OpenStack Object Storage: Swift vs CEPH

Giacinto Donvito
Marica Antonacci - INFN Bari

Scuola di Cloud Storage
Bari, Ottobre 2016

Outline

- Swift introduction
  - Key Elements & Concepts
  - Architecture
- Swift Geographically distributed cluster
- Hints on Ceph Object storage
- Swift vs Ceph

Swift

- Swift is the software behind the OpenStack Object Storage service.
- Written in python. Over 100+ contributors from many Org.
- Provides a simple storage service for applications using RESTful interfaces
- Provides maximum data availability and storage capacity.

Production deployments

- Wikipedia
- Rackspace
- Hp Cloud
- MercadoLibre
- Disney
- …

Swift Key Elements

Massive Scaling with Eventual Consistency

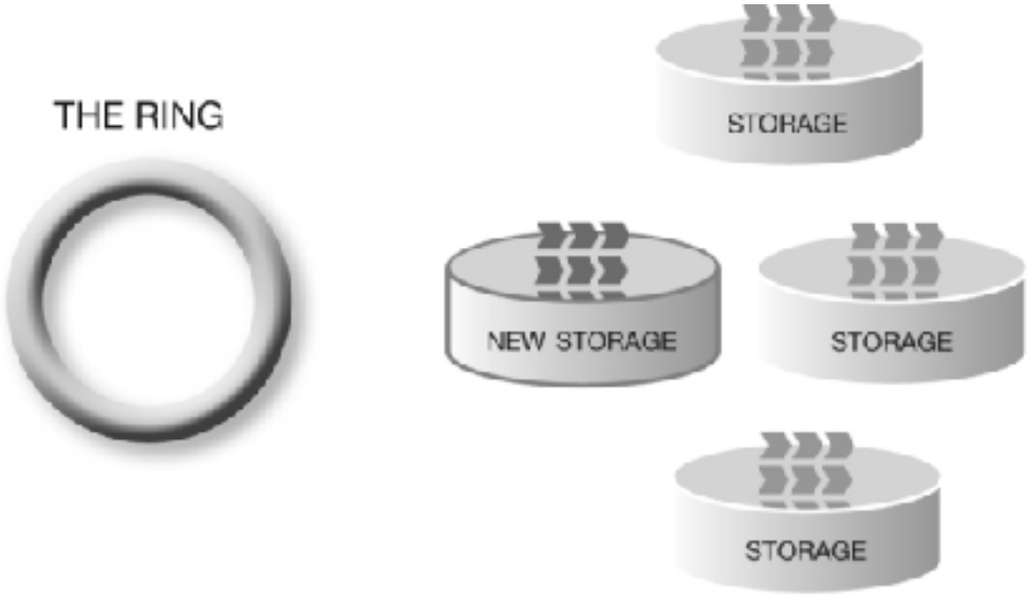
- Objects are protected by storing multiple copies of data so that if one node fails, the data can be retrieved from another node

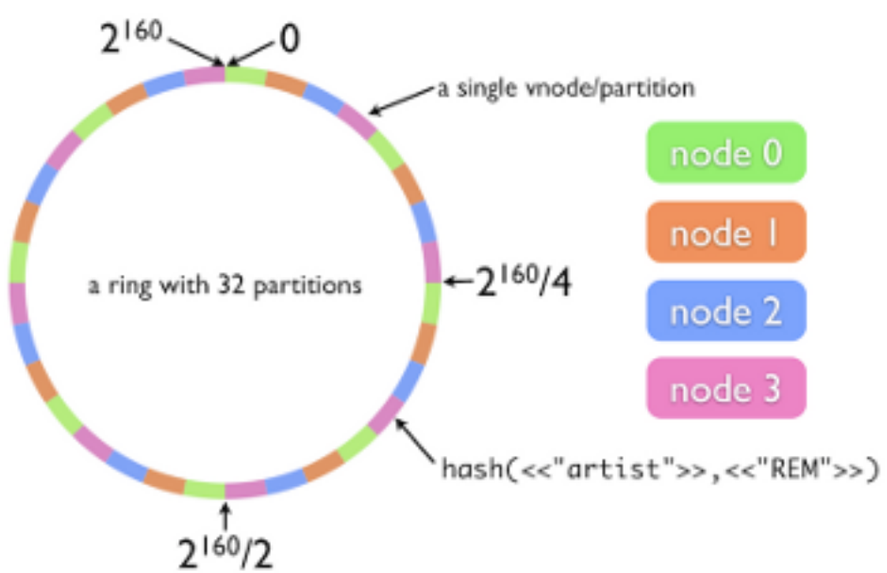
- That means that you'll always get your data, they will be dispersed on many places, but you could get an old version of them (or no data at all) in some odd cases (like some server overload or failure).

