DESIGNING HIGH-PERFORMANCE STORAGE TIERS

INTEL® ENTERPRISE EDITION FOR LUSTRE* SOFTWARE AND INTEL® NON-VOLATILE MEMORY EXPRESS (NVME) STORAGE SOLUTIONS

Designed specifically for high performance computing (HPC), the open source Lustre* parallel file system is one of the most powerful and scalable data storage systems currently available. It is in widespread use today in supercomputing scenarios where high-performance and enormous storage capacity is required. Lustre is currently running on 60% of the Top 100 clusters in the world.[1]

Intel® Solutions for Lustre* software can be further enhanced by employing Intel® SSD Data Center Family for Non-Volatile Memory Express (NVMe) to deliver a cost-effective, high-performance storage solution for specific HPC use cases.

This paper will explore the application of Intel® SSD DC P3700 Series to provide an efficient high-performance storage tier in Lustre for checkpointing applications.

"Lustre* is designed to achieve the maximum performance and scalability for POSIX applications that need outstanding streamed I/O."

HPC File System Options and New Trends

Lustre is a Portable Operating System Interface (POSIX) object-based file system that splits file metadata (such as the file system namespace, file ownership, and access permission) from the actual file data and stores them on different servers. File metadata is stored on a metadata server (MDS), and the file data is split into multiple objects and stored in parallel across several object storage targets (OST). The same Lustre file system can run on different heterogeneous networks, thanks to the Lustre network, a very powerful and fast abstraction layer.

Lustre is designed to achieve the maximum performance and scalability for POSIX applications that need outstanding streamed I/O. Users can create a single POSIX name space of up to 512PB and very large files up to 32PB. Several sites are currently in production with a Lustre cluster scaling beyond 1TB/sec and very high metadata operation rates (800K stats/sec).

Scale-up file systems (like NFSv3) work by designating a single node to function as the I/O server for the storage cluster. All I/O data reads and writes go through that single node. While this system is relatively simple to manage in a single cluster deployment, pushing all of an enterprise's I/O through one server node quickly presents a bottleneck for data-intensive workloads and for workloads that need a high number of threads/processes.

When scaling up an NFS-based environment, each of the disparate NFS clusters must be managed individually, adding to data bottlenecks, as well as management overhead and costs.

"The largest-scale HPC systems are moving parallel file systems to their limits in terms of aggregate bandwidth and numbers of clients. To further sustain the scalability of these file systems and manage specific use cases, Intel is exploring various storage system designs."

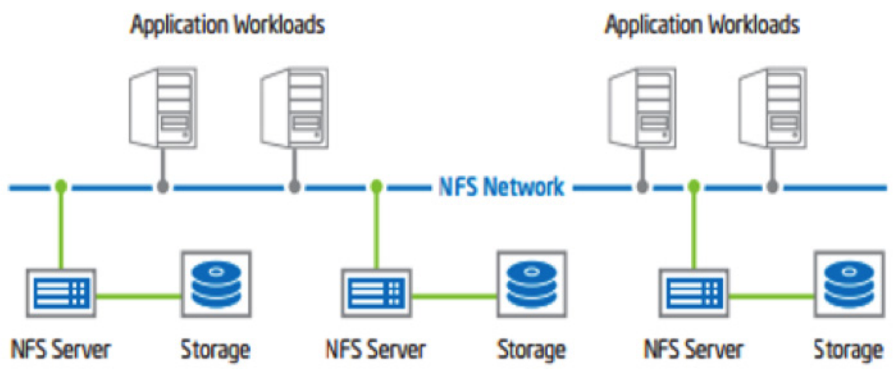


Figure 1. **Typical NFS file system configuration**

The largest-scale HPC systems are pushing parallel file systems to their limits in terms of aggregate bandwidth and numbers of clients. To further sustain the scalability of these file systems and manage specific use cases, Intel is exploring various storage system designs. One proposed storage system design integrates a tier of non volatile memory devices into the Lustre file system to absorb application I/O requests.

