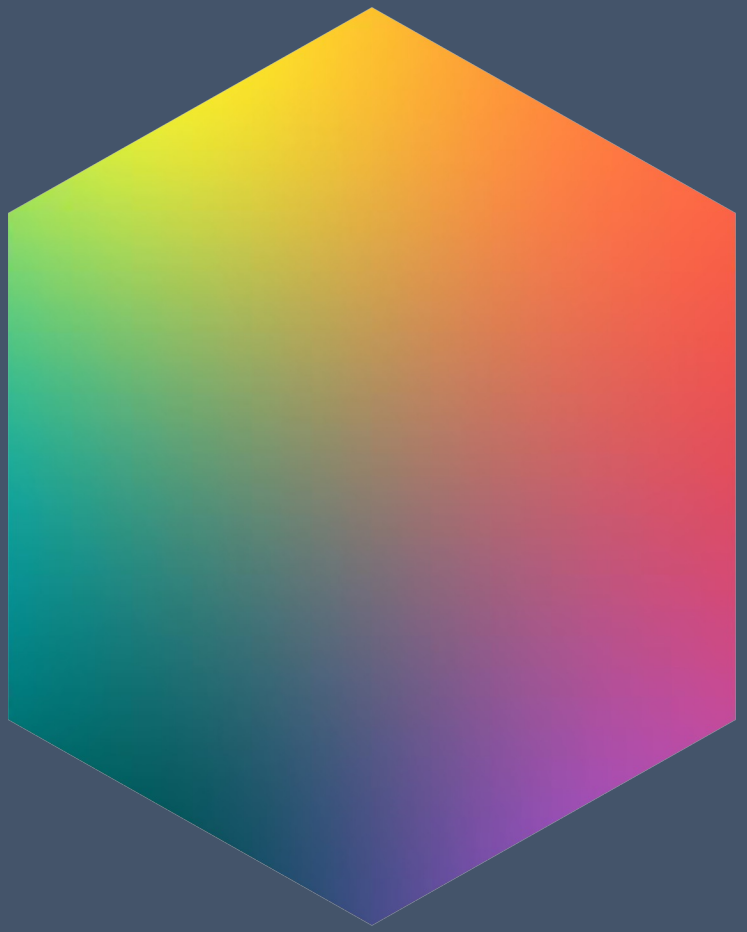
SCALITY

Scality and RainbowFS, the Search for the Grail Continues...



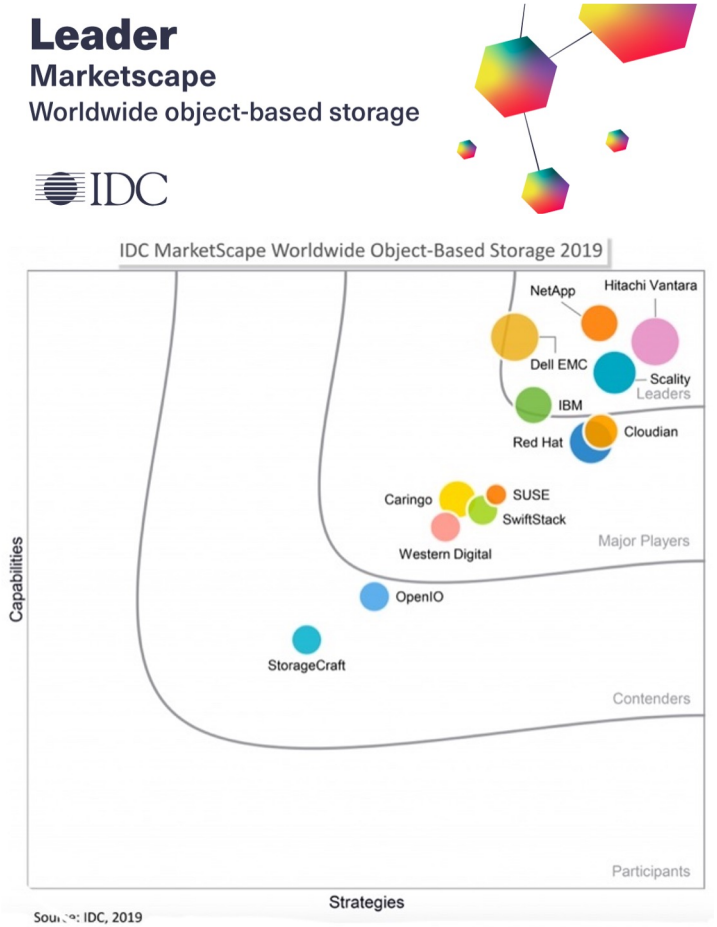
Bradley King, 28 March 2022

10+ YEARS PROVEN ENTERPRISE & CSP DEPLOYMENT EXPERIENCE

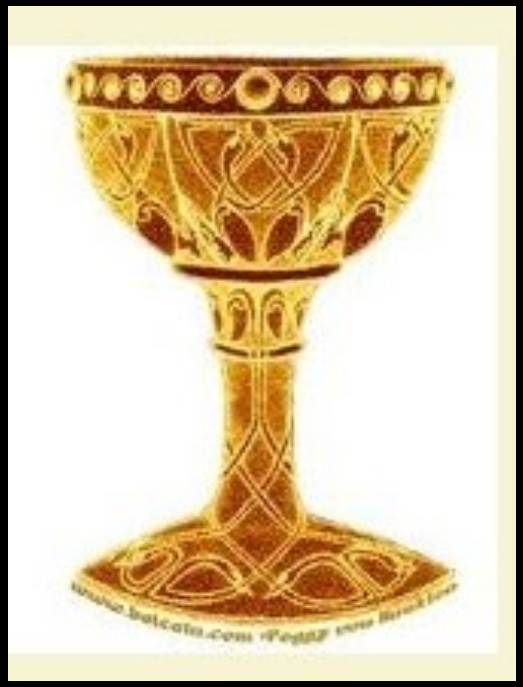
Scality supports customers in 46 countries worldwide

- 7 of the 15 **largest banks worldwide**, plus major insurance & brokerages in US & Europe
- **10 of the top 20** global Telcos & CSPs
- Video delivery and broadcasters in 5 countries
- **40+ large hospital systems** in US, LATAM, Europe, Japan & Israel
- Police & defense agencies in US, Israel & Europe
- **100s PBs in Govt. & intelligence** in US & EMEA
- Cloud and public services across Europe & Russia

WE ARE A RECOGNIZED MARKET LEADER



The POSIX Grail



Characteristics

- Distributed (worldwide)
- Multi-master
- POSIX Semantics
- Local filesystem latencies
- Local filesystem performance
- Strong Consistency
- Partition tolerance
- Highly available

February 2014 Vinh Tao begins CIFRE

- Can we use CRDTs to *bend* the rules and merge conflicts?
- Can we find POSIX and user compatible merging rules?

SCALITY

Some Industry Examples

- Microsoft DFSR "*The Distributed File System Replication (DFSR) service is a new multi-master replication engine that is used to keep folders synchronized on multiple servers.*"
  - **Speaking of issues due to conflicts** *" this is far less common than people like to believe"* **https://docs.microsoft.com/en-us/windows-server/troubleshoot/understanding-the-lack-of-distributed-file-locking-in-dfsr**
- Google Chubby and GFS with *advisory* locks
- The AFS with a weak consistency model
- Nasuni filesystem gateway (S3 based) supports global locking but is not partition tolerant

