We are OpenInfra!

Operating OpenStack Swift in real life

Seongsoo Cho (NHN Cloud / OpenInfra Korea User Group)

2025.02.01 FOSDEM 2025

OpenInfra USER GROUP Korea

Challenges Faced by Object Storage in Public Cloud Services

Unpredictable request volume and traffic.

Many small files rather than large files.

Strong dependency when used as backend storage for other services (e.g., CortexMetrics in Prometheus).

Must ensure stable service even during data rebalance.

OpenStack Swift

One of the initial OpenStack Projects

- Initially used as a storage backend by Glance for storing cloud OS image data.

Unlike other OpenStack components, it can be deployed independently.

Supports authentication systems: TempAuth (temporary), Keystone, and Custom Auth.

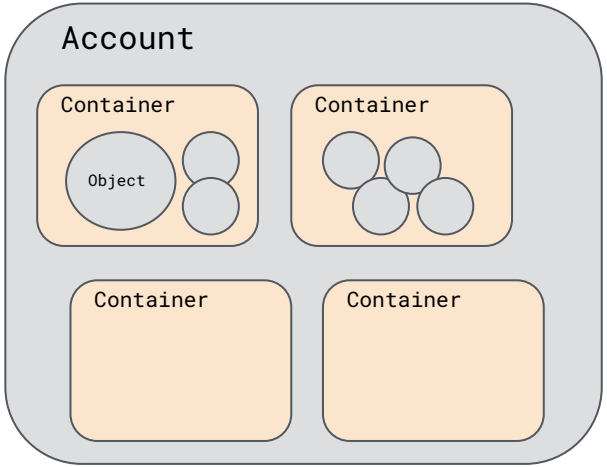
- Custom Auth : https://docs.openstack.org/swift/latest/development_auth.html

OpenStack Swift Components (Logical Concepts)

- Account
  The top-level concept in Swift is the Account, which is required to use Swift
  An Account serves as a namespace for defining Containers.

- Container
  A Container is a storage space for objects (It is not a folder).
  Container names must be unique within a single Account.

- Object
  Unstructured data such as documents, images, and videos.



