Whamcloud

Smart Policies for Data Placement and Storage Tiering of Lustre

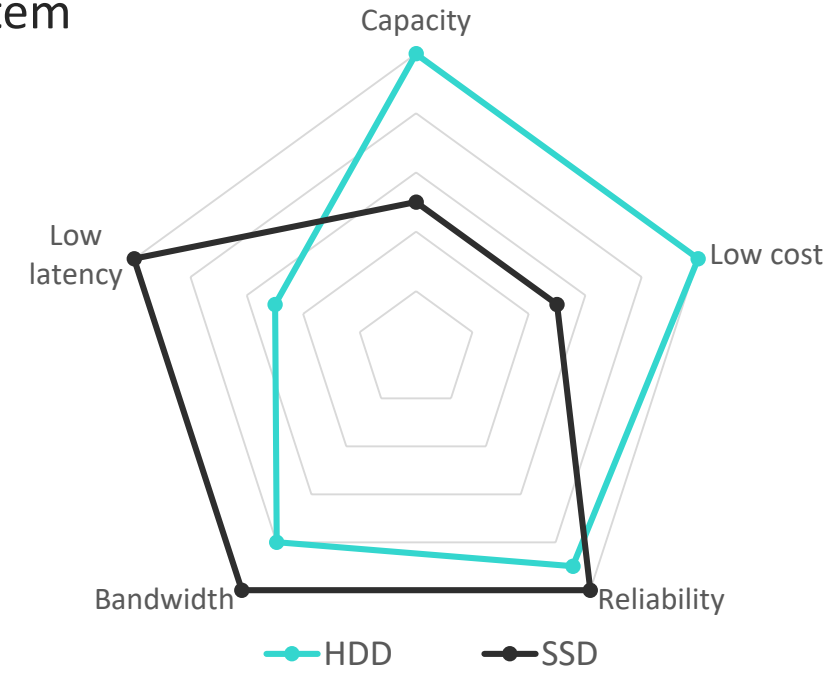
Li Xi

May 2019

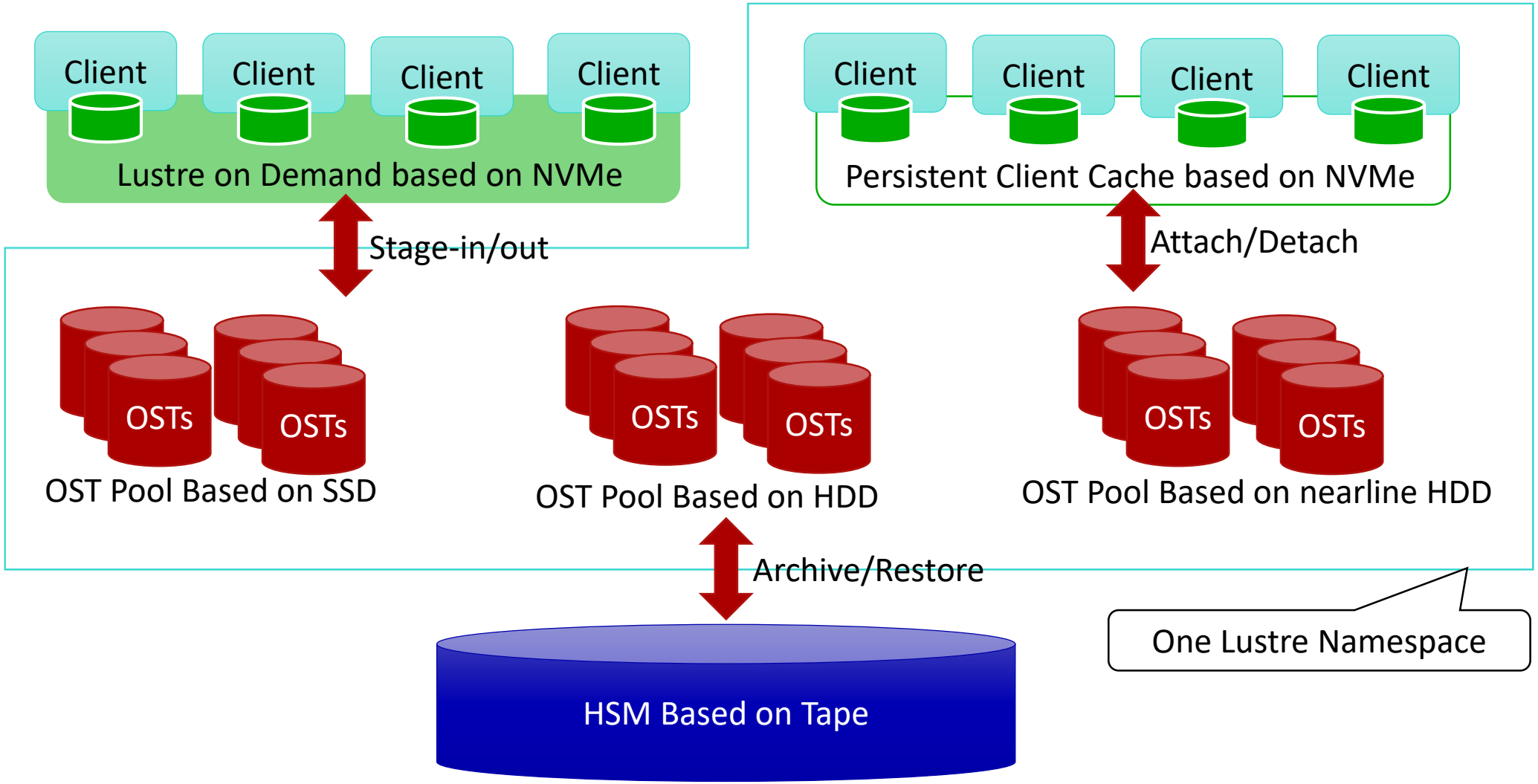
DDN® STORAGE

Background

- Lustre architecture is becoming more heterogeneous
- Heterogeneous media are becoming common in a Lustre file system
  - Different specifications: Capacity, Latency, Bandwidth, Reliability, Cost
  - HDD for big capacity
  - SSD/NVME for quick metadata operations
- Different network bandwidths to storages in a Lustre file system
  - Different network bandwidths from a client to different OSTs
  - Extreme condition: Local OSTs on a Lustre client
- Trend: multiple tiering levels inside Lustre
  - OST/MDT pools with different storage media
  - Persistent Client Cache within the same namespace of main Lustre
  - Hierarchical Storage Management for archive
  - Lustre on Demand for job-level cache outside of global Lustre



