bluestore - Feature #38816

Deferred writes do not work for random writes

03/19/2019 07:16 PM - Марк Коренберг

Status:	In Progress	**% Done:**	0%
Priority:	Normal		
Assignee:			
Category:			
Target version:	v13.2.6		
Source:	Community (user)	**Reviewed:**	
Tags:		**Affected Versions:**	v13.2.6
Backport:		**Pull request ID:**	

Description

Well, how to reproduce:

osd.11 is a bluestore OSD with RocksDB on SSD, and main data on HDD.

```
ceph osd pool create qwe 8 8
ceph osd pool set qwe size 1
for i in `seq 0 7`; do ceph osd pg-upmap 110.$i osd.11; done
rbd create -p qwe -s test 10G
```

```
fio -ioengine=rbd -name=test -bs=4k --sync=1  -iodepth=1 -pool=qwe -rbdname=test --rw=randwrite
```
gives about 170-200 IOPS

```
<same fio> --rw=write
```
gives about 1300 IOPS.

I suppose that it works as follows:
1. Small writes are deferred
2. After some criteria, bluestore starts flushing deferred writes to HDD
3. Since random IO on HDD is really SLOW, some small buffer connected to deferred writes gets filled, and write speed sticks to HDD speed.
4. Linear writes are much faster on HDD, so small deferred writes in linear mode are coalesced to big ones and HDD manage to write them until buffer gets filled.

So. I think the buffer is very small. And in random write scenario, (Bluestore + RocksDB on SSD) is MUCH SLOWER comparing to (Filestore + WAL on SSD).

I tried increasing bluestore_max_deferred_txc and bluestore_throttle_deferred_bytes -- this does not help.

History

#1 - 03/21/2019 12:18 AM - Марк Коренберг

I want bluestore to be able to buffer(defer), say, 30 seconds of random writes in RocksDB at SSD speed. I expect background writing the data to HDD in, say, 5 minutes without throttling any incoming write requests. Roughly the same as Filestore is able to.

#2 - 03/25/2019 11:02 AM - Igor Fedotov

Mark, I'm not sure your root cause analysis is 100% valid. And to avoid any speculations I'd prefer to arrange the benchmark in the following manner and collect the following corresponding info for analysis:

1) Before each benchmark remove rbd image and restart OSD
2) Collect perf counter dump after each benchmark
3) Collect fio report for each benchmark
4) set debug bluestore to 10 and repeat steps 1-3

And a question. To what degree did you increase bluestore_max_deferred_txc? Very simplified calculation shows that it should be about 1300 x 30 = 39000 to satisfy your 30 second burst interval. I doubt anybody tested such a threshold so this is just a hypothesis to try. Many other factors might also get into the game and (negatively?) impact the process.

#3 - 03/25/2019 01:55 PM - Vitaliy Filippov

I tried similar thing when Mark asked me. In summary, you **can** enlarge your deferred queue a bit, but you can't make it give you better results on average.

The options required to do so are

```
bluestore_max_deferred_txc = 50000
bluestore_deferred_batch_ops = 10000
bluestore_throttle_cost_per_io_hdd = 100
```

But there are several problems which stop you from wanting to do it:

1) Bluestore doesn't honor the max_deferred_txc parameter. It starts to flush operations as soon as there are bluestore_deferred_batch_ops operations available. This is the biggest problem because you either wait for 10000 operations to accumulate and then flush them all at once which makes OSD just hang for 30 seconds, or you flush when you have 32-64 operations which just makes latency inconsistent like in filestore (700-0-700-0-700-0 iops), but does not provide any sort of a buffer.

2) No kind of background flush is implemented in Bluestore. So when the deferred queue fills up you just wait for the OSD to be restarted. Until then it won't flush anything.

3) Deferred writes live in the RocksDB, so if you have a lot of them they'll migrate to next levels. However I don't know how it will affect the performance. It may be ok.

#4 - 03/28/2019 02:17 PM - Neha Ojha

- Status changed from New to Need More Info

#5 - 04/07/2019 07:33 PM - Марк Коренберг

- File results.tar.xz added

Igor, I have done what you want. During OSD log inspection, take into account mtime of attached JSON files.

#6 - 04/07/2019 07:46 PM - Марк Коренберг

ceph config dump fragment:

```
osd.11      advanced bluestore_max_deferred_txc         1000000

osd.11      advanced bluestore_throttle_deferred_bytes  13421772800
```

#7 - 04/07/2019 07:51 PM - Марк Коренберг

compare:
before1.json with after1.json - with debug turned off
before2.json with after2.json - with debug turned on

#8 - 04/09/2019 01:38 PM - Igor Fedotov

@Mark, thanks for the update.
From you perf counter dumps (after?.json) one can see the following small (~4K) write statistics:

```
"bluestore_write_small": 13945, - total amount of 'small write' requests
        "bluestore_write_small_bytes": 52751806,
        "bluestore_write_small_unused": 992, - amount of write requests that hit unused block in an existing e
xtent
        "bluestore_write_small_deferred": 3469, - amount of write requests that were deferred
        "bluestore_write_small_new": 9484,  - amount of write requests that were immediately written to new lo
cation
```

So just 3469 of 13945 writes were deferred. The rest were written to HDD! disk immediately.

