ceph days

How RGW Stores S3 Objects in Rados

Tobias Brunnwieser, Infrastructure Engineer, Hetzner Cloud GmbH

Why Bother?

- Understand and avoid bottlenecks

Why Bother?

- Understand and avoid bottlenecks
- Mitigate / resolve bugs

RGW Pools

RGW Pools

Dedicated pool per data category:

RGW Pools

Dedicated pool per data category:

- RGW makes heavy use of OMAPs, requires replicated pools

RGW Pools

Dedicated pool per data category:

- RGW makes heavy use of OMAPs, requires replicated pools
- Data better put in EC pools (overhead)

Metadata Pool: User Keys

`<zone>.rgw.meta:users.keys`

Metadata Pool: User Keys

`<zone>.rgw.meta:users.keys`

- RGW Multisite zone (“default”)

Metadata Pool: User Keys

`<zone>.rgw.meta:users.keys`

- RGW Multisite zone (“default”)
- Pool namespace: “users.keys”

Metadata Pool: User Keys

`<zone>.rgw.meta:users.keys`

- RGW Multisite zone ("default")
- Pool namespace: "users.keys"
  - Isolate groups of data within same pool

Metadata Pool: User Keys

`<zone>.rgw.meta:users.keys`

- RGW Multisite zone ("default")
- Pool namespace: "users.keys"
  - Isolate groups of data within same pool
  - We will cover some other namespaces shortly!

Metadata Pool: User Keys (Cont.)

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys ls
```

Metadata Pool: User Keys (Cont.)

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys ls

MD8H642GQ3H0OWL6IP6D
```

Metadata Pool: User Keys (Cont.)

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys ls

MD8H642GQ3H0OWL6IP6D
...

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys get MD8H642GQ3H0OWL6IP6D - | xxd
```

Metadata Pool: User Keys (Cont.)

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys ls

MD8H642GQ3H0OWL6IP6D
...

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys get MD8H642GQ3H0OWL6IP6D - | xxd

00000000: 0900 0000 7031 3130 3634 3632 31 \
    ....p11064621
```

Metadata Pool: User Keys (Cont.)

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys ls

MD8H642GQ3H0OWL6IP6D
...

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.keys get MD8H642GQ3H0OWL6IP6D - | xxd

00000000: 0900 0000 7031 3130 3634 3632 31 \
    ....p11064621
```

We get one object per key!

Metadata Pool: User Info

`<zone>.rgw.meta:users.uid`

Metadata Pool: User Info

<zone>.rgw.meta:users.uid

Metadata Pool: User Info

`<zone>.rgw.meta:users.uid`

```
# rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid ls | grep -F myuser

myuser
myuser.buckets
```

Metadata Pool: User Info

`<zone>.rgw.meta:users.uid`

```
# rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid ls | grep -F myuser

myuser
myuser.buckets
```

Two objects per user!

User Info

Object `myuser`

User Info

Object `myuser`

- Name

User Info

Object `myuser`

- Name
- Keys

User Info

Object `myuser`

- Name
- Keys
- ...

User Buckets List

Object `myuser.buckets`

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser
```

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser

nbg1-stage1-ceph3.rgw.meta/myuser.buckets \
    mtime 2025-10-24T16:17:35.000000+0000, size 0
```

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser

nbg1-stage1-ceph3.rgw.meta/myuser.buckets \
    mtime 2025-10-24T16:17:35.000000+0000, size 0

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid listomapkeys myuser.buckets
```

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser

nbg1-stage1-ceph3.rgw.meta/myuser.buckets \
    mtime 2025-10-24T16:17:35.000000+0000, size 0

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid listomapkeys myuser.buckets

mybucket
anotherbucket
```

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser

nbg1-stage1-ceph3.rgw.meta/myuser.buckets \
    mtime 2025-10-24T16:17:35.000000+0000, size 0

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid listomapkeys myuser.buckets

mybucket
anotherbucket
```

- Allows listing a user's buckets

User Buckets List

Object `myuser.buckets`

```
~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid stat myuser

nbg1-stage1-ceph3.rgw.meta/myuser.buckets \
    mtime 2025-10-24T16:17:35.000000+0000, size 0

~ # rados -p nbg1-stage1-ceph3.rgw.meta \
    -N users.uid listomapkeys myuser.buckets

mybucket
anotherbucket
```

