rbd - Fix #11056

librbd: aio calls may block

03/06/2015 11:44 PM - Josh Durgin

Status:	Resolved	**% Done:**	0%
Priority:	High	**Spent time:**	0.00 hour
Assignee:	Jason Dillaman		
Category:			
Target version:	v9.0.3		
Source:	other	**ceph-qa-suite:**	
Tags:		**Pull request ID:**	
Backport:	firefly, hammer	**Crash signature (v1):**	
Reviewed:		**Crash signature (v2):**	
Affected Versions:			

Description

QEMU runs these from its threads, so blocking here causes the vcpu to lock up from the guest's perspective, resulting in messages in linux guests like

```
BUG: soft lockup - CPU#1 stuck for 520s!
```

One cause that has been observed is via Objecter-level throttling of requests:

With caching disabled, this blocks directly in aio_read/write().

With caching enabled, the cache does i/o while holding its lock, and if it is throttled in its flusher thread, for example, readx() and writex() will block waiting on the cache lock.

An example of this occurring with caching enabled:

```
Thread 327 (Thread 0x7fbbd67fc700 (LWP 34432)):
#0  0x00007fbbf3977705 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
#1  0x00007fbbe830ecf9 in Throttle::_wait(long) () from /lib64/librados.so.2
#2  0x00007fbbe830fb9e in Throttle::get(long, long) () from /lib64/librados.so.2
#3  0x00007fbbe907b0b3 in Objecter::throttle_op(Objecter::Op*, int) () from /usr/lib64/qemu/librbd
.so.1
#4  0x00007fbbe9088635 in Objecter::op_submit(Objecter::Op*) () from /usr/lib64/qemu/librbd.so.1
#5  0x00007fbbe827a45e in librados::IoCtxImpl::aio_operate(object_t const&, ObjectOperation*, libr
ados::AioCompletionImpl*, SnapContext const&, int) () from /lib64/librados.so.2
#6  0x00007fbbe8256515 in librados::IoCtx::aio_operate(std::string const&, librados::AioCompletion
*, librados::ObjectWriteOperation*, unsigned long, std::vector<unsigned long, std::allocator<unsig
ned long> >&) () from /lib64/librados.so.2
#7  0x00007fbbe904c2d6 in librbd::AbstractWrite::send() () from /usr/lib64/qemu/librbd.so.1
#8  0x00007fbbe9077f72 in librbd::LibrbdWriteback::write(object_t const&, object_locator_t const&,
 unsigned long, unsigned long, SnapContext const&, ceph::buffer::list const&, utime_t, unsigned lo
ng, unsigned int, Context*) () from /usr/lib64/qemu/librbd.so.1
#9  0x00007fbbe90a9098 in ObjectCacher::bh_write(ObjectCacher::BufferHead*) () from /usr/lib64/qem
u/librbd.so.1
#10 0x00007fbbe90a9f31 in ObjectCacher::flusher_entry() () from /usr/lib64/qemu/librbd.so.1
#11 0x00007fbbe90ba2cd in ObjectCacher::FlusherThread::entry() () from /usr/lib64/qemu/librbd.so.1
#12 0x00007fbbf3973df3 in start_thread () from /lib64/libpthread.so.0
#13 0x00007fbbf067f1ad in clone () from /lib64/libc.so.6
```

