

Re: RFC: I/O bandwidth controller

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg05386.html>

- *From:* Andrea Righi <righi.andrea@xxxxxxxxxx>
 - *Date:* Tue, 12 Aug 2008 22:44:30 +0200
-

Fernando Luis Vázquez Cao wrote:

On Tue, 2008-08-12 at 22:29 +0900, Andrea Righi wrote:

Andrea Righi wrote:

Hirokazu Takahashi wrote:

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

Re: RFC: I/O bandwidth controller

trying
to
achieve
in
terms
of
I/O
bandwidth
shaping,
but,
as
discussed
below,
the
required
accuracy
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

Re: RFC: I/O bandwidth controller

to
guarantee
per-cgroup
minimum
bandwidth
requirements.

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.
An
I
missing
something?

Correct.
Proportional
bandwidth
automatically
allows to
guarantee
min

Re: RFC: I/O bandwidth controller

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

Re: RFC: I/O bandwidth controller

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.

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.

And I think it is also possible to add a new scheduling policy to guarantee the minimum bandwidth. It might

Re: RFC: I/O bandwidth controller

be cool if some group can use guaranteed bandwidths and the other share the rest on proportional bandwidth policy.

Thanks,
Hirokazu Takahashi.

With IO limiting approach minimum requirements are supposed to be guaranteed if the user configures a generic block device so that the sum of the limits doesn't exceed the total IO bandwidth of that device. But, in principle, there's nothing in "throttling" that guarantees "fairness" among different cgroups doing IO on the same block devices, that means there's nothing to guarantee minimum requirements (and this is the reason because I liked the Satoshi's CFQ-cgroup approach together with io-throttle).

A more complicated issue is how to evaluate the total IO bandwidth of a generic device. We can use some kind of averaging/prediction, but basically it would be inaccurate due to the mechanic of disks (head seeks, but also caching, buffering mechanisms implemented directly into the device, etc.). It's a hard problem. And the same problem exists also for proportional bandwidth as well, in terms of IO rate predictability I mean.

BTW as I said in a previous email, an interesting path to be explored IMHO could be to think in terms of IO time. So, look at the time an IO request is issued to the drive, look at the time the request is served, evaluate the difference and charge the consumed IO time to the appropriate cgroup. Then dispatch IO requests in function of the consumed IO time debts / credits, using for example a token-bucket strategy. And probably the best place to implement the IO time accounting is the elevator.

Please note that the seek time for a specific IO request is strongly correlated with the IO requests that preceded it, which means that the owner of that request is not the only one to blame if it takes too long to process it. In other words, with the algorithm you propose we may end

Re: RFC: I/O bandwidth controller

up charging the wrong guy.

mmh.. yes. The only scenario I can imagine where this solution is not fair is when there're a lot of guys always requesting the same near blocks and a single guy looking for a single distant block (supposing disk seeks are more expensive than read/write ops).

In this case it would be fair to charge a huge amount only to the guy requesting the single distant block and distribute the cost of the seek to move back the head equally among the other guys. Using the algorithm I proposed, instead, both the single "bad" guy and the first "good" guy that moves back the disk head would spend a large sum of IO credits.

—Andrea

—

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/>