Ceph - Fix #9566

osd: prioritize recovery of OSDs with most work to do

09/22/2014 08:43 AM - Sheldon Mustard

Status:	Need More Info	**Start date:**	09/22/2014
Priority:	High	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Source:	Support	**Affected Versions:**	
Tags:		**ceph-qa-suite:**	
Backport:		**Pull request ID:**	
Reviewed:		**Crash signature:**	

Description

Assume 72 hours for host replacement/reprovisioning SLA. When host goes down (hardware failure), we expect complete cluster recovery in ~48+ hours. If we lose one more disk anywhere else during this interval, we lose write access (min_size=2) to a subset of 36 million of objects. Hopefully, much smaller subset. If another disk fails, we lose data permanently. Losing a host and another two disks (out of 576 disks in total) within 48+ hours is a non-zero probability. While we understand that this is an inherent risk with any distributed system, we're not very happy about the fact that the most time spent in recovery is when less than 10% of objects are degraded (very long tail). If we maintained a more or less constant repair rate (for simplicity, let's not account for client/recover throttling), we could've reduced the exposure window from 48 to 12 or less hours.

Note: osd_max_backfill is the default (i.e. 10)

History

#1 - 09/28/2014 09:26 PM - Sage Weil

The recovery slows simply because there are fewer PGs left degraded and the per-pg (or per-osd) recovery rate is limited... there are fewer parallel backfills in progress.

I wonder if we can shorten the tail by prioritizing recovery reservations based on how many pending recoveries/backfills there are per OSD? that would make the OSDs with the most work to do get to the front of the line and improve overlap?

#2 - 09/29/2014 02:36 PM - Sage Weil

- Tracker changed from Feature to Fix

- Subject changed from concerns about backfilling rates to osd: prioritize recovery of OSDs with most work to do

- Priority changed from Normal to High

- Target version set to 0.88

Factor the number of backfill (or backfill_wait) pgs on the OSD into the recovery priority. Make sure this accounts for both work on the primary and the replicas.

#3 - 09/29/2014 02:37 PM - Sage Weil

- Backport set to giant,firefly

#4 - 09/30/2014 01:19 PM - Ian Colle

- Assignee set to Loic Dachary

#5 - 10/14/2014 01:11 PM - Samuel Just

- Target version changed from 0.88 to 0.89

#6 - 10/16/2014 05:11 PM - Loic Dachary

Related commits:

- osd: prioritize backfill based on **how** degraded
- osd: prioritize recovery for degraded pgs

#7 - 10/16/2014 05:24 PM - Loic Dachary

- Status changed from New to In Progress

#8 - 10/22/2014 04:47 PM - Loic Dachary

- Status changed from In Progress to Need Review

Here is a draft for review: https://github.com/ceph/ceph/pull/2778 if this sounds reasonable I'll write tests. Otherwise, I'll rework it ;-)

#9 - 11/11/2014 01:06 PM - Samuel Just

- Target version changed from 0.89 to v.91

#10 - 11/25/2014 01:03 PM - Samuel Just

- Target version changed from v.91 to v.actually90

#11 - 12/04/2014 01:23 PM - Loic Dachary

- Status changed from Need Review to In Progress

#12 - 12/05/2014 01:12 AM - Loic Dachary

- Status changed from In Progress to Need Review

#13 - 12/08/2014 02:50 AM - Loic Dachary

- Status changed from Need Review to In Progress

#14 - 12/08/2014 05:01 AM - Loic Dachary

```
$ rm -fr dev out ;  mkdir -p dev ; MON=1 OSD=10 ./vstart.sh -X -n -l mon osd
$ ./rados -p rbd bench 330 write --run-name backfill2 --no-cleanup
$ ./ceph osd pool set rbd hashpspool false
```

The cluster comes back clean within ~6 minutes with or without the patch.

#15 - 12/09/2014 12:05 PM - Loic Dachary

- Status changed from In Progress to Testing

- Target version changed from v.actually90 to v.actually91

#16 - 12/10/2014 05:09 AM - Loic Dachary

When the busiest OSD becomes less busy, the priority of the reservations queued when it was more busy should be lowered. In other words priorities should be re-evaluated from time to time otherwise they may have an effect that is inverse to what was originally intended.

#17 - 12/10/2014 08:47 AM - Loic Dachary

In a cluster with 10 OSDs and a osd_max_backfills of 5, it is possible for the OSD with the most PGs to backfill to wait for an OSD to be accept a reservation because they are all already busy. If the busiest OSD was given priority it would be less likely to wait. Since it will be last to complete, making sure it waits as little as possible reduces the duration of recovery.

#18 - 12/10/2014 09:20 AM - Loic Dachary

After a

```
./ceph osd crush reweight osd.0 10
```

in a cluster with ten identical OSDs and a single pool of 1024 PGs with three replicas.

```
rm -f osd.dump
while : ; do
  for id in $(seq 0 9) ; do
    echo -n "$id: "
    ./ceph --format xml daemon osd.$id dump_reservations 2>/dev/null | \
      xmlstarlet sel -t -m '//local_reservations/in_progress' -v 'format-number(count(*), "00")' -o ' ' \
                     -t -m '//remote_reservations/in_progress' -v 'format-number(count(*), "00")' -o ' '
  done
  echo
  sleep 1
done >> osd.dump
```

displays

```
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
0: 00 00 1: 05 00 2: 10 00 3: 10 00 4: 10 00 5: 10 01 6: 10 02 7: 10 00 8: 10 04 9: 10 05
0: 10 10 1: 10 04 2: 10 07 3: 10 03 4: 10 03 5: 10 04 6: 10 02 7: 10 02 8: 10 06 9: 10 04
0: 10 10 1: 10 04 2: 10 05 3: 10 01 4: 10 04 5: 10 02 6: 10 03 7: 10 03 8: 10 05 9: 10 06
0: 10 10 1: 10 05 2: 10 05 3: 10 01 4: 10 04 5: 10 01 6: 10 00 7: 10 04 8: 10 02 9: 10 05
0: 10 10 1: 10 05 2: 10 06 3: 10 02 4: 10 04 5: 10 02 6: 10 01 7: 10 03 8: 10 04 9: 10 03
0: 10 10 1: 10 07 2: 10 06 3: 10 01 4: 10 03 5: 10 04 6: 10 00 7: 10 05 8: 10 02 9: 10 03
0: 10 10 1: 10 04 2: 10 08 3: 10 03 4: 10 02 5: 10 04 6: 10 01 7: 10 03 8: 10 03 9: 10 03
0: 10 10 1: 10 05 2: 10 04 3: 10 02 4: 10 03 5: 10 02 6: 10 03 7: 10 04 8: 10 04 9: 10 03
0: 10 10 1: 10 04 2: 10 04 3: 10 04 4: 10 02 5: 10 02 6: 10 01 7: 10 03 8: 10 05 9: 10 02
0: 10 10 1: 10 04 2: 10 04 3: 10 03 4: 10 03 5: 10 01 6: 10 01 7: 10 04 8: 10 07 9: 10 04
0: 10 10 1: 10 03 2: 10 01 3: 10 03 4: 10 02 5: 10 02 6: 10 02 7: 10 04 8: 10 09 9: 10 03
0: 10 10 1: 10 04 2: 10 06 3: 10 02 4: 10 01 5: 10 03 6: 10 01 7: 10 04 8: 10 02 9: 09 04
0: 10 10 1: 10 03 2: 10 03 3: 10 05 4: 10 04 5: 10 02 6: 10 04 7: 10 04 8: 10 02 9: 08 03
0: 10 10 1: 10 03 2: 10 04 3: 10 05 4: 10 02 5: 10 03 6: 10 05 7: 10 04 8: 10 04 9: 07 02
0: 10 10 1: 10 02 2: 10 05 3: 10 03 4: 10 03 5: 10 04 6: 10 03 7: 10 06 8: 10 03 9: 06 03
0: 10 10 1: 10 01 2: 08 04 3: 10 03 4: 10 01 5: 10 07 6: 10 05 7: 10 05 8: 10 04 9: 05 02
0: 10 10 1: 10 00 2: 07 01 3: 10 02 4: 10 05 5: 10 09 6: 10 06 7: 09 01 8: 10 04 9: 04 03
0: 10 10 1: 08 00 2: 06 02 3: 10 02 4: 10 06 5: 10 05 6: 10 04 7: 08 01 8: 10 05 9: 04 05
0: 10 10 1: 08 02 2: 05 04 3: 10 01 4: 10 04 5: 10 05 6: 10 04 7: 07 04 8: 10 04 9: 03 02
0: 10 10 1: 07 01 2: 04 04 3: 10 02 4: 08 03 5: 10 03 6: 10 04 7: 06 06 8: 10 04 9: 02 02
0: 10 10 1: 06 01 2: 03 04 3: 10 03 4: 06 06 5: 09 01 6: 10 03 7: 06 04 8: 10 03 9: 00 01
0: 10 10 1: 04 05 2: 02 03 3: 10 04 4: 04 04 5: 08 02 6: 10 03 7: 05 04 8: 09 03 9: 00 01
0: 10 10 1: 02 06 2: 01 03 3: 10 04 4: 04 04 5: 07 02 6: 10 01 7: 03 04 8: 09 04 9: 00 04
0: 10 10 1: 01 03 2: 00 04 3: 10 02 4: 03 04 5: 06 02 6: 10 01 7: 01 05 8: 08 02 9: 00 04
0: 10 10 1: 00 05 2: 00 03 3: 10 03 4: 01 03 5: 04 01 6: 10 04 7: 00 04 8: 06 02 9: 00 04
0: 10 10 1: 00 06 2: 00 02 3: 10 04 4: 00 05 5: 03 00 6: 10 03 7: 00 04 8: 06 04 9: 00 02
0: 10 10 1: 00 04 2: 00 02 3: 08 05 4: 00 04 5: 00 01 6: 10 01 7: 00 06 8: 03 04 9: 00 03
0: 10 10 1: 00 02 2: 00 04 3: 05 04 4: 00 03 5: 00 03 6: 10 01 7: 00 05 8: 00 05 9: 00 03
0: 10 10 1: 00 02 2: 00 04 3: 04 02 4: 00 03 5: 00 02 6: 07 01 7: 00 07 8: 00 03 9: 00 02
0: 10 05 1: 00 03 2: 00 03 3: 00 03 4: 00 03 5: 00 02 6: 04 01 7: 00 03 8: 00 04 9: 00 02
0: 10 00 1: 00 04 2: 00 03 3: 00 03 4: 00 01 5: 00 03 6: 00 03 7: 00 00 8: 00 02 9: 00 00
0: 10 00 1: 00 03 2: 00 01 3: 00 02 4: 00 03 5: 00 04 6: 00 02 7: 00 02 8: 00 02 9: 00 01
0: 10 00 1: 00 03 2: 00 01 3: 00 02 4: 00 01 5: 00 02 6: 00 04 7: 00 04 8: 00 03 9: 00 02
0: 10 00 1: 00 02 2: 00 02 3: 00 01 4: 00 03 5: 00 01 6: 00 04 7: 00 05 8: 00 00 9: 00 03
0: 10 00 1: 00 01 2: 00 04 3: 00 03 4: 00 03 5: 00 02 6: 00 02 7: 00 03 8: 00 01 9: 00 02
0: 10 00 1: 00 03 2: 00 01 3: 00 03 4: 00 03 5: 00 02 6: 00 04 7: 00 02 8: 00 01 9: 00 02
0: 10 00 1: 00 03 2: 00 01 3: 00 01 4: 00 03 5: 00 02 6: 00 05 7: 00 02 8: 00 01 9: 00 01
0: 10 00 1: 00 02 2: 00 02 3: 00 04 4: 00 02 5: 00 02 6: 00 03 7: 00 01 8: 00 02 9: 00 03
0: 10 00 1: 00 04 2: 00 02 3: 00 04 4: 00 01 5: 00 03 6: 00 02 7: 00 01 8: 00 03 9: 00 04
0: 10 00 1: 00 04 2: 00 01 3: 00 02 4: 00 00 5: 00 02 6: 00 03 7: 00 01 8: 00 03 9: 00 03
0: 10 00 1: 00 04 2: 00 01 3: 00 04 4: 00 02 5: 00 03 6: 00 00 7: 00 01 8: 00 03 9: 00 01
```

```
0: 10 00 1: 00 02 2: 00 02 3: 00 03 4: 00 03 5: 00 04 6: 00 00 7: 00 00 8: 00 04 9: 00 01
0: 10 00 1: 00 01 2: 00 04 3: 00 03 4: 00 02 5: 00 03 6: 00 02 7: 00 00 8: 00 05 9: 00 00
0: 10 00 1: 00 02 2: 00 01 3: 00 06 4: 00 02 5: 00 01 6: 00 01 7: 00 01 8: 00 06 9: 00 02
0: 10 00 1: 00 00 2: 00 00 3: 00 02 4: 00 06 5: 00 03 6: 00 02 7: 00 01 8: 00 04 9: 00 04
0: 10 00 1: 00 00 2: 00 00 3: 00 02 4: 00 04 5: 00 01 6: 00 04 7: 00 02 8: 00 03 9: 00 03
0: 10 00 1: 00 00 2: 00 03 3: 00 00 4: 00 03 5: 00 03 6: 00 04 7: 00 02 8: 00 04 9: 00 02
0: 10 00 1: 00 02 2: 00 02 3: 00 03 4: 00 02 5: 00 02 6: 00 02 7: 00 01 8: 00 04 9: 00 03
0: 04 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
```