Example architecture of a heterogeneous Lustre file system



Motivations

Whamcloud

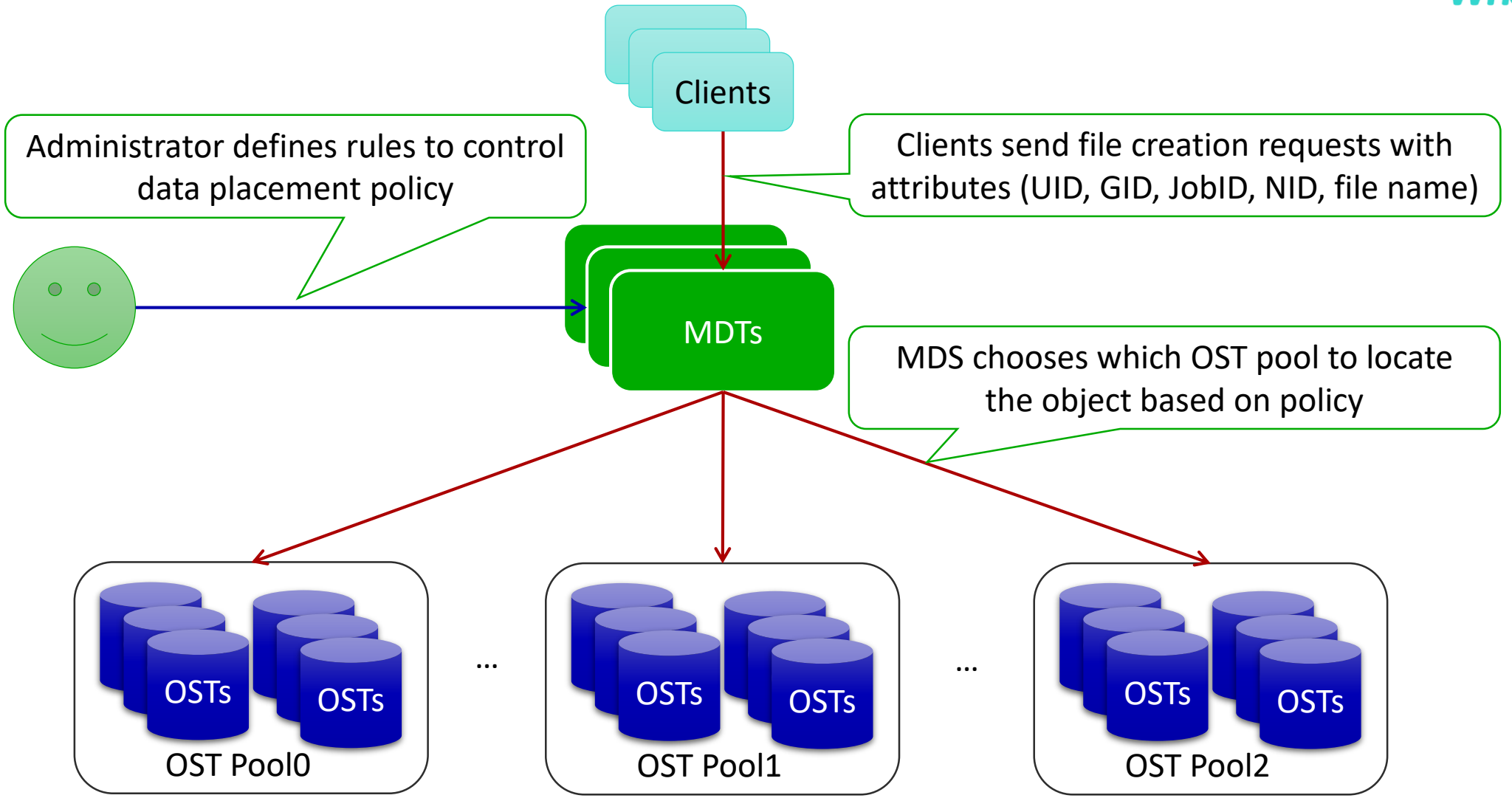
- Storage with higher performance (cost) needs to produce higher benefit
  - The high-performance storage should not be wasted by low-priority jobs
  - The high-performance storage should be carefully allocated between the critical jobs or work-flows
- Better QoS (Quality of service) guarantee
  - Different users/jobs have different priorities and different QoS requirement
  - Lustre administrator should have the tools to provide QoS guarantee
  - Users should have the choices to balance the cost and performance
- Utilize storage locality
  - Allocate file's data to OSTs that have quicker connections to the client
- Promote the entire efficiency of the file system
  - Slow storage for applications with sparse I/Os and quick storage for applications with intensive I/Os

Solution: Data Placement Policy for OST Pools

Whamcloud

- LU-11234: Data Placement Policy (DPP) mechanism for OST pools
  - Patch: https://review.whamcloud.com/#/c/33126
- Classification and separation of storage of different types by using OST pool
  - Virtual separation for flexibility
  - Dynamically configurable
  - Upcoming pool quota for space accounting and limitation
- Data placement policy based on classification of file creation requests
  - Global rules can be configured on MDS and then applied immediately to the whole file system
  - Rules can be based on UID, GID, project ID, JobID, NID, file name patterns
  - Rules can have complex expression with different IDs/filename
  - A series of ordered rules can be defined as a complete policy
  - If a rule matches the file creation request, the object(s) of the file will be created in the specified OST pool
- Re-use of common codes in NRS TBF policy
  - NRS TBF policy is able to classify the I/Os based on UID/GID/NID/JobID/Opcode and their combinations
  - Common policy code is moved from NRS TBF to shared library

Design of DPP



Interfaces of DPP

Whamcloud

- Add a filename extension rule:
  - `lctl set_param lod.*.layout_policy="add rule_sourcecode pool0 suffix={.h .c}"`
- Add a rule based on UID
  - `lctl set_param lod.*.layout_policy=“add rule_vip_users pool1 uid={500 1000}"`
- Add a rule based on GID
  - `lctl set_param lod.*.layout_policy="add rule_vip_groups pool2 gid={500 1000}"`
- Add a rule based on NID
  - `lctl set_param lod.*.layout_policy="add rule_vip_client pool3 nid={10.0.0.200@tcp}"`
- Add a complex rule based on combination of attributes
  - `lctl set_param lod.*.layout_policy= "add rule1 pool4 uid={500}&gid={1000},suffix={.c .h}"`
- Delete a rule
  - `lctl set_param lod.*.layout_policy="del rule1"`

