

---

Subject: Re: [ckrm-tech] [RFC] [PATCH 0/3] Add group fairness to CFS  
Posted by [William Lee Irwin III](#) on Thu, 31 May 2007 08:43:29 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, May 30, 2007 at 11:36:47PM -0700, William Lee Irwin III wrote:  
>> Temporarily, yes. All this only works when averaged out.

On Thu, May 31, 2007 at 02:03:53PM +0530, Srivatsa Vaddagiri wrote:  
> So essentially when we calculate delta\_mine component for each of those  
> 1000 tasks, we will find that it has executed for 1 tick (4 ms say) but  
> its fair share was very very low.  
> fair\_share = delta\_exec \* p->load\_weight / total\_weight  
> If p->load\_weight has been calculated after factoring in hierarchy (as  
> you outlined in a previous mail), then p->load\_weight of those 1000 tasks  
> will be far less compared to the p->load\_weight of one task belonging to  
> other user, correct? Just to make sure I get all this correct:

You've got it all correct.

On Thu, May 31, 2007 at 02:03:53PM +0530, Srivatsa Vaddagiri wrote:  
> User U1 has tasks T0 - T999  
> User U2 has task T1000  
> assuming each task's weight is 1 and each user's weight is 1 then:  
>  $WT0 = (WU1 / WU1 + WU2) * (WT0 / WT0 + WT1 + \dots + WT999)$   
>  $= (1 / 1 + 1) * (1 / 1000)$   
>  $= 1/2000$   
>  $= 0.0005$   
> WT1 ..WT999 will be same as WT0  
> whereas, weight of T1000 will be:  
>  $WT1000 = (WU1 / WU1 + WU2) * (WT1000 / WT1000)$   
>  $= (1 / 1 + 1) * (1/1)$   
>  $= 0.5$   
> ?

Yes, these calculations are correct.

On Thu, May 31, 2007 at 02:03:53PM +0530, Srivatsa Vaddagiri wrote:  
> So when T0 (or T1 ..T999) executes for 1 tick (4ms), their fair share would  
> be:  
> T0's fair\_share (delta\_mine)  
>  $= 4 \text{ ms} * 0.0005 / (0.0005 * 1000 + 0.5)$   
>  $= 4 \text{ ms} * 0.0005 / 1$   
>  $= 0.002 \text{ ms} (2000 \text{ ns})$   
> This would cause T0's ->wait\_runtime to go negative sharply, causing it to be  
> inserted back in rb-tree well ahead in future. One change I can foresee  
> in CFS is with regard to limit\_wait\_runtime() ..We will have to change

> its default limit, atleast when group fairness thingy is enabled.  
> Compared to this when T1000 executes for 1 tick, its fair share would be  
> calculated as:  
> T1000's fair\_share (delta\_mine)  
> =  $4 \text{ ms} * 0.5 / (0.0005 * 1000 + 0.5)$   
> =  $4 \text{ ms} * 0.5 / 1$   
> = 2 ms (2000000 ns)  
> Its ->wait\_runtime will drop less significantly, which lets it be  
> inserted in rb-tree much to the left of those 1000 tasks (and which indirectly  
> lets it gain back its fair share during subsequent schedule cycles).

This analysis is again entirely correct.

On Thu, May 31, 2007 at 02:03:53PM +0530, Srivatsa Vaddagiri wrote:

> Hmm ..is that the theory?  
> Ingo, do you have any comments on this approach?  
> /me is tempted to try this all out.

Yes, this is the theory behind using task weights to flatten the task group hierarchies. My prior post assumed all this and described a method to make nice numbers behave as expected in the global context atop it.

-- wli

---

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

---