Using Cloud Services behind SGI DMF

Greg Banks <gbanks@sgi.com>
Principal Engineer, Storage SW

Overview

- Cloud Storage
- SGI Objectstore
- Design
- Features & Non-Features
- Future Directions

Cloud Storage

What is Cloud Storage?

- Object based
  - Not file or block
  - Keyed blob
  - Typically a few MB to a few GB
- Distributed
- Non-RAID data protection
  - Replication, or
  - Erasure codes

What is Cloud Storage? (2)

- Access via RESTful protocol
  - Layered over HTTP
  - HTTP methods are verbs (GET/PUT/DELETE)
  - URIs encode nouns
- Runs on commodity hardware
- Software only solutions

Public Cloud

- Run by a vendor as a service
  - Storage As A Service
  - Less hassle for you
- Shared
  - Infrastructure
  - Expertise
- Charges
  - per byte input, output, stored

Public Cloud: Pros & Cons

- Easy to change
- Accessed over the Internet
  - You want on-the-wire encryption
- Your data in some else's datacentre
  - Possibly in another jurisdiction
  - You want at-rest encryption
  - The Nirvanix contingency

Private Cloud

- Run by yourself in-house
  - More hassle
  - More expense upfront
- More control
  - Cost, performance, privacy
- Accessed over your own network
- No shared infrastructure
  - Have to plan and pay for peak capacity

Examples: Amazon S3

- S3 = Simple Storage Service
- http://aws.amazon.com/s3/
- Public commercial $Cloud
- Grand-daddy of them all

Example: Openstack Swift

- http://www.openstack.org/
- Open source, in Python
- Multi-vendor
  - Rackspace, Dell, HP, IBM, RedHat
- Public or private, you choose

Example: Scality

- Private cloud storage, $commercial
- Scales well: many objects per bucket
- Many protocol connectors
- http://www.scality.com/

Examples: Eucalyptus Walrus

- Private cloud
- Open source+, in Java and C
- http://www.eucalyptus.com/
- Designed for maximal Amazon compatibility

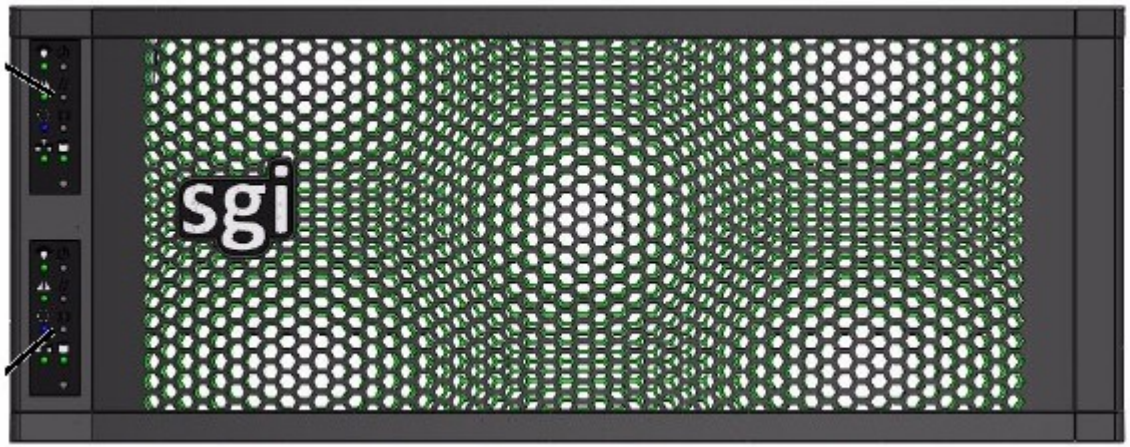
Some Other Examples

- Cleversafe
  - http://www.cleversafe.com/
- Amplidata
  - http://www.amplidata.com/
- CDMI
  - A “standard” protocol from SNIA
- Microsoft Windows Azure Blob Service
  - http://www.windowsazure.com/

