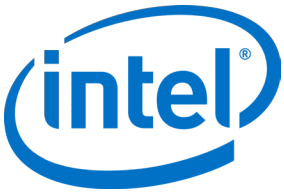
WHITE PAPER
Intel® Enterprise Edition for Lustre* Software
High Performance Data Division



Intel® EE for Lustre* Hierarchical Storage Management Framework

The Hierarchical Storage Management (HSM) Framework released in Lustre* 2.5 allows seamless integration between Lustre and third party HSM products. With HSM, files can be automatically and transparently migrated between Lustre and archive storage systems.

Contents

Lustre's Hierarchical Storage Management Framework brings key enterprise functionality to Lustre, benefitting both our HPC customers and our enterprise customers. - Brent Gorda, General Manager, Intel® High Performance Data Division.

Executive Summary

The volume of data being generated by high performance applications is expanding at unprecedented rates. This fact is coupled with the increasing demand for high-performance storage systems to support such computation. Data retention requirements add to these demands, so that data center managers are left struggling to balance the needs for high-performance storage for active data against regulatory compliance and other long-term data retention demands.

Hierarchical Storage Management in Lustre 2.5 provides a reliable mechanism for archiving data to a secondary, high-capacity and / or lower-cost storage tier. HSM frees space in Lustre, preserving

performance, and also providing additional assurance against data loss.

Administrators can create policies that determine which files need to be archived and how often, and when to release capacity in Lustre by removing local copies of archived files. Policies also can be created to limit how much "active" capacity individuals or groups can consume before their data is migrated to less-costly storage.

Data movers manage the migration of files, running in parallel for improved performance. When a user or program accesses a file that has been archived, that file is automatically restored for use. This allows Lustre to serve its customers' needs for the highest-performing file systems, while cutting the cost of long-term data storage.

Several partners are working with Intel® to integrate their HSM products with this new Lustre Hierarchical Storage Management Framework. For more information, contact your Intel® Lustre reseller, or email the Lustre team at Intel® at: hpdd-info@intel.com.

Intel® EE for Lustre* Software Introduces HSM Support

Intel® Enterprise Edition for Lustre* software version 2.0, which builds on version 2.5 of the open source distribution of Lustre, introduces support for hierarchical storage management (HSM) integration. All subsequent versions of Intel® EE for Lustre* software have the HSM framework.

HSM is a mechanism for moving data between different types of storage in a manner normally transparent to the end-user or application. Storage platforms are organized into tiers based on cost, performance, and capacity. High-performance storage is generally more expensive to acquire and maintain than slower storage, while slower storage tends to have higher overall capacity. HSM is used to exploit the best characteristics of both high performance and high capacity storage – the high performance storage is placed on an IO path that is logically close to where applications are running and new data is written there. As the data ages and as available capacity diminishes, the HSM platform copies older files to the high capacity storage tier, freeing up capacity in the high performance storage. If an application makes a request for a file that is stored on the slower, high capacity storage, that file is automatically retrieved and restored to the high performance storage.

The Lustre 2.5 file system has the features necessary to participate as the high performance storage in a multi-tiered HSM system. New data is written directly to Lustre; as the data ages and is less frequently accessed, it can be migrated to an associated high capacity storage platform, referred to as an archive. Archives may be based on a variety of technologies, of which tape is probably the most common.

In a completely integrated solution, the end-user only sees the Lustre file system and does not need to be aware of the HSM mechanism. Applications *do not* need to be re-written in order to participate in a Lustre file system that has HSM support.

Files are replicated onto the archive storage, so that initially, two copies exist: the original file on Lustre and the replica on the archive. As data is written to the Lustre file system, available capacity is consumed, reducing the amount of free space for new data. When Lustre runs out of space, or when a specified threshold is exceeded, older files are "released", which means the original, local, copy is deleted from Lustre and replaced with a stub file that points to the file on the archive. Applications are not aware that the data no longer resides locally: directory listings and the stat () system call work as before. If a request is made to read a file that has been released in this manner, the HSM software automatically retrieves the archived data and restores the file to Lustre. This happens

transparently to the application, although there may be a delay while the data is fetched from the archive.

Simplistically, there are two ways to think about how Lustre and HSM behave:

1. The Lustre file system acts as a fast cache for a large quantity of archive data stored on a slower system.
2. The archive system acts as paging / swap device for Lustre: when a request for more space on the Lustre file system is made, the least-frequently used files are "paged-out" to the archive until the allocation is satisfied.

HSM does not represent a backup solution. While implementations can and do vary, the archive tier may not version data, nor does it offer any guarantee that a file that has been permanently deleted from the high-performance tier can be retrieved from the archive tier. This behavior is policy-dependent, but in normal operation a file that has been deleted from Lustre will eventually be purged from the archive as well. This is in contrast to the behavior of a backup system, which is decoupled from the active file system. Deleting a file from Lustre has no effect on the copy stored in a backup. While some archive solutions do provide long-term backup features and versioning, these are implementation functions that are separate from Lustre.

Archive tiers are typically very reliable systems, as well as being capable of storing vast quantities of data (the largest tape libraries can measure in the exabyte range). There is no suggestion that the archive tier is less reliable than a backup system, only that the data may be managed differently, with potentially different goals.

Lustre HSM Architecture

Figure 1 depicts the HSM high-level architecture. As shown, users and applications issue *archive*, *release*, and *remove* commands to the MDS from Lustre clients. The MDS organizes and dispatches archive requests through the MDT Coordinator thread to one or more copytools running on HSM Agents. Copytools transfer files to and from the Archive. The policy engine host records changes to the file system by processing MDT changelogs, and issues archive tasks. Policies define archive and release criteria for the file system.

