

Re: RFC: I/O bandwidth controller

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg05566.html>

- *From:* " Û-" <baramsori72@xxxxxxxxxx>
 - *Date:* Wed, 13 Aug 2008 15:23:00 +0900
-

Hi, Takahashi-san,
Thank you for your comments

Hi,

Fernando Luis Vázquez Cao wrote:

This seems to be the easiest part, but the current cgroups infrastructure has some limitations when it comes to dealing with block devices: impossibility of creating/removing certain control structures dynamically and hardcoding of subsystems (i.e. resource controllers). This makes it difficult

Re: RFC: I/O bandwidth controller

to handle
block
devices that
can be
hotplugged
and go
away at any
time (this
applies not
only to usb
storage but
also
to some
SATA and
SCSI
devices). To
cope with
this
situation
properly we
would need
hotplug
support in
cgroups,
but, as
suggested
before and
discussed in
the past (see
(0) below),
there are
some
limitations.

Even in the
non-hotplug
case it
would be
nice if we
could treat
each
block I/O
device as an
independent
resource,
which
means we
could do
things like
allocating
I/O

Re: RFC: I/O bandwidth controller

bandwidth
on a
per-device
basis. As
long as
performance
is not
compromised
too much,
adding
some kind
of basic
hotplug
support to
cgroups is
probably
worth it.

(0)

<http://lkml.org/lkml/2008/5/21/12>

What about using
major,minor numbers to
identify each device and
account
IO statistics? If a device is
unplugged we could reset
IO statistics
and/or remove IO
limitations for that device
from userspace (i.e. by a
daemon), but
pluggin/unplugging the
device would not be
blocked/affected
in any case. Or am I
oversimplifying the
problem?

If a resource we want to control (a block
device in this case) is
hot-plugged/unplugged the corresponding
cgroup-related structures inside
the kernel need to be allocated/freed
dynamically, respectively. The
problem is that this is not always possible.
For example, with the
current implementation of cgroups it is not
possible to treat each block
device as a different cgroup
subsystem/resource controlled, because

Re: RFC: I/O bandwidth controller

subsystems are created at compile time.

The whole subsystem is created at compile time, but controller data structures are allocated dynamically (i.e. see struct mem_cgroup for memory controller). So, identifying each device with a name, or a key like major,minor, instead of a reference/pointer to a struct could help to handle this in userspace. I mean, if a device is unplugged a userspace daemon can just handle the event and delete the controller data structures allocated for this device, asynchronously, via userspace->kernel interface. And without holding a reference to that particular block device in the kernel. Anyway, implementing a generic interface that would allow to define hooks for hot-pluggable devices (or similar events) in cgroups would be interesting.

3. & 4. & 5.
– I/O
bandwidth
shaping &
General
design
aspects

The
implementation
of an I/O
scheduling
algorithm is
to a certain
extent
influenced
by what we
are trying to
achieve in
terms of I/O
bandwidth
shaping,
but, as
discussed
below, the
required
accuracy

Re: RFC: I/O bandwidth controller

can
determine
the layer
where the
I/O
controller
has to
reside. Off
the top of
my
head, there
are three
basic
operations
we may
want
perform:
– I/O nice
prioritization:
ionice–like
approach.

–
Proportional
bandwidth
scheduling:
each
process/group
of processes
has a weight
that
determines
the share of
bandwidth
they
receive.

– I/O
limiting: set
an upper
limit to the
bandwidth a
group of
tasks
can use.

Use a deadline–based IO
scheduling could be an
interesting path to be
explored as well, IMHO, to
try to guarantee per–cgroup
minimum bandwidth
requirements.

Re: RFC: I/O bandwidth controller

Please note that the only thing we can do is to guarantee minimum bandwidth requirement when there is contention for an IO resource, which is precisely what a proportional bandwidth scheduler does. Am I missing something?

Correct. Proportional bandwidth automatically allows to guarantee minimum requirements (instead of IO limiting approach, that needs additional mechanisms to achieve this).

In any case there's no guarantee for a cgroup/application to sustain i.e. 10MB/s on a certain device, but this is a hard problem anyway, and the best we can do is to try to satisfy "soft" constraints.

I think guaranteeing the minimum I/O bandwidth is very important. In the business site, especially in streaming service system, administrator requires the functionality to satisfy QoS or performance of their service.

Of course, IO throttling is important, but, personally, I think guaranteeing the minimum bandwidth is more important than limitation of maximum bandwidth

to satisfy the requirement in real business sites.

And I know Andrea's io-throttle patch supports the latter case well and it is very stable.

But, the first case(guarantee the minimum bandwidth) is not supported in any patches.

Is there any plans to support it? and Is there any problems in implementing it?

I think if IO controller can support guaranteeing the minimum bandwidth and work-conserving mode simultaneously, it more easily satisfies the requirement

of the business sites.

Additionally, I didn't understand "Proportional bandwidth automatically allows

to guarantee minimum requirements" and "soft constraints".

Can you give me a advice about this ?

Thanks in advance.

Dong-Jae Kang

I think this is what dm-ioband does.

Re: RFC: I/O bandwidth controller

Let's say you make two groups share the same disk, and give them 70% of the bandwidth the disk physically has and 30% respectively. This means the former group is almost guaranteed to be able to use 70% of the bandwidth even when the latter one is issuing quite a lot of I/O requests.

Yes, I know there exist head seek lags with traditional magnetic disks, so it's important to improve the algorithm to reduce this overhead.

In previous my posting, what I mean was absolute guaranteeing for minimum bandwidth, regardless of disk seek time, I/O type(sequential, random, or mixed &) of process.

I also basically prefer proportional share depending on priority or weight and I think it is meaningful, such like as dm-ioband, 2-layer CFQ(satoshi) and 2-Layer CFQ(vasily).

But, additionally in that situation, I think absolute guaranteeing of the minimum bandwidth will be required and several related companies want it to be supported. Because proportional share has inaccuracy of performance predictability, as Andrea mentioned before.

So, to complement the point, IMHO, I think so.

And I think it is also possible to add a new scheduling policy to guarantee the minimum bandwidth. It might be cool if some group can use guranteed bandwidths and the other share the rest on proportional bandwidth policy.

Yes, I agree with you. This was what I intend to say
Thank you,

Dong-Jae Kang

--

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

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

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