SGI Objectstore

SGI Objectstore

- SGI Objectstore = Scality object store on SGI MIS Server hardware

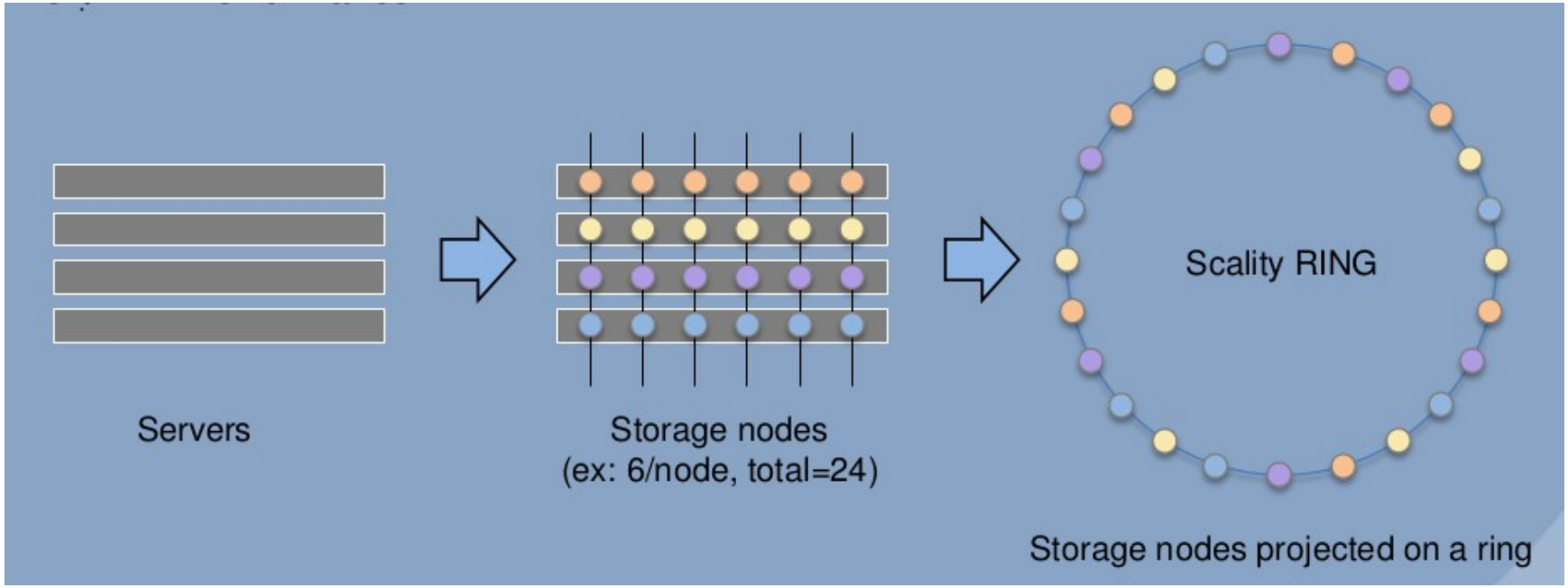


Scality: CHORD

- The RING uses the CHORD peer-to-peer protocol, developed at M.I.T., to achieve:
  - Decentralization – the nodes collectively form the system without any central coordination, avoiding all bottlenecks and single points of failure.
  - Scalability – the system scales efficiently from a dozen to thousands of nodes, all the while maintaining internode communication and evenly distributing the load among all the nodes.
  - Fault tolerance – the system stays reliable even with nodes continuously joining, leaving, and failing. Consistent hashing guarantees that only a small subset of keys are affected by a node failure.

