http://www.coincoin.fr.eu.org/?How-to-change-your-dm-cache-write

How to change your dm-cache write mode on the fly in Linux

- 6- Webographie -

Date de mise en ligne : samedi 30 août 2014

Suppose that you are using a dm-cache based SSD disk cache, probably through the latest versions of LVM (via Lars, and see also his lvcache). Dm-cache is what I'll call an 'interposition' disk read cache, where writes to your real storage go through it ; as a result it can be in either writethrough or writeback modes. It would be nice to be able to find out what mode your cache is and also to be able to change it. As it happens this is possible, although far from obvious. This procedure also comes with a bunch of caveats and disclaimers.

(The biggest disclaimer is that I'm fumbling my way around all of this stuff and I am in no way an expert on dm-cache. Instead I'm just writing down what I've discovered and been able to do so far.)

Let's assume we're doing our caching through LVM, partly because that's what I have and partly because if you're dealing directly with dm-cache you probably already know this stuff. Our cache LV is called testing/test and it has various sub-LVs under it (the original LV, the cache metadata LV, and the cache LV itself). Our first job is to find out what the device mapper calls it. # dmsetup ls â€”tree [...] testing-test (253:5) |-testing-test_corig (253:4) | `- (9:10)` `|-testing-cache_data_cdata (253:2) |`- (8:49) `-testing-cache_data_cmeta (253:3)`- (8:49)

We can see the current write mode with 'dmsetup status' on the top level object, although the output is what they call somewhat hard to interpret :

# dmsetup status testing-test

0 10485760 cache 8 259/128000 128 4412/81920 16168 6650 39569 143311 0 0 0 **1 writeback** 2 migration_threshold 2048 mq 10 random_threshold 4 sequential_threshold 512 discard_promote_adjustment 1 read_promote_adjustment 4 write_promote_adjustment 8

I have bolded the important bit. This says that the cache is in the potentially dangerous writeback mode. To change the writeback mode we must redefine the cache, which is a little bit alarming and also requires temporarily suspending and unsuspending the device ; the latter may have impacts if, for example, you are actually using it for a mounted filesystem at the time.

DM devices are defined through what the dmsetup manpage describes as 'a table that specifies a target for each sector in the logical device'. Fortunately the tables involved are relatively simple and better yet, we can get dmsetup to give us a starting point : # dmsetup table testing-test 0 10485760 cache 253:3 253:2 253:4 128 0 default 0

To change the cache mode, we reload an altered table and then suspend and resume the device to activate our newly loaded table. For now I am going to just present the new table with the changes bolded :

# dmsetup reload â€”table '0 10485760 cache 253:3 253:2 253:4 128 **1 writethrough** default 0' testing-test

# dmsetup suspend testing-test

# dmsetup resume testing-test

At this point you can rerun 'dmsetup status' to see that the cache device has changed to writethrough.

So let's talk about the DM table we (re)created here. The first two numbers are the logical sector range and the rest of it describes the target specification for that range. The format of this specification is, to quote the big comment in the kernel's devices/md/dm-cache-target.c : cache <#feature args> []* <#policy args> []*

The original table's ending of '0 default 0' thus meant 'no feature arguments, default policy, no policy arguments'. Our new version of '1 writethrough default 0' is a change to '1 feature argument of writethrough, still the default policy, no policy arguments'. Also, if you're changing from writethrough back to writeback you don't end the table with '1 writeback default 0' because it turns out that writeback isn't a feature, it's just the default state. So you write the end of the table as '0 default 0' (as it was initially here).

Now it's time for the **important disclaimers**. The first disclaimer is that **I'm not sure what happens to any dirty blocks on your cache device if you switch from writeback to writethrough mode**. I assume that they still get flushed back to your real device and that this happens reasonably fast, but I can't prove it from reading the kernel source or my own tests and I can't find any documentation. At the moment I would call this caveat emptor until I know more. In fact I'm not truly confident what happens if you switch between writethrough and writeback in general.

(I do see some indications that there is a flush and it is rapid, but I feel very nervous about saying anything definite until I'm sure of things.)

The second disclaimer is that **at the moment the Fedora 20 LVM cannot set up writethrough cache LVs**. You can tell it to do this and it will appear to have succeeded, but the actual cache device as created at the DM layer will be writeback. This issue is what prompted my whole investigation of this at the DM level. I have filed Fedora bug #1135639 about this, although I expect it's an upstream issue.

The third disclaimer is that all of this is as of Fedora 20 and its 3.15.10-200.fc20 kernel (on 64-bit x86, in specific). All of this may change over time and probably will, as I doubt that the kernel people consider very much of this to be a stable interface.

Given all of the uncertainties involved, I don't plan to consider using LVM caching until LVM can properly create writethrough caches. Apart from the hassle involved, I'm just not happy with converting live dm-cache setups from one mode to the other right now, not unless someone who really knows this system can tell us more about what's really going on and so on.

(A great deal of my basic understanding of dmsetup usage comes from Kyle Manna's entry SSD caching using dm-cache tutorial.)

Sidebar : forcing a flush of the cache

In theory, if you want to be more sure that the cache is clean in a switch between writeback and writethrough you can explicitly force a cache clean by switching to the cleaner policy first and waiting for it to stabilize.

# dmsetup reload â€”table '0 10485760 cache 253:3 253:2 253:4 128 **0 cleaner** 0' testing-test

# dmsetup wait testing-test

I don't know how long you have to wait for this to be safe. If the cache LV (or other dm-cache device) is quiescent at the user level, I assume that you should be okay when IO to the actual devices goes quiet. But, as before, caveat emptor applies ; this is not really well documented.

Sidebar : the output of 'dmsetup status'

The output of 'dmsetup status' is mostly explained in a comment in front of the cache_status() function in devices/md/dm-cache-target.c. To save people looking for it, I will quote it here :

<#used metadata blocks>/<#total metadata blocks>

<#used cache blocks>/<#total cache blocks>

<#read hits> <#read misses> <#write hits> <#write misses>

<#demotions> <#promotions> <#dirty>

<#features> *

<#core args>

<#policy args> *

Unlike with the definition table, 'writeback' is considered a feature here. By cross-referencing this with the earlier 'dmsetup status' output we can discover that mq is the default policy and it actually has a number of arguments, the exact meanings of which I haven't researched (but see here and here).