ORACLE

Architectural Lessons and Operational Case Studies from OCI's Managed Lustre Service

Aboo Valappil & Sonia Sharma

OCI LFS Engineering

LUG 2026

Agenda – Lustre in Oracle Cloud

1 Why Lustre is different in cloud?

2 A practical Lustre deployment in cloud

3 Key architecture decisions in OCI

4 Operational Challenges & Fixes

5 Lessons Learned

Cloud vs Traditional Lustre

	Traditional HPC	Cloud Reality
Compute	Static hardware	Standard Compute
Networking	Low latency fabric	Variable latency
Consumption model	Single tenant	Multi-tenant
Failure and Maintenance	Predictable maintenance	Unplanned maintenance

- Cloud makes it easy for customer to consume Lustre but implementing it in cloud breaks traditional HPC assumptions

Lustre in Cloud – Key Challenges

- Infrastructure
  - Standard Compute instances
  - Higher Network Latency & Variability
    - Availability Domains span multiple buildings (miles apart)
- Operations
  - Unplanned and frequent maintenance

- Scale and multi-tenancy
  - Large-scale metric generation (1000s of OSTs)
  - Strict multi-tenant isolation requirements.

Lustre in Cloud – Key design decisions

- Use overlay networking
  - OCI virtual network (VCN)

- Use
  - Standard instances (use and terminate as needed)
  - OCI's Block volumes for storage

- High availability
  - OSTs with IP address
  - Move the IP within sub-seconds

- Low Latency
  - Cluster placement groups

- Multi-tenant tenant architecture
  - Cross tenancy VNIC attachments
  - FS IP addresses in network namespaces

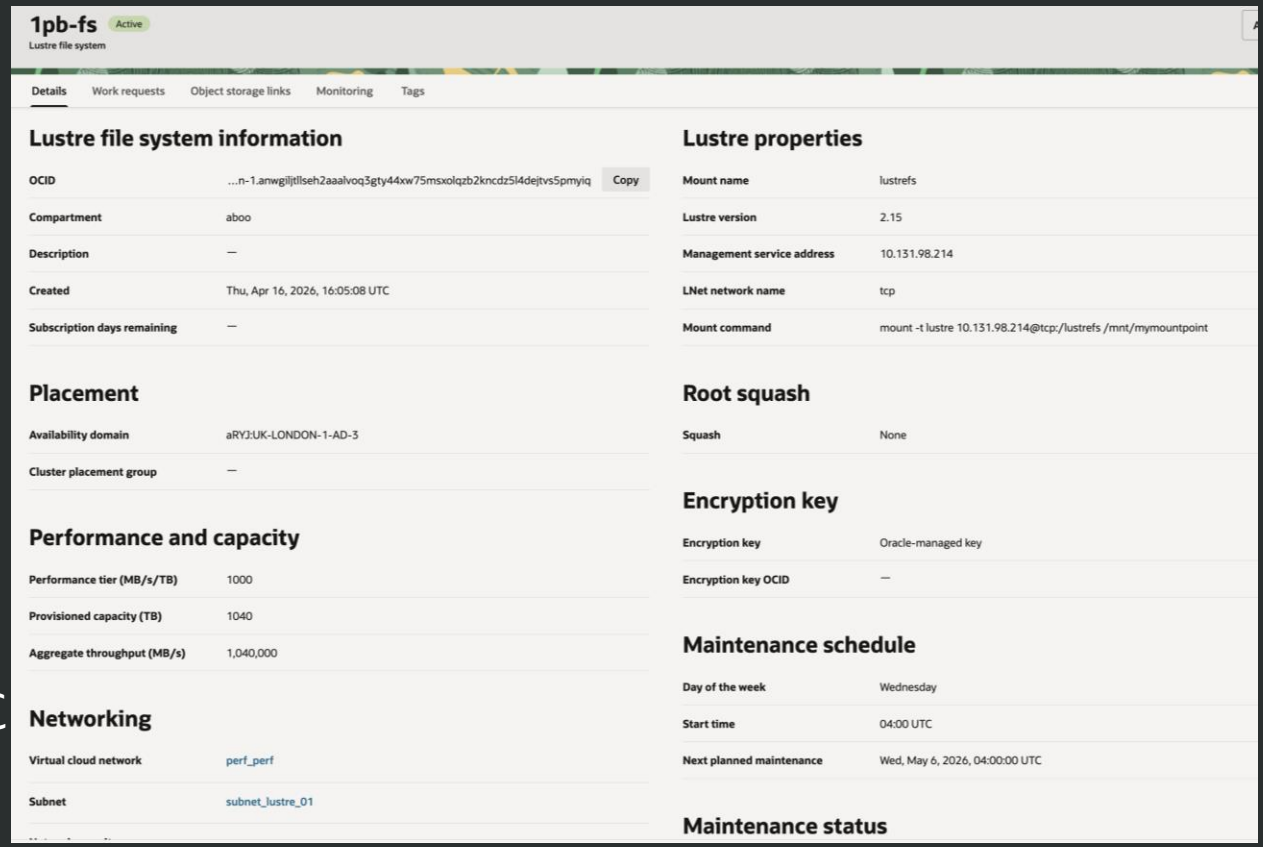
- Telemetry
  - Distributed collector and publish to internal and external OCI telemetry backend

