

---

Subject: [PATCH 1/3] i/o bandwidth controller documentation

Posted by [Andrea Righi](#) on Fri, 20 Jun 2008 10:05:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Documentation of the block device I/O bandwidth controller: description, usage, advantages and design.

Signed-off-by: Andrea Righi <righi.andrea@gmail.com>

---

```
Documentation/controllers/io-throttle.txt | 163 ++++++
1 files changed, 163 insertions(+), 0 deletions(-)
create mode 100644 Documentation/controllers/io-throttle.txt
```

```
diff --git a/Documentation/controllers/io-throttle.txt b/Documentation/controllers/io-throttle.txt
```

```
new file mode 100644
```

```
index 0000000..e1df98a
```

```
--- /dev/null
```

```
+++ b/Documentation/controllers/io-throttle.txt
```

```
@@ -0,0 +1,163 @@
```

```
+
```

```
+       Block device I/O bandwidth controller
```

```
+
```

```
+1. Description
```

```
+
```

```
+This controller allows to limit the I/O bandwidth of specific block devices for
+specific process containers (cgroups) imposing additional delays on I/O
+requests for those processes that exceed the limits defined in the control
+group filesystem.
```

```
+
```

```
+Bandwidth limiting rules offer better control over QoS with respect to priority
+or weight-based solutions that only give information about applications'
+relative performance requirements.
```

```
+
```

```
+The goal of the I/O bandwidth controller is to improve performance
+predictability and QoS of the different control groups sharing the same block
+devices.
```

```
+
```

```
+NOTE #1: if you're looking for a way to improve the overall throughput of the
+system probably you should use a different solution.
```

```
+
```

```
+NOTE #2: the current implementation does not guarantee minimum bandwidth
+levels, the QoS is implemented only slowing down i/o "traffic" that exceeds the
+limits specified by the user. Minimum i/o rate thresholds are supposed to be
+guaranteed if the user configures a proper i/o bandwidth partitioning of the
+block devices shared among the different cgroups (theoretically if the sum of
+all the single limits defined for a block device doesn't exceed the total i/o
+bandwidth of that device).
```

```
+
```

## +2. User Interface

+

+A new I/O bandwidth limitation rule is described using the file

+`blockio.bandwidth`.

+

+The same file can be used to set multiple rules for different block devices

+relative to the same cgroup.

+

+The syntax is the following:

```
+# /bin/echo DEVICE:BANDWIDTH > CGROUP/blockio.bandwidth
```

+

+- DEVICE is the name of the device the limiting rule is applied to,

+- BANDWIDTH is the maximum I/O bandwidth on DEVICE allowed by CGROUP (we can

+ use a suffix k, K, m, M, g or G to indicate bandwidth values in KB/s, MB/s

+ or GB/s),

+- CGROUP is the name of the limited process container.

+

+Examples:

+

+\* Mount the cgroup filesystem (blockio subsystem):

```
+ # mkdir /mnt/cgroup
```

```
+ # mount -t cgroup -oblockio blockio /mnt/cgroup
```

+

+\* Instantiate the new cgroup "foo":

```
+ # mkdir /mnt/cgroup/foo
```

+ --> the cgroup foo has been created

+

+\* Add the current shell process to the cgroup "foo":

```
+ # /bin/echo $$ > /mnt/cgroup/foo/tasks
```

+ --> the current shell has been added to the cgroup "foo"

+

+\* Give maximum 1MiB/s of I/O bandwidth on `/dev/sda1` for the cgroup "foo":

```
+ # /bin/echo /dev/sda1:1M > /mnt/cgroup/foo/blockio.bandwidth
```

```
+ # sh
```

+ --> the subshell 'sh' is running in cgroup "foo" and it can use a maximum I/O

+ bandwidth of 1MiB/s on `/dev/sda1` (blockio.bandwidth is expressed in

+ KiB/s).

+

+\* Give maximum 8MiB/s of I/O bandwidth on `/dev/sdb` for the cgroup "foo":

```
+ # /bin/echo /dev/sda5:8M > /mnt/cgroup/foo/blockio.bandwidth
```

```
+ # sh
```

+ --> the subshell 'sh' is running in cgroup "foo" and it can use a maximum I/O

+ bandwidth of 1MiB/s on `/dev/sda1` and 8MiB/s on `/dev/sda5`.