- But there are mechanisms built into Swift to minimize the potential data inconsistency window: they are responsible for data replication and consistency.

Consistent hashing

- **Ring**: represents the space of all possible computed hash values divided in equivalent parts. Each part of this space is called a partition.





- Rings are used to deduce where a particular piece of data is stored.

Data duplication

- By default, Swift stores **3 copies** of every objects, but that's configurable.

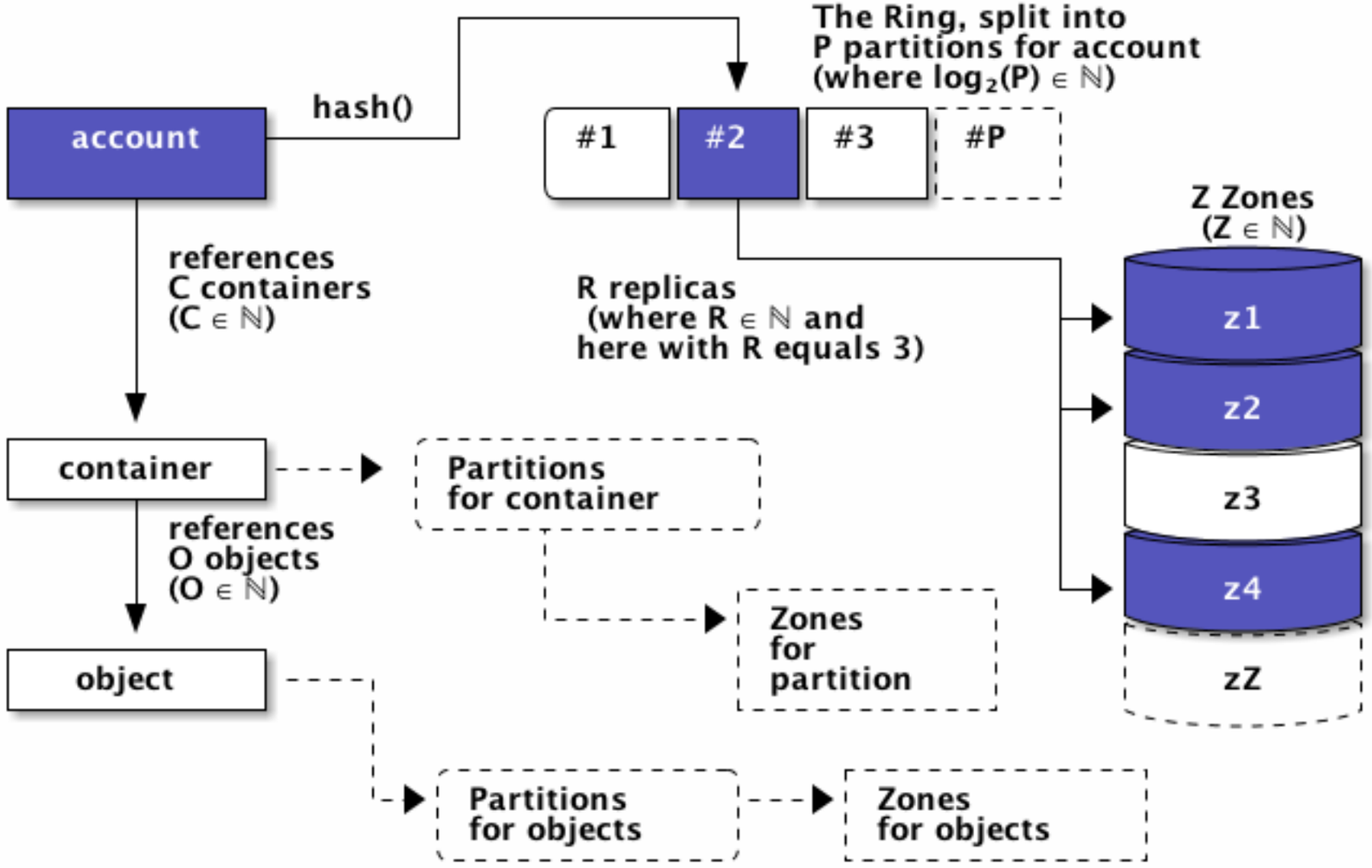
- **Zone**: is an isolated space that does not depend on other zone, so in case of an outage on a zone, the other zones are still available.

