Accelerating Lustre with SSDs and NVMe

James Coomer, DDN
DDN ExaScaler Software Components

- **Data Management**
  - Fast Data Copy
    - Tape
    - S3 (Cloud)
    - Object (WOS)
    - ExaScaler Data Management Framework

- **Management and Monitoring**
  - DDN DirectMon
  - DDN ExaScaler Monitor
    - Intel IML
  - DDN ExaScaler

- **Global Filesystem**
  - DDN Clients
    - NFS/CIFS/S3
    - DDN IME
    - Intel Hadoop
    - DDN Lustre Edition with L2RC
    - OFD Read Cache Layer

- **Local Filesystem**
  - Idiskfs
    - Intel DSS
    - OpenZFS
    - btrfs

- **Storage Hardware**
  - DDN Block Storage with SFX Cache
    - 3rd Party HW

Level 3 support provided by:

- DDN
- DDN & Intel
- Intel HPDD
- 3rd Party
### DDN | ES14K
Designed for Flash and NVMe

**Configuration Options**
- 72 SAS SSD or 48 NVMe
- SSDs or HDDs only
- HDDs with SSD caching
- SSDs with HDD tier

**Connectivity**
- FDR/EDR
- OmniPath
- 40/100GbE

**Industry Leading Performance in 4U**
- Up to 40 GB/sec throughput
- Up to 6 million IOPS to cache
- Up to 3.5 million IOPS to storage
- 1PB+ capacity (with 16TB SSD)
- 100 millisecond latency
ES14K Architecture

SFA14KE (Haswell)

SFA14KEX (Broadwell)
Why SSD Cache?
Don't blow the power/space/management with spindles

SSDs still pricey... So
► Optimise Data for SSDs
► Optimise SSDs for Data
Data residing on SSD

IO acceleration from SSD

Data benefitting from SSD

all data
all data

- Data residing on SSD
- IO acceleration from SSD
- Data benefitting from SSD
SSD Options

1. All SSD Lustre
   - Lustre HSM for Data Tiering to HDD namespace
   - Generic Lustre I/O
   - Millions of Read and Write IOPS

2. SFX
   - Block-Level Read Cache
   - Instant Commit, DSS, fadvice()
   - Millions of Read IOPS

3. L2RC
   - OSS-level Read cache
   - Heuristics with FileHeat
   - Millions of Read IOPS

4. IME
   - I/O level Write and Read Cache
   - Transparent + hints
   - 10s of Millions of Read/Write IOPS
1. Rack Performance: Lustre

IOR File-per-Process (GB/s)

- Write: 400 GB/s
- Read: 350 GB/s

4k Random Read IOPS

- Read: 5,000,000 IOPS

Capacity

- up to 4PB Flash Capacity

I/O Performance

- 4 Million IOPs

350GB/s Read and Write (IOR)
2. SFX & ReACT – Accelerating Reads

Integrated with Lustre DSS

DSS

OSS

SFX Tier

HDD Tier

DRAM Cache

SFX API

Small Rereads

Small Reads

Large Reads

Cache Warm
2. 4 KiB Random I/O

The chart shows the IOPS (Input/Output Operations Per Second) for various conditions:

- **No SFX/SSD Metadata**
  - First Time I/O: 15,587 Read, 14,486 Write
  - Second Time I/O: 14,184 Read, 14,984 Write

- **No SFX Metadata Mix**
  - First Time I/O: 17,070 Read, 13,008 Write

- **With SFX / Metadata Mix**
  - First Time I/O: 17,070 Read, 13,008 Write

- **SFX Read Hit**
  - First Time I/O: 174,344 Read, 13,001 Write

The chart indicates a significant decrease in IOPS after the first time I/O, especially in the Write operations, with an increase in Read operations for both First and Second Time I/O.
3. Lustre L2RC and File Heat

OSS-based Read Caching
- Uses SSDs (or SFA SSD pools) on the OSS as read cache
- Automatic prefetch management based on file heat
- File-heat is a relative (tunable) attribute that reflects file access frequency
- Indexes are kept in memory (worst case is 1 TB SSD for 10 GB memory)
- Efficient space management for the SSD cache space (4KB-1 MB extends)
- Full support for ladvice in Lustre
3. File Heat Utility