- Allows listing a user's buckets
- Or verify that given user is the owner

Metadata Pool: Bucket Metadata

`<zone>.rgw.meta:root`

Metadata Pool: Bucket Metadata

`<zone>.rgw.meta:root`

Metadata Pool: Bucket Metadata

`<zone>.rgw.meta:root`

```
# rados -p nbg1-prod1-ceph3.rgw.meta -N root ls \
    | grep -F mybucket
```

Metadata Pool: Bucket Metadata

`<zone>.rgw.meta:root`

```
# rados -p nbg1-prod1-ceph3.rgw.meta -N root ls \
    | grep -F mybucket
```

```
mybucket
.bucket.meta.mybucket:75f3087b-[...]
```

Metadata Pool: Bucket Metadata

`<zone>.rgw.meta:root`

```
# rados -p nbg1-prod1-ceph3.rgw.meta -N root ls \
    | grep -F mybucket
```

```
mybucket
.bucket.meta.mybucket:75f3087b-[...]
```

Again, two ojects per bucket!

Bucket Metadata (Cont.)

- Object `mybucket`: bucket info

Bucket Metadata (Cont.)

- Object `mybucket`: bucket info
- Object `.bucket.meta.mybucket:[...]`: bucket *instance* info

Bucket Metadata (Cont.)

- Object `mybucket`: bucket info
- Object `.bucket.meta.mybucket:[...]`: bucket *instance* info

Why bucket instances?

- Allows re-creating bucket with same name

Bucket Metadata (Cont.)

- Object `mybucket`: bucket info
- Object `.bucket.meta.mybucket:[...]`: bucket *instance* info

Why bucket instances?

- Allows re-creating bucket with same name
- Changing bucket layout (e.g. shards)

Bucket Metadata (Cont.)

- Object `mybucket`: bucket info
- Object `.bucket.meta.mybucket:[...]`: bucket *instance* info

Why bucket instances?

- Allows re-creating bucket with same name
- Changing bucket layout (e.g. shards)

For our purposes: only instance is relevant

Bucket Instance Metadata

```
.bucket.meta.mybucket:75f3087b-c3c3-[...]
```

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

- Creation time

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

- Creation time
- Owner

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

- Creation time
- Owner
- Placement (storage class)

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

- Creation time
- Owner
- Placement (storage class)
- Layout

Bucket Instance Metadata

`.bucket.meta.mybucket:75f3087b-c3c3-[...]`

- Bucket name
- Bucket *marker*

Object Contents:

- Creation time
- Owner
- Placement (storage class)
- Layout
- ...

Bucket Instance Metadata (Cont.)

Object xattrs:

Bucket Instance Metadata (Cont.)

Object xattrs:

```
# rados -p nbg1-stage1-ceph3.rgw.meta -N root \
    listxattr .bucket.meta.mybucket:f2b16e62-fe1b-[...]
```

Bucket Instance Metadata (Cont.)

Object xattrs:

```
# rados -p nbg1-stage1-ceph3.rgw.meta -N root \
    listxattr .bucket.meta.mybucket:f2b16e62-fe1b-[...]

ceph.objclass.version # encoding version
```

Bucket Instance Metadata (Cont.)

Object xattrs:

```
# rados -p nbg1-stage1-ceph3.rgw.meta -N root \
    listxattr .bucket.meta.mybucket:f2b16e62-fe1b-[...]

ceph.objclass.version # encoding version
user.rgw.acl # bucket ACLs in JSON
```

Bucket Instance Metadata (Cont.)

Object xattrs:

```
# rados -p nbg1-stage1-ceph3.rgw.meta -N root \
    listxattr .bucket.meta.mybucket:f2b16e62-fe1b-[...]

ceph.objclass.version # encoding version
user.rgw.acl # bucket ACLs in JSON
user.rgw.iam-policy # IAM policies
```

Bucket Index Pool

`<zone>.rgw.buckets.index`

Bucket Index Pool

`<zone>.rgw.buckets.index`

- Contains a list of objects per bucket

Bucket Index Pool

`<zone>.rgw.buckets.index`

- Contains a list of objects per bucket
- **Must** be a replicated pool because of OMAPs, default 3-times

Bucket Index

Whats inside?