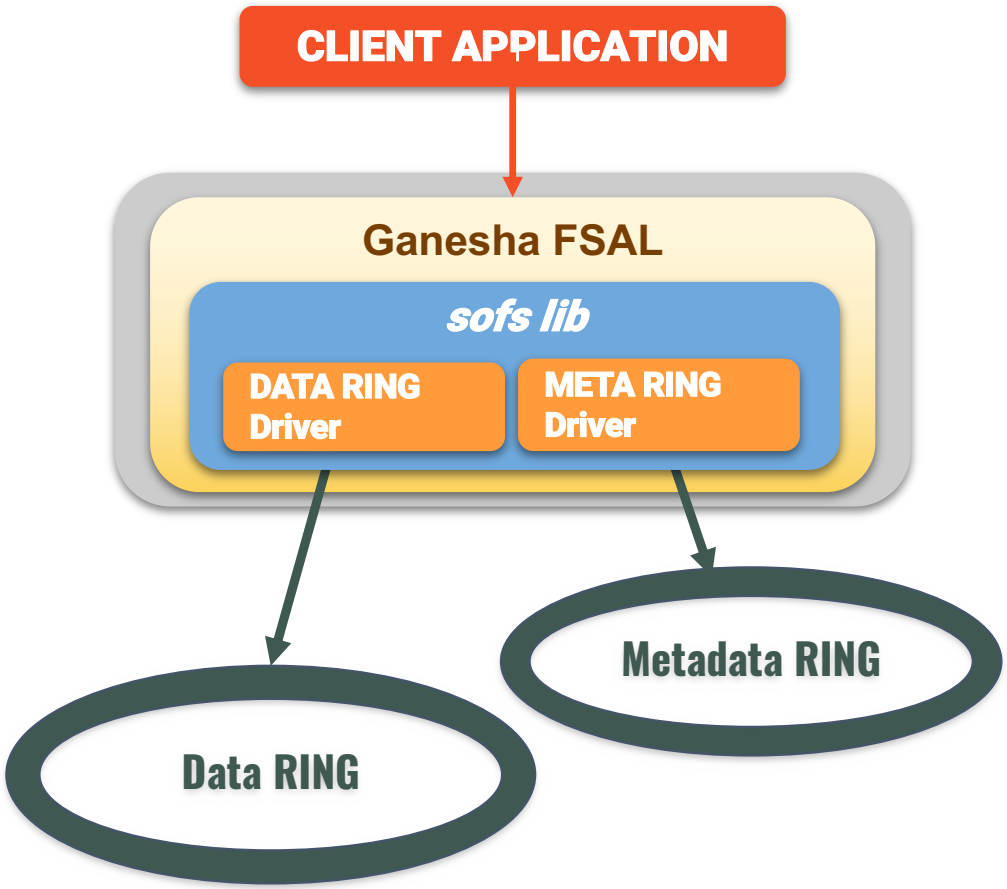
Example Use Cases

- Global Media
- Global Manufacturing
- Travel
- Medical

- Follow the sun usage patterns
- Bandwidth limitations
- Latency constraints
- Workers cannot be blocked by network issues

SCALITY

NFS v4 in RING8

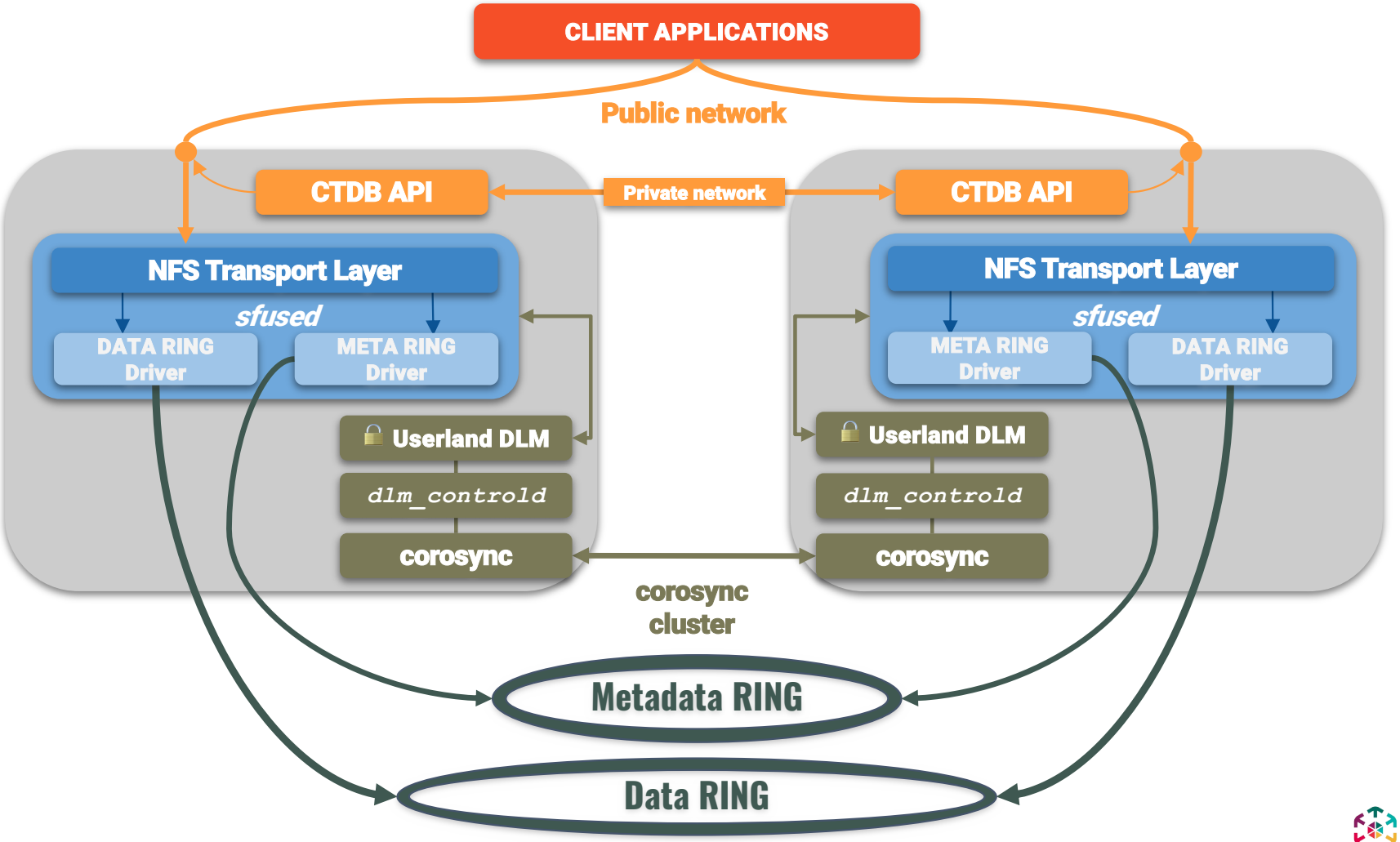


- Ganesha NFS server in userland.
- Supports NFS v3 and NFS v4.
- Supports max version 4.0.
- Native access to the RING via the SOFS lib
- Works with Quotas, Volume Protection, GeoSync, DLM.
- Full NFSv4 semantics.
- Kerberos capabilities.
- POSIX ACL's support
- IPv6 support.

MesaDB Overview

- Represented as a Tree.
- Each tree node is stored as an object **in the Scality Metadata RING:**
  - **The tree itself is described in a versioned object (main chunk).**
  - **All the nodes of the tree are immutable.**
- **Used to store** complex metadata **in the RING**
  - Catalog**: information associated to /, the** root directory of a filesystem,
  - Directory structure**, contains attributes, the** lists of directory entries.
  - File structure**, contains attributes, the** lists of sparse stripes' RING keys**: files are split in smaller stripes before being sent to the Data RING, and each stripe is assigned a RING key. These keys are part on the file structure.**
  - Quota indexes, Volume table, **…**

NFS Connector Overview w/ CTDB & DLM

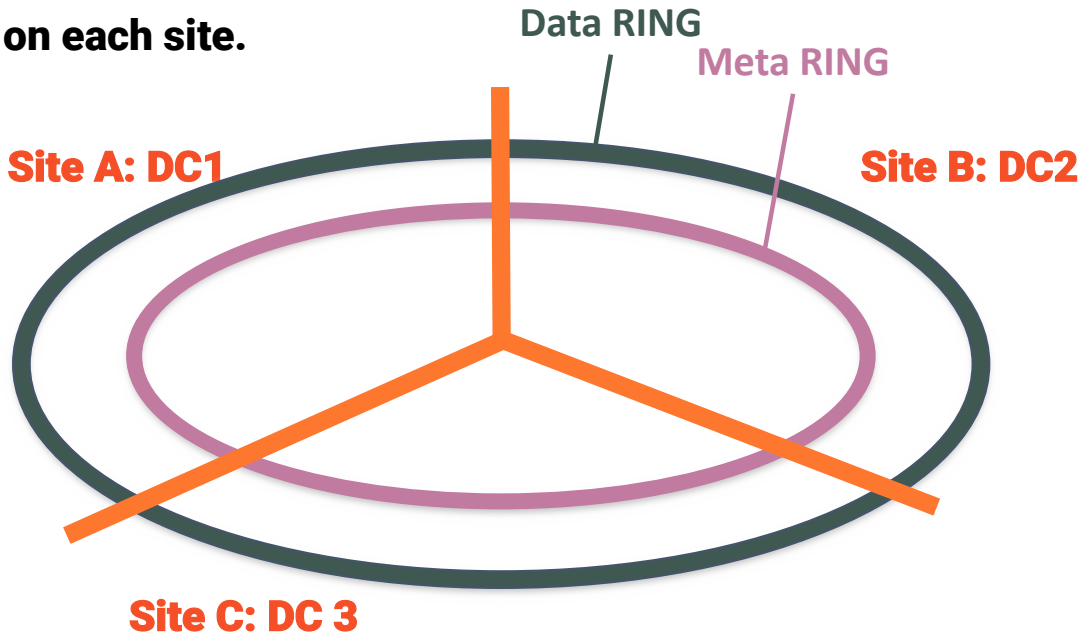


SCALITY

3-site+ Deployment

Standard Configuration

Equal number of storage servers (and capacity) on each site.



Data RING:

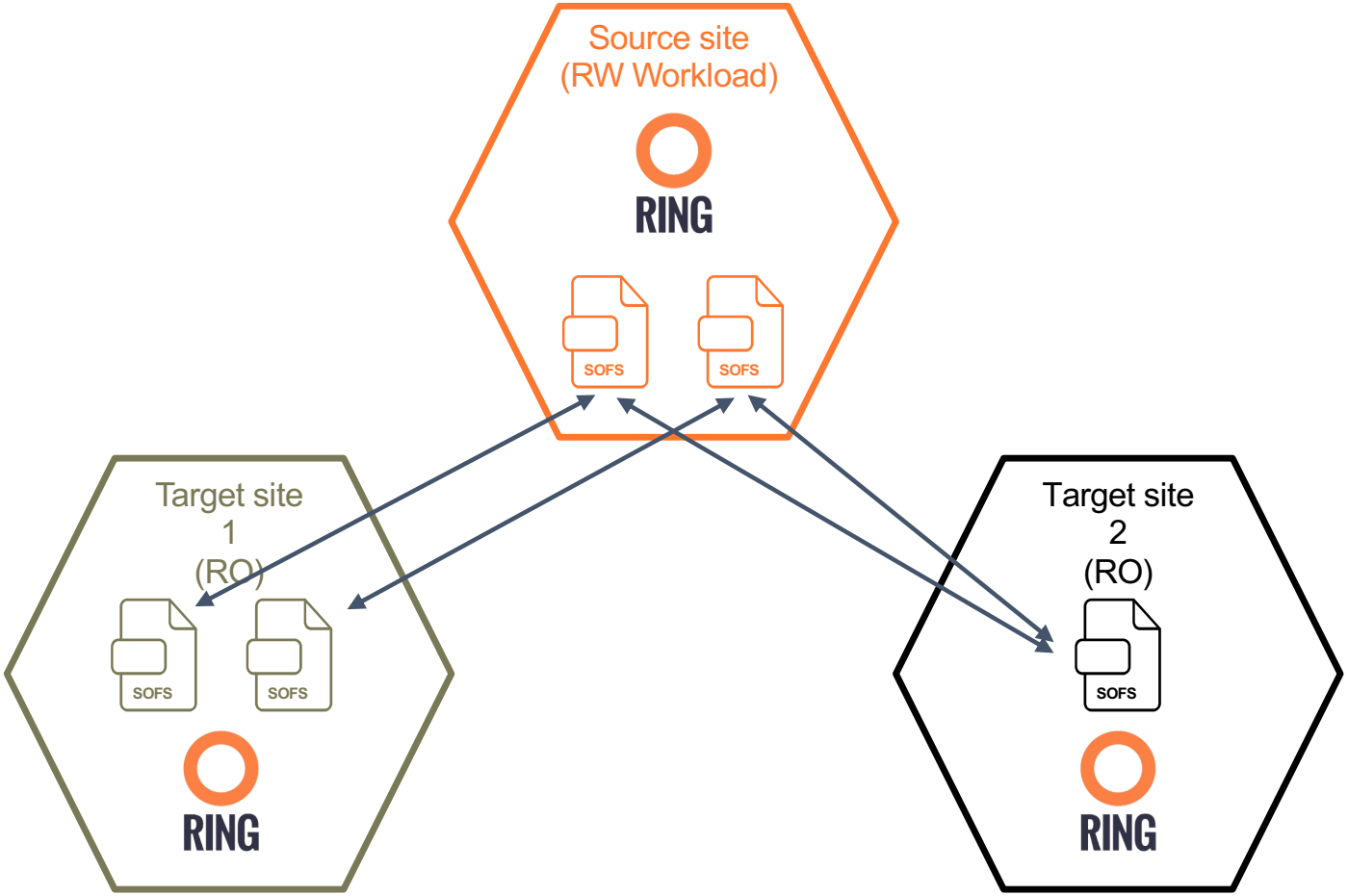
- **ARC7+5, CoS 3.**
- **Withstands** 1 site + 1 server offline **(or + 1 disk if less than 12 servers).**

