Merritt Cloud API

July 27, 2017

David Loy

Use

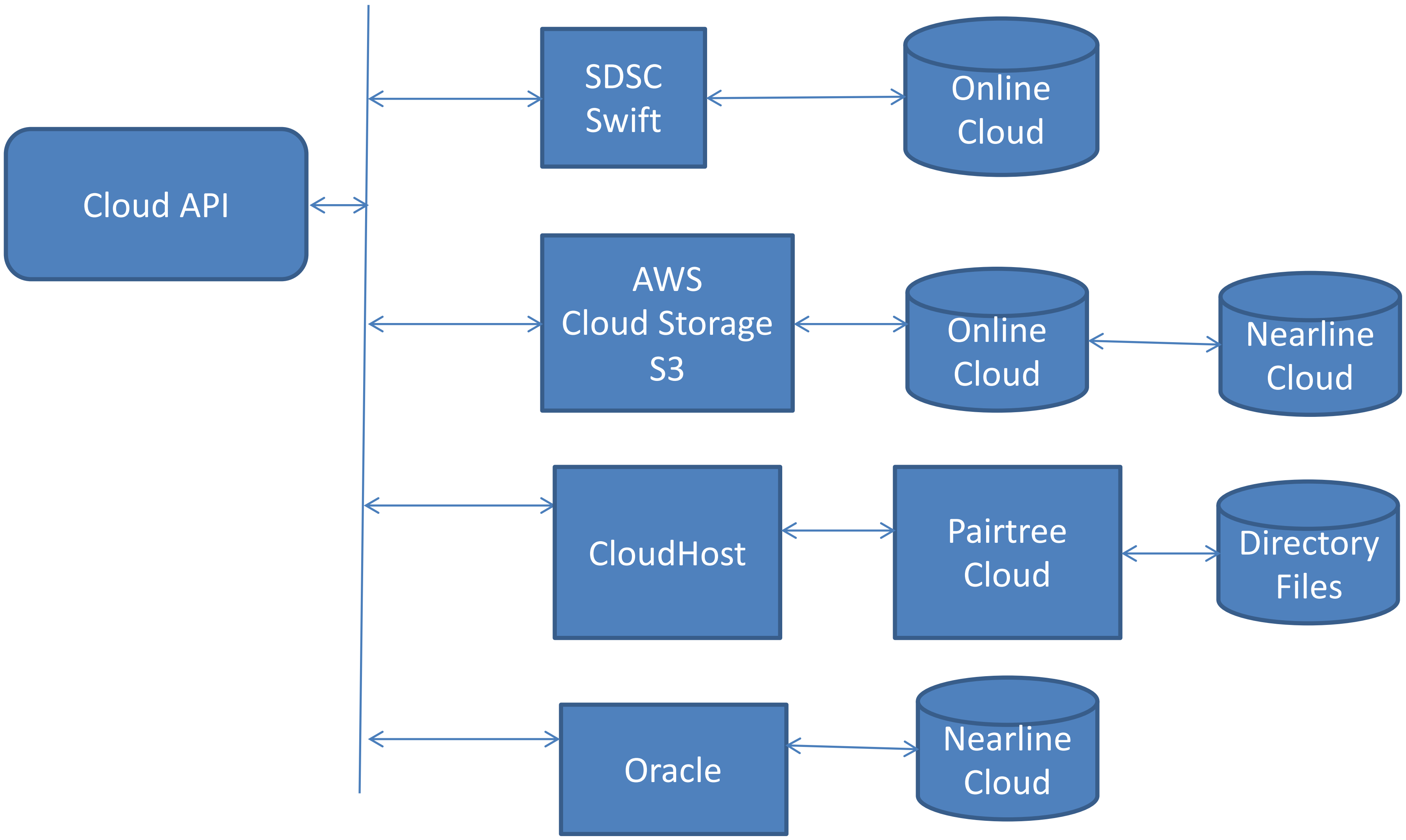
- Used in Storage, Replication and Audit
- Allowed to access, update, and delete content across different cloud providers
- Online accessible vs. nearline accessible

Cloud providers

- SDSC Swift - online
- AWS S3 - online
- AWS S3 <> Glacier

  online <> nearline
- Oracle archive - nearline
- Cloudhost - online
- Pairtree cloud - online

Clouds


Cloud API
SDSC Swift
Online Cloud
AWS Cloud Storage S3
Online Cloud
Nearline Cloud
CloudHost
Pairtree Cloud
Directory Files
Oracle
Nearline Cloud

Basic functions

- Retrieve content
- Upload content
- Delete content
- List content (not pairtree cloud or cloudhost)
- Restore content (nearline)
- Audit (cloudhost)

Access hierarchy

- Cloud service
- Cloud container (swift) / bucket (aws)
- Key

Note: AWS added a key prefix to simulate a directory base – we do not use

Cloud Storage considerations

- Retry logic required on almost all commands
- Completion doesn’t necessarily mean that content is available (swift delay between POST and GET)
- Containers have different storage requirements:
  - Swift – restricted number files per container unlimited containers
  - AWS – unrestricted files per container limited buckets
- File versioning may be allowed

Supported APIs

- SDSC swift – low level API written for Merritt
- AWS storage cloud
  - high level – automatically handles large content
  - Low level
- OracleTransfer
  - high level – automatically handles large content
  - Low level
- Pairtree cloud (local) and Cloudhost (remote)

SDSC Swift - Features

- Uses cloud storage model:

  service - container-key
- Cloud storage commands:

  Post, Get, Delete, Metadata, List
- One site
- Use Static Large Objects (SLO) with X-Object-Manifest for upload that requires manifest control by user
- Dynamic Large Objects (DLO was not supported

SDSC Swift – Merritt Authentication

- access_key=user
- secret_key=password
- host=cloud.sdsc.edu

SDSC Swift - Issues

- We were one of the first users of SDSC Swift
- Problems
  - Specific tests they used on storage destroyed content
  - Their S3 support claim did not support large content
  - Failure rates high – 20%
- Generally wary but they now have competent support

AWS Simple Storage Service (S3) Standard

- Uses cloud storage model:

  service - bucket – key
- Cloud storage commands:

  Post, Get, Delete, Metadata, List
- Multiple locations world wide
- Provides different levels of storage classes:
  - Standard
  - Reduced redundancy – used with public content
- S3 Mechanism for Large records

AWS S3 Glacier

- Combines Glacier with standard S3 handling
- Uses expiration time of content for migration from standard S3 bucket to Glacier
- Migrated content must be restored to be accessible
- Metadata always accessible
- Provides Glacier specific properties for determining status restore

FYI S3 Glacier properties

- storageClass – indicates if Glacier or not
- ongoingRestore – restore action taking place
- expiration – content still in S3 – not migrated

AWS S3 – Merritt Authentication

- Based on server profile property

AWS S3 - Support

- We use Glacier storage class for secondary archive content
- We use standard storage class on S3 before expiration to Glacier
- We use Reduced Redundancy on S3 for primary public content
- 2 java APIs used – one high level (large record) – one low

AWS S3 Issues

- Has typical Cloud issues - retry but not as bad as SDSC
- Statistics about stability of cloud content not confirmed!
- StorageClass specific to AWS and volatile

Oracle (no longer supported)

- Trial system to evaluate whether to use
- Based on Swift
- Nearline content
- 2 java APIs for large and small records
- Uses cloud storage model:

  service - bucket – key
- Locally defined API for handling remote – nearline not supported by Swift

Pairtree cloud

- Local simulation of cloud handling to minimize content movement in directory environment
- Uses directory base as container
- Used with UCLA directory
- Uses cloud storage model:

  service - bucket – key
- Cloud storage commands:

  Post, Get, Delete, Metadata, Audit
- No authentication

Pairtree example

- ./ar/k+/=9/01/35/=q/1d/v1/gt/9/ark+=90135=q1dv1gt9/1/system/mrt-object-map.ttl/component.txt
- ./ar/k+/=9/01/35/=q/1d/v1/gt/9/ark+=90135=q1dv1gt9/1/system/mrt-object-map.ttl/component.properties

Pairtree component.properties example

- key: ark:/90135/q1dv1gt9|1|system/mrt-object-map.ttl
- size: 4583
- sha256: b6925b1f29483b295ce18c630bd3df468e7c4919dc302865ab1b9d558145cc8d
- digesttype: md5
- digest: f90c37f565d22158b9a013a8a26c8c16
- update: 2017-07-14T15:36:05Z
- bucket: ./fileClouddpr2store@uc3-mrtstore2-dev:~/unm/prod/fileCloud$

Pairtree cloud Issues

- May only be used on locally accessible directory
- With UCLA control returned before content available on NFS mount – retry required

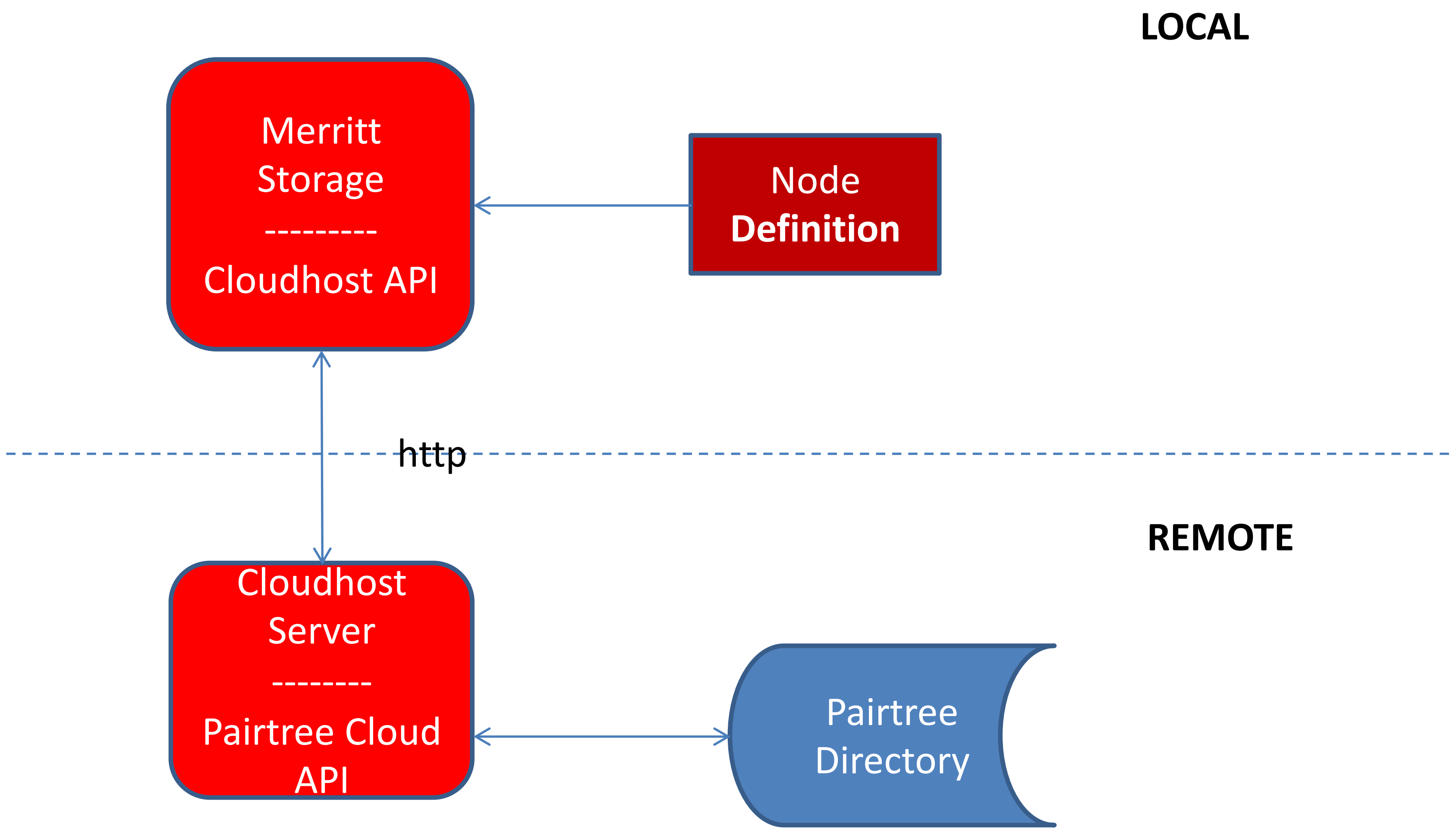
Cloudhost

- Uses cloud storage model:

  service - bucket – key
- Cloud storage commands:

  Post, Get, Delete, Metadata, Audit, State
- Executable as jar or tomcat war
- Servlet with pairtree cloud on backend

Cloudhost architecture



Cloudhost Issues

- Typical HTTP restrictions of open port
- Since locally directory based needs specifically open read-write directory
- Current jar includes all cloud APIs

Moving Forward

- For Swift replace my API with 3rd party if one exists
  - Important to have a more generic version that can be customized for different swift variations
  - Try Oracle version with SDSC
- With cloudhost find mechanism to build jar without other cloud APIs attached
- Add audit to all Cloud functions

Changes hg to git

- Split repository to mrt-cloud and mrt-conf-prv/s3-conf
- Modification of jar names specific to prv
- Drop old test repositories except mrt-test/s3-test
- Modification builds: builds-s3