Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Guillaume Chazarain on Wed, 23 May 2007 18:12:12 GMT
View Forum Message <> Reply to Message

Hi,

> 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)

Impressive numbers :-)

Testing this in qemu/UP/i386, I had to do this:

```
--- linux/kernel/sched_fair.c
+++ linux/kernel/sched_fair.c
@@ -350,9 +350,10 @@

  if (p->wait_start_fair) {
   delta_fair = lrq->fair_clock - p->wait_start_fair;
-  if (unlikely(p->load_weight != lrq->nice_0_load))
-   delta_fair = (delta_fair * p->load_weight) /
-       lrq->nice_0_load;
+  if (unlikely(p->load_weight != lrq->nice_0_load)) {
+   s64 m = delta_fair * p->load_weight;
+   delta_fair = do_div(m, lrq->nice_0_load);
+  }
   add_wait_runtime(lrq, p, delta_fair);
  }
```

to make it compile, otherwise it ends with:
kernel/built-in.o: In function `update_stats_wait_end':
/home/g/linux-group-fair/linux-2.6.21-rc1-cfs-v13-fair/kernel/sched_fair.c:354:
undefined reference to `__divdi3'
/home/g/linux-group-fair/linux-2.6.21-rc1-cfs-v13-fair/kernel/sched_fair.c:354:
undefined reference to `__divdi3'

Some observations:

o Doing an infinite loop as root seems to badly affect interactivity
much more than with a normal user. Note that this is subjective, so
maybe I'm smocking crack here.

o Nice values are not reflected across users. From my test, if user1

has a single busy loop at nice 19, and user2 a single busy loop at
nice 0, both process will have a 50% CPU share, this looks wrong. Note
that I have no idea how to solve this one.

Thanks for working in this very interesting direction.

--
Guillaume

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

---

Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Ingo Molnar on Wed, 23 May 2007 18:38:24 GMT
View Forum Message <> Reply to Message

* Guillaume Chazarain <guichaz@yahoo.fr> wrote:

> Some observations:
>
> o Doing an infinite loop as root seems to badly affect interactivity
> much more than with a normal user. Note that this is subjective, so
> maybe I'm smocking crack here.

hm, this shouldnt be the case. Can you see this with -v14?

> o Nice values are not reflected across users. From my test, if user1
> has a single busy loop at nice 19, and user2 a single busy loop at
> nice 0, both process will have a 50% CPU share, this looks wrong. Note
> that I have no idea how to solve this one.

for containers it's exactly the right behavior: group scheduling is
really a 'super' container concept that allows the allocation of CPU
time regardless of how a group uses it. The only additional control we
might want is to allocate different amount of CPU time to different
groups. (i.e. a concept vaguely similar to "nice levels", but at the
group level - using a different and saner API than nice levels.) Nice
levels are really only meaningful at the lowest level.

for 'friendly users' it's perhaps not what we want - but those do not
need to isolate themselves from each other anyway.

> Thanks for working in this very interesting direction.

seconded :)

Ingo

_____

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

Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Guillaume Chazarain on Wed, 23 May 2007 22:26:16 GMT
View Forum Message <> Reply to Message

> hm, this shouldnt be the case. Can you see this with -v14?

The group fairness patches fail to apply over -v14. Just to make it clearer, I am not talking about vanilla CFS but group fairness CFS, I have no problems with CFS.

The symptoms in the qemu box are:
o two busy loops with UID=1 remove a bit of interactivity only to UID=1 tasks
o two busy loops with UID=0 make the system unusable

Another tidbit: /proc/sched_debug now deals with users, so the task and PID fields show some random memory garbage.

My do_div patch was nonsense (do_div returns the remainder, not the quotient). Attached is a corrected patch.

> for containers it's exactly the right behavior: group scheduling is
> really a 'super' container concept that allows the allocation of CPU
> time regardless of how a group uses it. The only additional control we
> might want is to allocate different amount of CPU time to different
> groups. (i.e. a concept vaguely similar to "nice levels", but at the
> group level - using a different and saner API than nice levels.) Nice
> levels are really only meaningful at the lowest level.

OK for containers, but then this should not replace the standard nice implementation as this CPU repartition would be unexpected IMHO. I would expect the group CPU allocation to be averaged from the nice levels of its elements.

As a sidenote, while in CFS-v13 a nice=0 tasks seems to get 10x more CPU than a nice=10 one, with the group fairness patch, the ratio drops to less than 2x (for tasks with the same UID).

Regards.

--
Guillaume

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

Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Srivatsa Vaddagiri on Thu, 24 May 2007 17:13:11 GMT
View Forum Message <> Reply to Message

On Thu, May 24, 2007 at 12:26:16AM +0200, Guillaume Chazarain wrote:
> As a sidenote, while in CFS-v13 a nice=0 tasks seems to get 10x more CPU
> than a nice=10 one, with the group fairness patch, the ratio drops to
> less than 2x (for tasks with the same UID).

Thanks for reporting this regression. I am investigating this currently.
Hope to send out a patch at the earliest ..

--
Regards,
vatsa

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

Subject: Re: [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Srivatsa Vaddagiri on Fri, 25 May 2007 07:45:00 GMT
View Forum Message <> Reply to Message

On Thu, May 24, 2007 at 12:26:16AM +0200, Guillaume Chazarain wrote:
> As a sidenote, while in CFS-v13 a nice=0 tasks seems to get 10x more CPU
> than a nice=10 one, with the group fairness patch, the ratio drops to
> less than 2x (for tasks with the same UID).

gah ..silly me.

Can you repeat your tests with this patch pls? With the patch applied, I am
now getting the same split between nice 0 and nice 10 task as CFS-v13
provides (90:10 as reported by top)

```
 5418 guest     20   0  2464  304  236 R   90  0.0   5:41.40 3 hog
 5419 guest     30  10  2460  304  236 R   10  0.0   0:43.62 3 nice10hog
```

Fix a stupid bug, where I was not calling __check_preempt_curr_fair()
at task level during task_tick ..

Signed-off-by : Srivatsa Vaddagiri <vatsa@in.ibm.com>

```
diff -puN kernel/sched_fair.c~fix kernel/sched_fair.c
--- linux-2.6.22-rc1-cfs-group/kernel/sched_fair.c~fix 2007-05-25 12:28:52.000000000 +0530
+++ linux-2.6.22-rc1-cfs-group-vatsa/kernel/sched_fair.c 2007-05-25 12:30:06.000000000 +0530
@@ -577,11 +577,12 @@ static void entity_tick(struct lrq *lrq,
        *n = task_entity(next);

   if ((c == lrq->rq->idle) || (rt_prio(n->prio) &&
-      (n->prio < c->prio)))
+      (n->prio < c->prio))) {
    resched_task(c);
- } else
-  __check_preempt_curr_fair(lrq, next, curr,
-       *(lrq->sched_granularity));
+   return;
+  }
+ }
+ __check_preempt_curr_fair(lrq, next, curr, *(lrq->sched_granularity));
 }

 static void _update_load(struct lrq *this_rq)
_
```

--
Regards,
vatsa