#19 - 12/10/2014 10:23 AM - Loic Dachary

Same as above with a max backfill set to 5:

```
0: 00 00 1: 05 00 2: 05 00 3: 05 00 4: 05 01 5: 05 02 6: 05 01 7: 05 01 8: 05 01 9: 05 02
0: 05 05 1: 05 05 2: 05 03 3: 05 00 4: 05 01 5: 05 00 6: 05 00 7: 05 01 8: 05 00 9: 05 03
0: 05 05 1: 05 04 2: 05 03 3: 05 02 4: 05 01 5: 05 01 6: 05 01 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 03 2: 05 02 3: 05 02 4: 05 00 5: 05 01 6: 05 01 7: 05 02 8: 05 02 9: 05 00
0: 05 05 1: 05 03 2: 05 04 3: 05 02 4: 05 01 5: 05 00 6: 05 00 7: 05 02 8: 05 01 9: 05 02
0: 05 05 1: 05 03 2: 05 02 3: 05 00 4: 05 02 5: 05 01 6: 05 00 7: 05 03 8: 05 02 9: 05 02
0: 05 05 1: 05 02 2: 05 05 3: 05 01 4: 05 02 5: 05 02 6: 05 01 7: 05 01 8: 05 03 9: 05 00
0: 05 05 1: 05 02 2: 05 01 3: 05 02 4: 05 02 5: 05 02 6: 05 00 7: 05 03 8: 05 02 9: 05 01
0: 05 05 1: 05 03 2: 05 03 3: 05 02 4: 05 01 5: 05 02 6: 05 01 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 02 2: 05 02 3: 05 02 4: 05 02 5: 05 00 6: 05 00 7: 05 04 8: 05 02 9: 05 02
0: 05 05 1: 05 02 2: 05 01 3: 05 03 4: 05 02 5: 05 02 6: 05 01 7: 05 00 8: 05 01 9: 05 02
0: 05 05 1: 05 02 2: 05 02 3: 05 02 4: 05 03 5: 05 01 6: 05 01 7: 05 00 8: 05 02 9: 05 03
0: 05 05 1: 05 02 2: 05 02 3: 05 01 4: 05 01 5: 05 01 6: 05 01 7: 05 03 8: 05 01 9: 05 03
0: 05 05 1: 05 02 2: 05 02 3: 05 03 4: 05 01 5: 05 01 6: 05 03 7: 05 00 8: 05 01 9: 05 02
0: 05 05 1: 05 03 2: 05 03 3: 05 00 4: 05 03 5: 05 02 6: 05 04 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 01 2: 05 04 3: 05 04 4: 05 01 5: 05 02 6: 05 01 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 00 2: 05 00 3: 05 04 4: 05 01 5: 05 03 6: 05 00 7: 05 02 8: 05 03 9: 05 01
0: 05 05 1: 05 01 2: 05 03 3: 05 02 4: 05 00 5: 05 02 6: 05 03 7: 05 03 8: 05 01 9: 04 01
0: 05 05 1: 05 03 2: 05 03 3: 05 01 4: 05 03 5: 05 03 6: 05 01 7: 05 01 8: 05 02 9: 04 00
0: 05 05 1: 05 01 2: 05 03 3: 05 01 4: 05 01 5: 05 05 6: 05 01 7: 05 00 8: 05 02 9: 04 00
0: 05 05 1: 05 03 2: 05 02 3: 05 03 4: 05 01 5: 05 01 6: 05 02 7: 05 04 8: 05 01 9: 03 00
0: 05 05 1: 05 01 2: 05 01 3: 05 01 4: 05 01 5: 05 04 6: 05 01 7: 05 02 8: 05 02 9: 02 02
0: 05 05 1: 05 01 2: 05 02 3: 05 02 4: 05 02 5: 05 03 6: 05 01 7: 05 03 8: 05 01 9: 01 01
0: 05 05 1: 04 02 2: 03 01 3: 05 00 4: 05 02 5: 05 02 6: 05 03 7: 05 03 8: 05 02 9: 00 01
0: 05 05 1: 02 03 2: 02 00 3: 05 01 4: 04 02 5: 05 01 6: 05 04 7: 05 02 8: 05 01 9: 00 00
0: 05 05 1: 02 01 2: 01 01 3: 05 01 4: 02 01 5: 05 00 6: 05 02 7: 04 04 8: 05 02 9: 00 01
0: 05 05 1: 01 03 2: 01 01 3: 05 01 4: 01 03 5: 05 00 6: 05 01 7: 03 01 8: 05 01 9: 00 02
0: 05 05 1: 01 03 2: 00 02 3: 05 04 4: 01 01 5: 04 02 6: 05 02 7: 00 02 8: 05 01 9: 00 03
0: 05 05 1: 00 03 2: 00 01 3: 05 01 4: 00 02 5: 02 00 6: 05 01 7: 00 02 8: 04 02 9: 00 02
0: 05 05 1: 00 02 2: 00 02 3: 05 01 4: 00 02 5: 00 01 6: 05 01 7: 00 02 8: 02 00 9: 00 04
0: 05 05 1: 00 02 2: 00 02 3: 03 03 4: 00 00 5: 00 02 6: 05 00 7: 00 02 8: 00 03 9: 00 03
0: 05 05 1: 00 02 2: 00 02 3: 00 02 4: 00 01 5: 00 02 6: 05 01 7: 00 01 8: 00 00 9: 00 02
0: 05 05 1: 00 02 2: 00 02 3: 00 00 4: 00 02 5: 00 01 6: 04 03 7: 00 02 8: 00 02 9: 00 01
0: 05 00 1: 00 02 2: 00 02 3: 00 00 4: 00 02 5: 00 02 6: 00 03 7: 00 00 8: 00 01 9: 00 02
0: 05 00 1: 00 01 2: 00 00 3: 00 00 4: 00 02 5: 00 02 6: 00 01 7: 00 00 8: 00 01 9: 00 01
0: 05 00 1: 00 01 2: 00 00 3: 00 01 4: 00 01 5: 00 02 6: 00 01 7: 00 00 8: 00 03 9: 00 01
0: 05 00 1: 00 02 2: 00 01 3: 00 01 4: 00 00 5: 00 00 6: 00 02 7: 00 02 8: 00 02 9: 00 01
0: 05 00 1: 00 00 2: 00 00 3: 00 03 4: 00 00 5: 00 00 6: 00 02 7: 00 03 8: 00 01 9: 00 03
0: 05 00 1: 00 02 2: 00 01 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 01 8: 00 02 9: 00 03
0: 05 00 1: 00 00 2: 00 00 3: 00 00 4: 00 01 5: 00 02 6: 00 02 7: 00 02 8: 00 00 9: 00 04
0: 05 00 1: 00 00 2: 00 00 3: 00 00 4: 00 03 5: 00 01 6: 00 02 7: 00 00 8: 00 00 9: 00 00
0: 05 00 1: 00 00 2: 00 02 3: 00 01 4: 00 00 5: 00 03 6: 00 01 7: 00 00 8: 00 03 9: 00 01
0: 05 00 1: 00 00 2: 00 01 3: 00 00 4: 00 02 5: 00 00 6: 00 03 7: 00 00 8: 00 00 9: 00 01
0: 05 00 1: 00 00 2: 00 01 3: 00 01 4: 00 01 5: 00 00 6: 00 01 7: 00 02 8: 00 02 9: 00 00
0: 05 00 1: 00 00 2: 00 01 3: 00 00 4: 00 01 5: 00 00 6: 00 04 7: 00 01 8: 00 00 9: 00 02
0: 05 00 1: 00 01 2: 00 00 3: 00 00 4: 00 02 5: 00 01 6: 00 02 7: 00 00 8: 00 02 9: 00 03
0: 05 00 1: 00 01 2: 00 00 3: 00 02 4: 00 01 5: 00 01 6: 00 00 7: 00 00 8: 00 03 9: 00 01
0: 05 00 1: 00 00 2: 00 02 3: 00 01 4: 00 01 5: 00 01 6: 00 00 7: 00 00 8: 00 01 9: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
```

#20 - 12/10/2014 12:14 PM - Loic Dachary

```
while read osd weight ; do

      ./ceph osd crush reweight osd.$osd $weight
done <<EOF
0 1
1 1
2 1
3 1
4 1
5 1
6 4
7 5
8 7
9 10
EOF
```