- tune the arguments of file heat with proc interfaces
  
  /proc/fs/lustre/heat_period_second
  /proc/fs/lustre/heat_replacement_percentage

- Utils to get file heat values: `lfs heat_get <file>`

- Utils to set flags for file heat:
  

- Heat can be cleared by: `lfs heat_set --clear`

- Heat accounting of a file can be turned off by: `lfs heat_set --off`

- Heaps on OSTs which can be used to dump lists of FIDs sorted by heat:
  
  ```
  [root@server9-Centos6-vm01 cache]# cat /proc/fs/lustre/obdfilter/lustre-OST0000/heat_top
  [0x200000400:0x1:0x0] [0x100000000:0x2:0x0]: 0 740 0 775946240
  [0x200000400:0x9:0x0] [0x100000000:0x6:0x0]: 0 300 0 314572800
  [0x200000400:0x8:0x0] [0x100000000:0x5:0x0]: 0 199 0 208666624
  [0x200000400:0x7:0x0] [0x100000000:0x4:0x0]: 0 100 0 104857600
  [0x200000400:0x6:0x0] [0x100000000:0x3:0x0]: 0 100 0 104857600
  ```
3. Random Read Performance with L2RC

4KB Random Read IOPS (HDD/SSD based OST vs. OST & L2RC)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>IOPS (ops/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 x Spindle Drive</td>
<td>13,739</td>
</tr>
<tr>
<td>80 x Spindle Drive</td>
<td>26,064</td>
</tr>
<tr>
<td>160 x Spindle Drive</td>
<td>38,688</td>
</tr>
<tr>
<td>160 x Spindle Drive with L2RC</td>
<td>389,232</td>
</tr>
<tr>
<td>4 x OST on SSD</td>
<td>416,994</td>
</tr>
<tr>
<td>4 x SSD(RAID 1) Raw device</td>
<td>440,428</td>
</tr>
</tbody>
</table>

10x improvement
Lightweight IME client intercepts application I/O. Places fragments into buffers + parity

IME client sends fragments to IME servers

IME servers write buffers to NVM and manage internal metadata

IME servers write aligned sequential I/O to SFA backend

Parallel File system operates at maximum efficiency
4. IME Write Dataflow

1. Application issues fragmented, misaligned IO

2. IME clients send fragments to IME servers

3. Fragments are sent to IME servers and are accessible via DHT to all clients

4. Fragments to be flushed from IME are assembled into PFS stripes

5. PFS receives complete aligned PFS stripe
4. IME Erasure Coding

- Data protection against IME server or SSD Failure is optional
  - (the lost data is "just cache")
- Erasure Coding calculated at the Client
  - Great scaling with extremely high client count
  - Servers don't get clogged up
- Erasure coding does reduce useable Client bandwidth and useable IME capacity:
  - 3+1: 56Gb $\rightarrow$ 42Gb
  - 5+1: 56Gb $\rightarrow$ 47Gb
  - 7+1: 56Gb $\rightarrow$ 49Gb
  - 8+1: 56Gb $\rightarrow$ 50Gb
4. Rack Performance: IME

- **500GB/s Read and Write**
- **768TB Flash Capacity**
- **50 Million IOPs**

**IOR File-per-Process (GB/s)**
- Write: ~600 GB/s
- Read: ~550 GB/s

**4k Random IOPS**
- Write: ~100,000,000 IOPS
- Read: ~5,000,000 IOPS
Summary

• SSDs can today be seamlessly introduced into a Lustre Filesystem
  – Modest investment in SSDs
  – Intelligent policy-driven data moves the most appropriate blocks/files to SSD cache
  – Block level and Lustre Object Level data placement schemes

• IME is a ground-up NVM distributed cache which adds
  – Write Performance optimisation (not just read)
  – Small, random I/O optimisations
  – Shared (many-to-one) file optimisations
  – Improved SSD lifetime
  – Back-end Lustre IO optimisation