Use Case of Policies based on UID/GID

Whamcloud

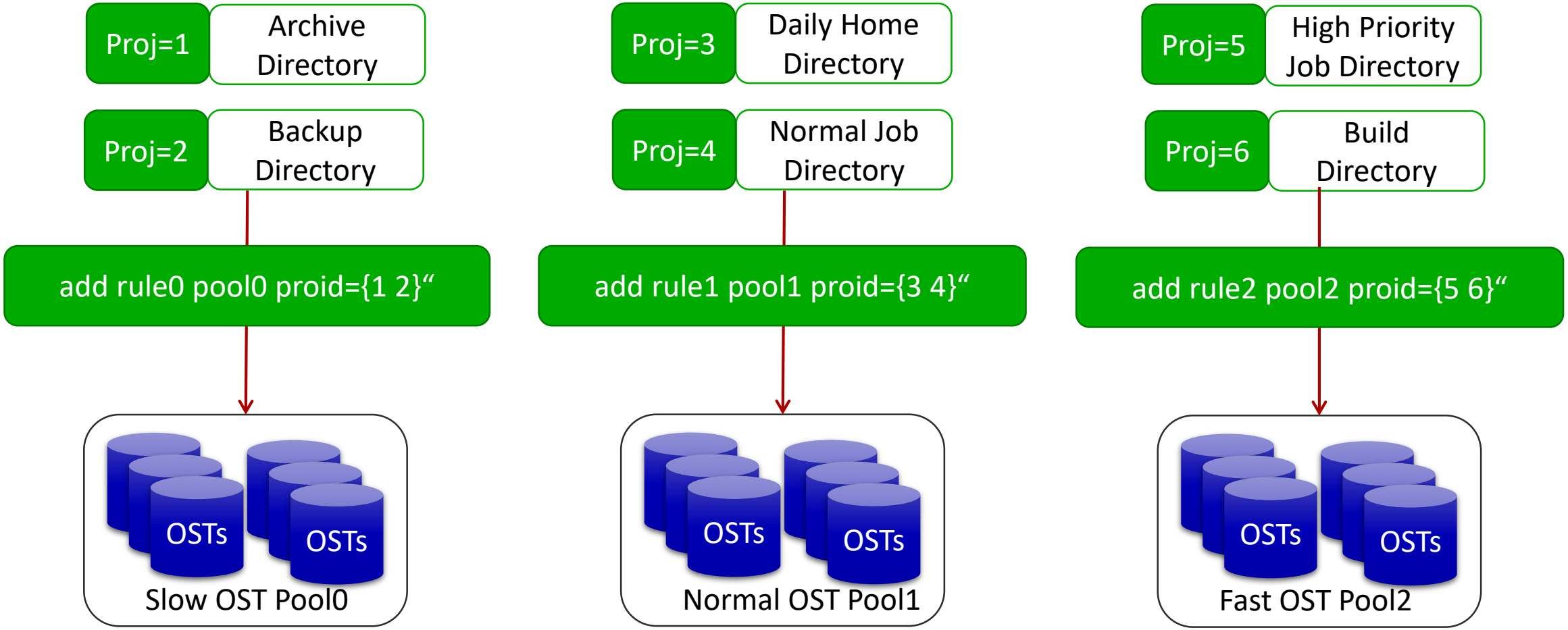
- Users/groups on a shared Lustre are usually charged according to space usage
  - UID/GID quota are commonly used for usage accounting and limitation
- Potential factors of payment
  - In a Lustre filesystem with heterogeneous media, storage types matters along with space usage
  - Fast storage costs more
  - Entire QoS guarantee is even better
  - Users should have a lot of choices to balance between performance and cost
- Pool quota (LU-11023)
  - Enables accounting and usage limitation of users/groups per OST pool
  - DPP + pool quota enable usage limitation of precious OST pools based on UID/GID/ProjID
- QoS of DPP based on UID/GID
  - Better QoS by combining DPP together with NRS TBF