OpenStack Swift URI Format

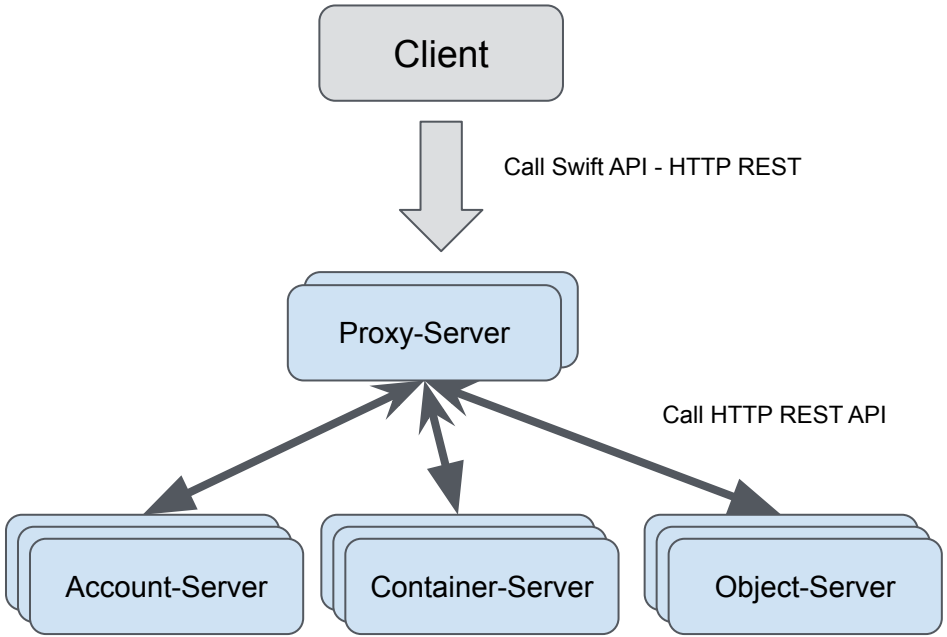
URI Format : /account/container/object

account : keystone project id

ex) /AUTH_testuser/test_container/a/b/c/test_object.txt

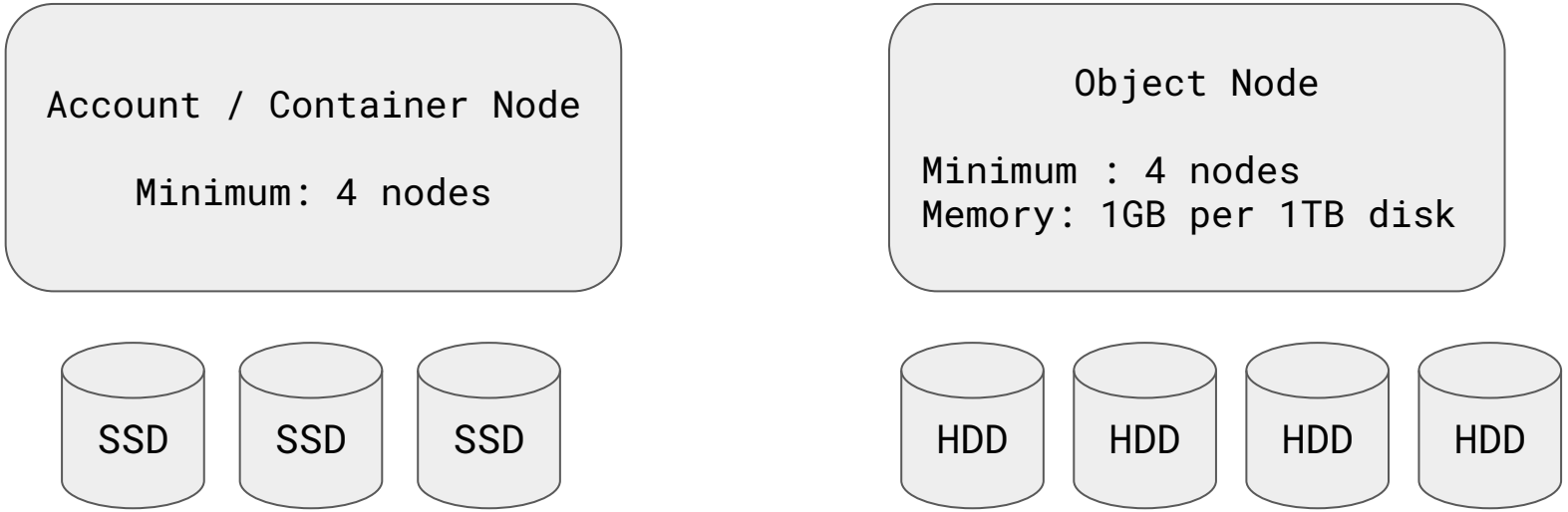
Sample: https://service.com/v1/AUTH_testuser/test_container/a.txt

OpenStack Swift Component (Node)



- **Proxy-Server**
  Handle Client request, auth, encrypt and forward the request to the backend
- **Account-Server**
  Storing account info into SQLite3
- **Container-Server**
  Storing Container info into SQLite3
- **Object-Server**

Ops Story: Minimal Deployment

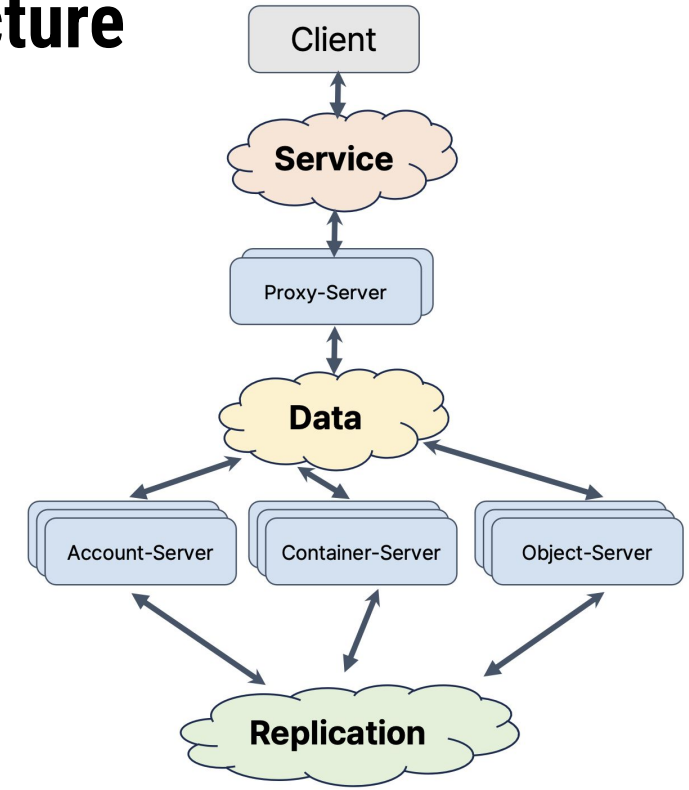


Separate Account and Container disks on the same node

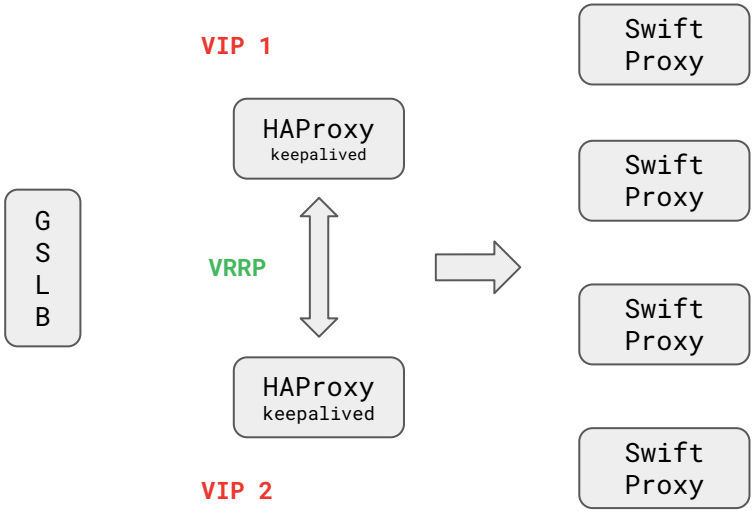
Ensure consistent disk sizes across all nodes whenever possible.