creates the following dump

```
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 04 5: 05 05 6: 05 00 7: 05 05 8: 05 05 9: 05 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 03 5: 05 05 6: 05 01 7: 05 05 8: 05 05 9: 05 00
0: 05 05 1: 05 03 2: 05 05 3: 05 05 4: 05 05 5: 05 05 6: 05 01 7: 05 05 8: 05 05 9: 05 00
0: 03 05 1: 05 04 2: 05 05 3: 05 05 4: 05 05 5: 05 05 6: 05 05 7: 05 05 8: 05 02 9: 05 01
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 04 5: 05 05 6: 05 05 7: 05 05 8: 05 05 9: 05 01
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 04 5: 05 04 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 04 5: 05 03 6: 05 05 7: 05 05 8: 05 02 9: 05 05
0: 05 04 1: 05 05 2: 05 05 3: 05 05 4: 05 03 5: 05 02 6: 05 05 7: 05 05 8: 05 03 9: 05 05
0: 05 02 1: 05 05 2: 05 05 3: 05 04 4: 05 03 5: 05 01 6: 05 05 7: 05 05 8: 05 00 9: 05 03
0: 05 03 1: 05 05 2: 05 05 3: 05 04 4: 05 03 5: 05 02 6: 05 05 7: 05 05 8: 05 05 9: 05 02
0: 05 02 1: 05 03 2: 05 05 3: 05 04 4: 05 05 5: 05 02 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 05 02 1: 05 02 2: 05 05 3: 05 04 4: 05 05 5: 05 03 6: 05 05 7: 05 05 8: 05 05 9: 05 01
0: 05 02 1: 05 02 2: 05 05 3: 05 05 4: 05 05 5: 05 04 6: 05 05 7: 05 05 8: 05 05 9: 05 03
0: 05 03 1: 05 01 2: 05 05 3: 05 03 4: 05 05 5: 05 05 6: 05 05 7: 05 05 8: 05 05 9: 05 01
0: 05 04 1: 05 01 2: 05 05 3: 05 04 4: 05 05 5: 05 05 6: 05 05 7: 05 05 8: 05 03 9: 05 02
0: 05 03 1: 05 01 2: 05 05 3: 05 02 4: 05 05 5: 05 05 6: 05 05 7: 05 05 8: 05 04 9: 05 03
0: 05 05 1: 05 01 2: 05 03 3: 05 02 4: 05 03 5: 05 04 6: 05 05 7: 05 05 8: 05 02 9: 05 05
0: 05 05 1: 05 01 2: 05 03 3: 05 03 4: 05 04 5: 05 04 6: 05 05 7: 05 05 8: 05 03 9: 05 02
0: 05 05 1: 05 01 2: 05 02 3: 05 04 4: 05 04 5: 05 02 6: 05 05 7: 05 05 8: 05 04 9: 05 04
0: 05 05 1: 05 02 2: 05 02 3: 05 04 4: 05 03 5: 05 00 6: 05 05 7: 05 05 8: 05 02 9: 05 05
0: 05 05 1: 05 01 2: 05 01 3: 05 04 4: 05 01 5: 04 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 05 05 1: 05 00 2: 05 04 3: 05 03 4: 05 04 5: 04 01 6: 05 05 7: 05 05 8: 05 04 9: 05 05
0: 05 05 1: 05 00 2: 05 04 3: 05 04 4: 05 03 5: 03 01 6: 05 05 7: 05 05 8: 05 01 9: 05 05
0: 05 02 1: 05 00 2: 05 05 3: 05 03 4: 05 01 5: 02 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 05 01 1: 05 00 2: 03 05 3: 05 03 4: 05 01 5: 01 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 05 00 1: 05 01 2: 02 05 3: 05 03 4: 05 02 5: 01 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 05 00 1: 05 00 2: 01 04 3: 05 03 4: 05 02 5: 01 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 02 00 1: 05 00 2: 01 04 3: 05 05 4: 03 01 5: 01 02 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 01 01 1: 05 02 2: 00 04 3: 05 04 4: 02 03 5: 00 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 01 1: 05 02 2: 00 04 3: 05 03 4: 02 02 5: 00 03 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 01 1: 05 01 2: 00 03 3: 05 03 4: 02 02 5: 00 05 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 00 01 1: 05 00 2: 00 02 3: 05 03 4: 01 04 5: 00 04 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 02 1: 05 00 2: 00 03 3: 05 03 4: 00 04 5: 00 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 03 1: 03 00 2: 00 03 3: 05 00 4: 00 05 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 03 1: 00 02 2: 00 04 3: 05 01 4: 00 02 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 03 1: 00 02 2: 00 03 3: 05 02 4: 00 02 5: 00 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 05 1: 00 01 2: 00 01 3: 04 01 4: 00 03 5: 00 03 6: 05 05 7: 05 05 8: 05 05 9: 05 02
0: 00 02 1: 00 02 2: 00 02 3: 01 01 4: 00 05 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 00
0: 00 01 1: 00 03 2: 00 00 3: 00 01 4: 00 05 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 01
0: 00 02 1: 00 04 2: 00 01 3: 00 04 4: 00 03 5: 00 03 6: 05 05 7: 05 04 8: 05 05 9: 05 03
0: 00 04 1: 00 01 2: 00 01 3: 00 02 4: 00 04 5: 00 04 6: 05 04 7: 05 05 8: 05 05 9: 05 03
0: 00 02 1: 00 01 2: 00 02 3: 00 03 4: 00 04 5: 00 04 6: 05 05 7: 05 03 8: 05 05 9: 05 03
0: 00 00 1: 00 00 2: 00 01 3: 00 02 4: 00 04 5: 00 04 6: 05 05 7: 05 03 8: 05 05 9: 05 02
0: 00 00 1: 00 02 2: 00 01 3: 00 03 4: 00 01 5: 00 04 6: 05 05 7: 05 05 8: 05 05 9: 05 03
0: 00 00 1: 00 04 2: 00 00 3: 00 01 4: 00 00 5: 00 03 6: 05 05 7: 05 04 8: 05 05 9: 05 03
0: 00 00 1: 00 04 2: 00 02 3: 00 01 4: 00 00 5: 00 03 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 00 00 1: 00 05 2: 00 03 3: 00 01 4: 00 02 5: 00 02 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 00 02 1: 00 05 2: 00 01 3: 00 01 4: 00 03 5: 00 00 6: 05 05 7: 05 05 8: 05 05 9: 05 02
0: 00 04 1: 00 01 2: 00 01 3: 00 03 4: 00 02 5: 00 04 6: 05 05 7: 05 03 8: 05 05 9: 05 02
0: 00 03 1: 00 03 2: 00 01 3: 00 01 4: 00 02 5: 00 03 6: 05 05 7: 05 04 8: 05 05 9: 05 04
0: 00 02 1: 00 00 2: 00 01 3: 00 01 4: 00 03 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 05
```

```
0: 00 01 1: 00 01 2: 00 01 3: 00 00 4: 00 02 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 04
0: 00 00 1: 00 00 2: 00 01 3: 00 00 4: 00 01 5: 00 01 6: 05 05 7: 05 05 8: 05 05 9: 05 03
0: 00 02 1: 00 00 2: 00 01 3: 00 00 4: 00 01 5: 00 01 6: 05 05 7: 05 03 8: 05 05 9: 05 04
0: 00 02 1: 00 00 2: 00 01 3: 00 02 4: 00 00 5: 00 02 6: 05 05 7: 05 03 8: 05 05 9: 05 05
0: 00 00 1: 00 00 2: 00 02 3: 00 00 4: 00 01 5: 00 03 6: 05 05 7: 05 04 8: 05 05 9: 05 05
0: 00 01 1: 00 02 2: 00 02 3: 00 01 4: 00 01 5: 00 02 6: 05 05 7: 05 05 8: 05 05 9: 05 05
0: 00 00 1: 00 01 2: 00 02 3: 00 02 4: 00 01 5: 00 03 6: 05 04 7: 05 04 8: 05 05 9: 05 05
0: 00 00 1: 00 01 2: 00 01 3: 00 01 4: 00 02 5: 00 03 6: 05 04 7: 05 05 8: 05 05 9: 05 05
0: 00 02 1: 00 01 2: 00 00 3: 00 02 4: 00 02 5: 00 01 6: 05 02 7: 05 05 8: 05 05 9: 05 05
0: 00 01 1: 00 02 2: 00 01 3: 00 03 4: 00 00 5: 00 03 6: 05 02 7: 05 05 8: 05 05 9: 05 05
0: 00 01 1: 00 00 2: 00 02 3: 00 02 4: 00 00 5: 00 02 6: 04 03 7: 05 05 8: 05 05 9: 05 05
0: 00 01 1: 00 01 2: 00 01 3: 00 01 4: 00 00 5: 00 01 6: 02 04 7: 05 04 8: 05 05 9: 05 05
0: 00 00 1: 00 01 2: 00 00 3: 00 02 4: 00 01 5: 00 02 6: 01 03 7: 05 04 8: 05 05 9: 05 05
0: 00 01 1: 00 01 2: 00 00 3: 00 03 4: 00 01 5: 00 01 6: 00 03 7: 05 04 8: 05 05 9: 05 05
0: 00 00 1: 00 04 2: 00 01 3: 00 00 4: 00 02 5: 00 00 6: 00 04 7: 05 03 8: 05 05 9: 05 05
0: 00 01 1: 00 04 2: 00 00 3: 00 02 4: 00 01 5: 00 01 6: 00 03 7: 05 04 8: 05 05 9: 05 05
0: 00 00 1: 00 02 2: 00 01 3: 00 02 4: 00 02 5: 00 00 6: 00 02 7: 05 05 8: 05 05 9: 05 05
0: 00 02 1: 00 02 2: 00 01 3: 00 01 4: 00 01 5: 00 03 6: 00 03 7: 05 03 8: 05 05 9: 05 05
0: 00 02 1: 00 04 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 05 7: 05 03 8: 05 05 9: 05 05
0: 00 01 1: 00 02 2: 00 00 3: 00 01 4: 00 00 5: 00 01 6: 00 05 7: 05 02 8: 05 05 9: 05 05
0: 00 00 1: 00 04 2: 00 00 3: 00 02 4: 00 00 5: 00 00 6: 00 05 7: 05 01 8: 05 05 9: 05 05
0: 00 00 1: 00 03 2: 00 01 3: 00 01 4: 00 02 5: 00 01 6: 00 03 7: 05 02 8: 05 05 9: 05 05
0: 00 00 1: 00 03 2: 00 01 3: 00 02 4: 00 03 5: 00 00 6: 00 02 7: 05 04 8: 05 05 9: 05 05
0: 00 01 1: 00 01 2: 00 03 3: 00 03 4: 00 01 5: 00 01 6: 00 02 7: 05 04 8: 05 04 9: 05 05
0: 00 01 1: 00 02 2: 00 03 3: 00 01 4: 00 02 5: 00 00 6: 00 02 7: 05 04 8: 05 05 9: 05 05
0: 00 01 1: 00 02 2: 00 03 3: 00 02 4: 00 01 5: 00 01 6: 00 05 7: 05 02 8: 05 05 9: 05 05
0: 00 00 1: 00 02 2: 00 02 3: 00 04 4: 00 02 5: 00 02 6: 00 02 7: 05 03 8: 05 03 9: 05 05
0: 00 00 1: 00 00 2: 00 03 3: 00 04 4: 00 01 5: 00 01 6: 00 03 7: 05 04 8: 05 04 9: 05 05
0: 00 00 1: 00 00 2: 00 03 3: 00 04 4: 00 02 5: 00 00 6: 00 03 7: 05 03 8: 05 05 9: 05 05
0: 00 01 1: 00 01 2: 00 04 3: 00 01 4: 00 02 5: 00 00 6: 00 03 7: 05 03 8: 05 05 9: 05 05
0: 00 04 1: 00 02 2: 00 00 3: 00 00 4: 00 01 5: 00 00 6: 00 05 7: 05 04 8: 05 03 9: 05 05
0: 00 05 1: 00 02 2: 00 00 3: 00 00 4: 00 01 5: 00 00 6: 00 05 7: 03 03 8: 05 02 9: 05 05
0: 00 04 1: 00 03 2: 00 00 3: 00 01 4: 00 00 5: 00 01 6: 00 04 7: 01 03 8: 05 01 9: 05 05
0: 00 01 1: 00 02 2: 00 00 3: 00 01 4: 00 01 5: 00 00 6: 00 04 7: 00 03 8: 05 03 9: 05 05
0: 00 01 1: 00 01 2: 00 00 3: 00 01 4: 00 00 5: 00 02 6: 00 04 7: 00 02 8: 05 04 9: 05 05
0: 00 01 1: 00 02 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 02 7: 00 03 8: 05 05 9: 05 05
0: 00 00 1: 00 02 2: 00 00 3: 00 01 4: 00 01 5: 00 00 6: 00 03 7: 00 04 8: 05 04 9: 05 05
0: 00 01 1: 00 00 2: 00 03 3: 00 00 4: 00 01 5: 00 00 6: 00 03 7: 00 02 8: 04 03 9: 05 04
0: 00 01 1: 00 00 2: 00 02 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 04 8: 00 02 9: 05 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 02 6: 00 01 7: 00 02 8: 00 03 9: 05 00
0: 00 00 1: 00 01 2: 00 00 3: 00 01 4: 00 00 5: 00 01 6: 00 02 7: 00 02 8: 00 02 9: 05 00
0: 00 00 1: 00 00 2: 00 00 3: 00 02 4: 00 01 5: 00 00 6: 00 01 7: 00 04 8: 00 02 9: 05 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
```

As expected the last OSD to finish is the busiest OSD which also is the one with the higher weight.
After it stabilizes, the weights are shifted again as follows:

```
while read osd weight ; do
      ./ceph osd crush reweight osd.$osd $weight
done <<EOF
0 10
1 7
2 5
3 4
4 1
5 1
6 1
7 1
8 1
9 1
EOF
```

