Subject: Re: [PATCH RFC] sched: boost throttled entities on wakeups
Posted by Vladimir Davydov on Thu, 18 Oct 2012 10:39:01 GMT
View Forum Message <> Reply to Message

There is an error in the test script: I forgot to initialize cpuset.mems of test cgroups - without it it is impossible to add a task into a cpuset cgroup.

Sorry for that.

Fixed version of the test script is attached.

On Oct 18, 2012, at 11:32 AM, Vladimir Davydov wrote:

> If several tasks in different cpu cgroups are contending for the same resource
> (e.g. a semaphore) and one of those task groups is cpu limited (using cfs
> bandwidth control), the priority inversion problem is likely to arise: if a cpu
> limited task goes to sleep holding the resource (e.g. trying to take another
> semaphore), it can be throttled (i.e. removed from the runqueue), which will
> result in other, perhaps high-priority, tasks waiting until the low-priority
> task continues its execution.
>
> The patch tries to solve this problem by boosting tasks in throttled groups on
> wakeups, i.e. temporarily unthrottling the groups a woken task belongs to in
> order to let the task finish its execution in kernel space. This obviously
> should eliminate the priority inversion problem on voluntary preemptable
> kernels. However, it does not solve the problem for fully preemptable kernels,
> although I guess the patch can be extended to handle those kernels too (e.g. by
> boosting forcibly preempted tasks thus not allowing to throttle).
>
> I wrote a simple test that demonstrates the problem (the test is attached). It
> creates two cgroups each of which is bound to exactly one cpu using cpusets,
> sets the limit of the first group to 10% and leaves the second group unlimited.
> Then in both groups it starts processes reading the same (big enough) file
> along with a couple of busyloops in the limited groups, and measures the read
> time.
>
> I've run the test 10 times for a 1 Gb file on a server with > 10 Gb of RAM and
> 4 cores x 2 hyperthreads (the kernel was with CONFIG_PREEMPT_VOLUNTARY=y). Here
> are the results:
>
> without the patch 40.03 +- 7.04 s
> with the patch   8.42 +- 0.48 s
>
> (Since the server's RAM can accommodate the whole file, the read time was the
> same for both groups)
>
> I would appreciate if you could answer the following questions regarding the
> priority inversion problem and the proposed approach:

```
>
> 1) Do you agree that the problem exists and should be sorted out?
>
> 2) If so, does the general approach proposed (unthrottling on wakeups) suits
> you? Why or why not?
>
> 3) If you think that the approach proposed is sane, what you dislike about the
> patch?
>
> Thank you!
>
> ---
> include/linux/sched.h   |   8 ++
> kernel/sched/core.c     |   8 ++
> kernel/sched/fair.c     | 182 +++++++++++++++++++++++++++++++++++++++++++++++-
> kernel/sched/features.h |   2 +
> kernel/sched/sched.h    |   6 ++
> 5 files changed, 204 insertions(+), 2 deletions(-)
>
> <sched-boost-throttled-entities-on-wakeups.patch><ioprio_inv_test.sh ><ATT00001.c>
```

File Attachments

1) ioprio_inv_test.sh, downloaded 1590 times