Meta RING:

- **CoS 4.**
- **Withstands** 1 site offline**. One more component offline will cause** temporary service disruption* **until META rebuilds are completed.**

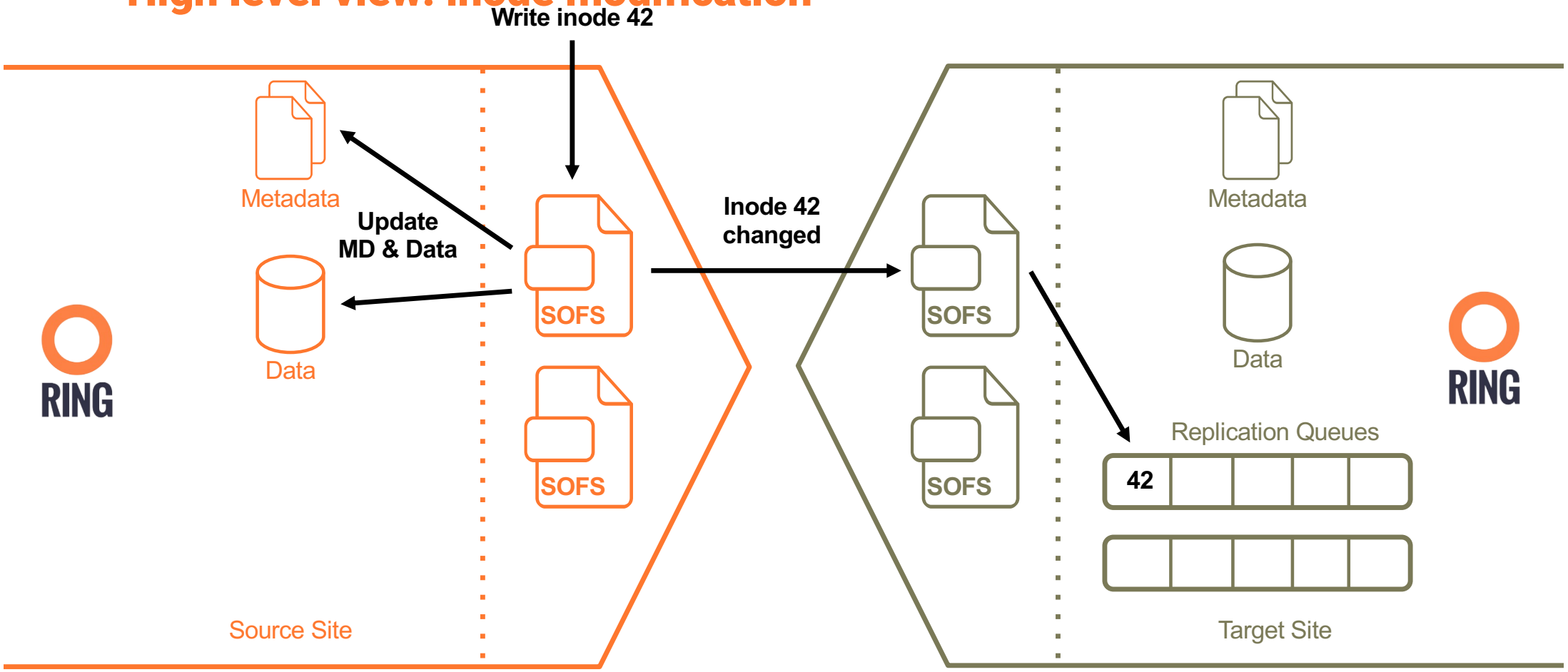
SCALITY

Multisite Asynchronous Replication for SOFS