immediately afterwards creates

```
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
0: 00 00 1: 05 00 2: 05 00 3: 05 00 4: 05 01 5: 03 01 6: 05 03 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 04 2: 05 01 3: 05 01 4: 05 01 5: 05 01 6: 05 02 7: 05 00 8: 05 00 9: 05 04
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 01 5: 05 03 6: 05 00 7: 05 01 8: 05 00 9: 05 02
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 02 5: 05 02 6: 05 00 7: 05 05 8: 05 03 9: 05 04
0: 05 05 1: 05 05 2: 05 05 3: 05 03 4: 05 01 5: 05 00 6: 05 01 7: 05 04 8: 05 04 9: 05 02
0: 05 05 1: 05 05 2: 05 04 3: 05 02 4: 05 01 5: 05 02 6: 05 03 7: 05 03 8: 05 05 9: 05 05
```

```
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 01 5: 05 00 6: 05 01 7: 05 01 8: 05 03 9: 05 05
0: 05 05 1: 05 05 2: 05 05 3: 05 04 4: 05 00 5: 05 00 6: 05 01 7: 05 02 8: 05 02 9: 05 04
0: 05 05 1: 05 05 2: 05 04 3: 05 04 4: 05 01 5: 05 01 6: 05 00 7: 05 01 8: 05 01 9: 05 03
0: 05 05 1: 05 05 2: 05 04 3: 05 02 4: 05 02 5: 05 00 6: 05 00 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 05 2: 05 02 3: 05 02 4: 05 03 5: 05 00 6: 05 00 7: 05 00 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 01 3: 05 02 4: 05 01 5: 05 00 6: 05 00 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 05 00 5: 05 00 6: 05 00 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 00 3: 05 02 4: 05 00 5: 05 00 6: 05 02 7: 05 02 8: 05 02 9: 05 00
0: 05 05 1: 05 05 2: 05 01 3: 05 01 4: 05 00 5: 05 00 6: 05 00 7: 05 01 8: 05 02 9: 05 02
0: 05 05 1: 05 05 2: 05 03 3: 05 01 4: 05 02 5: 05 01 6: 05 00 7: 05 00 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 01 3: 05 00 4: 05 02 5: 05 01 6: 05 00 7: 05 00 8: 05 02 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 05 00 5: 05 00 6: 05 01 7: 05 00 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 03 3: 05 04 4: 05 03 5: 05 01 6: 05 03 7: 05 00 8: 05 02 9: 05 00
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 05 02 5: 05 00 6: 05 00 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 02 3: 05 02 4: 05 02 5: 05 01 6: 05 02 7: 05 00 8: 05 00 9: 05 00
0: 05 05 1: 05 05 2: 05 03 3: 05 03 4: 05 00 5: 05 00 6: 05 01 7: 05 00 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 05 00 5: 05 00 6: 05 01 7: 05 01 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 04 4: 05 00 5: 05 02 6: 05 01 7: 05 01 8: 05 00 9: 05 02
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 05 00 5: 05 03 6: 05 00 7: 05 01 8: 05 00 9: 05 02
0: 05 05 1: 05 05 2: 05 02 3: 05 05 4: 04 00 5: 05 00 6: 05 02 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 04 02 5: 05 00 6: 05 00 7: 05 00 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 04 01 5: 05 00 6: 05 00 7: 05 00 8: 05 00 9: 05 02
0: 05 05 1: 05 05 2: 05 01 3: 05 02 4: 03 00 5: 05 00 6: 05 00 7: 05 01 8: 05 01 9: 05 00
0: 05 05 1: 05 05 2: 05 02 3: 05 03 4: 02 00 5: 05 00 6: 05 00 7: 05 01 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 04 3: 05 02 4: 02 00 5: 05 02 6: 05 00 7: 05 02 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 04 3: 05 04 4: 02 00 5: 05 03 6: 05 00 7: 05 02 8: 05 00 9: 05 00
0: 05 05 1: 05 04 2: 05 03 3: 05 05 4: 02 01 5: 05 02 6: 05 00 7: 05 01 8: 05 00 9: 05 01
0: 05 05 1: 05 04 2: 05 02 3: 05 05 4: 02 02 5: 05 03 6: 05 00 7: 05 00 8: 05 02 9: 05 00
0: 05 05 1: 05 02 2: 05 01 3: 05 05 4: 01 04 5: 05 01 6: 05 01 7: 05 00 8: 05 02 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 00 02 5: 05 02 6: 05 02 7: 05 02 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 02 3: 05 05 4: 00 00 5: 05 02 6: 05 01 7: 05 01 8: 05 00 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 00 01 5: 05 05 6: 05 01 7: 05 01 8: 05 01 9: 05 01
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 00 01 5: 05 03 6: 05 00 7: 05 01 8: 05 01 9: 05 02
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 00 01 5: 05 00 6: 05 00 7: 05 04 8: 05 00 9: 04 01
0: 05 05 1: 05 05 2: 05 04 3: 05 05 4: 00 01 5: 05 00 6: 05 00 7: 05 03 8: 05 01 9: 04 03
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 01 5: 05 01 6: 05 00 7: 05 02 8: 05 01 9: 04 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 00 5: 05 02 6: 05 01 7: 05 01 8: 05 01 9: 03 02
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 04 5: 03 00 6: 05 01 7: 05 01 8: 05 02 9: 03 01
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 03 5: 02 01 6: 05 02 7: 05 00 8: 05 01 9: 02 01
0: 05 05 1: 05 04 2: 05 05 3: 05 05 4: 00 02 5: 02 01 6: 05 01 7: 04 01 8: 05 00 9: 02 01
0: 05 05 1: 05 00 2: 05 04 3: 05 05 4: 00 01 5: 02 01 6: 05 02 7: 03 02 8: 05 00 9: 01 00
0: 05 05 1: 05 00 2: 05 02 3: 05 05 4: 00 01 5: 02 03 6: 05 01 7: 01 00 8: 05 00 9: 00 01
0: 05 05 1: 05 02 2: 05 05 3: 05 05 4: 00 00 5: 02 03 6: 04 00 7: 00 02 8: 05 02 9: 00 05
0: 05 05 1: 05 05 2: 05 03 3: 05 03 4: 00 01 5: 01 01 6: 03 01 7: 00 01 8: 05 01 9: 00 04
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 00 01 5: 00 00 6: 02 03 7: 00 01 8: 05 02 9: 00 03
0: 05 05 1: 05 05 2: 05 02 3: 05 04 4: 00 01 5: 00 01 6: 02 03 7: 00 02 8: 05 03 9: 00 01
0: 05 05 1: 05 05 2: 05 01 3: 05 04 4: 00 02 5: 00 00 6: 02 03 7: 00 01 8: 05 02 9: 00 01
0: 05 05 1: 05 05 2: 05 02 3: 05 03 4: 00 03 5: 00 01 6: 01 03 7: 00 00 8: 05 01 9: 00 03
0: 05 05 1: 05 05 2: 05 01 3: 05 05 4: 00 04 5: 00 01 6: 01 02 7: 00 01 8: 05 02 9: 00 01
0: 05 05 1: 05 05 2: 05 01 3: 05 05 4: 00 04 5: 00 01 6: 00 02 7: 00 02 8: 05 01 9: 00 04
0: 05 05 1: 05 05 2: 05 01 3: 05 04 4: 00 04 5: 00 02 6: 00 00 7: 00 02 8: 05 00 9: 00 01
0: 05 05 1: 05 05 2: 05 02 3: 05 02 4: 00 04 5: 00 01 6: 00 00 7: 00 03 8: 04 01 9: 00 01
0: 05 05 1: 05 05 2: 05 03 3: 05 02 4: 00 02 5: 00 03 6: 00 02 7: 00 02 8: 03 00 9: 00 00
0: 05 05 1: 05 05 2: 05 05 3: 05 01 4: 00 00 5: 00 04 6: 00 01 7: 00 00 8: 01 01 9: 00 00
0: 05 05 1: 05 05 2: 05 04 3: 05 03 4: 00 02 5: 00 03 6: 00 02 7: 00 00 8: 00 01 9: 00 03
0: 05 05 1: 05 05 2: 05 01 3: 05 04 4: 00 03 5: 00 03 6: 00 02 7: 00 02 8: 00 02 9: 00 02
0: 05 04 1: 05 05 2: 05 03 3: 05 05 4: 00 03 5: 00 03 6: 00 02 7: 00 02 8: 00 02 9: 00 03
0: 05 04 1: 05 05 2: 05 05 3: 05 05 4: 00 02 5: 00 03 6: 00 03 7: 00 02 8: 00 01 9: 00 01
0: 05 04 1: 05 05 2: 05 05 3: 05 05 4: 00 02 5: 00 02 6: 00 04 7: 00 03 8: 00 00 9: 00 01
0: 05 02 1: 05 05 2: 05 05 3: 05 03 4: 00 01 5: 00 01 6: 00 01 7: 00 04 8: 00 00 9: 00 02
0: 05 01 1: 05 05 2: 05 05 3: 05 04 4: 00 03 5: 00 00 6: 00 03 7: 00 02 8: 00 03 9: 00 00
0: 05 01 1: 05 05 2: 05 03 3: 05 05 4: 00 03 5: 00 00 6: 00 04 7: 00 02 8: 00 03 9: 00 01
0: 05 01 1: 05 05 2: 05 02 3: 05 05 4: 00 00 5: 00 01 6: 00 03 7: 00 04 8: 00 02 9: 00 03
0: 05 03 1: 05 05 2: 05 01 3: 05 05 4: 00 01 5: 00 03 6: 00 03 7: 00 04 8: 00 02 9: 00 02
0: 05 05 1: 05 05 2: 05 01 3: 05 05 4: 00 00 5: 00 03 6: 00 01 7: 00 05 8: 00 03 9: 00 01
0: 05 04 1: 05 05 2: 05 02 3: 05 05 4: 00 00 5: 00 01 6: 00 01 7: 00 05 8: 00 03 9: 00 00
0: 05 01 1: 05 05 2: 05 01 3: 05 05 4: 00 03 5: 00 01 6: 00 02 7: 00 05 8: 00 02 9: 00 01
0: 05 01 1: 05 05 2: 05 00 3: 05 03 4: 00 03 5: 00 03 6: 00 01 7: 00 05 8: 00 03 9: 00 00
0: 05 02 1: 05 05 2: 05 03 3: 05 03 4: 00 02 5: 00 02 6: 00 01 7: 00 05 8: 00 05 9: 00 02
0: 05 03 1: 05 05 2: 05 02 3: 05 05 4: 00 02 5: 00 01 6: 00 01 7: 00 05 8: 00 02 9: 00 02
0: 05 05 1: 05 05 2: 05 01 3: 05 05 4: 00 03 5: 00 03 6: 00 00 7: 00 02 8: 00 02 9: 00 03
0: 05 05 1: 05 05 2: 05 02 3: 05 05 4: 00 01 5: 00 01 6: 00 01 7: 00 02 8: 00 03 9: 00 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 02 5: 00 03 6: 00 01 7: 00 00 8: 00 01 9: 00 02
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 02 5: 00 04 6: 00 02 7: 00 00 8: 00 01 9: 00 03
0: 05 05 1: 05 05 2: 05 03 3: 05 05 4: 00 02 5: 00 02 6: 00 01 7: 00 01 8: 00 02 9: 00 02
```

```
0: 05 05 1: 05 05 2: 05 04 3: 05 03 4: 00 03 5: 00 03 6: 00 01 7: 00 01 8: 00 04 9: 00 03
0: 05 05 1: 05 05 2: 05 05 3: 05 00 4: 00 01 5: 00 00 6: 00 02 7: 00 01 8: 00 01 9: 00 01
0: 05 05 1: 05 05 2: 05 05 3: 05 03 4: 00 01 5: 00 02 6: 00 03 7: 00 00 8: 00 02 9: 00 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 00 5: 00 03 6: 00 00 7: 00 01 8: 00 00 9: 00 00
0: 05 05 1: 05 05 2: 05 05 3: 05 05 4: 00 01 5: 00 04 6: 00 00 7: 00 00 8: 00 00 9: 00 00
0: 05 05 1: 05 04 2: 05 05 3: 05 05 4: 00 01 5: 00 03 6: 00 00 7: 00 00 8: 00 00 9: 00 01
0: 05 05 1: 05 02 2: 05 05 3: 05 05 4: 00 02 5: 00 01 6: 00 01 7: 00 01 8: 00 00 9: 00 00
0: 05 05 1: 05 03 2: 05 05 3: 05 03 4: 00 00 5: 00 02 6: 00 01 7: 00 02 8: 00 01 9: 00 00
0: 05 05 1: 05 04 2: 05 05 3: 05 02 4: 00 01 5: 00 00 6: 00 02 7: 00 02 8: 00 01 9: 00 00
0: 05 05 1: 05 01 2: 05 05 3: 05 01 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 02 2: 04 05 3: 05 03 4: 00 00 5: 00 00 6: 00 00 7: 00 02 8: 00 01 9: 00 00
0: 05 05 1: 05 04 2: 03 05 3: 05 04 4: 00 00 5: 00 01 6: 00 01 7: 00 01 8: 00 02 9: 00 01
0: 05 05 1: 05 04 2: 01 05 3: 05 04 4: 00 01 5: 00 01 6: 00 01 7: 00 00 8: 00 03 9: 00 00
0: 05 05 1: 05 04 2: 00 05 3: 05 02 4: 00 03 5: 00 00 6: 00 01 7: 00 01 8: 00 01 9: 00 00
0: 05 05 1: 05 01 2: 00 05 3: 05 01 4: 00 00 5: 00 00 6: 00 02 7: 00 02 8: 00 01 9: 00 00
0: 05 05 1: 05 00 2: 00 05 3: 05 03 4: 00 00 5: 00 00 6: 00 03 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 01 2: 00 05 3: 05 03 4: 00 00 5: 00 00 6: 00 02 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 04 2: 00 05 3: 05 01 4: 00 00 5: 00 02 6: 00 01 7: 00 02 8: 00 00 9: 00 00
0: 05 05 1: 05 05 2: 00 05 3: 05 00 4: 00 01 5: 00 00 6: 00 01 7: 00 01 8: 00 00 9: 00 00
0: 05 05 1: 05 05 2: 00 05 3: 05 03 4: 00 01 5: 00 00 6: 00 01 7: 00 00 8: 00 03 9: 00 00
0: 05 05 1: 05 05 2: 00 05 3: 03 04 4: 00 01 5: 00 00 6: 00 01 7: 00 00 8: 00 02 9: 00 00
0: 05 02 1: 05 02 2: 00 05 3: 00 05 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 02 9: 00 01
0: 05 00 1: 05 03 2: 00 05 3: 00 05 4: 00 01 5: 00 01 6: 00 00 7: 00 00 8: 00 01 9: 00 00
0: 05 02 1: 05 03 2: 00 05 3: 00 05 4: 00 01 5: 00 01 6: 00 00 7: 00 00 8: 00 02 9: 00 01
0: 05 04 1: 05 04 2: 00 02 3: 00 05 4: 00 01 5: 00 02 6: 00 00 7: 00 01 8: 00 00 9: 00 01
0: 05 05 1: 05 03 2: 00 03 3: 00 02 4: 00 02 5: 00 02 6: 00 00 7: 00 00 8: 00 01 9: 00 01
0: 05 05 1: 05 03 2: 00 02 3: 00 02 4: 00 02 5: 00 01 6: 00 00 7: 00 01 8: 00 02 9: 00 01
0: 05 05 1: 05 05 2: 00 01 3: 00 05 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 02 9: 00 00
0: 05 05 1: 05 04 2: 00 03 3: 00 05 4: 00 01 5: 00 00 6: 00 01 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 03 2: 00 05 3: 00 03 4: 00 00 5: 00 01 6: 00 00 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 03 2: 00 05 3: 00 02 4: 00 00 5: 00 02 6: 00 01 7: 00 00 8: 00 02 9: 00 01
0: 05 05 1: 05 01 2: 00 03 3: 00 05 4: 00 00 5: 00 03 6: 00 02 7: 00 00 8: 00 00 9: 00 01
0: 05 05 1: 05 02 2: 00 05 3: 00 04 4: 00 00 5: 00 03 6: 00 00 7: 00 00 8: 00 00 9: 00 01
0: 05 05 1: 05 00 2: 00 05 3: 00 05 4: 00 01 5: 00 02 6: 00 00 7: 00 00 8: 00 00 9: 00 01
0: 05 05 1: 05 02 2: 00 05 3: 00 01 4: 00 03 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 00
0: 05 05 1: 05 04 2: 00 05 3: 00 00 4: 00 01 5: 00 00 6: 00 00 7: 00 03 8: 00 00 9: 00 00
0: 04 05 1: 05 04 2: 00 05 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 02 8: 00 00 9: 00 01
0: 01 05 1: 05 01 2: 00 04 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
0: 00 05 1: 05 00 2: 00 01 3: 00 00 4: 00 00 5: 00 02 6: 00 01 7: 00 01 8: 00 00 9: 00 00
0: 00 05 1: 05 00 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 01 8: 00 00 9: 00 02
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00
```

