

On Wed, May 30, 2007 at 01:13:59PM -0700, William Lee Irwin III wrote:
> On Wed, May 30, 2007 at 10:44:05PM +0530, Srivatsa Vaddagiri wrote:
> > Hmm ..so do you think this weight decomposition can be used to flatten
> > the tree all the way to a single level in case of cfs? That would mean we can
> > achieve group fairness with single level scheduling in cfs ..I am
> > somewhat skeptical that we can achieve group fairness with a single
> > level rb-tree (and w/o substantial changes to pick_next_task logic in cfs
> > that is), but if it can be accomplished would definitely be a great win.
>
> Yes, the hierarchy can be flattened completely and global task weights
> computed and used to achieve group fairness.

ok, lets say we are are considering a hierarchy of user->process->thread as below:

Users = {U1, U2, U3}

where process in a user are:

U1 = {P0, P1, P2, P3, P4}
U2 = {P5, P6}
U3 = {P7}

and where threads/tasks in a process are:

P0 = {T0, T1}
P1 = {T2}
P2 = {T3, T4, T5}
P3 = {T6}
P4 = {T8}
P5 = {T9, T10}
P6 = {T11}
P7 = {T14, T15, T16}

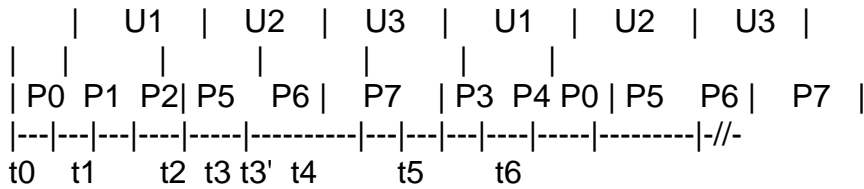
If we need to achieve group fairness given single-level hierarchy,
then tasks need to be spread out in rb-tree like something below?

U1 U2 U3 U1 U2 U3
|-----|-----|-----|-----|-----|-----|--- // ---
t0 t1 t2 t3 t4 t5 t6

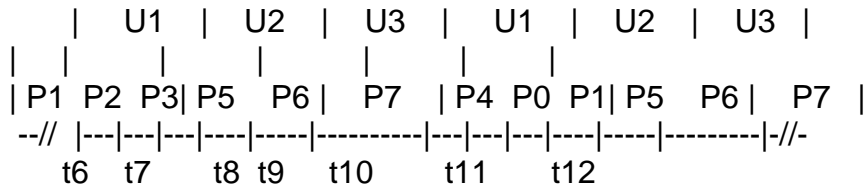
[t0, t1 ..tN are equally spaced points in time. t1 = t0 + K, t2 = t1 + K ..]

Viewed at the top hierarchy level (users) tasks are spread such that each user gets "equal" execution over some interval (lets say b/n t0-t3).

When viewed at the next lower hierarchy level (processes), it should look like:

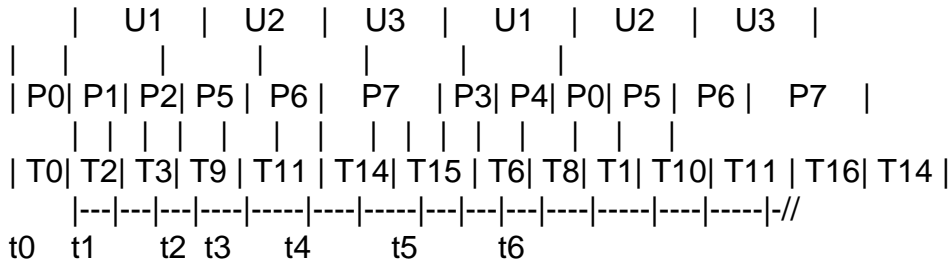


[contd below ..]

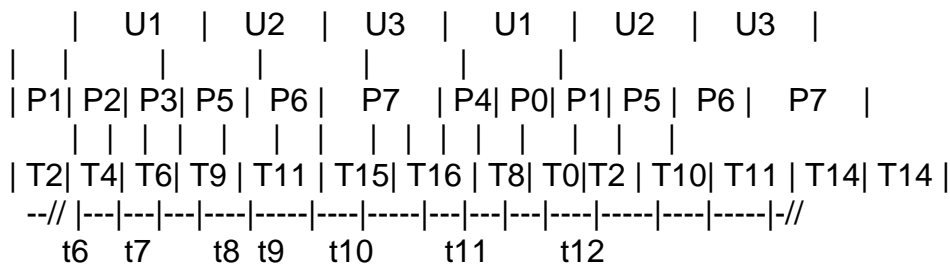


The available bandwidth to a user is dividided "equally" between various processes of the user over some time (say between t0 - t3').

When viewed at the next lower hierarchy level (threads), it should look like:



(continuting below)



Available bandwidth to a process is divided "equally" between threads of the process.

Although I have been using "equally" everywhere above, it needs to take into account relative importance of tasks/processes/users.

> The changes aren't to pick_next_task() but rather to the ->fair_key
> computations.

Thats the \$\$ question I guess :) What computation of ->fair_key can we use such that task execution sequence is (from the above example):

T0, T2, T3, T9, T11, T14, T15, T6, T8, T1, T10, T11, T16, T14 ...

?

Of course, this ->fair_key computation should default to what it is today when hierarchical res mgmt is disabled?

> In fact, I went a step beyond that.

>

>

> On Sat, May 26, 2007 at 08:41:12AM -0700, William Lee Irwin III wrote:

> >> In such a manner nice numbers obey the principle of least surprise.

>

> The step beyond was to show how nice numbers can be done with all that
> hierarchical task grouping so they have global effects instead of
> effects limited to the scope of the narrowest grouping hierarchy
> containing the task. I had actually assumed the weighting and
> flattening bits were already in your plans from some other post you
> made and was building upon that.

I would definitely be willing to try out any experiments you think of, esp those that allow the hierarchy to be flattened. atm fair_key calculation (in the context of cfs) seem to be the biggest challenge to surmount for this to work.

--

Regards,
vatsa

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