rgw - Bug #20861

Object data loss in RGW when multipart upload completion times out

07/31/2017 02:25 PM - Varada Kari

Status:	Resolved	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:			
Category:			
Target version:	v12.1.0		
Source:	Community (dev)	**Reviewed:**	
Tags:		**Affected Versions:**	v10.2.9, v12.1.0
Backport:	jewel kraken	**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	
Severity:	2 - major	**Crash signature:**	

Description

Recently in one of the clusters, lost some of objects if a multipart complete upload times out and been retried before the first request can be finished. Please refer to the following events happened on the cluster to result in a object loss.

```
2017-07-25 02:01:54.817492 7f4d67e86700  1 civetweb: 0x3bb3b90: X.X.X.X - - [25/Jul/2017:02:01:51 +0530] "POST TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64

2017-07-25 02:02:06.554676 7fdef1469700  1 civetweb: 0x45a1050: X.X.X.X - - [25/Jul/2017:02:01:54 +0530] "PUT TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64

2017-07-25 02:02:06.569207 7fb7eaf15700  1 civetweb: 0x3c66bd0: X.X.X.X - - [25/Jul/2017:02:02:06 +0530] "GET TESTBUCKET/file1.txt HTTP/1.1" 200 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64

2017-07-25 02:03:29.650093 7f5f52f1d700  1 civetweb: 0x7f5fc40: X.X.X.X - - [25/Jul/2017:02:02:06 +0530] "POST TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64 <<== first post request to complete the upload

2017-07-25 02:03:42.069385 7fe522ef5700  1 civetweb: 0x445aa10: X.X.X.X - - [25/Jul/2017:02:03:16 +0530] "POST TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64 <<== same request been retried again. Both requests succeeded

2017-07-25 04:17:53.543050 7fb19ce31700  1 civetweb: 0x4021c10: X.X.X.X - - [25/Jul/2017:04:17:53 +0530] "GET TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64  << This GET resulted in a 404

2017-07-25 04:17:53.548596 7f9062ffd700  1 civetweb: 0x44e0ee0: X.X.X.X - - [25/Jul/2017:04:17:53 +0530] "HEAD TESTBUCKET/file1.txt HTTP/1.1" -1 0 - Boto/2.40.0 Python/2.7.3 Linux/3.2.0-4-amd64 <<== This is sucessful call and we can do a s3 ls to get the size(fetching from omap) etc... but the object read fails.
```

Timeout happened when the cluster is in rebalance due to osd failure.
I can reproduce the same issue with help of delay in the following code. And the multipart objects are marked for delete by gc when the first request succeds after second request finishes.

```
=========================
diff --git a/src/rgw/rgw_op.cc b/src/rgw/rgw_op.cc
index aae6372..0ecf842 100644
--- a/src/rgw/rgw_op.cc
+++ b/src/rgw/rgw_op.cc
@ -5403,6 +5403,12 @ void RGWCompleteMultipart::execute()
if (op_ret < 0)
return;

+  if (g_conf->rgw_introduce_latency) {
+    dout(0) << func << "sleeping for 5 secs" << dendl;
+    sleep(5);
+  }
```

```
+
+
// remove the upload obj
int r = store->delete_obj(*static_cast<RGWObjectCtx *>(s->obj_ctx),
s->bucket_info, meta_obj, 0);
```

========================

So far i am not able to come up with a fix where we can store the state of the multipart complete upload progress and deleting the meta_obj. This problem exists in all the releases and results in a data loss. Following is the snippet from master branch.

Please find the logs and the required information below.

1. bin/ceph --admin-daemon /tmp/ceph-asok.kXejsq/client.rgw.22202.asok config get rgw_introduce_latency
    - DEVELOPER MODE: setting PATH, PYTHONPATH and LD_LIBRARY_PATH *** {
      "rgw_introduce_latency": "true"
      }

1. bin/radosgw-admin gc list --include-all

```
[     {
"tag": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.11\u0000",
"time": "2017-07-31 14:07:02.0.532372s",
"objs": [            {
"pool": "default.rgw.buckets.data",
"oid": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.1",
"key": "",
"instance": ""
},            {
"pool": "default.rgw.buckets.data",
"oid": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.2",
"key": "",
"instance": ""
},            {
"pool": "default.rgw.buckets.data",
"oid": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.3",
"key": "",
"instance": ""
},            {
"pool": "default.rgw.buckets.data",
"oid": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.4",
"key": "",
"instance": ""
},            {
"pool": "default.rgw.buckets.data",
"oid": "69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.5",
"key": "",
"instance": ""
}
]
}
]
```

1. bin/rados ls -p default.rgw.buckets.data

```
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1_junkfile
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.2
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.3
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.4
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.5
69510504-522a-4f6c-a169-62cb6c7eac51.4106.1__multipart_junkfile.2~4Br_wpZyqMbSpNniRBOxQbNij0efqrq.1
```

Once the gc process the requests, except the head object rest of the multiparts are deleted leading to the whole object loss. Omap contains the meta information about the object but fails to read the object and results in a 404.

