

[RFC] CPU controllers?

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-06/msg04343.html>

- *From:* Srivatsa Vaddagiri <vatsa@xxxxxxxxxx>
 - *Date:* Thu, 15 Jun 2006 19:16:32 +0530
-

Hello,

There have been several proposals so far on this subject and no consensus seems to have been reached on what an acceptable CPU controller for Linux needs to provide. I am hoping this mail will trigger some discussions in that regard. In particular I am keen to know what the various maintainers think about this subject.

The various approaches proposed so far are:

- CPU rate-cap (limit CPU execution rate per-task)
<http://lkml.org/lkml/2006/5/26/7>
- f-series CKRM controller (CPU usage guarantee for a task-group)
<http://lkml.org/lkml/2006/4/27/399>
- e-series CKRM controller (CPU usage guarantee/limit for a task-group)
<http://prdownloads.sourceforge.net/ckrm/cpu.ckrm-e18.v10.patch.gz?download>
- OpenVZ controller (CPU usage guarantee/hard-limit for a task-group)
<http://openvz.org/>
- vserver controller (CPU usage guarantee(?)/limit for a task-group)
<http://linux-vserver.org/>

(I apologize if I have missed any other significant proposal for Linux)

Their salient features and limitations/drawbacks, as I could gather, are summarized later below. To note is each controller varies in degree of complexity and addresses its own set of requirements.

In going forward for an acceptable controller in mainline it would help, IMHO, if we put together the set of requirements which the Linux CPU controller should support. Some questions that arise in this regard are:

- Do we need mechanisms to control CPU usage of tasks, further to what already exists (like nice)? IMO yes.
- What are the requirements of such a CPU controller? Some of them to consider are:

[RFC] CPU controllers?

- Should it operate on a per-task basis or on a per-task-group basis?
- Should it support more than one level of task-groups?
- If we want to allow on a per-task-group basis, which mechanism do we use for grouping tasks (Resource Groups, PAGG, uid/session id ..)?
- Should it support limit and guarantee both? In case of limit, should it support both soft and hard limit?
- What interface do we choose for user to specify limit/guarantee? system call or filesystem based (ex: /proc or Resource Group's rdfs)?
- Over what interval should guarantee/limit be monitored and controlled?
- With what accuracy should we allow the limit/guarantee to be expressed?
- Co-existence with CPUset – should guarantee/limit be enforced only on the set of CPUs attached to the cpuset?
- Should real-time tasks be outside the purview of this control?
- Load balance to be made aware of the guarantee/limit of tasks (or task-groups)? Ofcourse yes!

One possibility is to add a basic controller, that addresses some minimal requirements, to begin with and progressively enhance it capabilities. From this pov, both the f-series resource group controller and cpu rate-cap seem to be good candidates for a minimal controller to begin with.

Thoughts?

Salient features of various CPU controllers that have been proposed so far are summarized below. I have not captured OpenVZ and Vserver controller aspects well. Request the maintainers to fill-in!

1. CPU Rate Cap (by Peter Williams)

Features:

- * Limit CPU execution rate on a per-task basis.
- * Limit specified in terms of parts-per-thousand. Limit set thr' /proc interface.
- * Supports hard limit and soft limit
- * Introduces new task priorities where tasks that have exceeded their soft limit can be "parked" until the O(1) scheduler picks them for execution
- * Load balancing on SMP systems made aware of tasks whose execution rate is limited by this feature
- * Patch is simple

Limitations:

- * Does not support guarantee

[RFC] CPU controllers?

Drawbacks:

- * Limiting CPU execution rate of a group of tasks has to be tackled from an external module (user or kernel space) which may make this approach somewhat inconvenient to implement for task-groups.

2. Timeslice scaling (Maeda Naoaki and Kurosawa Takahiro)

Features:

- * Provide guaranteed CPU execution rate on a per-task-group basis
Guarantee provided over an interval of 5 seconds.
- * Hooked to Resource Group infrastructure currently and hence guarantee/limit set thr' Resource Group's RCFS interface.
- * Achieves guaranteed execution by scaling down timeslice of tasks who are above their guaranteed execution rate. Timeslice can be scaled down only to a minimum of 1 slice.
- * Does not scale down timeslice of interactive tasks (even if their CPU usage is beyond what is guaranteed) and does not avoid requeue of interactive tasks.
- * Patch is quite simple

Limitations:

- * Does not support limiting task-group CPU execution rate

Drawbacks:

(Some of the drawbacks listed are probably being addressed currently with a redesign – which we are yet to see)

- * Interactive tasks (and their requeuing) can come in the way of providing guaranteed execution rate to other tasks
- * SMP load balancing does not take into account guarantee provided to task groups.
- * It may not be possible to restrict CPU usage of a task group to only its guaranteed usage if the task-group has large number of tasks (each task is run for a minimum of 1 timeslice)
- * May not handle bursty loads

3. Resource Group e-series CPU controller

Features:

- * Provides both guarantee and limit for CPU execution rate of task groups (classes)
- * Two-level scheduling. Pick a task-group (class) to execute first and then a task within the task-group. Both are of O(1) complexity.
- * Classes are given priorities based on their guaranteed CPU usage, accumulated CPU execution and the highest priority task present within the group. Class with the highest priority picked up for execution next.
- * Guarantee/Limit specified in terms of shares

Drawbacks:

* Complexity

3. OpenVZ CPU controller

Features:

- Provides both guarantee [1] and (hard) limit for CPU execution rate of task group (containers)
- Multi-level scheduler (Pick a task-group to run first, then pick a virtual-cpu and then a task)
- Virtual cpu concept makes group-aware SMP load balancing easy
- Uses cycles (rather than ticks) consumed for accounting (?)

[1] - <http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf>

Limitations:

- ?

Drawbacks:

- ?

4. VServer CPU controller

Features:

- Token-bucket based

Drawbacks:

- ?

Limitations:

- ?

--

Regards,

vatsa

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>