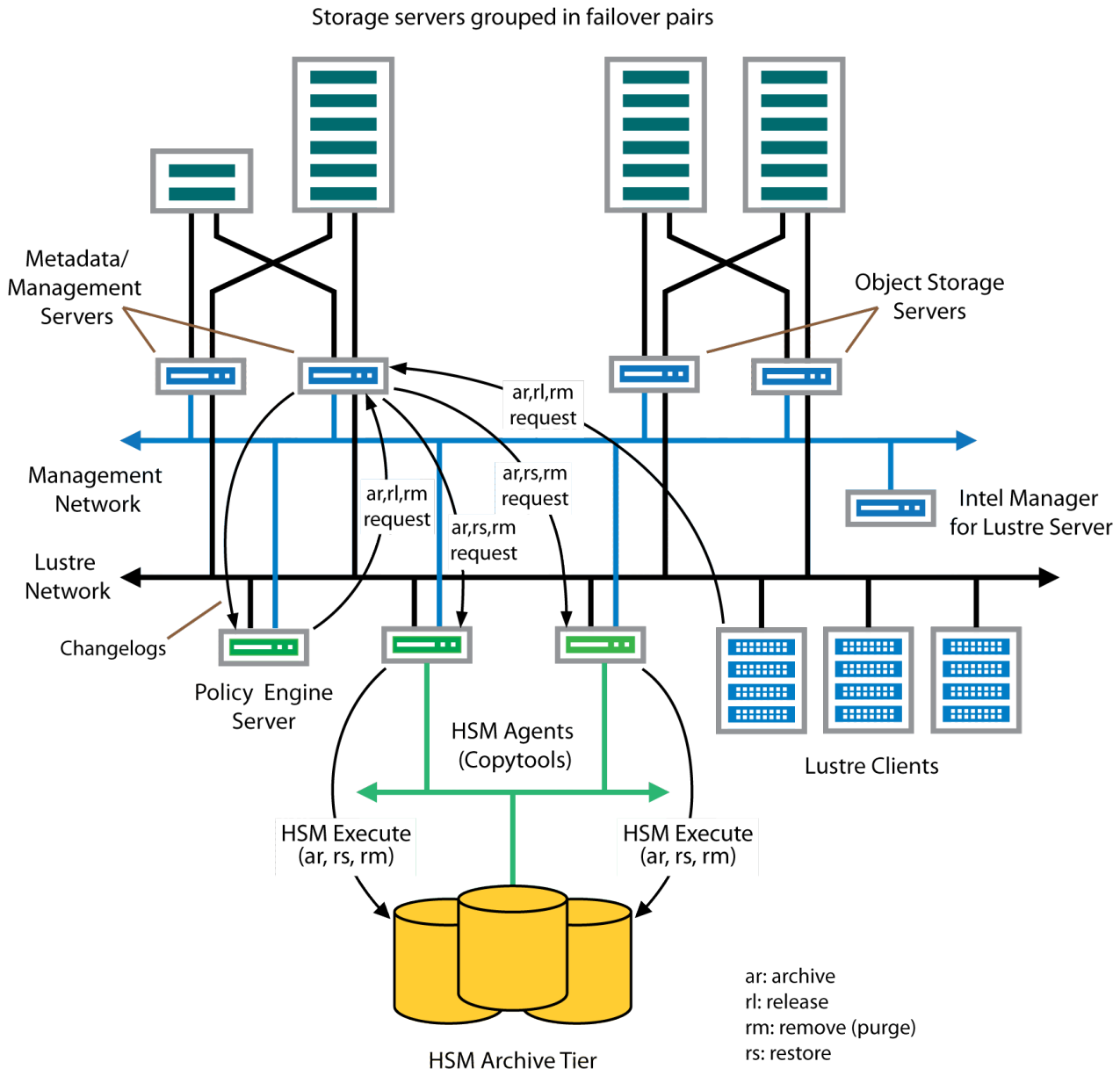


Figure 1. HSM high-level architecture

MDT Coordinator and Agent Copytool

HSM support in the Lustre file system is centered on two principal features: the MDT Coordinator and one or more copytool services. User and applications issue HSM requests to the MDT Coordinator, which is a thread running on the metadata server (MDS). The MDT Coordinator is responsible for queuing and dispatching requests to the storage archive via an available copytool service (which is a daemon running on a special-purpose Lustre client called an HSM Agent). HSM Agents are synonymous with Data Movers in other storage management environments.

The copytool is an intermediary between the Lustre file system and the archive tier, executing commands in response to requests from the MDT Coordinator. Typical commands are: archive, restore, and remove. Neither the MDT Coordinator nor the copytool manage the release of files. This is handled by a separate kernel thread on the metadata server, because releasing (or purging) a file from Lustre requires no

interaction with the archive. Note that archiving a file and releasing a file are discrete tasks. Files are not automatically purged from the Lustre file system when they are archived. This allows for a great deal of flexibility when designing data management policies, and can add an extra level of data protection since a file may exist in both the Lustre tier and the archive storage tier.

There is a single MDT Coordinator in each Lustre file system, but there can be many copytools. Deploying multiple copytools can improve the responsiveness of HSM requests and provide greater aggregate I/O than can be achieved with a single server. Each copytool runs on a separate Agent host and registers with the MDT Coordinator when it is launched. The MDT Coordinator maintains a list of registered copytools. It is possible for a single Lustre file system to be served by multiple storage archives, each with multiple copytools.

The MDT Coordinator provides flow control, removes duplicate requests, and ensures that incomplete requests can be replayed in the event of a request failure.

The copytool transfers data from the archive to Lustre and vice versa. Because the copytool has to map files from Lustre to the archive storage and back again, each copytool is specific to the target archive and must implement the storage protocol appropriate to that archive system. Lustre provides a reference implementation for a POSIX copytool that can interface to a POSIX-compliant storage archive. This means that one could create an archive that is itself another POSIX file system.

