
Subject: Re: [RFC][-mm] Simple stats for cpu resource controller v3
Posted by [Balaji Rao](#) on Fri, 02 May 2008 23:56:46 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Saturday 03 May 2008 05:11:33 am Andrew Morton wrote:

```
<snip>
> > > >
> > > > percpu_counter_init uses kmalloc to create percpu counters. This
> > > > raises an early exception as kmem_cache is not initialized that
> > > > early.
> > >
> > > whaa? kmalloc is ready to be used quite early in boot. It's a bit of
> > > a concern that the CPU resource controller is doing stuff before even
> > > kmalloc is ready to go.
> >
> > No it doesn't. I placed the call to percpu_counter_init in sched_init.
> > Hence the problem.
>
> I'm groping in the dark without seeing the diff.
oh! am sorry about that! some cosmetic changes remain. It also contains some
modifications done to the percpu_counter core.
```

```
<snip>
> > For the group_init_late() which you suggested, we should probably hook
> > into a place where kmalloc is ready and
> > account_user_time/account_system_time is not yet called even once. Does
> > it make sense ?
>
> yes, that would be good.
OK, so when does account_system_time get called for the first time ? after
IRQs are set up, is it ? So, where do we place the hook ?
```

Here's the patch.

```
diff --git a/include/linux/percpu_counter.h b/include/linux/percpu_counter.h
index 9007ccd..8a1b756 100644
--- a/include/linux/percpu_counter.h
+++ b/include/linux/percpu_counter.h
@@ -21,7 +21,7 @@ struct percpu_counter {
#ifdef CONFIG_HOTPLUG_CPU
    struct list_head list; /* All percpu_counters are on a list */
#endif
- s32 *counters;
+ s32 counters[NR_CPUS];
};

#if NR_CPUS >= 16
```

```

diff --git a/kernel/sched.c b/kernel/sched.c
index bbdc32a..fde11f8 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -68,6 +68,7 @@
#include <linux/hrtimer.h>
#include <linux/tick.h>
#include <linux/ftrace.h>
+#include <linux/percpu_counter.h>

#include <asm/tlb.h>
#include <asm/irq_regs.h>
@@ -248,10 +249,34 @@ struct cfs_rq;

static LIST_HEAD(task_groups);

#ifdef CONFIG_CGROUP_SCHED
+
#define CPU_CGROUP_STAT_THRESHOLD 1 << 30
+
enum cpu_cgroup_stat_index {
+ CPU_CGROUP_STAT_UTIME, /* Usertime of the task group */
+ CPU_CGROUP_STAT_STIME, /* Kerntime of the task group */
+
+ CPU_CGROUP_STAT_NSTATS,
+};
+
+struct cpu_cgroup_stat {
+ struct percpu_counter cpustat[CPU_CGROUP_STAT_NSTATS];
+};
+
+/* Called under irq disable. */
+static void __cpu_cgroup_stat_add(struct cpu_cgroup_stat *stat,
+ enum cpu_cgroup_stat_index idx, int val)
+{
+ percpu_counter_add(&stat->cpustat[idx], val);
+}
+#endif
+
/* task group related information */
struct task_group {
#ifdef CONFIG_CGROUP_SCHED
struct cgroup_subsys_state css;
+ struct cpu_cgroup_stat stat;
#endif

#ifdef CONFIG_FAIR_GROUP_SCHED
@@ -3837,6 +3862,16 @@ void account_user_time(struct task_struct *p, cputime_t cputime)

```

```

    cpustat->nice = cputime64_add(cpustat->nice, tmp);
else
    cpustat->user = cputime64_add(cpustat->user, tmp);
+
+ /* Charge the task's group */
+ #ifdef CONFIG_CGROUP_SCHED
+ {
+   struct task_group *tg;
+   tg = task_group(p);
+   __cpu_cgroup_stat_add(&tg->stat, CPU_CGROUP_STAT_UTIME,
+   cputime_to_msecs(cputime));
+ }
+ #endif
+ }

/*
@@ -3900,6 +3935,15 @@ void account_system_time(struct task_struct *p, int hardirq_offset,
    cpustat->idle = cputime64_add(cpustat->idle, tmp);
    /* Account for system time used */
    acct_update_integrals(p);
+
+ #ifdef CONFIG_CGROUP_SCHED
+ {
+   struct task_group *tg;
+   tg = task_group(p);
+   __cpu_cgroup_stat_add(&tg->stat, CPU_CGROUP_STAT_STIME,
+   cputime_to_msecs(cputime));
+ }
+ #endif
+ }

/*
@@ -7426,6 +7470,10 @@ void __init sched_init(void)
# ifdef CONFIG_GROUP_SCHED
    list_add(&init_task_group.list, &task_groups);
# endif
+
+ for(i = 0; i < CPU_CGROUP_STAT_NSTATS; i++) {
+   percpu_counter_init(&init_task_group.stat.cpustat[i], 0);
+ }

    for_each_possible_cpu(i) {
        struct rq *rq;
@@ -8183,10 +8231,12 @@ static struct cgroup_subsys_state *
cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)
{
    struct task_group *tg;
+   int i;