Mechanisms

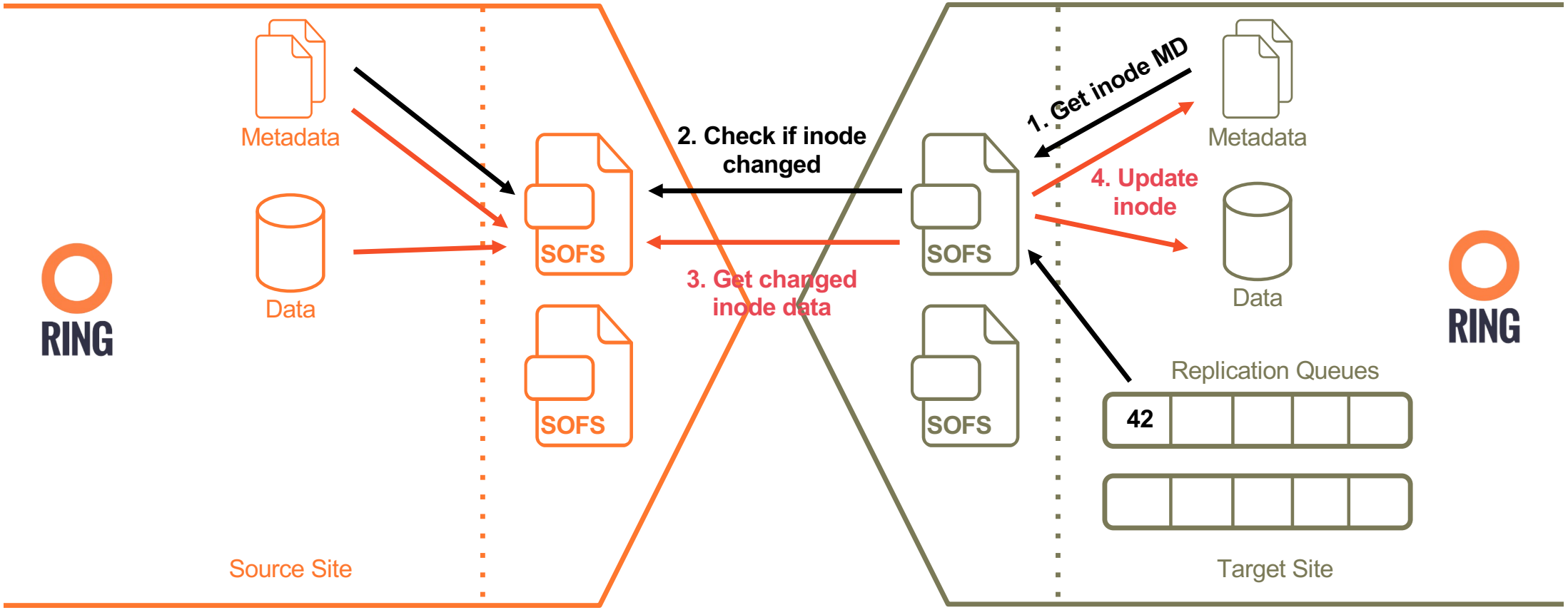
High level view: inode modification



SCALITY

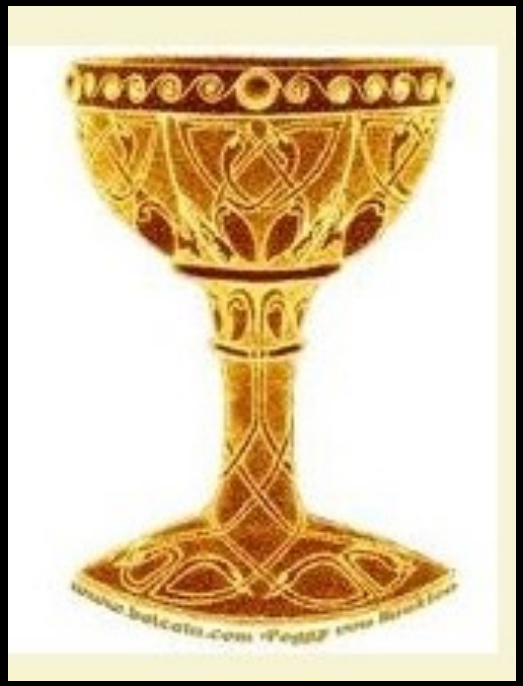
Mechanisms

High level view: inode replication



SCALITY

How goes the POSIX Grail?