As expected the last OSD to finish is the busiest OSD which also is the one with the higher weight. In the previous case it was the last one (osd.9) in this case it is the first one (osd.0). In both cases the busiest (last to finish) osd was capped by the max_backfill parameter. At no point did it not obtain a backfill peer because it raced against another OSD.

#21 - 12/10/2014 02:14 PM - Loic Dachary

The cluster was populated initially with

```
./ceph tell osd.* injectargs --osd_max_pg_log_entries=100 --osd_min_pg_log_entries=25
./ceph osd pool create mypool 1024 1024
./rados -p mypool bench 300 write -b 1024 --run-name backfill2 --no-cleanup
```

#22 - 12/10/2014 11:32 PM - Loic Dachary

When backfilling there are two possible situations that could be optimized with priorities:

- The longest(s) running OSDs are starving at some point during backfilling
- At the end of backfilling less than max_backfilling OSDs are involved during a significant amount of time and they could have arranged for their workload to happen in parallel earlier instead of sequentially at the end

#23 - 12/11/2014 02:47 AM - Loic Dachary

With max_backfill = 3

```
while read osd weight ; do
      ./ceph osd crush reweight osd.$osd $weight
done <<EOF
0 1
1 1
2 1
3 1
4 1
5 1
6 1
7 1
8 1
9 1
10 1
11 1
12 1
13 1
14 5
15 5
16 5
17 10
18 10
19 10
EOF
```

We get

```
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 01 00 10: 03 02 11: 03 03
12: 03 03 13: 03 02 14: 00 03 15: 00 00 16: 03 00 17: 03 00 18: 03 00 19: 02 00
```

0: 03 02 1: 03 02 2: 03 02 3: 03 01 4: 03 02 5: 03 01 6: 03 02 7: 03 03 8: 03 03 9: 03 01 10: 03 03 11: 03 00
12: 03 02 13: 03 03 14: 03 03 15: 03 03 16: 03 02 17: 03 00 18: 03 00 19: 03 00
0: 03 03 1: 03 01 2: 03 02 3: 03 02 4: 03 01 5: 03 00 6: 03 03 7: 03 03 8: 03 03 9: 03 02 10: 03 03 11: 03 00
12: 03 02 13: 03 02 14: 03 03 15: 03 03 16: 03 02 17: 03 00 18: 03 01 19: 03 01
0: 03 03 1: 03 02 2: 03 01 3: 03 03 4: 03 00 5: 03 01 6: 03 03 7: 03 03 8: 03 03 9: 03 03 10: 03 03 11: 03 00
12: 03 02 13: 03 02 14: 03 03 15: 03 02 16: 03 03 17: 03 01 18: 03 00 19: 03 00
0: 03 02 1: 03 02 2: 03 01 3: 03 03 4: 03 00 5: 03 02 6: 03 02 7: 03 03 8: 03 03 9: 03 01 10: 03 03 11: 03 00
12: 03 02 13: 03 03 14: 03 03 15: 03 02 16: 03 03 17: 03 02 18: 03 02 19: 03 03
0: 03 02 1: 03 02 2: 03 02 3: 03 03 4: 03 00 5: 03 01 6: 03 03 7: 03 03 8: 03 02 9: 03 02 10: 03 03 11: 03 00
12: 03 01 13: 03 03 14: 03 03 15: 03 02 16: 03 01 17: 03 03 18: 03 02 19: 03 01
0: 03 01 1: 03 01 2: 03 02 3: 03 03 4: 03 01 5: 03 01 6: 03 02 7: 03 03 8: 03 01 9: 03 01 10: 03 03 11: 03 00
12: 03 01 13: 03 01 14: 03 03 15: 03 01 16: 03 00 17: 03 03 18: 03 00 19: 03 01
0: 03 02 1: 03 01 2: 03 02 3: 03 03 4: 03 00 5: 03 03 6: 03 01 7: 03 03 8: 03 01 9: 03 01 10: 03 03 11: 03 00
12: 03 01 13: 03 01 14: 03 03 15: 03 03 16: 03 03 17: 03 02 18: 03 00 19: 03 00
0: 03 01 1: 03 00 2: 03 03 3: 03 03 4: 03 00 5: 03 02 6: 03 02 7: 03 01 8: 03 01 9: 03 01 10: 03 03 11: 03 01
12: 03 01 13: 03 01 14: 03 03 15: 03 03 16: 03 03 17: 03 01 18: 03 01 19: 03 00
0: 03 03 1: 03 02 2: 03 01 3: 03 01 4: 03 00 5: 03 01 6: 03 00 7: 03 01 8: 03 03 9: 03 02 10: 03 03 11: 03 02
12: 03 00 13: 03 01 14: 03 03 15: 03 03 16: 03 03 17: 03 02 18: 03 02 19: 03 03
0: 03 01 1: 03 02 2: 03 01 3: 03 03 4: 03 00 5: 03 01 6: 03 03 7: 03 02 8: 03 03 9: 03 00 10: 03 03 11: 03 00
12: 03 00 13: 03 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 00 19: 03 00
0: 03 02 1: 03 03 2: 03 00 3: 03 01 4: 03 00 5: 03 02 6: 03 02 7: 03 01 8: 03 01 9: 03 01 10: 03 03 11: 03 01
12: 03 01 13: 03 00 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 02
0: 03 01 1: 03 02 2: 03 00 3: 03 01 4: 03 00 5: 03 03 6: 03 01 7: 03 02 8: 03 01 9: 03 01 10: 03 03 11: 03 00
12: 03 02 13: 03 01 14: 03 03 15: 03 03 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 03 02 1: 03 01 2: 03 02 3: 03 01 4: 02 01 5: 03 01 6: 03 01 7: 03 01 8: 03 01 9: 03 01 10: 03 03 11: 03 01
12: 03 02 13: 03 00 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 00
0: 03 03 1: 03 01 2: 03 01 3: 03 00 4: 00 01 5: 03 00 6: 03 03 7: 03 01 8: 03 01 9: 03 02 10: 03 03 11: 03 00
12: 03 01 13: 03 02 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 03 02 1: 03 01 2: 03 02 3: 03 00 4: 00 02 5: 03 00 6: 03 02 7: 03 02 8: 03 00 9: 03 00 10: 03 03 11: 03 01
12: 03 01 13: 03 02 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 01
0: 03 02 1: 03 00 2: 03 02 3: 03 00 4: 00 01 5: 03 00 6: 03 02 7: 03 01 8: 03 01 9: 03 00 10: 03 03 11: 03 01
12: 03 01 13: 01 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 03 03 1: 03 00 2: 03 02 3: 03 00 4: 00 00 5: 03 00 6: 03 02 7: 03 00 8: 03 01 9: 03 00 10: 02 03 11: 03 00
12: 03 00 13: 01 00 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 01 19: 03 02
0: 03 01 1: 03 01 2: 03 02 3: 03 00 4: 00 00 5: 03 00 6: 03 00 7: 03 00 8: 03 02 9: 03 00 10: 00 03 11: 03 01
12: 03 01 13: 01 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 03 02 1: 03 00 2: 03 02 3: 03 00 4: 00 01 5: 03 02 6: 03 00 7: 03 01 8: 03 01 9: 03 01 10: 00 01 11: 03 01
12: 03 01 13: 01 01 14: 03 03 15: 03 03 16: 03 02 17: 03 03 18: 03 03 19: 03 03
0: 03 03 1: 03 00 2: 03 01 3: 03 00 4: 00 00 5: 03 00 6: 03 00 7: 03 02 8: 03 00 9: 00 01 10: 00 03 11: 03 01
12: 02 00 13: 01 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 02 19: 03 02
0: 03 02 1: 03 00 2: 03 02 3: 03 01 4: 00 01 5: 03 00 6: 03 01 7: 03 03 8: 02 01 9: 00 01 10: 00 02 11: 03 00
12: 01 00 13: 01 01 14: 03 03 15: 03 03 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 03 03 1: 03 00 2: 03 01 3: 03 02 4: 00 01 5: 03 00 6: 03 00 7: 03 01 8: 02 01 9: 00 02 10: 00 03 11: 03 01
12: 01 00 13: 01 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 03 02 1: 03 01 2: 03 01 3: 03 01 4: 00 02 5: 03 00 6: 03 01 7: 03 02 8: 02 00 9: 00 00 10: 00 03 11: 03 01
12: 00 00 13: 00 00 14: 03 03 15: 03 01 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 03 02 1: 03 01 2: 03 01 3: 03 03 4: 00 03 5: 00 01 6: 03 00 7: 03 01 8: 00 01 9: 00 00 10: 00 03 11: 03 01
12: 00 00 13: 00 01 14: 03 03 15: 03 03 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 01 02 1: 03 01 2: 03 02 3: 03 03 4: 00 01 5: 00 00 6: 03 00 7: 03 02 8: 00 01 9: 00 01 10: 00 03 11: 01 03
12: 00 00 13: 00 00 14: 03 03 15: 03 02 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 03 02 2: 03 01 3: 03 01 4: 00 02 5: 00 01 6: 03 00 7: 02 00 8: 00 01 9: 00 01 10: 00 03 11: 01 03
12: 00 02 13: 00 00 14: 03 03 15: 03 03 16: 03 02 17: 03 03 18: 03 03 19: 03 02
0: 00 01 1: 03 02 2: 02 01 3: 03 01 4: 00 01 5: 00 01 6: 03 01 7: 01 01 8: 00 00 9: 00 03 10: 00 03 11: 00 02
12: 00 00 13: 00 00 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 02 19: 03 02
0: 00 01 1: 03 02 2: 00 01 3: 03 00 4: 00 01 5: 00 01 6: 02 00 7: 01 01 8: 00 01 9: 00 01 10: 00 02 11: 00 01
12: 00 02 13: 00 01 14: 03 03 15: 03 01 16: 03 03 17: 03 03 18: 03 01 19: 03 02
0: 00 03 1: 03 01 2: 00 01 3: 03 01 4: 00 02 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 01 10: 00 02 11: 00 00
12: 00 00 13: 00 02 14: 03 03 15: 03 01 16: 03 03 17: 03 03 18: 03 03 19: 03 03
0: 00 01 1: 01 02 2: 00 00 3: 03 00 4: 00 01 5: 00 02 6: 00 02 7: 00 00 8: 00 01 9: 00 02 10: 00 02 11: 00 00
12: 00 00 13: 00 01 14: 03 03 15: 03 02 16: 03 03 17: 03 03 18: 03 02 19: 03 03
0: 00 01 1: 00 01 2: 00 02 3: 03 00 4: 00 01 5: 00 02 6: 00 01 7: 00 02 8: 00 03 9: 00 02 10: 00 02 11: 00 01
12: 00 01 13: 00 01 14: 03 03 15: 03 02 16: 03 03 17: 03 03 18: 03 03 19: 03 01
0: 00 03 1: 00 00 2: 00 02 3: 00 01 4: 00 01 5: 00 02 6: 00 01 7: 00 01 8: 00 01 9: 00 02 10: 00 02 11: 00 01
12: 00 00 13: 00 00 14: 03 02 15: 03 02 16: 03 03 17: 03 01 18: 03 01 19: 03 01
0: 00 01 1: 00 00 2: 00 01 3: 00 02 4: 00 02 5: 00 01 6: 00 00 7: 00 02 8: 00 01 9: 00 02 10: 00 01 11: 00 02
12: 00 00 13: 00 00 14: 03 03 15: 03 03 16: 03 03 17: 03 00 18: 03 00 19: 03 03
0: 00 02 1: 00 01 2: 00 01 3: 00 01 4: 00 02 5: 00 00 6: 00 00 7: 00 02 8: 00 00 9: 00 02 10: 00 01 11: 00 00
12: 00 02 13: 00 01 14: 03 02 15: 03 03 16: 03 03 17: 03 01 18: 03 03 19: 03 00
0: 00 01 1: 00 02 2: 00 02 3: 00 00 4: 00 02 5: 00 00 6: 00 01 7: 00 01 8: 00 00 9: 00 01 10: 00 01 11: 00 00
12: 00 02 13: 00 00 14: 03 02 15: 03 03 16: 03 03 17: 03 01 18: 03 03 19: 03 01
0: 00 02 1: 00 00 2: 00 00 3: 00 00 4: 00 02 5: 00 02 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 01 11: 00 02
12: 00 00 13: 00 00 14: 03 03 15: 03 02 16: 03 03 17: 03 03 18: 03 01 19: 03 00
0: 00 01 1: 00 01 2: 00 00 3: 00 00 4: 00 01 5: 00 00 6: 00 01 7: 00 00 8: 00 01 9: 00 01 10: 00 00 11: 00 00
12: 00 02 13: 00 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 01 19: 03 02
0: 00 00 1: 00 01 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 02 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00