OpenStack Swift Network Architecture

Service Network	access from public, Network handling incoming user traffic.
Data Network	private network, can’t access from public. Internal traffic for handling user requests.
Replication Network	private network, Network for transferring data to maintain replicas within Swift.



Ops Story: Load Balancing for Service Traffic



swift.service.com (CNAME)
- swift.gslb.service.com (A)
  - VIP 1 (50%)
  - VIP 2 (50%)

Next Ops Story

Scale out of Object Nodes

Determining data storage location - Ring

Swift’s unique Ring, based on Consistent Hash Ring. [1]

A file that determines where the Account, Container, and Object data should be stored (on which server and disk).

The Ring file must be stored on all nodes.

All Proxy, Account, Container, and Object servers must have the same Ring content.

[1] https://docs.openstack.org/swift/latest/overview_ring.html

Determining data storage location - Components of Ring

- Partition Power : Factors Determining the Number of Partitions
  - The number of partitions is calculated as 2^N.
  - The physical space where data (objects) are stored.
- Number of Replication
- Device List
- Device Lookup Table

Ring - Device List

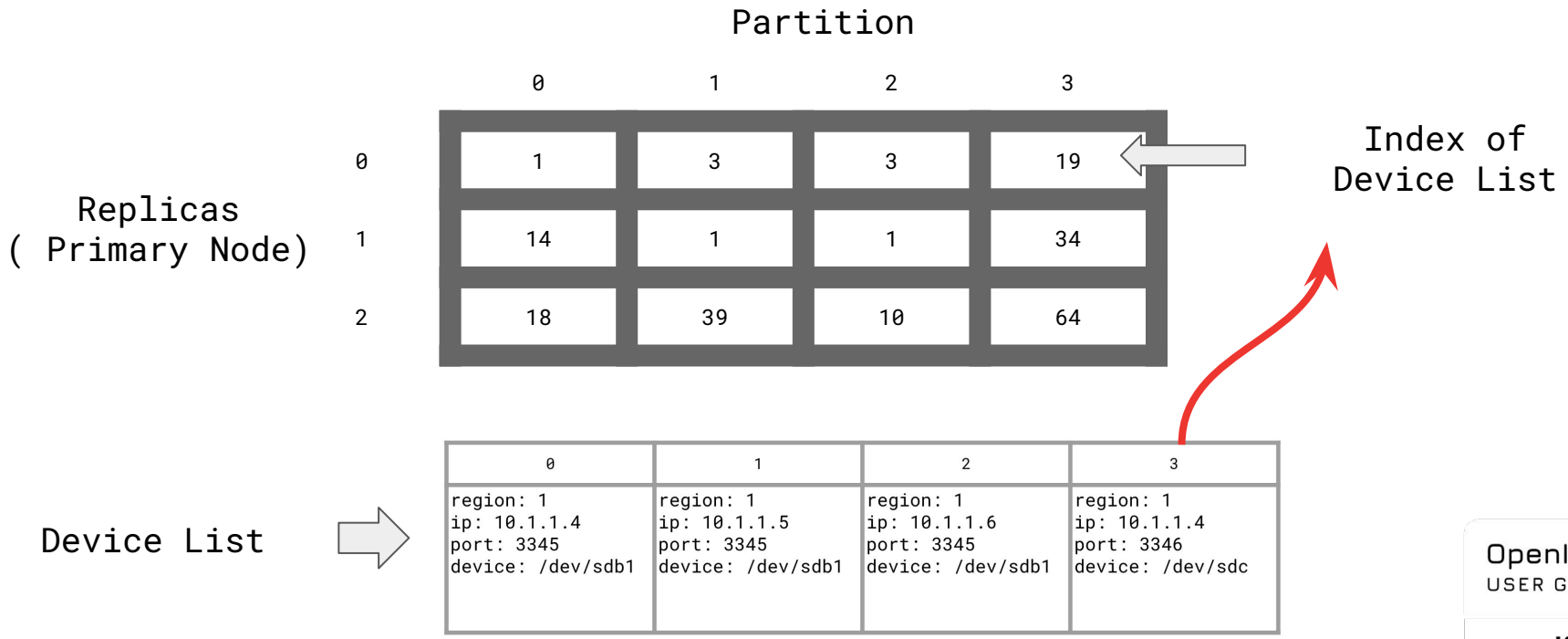
Device List

- An array of information about the disks that Swift will use.
- Each index contains information about how to access the disk.

0	1	2	3
region: 1 ip: 10.1.1.4 port: 3345 device: /dev/sdb1	region: 1 ip: 10.1.1.5 port: 3345 device: /dev/sdb1	region: 1 ip: 10.1.1.6 port: 3345 device: /dev/sdb1	region: 1 ip: 10.1.1.4 port: 3346 device: /dev/sdc

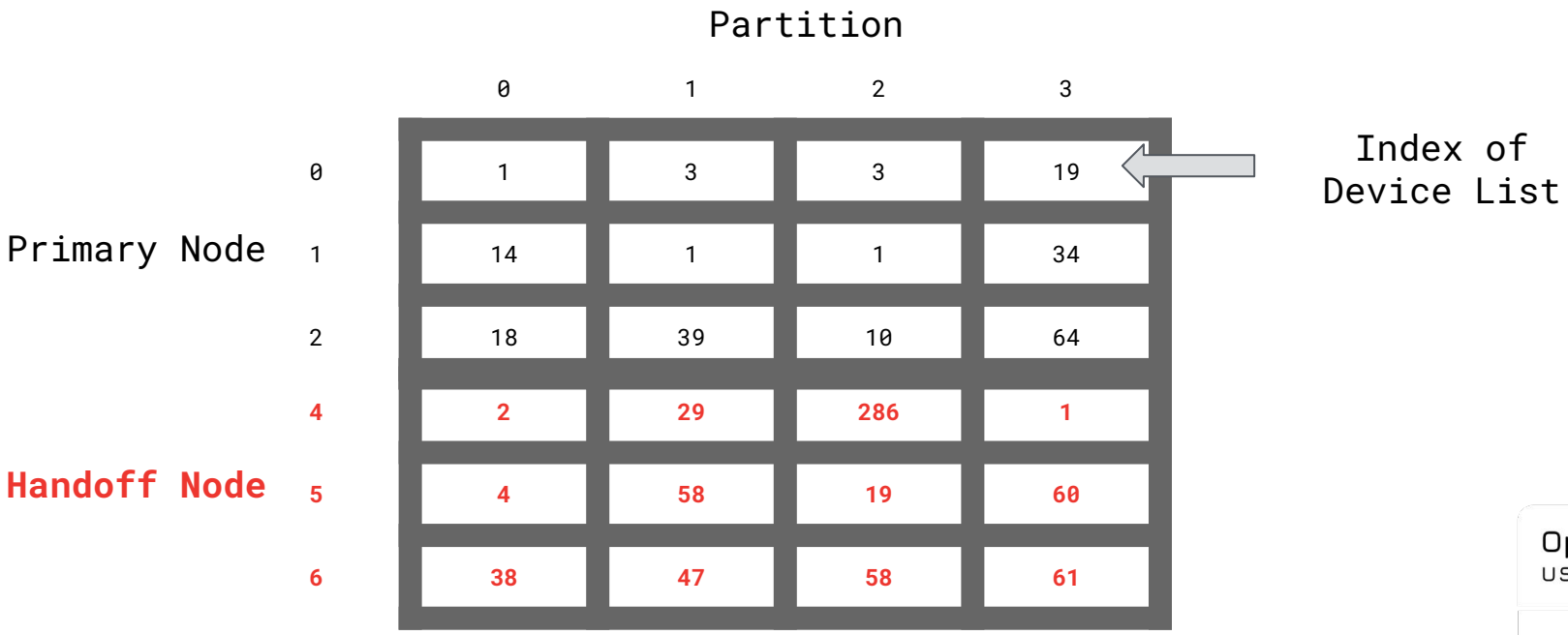
Ring - Device Lookup Table

A table that stores which disks each partition's 3 replica copies should be stored on