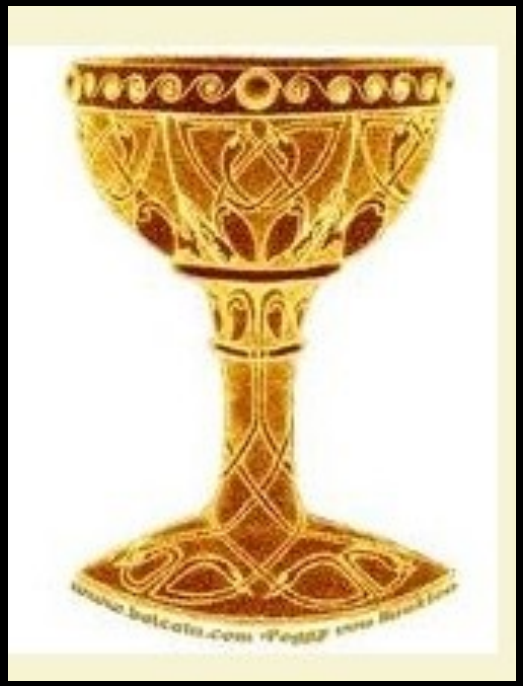


Characteristics

- Distributed (worldwide)
- ~~Multi-master~~
- POSIX Semantics
- Local filesystem latencies RO
- Local filesystem performance RO
- ~~Strong Consistency~~
- Partition tolerance (Failover/back)
- Highly available locally

SCALITY

The Rainbow FS
POSIX Grail



Characteristics

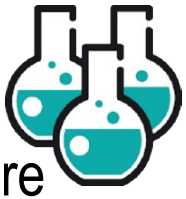
- Distributed (worldwide)
- Multi-master
- POSIX Semantics
- ~~Local filesystem latencies~~
- ~~Local filesystem performance~~
- Strong Consistency
- Partition tolerance
- Highly available

- Can we use CRDTs to *bend* the rules and merge conflicts? Yes
- Can we find POSIX and user compatible merging rules? Yes

SCALITY

Key Challenges

- Performance challenge is significant
- Antidote is not MesaDB
- Journal grows in an unbounded fashion
- Antidote codebase with Riak core discussed by Annette Bieniusa
- Caching (Ayush Pandey) and Journal truncation (Saalik Hatia) are works in progress
- Industrialization



What does the chasm look like?

SCALITY

Object Storage or File systems?

A clear trend with multiple implications

SCALITY

How do the rules change?

- AWS S3 the non-standard API standard
- Globally addressable via URI
- Replication is active-passive
- Renaming (move) is not possible
- Versioning is part of the API (while POSIX has no such feature)
- Early versions were eventually consistent
- Newer versions have consistency guarantees with last writer wins merge semantics + versioning

Distributed Object Storage

SCALITY

Industry Megatrends

Serverless Applications
Lambda and K8S

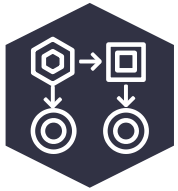
Greener IT

Cyber Physical Systems

AI/ML everywhere

API everywhere

“Smart” everything



Edge-to-core-to-cloud

Hybrid cloud

Key Points

1. Many applications now support Object APIs
2. AI/ML workloads need heterogeneous data sources
3. ”Cloud native” /Serverless / PaaS work well with API based WAN friendly storage
4. Metadata and ACLs are richer and better adapted to sharing
5. The POSIX contract has been replaced: pushing work to apps

https://www.cio.com/article/191758/object-storage-will-underpin-the-next-generation-of-it-are-you-ready.html

SCALITY

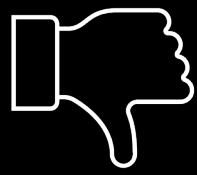
Positive Conclusions

A conflict merge model that respects POSIX invariants and “makes sense” to user appears viable

The CRDT based approach keeps its promise of maintaining availability in the case of partitions

There is a better way than just “hoping for the best”

Less Positive



The architecture is quite different from Scality’s current DB model

Significant effort would be required to develop the required functionality in MesaDB

The current trends in data storage militate for object storage with a REST API based immutable versioned approach going forward rather than multi-master POSIX

SCALITY