12: 00 00 13: 00 03 14: 03 03 15: 03 01 16: 03 03 17: 03 03 18: 03 00 19: 03 01
0: 00 00 1: 00 00 2: 00 01 3: 00 01 4: 00 00 5: 00 00 6: 00 02 7: 00 00 8: 00 00 9: 00 00 10: 00 02 11: 00 00
12: 00 01 13: 00 00 14: 03 03 15: 03 02 16: 03 03 17: 03 03 18: 03 00 19: 03 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 03 9: 00 02 10: 00 00 11: 00 02
12: 00 01 13: 00 01 14: 03 03 15: 03 02 16: 03 01 17: 03 03 18: 03 01 19: 03 00
0: 00 01 1: 00 00 2: 00 00 3: 00 01 4: 00 00 5: 00 03 6: 00 00 7: 00 01 8: 00 01 9: 00 02 10: 00 01 11: 00 00
12: 00 00 13: 00 00 14: 03 03 15: 03 03 16: 03 02 17: 03 03 18: 03 01 19: 03 01
0: 00 01 1: 00 01 2: 00 00 3: 00 01 4: 00 00 5: 00 01 6: 00 00 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 01
12: 00 00 13: 00 01 14: 03 03 15: 03 03 16: 03 03 17: 03 03 18: 03 01 19: 03 02
0: 00 01 1: 00 02 2: 00 01 3: 00 00 4: 00 00 5: 00 01 6: 00 01 7: 00 01 8: 00 00 9: 00 03 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 03 02 15: 03 03 16: 03 03 17: 03 03 18: 03 00 19: 03 02
0: 00 00 1: 00 00 2: 00 02 3: 00 01 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 01
12: 00 02 13: 00 00 14: 03 00 15: 03 03 16: 03 01 17: 03 03 18: 03 02 19: 03 03
0: 00 00 1: 00 00 2: 00 01 3: 00 02 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 02 9: 00 02 10: 00 00 11: 00 01
12: 00 01 13: 00 00 14: 03 02 15: 03 03 16: 03 01 17: 03 03 18: 03 01 19: 03 02
0: 00 01 1: 00 00 2: 00 01 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 01 8: 00 01 9: 00 00 10: 00 00 11: 00 01
12: 00 01 13: 00 00 14: 02 01 15: 03 03 16: 03 01 17: 03 03 18: 03 02 19: 03 02
0: 00 01 1: 00 00 2: 00 01 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 01 8: 00 02 9: 00 00 10: 00 01 11: 00 01
12: 00 01 13: 00 00 14: 00 01 15: 03 03 16: 03 01 17: 03 03 18: 03 02 19: 03 01
0: 00 01 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 01 8: 00 02 9: 00 00 10: 00 01 11: 00 00
12: 00 01 13: 00 01 14: 00 02 15: 03 03 16: 03 01 17: 03 03 18: 03 03 19: 03 02
0: 00 00 1: 00 01 2: 00 01 3: 00 00 4: 00 01 5: 00 00 6: 00 01 7: 00 00 8: 00 01 9: 00 00 10: 00 01 11: 00 01
12: 00 01 13: 00 02 14: 00 02 15: 03 01 16: 03 01 17: 03 03 18: 03 03 19: 03 01
0: 00 01 1: 00 03 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 01 13: 00 01 14: 00 02 15: 03 02 16: 03 02 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 02 7: 00 01 8: 00 00 9: 00 01 10: 00 01 11: 00 00
12: 00 00 13: 00 01 14: 00 01 15: 03 01 16: 03 03 17: 03 03 18: 03 03 19: 03 01
0: 00 00 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 01 7: 00 01 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 03 14: 00 02 15: 03 01 16: 03 02 17: 03 03 18: 03 03 19: 03 01
0: 00 02 1: 00 00 2: 00 00 3: 00 01 4: 00 03 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 03 10: 00 00 11: 00 02
12: 00 00 13: 00 01 14: 00 01 15: 03 02 16: 03 01 17: 03 03 18: 03 03 19: 03 02
0: 00 01 1: 00 02 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 03 10: 00 01 11: 00 00
12: 00 01 13: 00 00 14: 00 01 15: 03 03 16: 03 01 17: 03 01 18: 03 03 19: 03 01
0: 00 00 1: 00 00 2: 00 00 3: 00 01 4: 00 00 5: 00 01 6: 00 02 7: 00 00 8: 00 01 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 01 14: 00 03 15: 03 03 16: 03 01 17: 03 01 18: 03 03 19: 03 00
0: 00 00 1: 00 00 2: 00 01 3: 00 01 4: 00 00 5: 00 00 6: 00 01 7: 00 01 8: 00 00 9: 00 00 10: 00 00 11: 00 02
12: 00 00 13: 00 00 14: 00 00 15: 03 03 16: 03 02 17: 03 01 18: 03 03 19: 03 01
0: 00 00 1: 00 02 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 02 9: 00 00 10: 00 01 11: 00 01
12: 00 00 13: 00 01 14: 00 02 15: 03 03 16: 03 01 17: 03 01 18: 03 03 19: 03 03
0: 00 01 1: 00 02 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 01 8: 00 01 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 02 14: 00 03 15: 03 03 16: 03 00 17: 03 03 18: 03 03 19: 03 02
0: 00 01 1: 00 01 2: 00 02 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 01 8: 00 00 9: 00 01 10: 00 01 11: 00 00
12: 00 00 13: 00 00 14: 00 02 15: 03 03 16: 03 01 17: 03 02 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 02 5: 00 01 6: 00 01 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 01 13: 00 00 14: 00 03 15: 03 03 16: 03 02 17: 03 03 18: 03 03 19: 03 01
0: 00 01 1: 00 00 2: 00 01 3: 00 02 4: 00 00 5: 00 01 6: 00 01 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 01
12: 00 00 13: 00 00 14: 00 03 15: 03 01 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 00 02 1: 00 00 2: 00 00 3: 00 01 4: 00 01 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 01
12: 00 00 13: 00 01 14: 00 03 15: 03 01 16: 03 00 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 01 2: 00 01 3: 00 01 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 00
12: 00 00 13: 00 01 14: 00 03 15: 03 03 16: 03 02 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 01
12: 00 00 13: 00 01 14: 00 02 15: 03 03 16: 03 00 17: 03 03 18: 03 02 19: 03 03
0: 00 01 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 00 8: 00 01 9: 00 01 10: 00 01 11: 00 01
12: 00 00 13: 00 00 14: 00 01 15: 03 01 16: 03 00 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 02 2: 00 00 3: 00 00 4: 00 00 5: 00 02 6: 00 02 7: 00 00 8: 00 00 9: 00 01 10: 00 01 11: 00 02
12: 00 00 13: 00 00 14: 00 02 15: 03 03 16: 03 01 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 01
12: 00 00 13: 00 00 14: 00 03 15: 03 01 16: 03 02 17: 03 03 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 01 00 16: 00 01 17: 03 03 18: 03 03 19: 03 03
0: 00 01 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 01 8: 00 00 9: 00 01 10: 00 01 11: 00 00
12: 00 01 13: 00 00 14: 00 00 15: 00 01 16: 00 01 17: 03 02 18: 03 03 19: 03 02
0: 00 00 1: 00 01 2: 00 01 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 01
12: 00 00 13: 00 00 14: 00 00 15: 00 02 16: 00 01 17: 03 01 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 01 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 01 10: 00 01 11: 00 00
12: 00 00 13: 00 02 14: 00 00 15: 00 00 16: 00 03 17: 03 00 18: 03 03 19: 03 02
0: 00 00 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 01 8: 00 00 9: 00 01 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 01 15: 00 01 16: 00 00 17: 03 01 18: 03 03 19: 03 03
0: 00 00 1: 00 00 2: 00 01 3: 00 00 4: 00 01 5: 00 00 6: 00 00 7: 00 01 8: 00 00 9: 00 00 10: 00 01 11: 00 00
12: 00 00 13: 00 00 14: 00 01 15: 00 00 16: 00 00 17: 03 01 18: 03 01 19: 00 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 01 7: 00 01 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 01 16: 00 02 17: 03 00 18: 03 01 19: 00 03
0: 00 00 1: 00 00 2: 00 00 3: 00 01 4: 00 00 5: 00 00 6: 00 01 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 00
12: 00 01 13: 00 00 14: 00 01 15: 00 00 16: 00 00 17: 03 02 18: 03 00 19: 00 03

```
0: 00 01 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 03 15: 00 00 16: 00 00 17: 03 03 18: 03 00 19: 00 03
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 01 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 03 15: 00 01 16: 00 01 17: 03 00 18: 03 00 19: 00 03
0: 00 00 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 02 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 01 15: 00 00 16: 00 01 17: 03 01 18: 03 01 19: 00 03
0: 00 01 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 01 6: 00 00 7: 00 00 8: 00 00 9: 00 01 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 01 15: 00 02 16: 00 00 17: 02 00 18: 03 00 19: 00 03
0: 00 00 1: 00 01 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
0: 00 00 1: 00 00 2: 00 00 3: 00 00 4: 00 00 5: 00 00 6: 00 00 7: 00 00 8: 00 00 9: 00 00 10: 00 00 11: 00 00
12: 00 00 13: 00 00 14: 00 00 15: 00 00 16: 00 00 17: 00 00 18: 00 00 19: 00 00
```

Where one of the three busiest OSDs has nothing left to do toward the end. The second column for OSD 19: is zero meaning there are no local in progress from this OSD to other OSDs while OSD 18: and OSD 17: have maxbackfill local in progress.

```
17: 03 02 18: 03 03 19: 03 02
17: 03 01 18: 03 03 19: 03 03
17: 03 00 18: 03 03 19: 03 02
17: 03 01 18: 03 03 19: 03 03
17: 03 01 18: 03 01 19: 00 03
17: 03 00 18: 03 01 19: 00 03
17: 03 02 18: 03 00 19: 00 03
17: 03 03 18: 03 00 19: 00 03
17: 03 00 18: 03 00 19: 00 03
17: 03 01 18: 03 01 19: 00 03
17: 02 00 18: 03 00 19: 00 03
17: 00 00 18: 00 00 19: 00 00
```

If backfilling involving OSD 17: or OSD 18: and OSD: 19 could have been done earlier, the tail could be shortened.