Bucket Index

Whats inside?

```
$ rados -p nbg1-stage1-ceph3.rgw.buckets.index ls
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.103
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.40951.12
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.23988.3
[...]
```

Bucket Index

Whats inside?

```
$ rados -p nbg1-stage1-ceph3.rgw.buckets.index ls
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.103
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.40951.12
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.23988.3
[...]

$ rados -p nbg1-stage1-ceph3.rgw.buckets.index stat \
    .dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.103
[...] mtime 2025-09-08T14:16:01.000000+0000, size 0
```

Bucket Index

Whats inside?

```
$ rados -p nbg1-stage1-ceph3.rgw.buckets.index ls
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.103
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.40951.12
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.11286592.23988.3
[...]

$ rados -p nbg1-stage1-ceph3.rgw.buckets.index stat \
    .dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.103
[...] mtime 2025-09-08T14:16:01.000000+0000, size 0
```

These are OMAPs!

Bucket Index (Cont.)

```
.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103
```

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*
  - `radosgw-admin bucket stats --bucket mybucket | jq .marker`

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*
  - `radosgw-admin bucket stats --bucket mybucket | jq .marker`
- Bucket instance generation

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*
  - `radosgw-admin bucket stats --bucket mybucket | jq .marker`
- Bucket instance generation
  - Only present if not zero (multisite setups)

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*
  - `radosgw-admin bucket stats --bucket mybucket | jq .marker`
- Bucket instance generation
  - Only present if not zero (multisite setups)
  - See `radosgw-admin bucket layout --bucket mybucket`

Bucket Index (Cont.)

`.dir.f2b16e62-fe1b-46be-8f31-b23d5d4226d4.10974126.21102.1.103`

- Static prefix
- Bucket marker
  - Identifier for bucket *instance*
  - `radosgw-admin bucket stats --bucket mybucket | jq .marker`
- Bucket instance generation
  - Only present if not zero (multisite setups)
  - See `radosgw-admin bucket layout --bucket mybucket`
- Shard number

Bucket Index Sharding

Why?

- Each bucket index rados object gets mapped to a different PG

Bucket Index Sharding

Why?

- Each bucket index rados object gets mapped to a different PG
- Therefore all index operations are spread over multiple OSDs (throughput!)

Bucket Index Sharding

Why?

- Each bucket index rados object gets mapped to a different PG
- Therefore all index operations are spread over multiple OSDs (throughput!)

Mapping:

- Object ID (OID) hash $\Rightarrow$ bucket index shard

Bucket Index Sharding

Why?

- Each bucket index rados object gets mapped to a different PG
- Therefore all index operations are spread over multiple OSDs (throughput!)

Mapping:

- Object ID (OID) hash $\Rightarrow$ bucket index shard
- Within bucket shard: sorted by name (OMAP)

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard

# shard 0
boxy # sorted!
revenue
strategic
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard

# shard 0
boxy
revenue
strategic
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard

# shard 0
boxy
revenue
strategic

# shard 1
buddhism
chimp
periscope
```

Bucket Index (Cont.)

```
marker=f2b16e62-fe1b-46be-8f31-b23d5d4226d4.[...]
rados -p nbg1-stage1-ceph3.rgw.buckets.index listomapkeys \
    .dir.$marker.$shard

# shard 0
boxy
revenue
strategic

# shard 1
buddhism
chimp
periscope

[...]
```

Data Pool

`<zone>.rgw.buckets.<storage_class>.data`

Data Pool

`<zone>.rgw.buckets.<storage_class>.data`

- Named after served zone of RGW multisite ("default" if no MS)

Data Pool

`<zone>.rgw.buckets.<storage_class>.data`

- Named after served zone of RGW multisite ("default" if no MS)
- One pool per S3 storage class (placements)

Data Pool

`<zone>.rgw.buckets.<storage_class>.data`

- Named after served zone of RGW multisite ("default" if no MS)
- One pool per S3 storage class (placements)
- Usually EC for optimal overhead

Data Pool

`<zone>.rgw.buckets.<storage_class>.data`

- Named after served zone of RGW multisite ("default" if no MS)
- One pool per S3 storage class (placements)
- Usually EC for optimal overhead

Example: `us-east1.rgw.buckets.standard.data`

Data Pool: Contents

For “14MiB.bin” in the bucket index, we got one *head* object:

Data Pool: Contents

For “14MiB.bin” in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

Data Pool: Contents

For “14MiB.bin” in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

Data Pool: Contents

For “14MiB.bin” in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

Data Pool: Contents

For “14MiB.bin” in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

- First of possibly many RADOS objects

Data Pool: Contents

For "14MiB.bin" in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

- First of possibly many RADOS objects
- Size <= `rgw_obj_stripe_size` (default 4MiB)

Data Pool: Contents

For "14MiB.bin" in the bucket index, we got one *head* object:

`f2b16e62-fe1b-[...]_14MiB.bin`

- First of possibly many RADOS objects
- Size <= `rgw_obj_stripe_size` (default 4MiB)
- Parallel read / write to same S3 object → more throughput

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
```

