Subject: Re: [ckrm-tech] [RFC] [PATCH 0/3] Add group fairness to CFS
Posted by Peter Williams on Wed, 30 May 2007 00:09:28 GMT
View Forum Message <> Reply to Message

William Lee Irwin III wrote:
> William Lee Irwin III wrote:
>>> Lag should be considered in lieu of load because lag
>
> On Sun, May 27, 2007 at 11:29:51AM +1000, Peter Williams wrote:
>> What's the definition of lag here?
>
> Lag is the deviation of a task's allocated CPU time from the CPU time
> it would be granted by the ideal fair scheduling algorithm (generalized
> processor sharing; take the limit of RR with per-task timeslices
> proportional to load weight as the scale factor approaches zero).

Over what time period does this operate?

> Negative lag reflects receipt of excess CPU time. A close-to-canonical
> "fairness metric" is the maximum of the absolute values of the lags of
> all the tasks on the system. The "signed minimax pseudonorm" is the
> largest lag without taking absolute values; it's a term I devised ad
> hoc to describe the proposed algorithm.

So what you're saying is that you think dynamic priority (or its
equivalent) should be used for load balancing instead of static priority?

>
> William Lee Irwin III wrote:
>>> is what the
>>> scheduler is trying to minimize;
>
> On Sun, May 27, 2007 at 11:29:51AM +1000, Peter Williams wrote:
>> This isn't always the case. Some may prefer fairness to minimal lag.
>> Others may prefer particular tasks to receive preferential treatment.
>
> This comment does not apply. Generalized processor sharing expresses
> preferential treatment via weighting. Various other forms of
> preferential treatment require more elaborate idealized models.

This was said before I realized that your "lag" is just a measure of
fairness.

>
>
>>> load is not directly relevant, but
>>> appears to have some sort of relationship. Also, instead of pinned,
>>> unpinned should be considered.

>
> On Sun, May 27, 2007 at 11:29:51AM +1000, Peter Williams wrote:
>> If you have total and pinned you can get unpinned. It's probably
>> cheaper to maintain data for pinned than unpinned as there's less of it
>> on normal systems.
>
> Regardless of the underlying accounting,

I was just replying to your criticism of my suggestion to keep pinned
task statistics and use them.

> I've presented a coherent
> algorithm. It may be that there's no demonstrable problem to solve.
> On the other hand, if there really is a question as to how to load
> balance in the presence of tasks pinned to cpus, I just answered it.

Unless I missed something there's nothing in your suggestion that does
anything more about handling pinned tasks than is already done by the
load balancer.

>
>
> William Lee Irwin III wrote:
>>> Using the signed minimax pseudonorm (i.e. the highest
>>> signed lag, where positive is higher than all negative regardless of
>>> magnitude) on unpinned lags yields a rather natural load balancing
>>> algorithm consisting of migrating from highest to lowest signed lag,
>>> with progressively longer periods for periodic balancing across
>>> progressively higher levels of hierarchy in sched_domains etc. as usual.
>>> Basically skip over pinned tasks as far as lag goes.
>>> The trick with all that comes when tasks are pinned within a set of
>>> cpus (especially crossing sched_domains) instead of to a single cpu.
>
> On Sun, May 27, 2007 at 11:29:51AM +1000, Peter Williams wrote:
>> Yes, this makes the cost of maintaining the required data higher which
>> makes keeping pinned data more attractive than unpinned.
>> BTW keeping data for sets of CPU affinities could cause problems as the
>> number of possible sets is quite large (being 2 to the power of the
>> number of CPUs). So you need an algorithm based on pinned data for
>> single CPUs that knows the pinning isn't necessarily exclusive rather
>> than one based on sets of CPUs. As I understand it (which may be
>> wrong), the mechanism you describe below takes that approach.
>
> Yes, the mechanism I described takes that approach.
>
>
> William Lee Irwin III wrote:
>>> The smpnice affair is better phrased in terms of task weighting. It's

>>> simple to honor nice in such an arrangement. First unravel the
>>> grouping hierarchy, then weight by nice. This looks like
> [...]
>>> In such a manner nice numbers obey the principle of least surprise.
>
> On Sun, May 27, 2007 at 11:29:51AM +1000, Peter Williams wrote:
>> Is it just me or did you stray from the topic of handling cpu affinity
>> during load balancing to hierarchical load balancing? I couldn't see
>> anything in the above explanation that would improve the handling of cpu
>> affinity.
>
> There was a second issue raised to which I responded. I didn't stray
> per se. I addressed a second topic in the post.

OK.

To reiterate, I don't think that my suggestion is really necessary. I think that the current load balancing (stand fast a small bug that's being investigated) will come up with a good distribution of tasks to CPUs within the constraints imposed by any CPU affinity settings.

Peter
--
Peter Williams   pwil3058@bigpond.net.au

"Learning, n. The kind of ignorance distinguishing the studious."
  -- Ambrose Bierce