Ceph - Subtask #5213

Feature # 4929 (Resolved): Erasure encoded placement group

Subtask # 5046 (Resolved): Factor out PG logs, PG missing

unit tests for src/osd/PGLog.{cc,h}

05/30/2013 03:29 PM - Loic Dachary

Status:	Resolved	**Start date:**	05/30/2013
Priority:	Normal	**Due date:**	
Assignee:	Loic Dachary	**% Done:**	100%
Category:	OSD	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	20.50 hours
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:		**Pull request ID:**	

Description

work in progress

Focus on the functions related to log merging (merge_log and proc_replica_log) rather than the simpler functions. It may not be wise to cover 100% of the LOC because the redefinition of the API will force to rewrite the tests. When the logic is complex the tests will not need to be rewritten completely. But the accessors will change and since their semantic is trivial, the tests won't help in any way.

Pull requests

- PGLog::merge_old_entry
- PGLog::merge_log
- PGLog::proc_replica_log

Discussions

- rationale for a PGLog::merge_old_entry case
- PGLog::merge_log clarification
- PGLog::rewind_divergent_log use case

Associated revisions

Revision 04e89a40 - 06/10/2013 12:08 PM - Loic Dachary

unit tests for PGLog::merge_log

The tests covers 100% of the LOC of merge_log. It is broken down
in 7 cases to enumerate all the situations it must address. Each case
is isolated in a independant code block where the conditions are
reproduced. Where possible and sensible to read, a code block covers
as much lines as possible. For instance:

```
The log entry (1,3) deletes the object x9 but the olog entry (2,3)
  modifies it and is authoritative : the log entry (1,3) is divergent.
```

is the only test case covering a dozen "if" statements and half a
dozen "while/for" loops. It covers all the lines but it would be
useful to create others scenarii in the future.

Each test is made of a comment describing the test case, the
definition of the data structures to create the desired conditons, a
sequence of EXPECT_* checking that they are met, a single call to
merge_log and another sequence of EXPECT_* (ordered to be easy to
compare with the first sequence) checking all the desired side
effects.

The TestPGLog.cc file was untabified to improve the display of ascii art when it is output as part of a diff.

http://tracker.ceph.com/issues/5213 refs #5213

Signed-off-by: Loic Dachary <loic@dachary.org>

Revision 8f141c45 - 06/10/2013 12:08 PM - Loic Dachary

unit tests for PGLog::rewind_divergent_log

The tests covers 100% of the LOC of rewind_divergent_log. There are three situations :

- throw an assert because the data is inconsistent
- special case when the entire logs is divergent
- regular workflow where all divergent entries are run to merge_old_entry

http://tracker.ceph.com/issues/5213 refs #5213

Signed-off-by: Loic Dachary <loic@dachary.org>

Revision e11cc1c8 - 06/19/2013 06:52 PM - Loic Dachary

add constness to PGLog::proc_replica_log

The function is made const by replacing a single call to log.objects[] with log.objects.find. The olog argument is also a const and does not require any change.

http://tracker.ceph.com/issues/5213 refs #5213

Signed-off-by: Loic Dachary <loic@dachary.org>

Revision 4d77443d - 06/19/2013 06:52 PM - Loic Dachary

unit tests for PGLog::proc_replica_log

The tests covers 100% of the LOC of proc_replica_log. It is broken down in 7 cases to enumerate all the situations it must address. Each case is isolated in a independant code block where the conditions are reproduced.

All tests are done on omissing and oinfo because they are the only data structures that can be modified by proc_replica_log.

The first case is a noop and checks that only last_complete gets updated when there are no logs.

The following case includes entries that are supposed to be ignored (x7, x8 and xa), however this is not an actual proof that the code ignoring them is actually run : it only shows in the code coverage report.

```
The log entry (1,3) modifies the object x9 but the olog entry
   (2,3) deletes it : log is authoritative and the object is added
   to missing. x7 is divergent and ignored. x8 has a more recent
   version in the log and the olog entry is ignored. xa is past
   last_backfill and ignored.
```

The other cases are a variation of the first case with minimal changes to make them easier to understand and adapt. For instance most of them start with a tail that is the same (object with hash x5 and both at version 1,1).

http://tracker.ceph.com/issues/5213 refs #5213

Signed-off-by: Loic Dachary <loic@dachary.org>

History

#1 - 05/31/2013 11:51 AM - Loic Dachary

- Description updated

- % Done changed from 0 to 10

#2 - 05/31/2013 11:54 AM - Loic Dachary

- Description updated

#3 - 06/02/2013 04:43 AM - Loic Dachary

- Description updated

#4 - 06/07/2013 06:36 AM - Loic Dachary

- Description updated

#5 - 06/07/2013 06:38 AM - Loic Dachary

- Description updated

#6 - 06/07/2013 06:38 AM - Loic Dachary

- % Done changed from 10 to 30

#7 - 06/07/2013 06:39 AM - Loic Dachary

- Description updated

#8 - 06/07/2013 09:18 AM - Loic Dachary

- Description updated

#9 - 06/08/2013 01:19 AM - Loic Dachary

- Description updated

- % Done changed from 30 to 40

#10 - 06/10/2013 04:15 PM - Loic Dachary

related thread

```
<sjust> I'm not sure I understand your question
<sjust> it shouldn't happen with default settings
<loicd> that's what confused me I guess. When can it happen ?
<sjust> but if you turn the min log entries option to a small number, the primary might wake up after a brief nap to find that all of its entries are divergent
<sjust> although now it seems like that should cause backfill
<sjust> hmm
* loicd thinking
<sjust> ah, if log.tail == auth_log.head
<loicd> but
<loicd> assert(newhead > log.tail);
<loicd> would throw then, right ?
<sjust> yeah, you are right
<sjust> I don't think that case can happen
* loicd digging the history of changes for a hint
<sjust> loicd: I think there may be a conflict with calc_acting where it chooses which peers can be activated, you may want to make a bug
<loicd> ok, sjust I'll check this tomorrow morning. Thanks for the advice :-)
<sjust> I think the assert is actually wrong, but we'll never see the problem case due to osd_min_pg_log_entries
<loicd> the history goes straight back to when rewind_divergent_log was not yet a function on its own, difficult to compare
<sjust> yeah, the history probably won't be a lot of help
<loicd> not sure to understand why osd_min_pg_log_entries makes it impossible for newhead to be <= log.tail
* loicd thinking
<sjust> not impossible, but very unlikely
* loicd contemplates
<loicd>  // The logs must overlap.
<loicd>   assert(log.head >= olog.tail && olog.head >= log.tail);
<sjust> right, the question is whether the constraint is
<sjust> log.head >= olog.tail
<sjust> or
<sjust> log.head > olog.tail
<sjust> in the latter case, there may be no entries in common
<sjust> in the former case, there must be at least 1 entry in common
<loicd> so the assert can only be triggered if olog.head == log.tail because of the previous assert on  olog.head >= log.tail)
```

#11 - 06/17/2013 02:47 AM - Loic Dachary

- Description updated

- % Done changed from 40 to 60

#12 - 06/17/2013 12:25 PM - Loic Dachary

- Description updated

#13 - 06/19/2013 11:14 PM - Loic Dachary

- Status changed from In Progress to Resolved

- % Done changed from 60 to 100

- translation missing: en.field_remaining_hours set to 0.00

#14 - 08/05/2013 07:22 AM - Loic Dachary

- Category changed from common to OSD

- Estimated time set to 0.00 h