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
user.rgw.acl
```

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
user.rgw.acl
user.rgw.content_type
```

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
user.rgw.acl
user.rgw.content_type
user.rgw.etag
user.rgw.idtag
...
user.rgw.x-amz-content-sha256
user.rgw.x-amz-date
```

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
user.rgw.acl
user.rgw.content_type
user.rgw.etag
user.rgw.idtag
...
user.rgw.x-amz-content-sha256
user.rgw.x-amz-date
user.rgw.manifest
```

Head Object Extended Attributes

```
# rados \
    -p nbg1-stage1-ceph3.rgw.buckets.standard.data \
    listxattr f2b16e62-[...]_14MiB.bin
user.rgw.acl
user.rgw.content_type
user.rgw.etag
user.rgw.idtag
...
user.rgw.x-amz-content-sha256
user.rgw.x-amz-date
user.rgw.manifest
```

Manifest specifies *layout* of S3 object!

Object Manifest

```
/* radosgw-admin object stat --bucket mybucket
   --object 14MiB.bin */
{
```

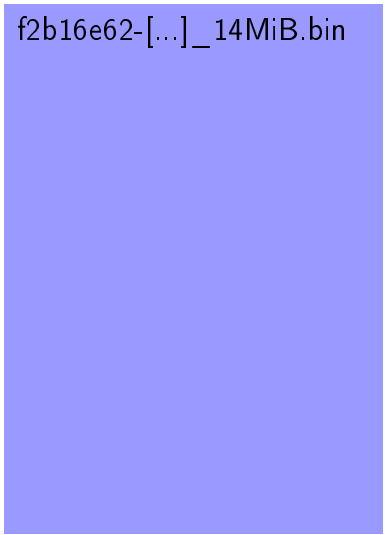
Object Manifest

```
/* radosgw-admin object stat --bucket mybucket
   --object 14MiB.bin */
{
    "obj_size": 14680064, /* S3 object bytes */
```

Object Manifest

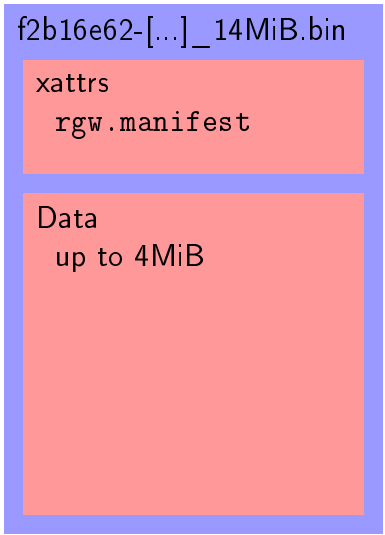
```
/* radosgw-admin object stat --bucket mybucket
   --object 14MiB.bin */
{
    "obj_size": 14680064, /* S3 object bytes */
    "head_size": 0, /* head object bytes */
    "...": ""
```

Head Object Contents

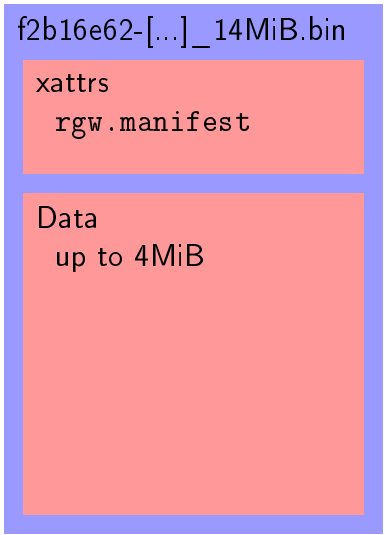


Head Object Contents

Head Object Contents

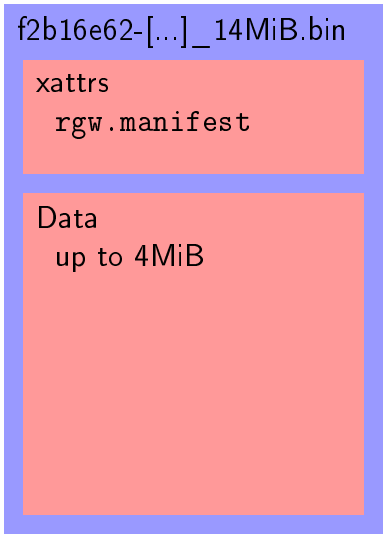


Head Object Contents



- S3 size <= stripe size: only head object required!