- Concretely, a zone is likely to be a disk, a server, or a whole cabinet, depending on the size of your cluster.

- Each partition is replicated; each replica is placed as uniquely as possible across the cluster.

accounts, containers, objects

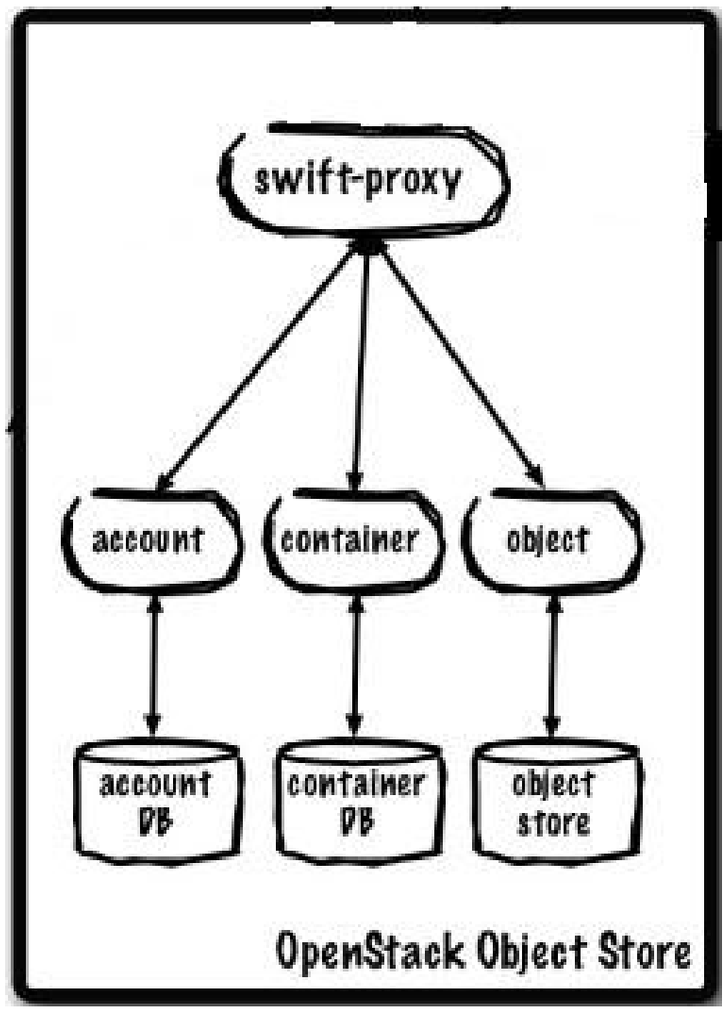
- In Swift, there are 3 categories of thing to store: account, container and objects —> 3 independent rings



