Ceph - Backport #14873

hammer: rados bench seq crashes (v0.94.6)

02/26/2016 06:42 AM - Christian Balzer

Status:	Resolved	**Spent time:**	0.00 hour
Priority:	Normal		
Assignee:	Alexey Sheplyakov		
Target version:	v0.94.7		
Release:	hammer	**Crash signature:**	

Description

https://github.com/ceph/ceph/pull/7817

Related issues:

Duplicated by Ceph - Bug #15556: "rados bench -p scbench 10 seq" core dump	**Duplicate**	**04/21/2016**

Associated revisions

Revision 6a6754f8 - 03/03/2016 12:36 PM - Alexey Sheplyakov

hammer: tools: fix race condition in seq/rand bench (part 2)

Commit c2c6d02591519dfd15ddcb397ac440322a964deb which is intended to
cherry-pick 9bcf5f065c4ed4b10d8f98961d1f99493bcdc9b8 incorrectly resolved
a conflict by adding code where it should have been removed. The incorrect
conflict resolution can been displayed with

```
commit=c2c6d02591519dfd15ddcb397ac440322a964deb
   picked_from=9bcf5f065c4ed4b10d8f98961d1f99493bcdc9b8
   diff -u --ignore-matching-lines '^[^+]' <(git show $picked_from) <(git show $commit)
```

```
--- /dev/fd/63    2016-03-03 14:09:51.354329129 0700
+++ /dev/fd/62    2016-03-03 14:09:51.358329122 0700
@@ -76,20 +79,18 @@
 ++data.in_flight;
 -    if (!no_verify) {
 -      snprintf(data.object_contents, data.object_size, "I'm the %16dth object!", current_index);
 --      lock.Unlock();
 lock.Unlock();
 -      if (memcmp(data.object_contents, cur_contents->c_str(), data.object_size) != 0) {
 -        cerr << name[slot] << " is not correct!" << std::endl;
 -        ++errors;
 -      }
 --    } else {
 --      lock.Unlock();
 --    }
 --
 lock.Unlock();
 name[slot] = newName;
 -  }

@@ -789,11 +791,14 @@ int ObjBencher::rand_read_bench(int seconds_to_run, int num_objects, int concurr
 +    if (memcmp(data.object_contents, cur_contents->c_str(), data.object_size) != 0) {
 +      cerr << name[slot] << " is not correct!" << std::endl;
 +      ++errors;
 } else {
 +      lock.Unlock();
 +    }
 +@@ -776,11 +785,14 @@ int ObjBencher::rand_read_bench(int seconds_to_run, int num_objects, int concurr
 }
 lc.cond.Wait(lock);
 }
```

http://tracker.ceph.com/issues/14873 Fixes: #14873

Signed-off-by: Alexey Sheplyakov <asheplyakov@mirantis.com>

History

#1 - 02/26/2016 09:54 AM - Dan van der Ster

Same here, only after upgrading clients (OSDs still running 0.94.5).
Full traceback here: http://pastebin.com/HXfVKKFB

#2 - 02/26/2016 10:10 AM - Dan van der Ster

I guess it's due to the new --no-verify option: https://github.com/ceph/ceph/commit/a619b621b0a7c670eeaf163d9e2b742d13c9f517

Here's some strange behaviour when I use seq and --no-verify:

```
[11:07][root@cephmond0 (production:ceph/dwight/mon*1) ~]# rados bench -p test 5 seq -b 4096 -t 1 --no-verify
sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
0       0         0         0         0         0         -         0
benchmark_data_p01001532077488.cern.ch_257739_object0 is not correct!
benchmark_data_p01001532077488.cern.ch_257739_object1 is not correct!
benchmark_data_p01001532077488.cern.ch_257739_object2 is not correct!
benchmark_data_p01001532077488.cern.ch_257739_object3 is not correct!
benchmark_data_p01001532077488.cern.ch_257739_object4 is not correct!
benchmark_data_p01001532077488.cern.ch_257739_object5 is not correct!
...
```

Full log: http://pastebin.com/2uePDDnp

#3 - 02/26/2016 12:10 PM - Alexey Sheplyakov

Fix (not tested yet): https://github.com/ceph/ceph/pull/7817

#4 - 02/26/2016 03:04 PM - Alexey Sheplyakov

The fix is incomplete, there are more locking errors. I'll keep digging.

