darktable - Bug #8741

Mipmap cache: possible race condition

06/03/2012 10:07 AM - Ulrich Pegelow

Status:	Fixed	**Start date:**	06/03/2012
Priority:	Medium	**Due date:**	
Assignee:	Johannes Hanika	**% Done:**	100%
Category:	General	**Estimated time:**	0.00 hour
Target version:	Candidate for next patch release		
Affected Version:	git development version	**bitness:**	64-bit
System:		**hardware architecture:**	amd64/x86

Description

I've come over a problem with our new mipmap cache. Seems that we fall into an endless loop under certain conditions when generating thumbs.

Here is how I can reproduce:

- start with an empty mipmap cache (rm .cache/darktable/mipmap*)
- start DT (current git master)
- have more than one background threads (4 in my case)
- watch recreation of thumbs in a film roll with about 100 to 200 images

Observation:

- a few dozen thumbs are created without problems
- then the following messages are given to console

[cache] write_release: bucket not found!
[cache] read_release: not locked!
[cache] write_release: bucket not found!
[cache] read_release: not locked!
[cache] write_release: bucket not found!
[cache] read_release: not locked!

- after that no further thumbs are created
- background threads continue to run with 4 (?) cpu cores at 100% utilization
- UI remains responsive, but when going into darkroom mode only "working .." is displayed, image is not processed

Further inspection into the running background processes:

- no system calls involved as shown by 'strace -p pid'
- gdb can attach to the threads and interrupt/continue them
- interruption will always find them in cache.c, in one of the following routines:

  dt_cache_lock
  dt_cache_unlock
  dt_cache_read_get
  dt_cache_remove_bucket
  dt_cache_gc

Attached is a forced backtrace generated by manually sending SIGSEGV to one of the background threads.

As a quick remedy it helps to reduce number of background threads to 1.

Further info on my setup:

- openSUSE 12.1 64bit
- i7-2600 cpu
- 16GB RAM
- size of mipmap cache: 536870912
- opencl: no effect if enabled or disabled

Related issues:

Related to darktable - Bug #8753: Crash when scrolling in lighttable	**Duplicate**	**06/05/2012**

History

#1 - 06/03/2012 02:20 PM - Ulrich Pegelow

- Affected Version set to git development version

- File dt.out added

Additional info.

When calling DT with -d cache I get the attached output. When thumbnail generation stalls, debug output is generated less and less frequently; in the end one line of output every 30 seconds or so.

#2 - 06/05/2012 05:32 AM - Johannes Hanika

- Assignee set to Johannes Hanika

[mipmap_cache] level 0 fill 37.84/81.10 MB (46.66% in 430/1024 buffers)
[mipmap_cache] level 1 fill 0.00/81.09 MB (0.00% in 0/256 buffers)
[mipmap_cache] level 2 fill 0.00/81.09 MB (0.00% in 0/64 buffers)
[mipmap_cache] level[cache] write_release: bucket not found!
[cache] read_release: not locked!
[dev_read_history] database history for image `IMG_2356.CR2' seems to be corrupted!
3 fill 0.00/81.08 MB (0.00% in 0/16 buffers)
[mipmap_cache] level 4 fill 0.00/81.08 MB (0.00% in 0/8 buffers)
[mipmap_cache] level [full] fill 4/4 slots (100.00% in 4/8 buffers)

this might be the culprit of it ^

after that the 4/4 slots in 4/8 buffers line doesn't agree anymore how many buffers are actually locked/in use.

i guess there's just one read lock drop missing in case of generic failure in something that usually doesn't fail.

i'll try to reproduce (by breaking some db stuff locally maybe) or solve it by staring at the code real hard.