Usage models for fast checkpointing with standard techniques:

- A new application's latest checkpoint is loaded into the burst buffer to be available for the new job to begin
- Fast visualization
- Fast application startup using standard read techniques
- Storage of large, shared databases/tables for very large data analytics

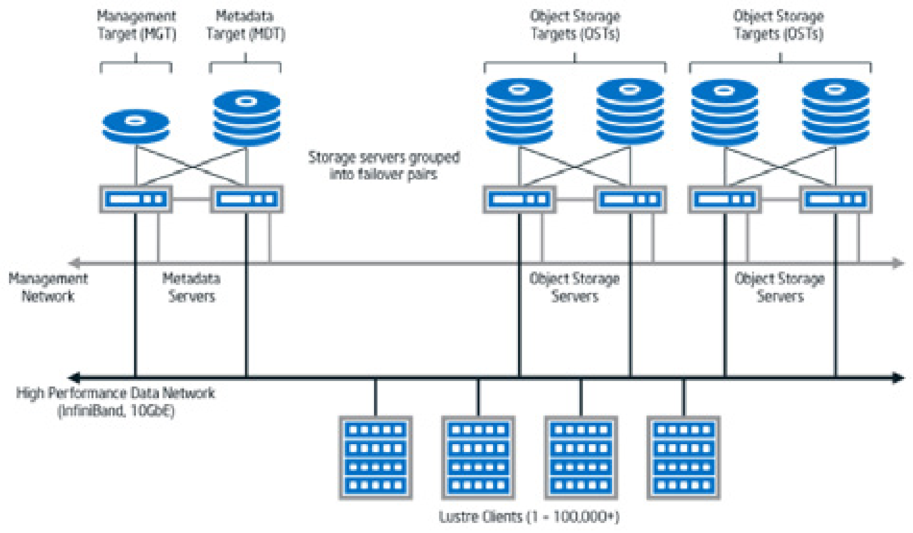

Management Target (MGT)
Metadata Target (MDT)
Object Storage Targets (OSTs)
Object Storage Targets (OSTs)
Storage servers grouped into failover pairs
Management Network
Metadata Servers
Object Storage Servers
Object Storage Servers
High Performance Data Network (InfiniBand, 10GbE)
Lustre Clients (1 - 100,000+)

Figure 2. **Figure 2. Intel® Lustre* file system configuration**

“Intel® Enterprise Edition for Lustre* software unleashes the performance and scalability of the Lustre parallel file system for HPC workloads, including technical big data applications common within today’s enterprises.”

Intel® Enterprise Edition for Lustre* software

Intel Enterprise Edition for Lustre software (Intel® EE for Lustre software) unleashes the performance and scalability of the Lustre parallel file system for HPC workloads, including technical big data applications common within today’s enterprises. It allows end users who need the benefits of large scale, high-bandwidth storage to tap the power and scalability of Lustre, with the simplified installation, configuration, and management features provided by Intel® Manager for Lustre* software, a management solution purpose-built by the Lustre experts at Intel for the Lustre file system. Further, Intel EE for Lustre software is backed by Intel, the recognized technical support providers for Lustre, including 24/7 service level agreement (SLA) coverage.

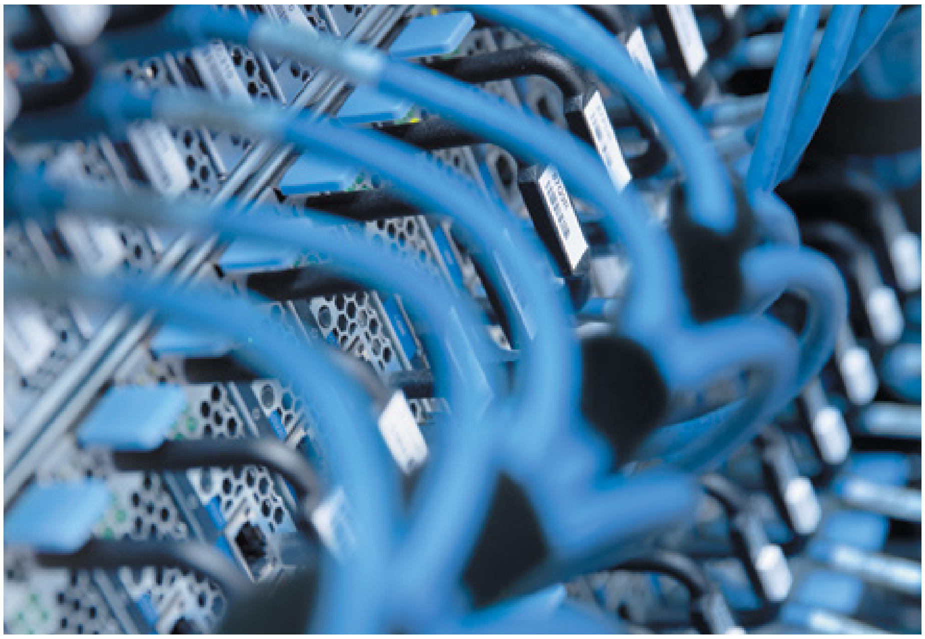
This enhanced implementation of Lustre reduces complexity and management costs, while accelerating the benefits of distributed, shared storage. It’s ideal for storage administrators and data scientists who need very fast and highly scalable storage that’s simple to configure and manage.

Intel Manager for Lustre also provides an administrator’s dashboard to speed real-time monitoring for one or more distributed Lustre file systems, including tracking usage, performance metrics, events, and errors at the Lustre software layer.

Intel SSD Data Center Family for NVMe (Non-Volatile Memory Express)

The Intel SSD DC P3700 Series offers the next generation of datacenter SSDs based on the NVMe specifications—combining fast, consistent performance with high endurance and strong data protection. With these SSDs, data is delivered at a breakneck pace, with consistently low latencies and tight IOPS distribution. Superior performance includes 4KB random read performance of up to 460,000 IOPS and 4KB random write performance of up to 175,000 IOPS.[2] Leading quality of service (QoS) <120µs for 99% reads of Intel SSD DC P3700 Series will ensure quick and consistent command response times.[2] With PCIe Gen3 support and a NVMe queuing interface, it delivers excellent sequential read performance of up to 2.8GB/s and sequential write speeds of up to 2.0GB/s depending on the drive capacity. Intel Lustre solutions fully utilize the benefits and high throughput capabilities of these drives. The optimized NVMe command stack allows for fewer CPU cycles, making the drives more efficient for high-performance data solutions. Advanced SSD controller features include end-to-end data-path protection and power loss protection, and guard against data loss and corruption.

“Intel® Enterprise Edition for Lustre* software provides a framework for incorporating a Lustre file system into a Hierarchical Storage Management (HSM) implementation—as the high-performance storage tier.”



Hierarchical Storage Management and Intel Lustre

Hierarchical storage management (HSM) is a collection of technologies and processes designed to provide a cost-effective storage platform that balances performance and capacity. Storage systems are organized into tiers, where the highest-performance tier is on the shortest path to the systems where applications are running; this is where the most active data is generated and consumed. As the high-performance tier fills, data that is no longer being actively used will be migrated to higher-capacity and generally lower-cost-per-terabyte storage platforms for long-term retention. Data migration is ideally managed automatically and transparently to the end user.

Intel EE for Lustre software provides a framework for incorporating a Lustre file system into an HSM implementation—as the high-performance storage tier. When a new file is created, a replica can be made on the associated HSM archive tier, so that two copies of the file exist. As changes are made to the file, these are replicated onto the archive copy as well. The process of replicating data between Lustre and the archive is asynchronous, so a delay in data generated on Lustre will be reflected in the archive tier. As the available capacity is gradually consumed on the Lustre tier, the older, least frequently used files are “released” from Lustre, meaning that the local copy is deleted from Lustre and replaced with a stub file that points to the archive copy. Applications are not aware of the locality of a file: there is no distinction between a file on the archive and a file on the Lustre file system from the perspective of directory listings or the stat() system call. Most important, applications do not need to be rewritten in order to work with data stored on an HSM system. If a system call is made to open a file that has been released, the HSM software automatically dispatches a request to retrieve the file from the archive and restore it to the Lustre file system. This may be noticeable in the form of a delay, but is otherwise transparent to the application.

The archive storage is usually the most reliable storage tier in this configuration. Data is expected to be held there for a long time, so one of the key characteristics of the archive storage tier is reliability.

"Designing a storage solution to sustain checkpointing across a large number of compute nodes can be challenging. It is becoming common to run compute nodes with more than 128GB of RAM at a scale greater than 1024."