+ NOTE: each partition needs its own limitation rule! In this case, for

+ example, there's no limitation on `/dev/sda5` for cgroup "foo".

+

+\* Run a benchmark doing I/O on `/dev/sda1` and `/dev/sda5`; I/O limits and usage

+ defined for cgroup "foo" can be shown as following:

```

+ # cat /mnt/cgroup/foo/blockio.bandwidth
+ === device (8,1) ===
+ bandwidth limit: 1024 KiB/sec
+ current i/o usage: 819 KiB/sec
+ === device (8,5) ===
+ bandwidth limit: 1024 KiB/sec
+ current i/o usage: 3102 KiB/sec
+
+ Devices are reported using (major, minor) numbers when reading
+ blockio.bandwidth.
+
+ The corresponding device names can be retrieved in /proc/diskstats (or in
+ other places as well).
+
+ For example to find the name of the device (8,5):
+ # sed -ne 's/^ \+8 \+5 \([^ ]\+\).*\1/p' /proc/diskstats
+ sda5
+
+ Current I/O usage can be greater than bandwidth limit, this means the i/o
+ controller is going to impose the limitation.
+
+* Extend the maximum I/O bandwidth for the cgroup "foo" to 8MiB/s:
+ # /bin/echo /dev/sda1:8M > /mnt/cgroup/foo/blockio-bandwidth
+
+* Remove limiting rule on /dev/sda1 for cgroup "foo":
+ # /bin/echo /dev/sda1:0 > /mnt/cgroup/foo/blockio-bandwidth
+
+3. Advantages of providing this feature
+
+* Allow I/O traffic shaping for block device shared among different cgroups
+* Improve I/O performance predictability on block devices shared between
+ different cgroups
+* Limiting rules do not depend of the particular I/O scheduler (anticipatory,
+ deadline, CFQ, noop) and/or the type of the underlying block devices
+* The bandwidth limitations are guaranteed both for synchronous and
+ asynchronous operations, even the I/O passing through the page cache or
+ buffers and not only direct I/O (see below for details)
+* It is possible to implement a simple user-space application to dynamically
+ adjust the I/O workload of different process containers at run-time,
+ according to the particular users' requirements and applications' performance
+ constraints
+* It is even possible to implement event-based performance throttling
+ mechanisms; for example the same user-space application could actively
+ throttle the I/O bandwidth to reduce power consumption when the battery of a
+ mobile device is running low (power throttling) or when the temperature of a
+ hardware component is too high (thermal throttling)
+* Provides zero overhead for non block device I/O bandwidth controller users
+

```

#### +4. Design

+

+The I/O throttling is performed imposing an explicit timeout, via  
+`schedule_timeout_killable()` on the processes that exceed the I/O bandwidth  
+dedicated to the cgroup they belong to. I/O accounting happens per cgroup.

+

+It just works as expected for read operations: the real I/O activity is reduced  
+synchronously according to the defined limitations.

+

+Write operations, instead, are modeled depending of the dirty pages ratio  
+(write throttling in memory), since the writes to the real block devices are  
+processed asynchronously by different kernel threads (pdflush). However, the  
+dirty pages ratio is directly proportional to the actual I/O that will be  
+performed on the real block device. So, due to the asynchronous transfers  
+through the page cache, the I/O throttling in memory can be considered a form  
+of anticipatory throttling to the underlying block devices.

+

+Multiple re-writes in already dirtied page cache areas are not considered for  
+accounting the I/O activity. This is valid for multiple re-reads of pages  
+already present in the page cache as well.

+

+This means that a process that re-writes and/or re-reads multiple times the  
+same blocks in a file (without re-creating it by `truncate()`, `ftruncate()`,  
+`creat()`, etc.) is affected by the I/O limitations only for the actual I/O  
+performed to (or from) the underlying block devices.

+

+Multiple rules for different block devices are stored in a linked list, using  
+the `dev_t` number of each block device as key to uniquely identify each element  
+of the list. RCU synchronization is used to protect the whole list structure,  
+since the elements in the list are not supposed to change frequently (they  
+change only when a new rule is defined or an old rule is removed or updated),  
+while the reads in the list occur at each operation that generates I/O. This  
+allows to provide zero overhead for cgroups that do not use any limitation.

