Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by William Lee Irwin III on Wed, 23 May 2007 18:03:16 GMT
View Forum Message <> Reply to Message

On Wed, May 23, 2007 at 10:18:59PM +0530, Srivatsa Vaddagiri wrote:
> Here's an attempt to extend CFS (v13) to be fair at a group level, rather than
> just at task level. The patch is in a very premature state (passes
> simple tests, smp load balance not supported yet) at this point. I am sending
> it out early to know if this is a good direction to proceed.

Well, SMP load balancing is what makes all this hard. sched_yield()
semantics are yet another twist.

On Wed, May 23, 2007 at 10:18:59PM +0530, Srivatsa Vaddagiri wrote:
> Salient points which needs discussion:
> 1. This patch reuses CFS core to achieve fairness at group level also.
> To make this possible, CFS core has been abstracted to deal with generic
> schedulable "entities" (tasks, users etc).

The ability to handle deeper hierarchies would be useful for those
who want such semantics.

On Wed, May 23, 2007 at 10:18:59PM +0530, Srivatsa Vaddagiri wrote:
> 2. The per-cpu rb-tree has been split to be per-group per-cpu.
> schedule() now becomes two step on every cpu : pick a group first (from
> group rb-tree) and a task within that group next (from that group's task
> rb-tree)

That assumes per-user scheduling groups; other configurations would
make it one step for each level of hierarchy. It may be possible to
reduce those steps to only state transitions that change weightings
and incremental updates of task weightings. By and large, one needs
the groups to determine task weightings as opposed to hierarchically
scheduling, so there are alternative ways of going about this, ones
that would even make load balancing easier.

On Wed, May 23, 2007 at 10:18:59PM +0530, Srivatsa Vaddagiri wrote:
> 3. Grouping mechanism - I have used 'uid' as the basis of grouping for
> timebeing (since that grouping concept is already in mainline today).
> The patch can be adapted to a more generic process grouping mechanism
> (like http://lkml.org/lkml/2007/4/27/146) later.

I'd like to see how desirable the semantics achieved by reflecting
more of the process hierarchy structure in the scheduler groupings are.
Users, sessions, pgrps, and thread_groups would be the levels of

hierarchy there, where some handling of orphan pgrps is needed.

On Wed, May 23, 2007 at 10:18:59PM +0530, Srivatsa Vaddagiri wrote:
> Some results below, obtained on a 4way (with HT) Intel Xeon box. All
> number are reflective of single CPU performance (tests were forced to
> run on single cpu since load balance is not yet supported).
> uid "vatsa" uid "guest"
> (make -s -j4 bzImage) (make -s -j20 bzImage)
>
> 2.6.22-rc1 772.02 sec 497.42 sec (real)
> 2.6.22-rc1+cfs-v13 780.62 sec 478.35 sec (real)
> 2.6.22-rc1+cfs-v13+this patch 776.36 sec 776.68 sec (real)
> [An exclusive cpuset containing only one CPU was created and the
> compilation jobs of both users were run simultaneously in this cpuset]
> I also disabled CONFIG_FAIR_USER_SCHED and compared the results with
> cfs-v13:
> uid "vatsa"
> make -s -j4 bzImage
>
> 2.6.22-rc1+cfs-v13 395.57 sec (real)
> 2.6.22-rc1+cfs-v13+this_patch 388.54 sec (real)
> There is no regression I can see (rather some improvement, which I can't
> understand atm). I will run more tests later to check this regression aspect.
> Request your comments on the future direction to proceed!

Kernel compiles are markedly poor benchmarks. Try lat_ctx from lmbench,
VolanoMark, AIM7, OAST, SDET, and so on.

-- wli