Not all storage archives implement a POSIX interface, so different archive vendors may provide their own copytool implementations to manage transactions between Lustre and the archive.

Lustre Clients

Lustre clients are provided with software to manage interaction with the HSM platform. Users issue commands using the "lfs" command shell. Commands allow users to archive, release and restore files, and also determine status.

A complete description of the command interface is available in the Lustre Operations manual.

Applications running on Lustre clients do not need to be modified to run on an HSM platform. Rather, they continue to interact with Lustre as they would with any other POSIX-compliant file system.

Policy Engine: Automating Storage Management Actions

Lustre's HSM support supplies the tools to implement the necessary infrastructure and integrate Lustre with an archive, via either the supplied POSIX copytool or through a 3rd-party implementation. Users can employ the command-line tools supplied with Lustre to mark the files that they want to archive and can send commands to the MDT Coordinator to manage capacity by releasing files on a per-file basis. However, this can quickly become cumbersome, particularly when working at the scale where both Lustre and archive storage solutions excel.

A way is needed to automate HSM management to a software proxy, acting on behalf of the storage administrators. This function is provided by a software system known as the Policy Engine. Administrators or other super-users create a description of the files and directories that are to be managed by the HSM solution and the criteria for undertaking certain actions, such as copying data to the archive or releasing capacity. These criteria might include:

- the how often files are archived

- the amount of active Lustre storage capacity that a single user or group of users may consume
- thresholds for file system or OST capacity that when exceeded, will trigger a purge of least frequently accessed data.

The Policy Engine executes these policies by sending commands (as a user would) to the MDT Coordinator.

Although the Policy Engine is not a Lustre component and is not supplied with the core, open source Lustre software, Robinhood is a 3rd-party, open source policy engine that has comprehensive Lustre HSM support, and the Intel® EE for Lustre* software HSM reference architecture employs Robinhood. Some archive storage vendors may instead provide their own policy engine software.

Archive Storage

The archive tier is the final piece in the Lustre HSM architecture. Archive systems provide the highest levels of capacity and data resilience of any storage system, but carry a significant performance penalty relative to platforms such as Lustre. In a Lustre HSM system, the archive provides the storage target when migrating data from the high performance Lustre file system.

The copytool writes data from Lustre to the archive storage system and retrieves files from the archive if they have been released.

The archive system can be built on any of several different storage technologies that are typically characterized by highly reliable data integrity and massive storage capacity. Other characteristics include lower total cost of ownership (TCO) and a lower power footprint for the capacity provided.

Lustre provides the building blocks for integrating an archive solution with Lustre, and many products exist to fulfill this function. Tape libraries are the most obvious archive medium, but several disk-based solutions exist as well.

Intel® EE for Lustre* Software, HSM Reference Design

Lustre's HSM framework comprises the MDT Coordinator (CDT or hsm_cdtr lightweight process), a copytool agent for managing data between Lustre and a POSIX-compliant archive, new RPCs in the communication API, and command line tools incorporated into the lfs shell.

The MDT Coordinator runs as a lightweight process (or thread) on the metadata server (MDS). Lustre is distributed with a POSIX copytool reference implementation exhibiting the features necessary to an HSM agent. Other agents can be created, based on this design, to provide access to archives that use protocols or semantics other than POSIX.

The reference design does not include an archive implementation. For evaluation purposes, any POSIX-compliant storage target can be employed, although care should be taken to establish an archive appropriate to the requirements of the environment under test.

HSM Copytool, POSIX Reference Implementation

Figure 2 shows two examples of HSM copytool implementations. In the first example, Archive 1 is connected to a single copytool instance running on an HSM Agent. The second example shows two copytool instances connected to Archive 2; each copytool instance runs on its own agent host.