#24 - 12/11/2014 04:05 AM - Loic Dachary

The results above are the same with or without the priority patch (below is the tail wit the patch).

```
17: 03 03 18: 03 03 19: 03 02
17: 03 03 18: 03 03 19: 03 03
17: 03 03 18: 03 03 19: 03 03
17: 03 03 18: 03 03 19: 03 03
```

```
17: 03 02 18: 03 03 19: 03 03
17: 03 00 18: 03 03 19: 02 03
17: 03 01 18: 03 01 19: 00 03
17: 03 01 18: 03 00 19: 00 03
17: 03 01 18: 03 01 19: 00 03
17: 03 00 18: 03 00 19: 00 03
17: 03 01 18: 03 00 19: 00 03
17: 03 00 18: 03 01 19: 00 03
17: 03 00 18: 03 00 19: 00 03
17: 03 00 18: 03 01 19: 00 03
17: 03 00 18: 00 01 19: 00 03
17: 00 00 18: 00 00 19: 00 00
```

#25 - 12/12/2014 02:13 AM - Loic Dachary

```
                          FROM

      +-----+-----+-----+-----+-----+-----+-----+
      |     |osd.0|osd.1|osd.2|osd.3|osd.4|osd.5|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.0|     | 10  |  6  |  0  | 10  | 200 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.1|  5  |     |  0  | 40  | 20  |  23 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.2|  2  |  9  |     | 400 | 26  |  13 |
 TO   +-----+-----+-----+-----+-----+-----+-----+
      |osd.3|  8  | 700 | 65  |     |  0  |  7  |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.4| 88  |  10 | 84  | 7   |     |  49 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.5| 94  |  3  | 22  | 3   |  9  |     |
      +-----+-----+-----+-----+-----+-----+-----+
```

A table representing the number of PGs that are moved from the OSD in a given column to the OSD in the corresponding row. At each step at most N PGs are sent from a given OSD and at most N PGs are received by a given OSD (osd_max_backfill). What algorithm minimizes the number of steps required to move all PGs to their destination OSD ?

A) In each column select the cell with the most PGs to transfert
and mark them with a F.

```
                          FROM

      +-----+-----+-----+-----+-----+-----+-----+
      |     |osd.0|osd.1|osd.2|osd.3|osd.4|osd.5|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.0|     | 10  |  6  |  0  | 10  |F200 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.1|  5  |     |  0  | 40  | 20  |  23 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.2|  2  |  9  |     |F400 |F26  |  13 |
 TO   +-----+-----+-----+-----+-----+-----+-----+
      |osd.3|  8  |F700 | 65  |     |  0  |  7  |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.4| 88  |  10 |F84  | 7   |     |  49 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.5|F94  |  3  | 22  | 3   |  9  |     |
      +-----+-----+-----+-----+-----+-----+-----+
```

B) In each row, select the cell with a F that has the most PGs to transfert and mark it with a T (maybe more than one if N > the amount in the cell).

```
                                FROM

      +-----+-----+-----+-----+-----+-----+-----+
      |     |osd.0|osd.1|osd.2|osd.3|osd.4|osd.5|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.0|     | 10  |  6  |  0  | 10  |F200T|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.1|  5  |     |  0  | 40  | 20  |  23 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.2|  2  |  9  |     |F400T|F26  |  13 |
TO    +-----+-----+-----+-----+-----+-----+-----+
      |osd.3|  8  |F700T| 65  |     |  0  |  7  |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.4| 88  |  10 |F84T | 7   |     |  49 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.5|F94T |  3  | 22  | 3   |  9  |     |
      +-----+-----+-----+-----+-----+-----+-----+
```

C) In each column select the cell with the most PGs to transfert and that is not already marked with a F or a T and mark them with a F.

```
                                FROM

      +-----+-----+-----+-----+-----+-----+-----+
      |     |osd.0|osd.1|osd.2|osd.3|osd.4|osd.5|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.0|     | 10  |  6  |  0  | 10  |F200T|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.1|  5  |     |  0  | 40  |F20  |  23 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.2|  2  |  9  |     |F400T|F26  |  13 |
TO    +-----+-----+-----+-----+-----+-----+-----+
      |osd.3|  8  |F700T| 65  |     |  0  |  7  |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.4| 88  |  10 |F84T | 7   |     |  49 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.5|F94T |  3  | 22  | 3   |  9  |     |
      +-----+-----+-----+-----+-----+-----+-----+
```

D) Loop back to B unless C found no cell without a T and a F.

```
                                FROM

      +-----+-----+-----+-----+-----+-----+-----+
      |     |osd.0|osd.1|osd.2|osd.3|osd.4|osd.5|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.0|     | 10  |  6  |  0  | 10  |F200T|
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.1|  5  |     |  0  | 40  |F20T |  23 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.2|  2  |  9  |     |F400T|F26  |  13 |
TO    +-----+-----+-----+-----+-----+-----+-----+
      |osd.3|  8  |F700T| 65  |     |  0  |  7  |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.4| 88  |  10 |F84T | 7   |     |  49 |
      +-----+-----+-----+-----+-----+-----+-----+
      |osd.5|F94T |  3  | 22  | 3   |  9  |     |
      +-----+-----+-----+-----+-----+-----+-----+
```

#26 - 12/12/2014 06:35 AM - Loic Dachary

It is probably best to decide which PG should be granted the next slot in the {local,remote}_reserver whenever a slot becomes available. For local reservations it should be one of the PG going to the OSD for which there is the highest number of PGs waiting for a local reservation. For remote reservations it should be the same.

If the priority is calculated when adding PGs to the list of pending reservations (as in the proposed patch https://github.com/ceph/ceph/pull/2778), there is a good chance that the priority no longer reflects the reality when the PG gets a reservation.

If the next PG to get a reservation is calculated using an algorithm that requires knowledge of the entire system, it would require too much resources. In addition, the ideal case where a step consumes N == osd_max_backfill does not apply: the reality is that decisions must be made each time a slot becomes available. The problem is therefore less about minimizing the number of steps and more about taking the right decision for moving a single PGs when the system is in a given state.

The current algorithm selects a PG randomly in the list of candidate PGs to move from one OSD to the other. It is therefore possible for the longest list of PGs destined to a given OSD to be processed after all the others and create a long tail. If the list of PGs waiting for a reservation is sorted to move in front of the list the PGs destined to the busiest OSD, it could avoid this long tail without requiring knowledge of the entire system. Sorting a list of a few hundreds elements (the goal is for each OSD to now have more than a few hundred PGs at any given time) is probably cheap enough to be done each time a reservation slot becomes available.

#27 - 12/12/2014 10:37 AM - Loic Dachary

- Amend the current patch so that priorities are calculated depending on which osd the pg belong to, not the total regardless of the osd because it's not what we want.
- Cancel reservations if the pgmap changes because priorities needs to be recalculated and not revising them would create problems instead of solving problems.

#28 - 12/13/2014 01:46 AM - Loic Dachary

```
<gregsfortytwo> I'm concerned that the local state doesn't tell us much about the global state and so won't so
lve the problem
<gregsfortytwo> ie, in the general case don't we expect an OSD to have either 0 or 1 PGs outstanding to anothe
r OSD?
<loicd> gregsfortytwo: if the cluster is large then it's right. And in this case you won't really have any lon
g tail. Or do you see that possible ?
<gregsfortytwo> I'm not sure, I thought we normally do see this on large clusters though?
<loicd> I think long tail are observed when an OSD must move many PG to another OSD
<gregsfortytwo> my assumption is that it has to do with one host going down and then some of the OSDs on it ou
tcompeting the others, or something
<loicd> therefore when the local resevation queue grows
<loicd> that's also my understanding
<loicd> they compete because they have more PGs to move than what osd_maxbackfill allows them to have running
simultaneously, and they need to queue these PGs
<gregsfortytwo> right, so in that case we have 7 OSDs which start out with the same amount of work, but then 5
 of them get all of their work done before the other 2 are halfway done, right?
<loicd> you have 7 OSDs, 2 of them with significantly more work than the others and because you're not lucky t
he 2 most busiest osd start their work *after* the first 5 are done because they are competing
<loicd> if you have 7 osds with the same workload they will complete at the same time
<gregsfortytwo> oh, but for them to be competing at all they need to be targeting the same peers in enough qua
ntity to exhaust its recovery max
<gregsfortytwo> okay, so how do local queue counts resolve that?
<gregsfortytwo> NB that I'm not super-familiar with the recovery queuing system at present
```

<loicd> because the 2 osds with most work (i.e. more PGs to move) will have a longer queue and their PG will go first
<gregsfortytwo> but I'm under the impression that an OSD which wants to send data grabs a local slot of "backfills I'm providing" and then asks for a remote slot "backfills receiving", and either of those can block
<gregsfortytwo> loicd: I thought you just said the other 5 had finished before the 2 even started, and again for them to have a longer queue don't they need to all be submitting a whole lot to the same OSD?
<gregsfortytwo> I could be wrong but I don't think that's the way we usually get backed up
<loicd> at the moment, when a PG on the primary OSD finds that it must backfill to a replica, it adds itself to the local_reserver queue (which only has osd_maxbackfill slots), then it asks the replica to get a slot in the remote_reserver (which also has osd_maxbackfill slots)
<gregsfortytwo> I think it's just the long tail OSD loses a bunch of local races
<gregsfortytwo> of relatively small size
<loicd> > but I'm under the impression that an OSD which wants to send data grabs a local slot of "backfills I'm providing" and then asks for a remote slot "backfills receiving", and either of those can block : yes, that's also my understanding
<loicd> except it does not block, it is queued and the length of the queue tells us how much work there is
<gregsfortytwo> that last statement is where I'm losing you — the length of *local* queue, right?
<gregsfortytwo> that only tells us how much local work there is to do
<gregsfortytwo> right?
<loicd> yes
<gregsfortytwo> my contention is that the amount of local work will be largely consistent between the OSDs — they will mostly have 1 unit of work, or maybe 2
<loicd> I don't follow. If an osd has 100 PG to move to various other osds, the queue will be 100
<loicd> the local_reserver queue that is
<loicd> 100 - maxbackfill
<gregsfortytwo> how is it that a single OSD would have a large number to move elsewhere?
<gregsfortytwo> that normally shouldn't happen
<loicd> local is the case of backfill/recovery means "what flows from the primary to the replica"
<gregsfortytwo> but I might be talking myself out of my concerns for other reasons
<loicd> say an OSD contains 300 PGs
<loicd> and 50 of them go reshuffled elsewhere because the cluster grows / other osds go down
<loicd> is it not realistic ?
<gregsfortytwo> so 1/6 of its total PGs get removed?
<gregsfortytwo> for an expansion maybe, but I don't think that's typical for a "something died and came back" scenario
<loicd> yes, or it receives 50 more PGs for which it is primary
<loicd> which would be more typical of "something dies" and all other osds get more PG to care for
<gregsfortytwo> I guess if you have a 6-node cluster and something dies then the others all get about that ratio as new primary assignments, which isn't super uncommon for us
<loicd> yes
<gregsfortytwo> but that's certainly not what some of the larger installations see as likely, and I think they've reported a long tail problem as well (I'm thinking cern here)
<gregsfortytwo> let's think about this from the other direction
<gregsfortytwo> if we have 7 OSDs coming back at once, and they each have 7 recovery slots, then there are 49 "incoming" slots in total to receive (say) 700 PGs from N primaries
<gregsfortytwo> if N < 7, then the primaries can definitely do self-balancing like you're suggesting
<gregsfortytwo> if N >> 7, then they can't do self-balancing but since the number of sending slots outnumbers the receiving slots I think it shouldn't be a problem
<gregsfortytwo> ...is that right?
<loicd> yes
<gregsfortytwo> as long as they don't ask for more than one slot at a time on the receiving one then that's right, because they might all target osd 1 first, but it'll fill up and then they'll get acks from osd 2-7
<gregsfortytwo> ...but that won't be the case if they have 7 backfill slots and 14 recovering OSDs; they could all target the first 7
<gregsfortytwo> it's probably not too difficult to prevent that simultaneous targeting, but the strategy needs to be thought through and I suspect that's part of what's happening right now
<gregsfortytwo> (lots of people are running with max_backfills set to 1)
<loicd> oh really ?
<gregsfortytwo> yeah
<loicd> interesting factor !
<loicd> now that you mention it, it rings a bell
<gregsfortytwo> but I don't think running with 1 is a requirement for this to be a problem, I think it just needs to be smaller than some ratio between backfills needed, backfill suppliers, and backfill receivers
<loicd> what is your reasoning if what happens in that you have 4 1TB osd coming back and 1 4TB coming back ? How do you make sure the largest OSD always gets work so that you don't waste significant time because it is not as busy as it could at the beginning ?
<gregsfortytwo> right, if max_backfills is less than the number of backfill receivers a supplier must eventually target, then having the suppliers target the receiver set in the same order will cause problems
<gregsfortytwo> and the expected number of PGs to supply is going to be (num_pg*replication_count)*(num_receivers/num_osds)
<gregsfortytwo> loicd: I don't think we need to worry about the weighting, actually; since the 4TB OSD can't really accept it faster than the others
<gregsfortytwo> we just want to keep them all as busy as they can handle
<loicd> well, not always because there may be just one OSD changing so it's num_pg * (something in the range of 1 to size -1)