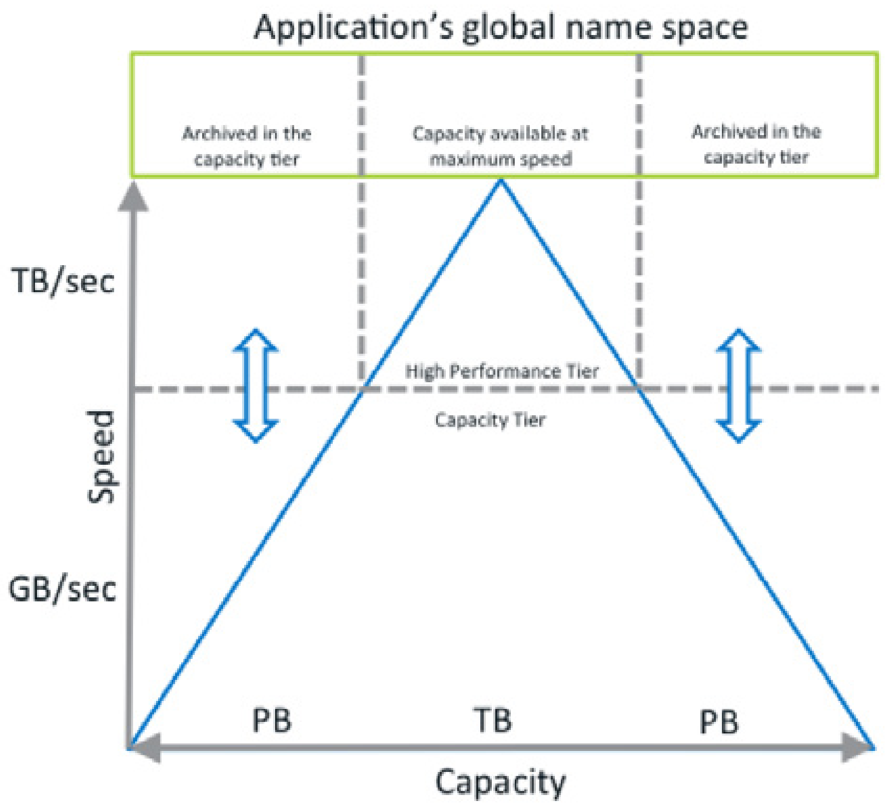

Application's global name space
Archived in the capacity tier
Capacity available at maximum speed
Archived in the capacity tier
TB/sec
Speed
GB/sec
High Performance Tier
Capacity Tier
PB
TB
PB
Capacity

Figure 3. **Hierarchical Storage Management (HSM)**

Checkpointing on Lustre Architecture

Designing a storage solution to sustain checkpointing across a large number of compute nodes can be challenging. It is becoming common to run compute nodes with more than 128GB of RAM at a scale greater than 1024. This means delivering around 128TB of memory to checkpoint as fast as possible.

In order to minimize the time wasted to achieve checkpointing, the parallel file system used to save the checkpoint should run at TB/sec. Sustaining a TB/sec file system using spinning disks, and considering a fair performance of 140MB/sec per hard disk drive (HDD), this would require 7,500 devices (without factoring in the RAID protection and hot spares). The same theoretical exercise using Intel NVMe SSD, able to write at 2GB/sec, would require 512 devices.

Designing a High-Performance Storage Tier

Next, let's look at designing a very fast high-performance storage tier (HPST) using Intel EE for Lustre software and Intel NVMe devices. We'll also use a reliable capacity storage tier (CST) to archive data with HSM technology.

Note, that the HSM agent (copytool) included in Intel EE for Lustre software and used here is not designed for production, but only as an example for evaluation. Intel's HPC solution partners can offer more efficient and scalable HSM agents.

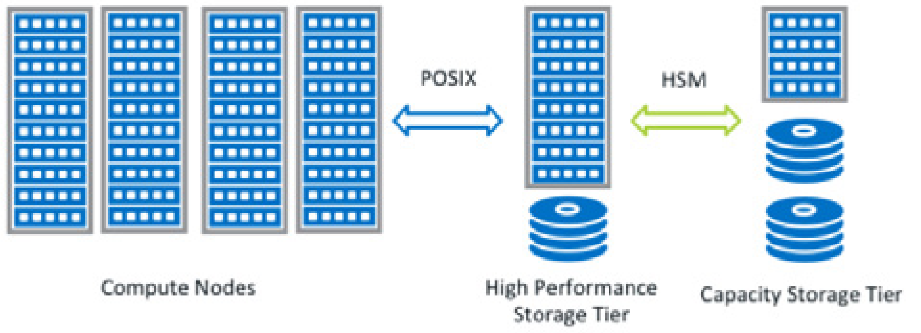


Figure 4. High-performance storage and capacity storage tiers

“The policy engine collects data about the state of the Lustre file system by reading the MDT change logs, and executes tasks based on rules applied to the collected data.”

In this example, we design the HPST layer using 4U Lustre server chassis in order to host as many PCI Express cards possible. Each OSS server has a dual-socket Intel® Xeon® E5-2643v2 CPU and 128GB of RAM. We install 4x Intel® SSD DC P3700 Series 400GB, 1x Mellanox* FDR card, and 1x Intel® True Scale Fabric QDR card. We use 4 OSS and a total of 16 Intel® SSD DC P3700 Series PCI express cards.

The Mellanox card connects computer nodes with the HPST. The Intel True Scale card is used to archive data from HPST to CST using HSM agent copytools. The CST layer is still an Intel Lustre file system composed by traditional spinning disks. We are running an HSM copytool on each OSS.

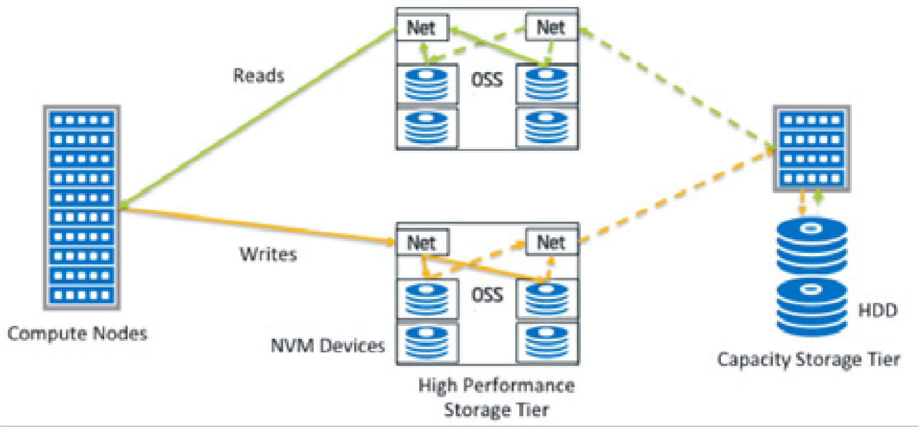


Figure 5. OSS configuration

Though we are not expecting high metadata performance in a checkpointing use case, we design the metadata server using the dual-socket Intel Xeon E5-2643v2 CPU and 128GB of RAM. We install 2x Intel SSD DC P3700 Series 400GB, 1x Mellanox FDR card, and 1x Intel True Scale QDR card.

In the HSM configuration, a central role is provided by the policy engine. A policy engine is a software application acting on behalf of a human operator in support of storage management. The policy engine collects data about the state of the Lustre file system by reading the MDT change logs, and executes tasks based on rules applied to the collected data. We design the policy engine using the same hardware configuration of the metadata server. In order to speed up the performance of the backend database for the policy engine, we are using two Intel SSD DC P3700 Series 400GB.

Lustre Configuration

The Lustre configuration is fairly straightforward: we should only use the NVMe devices provided by the Linux* operating system for the Intel SSD DC P3700 Series. For more information on how to format and mount the Lustre file system, please refer to the Lustre Operational Manual.

We'll use obdfilter survey in order to predict the performance of Lustre using NVMe devices. The obdfilter results are in line with the expectations for writes, but Lustre needs a tuning in order to improve reads performance.

"We'll use obdfilter survey in order to predict the performance of Lustre* using NVM devices."

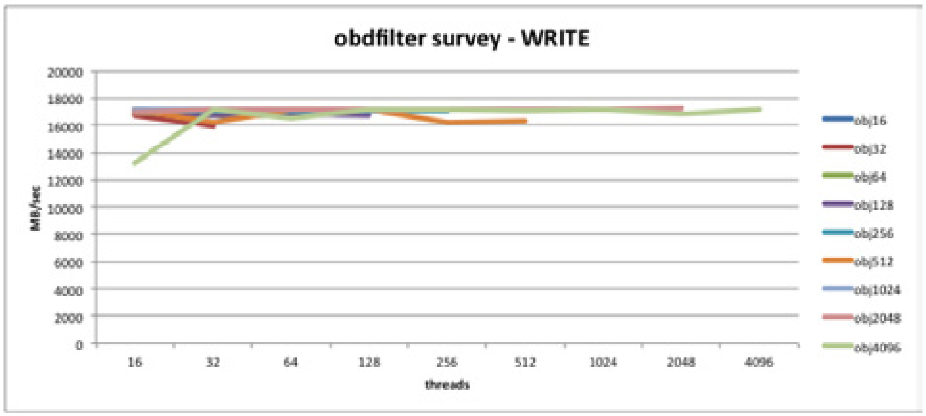


Figure 6. obdfilter survey: write performance

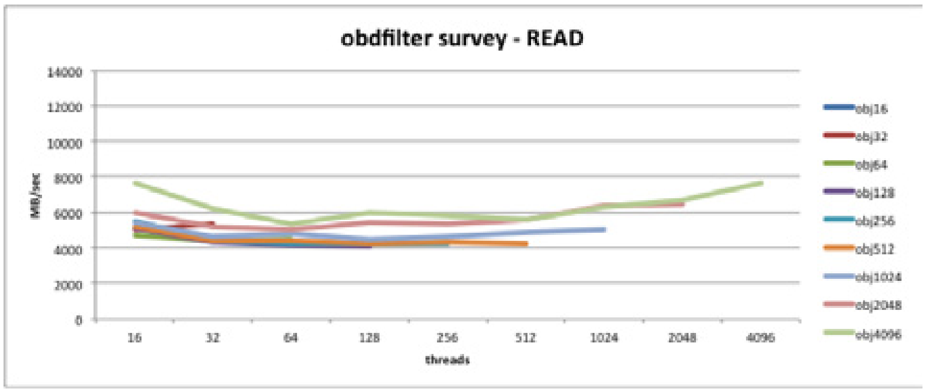


Figure 7. obdfilter survey: read performance default configuration for Lustre*

By disabling OSS read cache we achieve the expected performance

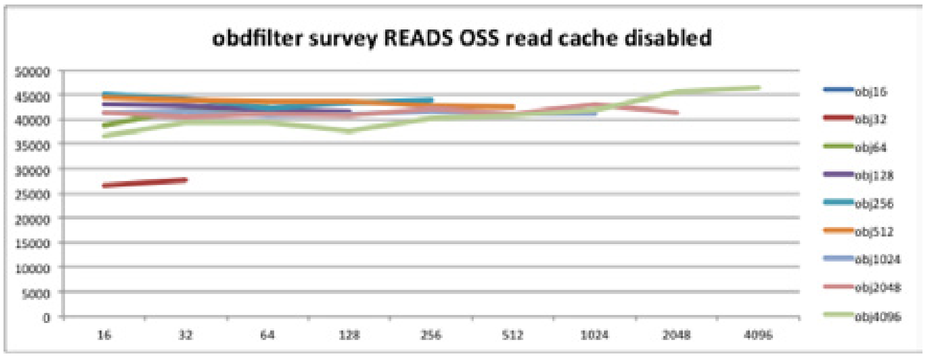


Figure 8. obdfilter survey: read performance disabling the OSS read cache

“In order to have good performance and parallelize HSM jobs, we will run copytool instances for each OSS.”

The obdfilter read benchmark is challenging the Lustre file system with streaming data that is never reused. It consumes a large amount of CPU resources and locking in order to put the data into cache and then evict it again. In addition, with the high streaming bandwidth and IOPS of the NVMe devices, the benefit of the RAM cache is significantly reduced, while the overhead is even higher compared to HDDs. With the OSS read cache disabled this overhead is gone.

Table 1 displays a comparison between the performance expected and the performance measured using obdfilter-survey after the tuning.

	1X INTEL® SSD DC P3700 400GB	THEORETICAL AGGREGATE PERFORMANCE OF 16X INTEL® SSD DC P3700 400GB	OBDFILTER SURVEY RESULTS
Sequential Read Performance	2.7GB/sec	43.2GB/sec	40-43GB/sec
Sequential Write Performance	1.08GB/sec	17.28GB/sec	16-17GB/sec

Table 1. A comparison of theoretical and measured performance with obdfilter

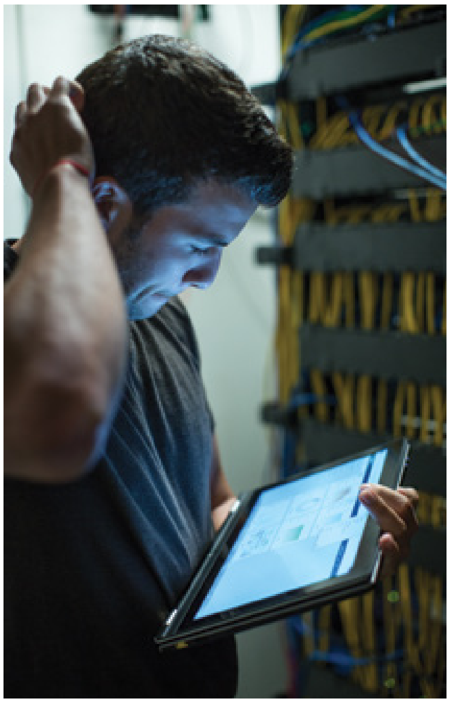
After completing the performance analysis, we are ready to enable HSM capabilities in this Lustre file system, configure and start HSM agents, and configure and run the policy engine.

On the MGS, running this command will enable HSM:

```
# lctl get_param mdt.*.hsm_control
```

In order to have good performance and parallelize HSM jobs, we will run copytool instances for each OSS. The copytool included with Intel EE for Lustre is a technology preview not designed for production. Intel, along with several of our partners, is working towards high-performance copytools that also enable tape library archiving and disaster recovery.