#5 - 02/26/2016 03:51 PM - Alexey Sheplyakov

The revised patch (https://github.com/ceph/ceph/pull/7817) fixes the problem.

#6 - 02/28/2016 07:44 AM - Nathan Cutler

- Status changed from New to Fix Under Review

#7 - 02/28/2016 07:45 AM - Nathan Cutler

Hi Alexey! Is this bug not present in master? (The usual process is to fix in master first and then backport.)

#8 - 03/01/2016 10:57 AM - Kefu Chai

Nathan and Alexey, probably we should backport https://github.com/ceph/ceph/pull/5853 also?

#9 - 03/02/2016 02:54 PM - Alexey Sheplyakov

Nathan,

> Is this bug not present in master?

No. The bug is caused by a patch (mis)cherry-picked from master to Hammer.

#10 - 03/02/2016 04:17 PM - Alexey Sheplyakov

> The bug is caused by a patch (mis)cherry-picked from master to Hammer.

For the record the patch in question is https://github.com/ceph/ceph/commit/a619b621b0a7c670eeaf163d9e2b742d13c9f517

#11 - 03/03/2016 05:42 AM - Loic Dachary

The faulty cherry pick comes from https://github.com/ceph/ceph/pull/6791

#12 - 03/03/2016 05:46 AM - Loic Dachary

- Tracker changed from Bug to Backport

- Status changed from Fix Under Review to In Progress

#13 - 03/03/2016 06:46 AM - Loic Dachary

commit=a619b621b0a7c670eeaf163d9e2b742d13c9f517 ; picked_from=$(git show --no-patch --pretty=%b $commit | perl -ne 'print if(s/.*cherry picked from commit (\w+).*/$1/)') ; diff -u --ignore-matching-lines '^[^+-]' <(git show $picked_from) <(git show $commit)

```
--- /dev/fd/63    2016-03-03 13:41:37.476976784 +0700
+++ /dev/fd/62    2016-03-03 13:41:37.476976784 +0700
@@ -47,17 +47,22 @@
     Note the decreased latencies, increased bandwidth and more reads performed.

     Signed-off-by: Piotr Dałek <piotr.dalek@ts.fujitsu.com>
+    (cherry picked from commit ca6abca63de813c83a6960f83624be8e1a86a1f8)
+
+    Conflicts:
+        src/common/obj_bencher.cc
+        src/common/obj_bencher.h

 diff --git a/src/common/obj_bencher.cc b/src/common/obj_bencher.cc
-index 128a544..8fd963b 100644
+index fbc9e34..2ed0c52 100644
 --- a/src/common/obj_bencher.cc
 +++ b/src/common/obj_bencher.cc
-@@ -169,7 +169,7 @@ void *ObjBencher::status_printer(void *_bencher) {
-
+@@ -154,7 +154,7 @@ void *ObjBencher::status_printer(void *_bencher) {
  int ObjBencher::aio_bench(
    int operation, int secondsToRun,
--  int concurrentios, int object_size, bool cleanup, const char* run_name) {
-+  int concurrentios, int object_size, bool cleanup, const char* run_name, bool no_verify) {
+-  int concurrentios, int object_size, bool cleanup, const char* run_name) {
++  int concurrentios, int object_size, bool cleanup, const char* run_name, bool no_verify) {
+   int maxObjectsToCreate,
+-  int concurrentios, int op_size, bool cleanup, const char* run_name) {
++  int concurrentios, int op_size, bool cleanup, const char* run_name, bool no_verify) {

     if (concurrentios <= 0)
       return -EINVAL;
@@ -101,7 +106,7 @@
 +        ++errors;
 +      }
```

```
 +    } else {
-+        lock.Unlock();
++      lock.Unlock();
       }
 +
       name[slot] = newName;
@@ -124,7 +129,7 @@
 +        ++errors;
 +      }
 +    } else {
-+        lock.Unlock();
++      lock.Unlock();
       }
       delete contents[slot];
     }
@@ -167,6 +172,7 @@
 -    if (memcmp(data.object_contents, contents[slot]->c_str(), data.object_size) != 0) {
 -      cerr << name[slot] << " is not correct!" << std::endl;
 -      ++errors;
++
 +    if (!no_verify) {
 +      snprintf(data.object_contents, data.object_size, "I'm the %16dth object!", index[slot]);
 +      lock.Unlock();
@@ -175,22 +181,24 @@
 +        ++errors;
 +      }
 +    } else {
-+        lock.Unlock();
++      lock.Unlock();
       }
++
       delete contents[slot];
     }
+
 diff --git a/src/common/obj_bencher.h b/src/common/obj_bencher.h
-index c67d429..e1d5e9d 100644
+index 4d89f41..52b3157 100644
 --- a/src/common/obj_bencher.h
 +++ b/src/common/obj_bencher.h
-@@ -67,8 +67,8 @@ protected:
+@@ -65,8 +65,8 @@ protected:
     int fetch_bench_metadata(const std::string& metadata_file, int* object_size, int* num_objects, int* prevPi
d);

-   int write_bench(int secondsToRun, int concurrentios, const string& run_name_meta);
+   int write_bench(int secondsToRun, int maxObjects, int concurrentios, const string& run_name_meta);
 -  int seq_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid);
 -  int rand_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid);
-+  int seq_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify
=false);
-+  int rand_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify
=false);
++  int seq_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify);
++  int rand_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify);

     int clean_up(int num_objects, int prevPid, int concurrentios);
     int clean_up_slow(const std::string& prefix, int concurrentios);
@@ -228,6 +236,7 @@
     if (i != opts.end()) {
       nspace = i->second;
     }
+-
 +  i = opts.find("no-verify");
 +  if (i != opts.end()) {
 +    no_verify = true;
```