<gregsfortytwo> that would be num_receivers=1
<gregsfortytwo> so if you have 10k PGs, 100 OSDs, 3x replication, and you lose 10 OSDs and have them come back , you have to backfill 3k PGs, that matches my intuition
<gregsfortytwo> and those 3k PGs are coming from 90 OSDs, which gives them 33 each
<loicd> that's assuming all is perfectly balanced, in which case there won't be a long tail at all, right ?
<gregsfortytwo> so they have enough local information to prioritize, but they're going to start out with essen tially the same priorities for each recipient
<loicd> I don't see how a long tail could happen unless there is an inbalance on one OSD, at least
<loicd> (I agree with what you're describing, 100%)
<gregsfortytwo> well, say there's a perfect distribution and each of the 90 good OSDs is sending exactly 33 PG s along to the bad OSDs
<gregsfortytwo> how do the good OSDs pick an order to reserve backfill slots?
<loicd> at the moment, it's random
<gregsfortytwo> really?
<loicd> yes
<gregsfortytwo> that's better than what I was expecting the answer to be :)
<gregsfortytwo> but nonetheless, if max_backfill = 5, you have 450 outgoing slots mapping to 50 incoming slots
<loicd> each of the 33 try to get a slot in the local reserver, and when they can't they add themselves to the waiting list and will be dequeued in the order in which they came in
<gregsfortytwo> most of them are going to get denied
<gregsfortytwo> so it's important that we actually make use of each of the 50 incoming slots
<gregsfortytwo> I have no idea how a random selection from each of the 450 outgoing slots works out there
<gregsfortytwo> but I suspect it's not what we want, and that it's what causes the long tail in most cases
-*- gregsfortytwo tries to read some wikipedia
<gregsfortytwo> because each sender is choosing 5 from a set of 10, and we need at least 5 (of 90) senders to have requested access to each receiver at every slot change
<loicd> if a sender (local) gets a reservation, it will ask the receiver for a reservation (remote) and will b lock until it gets it, which may block the local reservation idle for a significant amount of time if the rece iver is heavily targeted by senders.
<gregsfortytwo> loicd: yeah, that is exactly my concern about doing prioritization based only on local data :)
<gregsfortytwo> each "bad" OSD should know pretty quickly how many PGs it needs to recover, so I'm wondering i f there's a good opportunity for them to send that along at semi-regular intervals
<gregsfortytwo> ...although if you just rely on those values then the senders will just get stuck waiting for slots on the ones that have fallen behind, so there won't be a long tail but recovery will be slower
<gregsfortytwo> as the recovering OSDs leapfrog each other back and forth
<gregsfortytwo> unless I'm just totally wrong about this balancing being an issue, it kind of seems like we ne ed some information transfer back and forth to help with scheduling, so that waiting senders can register as i nterested and get told when a slot is available which they can compete for :/
<gregsfortytwo> anyway, sorry if this was unhelpful loicd, but the emergent behavior of our recovery system is something that concerns me (in that it has emergent behavior, and we need to think about it) and I wanted to make sure it was being considered :)
<loicd> *very* helpful on the contrary
<gregsfortytwo> anyway, the upshot is that here's a reasonably typical scenario in which each provider is only sending ~3 PGs to each of the recipient OSDs, and I'm not sure that ordering based on numbers 1, 2, or 3 is g oing to have a huge benefit ;)
<loicd> :-)
<gregsfortytwo> awesome, thanks for humoring me :)
<loicd> things are pretty stable in between osd map changes. maybe something global could be calculated when t he osd receives a new map that's going to trigger movement. something the local osd could use to guestimate wh at others will do an plan accordingly. of course it may not be 100% accurate but could be a help as long as n o new osdmap comes in. and when a new map comes in all local reservation/priorities are trashed because the as sumptions are known to be wrong.
<loicd> gregsfortytwo: do you see a problem with that idea ?
<loicd> it could, for instance, be used to know that the receiver is going to be mostly idle or very busy
<gregsfortytwo> loicd: I'm not sure, I think for that to work we'd need each OSD to calculate a globally deter ministic order or mapping of some kind
<gregsfortytwo> which sounds like it'd be too expensive, since it's basically calculating crush locations for every pg on every map change
<loicd> you are correct
<gregsfortytwo> I don't remember how often backfill sends what kind of messages, but maybe recipient OSDs coul d do something like share what PGs they know need backfilling and who is providing them to everybody who wants to send them a backfill
<loicd> I thought it would be limited to the PG the OSD already knows but it cannot be, it needs to discover w hich PG it will participate in
<gregsfortytwo> and you could do something with that
<gregsfortytwo> since the notifies will flow in very quickly relative to the backfill work completing
<loicd> smart
<gregsfortytwo> if the recipients shared that information, then the providers could determine which percentage of data they are responsible for and prioritize PGs to try and balance that, maybe?
<loicd> yes
<gregsfortytwo> actually it wouldn't even need to include provider OSDs, just "I have [80|70|80 PGs still need ing backfill" (that being 3 OSDs communicating with the provider) and the provider goes "well, I'm providing [40|4|10] to each of them, respectively, so I will bias myself towards the first guy"
<gregsfortytwo> I'd want to sketch it out a little more to demonstrate that it works properly among roughly-ev en distributions though

```
<gregsfortytwo> actually I like this, I think you'd want to apply it on both ends though so the recipient also
 knows to keep a slot open for the guy providing the most data if necessary
<gregsfortytwo> sender orders recipients by percentage of PGs he has to send and requests as many backfill slo
ts as his percentage of the total data is (ie "I owe 10% of the data, I will ask for 10% of your slots" which
 will presumably usually be one) until he runs out, and the recipient does the same calculation and keeps slots
 open for anybody who gets at least one slot
<gregsfortytwo> ...nope, that kind of reservation won't work if there's a single OSD who runs out all of his s
lots before getting to somebody he owes a bunch of data too, drat
<gregsfortytwo> oh, but if the sender shares how many he has to send to people, they could each allocate the m
inimum of the percentage ratios
<gregsfortytwo> and expect to use that many
<gregsfortytwo> so the normal case will be that nobody gets reserved slots, but in our earlier example each of
 the senders would be choosing 5 OSDs ordered by which one they had the most commonality with, and since OSD Y
 having less commonality with osd X means that the remaining set of OSDs have an increased commonality, I thin
k that guarantees 5 senders to each recipient
<gregsfortytwo> loicd: again though, that relies on communicating with everybody you need to backfill with bas
ically every time you finish a backfill, which I think is feasible but I'm not sure how much of that is alread
y built in to our backfill patterns or no
<gregsfortytwo> *or not
-*- loicd reading
<loicd> unless I missed something I think backfilling is exclusively handled by each PG in isolation. the only
 thing they share, via the OSD, is the reservation queue
<gregsfortytwo> uh, maybe?
<loicd> they share nothing
<gregsfortytwo> I'm not sure how that's relevant here though
```

#29 - 12/13/2014 12:48 PM - Loic Dachary

- Status changed from Testing to Need Review

#30 - 12/15/2014 05:20 AM - Loic Dachary

- Status changed from Need Review to In Progress

#31 - 12/15/2014 09:29 AM - Loic Dachary

- Description updated

#32 - 12/16/2014 05:45 AM - Loic Dachary

Go simulation of the current algorithm http://play.golang.org/p/I-0Z4u-O3C

```
package main

import "fmt"
```

```
type OSD struct {
    ID     int
    Count  int
    Orders Orders
    Input  chan int
    Stats  Orders
}

type Orders map[int]int

const OSDMaxBackfill = 5

var osds = []*OSD{
    &OSD{ID: 0, Count: 1000, Orders: Orders{1: 5, 2: 2, 3: 8, 4: 88, 5: 94}},
    &OSD{ID: 1, Count: 1000, Orders: Orders{0: 10, 2: 9, 3: 700, 4: 10, 5: 3}},
    &OSD{ID: 2, Count: 1000, Orders: Orders{0: 6, 3: 65, 4: 84, 5: 22}},
    &OSD{ID: 3, Count: 1000, Orders: Orders{1: 40, 2: 400, 4: 7, 5: 3}},
    &OSD{ID: 4, Count: 1000, Orders: Orders{0: 10, 1: 20, 2: 26, 5: 9}},
    &OSD{ID: 5, Count: 1000, Orders: Orders{0: 200, 1: 23, 2: 13, 3: 7, 4: 49}},
}

func (o *OSD) Send() {
    out := make(chan int, OSDMaxBackfill)
    for t, c := range o.Orders {
        go func(to int, count int) {
            fmt.Printf("OSD:%d sending %d PGs to OSD:%d\n", o.ID, count, to)
            for i := 0; i < count; i++ {
                out <- to
            }
        }(t, c)
    }
    for to := range out {
        osds[to].Input <- o.ID
        o.Stats[to]++
        fmt.Printf("OSD:%d->OSD:%d remaining: %d\n", o.ID, to, o.Orders[to] - o.Stats[to])
    }

}

func (o *OSD) Recv() {
    for from := range o.Input {
        fmt.Printf("OSD:%d<-OSD:%d\n", o.ID, from)
    }
}

func main() {
    for _, o := range osds {
        o.Input = make(chan int, OSDMaxBackfill)
        o.Stats = make(map[int]int)
        go func(o *OSD) {
            fmt.Printf("OSD:%d running\n", o.ID)
            go o.Recv()
            o.Send()
        }(o)
    }
    select {}
}
```

#33 - 12/16/2014 05:49 AM - Loic Dachary

- Solving Constraint Integer Programs http://scip.zib.de/
- A Free and Open-Source Java Library for Constraint Programming http://choco-solver.org/

#34 - 12/22/2014 09:05 AM - Sage Weil

- Target version changed from v.actually91 to v0.92

#35 - 12/22/2014 09:20 AM - Loic Dachary

- Status changed from In Progress to Need More Info

- Target version deleted (v0.92)

Waiting on https://github.com/ceph/ceph/tree/wip-read-hole which will help communicate information between OSDs and allow them to decide on a sensible priority.

#36 - 12/22/2014 09:22 AM - Loic Dachary

- Target version set to v0.92

#37 - 01/06/2015 01:06 PM - Samuel Just

- Target version deleted (v0.92)

#38 - 03/10/2015 08:21 AM - Loic Dachary

- Backport deleted (giant,firefly)

#39 - 09/08/2015 11:38 AM - Loic Dachary

- Assignee deleted (Loic Dachary)

I'm not making progress, give an opportunity to someone else to move forward.