This also prevents completions from being finished in the ObjectCacher:

```
Thread 329 (Thread 0x7fbbd77fe700 (LWP 34430)):
#0  0x00007fbbf3979f7d in __lll_lock_wait () from /lib64/libpthread.so.0
#1  0x00007fbbf3975d41 in _L_lock_790 () from /lib64/libpthread.so.0
#2  0x00007fbbf3975c47 in pthread_mutex_lock () from /lib64/libpthread.so.0
#3  0x00007fbbe82f18eb in Mutex::Lock(bool) () from /lib64/librados.so.2
#4  0x00007fbbe9079c2b in librbd::C_OrderedWrite::finish(int) () from /usr/lib64/qemu/librbd.so.1
#5  0x00007fbbe904bb89 in Context::complete(int) () from /usr/lib64/qemu/librbd.so.1
#6  0x00007fbbe90574df in librbd::rados_req_cb(void*, void*) () from /usr/lib64/qemu/librbd.so.1
#7  0x00007fbbe8282eed in librados::C_AioSafe::finish(int) () from /lib64/librados.so.2
#8  0x00007fbbe904bb89 in Context::complete(int) () from /usr/lib64/qemu/librbd.so.1
#9  0x00007fbbe8315728 in Finisher::finisher_thread_entry() () from /lib64/librados.so.2
#10 0x00007fbbf3973df3 in start_thread () from /lib64/libpthread.so.0
#11 0x00007fbbf067f1ad in clone () from /lib64/libc.so.6
Thread 328 (Thread 0x7fbbd6ffd700 (LWP 34431)):
#0  0x00007fbbf3979f7d in __lll_lock_wait () from /lib64/libpthread.so.0
#1  0x00007fbbf397c4fc in _L_cond_lock_791 () from /lib64/libpthread.so.0
#2  0x00007fbbf397c3e7 in __pthread_mutex_cond_lock () from /lib64/libpthread.so.0
#3  0x00007fbbf3977795 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
#4  0x00007fbbe90a3b3c in ObjectCacher::maybe_wait_for_writeback(unsigned long) () from /usr/lib64
/qemu/librbd.so.1
#5  0x00007fbbe90a3f79 in ObjectCacher::C_WaitForWrite::finish(int) () from /usr/lib64/qemu/librbd
.so.1
#6  0x00007fbbe904bb89 in Context::complete(int) () from /usr/lib64/qemu/librbd.so.1
#7  0x00007fbbe8315728 in Finisher::finisher_thread_entry() () from /lib64/librados.so.2
#8  0x00007fbbf3973df3 in start_thread () from /lib64/libpthread.so.0
#9  0x00007fbbf067f1ad in clone () from /lib64/libc.so.6
```

And here's the qemu thread blocked on acquiring the cache lock:

```
Thread 1 (Thread 0x7fbbf58cca40 (LWP 34418)):
#0  0x00007fbbf3979f7d in __lll_lock_wait () from /lib64/libpthread.so.0
#1  0x00007fbbf3975d41 in _L_lock_790 () from /lib64/libpthread.so.0
#2  0x00007fbbf3975c47 in pthread_mutex_lock () from /lib64/libpthread.so.0
#3  0x00007fbbe82f18eb in Mutex::Lock(bool) () from /lib64/librados.so.2
#4  0x00007fbbe9053a7b in librbd::ImageCtx::write_to_cache(object_t, ceph::buffer::list&, unsigned
 long, unsigned long, Context*) () from /usr/lib64/qemu/librbd.so.1
#5  0x00007fbbe9066f2b in librbd::aio_write(librbd::ImageCtx*, unsigned long, unsigned long, char
const*, librbd::AioCompletion*) () from /usr/lib64/qemu/librbd.so.1
#6  0x00007fbbf59e26fb in rbd_start_aio ()
#7  0x00007fbbf59e2790 in qemu_rbd_aio_writev ()
#8  0x00007fbbf59be1ea in bdrv_co_io_em ()
#9  0x00007fbbf59c2eda in bdrv_co_do_pwritev ()
#10 0x00007fbbf59c2eda in bdrv_co_do_pwritev ()
#11 0x00007fbbf59c3964 in bdrv_co_do_rw ()
#12 0x00007fbbf59fc34a in coroutine_trampoline ()
```

Related issues:

Copied to rbd - Backport #11769: librbd: aio calls may block	**Resolved**	**03/06/2015**
Copied to rbd - Backport #11770: librbd: aio calls may block	**Resolved**	**03/06/2015**

History

#1 - 04/08/2015 01:16 AM - Josh Durgin

- Tracker changed from Bug to Fix

- Subject changed from librbd: aio_read/write() may block to librbd: aio calls may block

Seems like the most reliable way to fix this is to entirely decouple starting the i/o at all (even running ictx_check()) and calls to rbd_aio_{read,write,discard,flush}.

#2 - 04/08/2015 01:21 AM - Josh Durgin

- Assignee set to Jason Dillaman

#3 - 04/08/2015 01:53 PM - Jason Dillaman

- Status changed from New to In Progress

#4 - 04/09/2015 06:04 PM - Jason Dillaman

- Status changed from In Progress to Fix Under Review

master PR: https://github.com/ceph/ceph/pull/4318

#5 - 04/14/2015 04:06 PM - Josh Durgin

- Target version set to v0.96

#6 - 04/28/2015 04:06 PM - Josh Durgin

- Target version changed from v0.96 to v9.0.2

#7 - 05/12/2015 04:03 PM - Josh Durgin

- Target version changed from v9.0.2 to v9.0.3

#8 - 05/26/2015 06:43 PM - Josh Durgin

- Status changed from Fix Under Review to Pending Backport

#9 - 09/04/2015 09:07 AM - Nathan Cutler

- Status changed from Pending Backport to Resolved