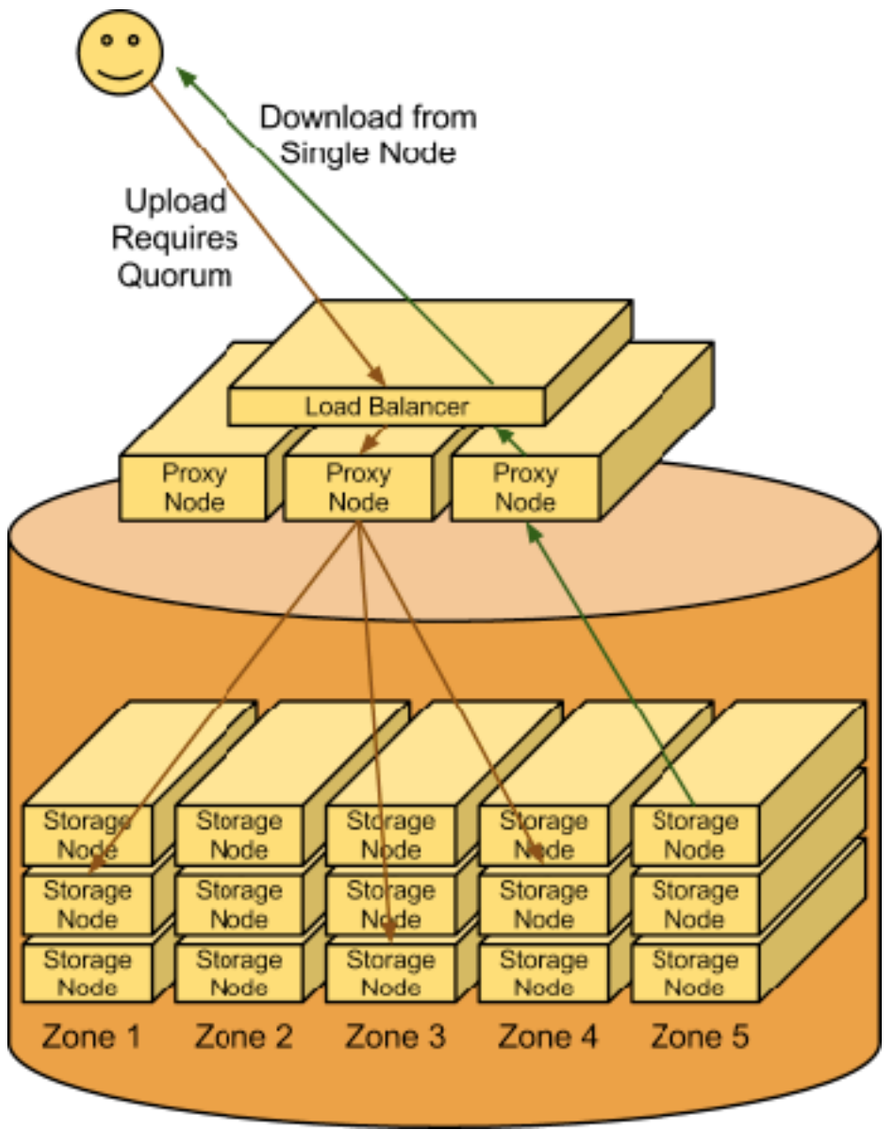
Swift Architecture

Swift Components

- Proxy Server
- Object Server
- Container Server
- Account Server
- Replication
- Updaters
- Auditors
- Reapers



Swift deployment



- Swift is a **two-tier** storage system consisting of
  - a ***proxy tier***, which handles all incoming requests;
  - an ***object storage tier*** where the actual data is stored.
- In addition, **consistency** processes run to ensure the integrity of the data.

Data Access

- By default, Swift OpenStack provides
  - RESTful APIs
  - CLI client (python-swiftclient)
- It is possible to enable standard interfaces, like **S3** (Amazon-compliant APIs) and **CDMI** (Cloud Data Management Interface), adding the corresponding middleware name in the proxy-server pipeline and its parameter section.
  S3 Example: /etc/swift/proxy-server.conf

```
[pipeline:main]
pipeline = healthcheck cache swift3 s3token  authtoken keystoneauth proxy-logging  proxy-server
…
[filter:swift3]
use = egg:swift3#swift3

[filter:s3token]
paste.filter_factory = keystone.middleware.s3_token:filter_factory
auth_port = 35357
auth_host = controller
auth_protocol = https
```

- For CDMI the following extra package has to be installed too: https://github.com/osaddon/cdmi

Data Security

- Protect the cluster endpoint, enabling SSL support in the proxy servers or using an SSL terminator (e.g. Pound)

- Node-to-node communication happens via HTTP —> deploy them on secure network (e.g. VLAN)

