Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Srivatsa Vaddagiri on Fri, 25 May 2007 10:56:41 GMT
View Forum Message <> Reply to Message

On Fri, May 25, 2007 at 10:29:51AM +0200, Ingo Molnar wrote:
> btw., what are you thoughts about SMP?

I was planning on reusing smpnice concepts here, with the difference
that we balance group weights across CPU in addition to total weight of
CPUs.

For ex, assuming weight of each task is 10

CPU0 => USER X (Wt 100) + USER Y (Wt 200) => total weight 300
CPU1 => USER X (Wt 30) + USER Y (Wt 80) => total weight 110

So first we notice that CPU0 and CPU1 are imbalanced by weight 190 and
target at reducing this imbalance by half i.e CPU1 has to pull total
weight of 95 (190/2) from CPU0. However while pulling weights, we apply
the same imbalance/2 rule at group level also. For ex: we cannot pull more than
70/2 = 35 from USER X on CPU0 or more than 120/2 = 60 from USER Y on CPU0.

Using this rule, after balance, the two CPUs may look like:

CPU0 => USER X (Wt 70) + USER Y (Wt 140) => total weight 210
CPU1 => USER X (Wt 60) + USER Y (Wt 140) => total weight 200

I had tried this approach earlier (in
https://lists.linux-foundation.org/pipermail/containers/2007-April/004580.html)
and had obtained decent results. It also required minimal changes to
smpnice.

Compared to this, what better degree of control/flexibilty does virtual
cpu approach give?

> it's a natural extension of your current code. I think the best approach
> would be to add a level of 'virtual CPU' objects above struct user. (how
> to set the attributes of those objects is open - possibly combine it
> with cpusets?)

are these virtual CPUs visible to users (ex: does smp_processor_id()
return virtual cpu id rather than physical id and does DEFINE_PER_CPU
create per-cpu data for virtual CPUs rather than physical cpus)?

> That way the scheduler would first pick a "virtual CPU" to schedule,

are virtual cpus pinned to their physical cpu or can they bounce around?
i.e can CPU #0 schedule VCPU D (in your example below)? If bouncing is

allowed, I am not sure whether that is a good thing for performance. How
do we minimize this performance cost?

> and then pick a user from that virtual CPU, and then a task from the user.
>
> To make group accounting scalable, the accounting object attached to the
> user struct should/must be per-cpu (per-vcpu) too. That way we'd have a
> clean hierarchy like:
>
> CPU #0 => VCPU A [40%] + VCPU B [60%]
> CPU #1 => VCPU C [30%] + VCPU D [70%]
>
> VCPU A => USER X [10%] + USER Y [90%]
> VCPU B => USER X [10%] + USER Y [90%]
> VCPU C => USER X [10%] + USER Y [90%]
> VCPU D => USER X [10%] + USER Y [90%]
>
> the scheduler first picks a vcpu, then a user from a vcpu. (the actual
> external structure of the hierarchy should be opaque to the scheduler
> core, naturally, so that we can use other hierarchies too)
>
> whenever the scheduler does accounting, it knows where in the hierarchy
> it is and updates all higher level entries too. This means that the
> accounting object for USER X is replicated for each VCPU it participates
> in.
>
> SMP balancing is straightforward: it would fundamentally iterate through
> the same hierarchy and would attempt to keep all levels balanced - i
> abstracted away its iterators already
>
> Hm?

--
Regards,
vatsa