Related issues:

Copied to rgw - Backport #20900: jewel: Object data loss in RGW when multipa...	**Resolved**
Copied to rgw - Backport #20901: kraken: Object data loss in RGW when multip...	**Rejected**

History

#1 - 08/01/2017 11:30 AM - Abhishek Varshney

- Project changed from Ceph to rgw

- Category deleted (22)

#2 - 08/01/2017 03:20 PM - Abhishek Varshney

Adding more details:

The root cause of missing parts of a multipart object is a race in 2 contending calls to RGWCompleteMultipart function in rgw_op.cc. This scenario happens when 1 thread is still writing the manifest file and has not deleted the meta object yet and a second thread comes, it is still able to read the meta object and proceeds ahead and adds all the existing entries in manifest file to gc before writing the entries to manifest again as it appears in RGWRados::update_gc_chain. This leads to parts being gc'ed. This problem is more prominent when there is degradation in the cluster and writes are latent, thus leading to race.

A proposed fix can be to lock the meta object in RGWCompleteMultipart. One problem in this approach is to identify the right timeout for the lock. We have seen request to RGWCompleteMultipart, taking a few minutes to complete. This PR can be referred for proposed fix ->
https://github.com/ceph/ceph/pull/16732/files?w=1

#3 - 08/01/2017 03:45 PM - Matt Benjamin

questions posted in PR

#4 - 08/03/2017 01:59 AM - Jeegn Chen

Using lock here should mitigate the issue but there seems still some problems.
For example, if rgw_mp_lock_max_time is small and the first RGWCompleteMultipart happens to take more than rgw_mp_lock_max_time to complete (maybe because the op is blocked by peering or recovery or some busy disks), then the race condition remain.
If rgw_mp_lock_max_time is quite large and the rgw serving RGWCompleteMultipart happens to crash, then the client cannot retry RGWCompleteMultipart successfully within rgw_mp_lock_max_time seconds.

What about a no-lock solution?
For example, before RGWCompleteMultipart put the parts list in the old manifest into the GC list, let's complete the old manifest to the the new one. If we are encountering the race condition, then the first parts in both manifests should be the same. If we find that part in common, we don't put the old parts into gc list so that the parts in both manifest won't be deleted by accident.

#5 - 08/03/2017 02:16 AM - Matt Benjamin

I'm a bit skeptical this approach is much more compelling. I'm a bit more interested if we can get substantially more atomicity and clarity. For example, maybe promote the lock into an atomic serializer playing the role of your compare-parts step? I think we will need better primitives eventually. OTOH, after merging this, I realized that we really should not be using these RADOS low-level primitives in rgw_op.{h,cc}. We ~~should~~ have an RGWRados api to do this. I'm not sure what it should be.

#6 - 08/03/2017 02:22 AM - Matt Benjamin

(updated this with more detail)

#7 - 08/03/2017 02:59 AM - Varada Kari

If we want to go with no lock, then should have the state saved in omap or something more relevant to store with a time stamp. we can have an upper bound to make the state invalid, in case of rgw crash or peering events etc...If any thread comes in that time, we can check the time bound state and return with a 500. If the same thread still continues to process, we can renew the time if not completed by upper bound of our time stamp.
But this approach also doesn't guarantee who will win, if we send the complete multipart upload to multiple rgw's at the same time, i hope that is not a valid scenario.
I agree with Matt on the placement and using the lock in rgw_op.cc, as pointed already we don't have correct semantics or primitives needed to achieve that in current code. If we want to come up with atomic primitives across all RGWs, we need to build kind of gossip protocol to acquire locks.

#8 - 08/03/2017 03:52 AM - Abhishek Varshney

Jeegn Chen : I agree that the current approach is not completely fool-proof and would only reduce the probability of data loss. Keeping in view the points you raised, the default value of rgw_mp_lock_max_time is kept as 600s. This is based on the default timeouts and retries that most popular S3 SDKs have, like boto has num_retries set to 5 and timeout set to 60s by default. The hope here is that no complete call would be made after 600s and the clients would have thrown exception before that. Again, this is not fool-proof if you have overridden these default values.
The alternate approach that you mentioned to compare existing objects in manifest to new ones before gc'ing them is something that I too had in mind and had proposed it in the PR thread. But I have limited understanding and I am not sure if there can be any side-effects of that.

#9 - 08/03/2017 05:57 PM - Matt Benjamin

- Status changed from New to Pending Backport

- Backport changed from jewel to jewel (kraken?)

#10 - 08/03/2017 05:58 PM - Matt Benjamin

jewel backport PR is 16767

#11 - 08/03/2017 06:31 PM - Nathan Cutler

- Backport changed from jewel (kraken?) to jewel kraken

The chances of a kraken 11.2.2 seeing the light of day are low-to-non-existent, but OK.

#12 - 08/03/2017 06:32 PM - Nathan Cutler

- Copied to Backport #20900: jewel: Object data loss in RGW when multipart upload completion times out added

#13 - 08/03/2017 06:32 PM - Nathan Cutler

- Copied to Backport #20901: kraken: Object data loss in RGW when multipart upload completion times out added

#14 - 09/12/2017 07:54 PM - Nathan Cutler

- Status changed from Pending Backport to Resolved