The copytool moves data from the HPST to the CST (and vice versa) and we need to have both the file systems mounted. In this case, we mounted the HPST under the `/mnt/bb` mount point and the CST under `/mnt/lustrefs`.



We run the HSM agent copytool using this command on all the OSS servers:

```
# lhsmtool_posix --daemon --hsm-root /mnt/lustrefs/bb/
--archive=1 /mnt/bb/
```

The MDT coordinator will dispatch the HSM jobs across all the available agents. To verify how many agents are connected and the status, we can run this command on the MGS:

```
# lctl get_param mdt.*.hsm.agents
```

Now from a client we can manually:

- Verify the status of a file:

```
# lfs hsm_state <file>
```

- Archive a file:

```
# lfs hsm_archive --archive 1 <file>
```

- Release the capacity on the HPST:

```
# lfs hsm_release <file>
```

- Retrieve a file from the CST:

```
# lfs hsm_restore <file>
```

In the CST file system, we will find the file archived as FID and the POSIX reference in the shadow directory.

For more information on how to configure Lustre HSM and POSIX copytool, please refer to the Lustre Operational Manual.

"The policy engine is the brain of every HSM implementation. Intel® Enterprise Edition for Lustre* software includes the open source Robinhood as its policy engine."

Smart Policy Engine Configuration

The policy engine is the brain of every HSM implementation. Intel EE for Lustre software includes the open source Robinhood as its policy engine.

We will configure Robinhood in order to:

- Increase the reliability of the solution and minimize the Recovery Point Objective (RPO)
- Maximize the number of checkpoint versions available in the HPST
- Increase the frequency of the checkpoint

This example is based on very fast PCI Express SSD cards without any RAID protection or high availability. The idea is to move the data from the HSPT to the CST a few seconds/minutes after the last checkpoint, and accept the risk of losing the last version of the checkpoint in case of problems on the HSPT. The RPO of the solution will be equal to the time needed to complete the checkpoint and to archive it in the CST.

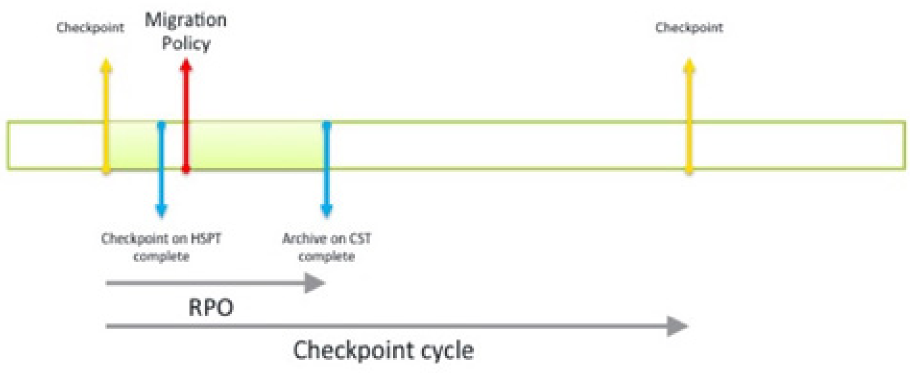


Figure 9. Configuring the migration policy of Robinhood to minimize the RPO

In the case of a restart of an application from a specific point in time, we would like to have the data available on the first tier, rather than waiting until the data is retrieved from the CST.

The purge policy in Robinhood is in charge to release files that have been archived in order to free capacity in the Lustre file system. When an archived file is released (purged), the file's data is deleted, freeing up space. The metadata for the file is available and continues to point to the archive copy.