- Parallel resource provisioning and mounts
  - OCI APIs to provision Lustre resources

Lustre in Cloud – A practical configuration and scale

- 1PB, 1000MB/s per TB performance
- Lustre config
  - 2.6TB volumes with 2600MB/s-
    - 300k IOPS
  - 100 OSS with 400 OSTs
  - 7 MDS with 7 MDT (1:1)

- Lustre servers
  - 126 core VMs with 1.4TB memory
  - 100gbps NICs



Elbencho - Benchmark with 44 x 200gbps NIC

IO Type	IO Size	Throughput
Random Read	1MB	1092.13 GB/s
Random Write	1MB	869 GB/s

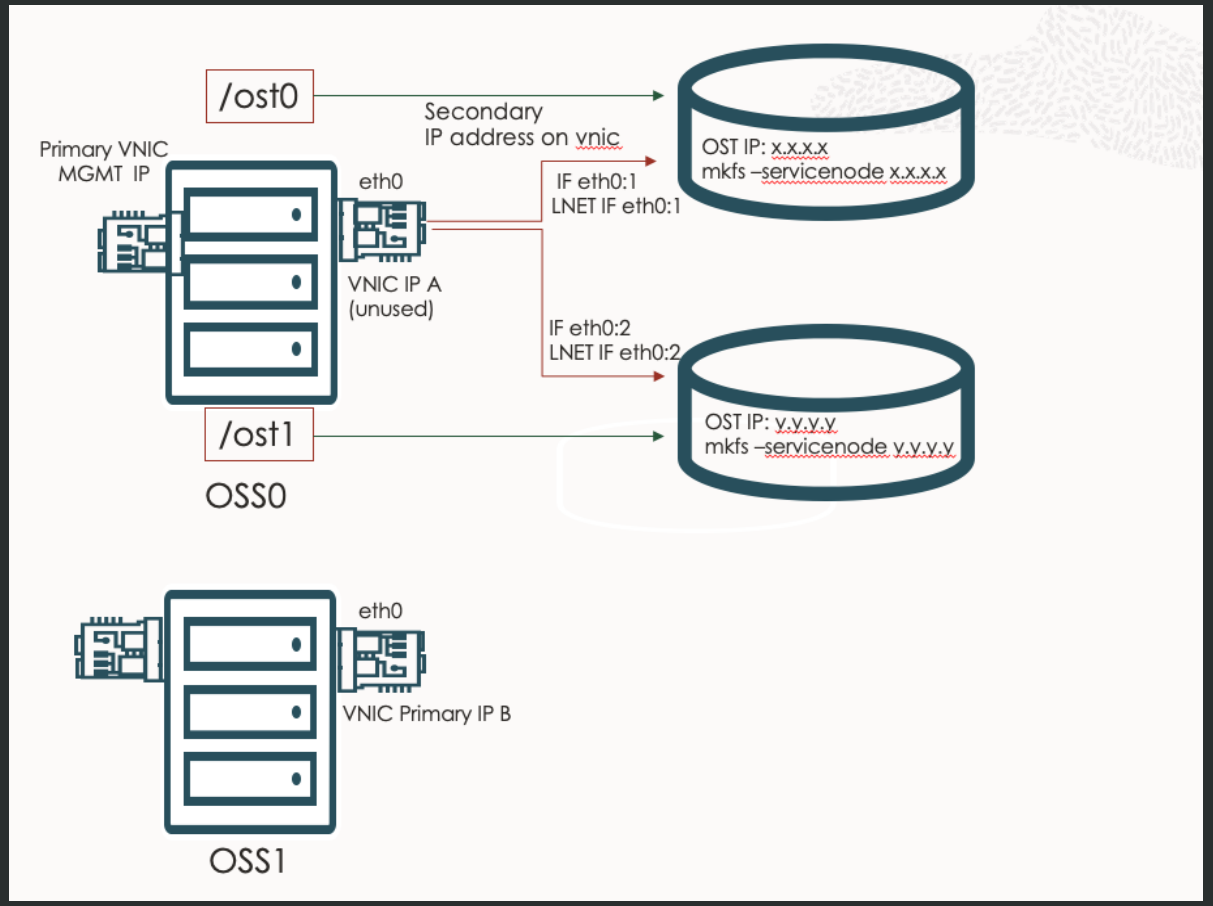
Total Files	CREATE op/s	CREATE/MDT ops/s	STAT op/s	STAT/MDT ops/s	DELETE op/s	DELETE/MDT ops/s
46,080,000	1,407,297	201,042	3,867,416	552,488	1,074,775	153,539

