NVRAM-oriented Lustre Persistent Cache on Client

Lingfang Zeng, Xi Li*, and Wen Cheng
Wuhan National Laboratory for Optoelectronics (WNLO)
Huazhong University of Science and Technology (HUST)
*DDN / Whamcloud
With Contributions from

- Yingjin Qian, Shuichi Ihara, Carlos Aoki Thomaz, and Shilong Wang @ DDN
- Andreas Dilger @ DDN/Whamcloud
- Tim Süß, and André Brinkmann @ JGU
- Chunyan Li, Fang Wang, and Dan Feng @ HUST
- LPCC
- SC2019
Outline

BACKGROUND
PROBLEM & TERMINOLOGY & OBJECTIVES

METHODS
HIERARCHICAL PERSISTENT CLIENT CACHING

IMPLEMENTATION
RW-PCC & RO-PCC & RULE-BASED TRIGGERING & POLICY ENGINE

EVALUATIONS
EXPERIMENT & RESULTS
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
Problems

- **Performance**
  - Speed defines the winner
  - Cache

- **Utilization rate (Lustre client devices)**
  - NVMe
  - Flash-based SSD
  - NVRAM/SCM

- **Data consistency**

- **Transparency**
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

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 (Erasure Coded)

Metadata Servers (~10's)  NVMe MDTs on client net
NVMe OSTs/LNet routers on client network "Burst Buffer"

High Performance Data Network

LPCC Agent (Copytool)

Management Node  Lustre Clients (~100,000+)

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

- **Client-side**
  - Lustre Persistence Client Cache (LPCC)

- **Server-side**
  - Seagate Nytro NXD @ Sanger
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
- \textit{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.
02 METHODS
HIERARCHICAL PERSISTENT CLIENT CACHING
Overview of LPCC Architecture

- MDT
- dir1
- file1
- file2
- file3
- file4
- file5

1. Metadata I/O path
2. HSM restore request
3. LPCC attach

Data Object creation

1. Normal I/O path
2. Data archive
3. Data restore

Client

Agent (copytool)

LPCC (HSM)

OSTs

OSS
I/O Path

User Application

VFS

Llite

Normal I/O

LOV

OSC-1 ... OSC-N

PTR-RPC

LNET

Cache I/O

NVM-FS

NVM
03 IMPLEMENTATION
RW-PCC & RO-PCC & RULE-BASED TRIGGERING & POLICY ENGINE
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

L.rdonly?

No

Take EX L. Lock

Set L.rdonly

Inc L.Gen

Release EX L. Lock

Grant CR L. Lock
to the client

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. \((\text{projid}=\{500,1000\} \& \text{fname}=*.h5), (\text{uid}=1001)\)
- 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

- Simulate DRAM to NVRAM
  - DRAM (28GB)/NVM(100GB)
  - Write latency 200ns
- Clients(8)
- OSS(3)
- MDS(1)

<table>
<thead>
<tr>
<th>Server</th>
<th>Inspur SA5248L</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel(R) Xeon(R) CPU E5-2620, 2.00GHz</td>
</tr>
<tr>
<td>DRAM</td>
<td>128GB</td>
</tr>
<tr>
<td>SSD</td>
<td>Kingston SA400S37/240G</td>
</tr>
<tr>
<td>HDD</td>
<td>SAS Disks</td>
</tr>
<tr>
<td>OS</td>
<td>Centos-7.5</td>
</tr>
<tr>
<td>Kernel</td>
<td>3.10.0-862.6.3</td>
</tr>
<tr>
<td>Network</td>
<td>1GB Ethenet</td>
</tr>
<tr>
<td>Lustre</td>
<td>Lustre 2.11.53</td>
</tr>
<tr>
<td>Fio</td>
<td>fio-3.1</td>
</tr>
<tr>
<td>Filebench</td>
<td>filebench 1.5-alpha3</td>
</tr>
<tr>
<td>IOR</td>
<td>IOR-3.2.0</td>
</tr>
</tbody>
</table>
Single Client Read Performance (fio)
Single Client Read Performance (fio)
Single Client **Read** Performance (fio)
Single Client **Write** Performance (fio)
Single Client **Write** Performance (fio)
Single Client **Write** Performance (fio)
## Write Performance (RW-PCC, filebench)

<table>
<thead>
<tr>
<th>Workload</th>
<th>Average file size</th>
<th>Num. of Files</th>
<th>Read-write ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileserv</td>
<td>128KB</td>
<td>0.3 Million</td>
<td>1:2</td>
</tr>
<tr>
<td>webserv</td>
<td>64KB</td>
<td>0.3 Million</td>
<td>10:1</td>
</tr>
<tr>
<td>varmail</td>
<td>32KB</td>
<td>0.3 Million</td>
<td>1:1</td>
</tr>
</tbody>
</table>

### Fileserver:

NVRAM-LPCC / SSD-LPCC vs. Lustre

Throughput:

- 4x
- 2x

### Webserver:

NVRAM-LPCC / SSD-LPCC Vs. Lustre

- 9.2x
- 8.3x

### Varmial:

NVRAM-LPCC / SSD-LPCC vs. Lustre

- 1.94x
- 1.5x
Read Scale Test (RW-PCC, IOR)
Write Scale Test (RW-PCC, IOR)
**RO-PCC Scalability (I/O block size=1MB)**

![Graph showing RO-PCC Scalability](image)

- NVRAM –LPCC vs. SSD-LPCC (**12.52x**),
- NVRAM –LPCC vs. Lustre (**44.20x**)

Data Size (G): 1G, 2G, 4G, 8G, 16G, 32G, 64G

Read Bandwidth (MB/s): 4000, 5000, 6000
RO-PCC Scalability

- Lustre
- SSD-LPCC
- NVRAM-LPCC

Read Bandwidth (MB/s) vs. Num. of the Clients
Summary

- The performance of the cache medium has a great influence on the LPCC performance
  - Kingston SA400S37/240G SSD (proposed)
  - 512GB Samsung 840 PRO SSD (SC19)
- High performance based on NVRAM
  - Less overhead, and network latencies and lock conflicts significantly reduced
  - Reduce the pressure on the OSTs
  - No page cache
  - Optimized flush (Load/Store)
  - NVRAM-LPCC >> pm-ext4-LPCC
Thanks!

NVRAM-oriented Lustre Persistent Cache on Client

Lingfang Zeng*, Xi Li#, and Wen Cheng*
*Wuhan National Laboratory for Optoelectronics (WNLO)
*Huazhong University of Science and Technology (HUST)
#DDN / Whamcloud