That's exactly how deferred writing supposed to work in BlueStore - **unaligned overwrite** (including partial overwrites) are the primary targets for deferred writing. When "unaligned" means lack of alignment with disk block size (=4K).

As a general comment I doubt it's a good idea to use this mechanics as a caching means. IMO this should be achieved either by OS or HW means. Maybe try dm-cache or lvmcache?

#9 - 04/11/2019 07:58 AM - Марк Коренберг

Igor, what about RBD? these writes are always aligned to 4K, so, as far as I understand, such writes will never be deferred. At least, it should be documented which type of write requests can be deferred.

Regarding lvmcache/dm-cache/bcache. Yes, it is possible, but makes OSD setup more complex. It will be nice if OSD could implement also this type of work. Also, scrubbing and especially deep-scrubbing will evict hot data from such caches and pull-up cold data in its place.

#10 - 04/11/2019 08:52 AM - Vitaliy Filippov

I think we are beginning to discuss something different from the original question, but ...

I checked the code and yes, it seems "write_small_new" is immediately written to the new location. If that's always the case, if that code path isn't intercepted by any previous "chunk-aligned deferred overwrite" or so... it's a problem. Does that mean that all small writes into unallocated space are written directly?

What's the point of implementing it like that?

#11 - 04/25/2019 03:49 PM - Sage Weil

Igor Fedotov wrote:

> @Mark, thanks for the update.
> From you perf counter dumps (after?.json) one can see the following small (~4K) write statistics:
>
> "bluestore_write_small": 13945, - total amount of 'small write' requests
> "bluestore_write_small_bytes": 52751806,
> "bluestore_write_small_unused": 992, - amount of write requests that hit unused block in an existing extent
> "bluestore_write_small_deferred": 3469, - amount of write requests that were deferred
> "bluestore_write_small_new": 9484, - amount of write requests that were immediately written to new location

This metric is misleading. It indicates we are writing into a new blob... but that new write is **still** deferred if it is smaller than the deferred ratio. I'll open a PR to fix that.   http://tracker.ceph.com/issues/38816

#12 - 04/25/2019 03:52 PM - Sage Weil

Vitaliy Filippov wrote:

> 1) Bluestore doesn't honor the max_deferred_txc parameter. It starts to flush operations as soon as there are bluestore_deferred_batch_ops operations available. This is the biggest problem because you either wait for 10000 operations to accumulate and then flush them all at once which makes OSD just hang for 30 seconds, or you flush when you have 32-64 operations which just makes latency inconsistent like in filestore (700-0-700-0-700-0 iops), but does not provide any sort of a buffer.

IIUC, the suggestion here is that we should defer more IO for longer, and when we do eventually perform the IO, instead of queueing **everything** that is deferred, only queue part of it? that way we can defer for a longer period without causing bigger spikes?

#13 - 04/26/2019 01:14 AM - Konstantin Shalygin

master: https://github.com/ceph/ceph/pull/27789
nautilus: https://github.com/ceph/ceph/pull/27819 merged

#14 - 05/20/2019 03:20 PM - Nathan Cutler

- Status changed from Need More Info to In Progress

#15 - 05/20/2019 04:10 PM - Vitaliy Filippov

Sage Weil wrote:

> Vitaliy Filippov wrote:
>
> > 1) Bluestore doesn't honor the max_deferred_txc parameter. It starts to flush operations as soon as there are bluestore_deferred_batch_ops operations available. This is the biggest problem because you either wait for 10000 operations to accumulate and then flush them all at once which makes OSD just hang for 30 seconds, or you flush when you have 32-64 operations which just makes latency inconsistent like in filestore (700-0-700-0-700-0 iops), but does not provide any sort of a buffer.
>
> IIUC, the suggestion here is that we should defer more IO for longer, and when we do eventually perform the IO, instead of queueing **everything** that is deferred, only queue part of it? that way we can defer for a longer period without causing bigger spikes?

Yes, I think so, and if in addition to that it has some kind of a background flush thread it will eventually clear the queue on idle. It can become a new Bluestore feature :) even though the current deferred write mechanism is also not useless. It's really optimal for small journaled writes to HDDs, it just doesn't provide buffering.

#16 - 05/30/2019 02:21 PM - Josh Durgin

- Tracker changed from Bug to Feature

Files

results.tar.xz	807 KB	04/07/2019	Марк Коренберг