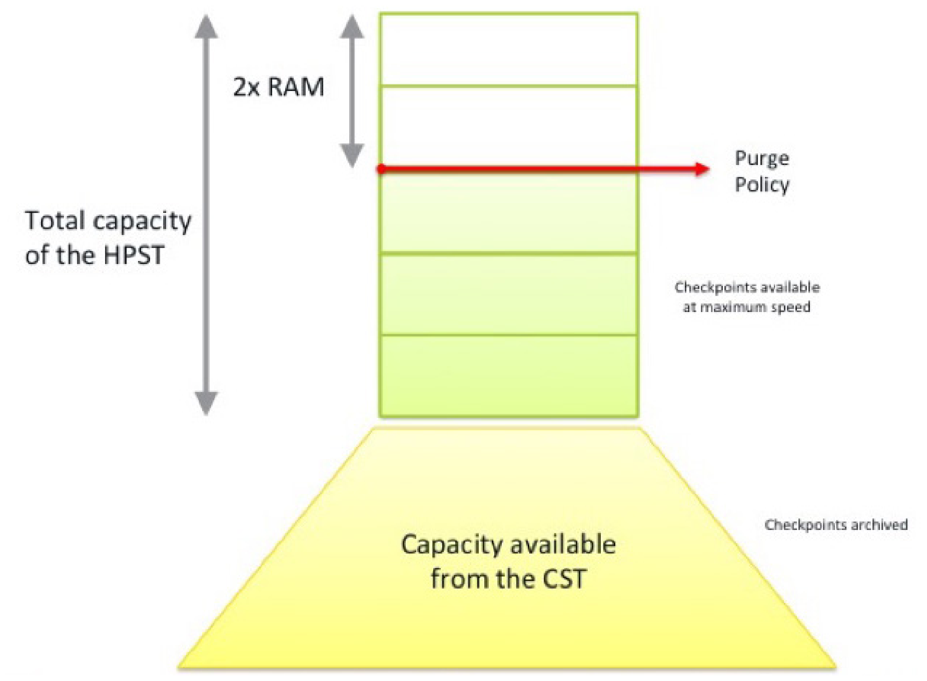

2x RAM
Total capacity of the HPST
Purge Policy
Checkpoints available at maximum speed
Capacity available from the CST
Checkpoints archived

Figure 10. Configuring the purge policy of Robinhood to maximize the checkpoint version available at maximum speed

We can configure the global allocation trigger in order to provide at least 2x the total RAM to checkpoint.

The smart configuration for the policy engine can also be used to decrease the checkpoint interval and increase the serviceability of the application.

In our lab, we have 16 computer nodes each with a total memory of 128GB. The aggregate space needed for a single checkpoint is 2TB.

The write performance on the HPST is 16GB/sec with a capacity of 16x 400GB = 6TB. The CST has a capacity of 170TB and a performance of 6GB/sec.

To calculate the RPO and set the migration policy, we have:

- 128 sec to write 2TB at 16GB/sec
- 300 sec to wait between the checkpoint and the archiving
- 512 sec to archive 2TB at 4GB/sec. The performance of a single Intel HSM copytool is capped at 1GB/sec. The Intel HSM copytool is not designed for parallel operations. We can't reach the full speed of the CST using only 4 copytools.
- RPO=940 sec

The performance of the solution and the RPO time was confirmed in experiments using IOR. To mimic the typical checkpoint workload, we configured IOR using one file per process and a large block size of 1 MByte.

For a complete configuration for Robinhood, see the Robinhood policy engine website.

For More Information

For more information about Intel Enterprise Edition for Lustre software, visit: lustre.intel.com

Resources

1. **Lustre* Operational Manual**
2. **Intel® Solid-State Drive Data Center Family for PCIe***
3. **Robinhood policy engine**
4. **Recovery Point Objective**

The most noticeable part of the Robinhood's configuration file for our experiment is:

```
Migration_policies {
  policy default {
    condition {
      last_mod > 5min
}
  }
}
```

To configure the high and low threshold with the capacity available (6TB):

- Free space equal to 2x aggregate RAM = 4TB
- At least 1 version (2TB) available on the HPST not released (purged)

```
Purge_trigger {
    trigger_on         = global_usage;
    high_threshold_pct = 66%;
    low_threshold_pct  = 33%;
    check_interval     = 5min;
}
```

Conclusion

We've explored the possibility of using Intel Enterprise Edition for Lustre to design a very fast, cost-effective parallel file system for checkpoint use cases.

We also tested the performance of the Intel® SSD Data Center Family for NVMe used in a Lustre environment. Our expectations were confirmed, even when some Lustre tuning is necessary. Intel is also actively working to better characterize applications that use Lustre running on Intel NVMe devices.[3]

Intel Enterprise Edition for Lustre software, in conjunction with the Intel SSD DC P3700 Series, can be considered in order to design a high-performance storage tier for well-defined use cases. The policy engine software can be configured in order to minimize the recovery point objective time, and to maximize the number of versions of the checkpoint data available at full performance.

This solution can dramatically increase the return on investment in a parallel file system for checkpointing using standard techniques.

1. Based on Intel analysis of November 2014 Top500: www.top500.org.
2. http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-dc-p3700-series.html
3. http://www.intel.co.uk/content/dam/www/public/us/en/documents/performance-briefs/lustre-cluster-file-system-performance-evaluation.pdf