Subject: Re: [PATCH 0/2] dm-band: The I/O bandwidth controller: Overview
Posted by Hirokazu Takahashi on Thu, 24 Jan 2008 08:11:45 GMT
View Forum Message <> Reply to Message

Hi,

> Hi,
>
> I believe this work is very important especially in the context of
> virtual machines. I think it would be more useful though implemented in
> the context of the IO scheduler. Since we already support a notion of
> IO priority, it seems reasonable to add a notion of an IO cap.

I agree that what you proposed is the most straightforward approach.
Ryo and I have also investigated the CFQ scheduler that It will be possible
to enhance it to support bandwidth control with quite a few modification.
I think both approach have pros and cons.

At this time, we have chosen the device-mapper approach because:
 - it can work with any I/O scheduler. Some people will want use the NOOP
   scheduler against high-end storages.
 - only people who need I/O bandwidth control should use it.
 - it is independent to the I/O schedulers so that it will be easy to maintain.
 - it can keep the CFQ implementation simple.

The current the CFQ scheduler has some limitations if you want to control the
bandwidths. The scheduler only has seven priority levels, which also means it
has only seven classes. If you assign the same io-priority A to several VMs
--- virtual machines ---, these machines have to share the I/O bandwidth which
is assign to the io-priority A class. If other VM with io-priority B which is
lower than io-priority A and there is no other VM in the same io-priority B
class, VM in the io-priority B class may be able to use large bandwidth than
VMs in io-priority the A class.

I guess two level scheduling should be introduced in the CFQ scheduler if needed.
The one is to choose the best cgroup or job, and the other is to choose the
highest io-priority class.

There is another limitation that io-priority is global so that it affects
all the disks to access. It isn't allowed to have a job use several io-priorities
to access several disks respectively. I think "per disk io-priority" will be
required.

But the device-mapper approach also has a bad points.
It is hard to get the capabilities and configurations of the underlying devices
such as information of partitions or LUNs. So some configuration tools may
probably be required.

Thank you,
Hirokazu Takahashi.

> Regards,
>
> Anthony Liguori
>
> Ryo Tsuruta wrote:
> > Hi everyone,
> >
> > I'm happy to announce that I've implemented a Block I/O bandwidth controller.
> > The controller is designed to be of use in a cgroup or virtual machine
> > environment. The current approach is that the controller is implemented as
> > a device-mapper driver.
> >
> > What's dm-band all about?
> > ========================
> > Dm-band is an I/O bandwidth controller implemented as a device-mapper driver.
> > Several jobs using the same physical device have to share the bandwidth of
> > the device. Dm-band gives bandwidth to each job according to its weight,
> > which each job can set its own value to.
> >
> > At this time, a job is a group of processes with the same pid or pgrp or uid.
> > There is also a plan to make it support cgroup. A job can also be a virtual
> > machine such as KVM or Xen.
> >
> > +------+ +------+ +------+   +------+ +------+ +------+
> > |cgroup| |cgroup| | the | | pid | | pid | | the | jobs
> > | A | | B | |others| | X | | Y | |others|
> > +--|---+ +--|---+ +--|---+   +--|---+ +--|---+ +--|---+
> > +--V----+---V---+----V---+   +--V----+---V---+----V---+
> > | group | group | default| | group | group | default| band groups
> > | | | group | | | | group |
> > +-------+-------+--------+   +-------+-------+--------+
> > | band1 | | band2 | band devices
> > +------------|------------+   +------------|------------+
> > +------------V--------------+--------------V------------+
> > | | |
> > | sdb1 | sdb2 | physical devices
> > +---------------------------+---------------------------+
> >
> >
> > How dm-band works.
> > ========================
> > Every band device has one band group, which by default is called the default
> > group.
> >
> > Band devices can also have extra band groups in them. Each band group

> > has a job to support and a weight. Proportional to the weight, dm-band gives
> > tokens to the group.
> >
> > A group passes on I/O requests that its job issues to the underlying
> > layer so long as it has tokens left, while requests are blocked
> > if there aren't any tokens left in the group. One token is consumed each
> > time the group passes on a request. Dm-band will refill groups with tokens
> > once all of groups that have requests on a given physical device use up their
> > tokens.
> >
> > With this approach, a job running on a band group with large weight is
> > guaranteed to be able to issue a large number of I/O requests.
> >
> >
> > Getting started
> > ==========
> > The following is a brief description how to control the I/O bandwidth of
> > disks. In this description, we'll take one disk with two partitions as an
> > example target.
> >
> > You can also check the manual at Document/device-mapper/band.txt of the
> > linux kernel source tree for more information.
> >
> >
> > Create and map band devices
> > ---------------------------
> > Create two band devices "band1" and "band2" and map them to "/dev/sda1"
> > and "/dev/sda2" respectively.
> >
> > # echo "0 `blockdev --getsize /dev/sda1` band /dev/sda1 1" | dmsetup create band1
> > # echo "0 `blockdev --getsize /dev/sda2` band /dev/sda2 1" | dmsetup create band2
> >
> > If the commands are successful then the device files "/dev/mapper/band1"
> > and "/dev/mapper/band2" will have been created.
> >
> >
> > Bandwidth control
> > ----------------
> > In this example weights of 40 and 10 will be assigned to "band1" and
> > "band2" respectively. This is done using the following commands:
> >
> > # dmsetup message band1 0 weight 40
> > # dmsetup message band2 0 weight 10
> >
> > After these commands, "band1" can use 80% --- 40/(40+10)*100 --- of the
> > bandwidth of the physical disk "/dev/sda" while "band2" can use 20%.
> >
> >

> > Additional bandwidth control
> > --------------------------
> > In this example two extra band groups are created on "band1".
> > The first group consists of all the processes with user-id 1000 and the
> > second group consists of all the processes with user-id 2000. Their
> > weights are 30 and 20 respectively.
> >
> > Firstly the band group type of "band1" is set to "user".
> > Then, the user-id 1000 and 2000 groups are attached to "band1".
> > Finally, weights are assigned to the user-id 1000 and 2000 groups.
> >
> > # dmsetup message band1 0 type user
> > # dmsetup message band1 0 attach 1000
> > # dmsetup message band1 0 attach 2000
> > # dmsetup message band1 0 weight 1000:30
> > # dmsetup message band1 0 weight 2000:20
> >
> > Now the processes in the user-id 1000 group can use 30% ---
> > 30/(30+20+40+10)*100 --- of the bandwidth of the physical disk.
> >
> > Band Device   Band Group   Weight
> >   band1   user id 1000   30
> >   band1   user id 2000   20
> >   band1   default group(the other users)   40
> >   band2   default group   10
> >
> >
> > Remove band devices
> > -------------------
> > Remove the band devices when no longer used.
> >
> >   # dmsetup remove band1
> >   # dmsetup remove band2
> >
> >
> > TODO
> > ========================
> >   - Cgroup support.
> >   - Control read and write requests separately.
> >   - Support WRITE_BARRIER.
> >   - Optimization.
> >   - More configuration tools. Or is the dmsetup command sufficient?
> >   - Other policies to schedule BIOs. Or is the weight policy sufficient?
> >
> > Thanks,
> > Ryo Tsuruta
>
> --

> 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/

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers