Distributed Storage Systems

Object Storage

Different Data Storage Systems

-

Different Data Storage Systems

- Relational databases (SQL)
-

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
-

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
- Block storage
-

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
- Block storage
  - Oldest and simplest
  - Data stored in fixed-sized chunks called “blocks”
  - Block typically houses a portion of the data
  - Address: only identifying part of a block — no block metadata
  - Good performance when app & storage are local, but can lead to more latency the further apart they are
  - Addressing requirements limit scalability

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
- Block storage
- File Storage
-

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
- Block storage
- File Storage
  - User names files/data, places them in folders, & can nest them under more folders (hierarchy)
  - File has a limited set of metadata associated with it
  - Works well with smaller files, some issues when retrieving large amounts of data
  - Scaling is a problem: it becomes harder to find Information
  - Unique addresses→ finite number of files you can store
  - Sharing via NAS: great locally, issues over wide area networks

Different Data Storage Systems

- Relational databases (SQL)
- NoSQL
- Block storage
- File Storage
- Object storage

What is an Object?

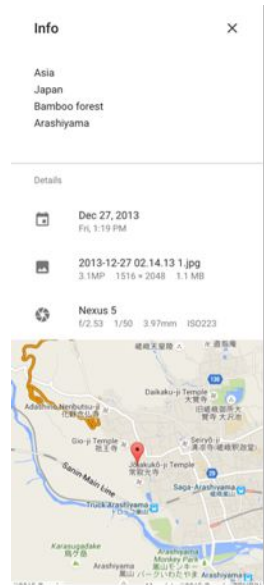
File:
15x6x3_10x00x41741x64561
_21x58x511_n.jpg

Slide adapted from Albert Chen

What is an Object?

File:
15x6x3_10x00x41741x64561
_21x58x511_n.jpg

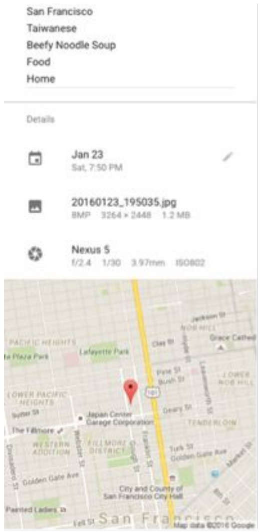
Metadata:
Image size: 1516x2048
Date taken: 2013-12-27 13:19
Tags: Asia,
Japan,
Bamboo forest,
Arashiyama
GPS: 35.016520, 135.670436



- **Object = File + Metadata**

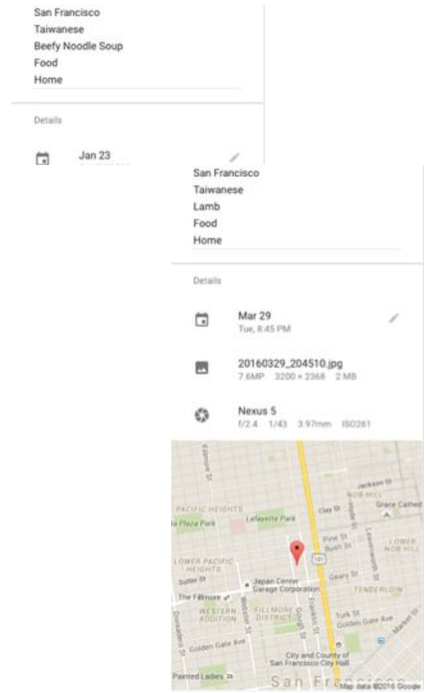
Slide adapted from Albert Chen

We have a lot of objects



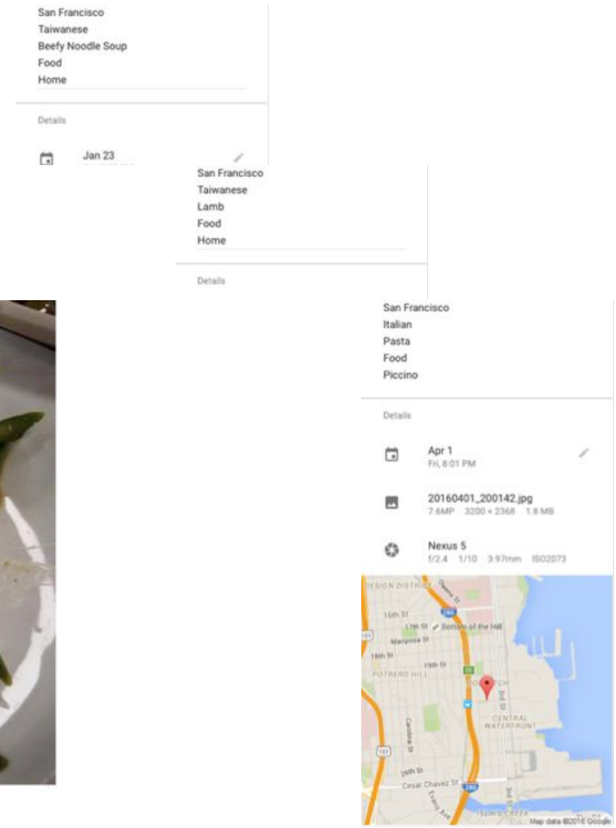
Slide adapted from Albert Chen

We have a lot of objects



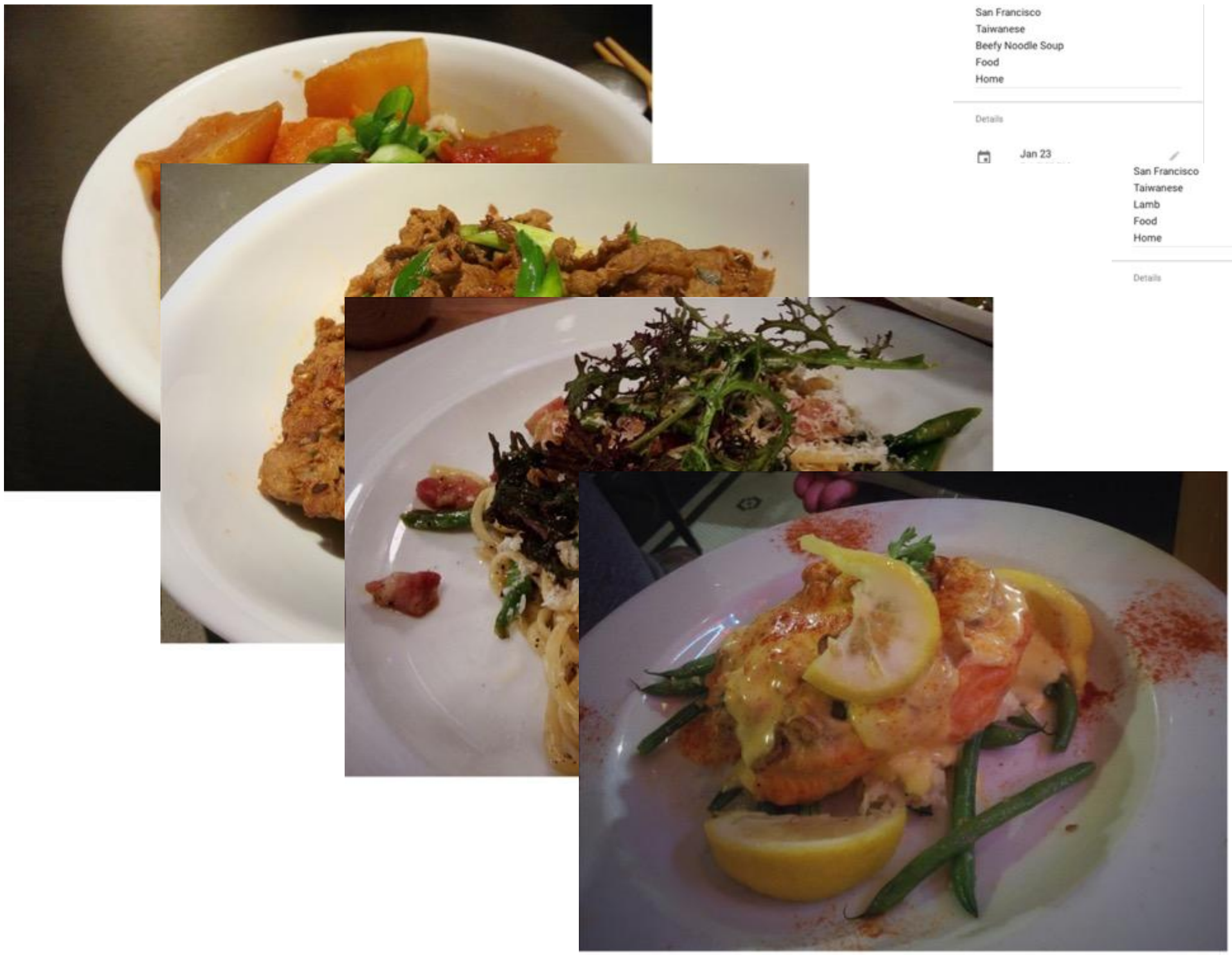
Slide adapted from Albert Chen

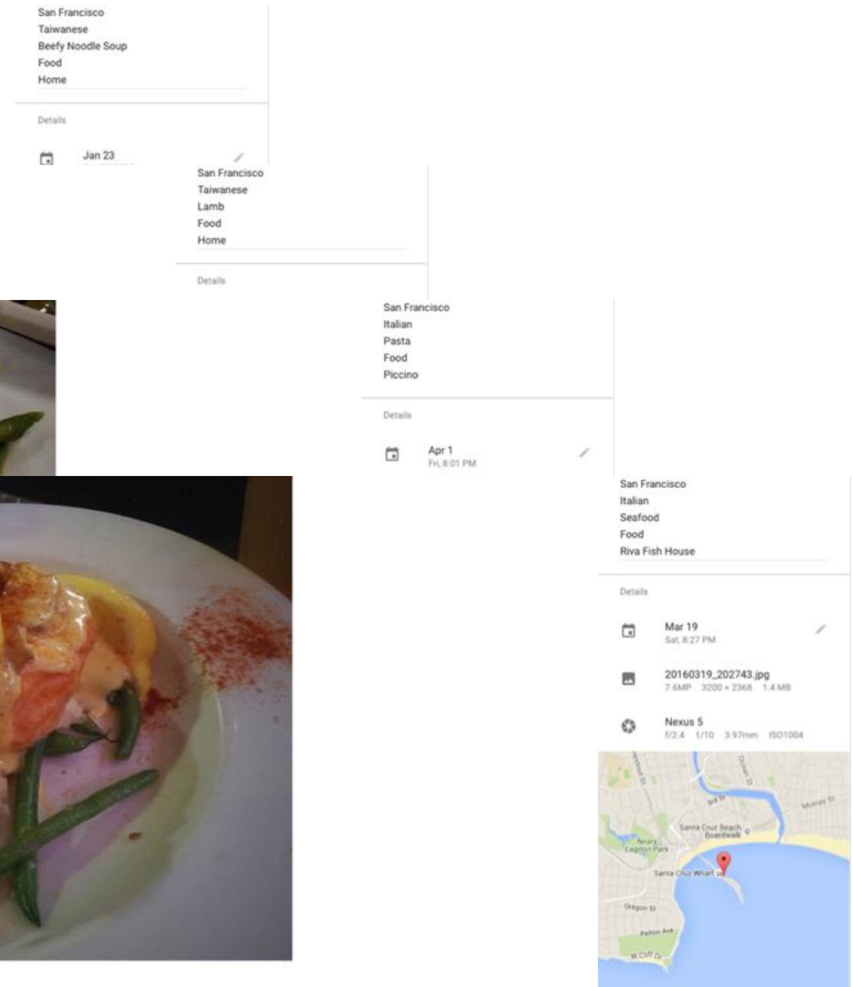
We have a lot of objects



Slide adapted from Albert Chen

We have a lot of objects





ert Chen

What is for Dinner?

Metadata Search:

Taiwanese Beefy Noodle

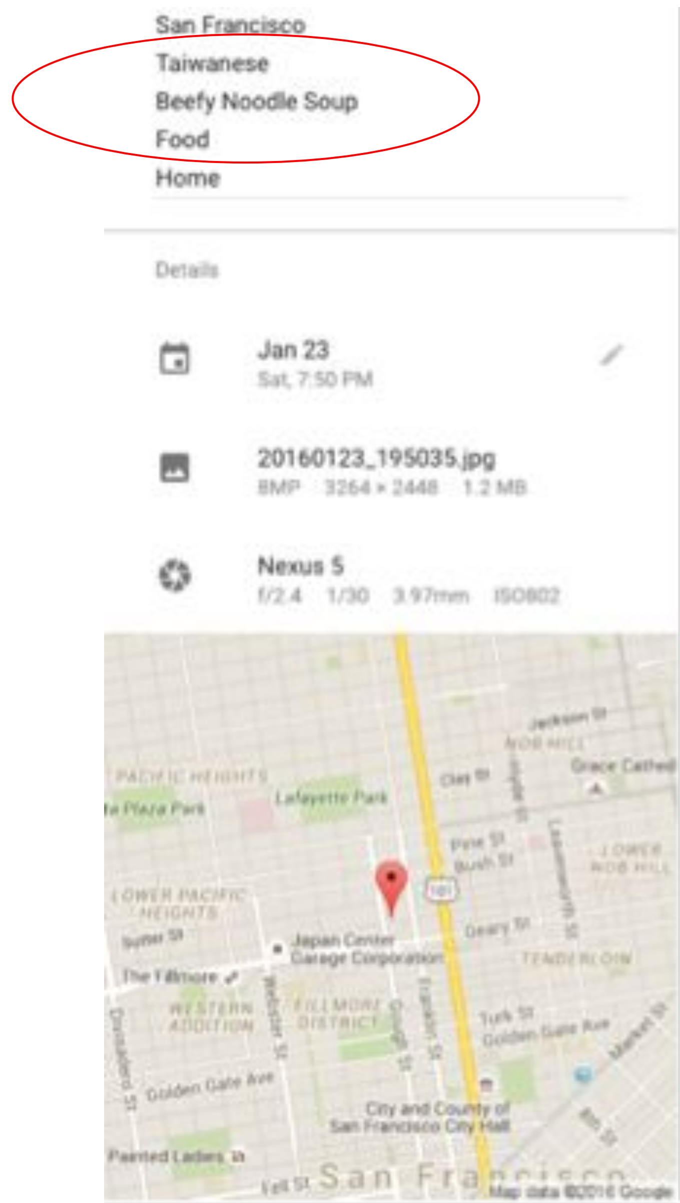
Metadata info is stored with:

- Object
- Searchable Index

This allows for meta data search

Slide adapted from Albert Chen

What is for Dinner?



Metadata Search:

Taiwanese Beefy Noodle

Metadata info is stored with:

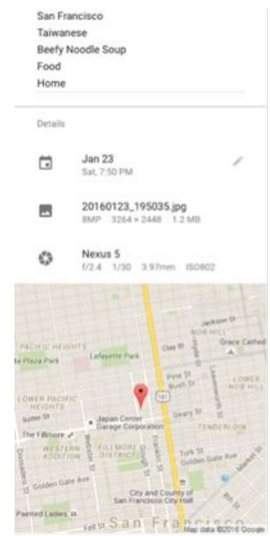
- Object
- Searchable Index

This allows for meta data search

Slide adapted from Albert Chen

What is for Dinner?

Metadata Search: Taiwanese Beefy Noodle

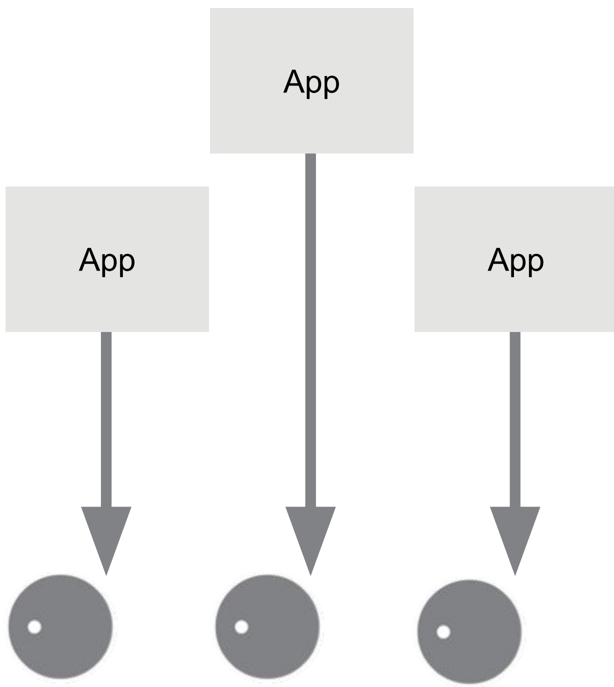


Slide adapted from Albert Chen

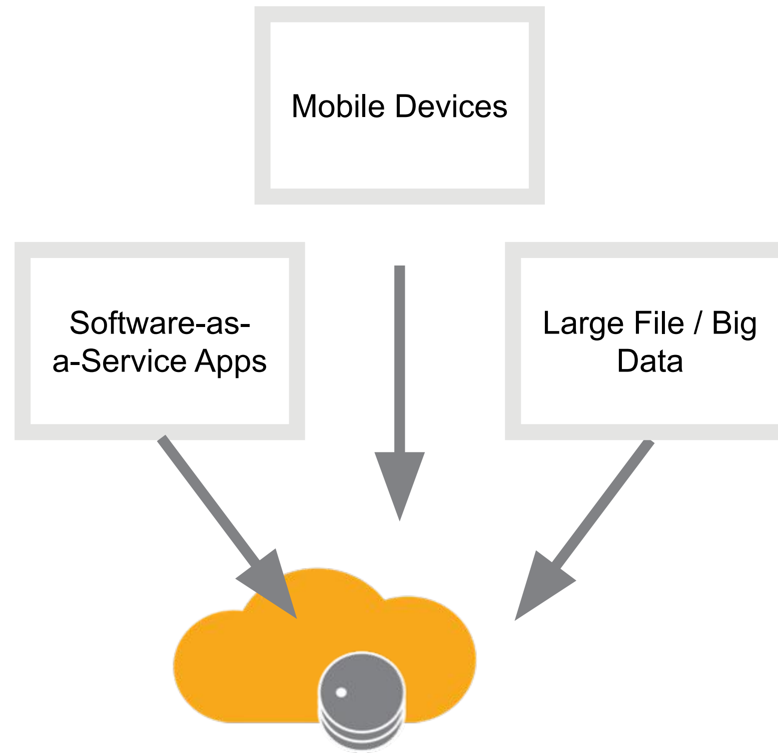
Why Object Storage?

Why would we need Objects?

Traditional - File-based



Object Storage - HTTP Namespace



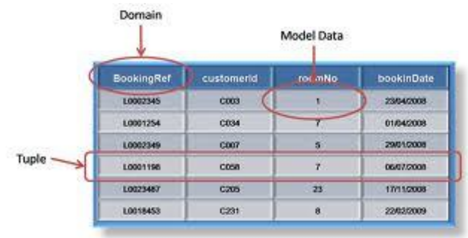
Slide adapted from Albert Chen

Right Tool for the Job in the Cloud

	Object Web-scale	File Site-scale	Block Application-scale
Price	$	$$	$$$
Data-type	Semi + Unstructured	Structured, Semi, Unstructured	Structured
Scales	Billions of Objects and Multi-Exabytes	Millions of Files and 1 Petabyte (ish)	Thousands of files and many TB
Protocols	HTTP REST APIs (e.g. S3, OpenStack Swift)	File Protocols (e.g. CIFS/NFS)	Block protocols (e.g. iSCSI, FC)
Optimized for	• Capacity • Scalability • Eventual consistency • Web accessibility • Broad geo distribution	• Fast file-sharing • Fast file-serving • File-locking • Canonical true files	• Fast random lookups • Reads and writes on small records • Storage for hypervisors
Typical apps	Everything under “File” + gene sequences, video, log files, photos, web content, large data sets	Office automation, design automation, collaborative engineering, word processing docs, presentation graphics	Transactional apps such as ERP, CRM, databases
Approach	An object = a File + all associated metadata + a globally unique identifier	File is stored in directory structure. Limited metadata stored in the file system itself (separate from file)	File is written in “blocks” on spinning media

What Object Storage is Not

- Distributed File System
  - Does not provide POSIX file system API support
- Relational Database
  - Does not support ACID semantics



- NoSQL Data Store
  - Not built on the Key-Value/Document/Column-Family model

- Block Storage System
  - Does not provide block-level storage service

Not a "One Size Fits All" Storage Solution

Examples of Object Storage

Object Storage Services:

- Amazon S3
- Google Cloud Storage
- Microsoft Azure
- Rackspace Cloud Files
- HP Cloud Object Storage
- IBM Bluemix Object Storage
- Oracle Cloud Storage
- OVH Cloud Storage
- ...

Open Source systems:

- OpenStack Object Storage ("Swift")
- Ceph
- Minio

Slide adapted from Albert Chen

(Some) Public Object Storage service regions

Publicly available Object Storage service regions of 10 global cloud providers (early 2020)

OpenStack Swift

OpenStack Swift

- Core OpenStack Service
  - One of the original 2 projects
  - 100% Python
  - ~ 40K LOC application, > 80K LOC test code

Top contributing companies include: SwiftStack, Intel, RedHat, IBM, HP, Rackspace, Hitachi, Fujitsu

Slide adapted from Paul Luse

OpenStack Swift & Swift API

Swift API accessible via HTTP

- Applications can consume storage from anywhere
- Larger range of functionality
- Puts the developer in control

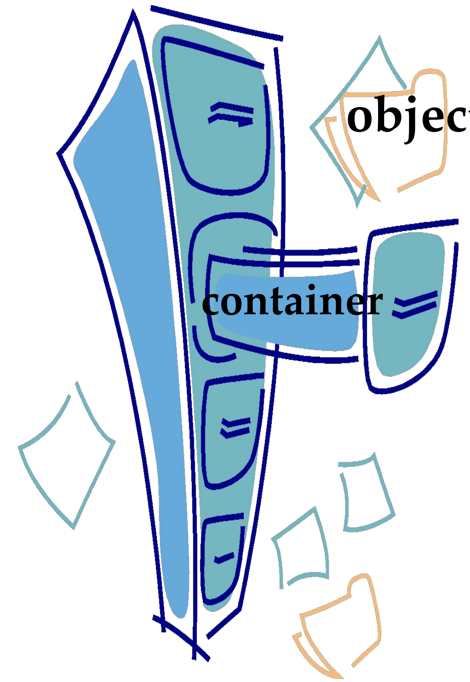
Swift Object Storage superior to Traditional Storage

- Designed to scale from TB -> PB -> EB
- Replication is automatic across Nodes, Zones, and Regions
- Supports both Replicas & Erasure Coding
- No Single Points of Failure -> Lose Nodes, Racks or Data Centers
- Ingress / Egress data from all Proxies - No Masters
- Balance Heterogeneous Commodity Hardware

Slide adapted from Albert Chen

OpenStack Swift Overview

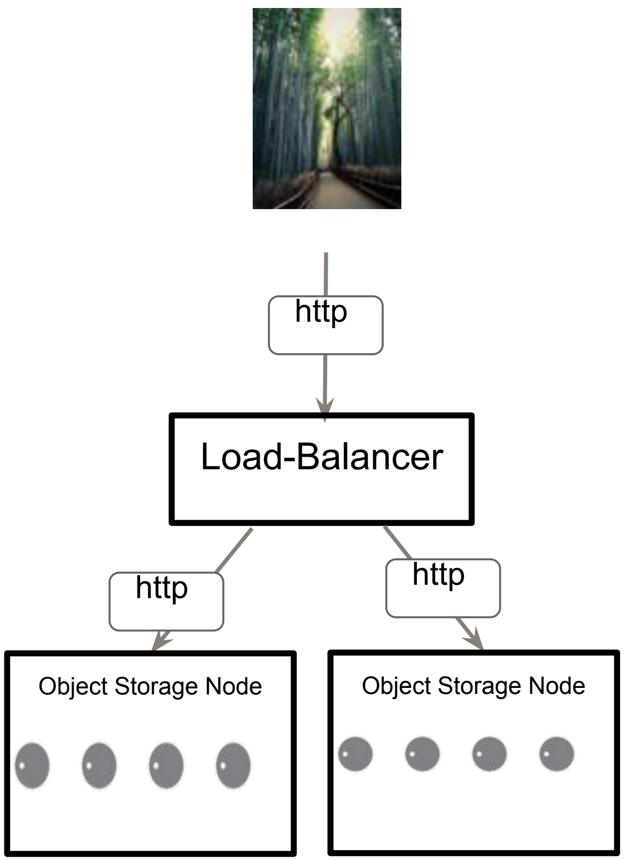
- Uses container model for grouping objects with like characteristics
  - Objects are identified by their paths and have user-defined metadata associated with them
- Accessed via RESTful interface
  - GET, PUT, DELETE
- Built upon standard hardware and highly scalable
  - Cost effective, efficient
- Eventually consistent
  - Designed for availability, partition tolerance



Objects are organized with containers

Slide adapted from Paul Luse

Access Method: RESTful HTTP API



There are also API operations for Accounts and Containers

Object Storage API Operations for Objects:

Method	Description
GET	Downloads an object with its metadata
PUT	Creates new object with specified data content and metadata
COPY	Copies an object to another object
DELETE	Deletes an object
HEAD	Shows object metadata
POST	Creates or updates object metadata

Every Object Has a URL

Base URL

API Version

http://example.com/v1/account/container/object

- Each account has its own URL
- Swift is multi-tenant

- Namespaces used to group objects within an account
- **Containers** are unlimited
- **Like folders, but can't nest them**

- Each object is addressed as a URL
- Users name the object
- **Objects are not organized based on hierarchy**
- Instead, object names may contain "/", so pseudo-nested directories are possible

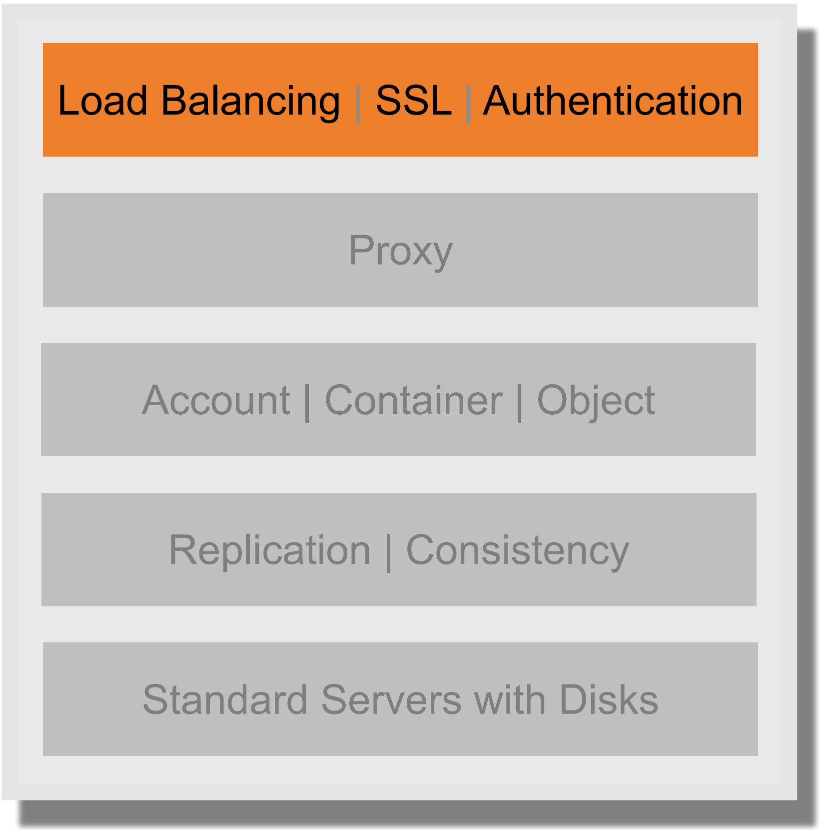
Slide adapted from Albert Chen

Swift High-Level Architecture

- Load Balancing/Authentication
- Proxy
- Account / Container / Object
- Replication and Consistency
- Standard Hardware

Slide adapted from Albert Chen

Swift Architecture



Load Balancing

- Requests are load balanced across all nodes running proxy server processes

SSL

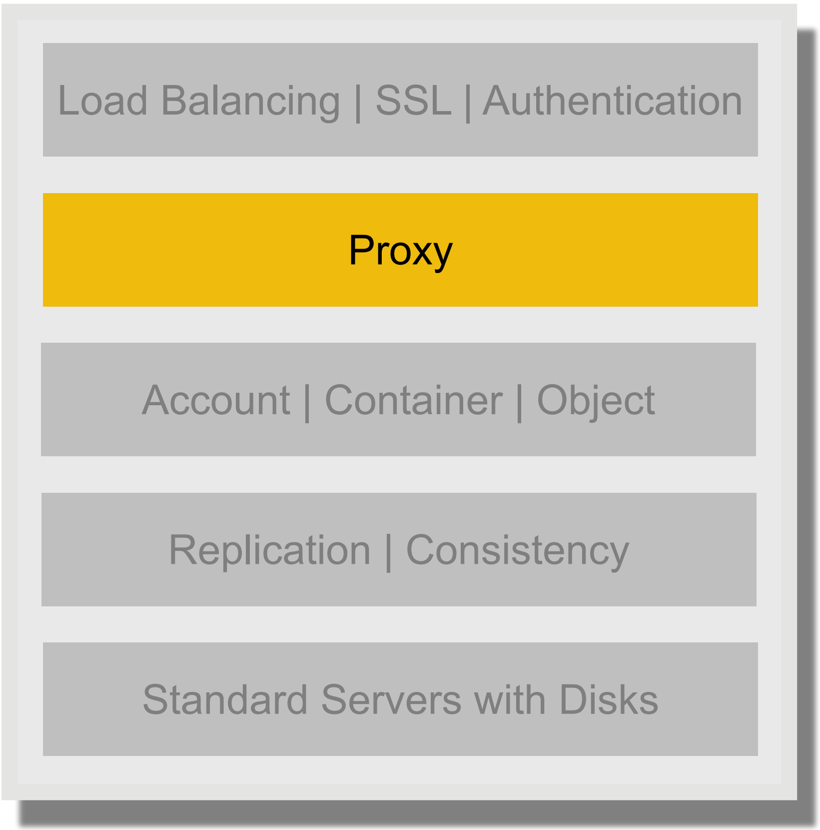
- Optionally, SSL termination can be enabled to encrypt data in flight

Authentication

- OpenStack Swift has a pluggable authentication system
- Modules include API/UI-driven, LDAP, AD, Keystone

Slide adapted from Albert Chen

Swift Architecture

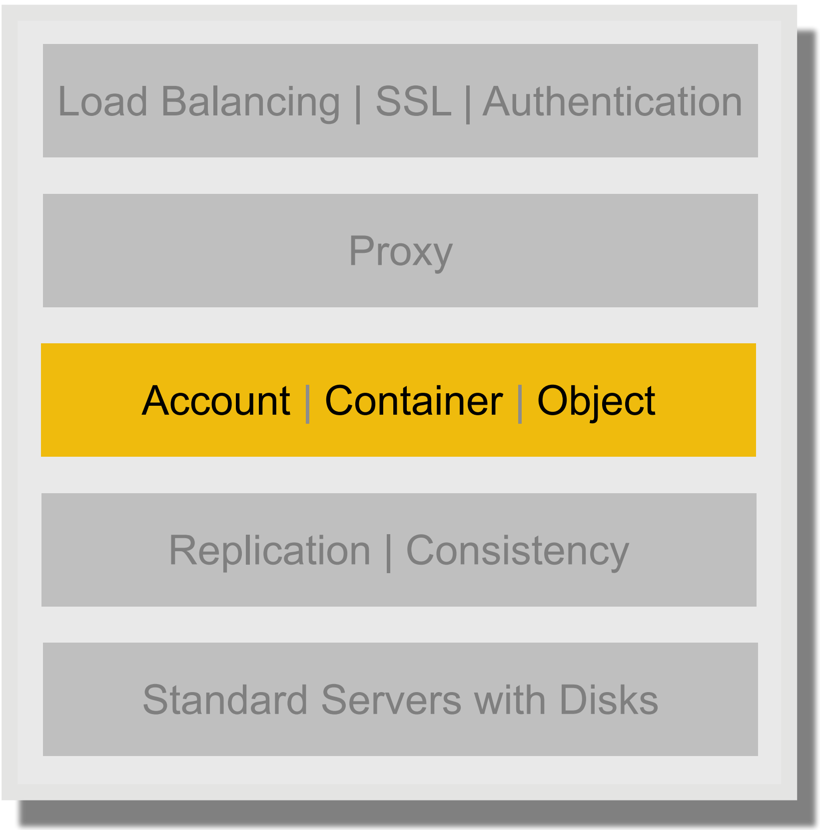


Proxy

- Only part of the cluster that “talks” to external clients
- Primarily with HTTP RESTful Swift API
- Routes requests from clients to disk
- Three replicas are simultaneously written
- Quorum required
- Uses single replica for reads
- Routes around failures
- Enforces ACLs set by user

Slide adapted from Albert Chen

Swift Architecture



Account / Container

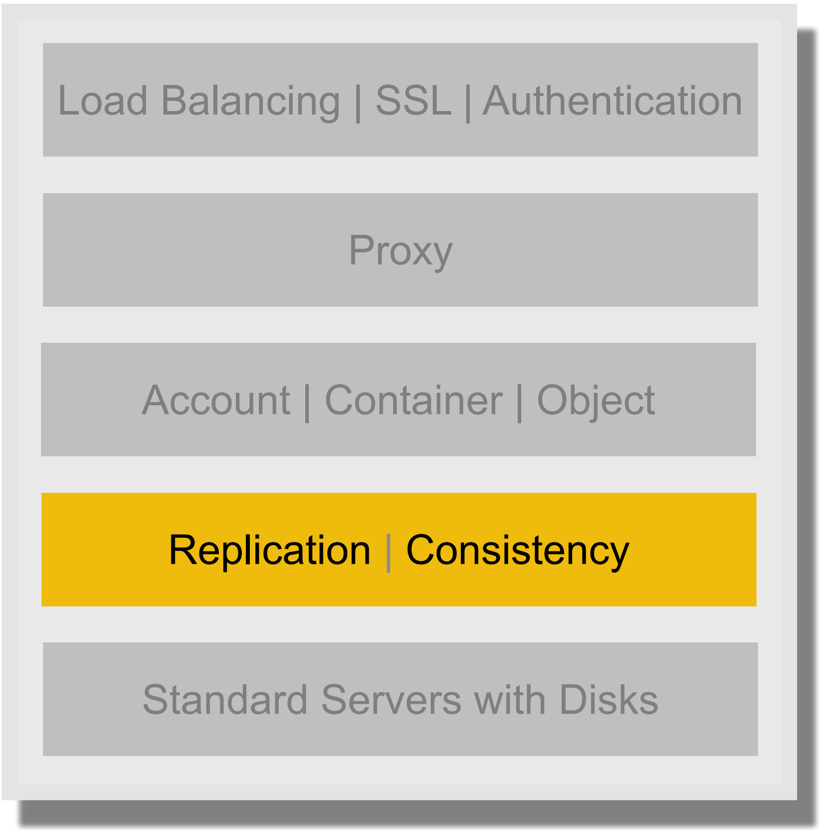
- Accounts keep records of containers
- Containers keep records of objects

Object

- The object servers store the data on disk
- Metadata is stored with the data
- Uses standard filesystem (XFS)

Slide adapted from Albert Chen

Swift Architecture



Replication

- Constantly checking for replicas status
- Only updates other replica sites, does not pull in newer versions of objects

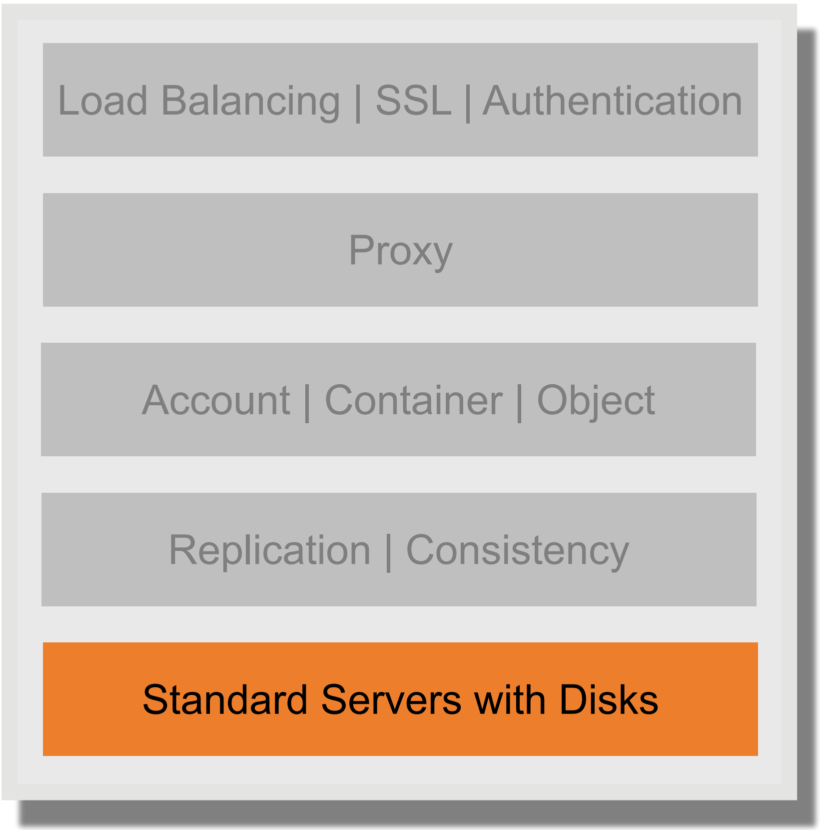
Consistency

- Constantly 'scrubbing' data to check for bad data

Note: These processes run in the background on Nodes where account, container, or object server processes are running.

Slide adapted from Albert Chen

Swift Architecture

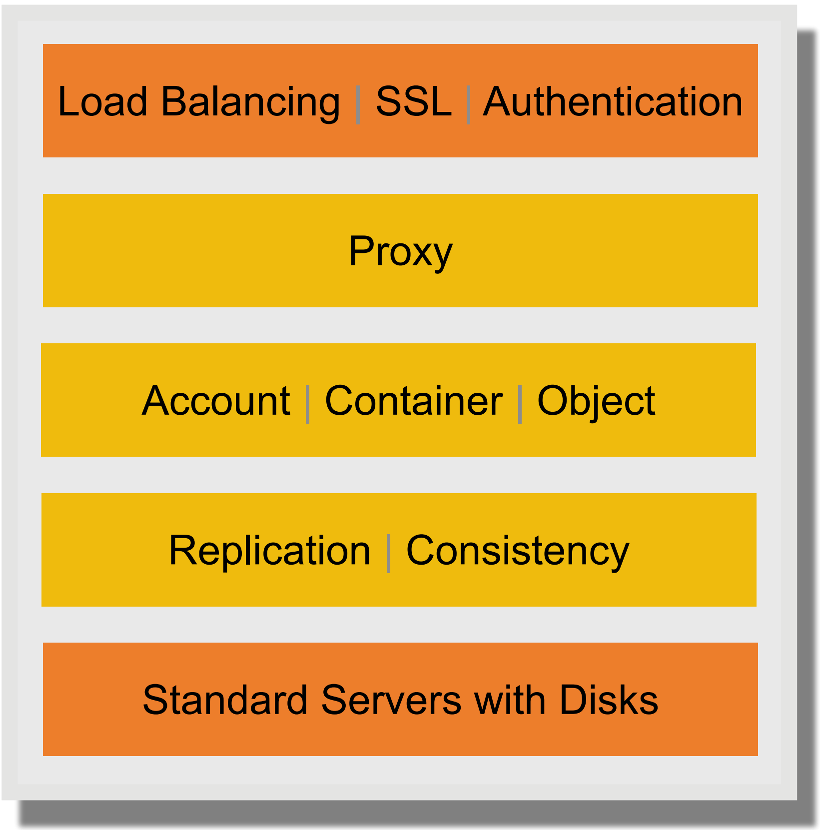


Standard Servers

- Runs on standard server hardware
- SATA or SAS disks
- No RAID
  - Visibility to hardware beneath
  - Swift is already providing data redundancy
  - Reduce cost

Slide adapted from Albert Chen

Swift Architecture Summary



- Native HTTP API
- Scales Linearly
- No Single-point of Failure
- Standard Servers and Linux
- Extremely Durable
- Resilient to hardware failures
- Routes around network failures
- Consistency Model Enables Multi-Data Center

Slide adapted from Albert Chen

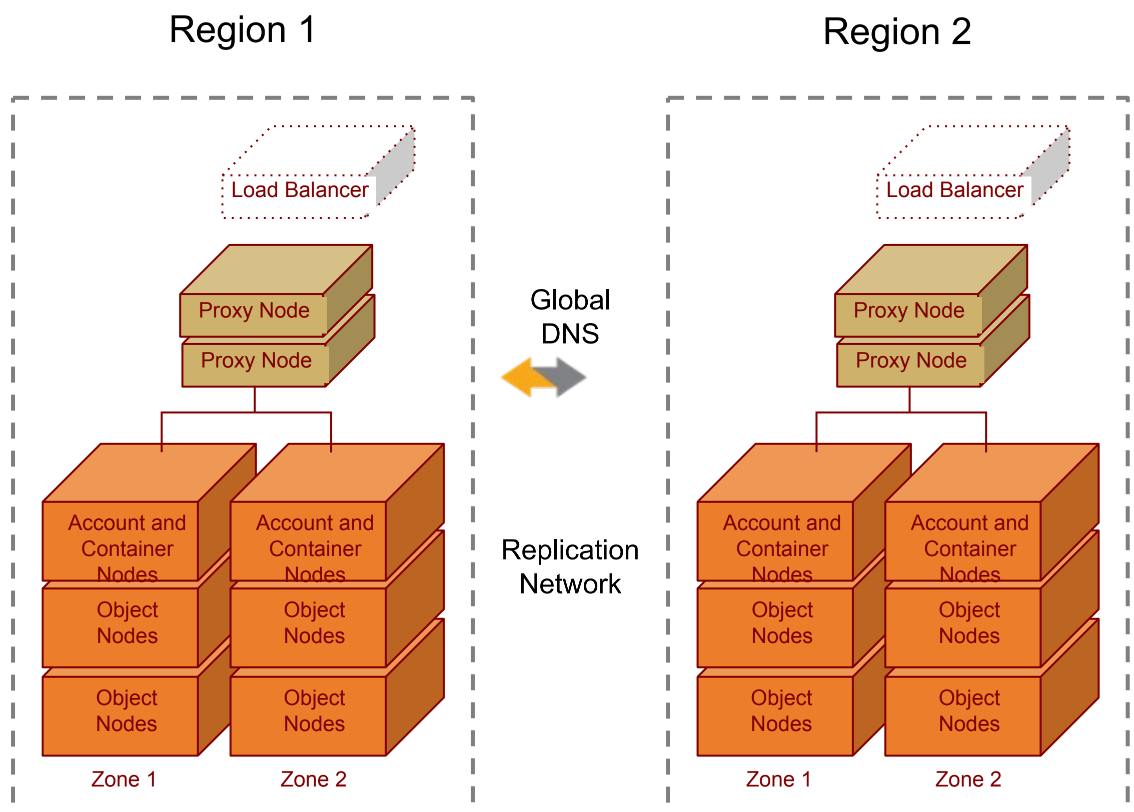
OpenStack Swift

Multi-Region and
Global Clusters

Regions and Zones

Regions

- Physically separate, often defined by geographical boundaries
- Minimum: 1 Region

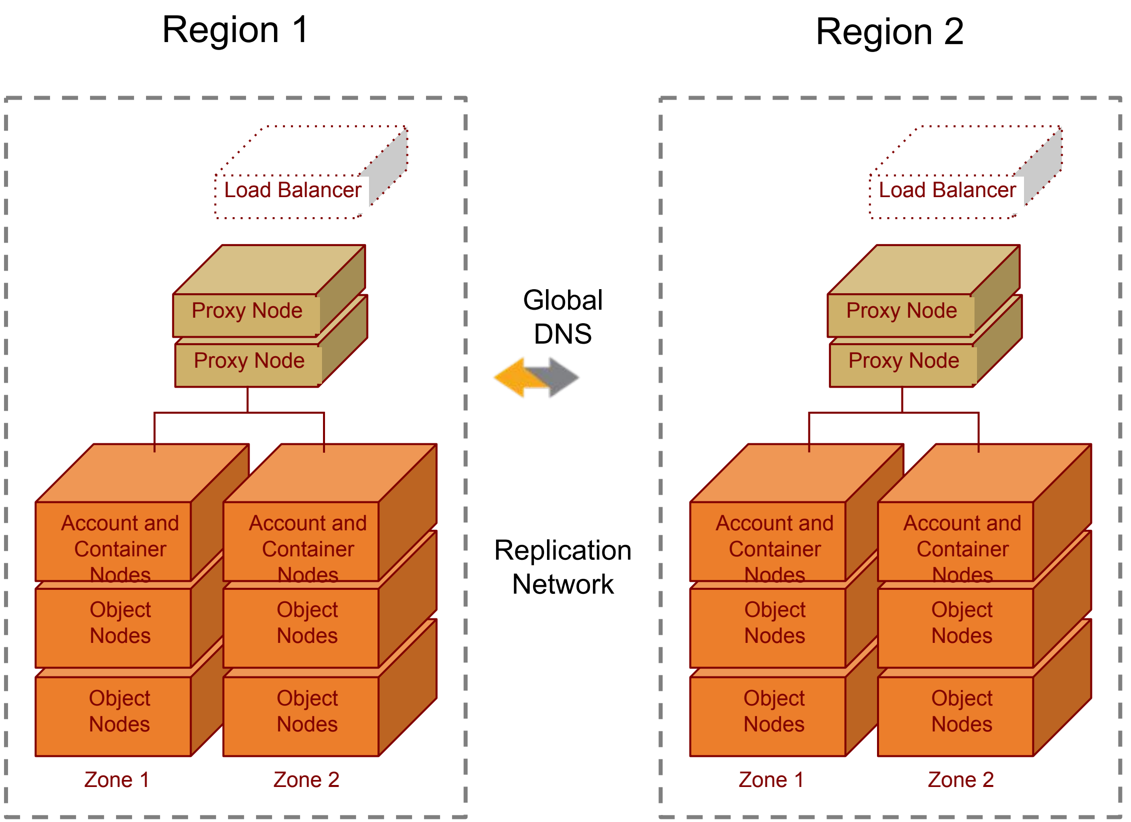


Slide adapted from Albert Chen

Regions and Zones

Zones

- Regions contain one or more zones
- Designate a group of nodes sharing a set of physical hardware
- Also referred to as failure domains

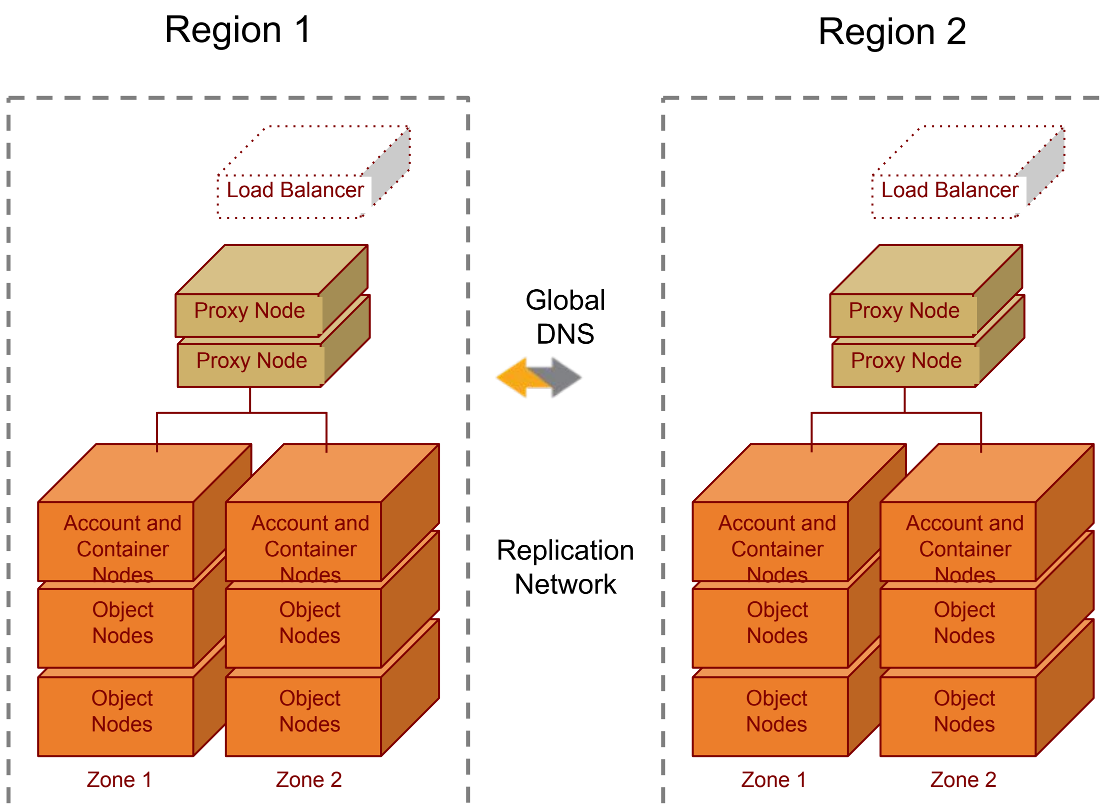


Slide adapted from Albert Chen

Regions and Zones

Multi-Region Clusters

- Goal: Tolerate failures across regions and zones
- Policies control placement across regions
- Send requests to "closest" region

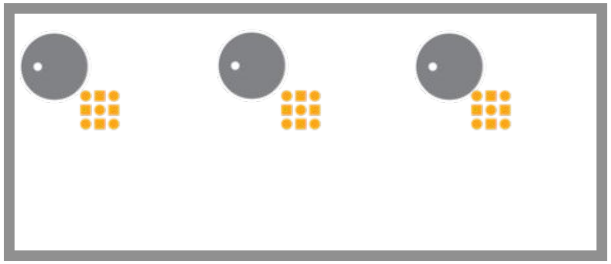


Slide adapted from Albert Chen

Data Placement: “Unique as Possible”

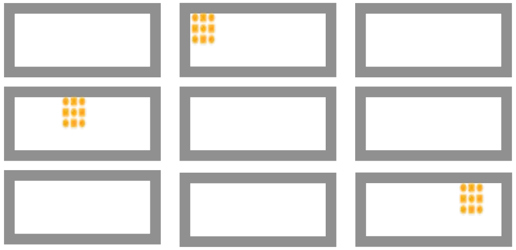
Single Node Cluster

Disks are “as-unique-as-possible”



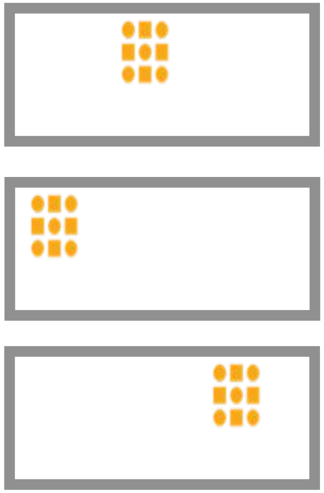
Large Cluster

Storage Racks are “as-unique-as-possible”



Small Cluster

Storage Nodes are “as-unique-as-possible”



Multi-Region

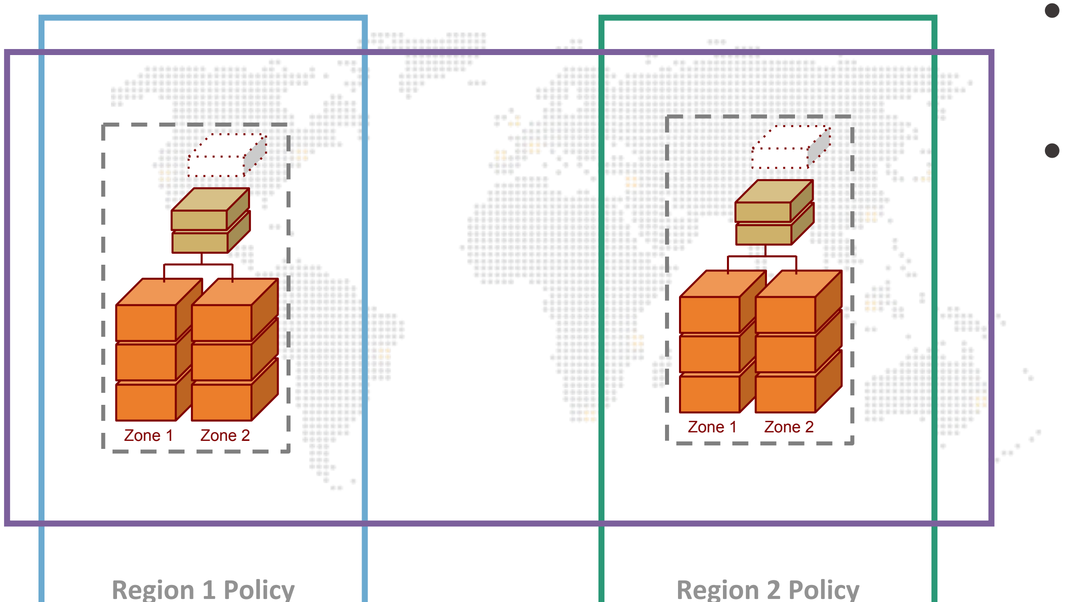
Distributed data centers are “as-unique-as-possible”

Slide adapted from Albert Chen

Storage Policies

Benefits

- Optimizes storage for applications and users
- Consolidates storage tiers under one system
- Simplifies management
- Lowers storage infrastructure TCO



Policies Encompass:

- Storage media
- Number of replicas
- Erasure Codes / Replicas

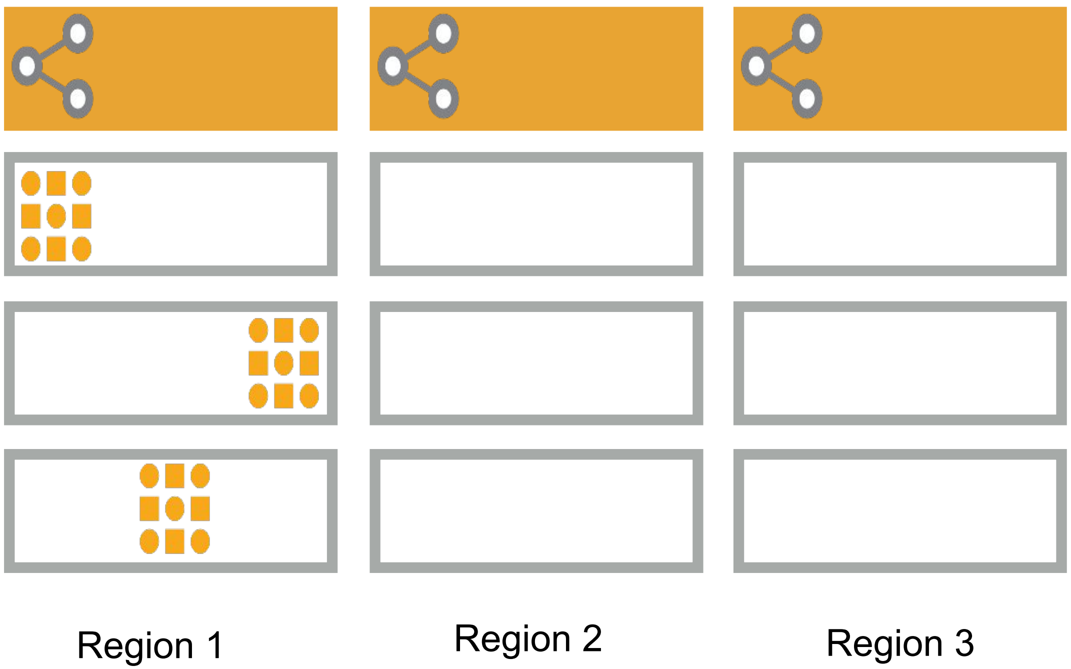
Common Policy Groupings:

- According to performance
- According to geography or political boundary
- According to data protection scheme, e.g., number of replicas or erasure coding

Slide adapted from Albert Chen

MRC: Write Affinity

Proxy Write Affinity

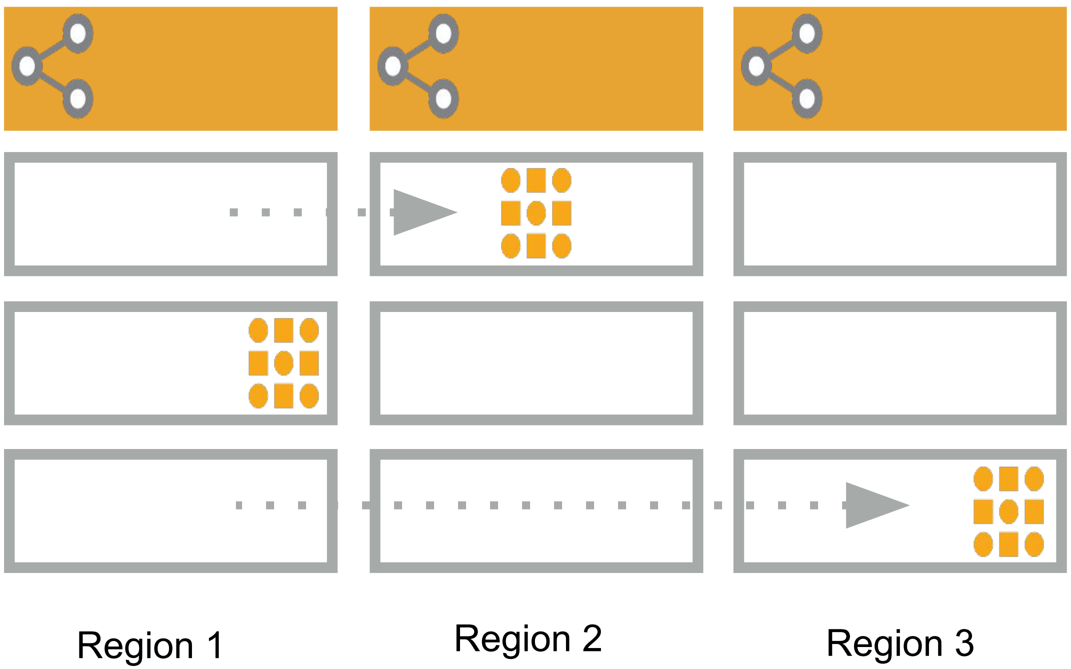


- By default, **objects are written** to **all the locations simultaneously**
- **Write Affinity:** writes all copies locally then transfer asynchronously to other regions

Slide adapted from Albert Chen

MRC: Write Affinity

Proxy Write Affinity



- By default, **objects are written** to **all the locations simultaneously**
- **Write Affinity:** writes all copies locally then transfer asynchronously to other regions

Slide adapted from Albert Chen

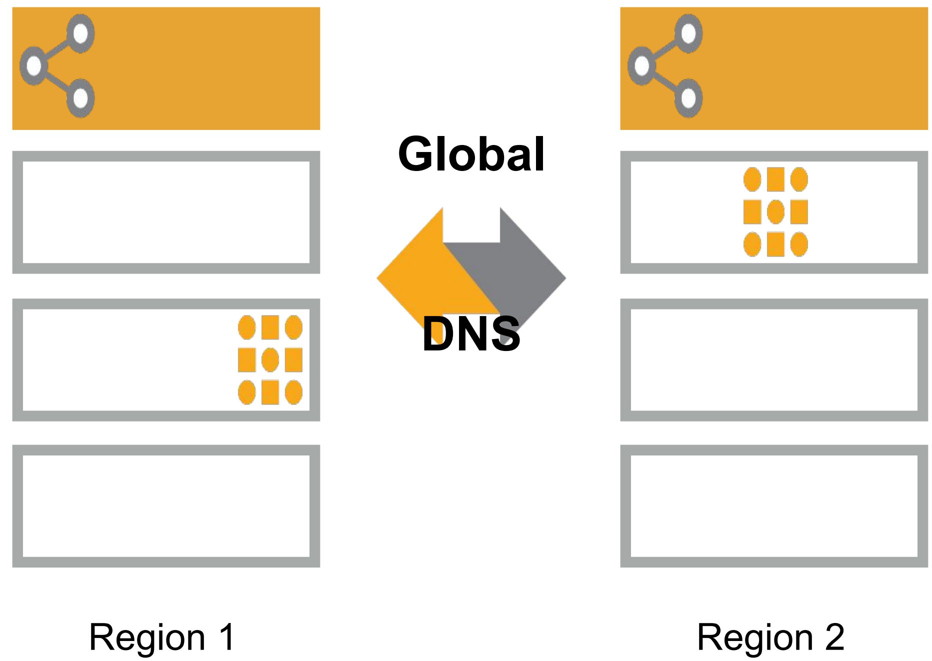
MRC: Read Affinity

Prioritizes "Nearby"

- Zone/Region, where am I?
- Latency to Storage Node

DNS Routes User

- Each proxy pool has its own hostname
- Routes user to closest region



Slide adapted from Albert Chen

Use Case – Backup

Backup

- Enterprise: Workstations
- MSP: Cloud backup service

Behavior

- Write optimized
- High throughput
- Low concurrency

Slide adapted from Albert Chen

Use Case – Big Data / MapReduce

Hadoop / Spark

Philosophy: Let HDFS do what it's best at: Serving data where you want it when you need it

Swift for warm and cold data storage. Why?

- Durability and reliability guarantees
- Managed capacity: Grow to and beyond petabytes
- Easier integration with various data input sources
- Share results using a common storage platform

Example: Run MapReduce job

- Read input data from SwiftStack
- Write transient results to HDFS
- Write result to Swift

Slide adapted from Albert Chen