Head Object Contents



- S3 size <= stripe size: only head object required!
- In this example: unused (multipart upload)

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
         "start_ofs": 0, /* from S3 object */
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
         "start_ofs": 0, /* from S3 object */
         "part_size": 5242880,
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
         "start_ofs": 0, /* from S3 object */
         "part_size": 5242880,
         "stripe_max_size": 4194304, /* split */
      },
```

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
         "start_ofs": 0, /* from S3 object */
         "part_size": 5242880,
         "stripe_max_size": 4194304, /* split */
      },
```

- *Trivial* parts: single item only

Object Manifest (Cont.)

```
"...": "",
"prefix": "14MiB.bin.2~sG1iwZwlI[...]",
"rules": [ /* describe parts */
   {
      "key": 0,
      "val": {
         "start_part_num": 0, /* part number */
         "start_ofs": 0, /* from S3 object */
         "part_size": 5242880,
         "stripe_max_size": 4194304, /* split */
      },
```

- *Trivial* parts: single item only
- *Multipart* parts: one for each S3 multipart!

Object Parts

- Example: 14MiB uploaded in 3 parts max. 5MiB

Object Parts

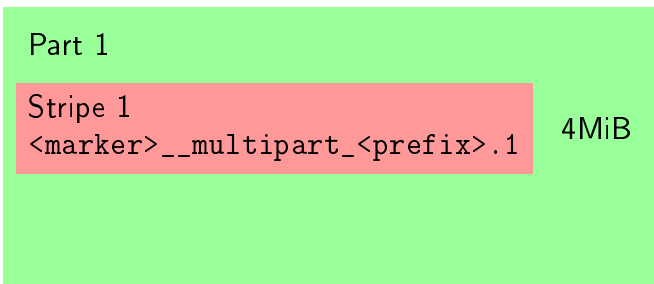
- Example: 14MiB uploaded in 3 parts max. 5MiB
- We expect: 5MiB, 5MiB, 4MiB

Object Parts

- Example: 14MiB uploaded in 3 parts max. 5MiB
- We expect: 5MiB, 5MiB, 4MiB

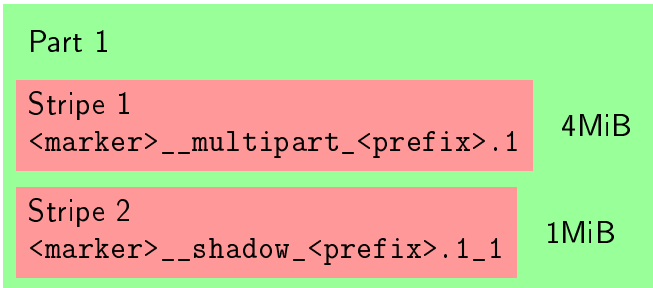
Object Parts

- Example: 14MiB uploaded in 3 parts max. 5MiB
- We expect: 5MiB, 5MiB, 4MiB



Object Parts

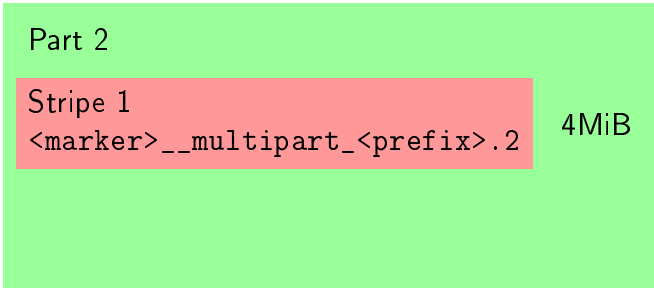
- Example: 14MiB uploaded in 3 parts max. 5MiB
- We expect: 5MiB, 5MiB, 4MiB



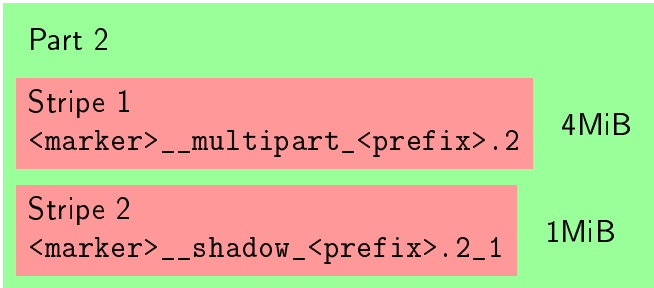
Object Parts (Cont.)