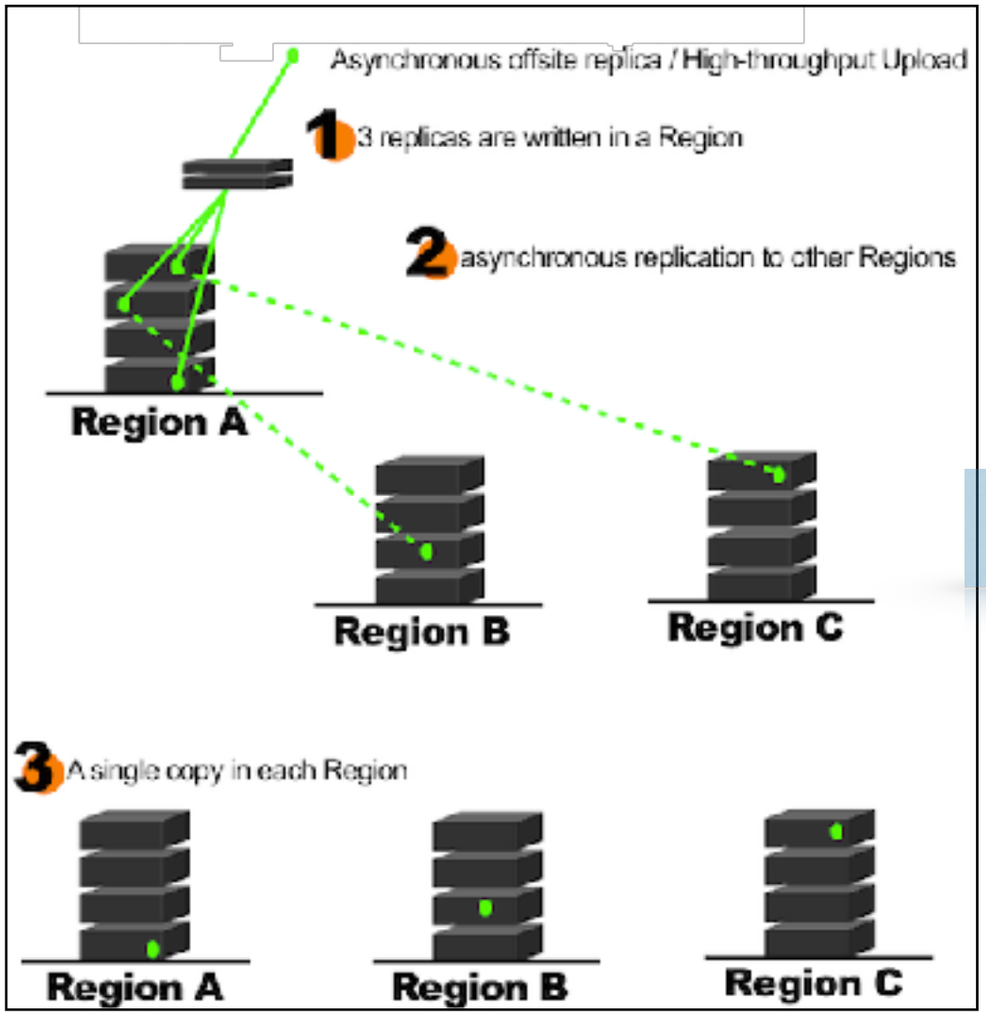
- Node-to-node replication: rsync traffic is not encrypted —> use a secure network

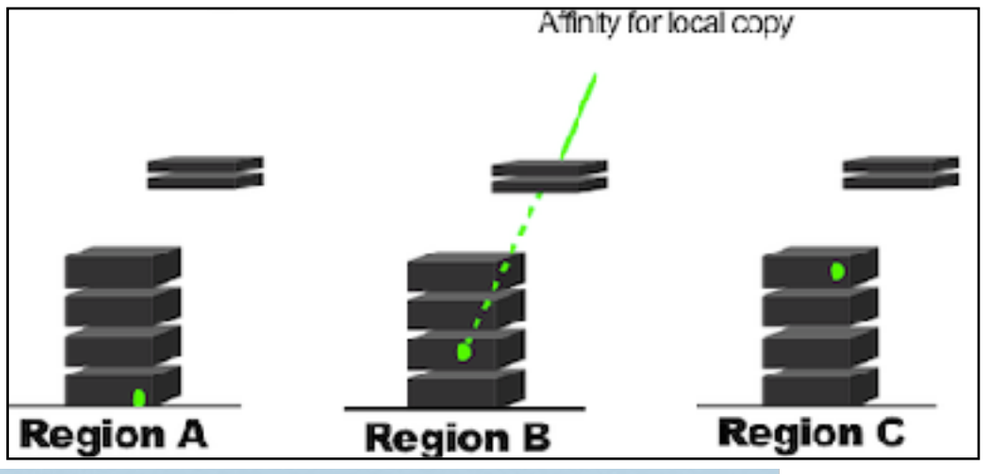
- Data encryption: relies on the backend storage system

