Subject: Re: [PATCH v3 5/6] Also record sleep start for a task group
Posted by Glauber Costa on Wed, 30 May 2012 12:24:40 GMT

View Forum Message <> Reply to Message

On 05/30/2012 03:35 PM, Paul Turner wrote:

```
> On Wed, May 30, 2012 at 2:48 AM, Glauber Costa<glommer@parallels.com>  wrote:
>> When we're dealing with a task group, instead of a task, also record
>> the start of its sleep time. Since the test agains TASK_UNINTERRUPTIBLE
>> does not really make sense and lack an obvious analogous, we always
>> record it as sleep_start, never block_start.
>>
>> Signed-off-by: Glauber Costa<glommer@parallels.com>
>> CC: Peter Zijlstra<a.p.zijlstra@chello.nl>
>> CC: Paul Turner<pjt@google.com>
>> ---
>>   kernel/sched/fair.c |    3 ++-
>>   1 file changed, 2 insertions(+), 1 deletion(-)
>>
>> diff --git a/kernel/sched/fair.c b/kernel/sched/fair.c
>> index c26fe38..d932559 100644
>> --- a/kernel/sched/fair.c
>> +++ b/kernel/sched/fair.c
>> @@ -1182,7 +1182,8 @@ dequeue_entity(struct cfs_rq *cfs_rq, struct sched_entity *se, int flags)
>>                                 se->statistics.sleep_start = rq_of(cfs_rq)->clock;
>>                         if (tsk->state&  TASK_UNINTERRUPTIBLE)
>>                                 se->statistics.block_start = rq_of(cfs_rq)->clock;
>> -               }
>> +               } else
>> +                       se->statistics.sleep_start = rq_of(cfs_rq)->clock;
>
> You can't sanely account sleep on a group entity.
>
> Suppose you have 2 sleepers on 1 cpu: you account 1s/s of idle
> Suppose you have 2 sleepers now on 2 cpus: you account 2s/s of idle
>
> Furthermore, in the latter case when one wakes up you still continue
> to accrue sleep time whereas in the former you don't.
>
> Just don't report/collect this.
```

sleep_start is not for iowait. This is for idle. And I know no other way
to collect idle time per cgroup, other than the time during which it was
out of the runqueue.

Now what you say about the sleepers don't make that much sense for idle
because this information is per-cpu as well.

When the se is being dequeued, it means none of its children is running
on that runqueue. That's idle.
>>   #endif
>>   }
>>
>> --
>> 1.7.10.2
>>