Subject: Re: [RFC] Default child of a cgroup
Posted by Peter Zijlstra on Thu, 31 Jan 2008 20:37:42 GMT
View Forum Message <> Reply to Message

On Thu, 2008-01-31 at 23:39 +0530, Balbir Singh wrote:
> Srivatsa Vaddagiri wrote:
> > Hi,
> > As we were implementing multiple-hierarchy support for CPU
> > controller, we hit some oddities in its implementation, partly related
> > to current cgroups implementation. Peter and I have been debating on the
> > exact solution and I thought of bringing that discussion to lkml.
> >
> > Consider the cgroup filesystem structure for managing cpu resource.
> >
> > # mount -t cgroup -ocpu,cpuacct none /cgroup
> > # mkdir /cgroup/A
> > # mkdir /cgroup/B
> > # mkdir /cgroup/A/a1
> >
> > will result in:
> >
> > /cgroup
> > |------<tasks>
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |----[A]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[a1]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[B]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > |
> >
> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> > task-group in A/a1/tasks? In otherwords do they form siblings

> > of the same parent A?
> >
>
> I consider them to be the same relationship between directories and files.
> A/tasks are siblings of A/a1 and A/other children, *but* the entities of
> interest are A and A/a1.
>
> > 2. Somewhat related to the above question, how much resource should the
> > task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> > A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> > in A/tasks)?
> >
>
> I propose that it gets 1/2 of the bandwidth, here is why
>
> 1. Assume that a task in A/tasks forks 1000 children, what happens to the
> bandwidth of A/a1's tasks then? We have no control over how many tasks can be
> created on A/tasks as a consequence of moving one task to A/tasks. Doing it the
> other way would mean, that A/a1/tasks will get 1/1001 of the bandwidth (sounds
> very unfair and prone to Denial of Service/Fairness)

And I oppose this, it means not all siblings are treated equal. Also, I
miss the story of the 'hidden' group here. The biggest objection is this
hidden group with no direct controls.

My proposal is to make it a hard constraint, either a group has task
children or a group has group children, but not mixed. That keeps the
interface explicit and doesn't hide the tricks we play.

> > 3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
> > of all siblings put together? It can reflect only one, in which case
> > user has to manually derive the other component of the statistics.
> >
>
> It should reflect the accumulated usage of A's children and the tasks in A.

A's children includes tasks in this context. See where the confusion is?

> > It seems to me that tasks in A/tasks form what can be called the
> > "default" child group of A, in which case:
> >
> > 4. Modifications to A/cpu.shares should affect the parent or its default
> > child group (A/tasks)?
> >
> > To avoid these ambiguities, it may be good if cgroup create this
> > "default child group" automatically whenever a cgroup is created?
> > Something like below (not the absence of tasks file in some directories
> > now):

> >
>
> I think the concept makes sense, but creating a default child is going to be
> confusing, as it is not really a child of A.

Quite so. I really hate this hidden group.

> >
> > /cgroup
> > |
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |---[def_child]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[A]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>
> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> > | |
> > | |---[a1]
> > | | |
> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> > | | |---[def_child]
> > | | | |---<tasks>
> > | | | |---<cpuacct.usage>
> > | | | |---<cpu.shares>
> > | | | |
> > | |
> > |----[B]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>

> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> >
> > Note that user cannot create subdirectories under def_child with this
> > scheme! I am also not sure what impact this will have on other resources
> > like cpusets ..
> >
>
> Which means we'll need special logic in the cgroup filesystem to handle
> def_child. Not a very good idea.

agreed.

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