Geographically distributed Swift cluster

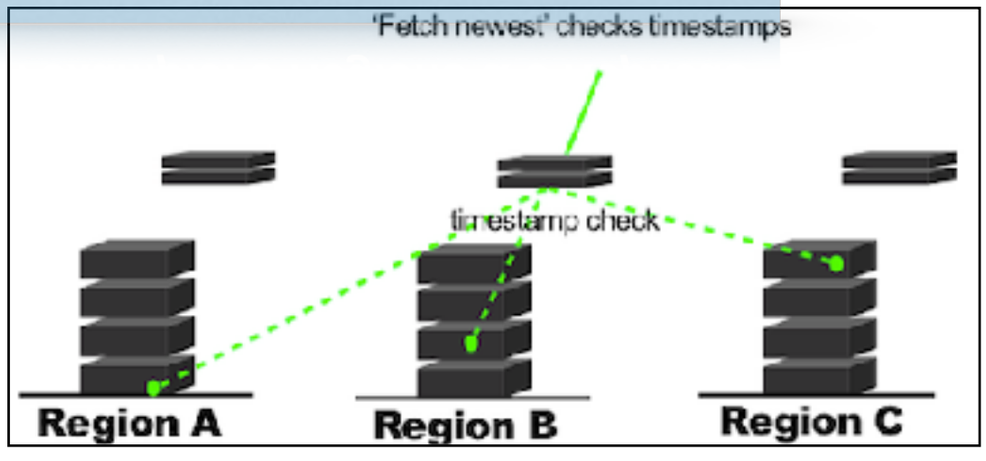
Multi-regional clusters

- A **Region** represents an additional level of tiering, or a group of zones, so all the devices that belong to zones constituting a single region must belong to this region.
- The proxy nodes will have an **affinity** to a Region and be able to optimistically write to storage nodes based on the storage nodes' Region. Optionally, the client will have the option to perform a write or read that goes across Regions (ignoring local affinity),if required.





An example: 3 Regions & 3 Replicas



Swift vs Ceph

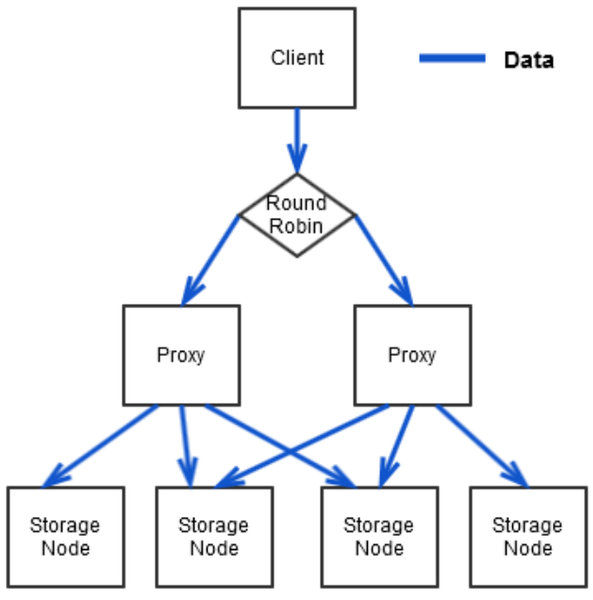
Architectural perspective

OpenStack Swift transfers data through proxy servers which then distribute data to the Storage nodes.

Independent proxy servers

No caching

OpenStack Swift

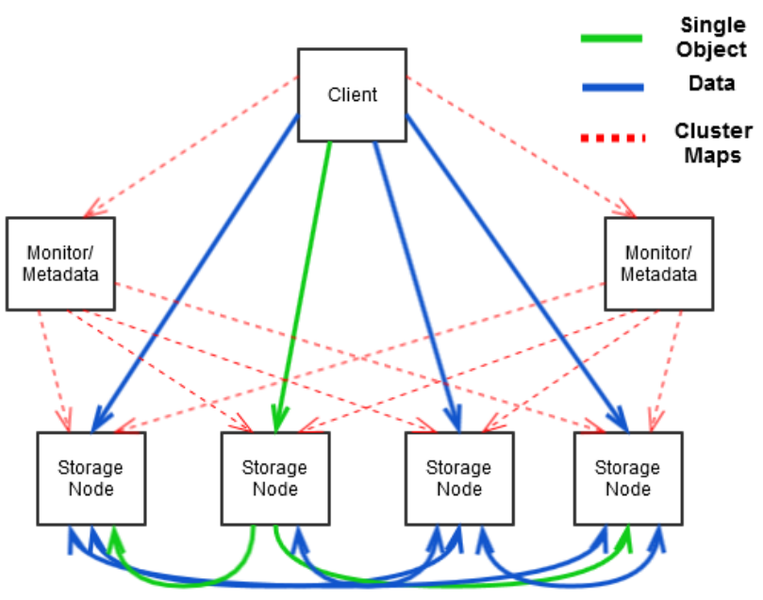


Ceph clients connect directly to the Storage nodes eliminating any bottleneck.