This conflict resolution has one notable difference. The original patch had

```
-+  int seq_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify=fals
e);
-+  int rand_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify=fal
```

```
se);
```

the hammer patch has

```
++  int seq_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify);
++  int rand_read_bench(int secondsToRun, int num_objects, int concurrentios, int writePid, bool no_verify);
```

which is a different prototype. The modification was probably unnecessary but also harmless because any call to these functions without the last argument would be caught at compile time.

The other resolution are trivial (space change and protypte change).

#14 - 03/03/2016 07:10 AM - Loic Dachary

commit=c2c6d02591519dfd15ddcb397ac440322a964deb ; picked_from=9bcf5f065c4ed4b10d8f98961d1f99493bcdc9b8 ; diff -u --ignore-matching-lines '^[^+-]' <(git show $picked_from) <(git show $commit)

```
--- /dev/fd/63    2016-03-03 14:09:51.354329129 +0700
+++ /dev/fd/62    2016-03-03 14:09:51.358329122 +0700
@@ -76,20 +79,18 @@
      ++data.in_flight;
 -    if (!no_verify) {
 -      snprintf(data.object_contents, data.object_size, "I'm the %16dth object!", current_index);
--      lock.Unlock();
+       lock.Unlock();
 -      if (memcmp(data.object_contents, cur_contents->c_str(), data.object_size) != 0) {
 -        cerr << name[slot] << " is not correct!" << std::endl;
 -        ++errors;
 -      }
--    } else {
--      lock.Unlock();
--    }
--
-+    lock.Unlock();
-     name[slot] = newName;
-   }
-
-@@ -789,11 +791,14 @@ int ObjBencher::rand_read_bench(int seconds_to_run, int num_objects, int concurr
++    if (memcmp(data.object_contents, cur_contents->c_str(), data.object_size) != 0) {
++      cerr << name[slot] << " is not correct!" << std::endl;
++      ++errors;
+     } else {
+       lock.Unlock();
+     }
+@@ -776,11 +785,14 @@ int ObjBencher::rand_read_bench(int seconds_to_run, int num_objects, int concurr
        }
        lc.cond.Wait(lock);
      }
```

The conflict has been resolved incorrectly and added a code block **if (memcmp** instead of removing the intended code block. For some reason **lock.Unlock();** was not removed and creates a double unlock problem.

#15 - 03/03/2016 11:39 AM - Nathan Cutler

- Related to Bug #14958: PK11_DestroyContext() is called twice if PK11_DigestFinal() fails added

#16 - 03/03/2016 11:41 AM - Nathan Cutler

Loic, can we do the backport through #14958 which I have properly staged for backport to infernalis and hammer?

