[ICTC 2025] P06-26

Toward Heterogeneity-Aware Striping in Lustre

Sooyoung Lim, Jaegi Son, and Dongmin Kim

slim@keti.re.kr, jgson@keti.re.kr, dmkim@keti.re.kr

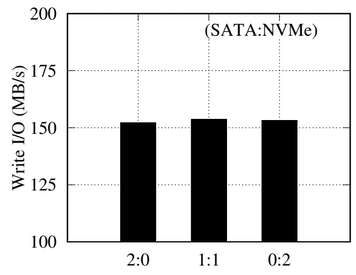
Intelligent IDC Group, **Korea Electronics Technology Institute (KETI)**
Seongnam-si, Gyeonggi-do, Republic of Korea

Background

- ❑ **Lustre in HPC** – De facto parallel file system for large-scale I/O
- ❑ **Heterogeneous Storage** – Modern clusters mix SATA, SAS, and NVMe devices
- ❑ **Lustre Striping Limitation** – Originally ignores device performance, and causes **imbalance**
- ❑ **Prior Approaches** – Overstriping & dynamic striping adapt to file size or access patterns, but not heterogeneity
- ❑ **Key Insight** – Lustre performance remains insensitive to storage type → **Policy**, not hardware, is the bottleneck!

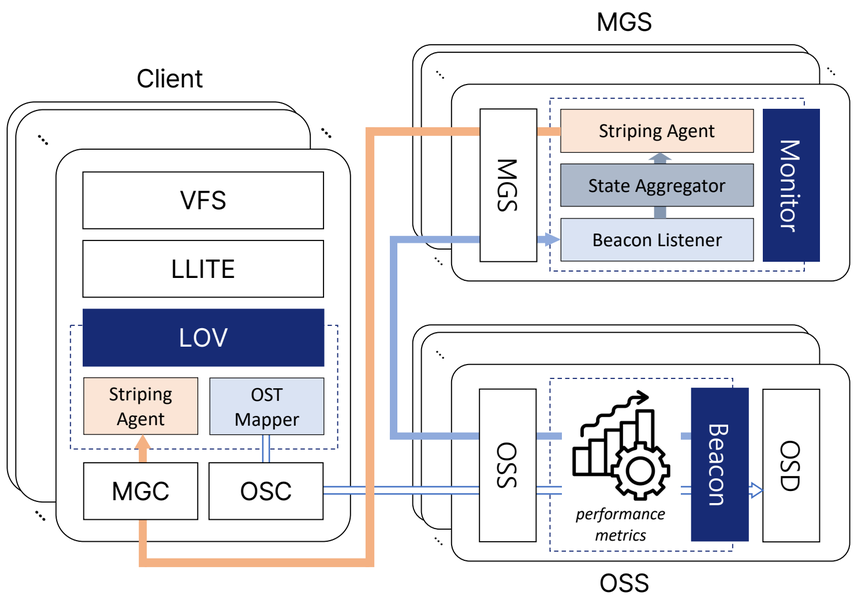
Motivation

- ❑ **Result** – Nearly **identical** write bandwidth across SATA-only, mixed SATA-NVMe, and NVMe-only striping
- ❑ **Problem** – High-performance devices remain underutilized
- ❑ **Cause** – Lustre's default striping assigns I/O statically, without considering the storage performance gap
- ❑ **Contribution** – **Striping** is the solution of **heterogeneous** Lustre!



Heterogeneous Storage
IOR Write Performance

Proposed Architecture



Heterogeneity-Aware Lustre Striping

- ❑ **Client: Resource-Aware Striping**
  - At file creation, the client queries MGS for up-to-date resource insights and selects OSTs based on performance-aware policies.
- ❑ **Management Server: Centralized Aggregation**
  - Beacon Listener, State Aggregator, and Striping Agent process OSS metrics to detect overloaded OSTs and provide ranked resource summaries.
- ❑ **Storage Server: Lightweight Monitoring**
  - Each OSS periodically collects CPU, memory, and storage performance metrics and sends them as *ResourceBeacons* to MGS.

Client

Objective

- To enable **resource-aware stripe allocation** during file creation

Our Design

- ❑ **Striping Agent**
  - Queries the MGS for updated OST resource information before stripe selection
- ❑ **OST Mapper**
  - Selects optimal OSTs based on resource-aware policies
- ❑ **Legacy Integration**
  - Preserves existing workflow through *lov_object_alloc()*

Management Server

Objective

- To provide **centralized aggregation** for resource state management

Our Design

- ❑ **Beacon Listener**
  - Receives *ResourceBeacons* from OSS nodes
- ❑ **State Aggregator**
  - Maintains per-OST resource states
  - Flags overloaded devices
  - Exports summaries
- ❑ **Striping Agent**
  - Responds to client queries with ranked or filtered OST lists

Storage Server

Objective

- To enable **lightweight** and continuous **monitoring** of OSS resources

Our Design

- ❑ **Lightweight Resource Beaconing**
  - Runs on each OSS to collect CPU, memory, and storage utilization
- ❑ **OST-level Metrics**
  - Extracts detailed statistics from the OSD layer via *procfs*
- ❑ **Periodic Reporting**
  - Encapsulates collected metrics and transmits them to the MGS

Future Work

- ❑ **Module Implementation** – Integrates the proposed client striping, beaconing, and monitoring modules into Lustre
- ❑ **Performance Evaluation** – Validates the design under diverse and heterogeneous workload scenarios
- ❑ **Adaptive Re-Striping** – Explores real-time striping adjustments based on workload prediction

The 16th International Conference on ICT Convergence (ICTC 2025)

This work was supported by IITP Grant RS-2025-02215256.

Korea Electronics Technology Institute