Lustre HSM for HPC
(High Performance Distributed Data Mover)

SEAGATE

Ujjwal Lanjewar
Bikrant Singh
Ajay Nair
Bhagyesh Dudhediya

LUG 2017 (Bloomington, IN)

Agenda

Lustre HSM State

- Lustre HSM Highlights
- Summary and Facts

Target Environment

- Development Environment
- HSM Use Case for HPC

Distributed Data Mover

- HSM Solution Overview
- Architecture of Distributed Data Mover

Lustre HSM Improvements

- Challenges with Lustre HSM
- Wishlist of changes in Lustre HSM for HPC use cases

Lustre HSM Highlights and Facts

Lustre HSM Policy / Commands

- HSM Operations (Archival, Release and Restore) involving whole file
- Sparse Files handling is delegated to Copy Tool

Lustre HSM Coordinator integration via Copy Tool

- Single file is handled by any one copy tool instance
- Loose coupling with Copy Tool (Startup, state detection, recovery, etc)

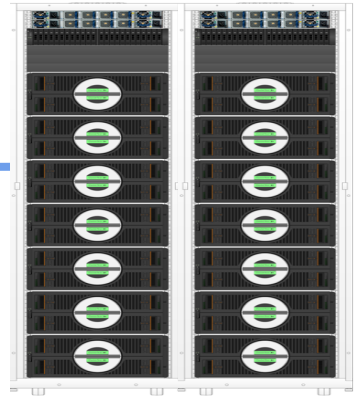
Lustre Copy Tools

- Posix Copy Tool - Specific to POSIX backend, suitable for small / mid-sized files
- Other Copy Tools - Developed for specific targets/use cases

Target Environment

Primary Storage
(Lustre 2.7.0)

HSM Solution
(Data Mover Nodes)



Active Archive Store

HPC Requirements

Archival / Restore
Large sized files (~PB range)
At a high speed (Performance)

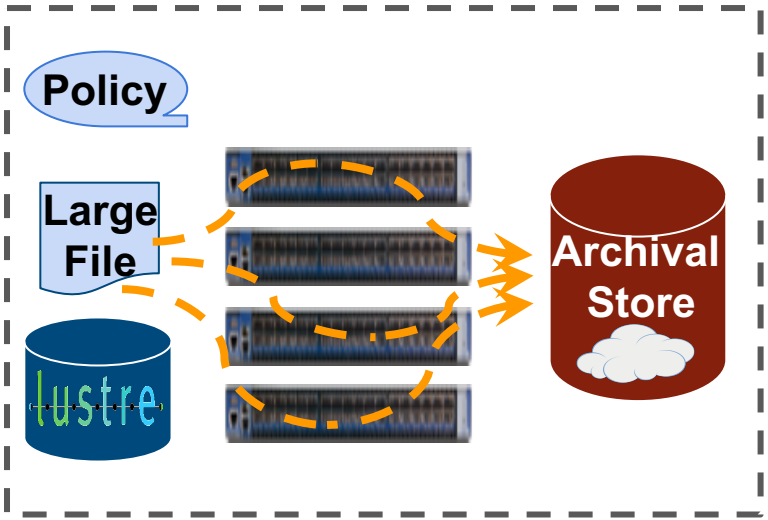
HPC Archival Storage

Scalable Object Storage
Key Value Store (for metadata)
Layout (Versioning, Snapshot, HSM)

Distributed Data Mover Overview

Primary Objective

Policy based Archive/Restore a single large file over multiple nodes / threads



High Speed Archival / Restore

- Single large file over multiple nodes / threads
- Partial File Archival / Restore

Lustre HSM Operations

- Remove, Import, Rebind, Cancel, Recovery
- Robinhood Policy HSM Operations

Performance Optimizations

- Throttling, Optimized Data Transfer
- Tunable for large sized buffers, parallelism

Availability and Reliability

- Reliable Archival
- Recovery from Failure
- Version Tracking, Bandwidth Control