+

+WARNING: per-block device limiting rules always refer to the `dev_t` device  
+number. If a block device is unplugged (i.e. a USB device) the limiting rules  
+associated to that device persist and they are still valid if a new device is  
+plugged in the system and it uses the same major and minor numbers.

--

#### 1.5.4.3

---

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

Subject: Re: [PATCH 1/3] i/o bandwidth controller documentation

Posted by [Randy Dunlap](#) on Fri, 20 Jun 2008 17:08:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Fri, 20 Jun 2008 12:05:33 +0200 Andrea Righi wrote:

```
> Documentation of the block device I/O bandwidth controller: description, usage,
> advantages and design.
>
> Signed-off-by: Andrea Righi <righi.andrea@gmail.com>
> ---
> Documentation/controllers/io-throttle.txt | 163 ++++++
> 1 files changed, 163 insertions(+), 0 deletions(-)
> create mode 100644 Documentation/controllers/io-throttle.txt
>
> diff --git a/Documentation/controllers/io-throttle.txt b/Documentation/controllers/io-throttle.txt
> new file mode 100644
> index 0000000..e1df98a
> --- /dev/null
> +++ b/Documentation/controllers/io-throttle.txt
> @@ -0,0 +1,163 @@
> +
> +          Block device I/O bandwidth controller
> +
> +1. Description
> +
> +This controller allows to limit the I/O bandwidth of specific block devices for
> +specific process containers (cgroups) imposing additional delays on I/O
> +requests for those processes that exceed the limits defined in the control
> +group filesystem.
> +
> +Bandwidth limiting rules offer better control over QoS with respect to priority
> +or weight-based solutions that only give information about applications'
> +relative performance requirements.
> +
> +The goal of the I/O bandwidth controller is to improve performance
> +predictability and QoS of the different control groups sharing the same block
> +devices.
> +
> +NOTE #1: if you're looking for a way to improve the overall throughput of the
```

I would s/if/If/

```
> +system probably you should use a different solution.
> +
> +NOTE #2: the current implementation does not guarantee minimum bandwidth
```

s/the/The/

> +levels, the QoS is implemented only slowing down i/o "traffic" that exceeds the

Please consistently use "I/O" instead of "i/o".

Above comma makes a run-on sentence. A period or semi-colon would be better IMO.

> +limits specified by the user. Minimum i/o rate thresholds are supposed to be  
> +guaranteed if the user configures a proper i/o bandwidth partitioning of the  
> +block devices shared among the different cgroups (theoretically if the sum of  
> +all the single limits defined for a block device doesn't exceed the total i/o  
> +bandwidth of that device).

> +

> +2. User Interface

> +

> +A new I/O bandwidth limitation rule is described using the file  
> +blockio.bandwidth.

> +

> +The same file can be used to set multiple rules for different block devices  
> +relative to the same cgroup.

> +

> +The syntax is the following:

> +# /bin/echo DEVICE:BANDWIDTH > CGROUP/blockio.bandwidth

> +

> +- DEVICE is the name of the device the limiting rule is applied to,  
> +- BANDWIDTH is the maximum I/O bandwidth on DEVICE allowed by CGROUP (we can  
> + use a suffix k, K, m, M, g or G to indicate bandwidth values in KB/s, MB/s  
> + or GB/s),  
> +- CGROUP is the name of the limited process container.

> +

> +Examples:

> +

> +\* Mount the cgroup filesystem (blockio subsystem):

> + # mkdir /mnt/cgroup

> + # mount -t cgroup -oblockio blockio /mnt/cgroup

> +

> +\* Instantiate the new cgroup "foo":

> + # mkdir /mnt/cgroup/foo

> + --> the cgroup foo has been created

> +

> +\* Add the current shell process to the cgroup "foo":

> + # /bin/echo \$\$ > /mnt/cgroup/foo/tasks

> + --> the current shell has been added to the cgroup "foo"

> +

> +\* Give maximum 1MiB/s of I/O bandwidth on /dev/sda1 for the cgroup "foo":