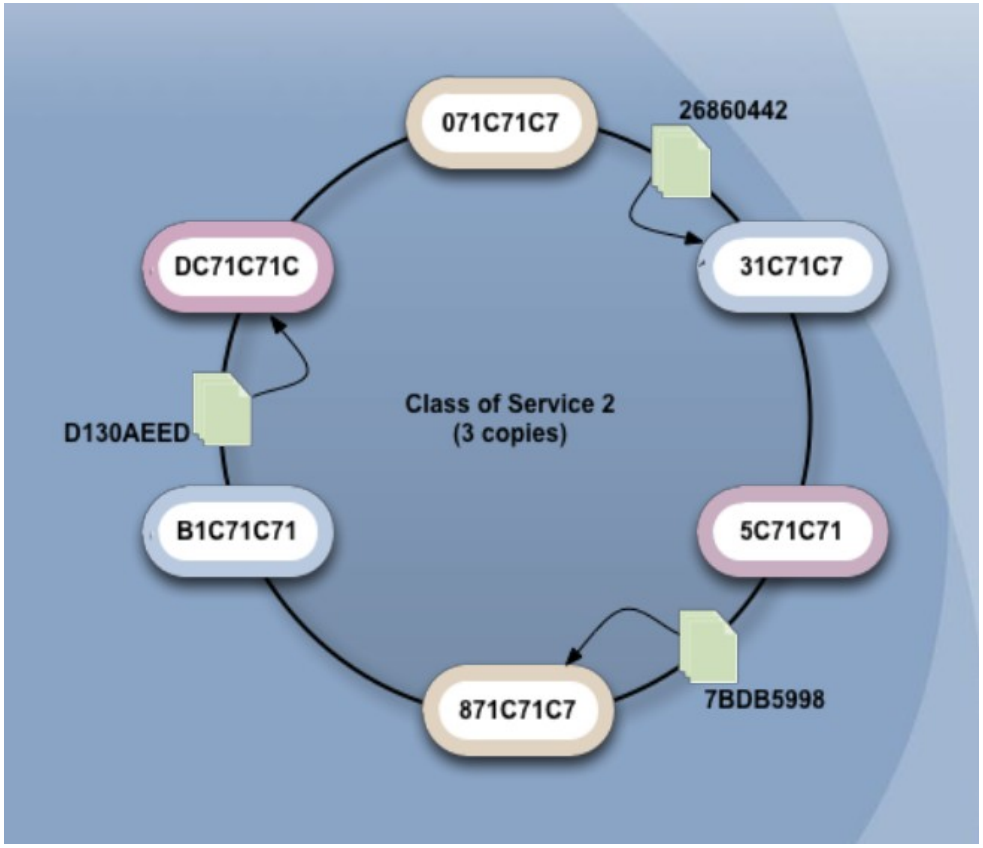
Scality: Making a RING

- A “node” is a server process + disk
  - Many nodes per physical server
- Nodes form into a logical RING



Scality: Key Space

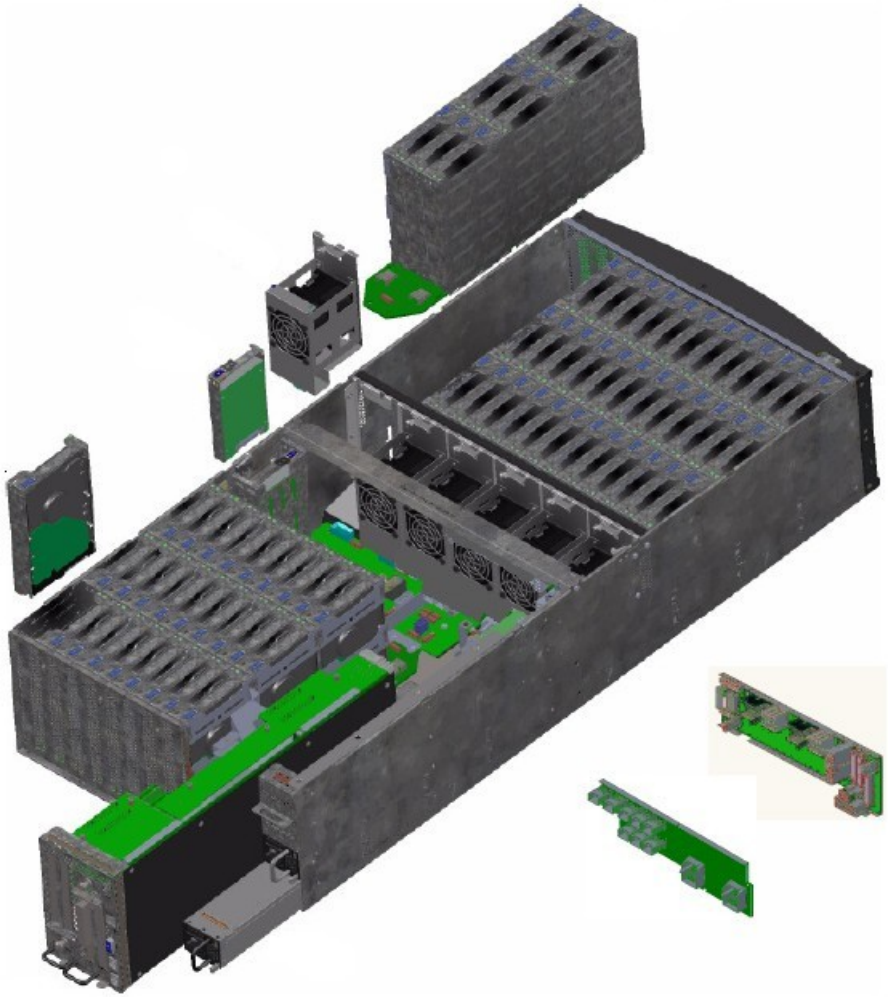
- 160 bit binary key maps to positions on the RING
- Objects are replicated on different servers
  - Replica keys are calculate and do not need to be stored
- Class of service between 0 and 5 replicas per object
- Self-healing
  - Balances misplaced objects
  - Rebuilds missing replicas



Base Unit: MIS 1.5 Dual Server

- 4U enclosure
- 72 disks, 2.5” or 3.5” HD or SSD
- Two server units, each:
  - 2 Xeon ES-2620 (6 cores, 12 threads, 2GHz)
  - 128 GB RAM
  - 2 mirrored boot disks
  - 10 GbE dual port NIC

In One Base Unit



- 288 TB
  - (4TB disks)
- 24 cores
- 100 kg
- 2.2 kW

In One Standard 19” 42U Rack

- 10 Units
- 2.88 PB
- 240 cores
- ~1 ton
- 22 kW

Testing by ESG

- “These numbers can be extrapolated to 56 GB/s of read throughput using replication mode and 16.6 GB/s of write throughput using erasure coding with a full rack of storage with 10 MIS Servers.”

- “The performance and data durability features of SGI's hardware and Scality's object store clearly demonstrate the maturing of object store technology and its readiness to handle enterprise-class requirements.”

Design

Design Goals

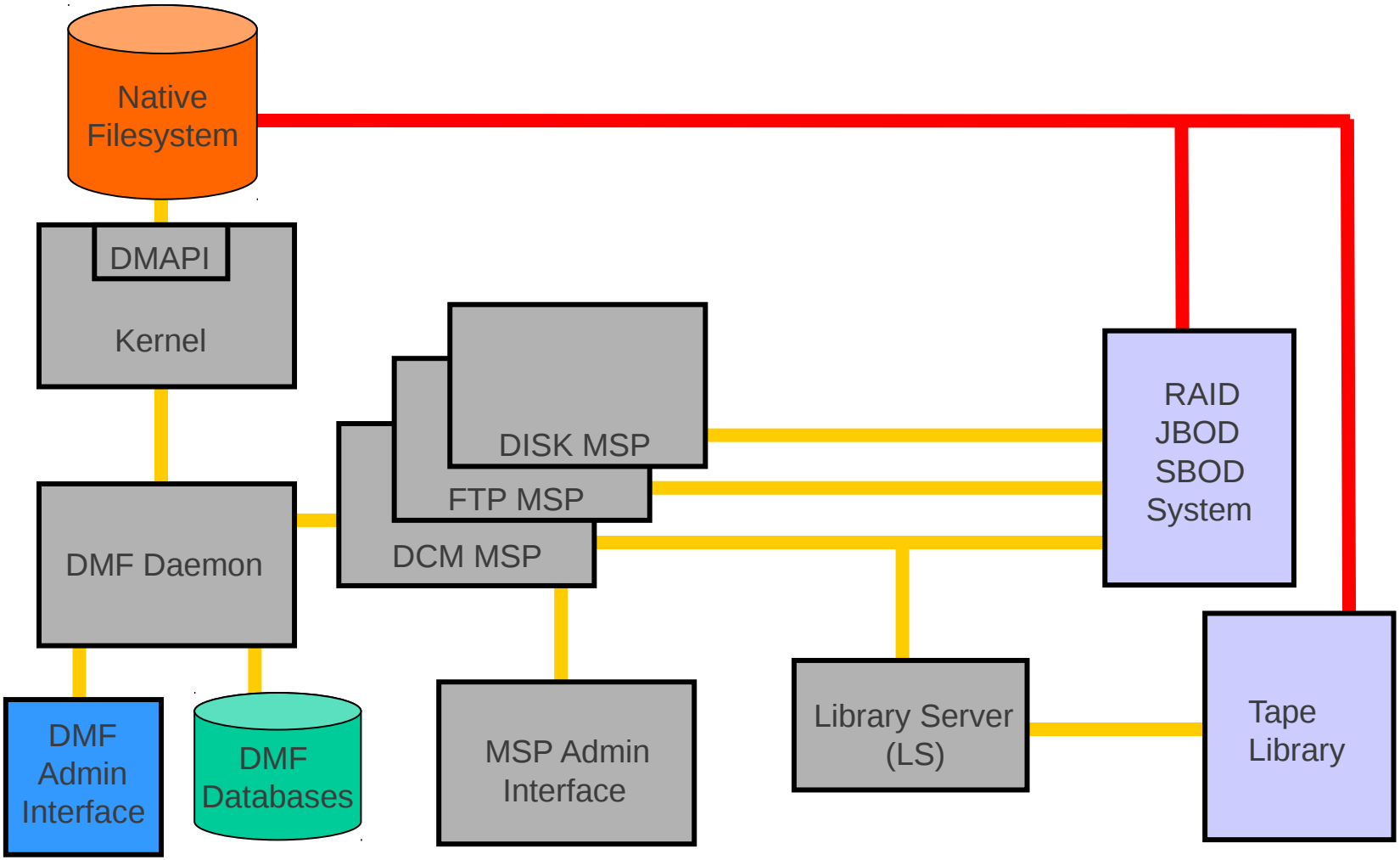
- Support SGI Objectstore behind DMF
  - More cloud providers later
  - Public cloud later
- Correctness first
  - performance later
- Availability
  - ISSP 3.2
  - patch back to ISSP 3.1

Architecture

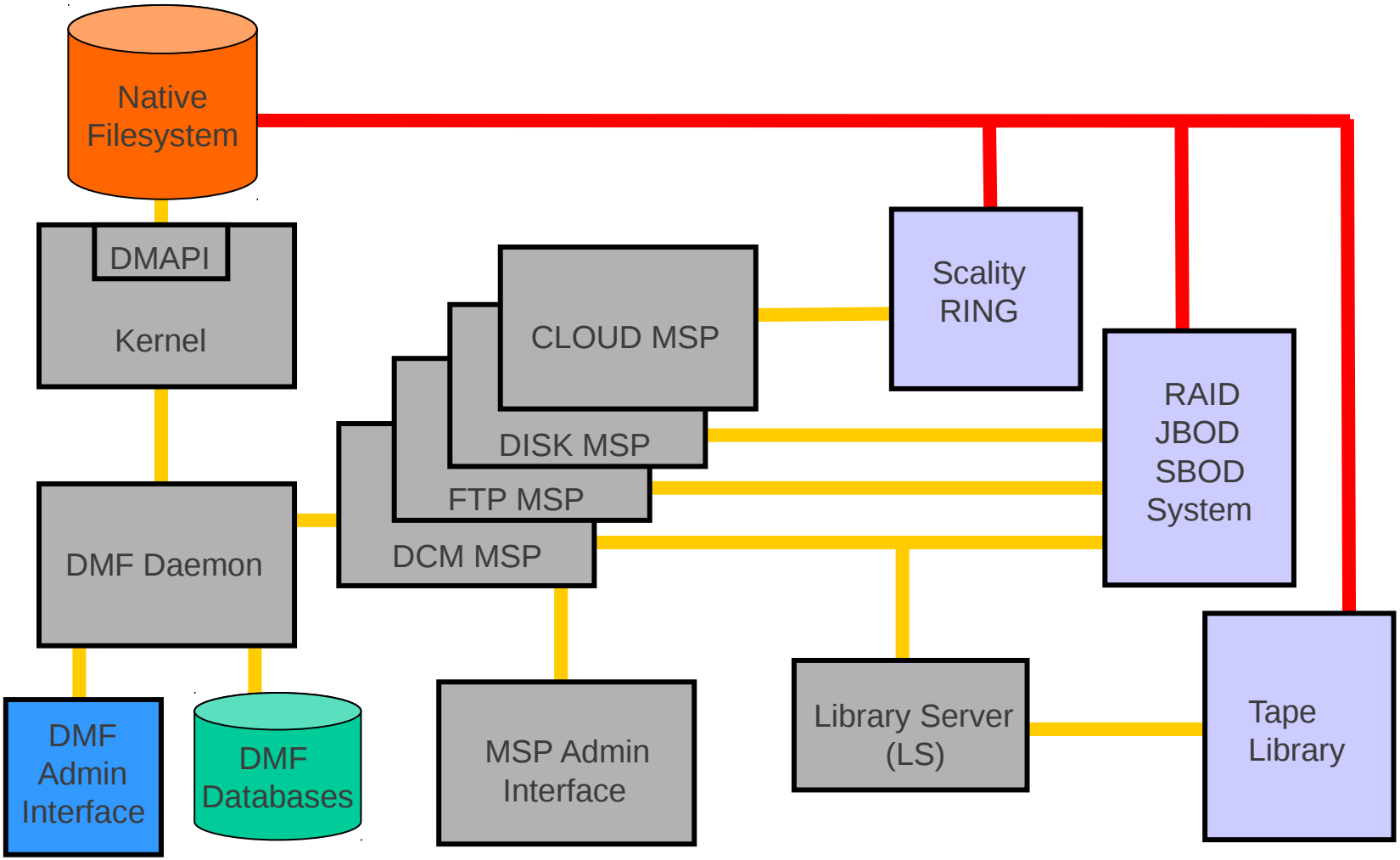
- Implemented as an MSP
- Based on FTP MSP
- Uses the open source Droplet library

  https://github.com/scality/Droplet/
- No additional DMF-side database

Architecture Diagram



Architecture Diagram



Internals: Write Flow

- Managed files formatted → media file
- Media file sliced into 1MiB chunks
- Each chunk has a RING key derived from the BFID and the chunk offset
- Each chunk is PUT as a RING object
- The first chunk has metadata attached
  - File size and the chunk size

Features & Non-Features

Features

- Configuration via dmf.conf
  - Manually
  - Using DMFMan
- Get, Put and Delete a file
- Logs to standard DMF MSP logs
- Provides PCP statistics
  - an MSP instance
- Single threaded

Non-Features

- On-the-wire / at-rest encryption
  - In v1, private cloud only so we assume this doesn't matter
- On-the-wire / at-rest compression
  - In v1 we assume you have enough network and disk capacity
- Deduplication
  - Later, maybe?

More Non-Features

- Cloud-side resource usage tracking
  - Providers usually have better tools anyway
- Resource usage optimisation
  - Requires significant scheduling brains which simply isn't present in the FTP MSP
- Client-side charging model
  - Difficult to do accurately, we don't want to provide inaccurate $numbers

Future Directions

ISSP 3.3

- Support for more providers
  - Amazon S3
  - OpenStack Swift
- Support for public clouds
  - On the wire encryption
- Scale up single node performance
  - Multiple threads & HTTP connections
  - HTTP pipelining

ISSP 3.4

- Audit feature
  - Verify cloud-side data without DMF state
- Scale out to multiple nodes
  - Parallelise the MSP model
  - Enables a PDMO-like architecture

Maybe Later??

- Support for more providers
  - Cleversafe
  - Amplidata
  - Amazon Glacier
- Disaster Recovery
  - Recover the managed filesystem from cloud data
- IPv6 support
- Cancel/reprioritize requests

Acknowledgements

- Ron Kerry
  - DMF Architecture Diagram
- Scality
  - text and diagrams about RING

Bits & Bobs

- TODO: people use cloud storage as an archival medium
- TODO: challenges of supporting Amazon Glacier
  - It fits the slow tape mounting model well
  - But the MSP model doesn't help with that
- TODO: Challenges of parallelising the MSP model

RESTful protocol

- Layered over HTTP
- HTTP is the new TCP

- HTTP methods are the verbs
  - GET, PUT, HEAD, DELETE
- URIs are the nouns
  - /
- Query parameters are adverbs
  - ?prefix=photos/&delimiter=/
- Request headers and reply headers carry metadata

Layering on HTTP: Pros

- Firewalls love port 80
- Proxies and clients already exist
- On the wire encryption via SSL
- Authorization infrastructure
  - Basic, Digest, extendible to other schemes
- Good for disconnected/mobile clients
- Internet-friendly
- Supports web browsers
  - Browser fetch is an unauthenticated read-only subset
  - So you can serve a simple static website

Layering on HTTP: Cons

- ASCII based protocol
- HTTP /1.0 connection model
  - Pessimal
  - Some clients still use it
- HTTP/1.1 pipelining
  - Limited by strict ordering
  - Not all servers support it
- HTTP/2 will solve a lot
  - but it's not here yet