Subject: Re: [ckrm-tech] [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Srivatsa Vaddagiri on Fri, 25 May 2007 15:34:50 GMT
View Forum Message <> Reply to Message

On Fri, May 25, 2007 at 05:05:16PM +0400, Kirill Korotaev wrote:
> > That way the scheduler would first pick a "virtual CPU" to schedule, and
> > then pick a user from that virtual CPU, and then a task from the user.
>
> don't you mean the vice versa:
> first use to scheduler, then VCPU (which is essentially a runqueue or rbtree),
> then a task from VCPU?
>
> this is the approach we use in OpenVZ [...]

So is this how it looks in OpenVZ?

CONTAINER1 => VCPU0 + VCPU1
CONTAINER2 => VCPU2 + VCPU3

PCPU0 picks a container first, a vcpu next and then a task in it
PCPU1 also picks a container first, a vcpu next and then a task in it.

Few questions:

1. Are VCPU runqueues (on which tasks are present) global queues?

   That is, let's say that both PCPU0 and PCPU1 pick CONTAINER1 to schedule (first level) at the same time and next (let's say) they pick same vcpu VCPU0 to schedule (second level). Will the two pcpu's now have to be serialized for scanning task to schedule next (third level) within VCPU0 using a spinlock? Won't that shootup scheduling costs (esp on large systems), compared to (local scheduling + balance across cpus once in a while, the way its done today)?

   Or do you required that two pcpus don't schedule the same vcpu at the same time (the way hypervisors normally work)? Even then I would imagine a fair level of contention to be present in second step (pick a virtual cpu from a container's list of vcpus).

2. How would this load balance at virtual cpu level and sched domain based load balancing interact?

   The current sched domain based balancing code has many HT/MC/SMT related optimizations, which ensure that tasks are spread across physical threads/cores/packages in a most efficient manner - so as to utilize hardware bandwidth to the maximum. You would now need to introduce those optimizations essentially at schedule() time ..? Don't know if that is a wise thing to do.

3. How do you determine the number of VCPUs per container? Is there any
   relation for number of virtual cpus exposed per user/container and
   the number of available cpus? For ex: in case of user-driven
   scheduling, we would want all users to see the same number of cpus
   (which is the number available in the system).

4. VCPU ids (namespace) - is it different for different containers?

   For ex: can id's of vcpus belonging to different containers (say VCPU0 and
   VCPU2), as seen by users thr' vgetcpu/smp_processor_id() that is, be same?
   If so, then potentially two threads belonging to different users may find
   that they are running -truly simultaneously- on /same/ cpu 0 (one on
   VCPU0/PCPU0 and another on VCPU2/PCPU1) which normally isn't possible!

   This may be ok for containers, with non-overlapping cpu id namespace,
   but when applied to group scheduling for, say, users, which require a
   global cpu id namespace, wondering how that would be addressed ..

> and if you don't mind I would propose to go this way for fair-scheduling in
> mainstream.
> It has it's own advantages and disatvantages.
>
> This is not the easy way to go and I can outline the problems/disadvantages
> which appear on this way:
> - tasks which bind to CPU mask will bind to virtual CPUs.
>   no problem with user tasks, [...]

Why is this not a problem for user tasks? Tasks which bind to different
CPUs for performance reason now can find that they are running on same
(physical) CPU unknowingly.

> but some kernel threads
>   use this to do CPU-related management (like cpufreq).
>   This can be fixed using SMP IPI actually.
> - VCPUs should no change PCPUs very frequently,
>   otherwise there is some overhead. Solvable.
>
> Advantages:
> - High precision and fairness.

I just don't know if this benefit of high degree of fairness is worth the
complexity it introduces. Besides having some data which shows how much better
is is with respect to fairness/overhead when compared with other approaches
(like smpnice) would help I guess. I will however let experts like Ingo make
the final call here :)

--
Regards,
vatsa

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers