Subject: Re: [PATCH 0/3] cgroup: block device i/o bandwidth controller
Posted by Andrea Righi on Mon, 02 Jun 2008 11:17:31 GMT
View Forum Message <> Reply to Message

Andrea Righi wrote:
> So, my approach has the disadvantage or the advantage (depending on the
> context and the requirements) to explicitly choke applications'
> requests. Other solutions that operates in the subsystem used to
> dispatch i/o requests are probably better to maximize overall
> performance, but do not offer the same control over a real QoS as
> request limiting can do.
>
> Another difference is that all of them are priority/weighted based. The
> io-throttle controller, instead, allows to define direct bandwidth
> limiting rules. The difference here is that priority based approach
> propagates bursts and does no smoothing. Bandwidth limiting approach
> controls bursts by smoothing the i/o rate. This means better performance
> predictability at the cost of poor throughput optimization.
>
> I'll run some benchmarks and post the results ASAP. It would be also
> interesting to run the same benchmarks using the other i/o controllers
> and compare the results in terms of fairness, performance
> predictability, responsiveness, throughput, etc. I'll see what I can do.

Here is the test report of the benchmarks. Comments, suggestions,
disapprovals, etc. are welcome.

Tests are mainly focused to demonstrate how the io-throttle controller
can be used to improve the i/o performance predictability on shared
block devices (usually any block device shared by many users and
processes, that can be grouped togheter by cgroup).

The goal of the tests is *not* to compare the overall performance of the
different solutions.

Tests have been performed using an ext3 fs + CFQ i/o scheduler on
/dev/sda.

Following some details of the testbed host:
# hdparm -i /dev/sda
/dev/sda:

 Model=SAMSUNG HS08XJC , FwRev=GR100-01,
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs }
 RawCHS=16383/16/63, TrkSize=34902, SectSize=0, ECCbytes=4
 BuffType=DualPortCache, BuffSize=7986kB, MaxMultSect=16, MultSect=?8?
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=156301488
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}

PIO modes:  pio0 pio1 pio2 pio3 pio4
DMA modes:  mdma0 mdma1 mdma2
UDMA modes: udma0 udma1 udma2 udma3 udma4 *udma5
AdvancedPM=yes: unknown setting WriteCache=enabled
Drive conforms to: ATA/ATAPI-6 T13 1410D revision 1:  ATA/ATAPI-1,2,3,4,5,6,7

* signifies the current active mode

```
# grep "model name" /proc/cpuinfo
model name : Intel(R) Core(TM)2 CPU         U7600  @ 1.20GHz
model name : Intel(R) Core(TM)2 CPU         U7600  @ 1.20GHz

# grep MemTotal /proc/meminfo
MemTotal:      2021616 kB
```

Latest io-throttle patch, raw data of the benchmarks and scripts used to calculate the results can be found here:

http://download.systemimager.org/~arighi/linux/patches/io-throttle/

== Test #0: basic i/o ops ==

The goal of this pre-test is to demonstrate the basic functionalities of the io-throttle controller. In particular to show that the i/o bandwidth limiting is applied only on the actual i/o and does not involve read/write operations in the page cache.

A simple script (dd-band.sh) creates a 64MiB file, re-write it (in the page cache), read the file in O_DIRECT mode and re-read the file from the page cache.

The i/o limitations are performed during the first write (because it generates real writes processed asynchronously by the pdflush kernel thread outside the context of the process) and during the O_DIRECT read. Re-writes and re-reads that only affect the page cache are never limited.

Running in cgroup "foo" without any i/o limitation:

```
$ echo $$ | sudo tee /cgroup/foo/tasks
$ ./dd-band.sh
===
write:    67108864 bytes (67 MB) copied, 0.917311 s, 73.2 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.437337 s, 153 MB/s
read:     67108864 bytes (67 MB) copied, 2.86287 s, 23.4 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0966508 s, 694 MB/s
===
write:    67108864 bytes (67 MB) copied, 0.893401 s, 75.1 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.447029 s, 150 MB/s
```

read:  67108864 bytes (67 MB) copied, 2.94607 s, 22.8 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.093962 s, 714 MB/s
===
write:  67108864 bytes (67 MB) copied, 0.896389 s, 74.9 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.437567 s, 153 MB/s
read:  67108864 bytes (67 MB) copied, 2.8686 s, 23.4 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.10307 s, 651 MB/s
===
write:  67108864 bytes (67 MB) copied, 0.896151 s, 74.9 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.436411 s, 154 MB/s
read:  67108864 bytes (67 MB) copied, 3.19625 s, 21.0 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0914293 s, 734 MB/s
===
write:  67108864 bytes (67 MB) copied, 0.896905 s, 74.8 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.436574 s, 154 MB/s
read:  67108864 bytes (67 MB) copied, 2.77595 s, 24.2 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0955861 s, 702 MB/s

The i/o bandwidth of cgroup "foo" is limited up to 8MB/s:
# echo 8192 > /cgroup/foo/blockio.bandwidth
$ ./dd-band.sh
===
write:  67108864 bytes (67 MB) copied, 8.51182 s, 7.9 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.473478 s, 142 MB/s
read:  67108864 bytes (67 MB) copied, 8.19994 s, 8.2 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0886488 s, 757 MB/s
===
write:  67108864 bytes (67 MB) copied, 8.64248 s, 7.8 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.440868 s, 152 MB/s
read:  67108864 bytes (67 MB) copied, 8.19815 s, 8.2 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0956493 s, 702 MB/s
===
write:  67108864 bytes (67 MB) copied, 8.65532 s, 7.8 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.477425 s, 141 MB/s
read:  67108864 bytes (67 MB) copied, 8.20111 s, 8.2 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0978265 s, 686 MB/s
===
write:  67108864 bytes (67 MB) copied, 8.6522 s, 7.8 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.439753 s, 153 MB/s
read:  67108864 bytes (67 MB) copied, 8.21767 s, 8.2 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0991603 s, 677 MB/s
===
write:  67108864 bytes (67 MB) copied, 8.66015 s, 7.7 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.449204 s, 149 MB/s
read:  67108864 bytes (67 MB) copied, 8.27809 s, 8.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0874324 s, 768 MB/s

The i/o bandwidth of cgroup "foo" is increased up to 16MB/s:

```
# echo 16384 > /cgroup/foo/blockio.bandwidth
$ ./dd-band.sh
===
write:    67108864 bytes (67 MB) copied, 4.37976 s, 15.3 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.460999 s, 146 MB/s
read:     67108864 bytes (67 MB) copied, 4.36709 s, 15.4 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0978816 s, 686 MB/s
===
write:    67108864 bytes (67 MB) copied, 4.40452 s, 15.2 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.437004 s, 154 MB/s
read:     67108864 bytes (67 MB) copied, 4.28799 s, 15.7 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0912823 s, 735 MB/s
===
write:    67108864 bytes (67 MB) copied, 4.45284 s, 15.1 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.440571 s, 152 MB/s
read:     67108864 bytes (67 MB) copied, 4.21476 s, 15.9 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0957647 s, 701 MB/s
===
write:    67108864 bytes (67 MB) copied, 4.4364 s, 15.1 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.440202 s, 152 MB/s
read:     67108864 bytes (67 MB) copied, 4.19432 s, 16.0 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0930223 s, 721 MB/s
===
write:    67108864 bytes (67 MB) copied, 4.45086 s, 15.1 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.4487 s, 150 MB/s
read:     67108864 bytes (67 MB) copied, 4.37955 s, 15.3 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0983858 s, 682 MB/s

The i/o bandwidth of cgroup "foo" is reduced to 4MB/s:
# echo 4096 > /cgroup/foo/blockio.bandwidth
$ ./dd-band.sh
==
write:    67108864 bytes (67 MB) copied, 16.783 s, 4.0 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.456829 s, 147 MB/s
read:     67108864 bytes (67 MB) copied, 16.3971 s, 4.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.101392 s, 662 MB/s
===
write:    67108864 bytes (67 MB) copied, 16.8167 s, 4.0 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.472562 s, 142 MB/s
read:     67108864 bytes (67 MB) copied, 16.3957 s, 4.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0930801 s, 721 MB/s
===
write:    67108864 bytes (67 MB) copied, 16.8834 s, 4.0 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.445081 s, 151 MB/s
read:     67108864 bytes (67 MB) copied, 16.3901 s, 4.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0875281 s, 767 MB/s
===
write:    67108864 bytes (67 MB) copied, 16.8157 s, 4.0 MB/s
```

re-write: 67108864 bytes (67 MB) copied, 0.469043 s, 143 MB/s
read:  67108864 bytes (67 MB) copied, 16.3917 s, 4.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.0922022 s, 728 MB/s
===
write:  67108864 bytes (67 MB) copied, 16.9313 s, 4.0 MB/s
re-write: 67108864 bytes (67 MB) copied, 0.510635 s, 131 MB/s
read:  67108864 bytes (67 MB) copied, 16.3983 s, 4.1 MB/s
re-read:  67108864 bytes (67 MB) copied, 0.089941 s, 746 MB/s

Following are reported more complex benchmarks, using iozone to generate different i/o workloads. In particular all the results below are evaluated considering read, re-read, write, re-write, random-read and random-write operations on files ranging from 8MB up to 32MB and record size ranging from 1K to 8K.

Using more linear i/o workloads (like the simple "dd"s) strongly reduces errors and deviations respect to the expected performance results, but the purpose of the test is to actually verify the effectiveness of the throttling approach in performance predictability, in particular in presence of heterogeneous and complex workloads.

== Test #1: parallel iozone, balanced load (without cgroup limitations) ==

This test is a run of 2 parallel iozone: 1 in cgroup "foo" and the other in cgroup "bar", without any bandwidth limitation.

cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m

Results

| | cgroup "foo" | cgroup "bar" |
|---|---|---|
| Average i/o rate (KB/s) | 15869.72 | 14404.63 |
| Standard dev. | 8002.53 | 6147.71 |
| Percentage err.* | 50.43% | 42.68% |

* Percentage error is evaluated as: (standard deviation / mean) * 100

== Test #2: parallel iozone, unbalanced load (without cgroup limitations) ==

This test is a run of 5 parallel iozone: 1 in cgroup "foo" and 4 iozone(s) in cgroup "bar" without any bandwidth limitation.

```
cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
```

Results

| | cgroup "foo" | cgroup "bar" |
|---|---|---|
| Average i/o rate (KB/s) | 7139.43 | 26161.22 |
| Standard dev. | 4484.00 | 9098.80 |
| Percentage err. | 62.81% | 34.78% |

== Test #3: parallel iozone, balanced load (limitation: 8MB/s | 8MB/s)==

This test is a run of 2 parallel iozone: 1 in cgroup "foo" with 8MB/s
bandwidth limitation and the other in cgroup "bar" with 8MB/s as well.

```
# echo 8192 > /cgroup/foo/blockio.bandwidth
# echo 8192 > /cgroup/bar/blockio.bandwidth
cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
```

Results

| | cgroup "foo" | cgroup "bar" |
|---|---|---|
| Expected i/o rate (KB/s) | 8192.00 | 8192.00 |
| Average i/o rate (KB/s) | 8339.79 | 8242.14 |
| Standard dev. | 734.66 | 896.59 |
| Percentage err.* | 8.81% | 10.88% |

* Percentage error is evaluated as: (standard deviation / mean) * 100

== Test #4: parallel iozone, balanced load (limitation: 8MB/s | 16MB/s)==

This test is a run of 2 parallel iozone: 1 in cgroup "foo" and the other

in cgroup "bar", using different i/o bandwidth limitations: 8MB/s for
cgroup "foo" and 16MB/s for cgroup "bar".

```
# echo 8192 > /cgroup/foo/blockio.bandwidth
# echo 16384 > /cgroup/bar/blockio.bandwidth
cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
```

Results

| | cgroup "foo" | cgroup "bar" |
|---|---|---|
| Expected i/o rate (KB/s) | 8192.00 | 16384.00 |
| Average i/o rate (KB/s) | 7978.03 | 14117.72 |
| Standard dev. | 1028.41 | 2479.05 |
| Percentage err. | 12.89% | 17.56% |

== Test #5: parallel iozone, unbalanced load (limitation: 8MB/s | 8MB/s)==

This test is a run of 5 parallel iozone: 1 in cgroup "foo" with 8MB/s
bandwidth limitation and 4 parallel iozone(s) in cgroup "bar" with
8MB/s.

```
# echo 8192 > /cgroup/foo/blockio.bandwidth
# echo 8192 > /cgroup/bar/blockio.bandwidth
cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
```

Results

| | cgroup "foo" | cgroup "bar" |
|---|---|---|
| Expected i/o rate (KB/s) | 8192.00 | 8192.00 |
| Average i/o rate (KB/s) | 7873.29 | 8831.25 |
| Standard dev. | 954.51 | 1191.74 |

```
+------------------+-----------------+-----------------+
| Percentage err.  |  12.12%         |  13.49%         |
+------------------+-----------------+-----------------+
```

== Test #6: parallel iozone, unbalanced load (limitation: 4MB/s | 8MB/s)==

This test is a run of 5 parallel iozone: 1 in cgroup "foo" with 4MB/s bandwidth limitation and 4 parallel iozone(s) in cgroup "bar" with 8MB/s.

```
# echo 4096 > /cgroup/foo/blockio.bandwidth
# echo 8192 > /cgroup/bar/blockio.bandwidth
cgroup:foo$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
cgroup:bar$ iozone -I -a -i0 -i1 -i2 -i3 -i4 -i5 -y 1024 -q 8192 -n 8m -g 32m &
```

Results

```
                +==================+==================+
                |  cgroup "foo"    |  cgroup "bar"    |
+---------------+==================+==================+
| Expected i/o rate |  4096.00     |  8192.00         |
|     (KB/s)        |              |                  |
+-------------------+--------------+------------------+
| Average i/o rate  |  4166.57     |  8770.42         |
|     (KB/s)        |              |                  |
+-------------------+--------------+------------------+
| Standard dev.     |  487.04      |  988.03          |
+-------------------+--------------+------------------+
| Percentage err.   |  11.69%      |  11.26%          |
+-------------------+--------------+------------------+
```

== Final Results ==

```
                  +==================+==================+
                  |  cgroup "foo"    |  cgroup "bar"    |
+---+-------------+==================+==================+
| 1 | Percentage err.  |  50.43%      |  42.68%          |
+---+------------------+--------------+------------------+
| 2 | Percentage err.  |  62.81%      |  34.78%          |
+---+------------------+--------------+------------------+
| 3 | Percentage err.  |  8.81%       |  10.88%          |
+---+------------------+--------------+------------------+
| 4 | Percentage err.  |  12.89%      |  17.56%          |
+---+------------------+--------------+------------------+
| 5 | Percentage err.  |  12.12%      |  13.49%          |
+---+------------------+--------------+------------------+
| 6 | Percentage err.  |  11.69%      |  11.26%          |
```

+---+------------------+-----------------+-----------------+

1) (no limiting)          + balanced   i/o load
2) (no limiting)          + unbalanced i/o load
3) i/o limit: 8MB/s|8MB/s  + balanced   i/o load
4) i/o limit: 8MB/s|16MB/s + balanced   i/o load
5) i/o limit: 8MB/s|8MB/s  + unbalanced i/o load
6) i/o limit: 4MB/s|8MB/s  + unbalanced i/o load

Conclusion: limiting the i/o bandwidth of cgroups by the io-throttle controller allows to reduce the percentage error of i/o performance. This permits to guarantee certain performance levels according to the defined i/o limitations and to implement more effectiveness i/o shaping policies (QoS) for user applications using the cgroups subsystem.

-Andrea

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