#3 - 06/05/2012 07:54 AM - Simon Spannagel

- Target version set to Candidate for next patch release

- Status changed from New to In Progress

#4 - 06/08/2012 09:53 AM - Johannes Hanika

bt

#0 0x00007ffff73ae445 in *GI_raise (sig=<optimized out>)*
at ../nptl/sysdeps/unix/sysv/linux/raise.c:64
#1 0x00007ffff73b1bab in __GI_abort () at abort.c:91
#2 0x00007ffff73a710e in __assert_fail_base (fmt=<optimized out>,
assertion=0x7ffff7b36f3c "bucket->read == 1",
file=0x7ffff7b36ed8 "/home/jo/vcs/darktable/src/common/cache.c", line=<optimized out>,
function=<optimized out>) at assert.c:94
*#3 0x00007ffff73a71b2 in __GI_*assert_fail (assertion=0x7ffff7b36f3c "bucket->read == 1",
file=0x7ffff7b36ed8 "/home/jo/vcs/darktable/src/common/cache.c", line=200,
function=0x7ffff7b37340 "dt_cache_bucket_write_lock") at assert.c:103
#4 0x00007ffff79d7b0c in dt_cache_bucket_write_lock (bucket=0x65be78)
at /home/jo/vcs/darktable/src/common/cache.c:200
#5 0x00007ffff79d7e05 in add_key_to_end_of_list (cache=0x8ed9d0, keys_bucket=0x65be50,
free_bucket=0x65be78, hash=2684354674, key=2684354674, last_bucket=0x0)
at /home/jo/vcs/darktable/src/common/cache.c:264
#6 0x00007ffff79d9119 in dt_cache_read_get (cache=0x8ed9d0, key=2684354674)
at /home/jo/vcs/darktable/src/common/cache.c:747
#7 0x00007ffff7a12560 in dt_mipmap_cache_read_get (cache=0x8ed760, buf=0x7fffe5fe8750,
imgid=115, mip=DT_MIPMAP_FULL, flags=DT_MIPMAP_BLOCKING)
at /home/jo/vcs/darktable/src/common/mipmap_cache.c:796
#8 0x00007ffff7a017d0 in dt_imageio_export_with_flags (imgid=115,
filename=0x7ffff7b3e113 "unused", format=0x7fffe5ffa0b0, format_params=0x7fffe5ff92f0,
ignore_exif=1, display_byteorder=1, high_quality=0, thumbnail_export=1)
at /home/jo/vcs/darktable/src/common/imageio.c:484
#9 0x00007ffff7a13b70 in _init_8 (buf=0x7fff8ba23140 "", width=0x7fff8ba23130,
height=0x7fff8ba23134, imgid=115, size=DT_MIPMAP_0)
at /home/jo/vcs/darktable/src/common/mipmap_cache.c:1238
#10 0x00007ffff7a12a2b in dt_mipmap_cache_read_get (cache=0x8ed760, buf=0x7fffe5ffab70,
imgid=115, mip=DT_MIPMAP_0, flags=DT_MIPMAP_BLOCKING)
at /home/jo/vcs/darktable/src/common/mipmap_cache.c:888
#11 0x00007ffff7a2f5cb in dt_image_load_job_run (job=0xea4a00)
at /home/jo/vcs/darktable/src/control/jobs/image_jobs.c:38
#12 0x00007ffff7a27d1c in dt_control_run_job (s=0x656b80)
at /home/jo/vcs/darktable/src/control/control.c:850
#13 0x00007ffff7a2859e in dt_control_work (ptr=0x656b80)
at /home/jo/vcs/darktable/src/control/control.c:1041
#14 0x00007ffff773ce9a in start_thread (arg=0x7fffe5ffb700) at pthread_create.c:308
#15 0x00007ffff746a4bd in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:112
#16 0x0000000000000000 in ?? ()

happens in thread 3 and in thread 9 at the same time for different images and thumb sizes. they start at their respective entry points:

print start_bucket
$17 = (dt_cache_bucket_t * const) 0x65be50

print start_bucket
$13 = (dt_cache_bucket_t * const) 0x65be28

and thus lock different segment locks.

the list between the buckets seems to be broken though, both end up pointing to the same new bucket and try to get the read lock at the same time (which they get and then hell breaks loose).

the big question is when does the skiplist break (i assume the garbage collection routine is at fault, will need to introduce a lot more debugging tests/assertions).

#5 - 06/08/2012 10:55 AM - Johannes Hanika

- % Done changed from 0 to 50

just scrolled through my >10k db at lowest zoom rate, with and without opencl (to confuse the threads with different task durations) after fixing a possible source of this (ed2391f).

of course this way around that doesn't provide proof that it's fixed, so i'll leave the ticket open until someone can confirm it's hard to reproduce now.

this would also be a fix i should possibly discuss with the original authors of hopscotch hashing.

#6 - 06/08/2012 01:55 PM - Ulrich Pegelow

Absolutely cool! Thanks.

I just re-created hundreds of thumbs without any issues (same settings as in my original bug report).

This really seems to have solved the issue!

#7 - 06/10/2012 05:00 AM - Johannes Hanika

- % Done changed from 50 to 100

- Status changed from In Progress to Fixed

great then :)

Files

darktable_bt_7AS3EW.txt	55 KB	06/03/2012	Ulrich Pegelow
dt.out	68.6 KB	06/03/2012	Ulrich Pegelow