Architecture choice – Achieving high availability

- Volume durability and performance
  - OCI block volumes with SLA for availability and performance
  - ISCSI or NVMe attachments
  - Predictable throughput at different VPU tiers
  - Volume attachments to multiple servers
- IO Fencing
  - ISCSI3 persistent reservation
  - Ext4 multi-mount protection (MMP)
- Fault domain aware provisioning
  - Provision resources across three fault domains

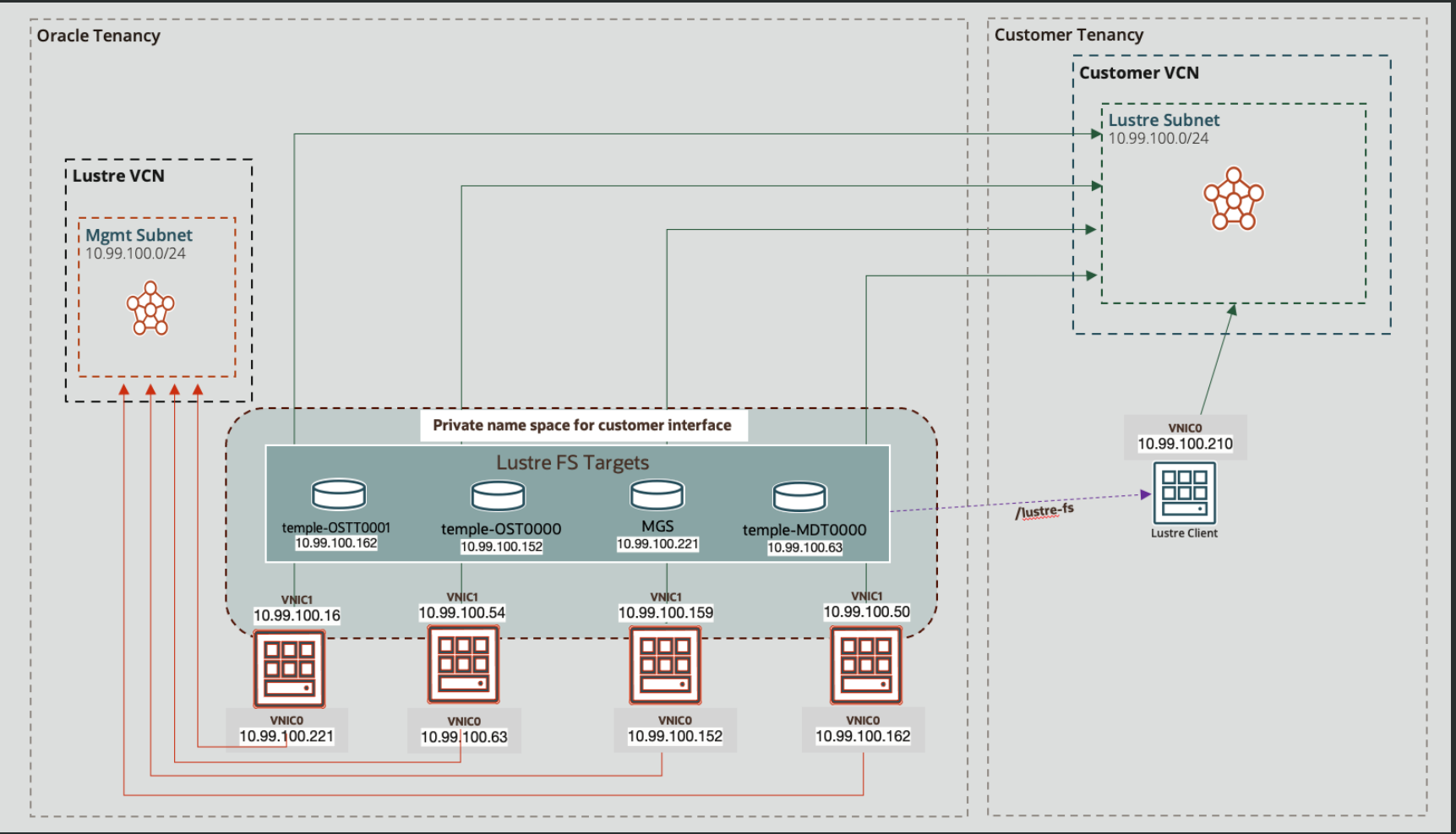
Architecture choice – Achieving high availability

- Each OST/MDT/MGT tied to a unique IP (not host)
  - Format with IP address identity
    - mkfs.lustre --servicenode <ip>
- Failover
  - Move volume to new OSS
  - Move IP to new OSS
- Benefits
  - Fast recovery
  - Decouple compute and storage
- Trade-offs
  - Disable multi-rail
    - options lnet lnet_peer_discovery_disabled=1
  - Disable imports from loopback
    - options lnet local_nid_dist_zero=0



Architecture choice – Multi-tenant isolation

Enables Lustre-as-a-service with strong tenant isolation



- Dual attached OSS
  - Lustre VNIC attached to customer VCN
  - Other VNIC attached to OCI managed network
- Linux IP name spaces
  - Separate IP spaces for customer and service
- Security of native OCI networking
  - OCI native security controls

Architecture choice - Low and predictable latency

Cluster placement groups

- Problem
  - Availability domain spans across multiple buildings
  - Cross building traffic introduces latency and congestion
- Solution
  - Cluster placement group
    - Provision all resources (Compute, Storage and Clients) using CPG
  - Co-located resources
  - Low latency < 100us

Architecture choice - Distributed Telemetry

- Challenge
  - Metric cardinality explosion
    - Dimensions multiply total streams
      - Example: 4000 OST, 100 clients, 2 operations across 100 FS = 80M
    - Cloud metric pipeline has capacity limits
    - Scalability: Centralized publishing does not scale

- Solutions
  - Distributed metric emission by OST/MDTs
    - No coordination and no retries
  - Aggregate at query time
    - Example: read_bytes.sum()
  - Limit high cardinality and unbound dimensions
  - Reduce frequency