#17 - 03/03/2016 11:42 AM - Nathan Cutler

In other words, we could close this as a duplicate of #14961

#18 - 03/03/2016 11:43 AM - Nathan Cutler

- Related to deleted (Bug #14958: PK11_DestroyContext() is called twice if PK11_DigestFinal() fails)

#19 - 03/03/2016 11:43 AM - Nathan Cutler

- Duplicates Backport #14961: hammer: PK11_DestroyContext() is called twice if PK11_DigestFinal() fails added

#20 - 03/03/2016 11:46 AM - Nathan Cutler

- Duplicates deleted (Backport #14961: hammer: PK11_DestroyContext() is called twice if PK11_DigestFinal() fails)

#21 - 03/03/2016 11:47 AM - Nathan Cutler

Nevermind! I see now that's a totally different double-something issue.

#22 - 03/03/2016 01:06 PM - Loic Dachary

- Assignee set to Alexey Sheplyakov

#23 - 03/03/2016 01:07 PM - Loic Dachary

- Description updated

original description

```
On my crappy test cluster (Debian Jessie, Hammer 0.94.6) I'm seeing rados
bench crashing doing "seq" runs.
As I'm testing cache tiers at the moment I also tried it with a normal,
replicated pool with the same result.

After creating some benchmark objects with:
---
rados -p data bench 20 write -t 32 --no-cleanup
---

A consecutive run of this ends in tears:
---
# rados -p data bench 10 seq -t 32
   sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
     0       0         0         0         0         0         -         0
rados: ./common/Mutex.h:96: void Mutex::_pre_unlock(): Assertion `nlock > 0' failed.
*** Caught signal (Aborted) **
 in thread 7f1894100780
 ceph version 0.94.6 (e832001feaf8c176593e0325c8298e3f16dfb403)
 1: rados() [0x4e5e23]
 2: (()+0xf8d0) [0x7f18915268d0]
 3: (gsignal()+0x37) [0x7f188fde6067]
 4: (abort()+0x148) [0x7f188fde7448]
 5: (()+0x2e266) [0x7f188fddf266]
 6: (()+0x2e312) [0x7f188fddf312]
 7: (Mutex::Unlock()+0xb3) [0x4fda93]
 8: (ObjBencher::seq_read_bench(int, int, int, int, bool)+0x127c) [0x4da37c]
 9: (ObjBencher::aio_bench(int, int, int, int, int, bool, char const*, bool)+0x2df) [0x4ded8f]
 10: (main()+0xa664) [0x4be834]
 11: (__libc_start_main()+0xf5) [0x7f188fdd2b45]
 12: rados() [0x4c2c97]
2016-02-26 14:18:52.641052 7f1894100780 -1 *** Caught signal (Aborted) **
```

```
 in thread 7f1894100780
---

There's nothing particular outstanding or malicious in the recent events,
here are the last 2:
---
    -2> 2016-02-26 14:23:12.439214 7f18c113f780  1 -- 10.0.0.83:0/877189211 --> 10.0.0.85:6804/2921 -- osd_op(
client.31691145.0:34 benchmark_data_engtest03_32406_object32 [read 0~4096] 0.def1bb6e ack+read+known_if_redire
cted e11724) v5 -- ?+0 0x39090d0 con 0x389bed0
    -1> 2016-02-26 14:23:12.439930 7f18b4549700  1 -- 10.0.0.83:0/877189211 <== osd.11 10.0.0.34:6802/2973 1 =
=== osd_op_reply(9 benchmark_data_engtest03_32406_object7 [read 0~4096] v0'0 uv15 ondisk = 0) v6 ==== 205+0+40
96 (2792458300 0 1108541644) 0x7f1864000ca0 con 0x38bbf80
---

Note that "rand" works fine, as does "seq" on a 0.95.5 cluster.
```

#24 - 03/03/2016 01:07 PM - Loic Dachary

- Description updated

#25 - 03/03/2016 01:40 PM - Loic Dachary

- Subject changed from rados bench seq crashes in latest Hammer to hammer: rados bench seq crashes (v0.94.6)

#26 - 03/11/2016 03:09 AM - Loic Dachary

- Status changed from In Progress to Resolved

- Target version set to v0.94.7

#27 - 07/11/2016 05:26 PM - Nathan Cutler

- Related to Bug #15556: "rados bench -p scbench 10 seq" core dump added

#28 - 07/12/2016 06:46 AM - Nathan Cutler

- Related to deleted (Bug #15556: "rados bench -p scbench 10 seq" core dump)

#29 - 07/12/2016 06:46 AM - Nathan Cutler

- Duplicated by Bug #15556: "rados bench -p scbench 10 seq" core dump added