Monitor quorum

Journal and Cache tier

Ceph



CEPH Object storage

- Nelle ultime release il supporto all'Object Storage in CEPH è stato oggetto di miglioramenti importanti.
  - The new multitenancy infrastructure improves compatibility with Swift, which provides a separate container namespace for each user/tenant.
  - The OpenStack Keystone v3 API is now supported. There are a range of other small Swift API features and compatibility improvements as well, including bulk delete and SLO (static large objects).
  - The multisite feature has been almost completely rearchitected and rewritten to support any number of clusters/sites, bidirectional fail-over, and active/active configurations.
  - You can now access radosgw buckets via NFS (experimental).
  - The AWS4 authentication protocol is now supported.
  - There is now support for S3 request payer buckets.
  - The Swift API now supports object expiration.

RGW: Swift compliance

FEATURES SUPPORT

The following table describes the support status for current Swift functional features:

Feature	Status	Remarks
Authentication	Supported	
Get Account Metadata	Supported	
Swift ACLs	Supported	Supports a subset of Swift ACLs
List Containers	Supported	
Delete Container	Supported	
Create Container	Supported	
Get Container Metadata	Supported	
Update Container Metadata	Supported	
Delete Container Metadata	Supported	
List Objects	Supported	
Static Website	Not Supported	
Create Object	Supported	
Create Large Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Copy Object	Supported	
Get Object Metadata	Supported	
Update Object Metadata	Supported	
Expiring Objects	Supported	
Object Versioning	Supported	
CORS	Not Supported	

RGW: S3 compliance

FEATURES SUPPORT

The following table describes the support status for current Amazon S3 functional features:

Feature	Status	Remarks
List Buckets	Supported	
Delete Bucket	Supported	
Create Bucket	Supported	Different set of canned ACLs
Bucket Lifecycle	Not Supported	
Policy (Buckets, Objects)	Not Supported	ACLs are supported
Bucket Website	Not Supported	
Bucket ACLs (Get, Put)	Supported	Different set of canned ACLs
Bucket Location	Supported	
Bucket Notification	Not Supported	
Bucket Object Versions	Supported	
Get Bucket Info (HEAD)	Supported	
Bucket Request Payment	Supported	
Put Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Object ACLs (Get, Put)	Supported	
Get Object Info (HEAD)	Supported	
POST Object	Supported	
Copy Object	Supported	
Multipart Uploads	Supported	(missing Copy Part)

CEPH Object storage

- CONFIGURING CEPH OBJECT GATEWAY WITH APACHE/FASTCGI
- CREATE A USER AND KEYRING
- CREATE POOLS
- USING THE GATEWAY:
    - CREATE A RADOSGW USER FOR S3 ACCESS:
        - radosgw-admin user create —uid="testuser"
    - CREATE A SWIFT USER
        - radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --access=full
        - radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-secret
    - swift -A http://{IP ADDRESS}/auth/1.0 -U testuser:swift -K ‘{swift_secret_key}’ list

CEPH Object storage

- ADD / REMOVE ADMIN CAPABILITIES

- The Ceph Object Gateway enables you to set quotas on users and buckets owned by users. Quotas include the maximum number of objects in a bucket and the maximum storage size in megabytes.

    - Bucket: The --bucket option allows you to specify a quota for buckets the user owns.

    - Maximum Objects: The --max-objects setting allows you to specify the maximum number of objects. A negative value disables this setting.

    - Maximum Size: The --max-size option allows you to specify a quota size in B/K/M/G/T. A negative value disables this setting.

    - Quota Scope: The --quota-scope option sets the scope for the quota. The options are bucket and user. Bucket quotas apply to buckets a user owns. User quotas apply to a user.

CEPH Object storage

- The Ceph Object Gateway logs usage for each user. You can track user usage within date ranges too.

- Options include:

  - Start Date: The --start-date option allows you to filter usage stats from a particular start date (format: yyyy-mm-dd[HH:MM:SS]).

  - End Date: The --end-date option allows you to filter usage up to a particular date (format: yyyy-mm-dd[HH:MM:SS]).

  - Log Entries: The --show-log-entries option allows you to specify whether or not to include log entries with the usage stats (options: true | false).