Distributed Data Mover Architecture

Peer-Peer Architecture

- All nodes register with Lustre
- All nodes participate in data movement

Plug-in Based

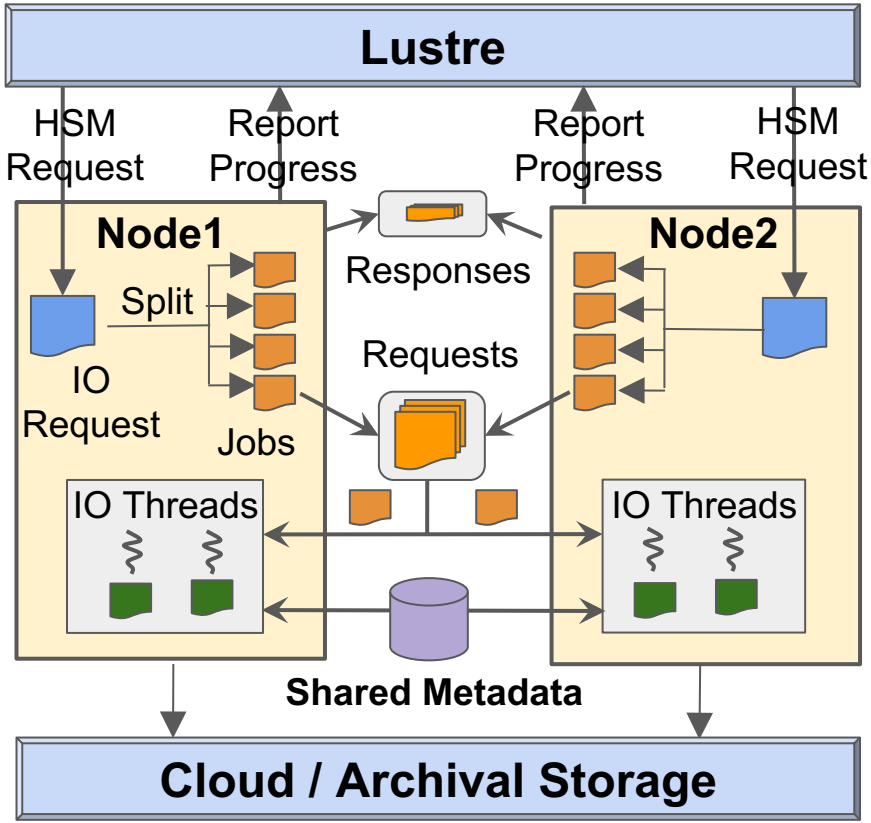
- Plug-in Integration with back-end storage
- Lustre or other filesystems as tiers

Distributed / Parallel IO

- Shared IO Request Queue
- Multithreaded, Async, Balanced IO

Metadata

- Shared Metadata across nodes
- Integration with KV store



Distributed IO Request Handling

Example Conf→ B=32M, C=1, M=4, N=2 (nodes)
Input File Size → 256M, Job Size=BxCxM
IO → Jobs: 2x128M, N1: 4x32M, N2: 4x32M

