Subject: Re: [RFC | PATCH 0/9] CPU controller over process container
Posted by Herbert Poetzl on Thu, 12 Apr 2007 17:56:47 GMT
View Forum Message <> Reply to Message

On Thu, Apr 12, 2007 at 11:21:11PM +0530, Srivatsa Vaddagiri wrote:
> Here's a respin of my earlier CPU controller to work on top of Paul
> Menage's process container patches.
>
> Problem:
>
> Current CPU scheduler is very task centric, which makes it
> difficult to manage cpu resource consumption of a group of
> (related) tasks.
>
> For ex: with the current O(1) scheduler, it is possible for a user to
> monopolize CPU simply by spawning more and more threads, causing DoS
> to other users.
>
>
> Requirements:
>
> A few of them are:
>
> - Provide means to group tasks from user-land and
> specify limits of CPU bandwidth consumption of each group.
> CPU bandwidth limit is enforced over some suitable time
> period. For ex: a 40% limit could mean the task group's usage
> is limited to 4 sec every 10 sec or 24 sec every minute.
>
> - Time period over which bandwidth is controlled to each group
> to be configurable (?)
>
> - Work conserving - Do not let the CPU be idle if there are
> runnable tasks (even if that means running task-groups that
> are above their allowed limit)
>
> - SMP behavior - Limit to be enforced on all CPUs put together
>
> - Real-time tasks - Should be left alone as they are today?
> i.e real time tasks across groups should be scheduled as if
> they are in same group
>
> - Should cater to requirements of variety of workload characteristics,
> including bursty ones (?)
>
>
> Salient points about this patch:
>

> - Each task-group gets its own runqueue on every cpu.

how does that scale for, let's say 200-300 guests on a
'typical' dual CPU machine?

> - In addition, there is an active and expired array of
> task-groups themselves. Task-groups that have expired their
> quota are put into expired array.

how much overhead does that add to the scheduler, cpu
and memory wise?

> - Task-groups have priorities. Priority of a task-group is the
> same as the priority of the highest-priority runnable task it
> has. This I feel will retain interactiveness of the system
> as it is today.
>
> - Scheduling the next task involves picking highest priority task-group
> from active array first and then picking highest-priority task
> within it. Both steps are O(1).

how does that affect interactivity?

> - Token are assigned to task-groups based on their assigned
> quota. Once they run out of tokens, the task-group is put
> in an expired array. Array switch happens when active array
> is empty.
>
> - SMP load-balancing is accomplished on the lines of smpnice.

what about strict CPU limits (i.e. 20% regardless of
the idle state of the machine)

TIA,
Herbert

> Results of the patch
> ====================
>
> Machine : 2way x86_64 Intel Xeon (3.6 GHz) box
>
> Note: All test were forced to run on only one CPU using cpusets
>
> 1. Volanomark [1]
>
> ---
> Group A [50% limit] Group B [50% limit]
>

> Elapsed time 35.83 sec 36.6002
> Avg throughput 11179.3 msg/sec 10944.3 msg/sec
>
> ---
>
>
> ---
> Group A [80% limit] Group B [20% limit]
>
> Elapsed time 23.4466 sec 36.1857
> Avg throughput 17072 msg/sec 11080 msg/sec
>
> ---
>
> 2. Kernel compilation
>
>
> ---
> Group A [50% limit] Group B [50% limit]
> time -p make -j4 bzImage time -p make -j8 bzImage
>
> real 771.00 sec 769.08 sec
>
> ---
>
>
> ---
> Group A [80% limit] Group B [20% limit]
> time -p make -j4 bzImage time -p make -j8 bzImage
>
> real 484.12 sec 769.70 sec
>
> ---
>
>
>
> --
> Regards,
> vatsa

_____

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

---

Subject: Re: [RFC | PATCH 0/9] CPU controller over process container
Posted by Srivatsa Vaddagiri on Thu, 12 Apr 2007 18:19:28 GMT

On Thu, Apr 12, 2007 at 07:56:47PM +0200, Herbert Poetzl wrote:
> > - Each task-group gets its own runqueue on every cpu.
>
> how does that scale for, let's say 200-300 guests on a
> 'typical' dual CPU machine?

Scheduling complexity is still O(1) and hence I would say CPU-wise, it should be very scalable. Memory-wise, I agree that this can consume more memory if number of guests are large ..But this I feel is a memory vs cpu tradeoff ..If you had only one queue in which tasks from all groups were present, then it increases the schedule() complexity ?

If there are specific tests you had in mind to test this scalability aspect, I would be happy to run them.

> > - In addition, there is an active and expired array of
> > task-groups themselves. Task-groups that have expired their
> > quota are put into expired array.
>
> how much overhead does that add to the scheduler, cpu
> and memory wise?

cpu-wise, it should add very little overhead (since O(1) behavior is retained). memory-wise, same points as above.

> > - Scheduling the next task involves picking highest priority task-group
> > from active array first and then picking highest-priority task
> > within it. Both steps are O(1).
>
> how does that affect interactivity?

Note that I define task-group priority = highest priority tasks the group has, which IMO should give decent if not good interactivity ..But is (good) interactivity a big requirement here? As we know, that's a hard thing to achieve even in today's O(1) scheduler ..

> > - SMP load-balancing is accomplished on the lines of smpnice.
>
> what about strict CPU limits (i.e. 20% regardless of
> the idle state of the machine)

Not supported in these patches. Any idea how/where that would be usefull?

--
Regards,
vatsa

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