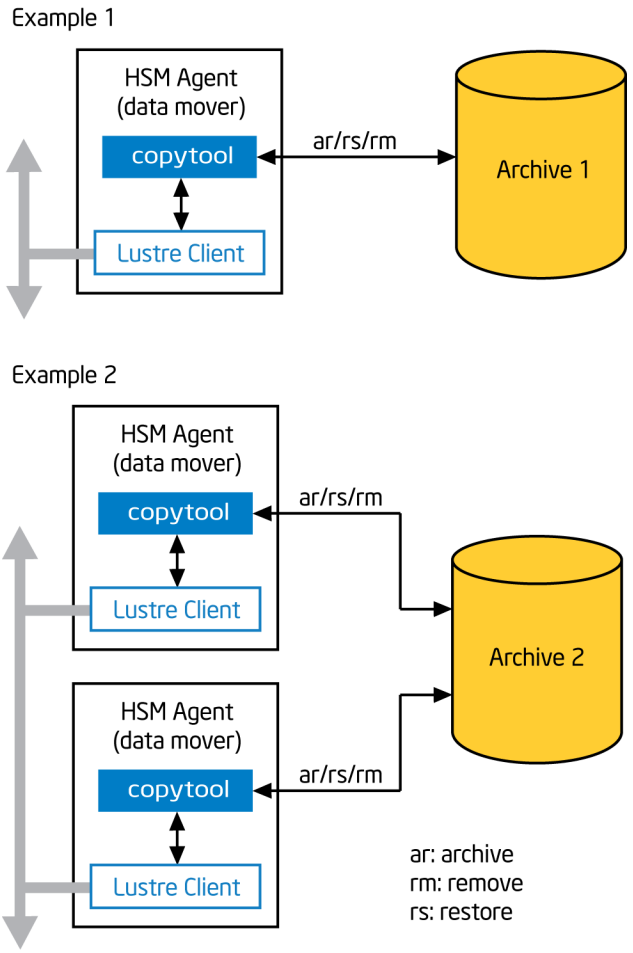


Figure 2. Examples of copytool running on Lustre

HSM Agent hosts are special-purpose Lustre clients that have an interface to the HSM archive storage tier. This interface is managed by the copytool. As stated before, the copytool migrates files to the archive platform, and retrieves files that were previously released and restores them to Lustre. The copytool also records file metadata, including Lustre stripe information. Different archive storage platforms can have different interfaces or APIs, each of which will require a specific copytool. Intel® EE for Lustre* software is supplied with an HSM copytool that will interface to a POSIX-compliant archive platform.

Every Lustre HSM file system requires at least one HSM agent host to be able to transact with an archive system. An agent runs a single copytool instance, connecting to a single archive. To provide additional performance, it is possible to deploy many HSM agents for an archive, each running an instance of the copytool for that archive. Lustre clients do not interact directly with the copytool processes; instead, HSM requests are sent to the MDT Coordinator which organizes and dispatches commands to the copytool. The MDT Coordinator will try to optimize the dispatch of commands to the copytool, e.g., by consolidating duplicate requests.

The POSIX copytool supplied with Intel® EE for Lustre* software currently supports connection to a single archive on each Agent. To provide scalability or to support multiple archives for a Lustre file system, it is necessary to deploy more Agent servers. Other copytool implementations could potentially serve multiple archives per Agent server.

When a copytool is launched, it registers with the MDS, providing information to the MDT Coordinator thread about the archive it supports. The MDS must have the MDT Coordinator thread enabled before starting the copytool. The MDS keeps a list of all of the copytools that have been registered. When subsequent copytools are started, they are added to this list.

The POSIX copytool assumes that the archive is mounted as a directory on the Agent server. As data is archived, the copytool automatically populates the archive with a directory structure based on the Lustre FID (file identifier), and will also record Lustre extended attributes (such as striping information) for the file. In addition to the FID structure, a shadow directory structure is also created to mimic the file system layout of the Lustre file system as seen by the end-user or application. The shadow file system area maps file names to FIDs by way of soft links.

