Subject: Re: [PATCH 1/1, v7] cgroup/freezer: add per freezer duty ratio control
Posted by jacob.jun.pan on Wed, 16 Feb 2011 18:11:42 GMT
View Forum Message <> Reply to Message

On Tue, 15 Feb 2011 11:18:57 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> On Mon, 14 Feb 2011 15:07:30 -0800
> Andrew Morton <akpm@linux-foundation.org> wrote:
>
> > On Sun, 13 Feb 2011 19:23:10 -0800
> > Arjan van de Ven <arjan@linux.intel.com> wrote:
> >
> > > On 2/13/2011 4:44 PM, KAMEZAWA Hiroyuki wrote:
> > > > On Sat, 12 Feb 2011 15:29:07 -0800
> > > > Matt Helsley<matthltc@us.ibm.com> wrote:
> > > >
> > > >> On Fri, Feb 11, 2011 at 11:10:44AM -0800,
> > > >> jacob.jun.pan@linux.intel.com wrote:
> > > >>> From: Jacob Pan<jacob.jun.pan@linux.intel.com>
> > > >>>
> > > >>> Freezer subsystem is used to manage batch jobs which can start
> > > >>> stop at the same time. However, sometime it is desirable to
> > > >>> let the kernel manage the freezer state automatically with a
> > > >>> given duty ratio.
> > > >>> For example, if we want to reduce the time that backgroup apps
> > > >>> are allowed to run we can put them into a freezer subsystem
> > > >>> and set the kernel to turn them THAWED/FROZEN at given duty
> > > >>> ratio.
> > > >>>
> > > >>> This patch introduces two file nodes under cgroup
> > > >>> freezer.duty_ratio_pct and freezer.period_sec
> > > >> Again: I don't think this is the right approach in the long
> > > >> term. It would be better not to add this interface and instead
> > > >> enable the cpu cgroup subsystem for non-rt tasks using a
> > > >> similar duty ratio concept..
> > > >>
> > > >> Nevertheless, I've added some feedback on the code for you
> > > >> here :).
> > > >>
> > > > AFAIK, there was a work for bandwidth control in CFS.
> > > >
> > > > http://linux.derkeiler.com/Mailing-Lists/Kernel/2010-10/msg0 4335.html
> > > >
> > > > I tested this and worked fine. This schduler approach seems
> > > > better for my purpose to limit bandwidth of apprications rather
> > > > than freezer.
> > >

> > > for our purpose, it's not about bandwidth.
> > > it's about making sure the class of apps don't run for a long
> > > period (30-second range) of time.
> > >
> >
> > The discussion about this patchset seems to have been upside-down:
> > lots of talk about a particular implementation, with people walking
> > back from the implemetnation trying to work out what the
> > requirements were, then seeing if other implementations might suit
> > those requirements. Whatever they were.
> >
> > I think it would be helpful to start again, ignoring (for now) any
> > implementation.
> >
> >
> > What are the requirements here, guys? What effects are we actually
> > trying to achieve? Once that is understood and agreed to, we can
> > think about implementations.
> >
> >
> > And maybe you people _are_ clear about the requirements. But I'm
> > not and I'm sure many others aren't too. A clear statement of them
> > would help things along and would doubtless lead to better code.
> > This is pretty basic stuff!
> >
>
> Ok, my(our) reuquirement is mostly 2 requirements.
>
> - control batch jobs.
> - control kvm and limit usage of cpu.
>
> Considering kvm, we need to allow putting intaractive jobs and
> batch jobs onto a cpu. This will be difference in requirements.
> We need some latency sensitive control and static guarantee in
> peformance limit. For example, when a user limits a process to use
> 50% of cpu. Checks cpu usage by 'top -d 1', and should see almost
> '50%' value.
>
>
> IIUC, freezer is like a system to deliver SIGSTOP. set tasks as
> TASK_UNINTERRUPTIBLE and make them sleep. This check is done at
> places usual signal-check and some hooks in kernel threads.
> This means the subsystem checks all threads one by one and set flags,
> make them TASK_UNINTERRUPTIBLE finally when them wakes up.
> So, sleep/wakeup cost depeneds on the number of tasks and a task may
> not be freezable until it finds hooks of try_to_freeze().
>
> I hear when using FUSE, a task may never freeze if a process for FUSE

> operation is freezed before it freezes. This sounds freezer cgroup is
> not easy to use.
>
> CFS+bandwidh is a scheduler.
> It removes a sub scheduler entity from a tree when it exceeds allowed
> time slice. The cost of calculation of allowed time slice is involved
> in scheduler but I think it will not be too heavy. (Because
> MAINTAINERS will see what's going on and they are sensitive to the
> cost.) Tasks are all RUNNABLE. A task in group releases cpu when it
> see 'reschedule' flag. We have plenty of hooks of cond_resched().
> (And we know we tries to change spin_lock to mutex if spin_lock is
> huge cost)
>
> This will show a good result of perofmance even with 'top -d 1'.
> We'll not see TASK_RUNNING <-> TASK_INTERRUPTIBLE status change. And
> I think we can make period of time slice smaller than using freezer
> for better latency.
>

Thanks for the info. I will give it a try in my setup and get back to
you all.

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs