Subject: Re: [PATCH v2 2/5] account guest time per-cgroup as well.
Posted by Glauber Costa on Mon, 28 May 2012 09:03:19 GMT
View Forum Message <> Reply to Message

On 05/26/2012 08:44 AM, Paul Turner wrote:
> On 04/09/2012 03:25 PM, Glauber Costa wrote:
>> In the interest of providing a per-cgroup figure of common statistics,
>> this patch adds a nr_switches counter to each group runqueue (both cfs
>> and rt).
>>
>> To avoid impact on schedule(), we don't walk the tree at stat gather
>> time. This is because schedule() is called much more frequently than
>> the tick functions, in which we do walk the tree.
>>
>> When this figure needs to be read (different patch), we will
>> aggregate them at read time.
>>
>> Signed-off-by: Glauber Costa<glommer-bzQdu9zFT3WakBO8gow8eQ@public.gmane.org>
>> ---
>> kernel/sched/core.c | 32 ++++++++++++++++++++++++++++++++
>> kernel/sched/sched.h | 3 +++
>> 2 files changed, 35 insertions(+), 0 deletions(-)
>>
>> diff --git a/kernel/sched/core.c b/kernel/sched/core.c
>> index 1cfb7f0..1ee3772 100644
>> --- a/kernel/sched/core.c
>> +++ b/kernel/sched/core.c
>> @@ -3168,6 +3168,37 @@ pick_next_task(struct rq *rq)
>> }
>>
>> /*
>> + * For all other data, we do a tree walk at the time of
>> + * gathering. We want, however, to minimize the impact over schedule(),
>> + * because... well... it's schedule().
>> + *
>> + * Therefore we only gather for the current cgroup, and walk the tree
>> + * at read time
>> + */
>> +static void update_switches_task_group(struct rq *rq,
>> +   struct task_struct *prev,
>> +   struct task_struct *next)
>> +{
>> +#ifdef CONFIG_CGROUP_SCHED
>> + int cpu = cpu_of(rq);
>> +
>> + if (rq->curr_tg ==&root_task_group)
>> + goto out;
>> +

>> +#ifdef CONFIG_FAIR_GROUP_SCHED
>> + if (prev->sched_class ==&fair_sched_class)
>> + rq->curr_tg->cfs_rq[cpu]->nr_switches++;
>> +#endif
>> +#ifdef CONFIG_RT_GROUP_SCHED
>> + if (prev->sched_class ==&rt_sched_class)
>> + rq->curr_tg->rt_rq[cpu]->nr_switches++;
>> +#endif
>
> With this approach why differentiate cfs vs rt? These could both just
> be on the task_group.
>
> This could then just be
> if (prev != root_task_group)
>   task_group(prev)->nr_switches++;

well, no. Then it needs to be an atomic update, or something like it.
The runqueue is a percpu data, the task_group is not. That's why I
choosed to use a rq (and the runqueues are separated between classes).

It all boils down to the fact that I wanted to avoid an atomic update in
this path.

But if you think that would be okay, I could change it. Alternatively, I
could come up with another percpu storage as well, since we're
ultimately just reading it later (and for that we need to iterate on all
cpus anyway).

> Which you could wrap in a nice static inline that disappears when
> CONFIG_CGROUP_PROC_STAT isn't there

That I can do.

>
> Another way to go about this would be to promote (demote?) nr_switches
> to the sched_entity. At which point you know you only need to update
> yours, and conditionally update your parents.

You mean the global one ?

Not sure it will work, because that always refer to the root cgroup...

> But.. that's still gross.. Hmm..
>
>> +out:
>> + rq->curr_tg = task_group(next);
>
> If you're going to task_group every time anyway you might as well just

> take it against prev -- then you don't have to cache rq->curr_tg?
>
> Another way to do this would be:
>
> On cfs_rq, rt_rq add:
> int prev_rq_nr_switches, nr_switches
>
> On put_prev_prev_task_fair (against a task)
> cfs_rq_of(prev->se)->prev_rq_nr_switches = rq->nr_switches
>
> On pick_next_task_fair:
> if (cfs_rq_of(prev->se)->prev_rq_nr_switches != rq->nr_switches)
> cfs_rq->nr_switches++;
>
> On aggregating the value for read: +1 if prev_rq_nr_running != rq->nr_running
> [And equivalent for sched_rt]
>
> While this is no nicer (and fractionally more expensive but this is
> never something we'd enable by default), it at least gets the goop out
> of schedule().

At first look this sounds a bit weird to me, but OTOH, this is how a lot of the stuff is done... All the other statistics in the patch set are collected this exact same way - because it draws from schedstats, that always touch the specific rqs, so maybe this gain points for consistency.

I'll give it a shot.

BTW, this means that your first comment about merging cfq and rt is basically to be disconsidered should I take this route, right ?