Providing Persistent Client Caching Services with a Lustre Global Namespace

Lingfang Zeng (曾令仿)
Wuhan National Laboratory for Optoelectronics (WNLO)
Huazhong University of Science and Technology (HUST)
With Contributions from

- Yingjin Qian, Xi Li, Shuichi Ihara, Carlos Aoki Thomaz, and Shilong Wang @ DDN
- Andreas Dilger @ DDN Whamcloud
- Tim Süß, and André Brinkmann @ JGU
- Wen Cheng, Chunyan Li, Fang Wang, and Dan Feng @ HUST
ACKNOWLEDGMENTS
01 BACKGROUND
PROBLEM & TERMINOLOGY & OBJECTIVES
Hierarchical Storage Management (HSM)

HPC workloads were too big to be stored only on flash

https://semiengineering.com/a-new-memory-contender/
https://storageswiss.com/2018/01/10/enterprise-needs-to-learn-from-hpc-environments/
HSM Tier

- **Compute servers**
  - HBM
  - NVRAM/SCM

- **Performance storage**
  - DRAM
  - SSD
  - (performance HDD)

- **Capacity storage**
  - DRAM
  - Capacity HDD

Lang’s Law: the more tiers, the more tears
Industry and Academic Solutions

- Andrew File System [TOCS’88, CMU]
- Coda File System [TOCS’88, CMU]
- FS-Cache [Linux Symposium’06, Red Hat]
- BWCC [CLUSTER’12, CAS]
- Nache [FAST’07, RU & IBM]
- Panache [FAST’10, IBM]
- Mercury [MSST’12, NetApp]
- GPFS’ LROC [IBM]
- TRIO [CLUSTER’15, FSU & ORNL & AU]
- BurstFS [SC’16, FSU & LLNL]
- MetaKV [IPDPS’17, FSU & LLNL]
- Dmcache [TOCS’88, CMU]
- Xcachefs [SBU, 2005]
- FlashCache [CASES’06, UM]
- Bcache [LWN, 2010]
Related Work

- Read-only cache
- Tolerate I/O failures in cache

- File system meta-operations (both cache and source)

Reference: Howells, FS-Cache: A Network Filesystem Caching Facility, Red Hat UK Ltd.
Related Work

Reference: Eshel+, Panache: A parallel file system cache for global file access, FAST'10
Wang+, An ephemeral burst-buffer file system for scientific applications, SC'16
Lustre File System

Management Target (MGT)  Metadata Targets (MDTs)  SAS Object Storage Targets (OSTs)  SATA SMR Archive OSTs

Metadata Servers (~10's)  Metadata Servers (~100,000+)  NVMe OSTs/LNet routers on client network "Burst Buffer"

High Performance Data Network  LPCC Agent (Copytool)  Policy Engine (Robinhood)

Figure based on Andreas Dilger's Lustre User Group 2018 presentation: Lustre 2.12 and beyond (see http://opensfs.org/lug-2018-agenda/)
HSM Tier

- Shared
  - DDN IME @ ICHEC
  - Cray Trinity @ LANL

- Data plane
- Control plane
- Erasure coding
HSM Tier

- **Shared**
  - DDN IME @ ICHEC
  - Cray Trinity @ LANL

- **Server-side**
  - Seagate Nytro NXD @ Sanger

- Storage-side flash acceleration
- I/O histogram
- Performance statistics
- Dynamic flush
HSM Tier

- Shared
  - DDN IME @ ICHEC
  - Cray Trinity @ LANL
- Server-side
  - Seagate Nytro NXD @ Sanger
- Client-side
  - Intel/Cray Aurora (A21) @ Argonne National Laboratory?
  - Lustre Persistence Client Cache (LPCC)
Lustre’s DLM and Layout Lock

- Distributed lock manager (DLM)
  - Data and metadata consistency
  - A separate namespace
- Exclusive mode (EX) lock
- Concurrent read mode (CR) lock
- \(\text{L.Gen}\) field
Lustre HSM

- **Agents** – Lustre file system clients running Copytool
- **Coordinator** – Act as an interface between the policy engine, the metadata server (MDS) and the Copytool

Key Idea

- Logical two-tier (with physical multitier)
  - Simple and efficient architecture (memory vs. disk)
- A global namespace
  - Space efficient
- Latencies and lock conflicts can be significantly reduced
- Caching reduces the pressure on (OSTs)
  - small or random I/Os can be regularized to big sequential I/Os and temporary files do not even need to be flushed to OSTs.
HIERARCHICAL PERSISTENT CLIENT CACHING
Overview of LPCC Architecture
Overview of LPCC Architecture

- Management Node
- Lustre Clients (~50,000)
- OSSes (~1,000)
- MDSes (~10)
- MGT
- MDTs
- OSTs

Management Network

High Performance Data Network

HSM Storage Network

Archive

Policy Engine (Robinhood)

Copytool

HSM Agent (Copytool)

OSSes (~1,000)

OSTs

MDT

MGT

Copytool

Coordinator

OSTs

OSTs

OSTs
03 IMPLEMENTATION
RW-PCC & RO-PCC & RULE-BASED TRIGGERING & POLICY ENGINE
Lustre Read-Write PCC Caching (attach)

Client

Attach

Exclusive Open

Create PCC mirror file

Copy data from OSTs to PCC (archive)

Close with Intent

MDT

Take EX Layout Lock

Release File from Lustre

Set HSM released and Archived

Inc L.Gen

Release Layout Lock

Close reply piggyback with L.Gen

Set PCCI.LayoutGen = L.Gen

Attach Finished
Lustre Read-Write PCC Caching (restore)

- Notify all clients having cached the layout to invalidate their layouts.
Lustre Read-only PCC Caching (attach)

Client → MDT

Attach

Layout Write Intent RPC

RPC Reply with L.Gen → yes

L.rdonly?

No →
- Take EX L. Lock
- Set L.rdonly
- Inc L.Gen
- Release EX L. Lock
- Grant CR L. Lock to the client

Yes →

RPC Reply with granted L. Lock and L.Gen

Set PCCI.LayoutGen = L.Gen

Copy Data from Lustre into PCC file

Attach Finished
Lustre Read-only PCC Caching (I/O flow)
Rule-based Persistent Client Caching

- Different user, groups, and projects or filenames
  - E.g. (projid={500,1000} & fname=*.h5), (uid=1001)
- Quota limitation
  - Cache isolation
- Auto LPCC caching mechanism
Cache Prefetching and Replacement

- Policy engine
  - Manage data movement
- Lustre changelogs
  - Periodic prefetching decision
- LRU and SIZE
04 EVALUATIONS EXPERIMENT & RESULTS
Evaluation Setup

- CentOS 7 Linux (3.10.0) and Lustre (2.11.53)
- All client nodes included
  - An Intel Xeon E5-2650 processors with 128GB of memory
  - 512GB Samsung 840 PRO series SSD as LPCC cache (ext4-based LPCC)
- Lustre OSS DDN SFA14KXE with 10 OSTs (ext4-based ldiskfs)
- MDS Toshiba 200GB SSD (ext4-based ldiskfs)
- “stripe=n” means file data is striped over n OSTs
- Lustre Data on MDT (DoM)
  - To improve small file performance by allowing the data on the MDT
- FS-Cache mechanism
Benchmark Tools

- fio
- IOR (file-per-processor (FPP))
- mdtest
- filebench
- HACC I/O
  - HPC application simulation in FPP mode
- Compliebench
  - Simulate kernel compiles with target to metadata and small file operations
Single Thread Performance

**Write Bandwidth (MiB/s)**

- ext4
- RW-PCC
- Idiskfs
- Lustre

**Read Bandwidth (MiB/s)**

- ext4
- RW-PCC
- Idiskfs
- Lustre
RW-PCC Scalability Evaluation
Metadata Performance

RW-PCC achieves the best
Small Files with Various Size

- File creation (IOPS)
  - stripe=1
  - stripe=4
  - DoM
  - RW-PCC

- File read (IOPS)

- File removal (IOPS)
File create with no data is slow

But, the better small write performance on local SSDs compensated for this and allowed a speedup compared to DoM and standard lustre

<table>
<thead>
<tr>
<th>Operation</th>
<th>File creation</th>
<th>File read</th>
<th>File removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>stripe=1</td>
<td>2776</td>
<td>3614</td>
<td>2974</td>
</tr>
<tr>
<td>stripe=4</td>
<td>2610</td>
<td>3552</td>
<td>2573</td>
</tr>
<tr>
<td>DoM</td>
<td>2793</td>
<td>3478</td>
<td>3126</td>
</tr>
<tr>
<td>RW-PCC</td>
<td>1714</td>
<td>3591</td>
<td>3271</td>
</tr>
</tbody>
</table>
Small File for Compilebench
**Read Performance**

- File data is read for the first time and loaded into cache
- By repeating the test immediately after the “Cold” one
- Directly from the persistent cache after cleaning all page caches
RO-PCC Scalability Evaluation

- RO-PCC performance in “Warm” and “Cache” state
- Scale nearly linearly with the increasing client number

<table>
<thead>
<tr>
<th>Client count</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold (MiB/s)</td>
<td>478</td>
<td>688</td>
<td>718</td>
<td>1389</td>
</tr>
<tr>
<td>Warm (MiB/s)</td>
<td>4374</td>
<td>8746</td>
<td>17520</td>
<td>34943</td>
</tr>
<tr>
<td>Cache (MiB/s)</td>
<td>521</td>
<td>1042</td>
<td>2074</td>
<td>4029</td>
</tr>
</tbody>
</table>
Metrics Statistic

“SIZE” evicts the least number of cached file.
File Hit Rate

“LRU” has the highest hit rate
Summary

- A global namespace
  - Space efficient
  - Simple and transparent
- Less overhead, and network latencies and lock conflicts significantly reduced
- Simpler I/O stack: no interference with I/Os from other clients
- Small requirements on the HW inside the client nodes (SSD/HDD)
- LPCC reduces the pressure on the OSTs
Thanks for your attention!