Part 2

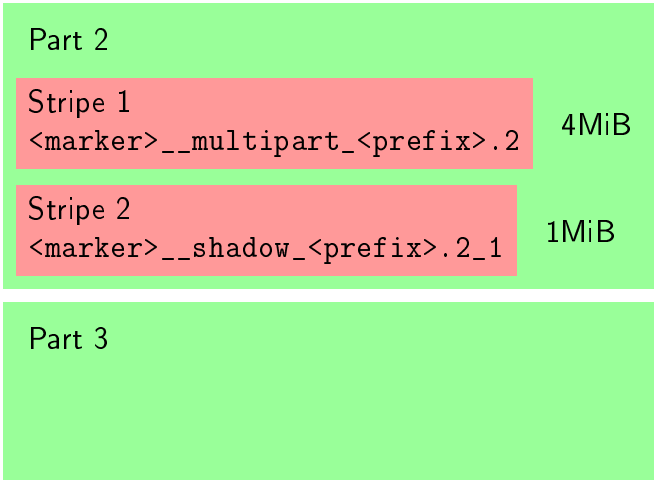
Object Parts (Cont.)



Object Parts (Cont.)



Object Parts (Cont.)



Object Parts (Cont.)

Part 2

Stripe 1
`<marker>__multipart_<prefix>.2` — 4MiB

Stripe 2
`<marker>__shadow_<prefix>.2_1` — 1MiB

Part 3

Stripe 1
`<marker>__multipart_<prefix>.3` — 4MiB

Object Parts (Cont.)

This setup allows processing:

- parts *in parallel*

Object Parts (Cont.)

This setup allows processing:

- parts *in parallel*
- parts *out of order*

Object Parts (Cont.)

This setup allows processing:

- parts *in parallel*
- parts *out of order*
- stripes while they are received

Other Pools

Log pool `<zone>.rgw.log`

Other Pools

Log pool `<zone>.rgw.log`

- Data- and metadata log multisite sync

Other Pools

Log pool `<zone>.rgw.log`

- Data- and metadata log multisite sync
- Garbage collection (namespace gc)

Questions?

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)
- Therefore we need to increase the number of shards: resharding

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)
- Therefore we need to increase the number of shards: resharding

```
# get current number of shards
$ radosgw-admin bucket stats --bucket mybucket \
    | jq -r .num_shards
```

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)
- Therefore we need to increase the number of shards: resharding

```
# get current number of shards
$ radosgw-admin bucket stats --bucket mybucket \
    | jq -r .num_shards
11
```

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)
- Therefore we need to increase the number of shards: resharding

```
# get current number of shards
$ radosgw-admin bucket stats --bucket mybucket \
    | jq -r .num_shards
11
# increase manually - use prime numbers
$ radosgw-admin bucket reshard --bucket mybucket \
    --num-shards 101
```

Bucket Index Resharding

- When a bucket grows, entries per shard may reach critical level (large OMAP warning)
- Therefore we need to increase the number of shards: resharding

```
# get current number of shards
$ radosgw-admin bucket stats --bucket mybucket \
    | jq -r .num_shards
11
# increase manually - use prime numbers
$ radosgw-admin bucket reshard --bucket mybucket \
    --num-shards 101
$ radosgw-admin reshard process
$
```

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999
  - Hard upper limit: there only primes up to 1999 defined!

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999
  - Hard upper limit: there only primes up to 1999 defined!
- *Blocks* writes!

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999
  - Hard upper limit: there only primes up to 1999 defined!
- *Blocks* writes!
  - $10 * 10 * 5s = 500s$ (10 attempts, 10 retries, 5s timeout)

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999
  - Hard upper limit: there only primes up to 1999 defined!
- *Blocks* writes!
  - $10 * 10 * 5s = 500s$ (10 attempts, 10 retries, 5s timeout)
  - Any write to bucket index fails with `ERR_BUSY_RESHARDING`

Bucket Index Resharding: Automatic

- Automatically based on *number of objects* in bucket
- Up to a configurable maximum
  - `rgw_max_dynamic_shards`, default 1999
  - Hard upper limit: there only primes up to 1999 defined!
- *Blocks* writes!
  - $10 * 10 * 5s = 500s$ (10 attempts, 10 retries, 5s timeout)
  - Any write to bucket index fails with `ERR_BUSY_RESHARDING`
  - But: pending implementation that would eliminate this block