- radosgw-admin usage show --uid=johndoe --start-date=2012-03-01 --end-date=2012-04-01

CEPH Object storage

- INTEGRATING WITH OPENSTACK KEYSTONE

```
openstack service create --name=swift \
                         --description="Swift Service" \
                         object-store
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Swift Service                    |
| enabled     | True                             |
| id          | 37c4c0e79571404cb4644201a4a6e5ee |
| name        | swift                            |
| type        | object-store                     |
+-------------+----------------------------------+

openstack endpoint create --region RegionOne \
     --publicurl   "http://radosgw.example.com:8080/swift/v1" \
     --adminurl    "http://radosgw.example.com:8080/swift/v1" \
     --internalurl "http://radosgw.example.com:8080/swift/v1" \
     swift
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| adminurl     | http://radosgw.example.com:8080/swift/v1 |
| id           | e4249d2b60e44743a67b5e5b38c18dd3        |
| internalurl  | http://radosgw.example.com:8080/swift/v1 |
| publicurl    | http://radosgw.example.com:8080/swift/v1 |
| region       | RegionOne                               |
| service_id   | 37c4c0e79571404cb4644201a4a6e5ee        |
| service_name | swift                                   |
| service_type | object-store                            |
+--------------+-----------------------------------------+

$ openstack endpoint show object-store
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| adminurl     | http://radosgw.example.com:8080/swift/v1 |
| enabled      | True                                    |
| id           | e4249d2b60e44743a67b5e5b38c18dd3        |
| internalurl  | http://radosgw.example.com:8080/swift/v1 |
| publicurl    | http://radosgw.example.com:8080/swift/v1 |
| region       | RegionOne                               |
| service_id   | 37c4c0e79571404cb4644201a4a6e5ee        |
| service_name | swift                                   |
| service_type | object-store                            |
+--------------+-----------------------------------------+
```

CEPH Object storage

- **RGW MULTI-TENANCY: New in version Jewel.**

- The multi-tenancy feature allows to use buckets and users of the same name simultaneously by segregating them under so-called tenants. This may be useful, for instance, to permit users of Swift API to create buckets with easily conflicting names such as “test” or “trove”.

- From the Jewel release onward, each user and bucket lies under a tenant. For compatibility, a “legacy” tenant with an empty name is provided. Whenever a bucket is referred without an explicit tenant, an implicit tenant is used, taken from the user performing the operation. Since the pre-existing users are under the legacy tenant, they continue to create and access buckets as before. The layout of objects in RADOS is extended in a compatible way, ensuring a smooth upgrade to Jewel.

- S3:

  - # radosgw-admin --tenant testx --uid tester --display-name "Test User" --access_key TESTER --secret test123 user create

- SWIFT:

  - # radosgw-admin --tenant testx --uid tester --display-name "Test User" --subuser tester:test --key-type swift --access full user create

  - # radosgw-admin --subuser 'testx$tester:test' --key-type swift --secret test123

- https://ep.host.dom/tenant:bucket

PYTHON S3 EXAMPLES

CREATING AN OBJECT

This creates a file `hello.txt` with the string `"Hello World!"`

```
key = bucket.new_key('hello.txt')
key.set_contents_from_string('Hello World!')
```

CHANGE AN OBJECT'S ACL

This makes the object `hello.txt` to be publicly readable, and `secret_plans.txt` to be private.

```
hello_key = bucket.get_key('hello.txt')
hello_key.set_canned_acl('public-read')
plans_key = bucket.get_key('secret_plans.txt')
plans_key.set_canned_acl('private')
```

DOWNLOAD AN OBJECT (TO A FILE)

This downloads the object `perl_poetry.pdf` and saves it in `/home/larry/documents/`

```
key = bucket.get_key('perl_poetry.pdf')
key.get_contents_to_filename('/home/larry/documents/perl_poetry.pdf')
```

PYTHON S3 EXAMPLES

GENERATE OBJECT DOWNLOAD URLS (SIGNED AND UNSIGNED)

This generates an unsigned download URL for `hello.txt`. This works because we made `hello.txt` public by setting the ACL above. This then generates a signed download URL for `secret_plans.txt` that will work for 1 hour. Signed download URLs will work for the time period even if the object is private (when the time period is up, the URL will stop working).

```
hello_key = bucket.get_key('hello.txt')
hello_url = hello_key.generate_url(0, query_auth=False, force_http=True)
print hello_url

plans_key = bucket.get_key('secret_plans.txt')
plans_url = plans_key.generate_url(3600, query_auth=True, force_http=True)
print plans_url
```

The output of this will look something like:

```
http://objects.dreamhost.com/my-bucket-name/hello.txt
http://objects.dreamhost.com/my-bucket-name/secret_plans.txt?Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXX&Expires=1
```

Issues

- **Swift** main issues:
  - traffic to and from the Swift cluster flows through the proxy servers
  - eventual consistency: object replicas aren't necessarily updated at the same time
- **Ceph** main issues:
  - Ceph is quite sensitive to clock drift
  - multi-region support based on a master-slave model
  - security: RADOS clients on the compute nodes communicates directly with the RADOS server