IO Request is split into P Jobs

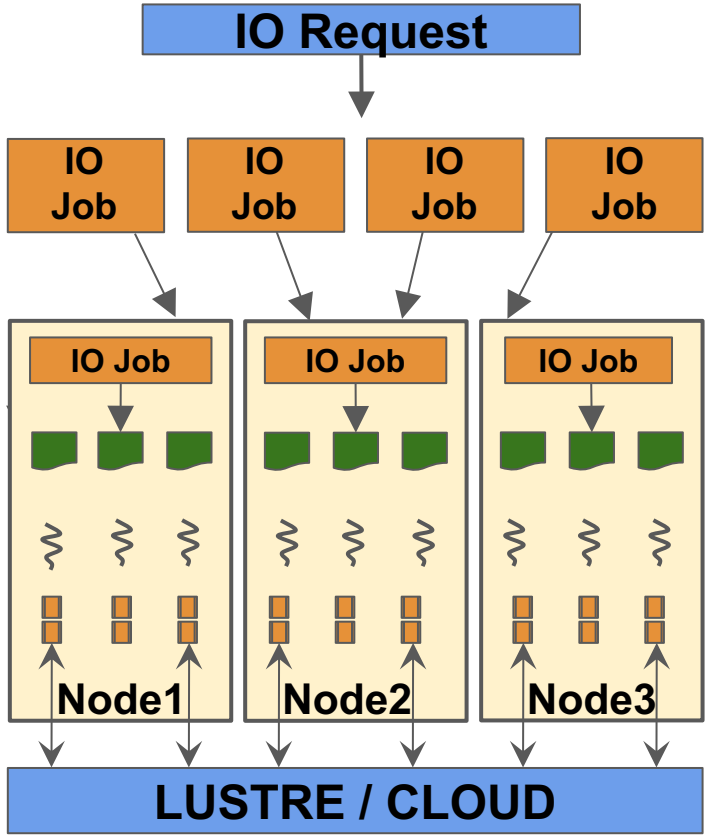
P Jobs are distributed over N nodes

Each Job partitioned into M Chunks

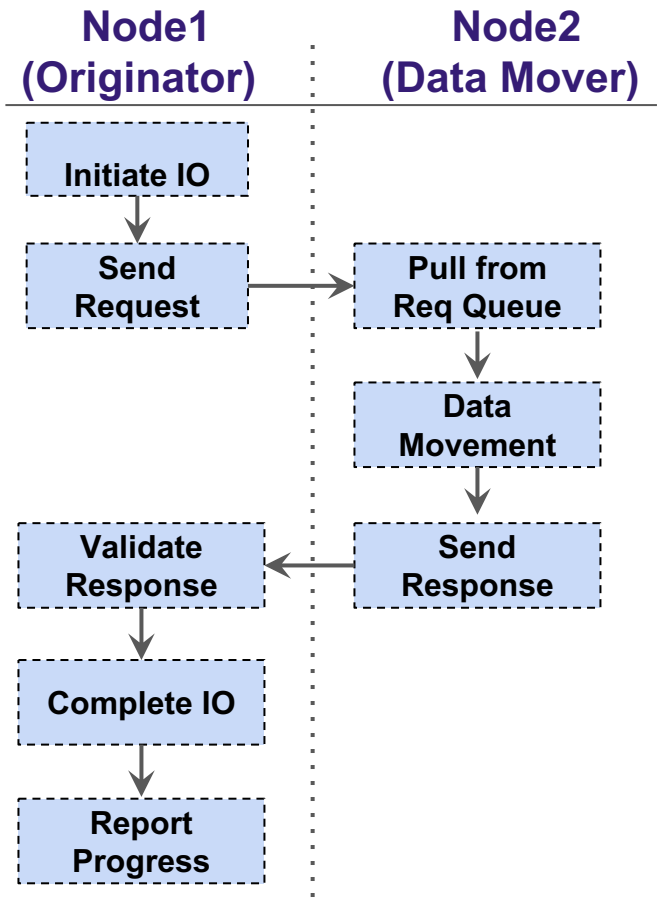
M Chunks scheduled over K Threads

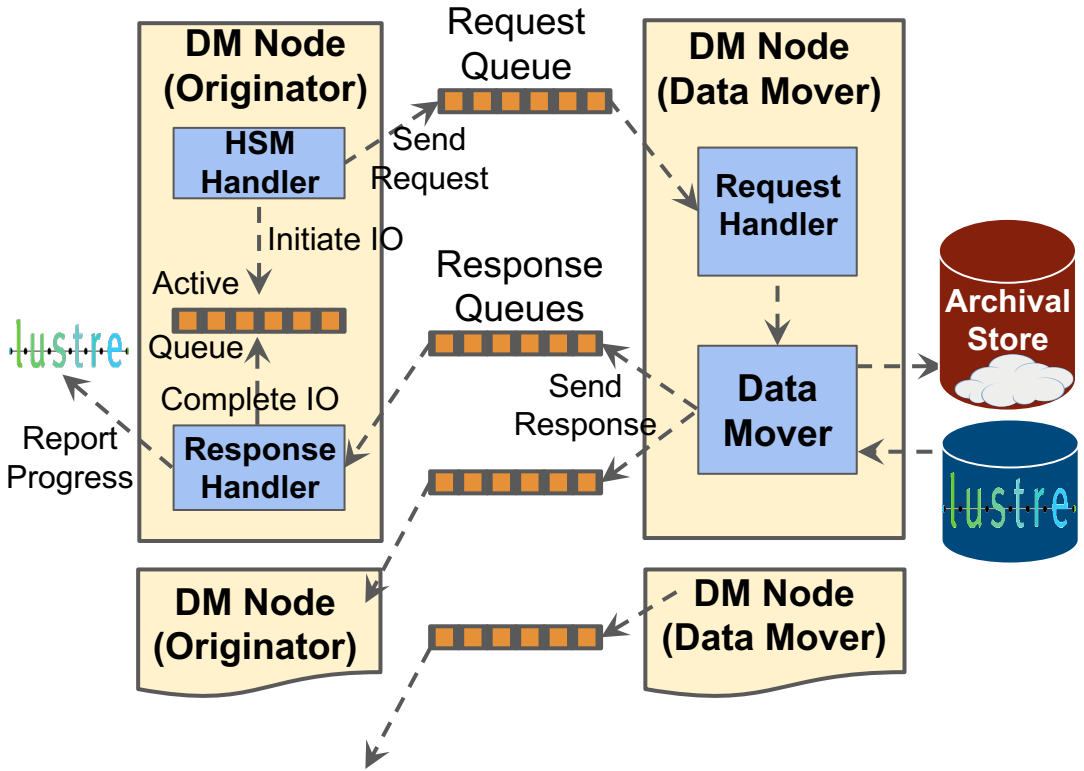
Each Chunk contains C Buffers of Size B

IO is processed by N x K channels (Best Case)



IO Monitoring and Progress Reporting





Current Status and Plan

ClusterStor HSM v1.x - Available on A200 Active Archive

- **Single File over Single Node**
- **Many Files over Multiple Nodes (Each file through one node)**
- **Multithreaded IO**
- **Verified on Lustre 2.7.0**

ClusterStor HSM v2 - Under Development

- **Multi-Node Support for Single File**
- **Performance Features**
  - Distributed and Parallel IO
  - Performance Optimizations e.g. zero copy with backend capabilities
- **Integrated Metadata with backend (KV Store)**

Challenges faced with Lustre HSM

Copy Tool Integration

- **Synchronization and Handling of Copy Tool**
  - Copy Tool Startup with Lustre
  - Impact on HSM action on Copy Tool Restart / Hang
- **Multiple Coordinated Copy Tools**
  - One HSM operation by one copy tool instance
  - No retry of HSM action another copy tool instance.

HSM Operations

- **Archival and Restore**
  - Partial File, Sparse File Archival and Restore
  - Restore via volatile file. Volatile files are limited to single node and hence restore.
- **Policy and HSM actions**
  - Issues with hsm_remove policy command due to SOFTRM table updates
  - Lack of advanced monitoring e.g. time out for HSM actions

Lustre HSM Wishlist for HPC