> + # /bin/echo /dev/sda1:1M > /mnt/cgroup/foo/blockio.bandwidth

> + # sh

> + --> the subshell 'sh' is running in cgroup "foo" and it can use a maximum I/O  
> + bandwidth of 1MiB/s on /dev/sda1 (blockio.bandwidth is expressed in

```

> +   KiB/s).
> +
> +* Give maximum 8MiB/s of I/O bandwidth on /dev/sdb for the cgroup "foo":
> + # /bin/echo /dev/sda5:8M > /mnt/cgroup/foo/blockio.bandwidth
> + # sh
> + --> the subshell 'sh' is running in cgroup "foo" and it can use a maximum I/O
> +   bandwidth of 1MiB/s on /dev/sda1 and 8MiB/s on /dev/sda5.
> +   NOTE: each partition needs its own limitation rule! In this case, for
> +   example, there's no limitation on /dev/sda5 for cgroup "foo".
> +
> +* Run a benchmark doing I/O on /dev/sda1 and /dev/sda5; I/O limits and usage
> + defined for cgroup "foo" can be shown as following:
> + # cat /mnt/cgroup/foo/blockio.bandwidth
> + ==== device (8,1) ====
> +   bandwidth limit: 1024 KiB/sec
> +   current i/o usage: 819 KiB/sec
> + ==== device (8,5) ====
> +   bandwidth limit: 1024 KiB/sec
> +   current i/o usage: 3102 KiB/sec

```

Ugh, this makes it look like the output does "pretty printing" (formatting), which is generally not a good idea. Let some app be responsible for that, not the kernel. Basically this means don't use leading spaces just to make the ":"s line up in the output.

```

> +
> + Devices are reported using (major, minor) numbers when reading
> + blockio.bandwidth.
> +
> + The corresponding device names can be retrieved in /proc/diskstats (or in
> + other places as well).
> +
> + For example to find the name of the device (8,5):
> + # sed -ne 's/^ \+8 \+5 \([^ ]+\).*\1/p' /proc/diskstats
> + sda5
> +
> + Current I/O usage can be greater than bandwidth limit, this means the i/o

```

Run-on sentence. Change , to . (with This) or use ;

```

> + controller is going to impose the limitation.
> +
> +* Extend the maximum I/O bandwidth for the cgroup "foo" to 8MiB/s:
> + # /bin/echo /dev/sda1:8M > /mnt/cgroup/foo/blockio-bandwidth
> +
> +* Remove limiting rule on /dev/sda1 for cgroup "foo":
> + # /bin/echo /dev/sda1:0 > /mnt/cgroup/foo/blockio-bandwidth

```

> +  
> +3. Advantages of providing this feature  
> +  
> +\* Allow I/O traffic shaping for block device shared among different cgroups  
> +\* Improve I/O performance predictability on block devices shared between  
> + different cgroups  
> +\* Limiting rules do not depend of the particular I/O scheduler (anticipatory,  
> + deadline, CFQ, noop) and/or the type of the underlying block devices  
> +\* The bandwidth limitations are guaranteed both for synchronous and  
> + asynchronous operations, even the I/O passing through the page cache or  
> + buffers and not only direct I/O (see below for details)  
> +\* It is possible to implement a simple user-space application to dynamically  
> + adjust the I/O workload of different process containers at run-time,  
> + according to the particular users' requirements and applications' performance  
> + constraints  
> +\* It is even possible to implement event-based performance throttling  
> + mechanisms; for example the same user-space application could actively  
> + throttle the I/O bandwidth to reduce power consumption when the battery of a  
> + mobile device is running low (power throttling) or when the temperature of a  
> + hardware component is too high (thermal throttling)  
> +\* Provides zero overhead for non block device I/O bandwidth controller users  
> +  
> +4. Design  
> +  
> +The I/O throttling is performed imposing an explicit timeout, via  
> +`schedule_timeout_killable()` on the processes that exceed the I/O bandwidth  
> +dedicated to the cgroup they belong to. I/O accounting happens per cgroup.  
> +  
> +It just works as expected for read operations: the real I/O activity is reduced  
> +synchronously according to the defined limitations.  
> +  
> +Write operations, instead, are modeled depending of the dirty pages ratio  
> +(write throttling in memory), since the writes to the real block devices are  
> +processed asynchronously by different kernel threads (`pdflush`). However, the  
> +dirty pages ratio is directly proportional to the actual I/O that will be  
> +performed on the real block device. So, due to the asynchronous transfers  
> +through the page cache, the I/O throttling in memory can be considered a form  
> +of anticipatory throttling to the underlying block devices.  
> +  
> +Multiple re-writes in already dirtied page cache areas are not considered for  
> +accounting the I/O activity. This is valid for multiple re-reads of pages  
> +already present in the page cache as well.  
> +  
> +This means that a process that re-writes and/or re-reads multiple times the  
> +same blocks in a file (without re-creating it by `truncate()`, `ftruncate()`,  
> +`creat()`, etc.) is affected by the I/O limitations only for the actual I/O  
> +performed to (or from) the underlying block devices.  
> +



> +Multiple rules for different block devices are stored in a linked list, using  
> +the dev\_t number of each block device as key to uniquely identify each element  
> +of the list. RCU synchronization is used to protect the whole list structure,  
> +since the elements in the list are not supposed to change frequently (they  
> +change only when a new rule is defined or an old rule is removed or updated),  
> +while the reads in the list occur at each operation that generates I/O. This  
> +allows to provide zero overhead for cgroups that do not use any limitation.  
> +  
> +WARNING: per-block device limiting rules always refer to the dev\_t device  
> +number. If a block device is unplugged (i.e. a USB device) the limiting rules  
> +associated to that device persist and they are still valid if a new device is

associated with (?)

> +plugged in the system and it uses the same major and minor numbers.  
> --

---

~Randy

Linux Plumbers Conference, 17-19 September 2008, Portland, Oregon USA

<http://linuxplumbersconf.org/>

---

Containers mailing list

[Containers@lists.linux-foundation.org](mailto:Containers@lists.linux-foundation.org)

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/3] i/o bandwidth controller documentation

Posted by [Andrea Righi](#) on Sat, 21 Jun 2008 10:35:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Thanks Randy, I've applied all your fixes to my local documentation,  
next patchset version will include them. A few small comments below.

Randy Dunlap wrote:

>> +\* Run a benchmark doing I/O on /dev/sda1 and /dev/sda5; I/O limits and usage

>> + defined for cgroup "foo" can be shown as following:

>> + # cat /mnt/cgroup/foo/blockio.bandwidth

>> + === device (8,1) ===

>> + bandwidth limit: 1024 KiB/sec

>> + current i/o usage: 819 KiB/sec

>> + === device (8,5) ===

>> + bandwidth limit: 1024 KiB/sec

>> + current i/o usage: 3102 KiB/sec

>

> Ugh, this makes it look like the output does "pretty printing" (formatting),  
> which is generally not a good idea. Let some app be responsible for that,  
> not the kernel. Basically this means don't use leading spaces just to make the

> ":"s line up in the output.

Sounds reasonable. I think the output could be further reduced, the following format should be explanatory enough.

```
device: %u,%u
bandwidth: %lu KiB/sec
usage: %lu KiB/sec
```

```
>> +WARNING: per-block device limiting rules always refer to the dev_t device
>> +number. If a block device is unplugged (i.e. a USB device) the limiting rules
>> +associated to that device persist and they are still valid if a new device is
>
> associated with (?)
```

what about:

...the limiting rules defined for that device...

-Andrea

---

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

---

---

Subject: Re: [PATCH 1/3] i/o bandwidth controller documentation  
Posted by [Randy Dunlap](#) on Sun, 22 Jun 2008 16:03:43 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

--- Original Message ---

```
> Thanks Randy, I've applied all your fixes to my local
> documentation,
> next patchset version will include them. A few small comments
> below.
>
> >> +WARNING: per-block device limiting rules always refer to the dev_t device
> >> +number. If a block device is unplugged (i.e. a USB device) the limiting rules
> >> +associated to that device persist and they are still valid if a new device is
> >
> > associated with (?)
>
> what about:
>
> ...the limiting rules defined for that device...
```

Hi Andrea,

Yes, that's fine.

Thanks.

---

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

---