```

```

if (!cgrp->parent) {
    /* This is early initialization for the top cgroup */
    init_task_group.css.cgroup = cgrp;
+ /* Initialize init_task_grp's stat object */
    return &init_task_group.css;
}

@@ -8201,6 +8251,10 @@ cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)
    /* Bind the cgroup to task_group object we just created */
    tg->css.cgroup = cgrp;

+ /* Initialize the stats object */
+ for(i = 0; i < CPU_CGROUP_STAT_NSTATS; i++)
+ percpu_counter_init(&tg->stat.cpustat[i], 0);
+
    return &tg->css;
}

@@ -8249,6 +8303,36 @@ static u64 cpu_shares_read_u64(struct cgroup *cgrp, struct cftype
*cft)

    return (u64) tg->shares;
}
+
+static s64 cpu_cgroup_read_stat(struct cpu_cgroup_stat *stat,
+ enum cpu_cgroup_stat_index idx)
+{
+ return percpu_counter_read(&stat->cpustat[idx]);
+}
+
+static const struct cpu_cgroup_stat_desc {
+ const char *msg;
+ u64 unit;
+} cpu_cgroup_stat_desc[] = {
+ [CPU_CGROUP_STAT_UTIME] = { "utime", 1, },
+ [CPU_CGROUP_STAT_STIME] = { "stime", 1, },
+};
+
+static int cpu_cgroup_stats_show(struct cgroup *cgrp, struct cftype *cft,
+ struct cgroup_map_cb *cb)
+{
+ struct task_group *tg = cgroup_tg(cgrp);
+ struct cpu_cgroup_stat *stat = &tg->stat;
+ int i;
+
+ for (i = 0; i < CPU_CGROUP_STAT_NSTATS; i++) {
+ s64 val;

```

```

+ val = cpu_cgroup_read_stat(stat, i);
+ val *= cpu_cgroup_stat_desc[i].unit;
+ cb->fill(cb, cpu_cgroup_stat_desc[i].msg, val);
+ }
+ return 0;
+}
#endif

#ifdef CONFIG_RT_GROUP_SCHED
@@ -8295,6 +8379,10 @@ static struct cftype cpu_files[] = {
    .write_u64 = cpu_rt_period_write_uint,
    },
#endif
+ {
+ .name = "stat",
+ .read_map = cpu_cgroup_stats_show,
+ },
+};

static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cont)
@@ -8310,7 +8398,7 @@ struct cgroup_subsys cpu_cgroup_subsys = {
    .attach = cpu_cgroup_attach,
    .populate = cpu_cgroup_populate,
    .subsys_id = cpu_cgroup_subsys_id,
- .early_init = 1,
+// .early_init = 1,
};

#endif /* CONFIG_CGROUP_SCHED */
diff --git a/lib/percpu_counter.c b/lib/percpu_counter.c
index 1191744..a7e84f9 100644
--- a/lib/percpu_counter.c
+++ b/lib/percpu_counter.c
@@ -14,6 +14,7 @@ static LIST_HEAD(percpu_counters);
static DEFINE_MUTEX(percpu_counters_lock);
#endif

+#define per_cpu_ptr(var,cpu) &var[cpu]
void percpu_counter_set(struct percpu_counter *fbc, s64 amount)
{
    int cpu;
@@ -74,7 +75,6 @@ int percpu_counter_init(struct percpu_counter *fbc, s64 amount)
{
    spin_lock_init(&fbc->lock);
    fbc->count = amount;
- fbc->counters = alloc_percpu(s32);
    if (!fbc->counters)
        return -ENOMEM;

```

```
#ifdef CONFIG_HOTPLUG_CPU
@@ -102,7 +102,6 @@ void percpu_counter_destroy(struct percpu_counter *fbc)
    return;

    free_percpu(fbc->counters);
- fbc->counters = NULL;
#ifdef CONFIG_HOTPLUG_CPU
    mutex_lock(&percpu_counters_lock);
    list_del(&fbc->list);
```

--
Warm Regards,

Balaji Rao
Dept. of Mechanical Engineering
NITK

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