Archival and Restore for Partial File Range

- **Representation of partially archived or restored file**
  - Progressive File Layout (LU-8998)
- **Release of partial file blocks**
  - Punch Hole i.e. FALLOC_FL_PUNCH_HOLE Support (LU-3606)

HSM Performance

- **Multi-Node HSM Operations**
  - Coordinated Copy Tools e.g. failover HSM action to another copy tool instance
  - HSM Restore: Use of multiple volatile file streams. Prioritized restore operation
- **HSM Request Handling**
  - Throttling of HSM requests, Size of HSM action queue (LU-8626)
- **Policy Engine Improvements**
  - Faster HSM scan with Robinhood improvements
  - Lustre HSM Policy engine (LU-8674)
  - Policy rules for Sparse File. Better Sparse File Handling for HSM (LU-3833, LU-6848)

Lustre HSM Wishlist for HPC (cont..)

Recovery Reliability and Consistency

- **Synchronization with Copy Tool State**
  - Automatic restart of Copy Tool and/or HSM Events related to FS mount, unmount
  - Heart Beats from Copy Tool. State Detection / Recovery from Hang/Crash (LU-5216)
- **Recovery of HSM Actions**
  - Recovery of failed HSM Actions with HP_FLAG_RETRY Implementation
- **Others**
  - Data Consistency: Checkpointing, Locking interfaces
  - Issues with hsm_remove (LU-9255)
  - The Incorrect block size value of the restored files (LU-6848)

Summary

- Review of evolving HSM requirements for HPC
- Evaluation of Lustre HSM w.r.t. HPC requirements
- Distributed Data Mover - An approach for targeting HPC needs for HSM
- Lustre HSM Limitations / Issues
- Wishlist of Lustre HSM Improvements for HPC

Thank You !

Ujjwal Lanjewar
(ujjwal.lanjewar@seagate.com)