Subject: Re: How to draw values for /proc/stat
Posted by Glauber Costa on Fri, 09 Dec 2011 14:55:53 GMT
View Forum Message <> Reply to Message

On 12/09/2011 12:03 PM, Peter Zijlstra wrote:
> On Mon, 2011-12-05 at 07:32 -0200, Glauber Costa wrote:
>> Hi,
>>
>> Specially Peter and Paul, but all the others:
>>
>> As you can see in https://lkml.org/lkml/2011/12/4/178, and in my answer
>> to that, there is a question - one I've asked before but without that
>> much of an audience - of whether /proc files read from process living on
>> cgroups should display global or per-cgroup resources.
>>
>> In the past, I was arguing for a knob to control that, but I recently
>> started to believe that a knob here will only overcomplicate matters:
>> if you live in a cgroup, you should display only the resources you can
>> possibly use. Global is for whoever is in the main cgroup.
>>
>> Now, it comes two questions:
>> 1) Do you agree with that, for files like /proc/stat ? I think the most
>> important part is to be consistent inside the system, regardless of what
>> is done
>
> Personally I don't give a rats arse about (/proc vs) cgroups :-)
> Currently /proc is unaffected by whatever cgroup you happen to be in and
> that seems to make some sort of sense.
>
> Namespaces seem to be about limiting visibility, cgroups about
> controlling resources.
>
> The two things are hopelessly disjoint atm, but I believe someone was
> looking at this mess.

I did take a look at this (if anyone else was, I'd like to know so we
can share some ideas), but I am not convinced we should do anything to
join them anymore. We virtualization people are to the best of my
knowledge the only ones doing namespaces. Cgroups, OTOH, got a lot bigger.

What I am mostly concerned about now, is how consistent they will be.
/proc always being always global indeed does make sense, but my question
still stands: if you live in a resource-controlled world, why should you
even see resources you will never own ?

> IOW a /proc namespace coupled to cgroup scope would do what you want.
> Now my head hurts..

Mine too. The idea is good, but too broad. Boils down to: How do you
couple them? And none of the methods I thought about seemed to make any
sense.

If we really want to have the values in /proc being opted-in, I think
Kamezawa's idea of a mount option is the winner so far.

>> 2) Will cpuacct stay?
>
> I really want to kill it. Balbir seems to want to retain a control-less
> accounting capability, but I really don't see the point in that. Nor is
> anybody selling it convincingly.
>
> So I think I'll simply deprecate it and schedule it for removal and
> anybody wanting something like this is free to send patches to implement
> what they want in the cpu controller and convince me its worth the pain.
>
>> I think if it does, that becomes almost mandatory
>
> You're referring to #1 here, right?

Yes. But that was before Kame suggested a mount option. That would do
just fine in this use case as well/

>> (at least the bind mount idea is pretty much over here), because drawing
>> value for /proc/stat becomes quite complex.
>> The cpuacct cgroup can provide user, sys, etc values. But we also have:
>>
>> * nr_context_switches,
>> * jiffies since boot,
>> * total_forks,
>> * nr_running,
>> * nr_iowait,
>>
>> Now I doubt any of us want to see /proc/stat extended to accommodate
>> things like nr_context_switches, or even worse, nr_running. The way I
>> see it, there are two options here:
>
> Why would we want to display all those anyway?

Because we care about isolation. Our users (and more importantly, their
tools), need to feel they own the box. That's the whole point of it.

Whatever is displayed to the users, need to be (with a switch/mount opt,
if my initial point is really defeated) related to the resources he is
entitled to, not to some global entity he'll never own.
>

>>   a) moving everything to cpu cgroup so we keep all values being drawn
>>   from the same place
>
> This is /cgroup/$controller.stat, right?

Sorry?

(I just noticed I wasn't that clear myself) I referred more to the act
of collecting values, and where from, than how to.

>
>>   b) Collect that info from multiple places in a transparent way. ctx,
>>   nr_running and nr_iowait will probably come from cpu. jiffies can
>>   come from wherever, and maybe we can even draw total_forks
>>   from Frederic's and avoid counting it twice.
>
> trouble with that is 'drawing from someplace' is a total nightmare, what
> happens if that fork muck from Frederic isn't co-mounted with the cpu
> controller?

If it is not co-mounted, we draw the global value. If you don't mount
it, I someone does not mount it, I can assure you he doesn't care about
it. We for sure will.