Maintenance in Managed Lustre

- Challenge
  - Periodic maintenance needed,
  - Provider defined schedules
  - Client recovery during OST/MDT failover takes minutes
    - Recovery time outs due to unresponsive clients
    - Affects extreme sensitive workloads

- Solutions
  - Configurable maintenance window
    - Ability to configure time for maintenance
    - Ability to override next maintenance activity

Operational issues - Protect Lustre servers

- Problem
  - Heavy metadata workloads -> 100% CPU and memory on MDS
  - Heartbeat failure -> Repeated failovers
- Root cause
  - Large number of active locks on MDS
  - High cost of managing locks, especially lock cancel
- Mitigation
  - CPU partitioning and CPU for system tasks
  - Tune OOM for system services
  - Client side LDLM tuning
    - `lctl set_param ldlm.namespaces.*.lru_max_age=600000`
    - `lctl set_param ldlm.namespaces.*.lru_size`

Operational issues – Parallel mount

- Situation
  - OCI supports large file systems at 10 PB, 20 PB, and beyond.
  - 10+ hours deployment time with about 4K OSTs and serial OST mounts.

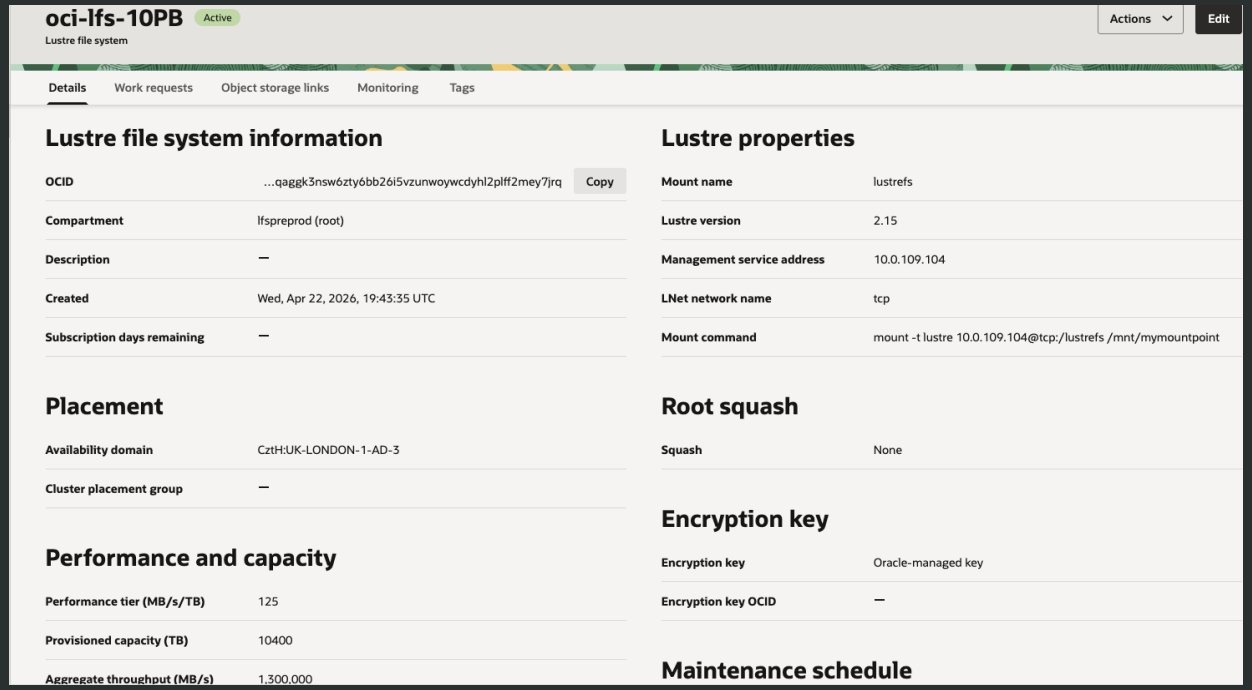
- Problem
  - Parallel mount exposed MDS crashes under heavy concurrent OST initialization.
  - A big problem during file system expansion as customer can't access data
    - Lustre write-conf needed

