Subject: Re: [PATCH v3 1/6] measure exec_clock for rt sched entities
Posted by Paul Turner on Wed, 30 May 2012 11:00:37 GMT
View Forum Message <> Reply to Message

On Wed, May 30, 2012 at 3:42 AM, Glauber Costa <glommer@parallels.com> wrote:
> On 05/30/2012 02:42 PM, Peter Zijlstra wrote:
>>
>> On Wed, 2012-05-30 at 14:32 +0400, Glauber Costa wrote:
>>>>>
>>>>> +   for_each_sched_rt_entity(rt_se) {
>>>>> +   rt_rq = rt_rq_of_se(rt_se);
>>>>> +   schedstat_add(rt_rq, exec_clock, delta_exec);
>>>>> +   }
>>>>> +
>>>>>   if (!rt_bandwidth_enabled())
>>>>>   return;
>>>>
>>>>
>>>> See, this just makes me sad.. you now have a double
>>>> for_each_sched_rt_entity() loop.
>>>
>>>
>>> The way I read the rt.c code, it it is called from enqueue_task_rt only
>>> once.
>>
>>
>> Ah, what I meant was, right after that !rt_bandwidth_enabled() muck we
>> do another for_each_sched_rt_entity() walk.
>
> I guess I can fold it there...
>

Does this even need to be hierarchical? While it's natural for it to
be in the CFS case, it feels forced here.

You could instead make this rt_rq->local_exec_clock charging only to
the parenting rt_rq and post-aggregate when you want to report. The
only thing you'd need to be careful of is also accounting children
somewhere on the parent on destruction (reaped_exec_clock?).

Harking back to symmetry, local_exec_clock is also a potentially
useful stat on the CFS side of things since it allows you to usefully
disambiguate versus your children (common case where this is useful is
calculating usage of threads in the root cgroup); so it wouldn't need
to be unique to rt_rq.