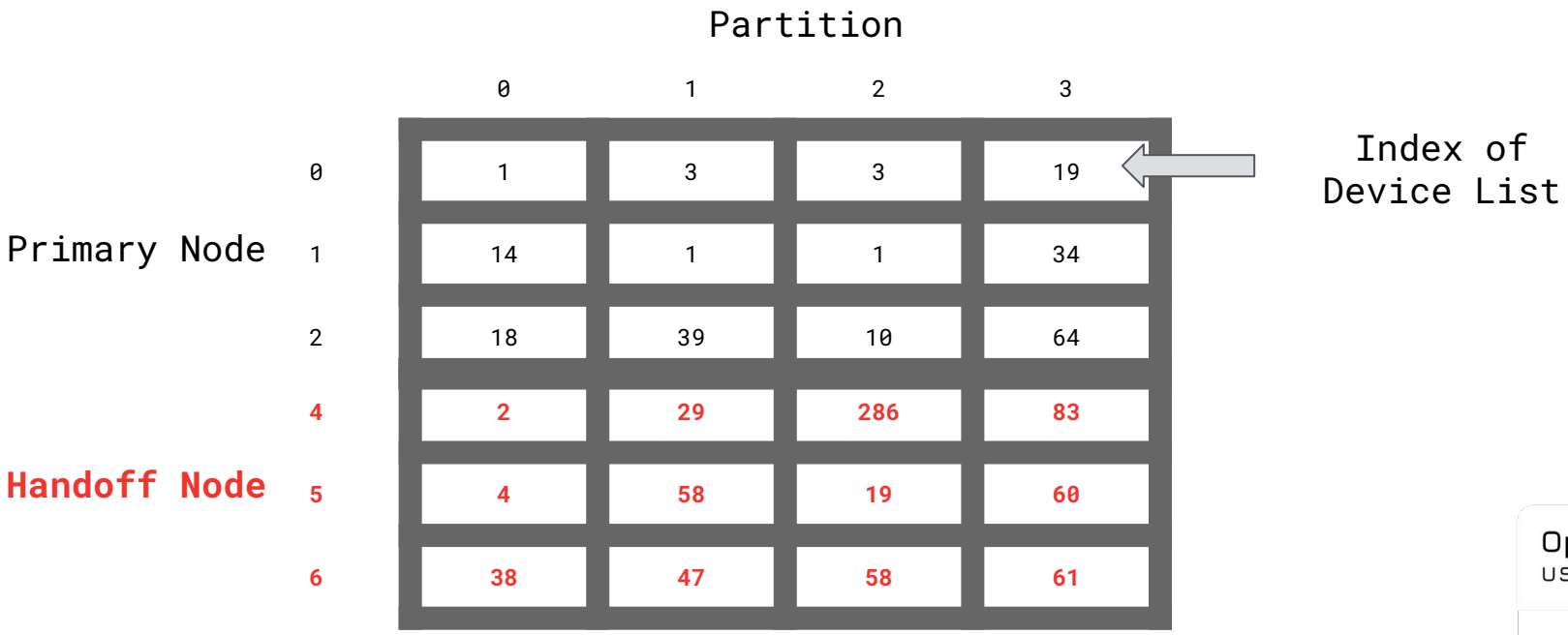
Ring - Handoff Node

If the primary node fails, a handoff node is temporarily designated in the Ring to store the data.



Object Storage Location Determination Method

Where should “AUTH_test/container/my_object.txt” be stored?



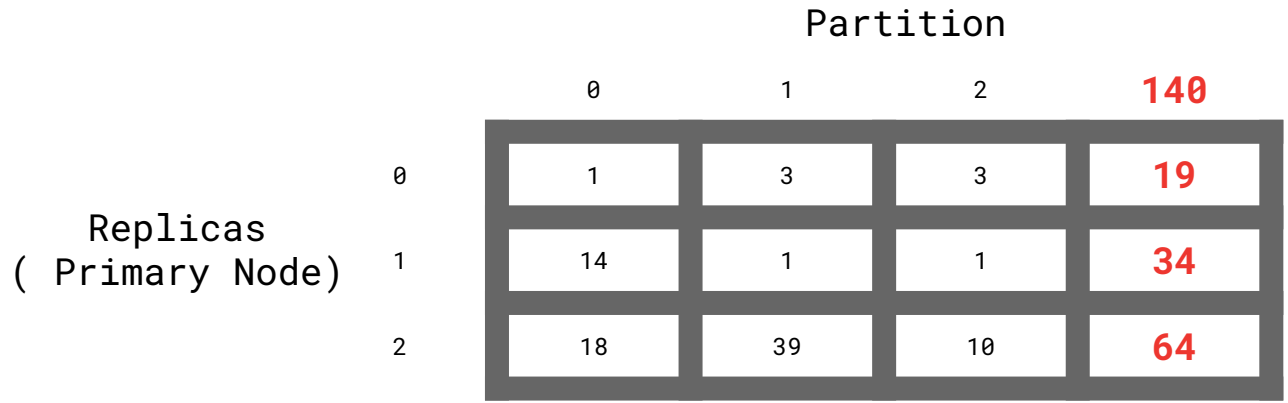
OpenInfra
USER GROUP
Korea

Object Storage Location Determination Method

The Proxy server computes the MD5 hash of the request URI.
ex) md5(“AUTH_test/container”)

The MD5 value is then taken modulo the number of partitions to determine the result.
ex) “2205229274494a9243f23f7e653977c” % 2048 = 140

140 represents the partition number in the device lookup table.
The data location is determined by using the device index number assigned to the primary node for partition 140 in the device lookup table.



Physical Data Storage Structure

```
root@object_node: /srv/node/sdb/objects # ls
1003  1200  1302  1557  1846  2096  2307  2666  2917  313   3363  3500  3704  3839  487  701
1015  1203  1312  1568  1870  2133  2321  2674  2966  3153  3372  3536  3707  386   532  704
1020  1233  1318  1628  1874  214   2355  2678  297   3165  3378  3543  3713  3902  537  706
1037  1243  1346  1629  1876  2140  2356  2679  2988  3179  3380  3546  3715  3953  569  711
```

How to add a new object node?

