Subject: Re: [RFC][PATCH 3/6] core changes in CFS
Posted by Srivatsa Vaddagiri on Tue, 12 Jun 2007 04:22:47 GMT
View Forum Message <> Reply to Message

On Tue, Jun 12, 2007 at 07:59:22AM +0530, Balbir Singh wrote:
> > +#define entity_is_task(se) 1
>
> Could you add some comments as to what this means?

sure. Basically this macro tests whether a given schedulable entity is
task or not. Other possible schedulable entities could be process, user,
container etc. These various entities form a hierarchy with task being
at the bottom of the hierarchy.

> Should be it boolean instead (true)

I don't have a good opinion on this. Would it make sparse friendly?

> > + * Enqueue a entity into the rb-tree:
>
> Enqueue an entity

yes

>
> > -static void limit_wait_runtime(struct rq *rq, struct task_struct *p)
> > +static void limit_wait_runtime(struct lrq *lrq, struct sched_entity *p)
>
> p is a general convention for tasks in the code, could we use something
> different -- may be "e"?

'se' perhaps as is used elsewhere. I avoided making that change so that
people will see less diff o/p in the patch :) I agree though a better
name is needed.

> > static s64 div64_s(s64 divident, unsigned long divisor)
> > @@ -183,49 +219,51 @@
> >   * Update the current task's runtime statistics. Skip current tasks that
> >   * are not in our scheduling class.
> >   */
> > -static inline void update_curr(struct rq *rq, u64 now)
> > +static inline void update_curr(struct lrq *lrq, u64 now)
> > {
> > - unsigned long load = rq->lrq.raw_weighted_load;
> > + unsigned long load = lrq->raw_weighted_load;
> >   u64 delta_exec, delta_fair, delta_mine;
> > - struct task_struct *curr = rq->curr;
> > + struct sched_entity *curr = lrq_curr(lrq);

>
> How about curr_entity?

I prefer its current name, but will consider your suggestion in next
iteration.

> > + struct rq *rq = lrq_rq(lrq);
> > + struct task_struct *curtask = rq->curr;
> >
> > - if (curr->sched_class != &fair_sched_class || curr == rq->idle || !load)
> > + if (!curr || curtask == rq->idle || !load)
>
> Can !curr ever be true? shoudn't we look into the sched_class of the task
> that the entity belongs to?

Couple of cases that we need to consider here:

CONFIG_FAIR_GROUP_SCHED disabled:

lrq_curr() essentially returns NULL if currently running task
doesnt belong to fair_sched_class, else it returns &rq->curr->se
So the check for fair_sched_class is taken care in that
function.

CONFIG_FAIR_GROUP_SCHED enabled:

lrq_curr() returns lrq->curr. I introduced ->curr field in lrq
to optimize on not having to update lrq's fair_clock
(update_curr upon enqueue/dequeue task) if it was not currently
"active".

Lets say that there are two groups 'vatsa' and 'guest'
with their own lrqs on each cpu. If CPU0 is currently running a
task from group 'vatsa', then lrq_vatsa->curr will point to
the currently running task, while lrq_guest->curr will be
NULL. While the task from 'vatsa' is running, if we were to
enqueue/dequeue task from group 'guest', we need not
update lrq_guest's fair_clock (as it is not active currently).
This optimization in update_curr is made possible by maintaining
a 'curr' field in lrq.

Hope this answers your question.

--
Regards,
vatsa

_____
Containers mailing list