dirty = *dirtyp;
>> + uint64_t tmp;
>> + long numerator;
>> + long denominator;
>> +
>> + BUG_ON(*dirtyp < 0);
>> +
>> + rcu_read_lock();
>> + dlcg = css_to_dlcg(task_to_css(t));

```

```

>> + getfraction(dlcg, &numerator, &denominator);
>> + rcu_read_unlock();
>> +
>> + tmp = (uint64_t)(dirty >> 1) * numerator;
>> + do_div(tmp, denominator);
>> + *dirty = dirty - (unsigned long)tmp;
>> +}
>> +
>> +void
>> +memdirtylimitcgroup_change_shift(int shift)
>> +{
>> +
>> + prop_change_shift(&vm_cgroup_dirties, shift);
>> +}
>> +
>> +void
>> +memdirtylimitcgroup_init(int shift)
>> +{
>> +
>> + prop_descriptor_init(&vm_cgroup_dirties, shift);
>> +}
>> +
>> +/* ----- */
>> +
>> +static u64
>> +memdirtylimit_cgroup_read_fraction(struct cgroup *cg, struct cftype *cft)
>> +{
>> + struct memdirtylimit_cgroup *dlcg;
>> + uint64_t result;
>> + long numerator;
>> + long denominator;
>> +
>> + dlcg = cg_to_dlcg(cg);
>> + getfraction(dlcg, &numerator, &denominator);
>> + result = (uint64_t)100 * numerator;
>> + do_div(result, denominator);
>> + return result;
>> +}
>> +
>> +static const struct cftype files[] = {
>> + {
>> + .name = "fraction",
>> + .read_u64 = memdirtylimit_cgroup_read_fraction,
>> + },
>> +};
>> +
>> +static int
>> +memdirtylimit_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cg)

```

```

>> +{
>> +
>> + return cgroup_add_files(cg, ss, files, ARRAY_SIZE(files));
>> +}
>> +
>> +static struct cgroup_subsys_state *
>> +memdirtylimit_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cg)
>> +{
>> + struct memdirtylimit_cgroup *dlcg;
>> + int error;
>> +
>> + dlcg = kzalloc(sizeof(*dlcg), GFP_KERNEL);
>> + if (dlcg == NULL)
>> + return ERR_PTR(-ENOMEM);
>> + error = prop_local_init_single(&dlcg->dlcg_dirties);
>> + if (error != 0) {
>> + kfree(dlcg);
>> + return ERR_PTR(error);
>> + }
>> + spin_lock_init(&dlcg->dlcg_lock);
>> + return &dlcg->dlcg_css;
>> +}
>> +
>> +static void
>> +memdirtylimit_cgroup_destroy(struct cgroup_subsys *ss, struct cgroup *cg)
>> +{
>> + struct memdirtylimit_cgroup *dlcg = cg_to_dlcg(cg);
>> +
>> + prop_local_destroy_single(&dlcg->dlcg_dirties);
>> + kfree(dlcg);
>> +}
>> +
>> +struct cgroup_subsys memdirtylimit_cgroup_subsys = {
>> + .name = "memdirtylimit",
>> + .subsys_id = memdirtylimit_cgroup_subsys_id,
>> + .create = memdirtylimit_cgroup_create,
>> + .destroy = memdirtylimit_cgroup_destroy,
>> + .populate = memdirtylimit_cgroup_populate,
>> +};
>> diff --git a/mm/page-writeback.c b/mm/page-writeback.c
>> index e6fa69e..f971532 100644
>> --- a/mm/page-writeback.c
>> +++ b/mm/page-writeback.c
>> @@ -34,6 +34,7 @@
>> #include <linux/syscalls.h>
>> #include <linux/buffer_head.h>
>> #include <linux/pagevec.h>
>> +#include <linux/memdirtylimitcgroup.h>

```

```

>>
>> /*
>> * The maximum number of pages to writeout in a single bdflush/kupdate
>> @@ -152,6 +153,7 @@ int dirty_ratio_handler(struct ctl_table *table, int write,
>>     int shift = calc_period_shift();
>>     prop_change_shift(&vm_completions, shift);
>>     prop_change_shift(&vm_dirties, shift);
>> + memdirtylimitcgroup_change_shift(shift);
>> }
>> return ret;
>> }
>> @@ -393,6 +395,8 @@ get_dirty_limits(long *pbackground, long *pdirty, long *pbdi_dirty,
>>     if (bdi) {
>>         u64 bdi_dirty;
>>         long numerator, denominator;
>> + long task_dirty;
>> + long cgroup_dirty;
>>
>>     /*
>>      * Calculate this BDI's share of the dirty ratio.
>> @@ -408,7 +412,11 @@ get_dirty_limits(long *pbackground, long *pdirty, long *pbdi_dirty,
>>
>>     *pbdi_dirty = bdi_dirty;
>>     clip_bdi_dirty_limit(bdi, dirty, pbdi_dirty);
>> - task_dirty_limit(current, pbdi_dirty);
>> + task_dirty = *pbdi_dirty;
>> + task_dirty_limit(current, &task_dirty);
>> + cgroup_dirty = *pbdi_dirty;
>> + memdirtylimitcgroup_dirty_limit(current, &cgroup_dirty);
>> + *pbdi_dirty = min(task_dirty, cgroup_dirty);
>> }
>> }
>>
>> @@ -842,6 +850,7 @@ void __init page_writeback_init(void)
>>     shift = calc_period_shift();
>>     prop_descriptor_init(&vm_completions, shift);
>>     prop_descriptor_init(&vm_dirties, shift);
>> + memdirtylimitcgroup_init(shift);
>> }
>>
>> /**
>> @@ -1105,6 +1114,7 @@ int __set_page_dirty_nobuffers(struct page *page)
>> }
>>
>>     task_dirty_inc(current);
>> + memdirtylimitcgroup_dirty_inc(current);
>>
>>     return 1;

```

> > }

> >

>

>

> Containers mailing list

> Containers@lists.linux-foundation.org

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

Containers mailing list

Containers@lists.linux-foundation.org

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