Multi-tiering Storage with Lustre

HPSS 2019 User Forum
Carlos Thomaz – Technical Product Management
Customers requiring tiered storage

Orders of Magnitude for Exascale systems

- Capacity requirements: 10’s to 100’s of Petabytes
- Performance requirements: 1 to 10 TiB/sec
- Per node network bandwidth: 100’s of Gbps / node (or 10’s of GiB/sec)

Technology trends

- Flash based storage over fabrics
- New redundancy domains and technologies
- Automation and intelligent tier management
The utopic (still) design

Very Close to memory

Far from memory

One Single transparent Storage Domain

Fabric agnostic

High Performance Flash

High Capacity Storage

Extreme reliability
The current reality

- **Performance Layer (NVMe or SSD)**
- **Capacity Layer (HDD)**
- **Reliability Layer (HDD and/or Tape)**

- **Not totally transparent and still far from memory hierarchy**
- **Fabric dependent**
- **Protocol dependent**

- **Not reliable and fragile data movement mechanisms**
- **Semi automated and complex policies**
- **Limited availability and compatibility**

**Burst Buffer**

**Parallel File System**

**Tape or Cloud based archiving**
What makes the architecture fragile?

• Too many components
• Different levels of integration
• Heterogenous fabric
• Different vendors
• Very complex design
Key Lustre features for tiered storage

**FLR – File Level Replication**

- Lustre level mirroring for files, configured arbitrarily per file/directory
- Mirror NOSYNC flag + timestamp to allow file version/snapshot (LU-11440)
- Mount client directly on OSS without impacting recovery (LU-12722)
- `lfs mirror resync/delete --pool` to simplify tiering (LU-11022)
- Erasure coding adds redundancy without mirror overhead (LU-10911)
- HSM in composite layout (LU-10606)
  - Allow multiple archives per file (S3, tape, disk, ...)
  - Allow partial file restore from archive

<table>
<thead>
<tr>
<th>Replica 0</th>
<th>Flash Object j (PRIMARY, PREFERRED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replica 1</td>
<td>HDD Object k (STALE) delayed resync</td>
</tr>
<tr>
<td>Replica 2</td>
<td>HSM S3 Archive</td>
</tr>
</tbody>
</table>
Key Lustre features for tiered storage
LPCC – Persistent Client Cache (LU-10092)

► Reduce cache latency, improve small/unaligned IOPS, reduce net traffic
► PCC integrates Lustre with persistent per-client local cache storage
  • A local filesystem is created on client device
  • Data is local to the client, no global/visible namespace is provided
  • HSM POSIX copytool fetches whole files into PCC by user command, job script or policy
  • New files created in PCC are also created on Lustre MDS
► Lustre uses local data if in PCC, or normal OST RPCs
  • Further file read/write access “directly” to cache file
  • No data/iops/attributes off client while file in PCC
  • File migrated out of PCC via HSM upon remote access
► Separated shared read vs. exclusive writes cache
► Integrated with DAX for NVRAM cache device
Key Lustre features for tiered storage

DoM improvements

► Convert write locks to read locks without cache flush (LU-10175)
► General usability and stability improvements
► FLR mirror/migrate DoM file (LU-11421)
  • Mirror DoM data to OST object
  • Migrate DoM data to/from OST object
  • No MDT-MDT mirroring yet
► Performance and functional improvements
► Dynamic DoM component size by MDT free space (LU-12785)
► Merge data write with MDS_CLOSE RPC (LU-11428)
► Cross file data prefetch via statahead (LU-10280)
► Allow MDT-only file system (LU-10995)
Key Lustre features for tiered storage

Flash storage improvements

► Reduced server CPU overhead to improve small flash IOPS (LU-11164)
  • Reduced CPU usage translate directly to improved IOPS

► Avoid Page caching on ldiskfs flash OSS (LU-11347)
  • Avoids CPU/lock overhead/lock for page eviction

► TRIM support for flash on ldiskfs (LU-11355)
  • Release unused blocks of filesystem via fstrim

► Self Extending layouts (LU-10070, Cray)
  • Avoids out-of-space in the middle of files
  • Good for PFL with smaller flash OSTs than disk OSTs

► Continued reductions of overhead and latency
  • Improve small, unaligned and interleaved writes
Glueing all together

- Performance Layer (NVMe or SSD)
- Capacity Layer (HDD)
- Reliability Layer (HDD and/or Tape)

- Stratagem
- Parallel File System
- Burst Buffer
- Tape or Cloud based archiving
What is Stratagem

- Policy Engine for Data Automation
- Doesn’t rely on external mechanisms
- Tightly integrated with Lustre
- Fast and scalable data scanning
- API driven for external integration such as HPSS copy tool
- Integrated with EXA5 product line
Stratagem Hot Pools USE CASE

Data is written to flash (maximum performance)

Data Object

Lustre FLR delayed sync replicate the object on HDDs

Stratagem analyze the data and applies policies and trigger actions
Stratagem Purging with Trash-bin USE CASE

Data Object allocated on flash and replicated but not in use.

- Stratagem scans the file system and located objects
- Stratagem triggers an object copy/move to “trash bin”
- After a period of time on trash bin if the object isn’t reclaimed another policy is triggered or object is delete

Flash OSTs

Capacity HDD OSTs

Long Term Retention storage
Summary

► Most of the new features has been/are being designed to address multi-tiering and hierarchical storage problems
  • Performance
  • Data hierarchy
  • Cost and economics of Flash / HDD / long term retention storage
► Not all the problems can be solved with file system features thus requiring tools to orchestrate the data flow
► Stratagem is one of the tools that integrate the file system features within the Data Management workflow