Use Case of Policies based on Project ID

Whamcloud

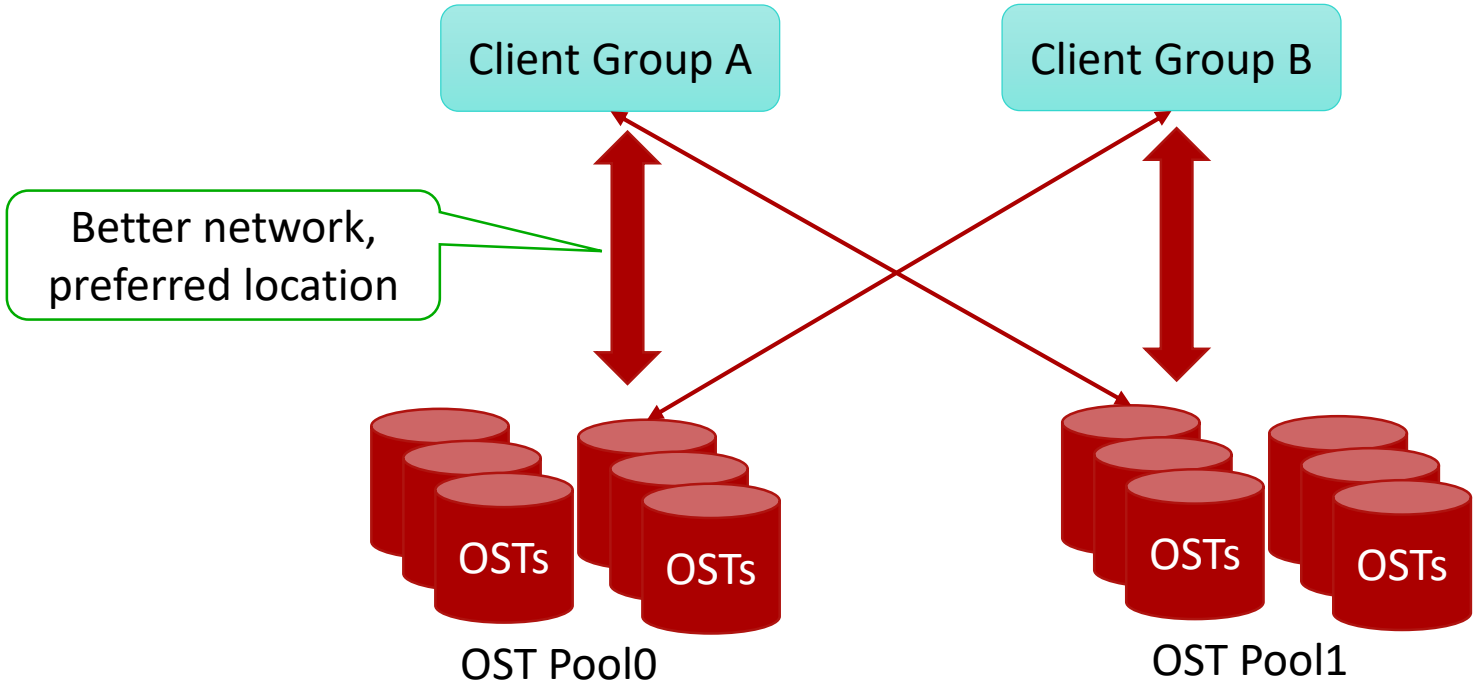
- Project quota for capacity/inode accounting and usage limitation
  - Project ID
    - Inodes that belong to the same project possess the same identification, just like user/group ID
  - Inherit flag
    - An inode flag which defines the behavior related to projects
  - Directory with inherit flag:
    - All newly created sub-files inherit project IDs from the parent
    - No renaming of an inode with different project ID to the directory is allowed (**EXDEV** returned)
    - No hard-links from an inode with different project ID to the directory is allowed (**EXDEV** returned)
- DPP based on project ID
  - A whole directory tree with a given project ID located into a given OST pool
- Examples of use cases:
  - Build directory tree is located to SSD OST pool for accelerated build
  - Archive directory tree is located to cheap nearline OSTs
  - Job directory tree is located to quick OST pool for quick run

Examples of Policies based on Project ID



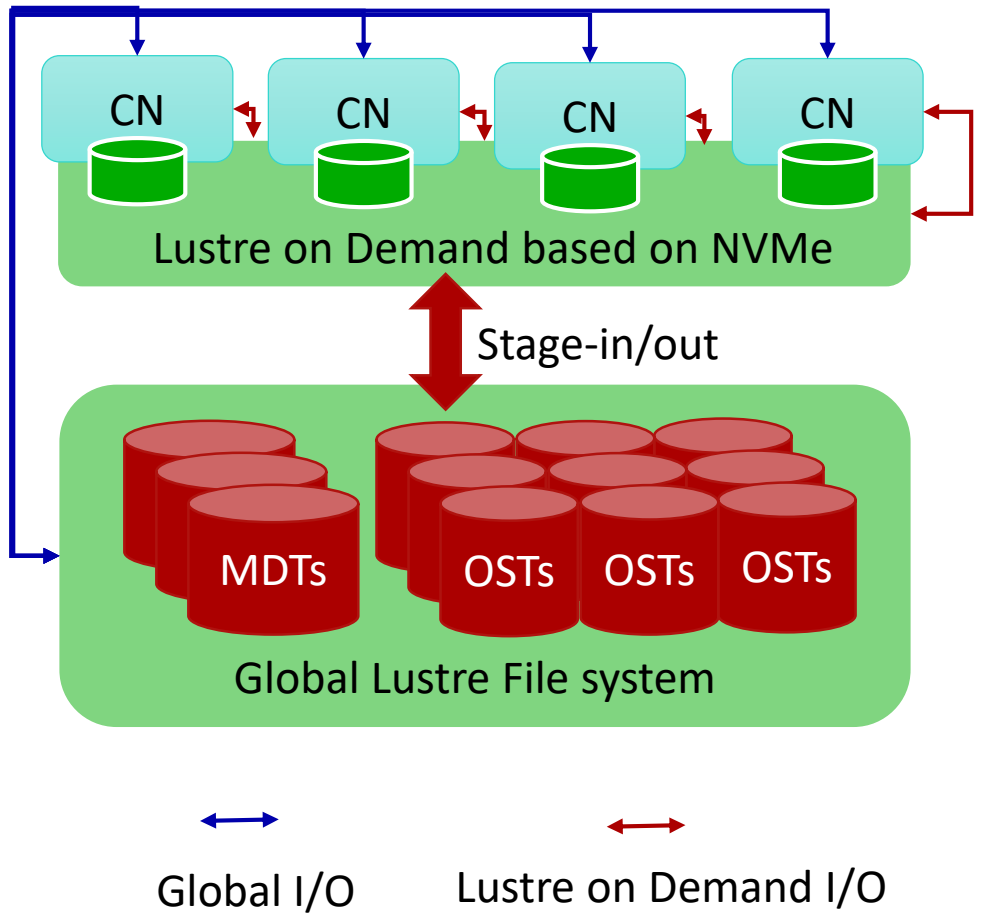
Use Cases of Policies based on NID (1)

- Network bandwidths between a Lustre client and different OSTs might be different
- DPP rules based on NID can be defined to select OSTs with better network connection to the client



Use Cases of Policies based on NID (2)

- Lustre on Demand
  - Fast Lustre inside compute nodes with embedded OSTs/MDTs
  - Integration with job scheduler
  - Transparent and automated stage-in/out
- DPP for Lustre on Demand
  - Each compute node has its own local OST/MDT
  - DPP based on NID is able to locate the files created by a compute node to its local OST/MDT
- Benefit
  - No RPC latency with data locates on local OST/MDT
  - “Infinite” network bandwidth between client and local OST/MDT
- Combination with Data on MDT
  - Performance on local MDTs will be close to local file system both for metadata and data

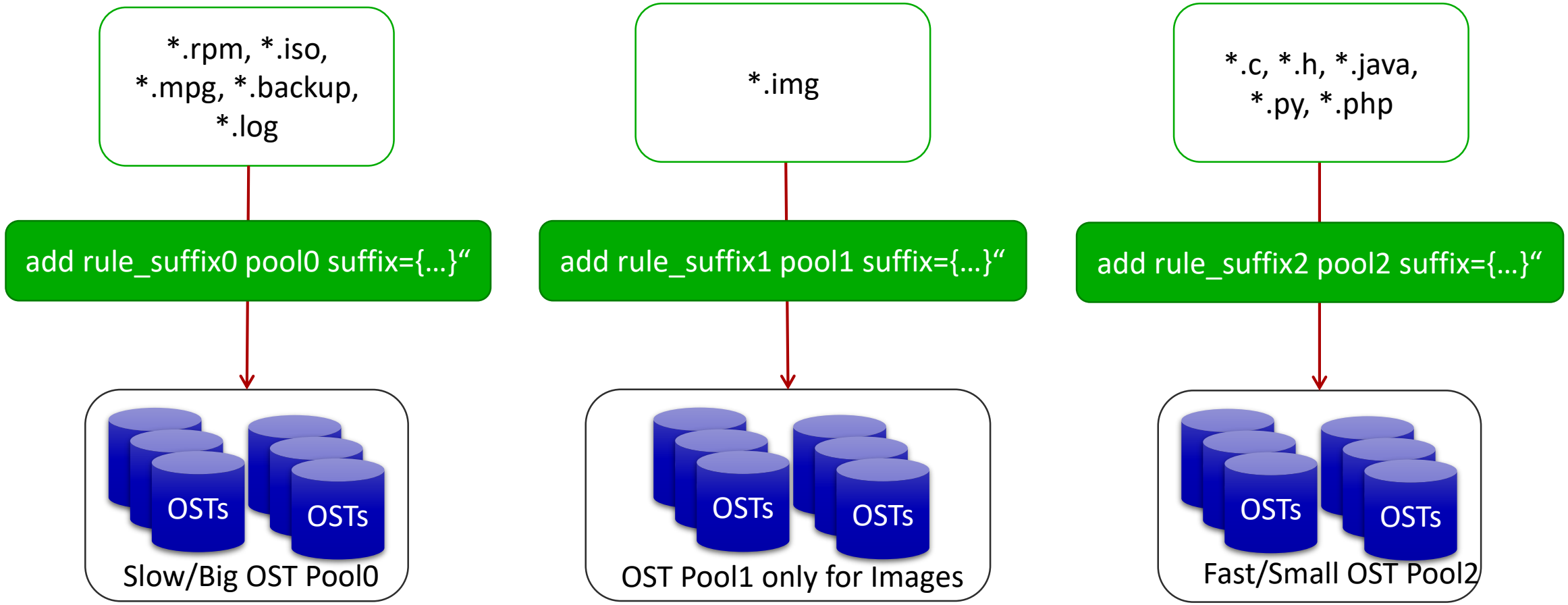


Use Cases of Policies based on Filename Extension

Whamcloud

- Filename extension
  - An important metadata of the file
  - Usually indicates the type of file
- Prediction of file size based on filename extension
  - .rpm: usually has significant size with binaries included, for example:
    - Lustre-2.12.1 server RPMs has 692MB with 34 files, 20MB per file
    - CentOS7 RPMs has 905MB with 436 files, 2MB per file
  - .git: usually contains a log of small files, for example:
    - Lustre has 229MB with 4839 files, 47KB per file
    - Linux kernel has 3.5GB with 65404 files, 53KB per file
  - .iso: usually big files
    - centos/7/isos/x86_64 has 18GB with 6 files, 3GB per file
- Prediction of access pattern based on filename extension
  - .mpg/.mpeg: movies are usually sequentially read
  - .log: log files are usually written with append mode
- Reliability guarantee based on filename extension
  - .c/.h/.java/.py/.php: source code files usually need higher reliability guarantee

Examples of Policies based on Filename Extension



Conclusion

Whamcloud

- We developed a mechanism (DPP) which enables configurable and flexible policies for data placement
- DPP is able to enable a lot of use cases especially for storage tiering of Lustre
- Future work
  - Integrate DPP with different tiers of Lustre, e.g. Lustre on Demand, Persistent Client Cache and HSM
  - Integrate DPP with upcoming pool quota
  - Automatic DPP policies based on heuristic algorithms or machine learning
  - Smart policies for metadata placement of DNE based on MDT pool
  - Smart policies for Progressive File Layout
  - Smart policies for upcoming feature of File Level Redundancy
    - Mirror important files (e.g. source code files) to increase data reliability

Whamcloud

DDN®
STORAGE