The Device List contains not only disk information but also a value called weight.

Weight: The proportion of the disk in the overall pool

- A larger weight means the disk is assigned to more partitions.

I set the weight as the disk size in GB.

0	1	2	3
region: 1 ip: 10.1.1.4 port: 3345 device: /dev/sdb1 **weight: 4000**	region: 1 ip: 10.1.1.5 port: 3345 device: /dev/sdb1 **weight: 5000**	region: 1 ip: 10.1.1.6 port: 3345 device: /dev/sdb1 **weight: 4000**	region: 1 ip: 10.1.1.4 port: 3346 device: /dev/sdc **weight: 7000**

Adding a new object node.

Adding a new node means adding a new disk to the Device List.

The newly added disk must be placed in the Device Lookup Table.

At this point, the contents of the Device Lookup Table change,
== disks are reassigned to partitions,
== data migration occurs.

Data migration uses the replication network to avoid impacting service traffic.

OpenStack Swift Ring

```
$ swift-ring-builder object.builder
object.builder, build version 287, id d389325d9a194a9f966123033b60962c
4096 partitions, 3.000000 replicas, 1 regions, 4 zones, 48 devices, 0.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file object.ring.gz is up-to-date
Devices:   id region zone   ip address:port replication ip:port  name weight partitions balance flags
meta
            0      1    7 172.17.49.19:6001   172.17.69.19:6100   sdb 3700.00        256    0.00
            1      1    7 172.17.49.19:6002   172.17.69.19:6100   sdc 3700.00        256    0.00
            2      1    7 172.17.49.19:6003   172.17.69.19:6100   sdd 3700.00        256    0.00
            3      1    7 172.17.49.19:6004   172.17.69.19:6100   sde 3700.00        256    0.00
            4      1    7 172.17.49.19:6005   172.17.69.19:6100   sdf 3700.00        256    0.00
```

[1] https://rackerlabs.github.io/swift-ppc/

Increase Partition Power

Adding nodes means increasing the number of disks in the Ring.

The number of partitions is determined when the Ring is first created.

With a Partition Power of 10, up to 1024 disks can be used.

Swift provides a feature to increase Partition Power. (Increase Partition Power) [1]

- This process creates hard links for all objects, causing high disk load.

[1] https://docs.openstack.org/swift/latest/ring_partpower.html

OpenStack Swift Ring Calculator

Get your ring calculation on!

Calculate your Openstack Swift Partition Power

Max Drives:	1000
Min Drives:	10
Replicas:	3

Enter the maximum number of drives you ever expect to have in your cluster, how many drives you plan to start with, and how many replicas (copies of your data) you want.

Ring Partition Info

Recommended Part Power: 16

Total Partition Count: 65536

Max per drive: 19661

Min per drive: 197

Ring Builder Commands

```
swift-ring-builder account.builder create 16 3 1
swift-ring-builder container.builder create 16 3 1
swift-ring-builder object.builder create 16 3 1
```

[1] https://rackerlabs.github.io/swift-ppc/

Ops Story: Slow object replication

Even with sufficient disk utilization and network bandwidth, object replication can be very slow.

If a partition contains a large amount of data, updating hashes.pkl takes a significant amount of time.

In my case, each partition contained tens of millions of files, and processing a single disk with 100 partitions took three days.

Ops Story: Container DB 3 copy mismatch.

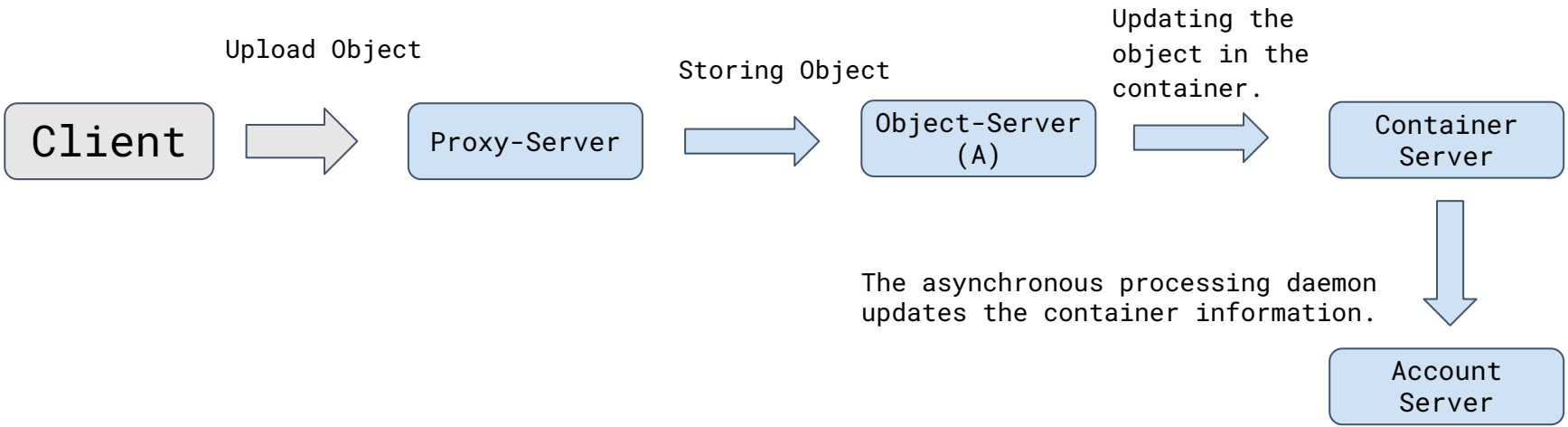
The Container DB stores the object list and container information.

