

Re: VST and Sched Load Balance

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-05/0945.html>

From: Nick Piggin (nickpiggin_at_yahoo.com.au)

Date: 05/05/05

Date: Fri, 06 May 2005 00:52:48 +1000

To: vatsa@in.ibm.com

Srivatsa Vaddagiri wrote:

> On Thu, Apr 07, 2005 at 11:07:55PM +1000, Nick Piggin wrote:

>>I think we do care, yes. It could be pretty harmful to sleep for
>>even a few 10s of ms on a regular basis for some workloads. Although
>>I guess many of those will be covered by `try_to_wake_up` events...
>>

>>Not sure in practice, I would imagine it will hurt some multiprocessor
>>workloads.

>

>

> I am looking at the recent changes in load balance and I see that load
> balance on fork has been introduced (`SD_BALANCE_FORK`). I think this changes
> the whole scenario.

>

> Considering the fact that there was already balance on `wake_up` and the
> fact that the scheduler checks for imbalance before running the idle task
> (`load_balance_newidle`), I don't know if sleeping idle CPUs can cause a
> load imbalance (fork/wakeup happening on other CPUs will probably push
> tasks to it and wake it up anyway? exits can change the balance, but probably
> is not relevant here?)
>

>

Well, there are a lot of ifs and buts. Some domains won't implement
fork balancing, others won't do newidle balancing, wake balancing,
wake to idle, etc etc.

I think my idea of allowing `max_interval` to be extended to a
sufficiently large value if one CPU goes idle, and shutting off all
CPU's rebalancing completely if no tasks are running for some time
should cater to both hypervisor images and power saving concerns.
While not being very intrusive to normal operation of the scheduler.

> Except for a small fact: if the CPU sleeps w/o taking `rebalance_ticks`,
> its `cpu_load[]` can become incorrect over a period.

>

> I noticed that `load_balance_newidle` uses `newidle_idx` to gauge the current cpu's

Linux-Kernel: Re: VST and Sched Load Balance

- > load. As a result, it can see non-zero load for the idle cpu. Because of this
- > it can decide to not pull tasks.
- >
- > The rationale here (of using non-zero load): is it to try and let the
- > cpu become idle? Somehow, this doesn't make sense, because in the very next
- > rebalance_tick (assuming that the idle cpu does not sleep), it will start using
- > the idle_idx, which will cause the load to show up as zero and can cause the
- > idle CPU to pull some tasks.
- >
- > Have I missed something here?
- >

No. I think you are right in that we'll want to make sure cpu_load is zero before cutting the timer tick.

- > Anyway, if the idle cpu were to sleep instead, the next rebalance_tick will
- > not happen and it will not pull the tasks to restore load balance.
- >
- > If my above understanding is correct, I see two potential solutions for this:
- >
- >
- > A. Have load_balance_newidle use zero load for current cpu while
- > checking for busiest cpu.
- > B. Or, if we want to retain load_balance_newidle the way it is, have
- > the idle thread call back scheduler to zero the load and retry
- > load balance, _when_ it decides that it wants to sleep (there
- > are conditions under which a idle cpu may not want to sleep. for ex:
- > the next timer is only a tick, 1ms, away).
- >
- > In either case, if the load balance still fails to pull any tasks, then it means
- > there is really no imbalance. Tasks that will be added into the system later
- > (fork/wake_up) will be balanced across the CPUs because of the load-balance
- > code that runs during those events.
- >
- > A possible patch for B follows below:
- >

Yeah something like that should do it.

--

SUSE Labs, Novell Inc.

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>