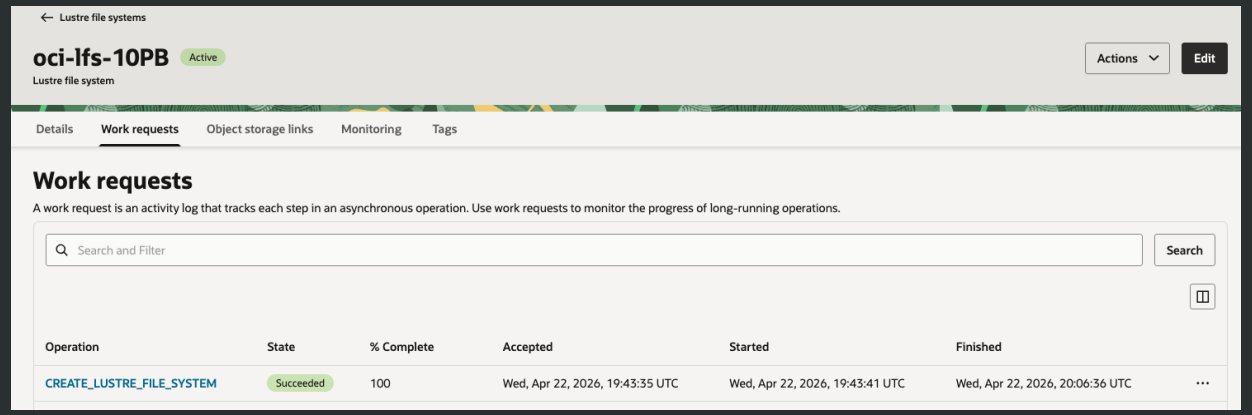
- Fix
  - LU-18495 osd: take a semaphore if non-allocated block met
  - LU-18763 osd: fix credits calculation for overwrites

- Result
  - 10 PB FS deployment time in less than 30 minutes

From nothing to 10PB FS in < 25 mins

Operation	Time
Resource provisioning	17m 42s
Format	51s
Mount MDT/MGT	2m 43s
Mount OST	1m 15s
Total time	**23m**





Operational issues – Full OSTs

- Problem
  - OST reached 100% used
  - OST crashes and cannot remount!
    - Big problem as customer cannot access data

- Work around
  - Expand LDISKFS on affected OST
  - Increase osp.*.reserved_mb_low and osp.*.reserved_mb_high to prevent future issue

- Fixes - Are these enough?
  - LU-17985 osd-ldiskfs drop ost object if failed to create
  - LU-14714 server to mount without local config
  - LU-19779 adds observability around full OSTs

Operational issues – ARM64 Clients with x86 Servers

- Problem
  - Wider adoption of 64K ARM GPU nodes for AI/ML workloads
  - OCI LFS servers are x86 with Lustre LTS 2.15.5

- Solutions
  - Backported unaligned-DIO support to our 2.15.5 servers
  - Simpler support model across a diverse customer base
    - One server baseline to support all clients universally

- Challenges
  - Continuous support for newfound X-compat issues on 2.15.*/2.16.* client versions
  - Lustre 2.17 client issues due to hybrid IO bug LU-19932
    - Works with disabled hybrid IO

Key Learning

- Running Lustre in Cloud is not same as traditional deployments

- Sensitive customer workload could still be affected by OST/MDT failover/failbacks
  - Provide customer with configurable maintenance window.

- Design for failure
  - Recover faster rather than preventing failure
  - Replace failed resources(eg: Compute) instead of fixing them
  - Reduce tight dependencies with other cloud services

- Telemetry at scale
  - Cardinality grows rapidly
    - Control: Interval and dimensions
  - Aggregate when querying and distributed collection

- Provide service stability
  - Design for unpredictable customer workload or user behavior

In cloud, reliability comes from fast recovery not from preventing failures.

Closing

OCI LFS has completed one year in service with great learnings!

We acknowledge the support from Whamcloud & Lustre community in reaching the milestone.

Questions?