The Container DB also has 3 copies for replication.

SQLite3 is used for the database.

The process of updating the object in the container DB

The Account/Container DB must be updated for each object upload/delete.



The impact of container DB 3-copy data mismatch

- As the service grows, data inconsistencies can occur in the container DB.

-> This should never happen, but it can occur if the container pool is too small compared to the overall request volume.

The impact of container DB 3-copy data mismatch

- As the service grows, data inconsistencies can occur in the container DB.

  -> This should never happen, but it can occur if the container pool is too small compared to the overall request volume.

- When data inconsistency occurs among the three copies of the container DB, several troublesome situations arise.
  - Files uploaded via DLO become undownloadable (e.g., cloud os images).
  - The multipart file list for a.img is stored with a part prefix, and data is retrieved via object listing. If the container DB is unstable, listing fails.
- From the user’s perspective, it seems like the object has disappeared.

How are we solving the problem?

Perform a full scan of the object list in the 3-copy DB and manually correct the discrepancies.

For example, if data exists in copies A and B but not in C, we add the data to C.

How to monitor OpenStack Swift?

What metrics are being monitored?

System Resource

- CPU / Memory / Network / Etc..

API

- Collect logs to monitor request volume, response time, and response codes.

Recon

- You can check the status of Swift daemons.

Statsd

- You can view the timing data within Swift.

API Monitoring

- Collect and analyze logs, then visualize them on a dashboard
- Tools used : Filebeat / Logstash / ElasticSearch / Grafana
- Metrics
  - API request volume trends
  - Response code trends
  - Top 10 accounts/containers with the most uploads/downloads
  - Upload/download traffic volume derived from logs

Recon Monitoring

`swift-recon (option) (target)`

- ring md5
- time sync
- daemon running stats
- disk usage

```
$ swift-recon
===============================================================================
Usage:
        usage: swift-recon <server_type> [<server_type> [<server_type>]]
        [-v] [--suppress] [-a] [-r] [-u] [-d] [-R]
        [-l] [-T] [--md5] [--auditor] [--updater] [--expirer] [--sockstat]
        [--human-readable]

        <server_type>   account|container|object
        Defaults to object server.

        ex: swift-recon container -l --auditor


Options:
  -h, --help            show this help message and exit
  -v, --verbose         Print verbose info
  --suppress            Suppress most connection related errors
  -a, --async           Get async stats
  -r, --replication     Get replication stats
  -R, --reconstruction  Get reconstruction stats
  --auditor             Get auditor stats
  --updater             Get updater stats
  --expirer             Get expirer stats
  --sharding            Get sharding stats
  -u, --unmounted       Check cluster for unmounted devices
  -d, --diskusage       Get disk usage stats
  --human-readable      Use human readable suffix for disk usage stats
  -l, --loadstats       Get cluster load average stats
  -q, --quarantined     Get cluster quarantine stats
  --validate-servers    Validate servers on the ring
  --md5                 Get md5sum of servers ring and compare to local copy
  --sockstat            Get cluster socket usage stats
  --driveaudit          Get drive audit error stats
  -T, --time            Check time synchronization
  --jitter=JITTER       Maximal allowed time jitter
  --swift-versions      Check swift versions
  --top=COUNT           Also show the top COUNT entries in rank order.
  --lowest=COUNT        Also show the lowest COUNT entries in rank
                        order.
  --all                 Perform all checks. Equal to
                        -arRudlqT --md5 --sockstat --auditor --updater
                        --expirer --driveaudit --validate-servers --swift-
                        versions
  --region=REGION       Only query servers in specified region
  -z ZONE, --zone=ZONE  Only query servers in specified zone
  -t SECONDS, --timeout=SECONDS
                        Time to wait for a response from a server
  --swiftdir=SWIFTDIR   Default = /etc/swift
  -p POLICY, --policy=POLICY
                        Only query object servers in specified storage policy
                        (specified as name or index).
```

Recon Monitoring

Ex) Check the replication status of the Object Replicator.

```
$ swift-recon -r object
===============================================================================
--> Starting reconnaissance on 4 hosts (object)
===============================================================================
[2024-12-09 01:23:34] Checking on replication
[replication_time] low: 0, high: 1, avg: 1.1, total: 4, Failed: 0.0%, no_result: 0, reported: 4
[replication_failure] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported: 4
[replication_success] low: 0, high: 6186, avg: 4626.8, total: 18507, Failed: 0.0%, no_result: 0, reported: 4
[replication_attempted] low: 0, high: 3086, avg: 2310.2, total: 9241, Failed: 0.0%, no_result: 0, reported: 4
Oldest completion was 2024-11-09 01:21:32 (2 minutes ago) by 192.168.0.22:6001.
Most recent completion was 2024-11-09 01:23:29 (4 seconds ago) by 192.168.0.102:6001.
===============================================================================
```

Statsd timing Monitoring

Metrics showing the time taken for specific operations within Swift as time-series data.

Example: Time taken to receive the first byte when downloading data from the proxy server to the object server.

proxy-logging Middleware

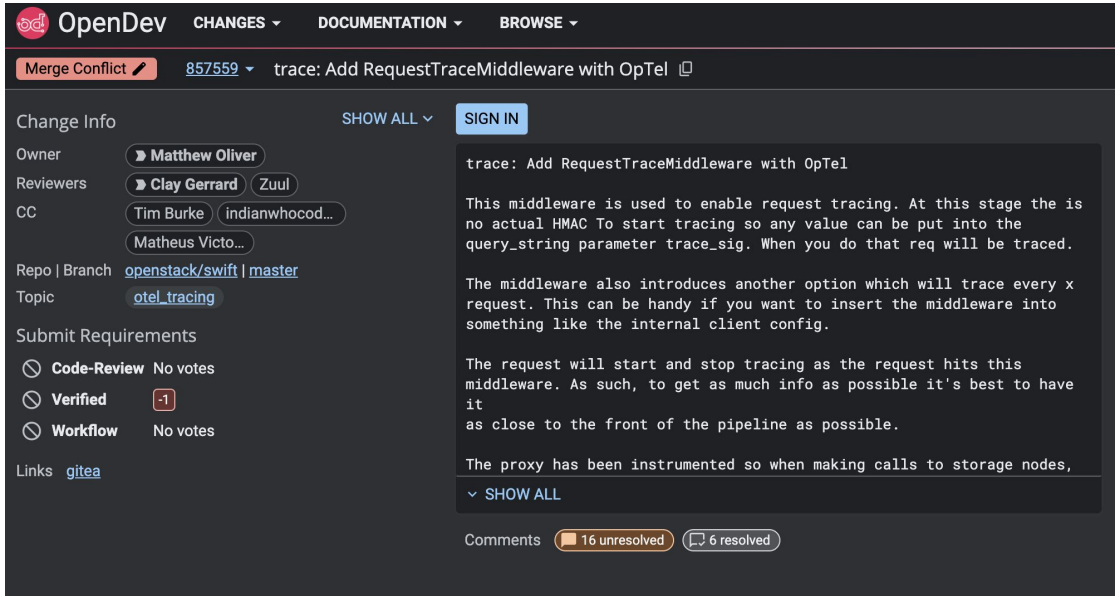
In the table, **<type>** is either the proxy-server controller responsible for the request: **account**, **container**, **object**, or the string **SOS** if the request came from the Swift Origin Server middleware. The **<verb>** portion will be one of **GET**, **HEAD**, **POST**, **PUT**, **DELETE**, **COPY**, **OPTIONS**, or **BAD_METHOD**. The list of valid HTTP methods is configurable via the **log_statsd_valid_http_methods** config variable and the default setting yields the above behavior.

Metric Name	Description
proxy-server.<type>.<verb>.<status>.timing	Timing data for requests, start to finish. The <status> portion is the numeric HTTP status code for the request (e.g. "200" or "404").
proxy-server.<type>.GET.<status>.first-byte.timing	Timing data up to completion of sending the response headers (only for GET requests). <status> and <type> are as for the main timing metric.
proxy-server.<type>.<verb>.<status>.xfer	This counter metric is the sum of bytes transferred in (from clients) and out (to clients) for requests. The <type>, <verb>, and <status> portions of the metric are just like the main timing metric.

New feature in progress upstream - OpenTelemetry

https://review.opendev.org/c/openstack/swift/+/857559

Development of OpenTelemetry Integration Module for Swift Request Tracing



Overall Evaluation of OpenStack Swift

Operational experience and knowledge are not as widely available as they are for Ceph.

Operators need to have code-level understanding to accurately diagnose and address issues.

Developing monitoring and operational tools requires significant effort.

There is a lack of additional features. (e.g encryption, IP ACL)

Thank you

Seongsoo (OpenInfra Korea User Group)

https://www.linkedin.com/in/seongsoocho/

OpenInfra
USER GROUP

Korea