Thus, as the archive grows, two structures emerge: one to define the logical structure of the file system and one to record the data and extended attributes.

HSM Policy Engine – Robinhood

The Intel® EE for Lustre* software HSM reference architecture employs Robinhood as the policy engine. The Lustre HSM interface is an open API and does not preclude the development of other applications for the policy engine function.

Robinhood collects data from the Lustre file system by monitoring the MDT Changelogs and records this persistently in a relational database. It then analyses this information and takes action on behalf of users and administrators based on a set of defined policies for the handling of data stored on the Lustre file system. The Robinhood database also facilitates detailed reporting and monitoring.

Because it uses the MDT Changelogs, Robinhood rarely has to scan the file system namespace directly, removing costly overhead.

Policies define the conditions under which to act (triggers) and the action to take when a condition is met. For HSM, actions include migrating data to the archive and releasing capacity in Lustre by purging least-frequently-used files.

Triggers are threshold definitions, typically based on the relative age of the target file (e.g., time since last access or last modification) or the consumed capacity of the file system. Further refinement can be made based on ownership of files, both for users and for groups.

Robinhood does not restore files from the archive. If a request is made to access an archived file with no local copy (a *purged* or *released* file), it is automatically retrieved and placed into the local storage tier by Lustre, transparently to the user's application. When a file that has been released is accessed by an application, the MDS automatically queues a request with the MDT Coordinator to restore the file form the archive copy. The MDT Coordinator then dispatches this request to a copytool, which implements the actual file restore, copying the file from the archive back to the Lustre file system. This is handled directly by Lustre HSM, not the policy engine.

The most common use case for HSM is to provide additional storage capacity to an environment while containing costs: the local storage tier acts much like a high performance cache for the archive storage tier (which can be anything from high-capacity disk to a tape library).

Performance and Availability Considerations

To provide horizontal scalability of the Policy Engine service, processes can be spread across multiple physical machines. This allows the Policy Engine to be hosted separately from the database, for example. This logical separation of processes and the database allows the development of a solution architecture that is both scalable and robust. In the reference implementation, Robinhood server processes are also multi-threaded for increased parallelism.

A single box HSM policy engine service is suitable for small-to-medium scale environments and for systems that can absorb a temporary loss of the HSM policy engine function. For mission critical platforms, a high-availability configuration for the Policy Engine may be preferable. Nevertheless, a "single box" solution will suffice for many production installations because failure of the policy engine does not affect the availability or performance of the Lustre file system itself.

Generally, if there is a strong requirement for a high-availability policy engine, it is suggested that implementers apply a standard cluster design pattern based on an HA cluster pair for the Policy Engine application, running in an active-standby resource group. RDBMS platforms often have their own recommendations for providing high availability; in some cases this can be incorporated into the same

cluster configuration as the Policy Engine but a separate database cluster could also be used. (An RDBMS is not part of the Intel® Enterprise Edition for Lustre* software.)

Because the cost of a highly-available Robinhood service may outweigh the benefit provided, a "cold-standby" hardware replacement strategy might be more cost-effective. In any case, the storage hardware must be reliable and processes must be in place to facilitate recovery after a failure.

As file system capacity requirements increase and HSM becomes ever more critical to the operational running of the file system, it will become necessary to consider a more robust policy engine architecture to assure continuous operation of all data management services. To address this concern, a future proposal may be developed that would address the requirements of high availability installations with strict SLA terms.

For our reference design, we'll describe a single box policy engine server.

Robinhood Policy Engine - Single Box Architecture

Referring to Figure 3, the Robinhood policy engine runs on a Lustre client in the file system network. Robinhood functions are executed as threads within a single process. MDT Changelogs are processed in near-real time and separate threads act upon policy definitions.

The "migrate" and "purge" Robinhood policies map to "archive" and "release" HSM commands, respectively. The RDBMS is a MySQL instance and reports are generated by Robinhood's command line interface.

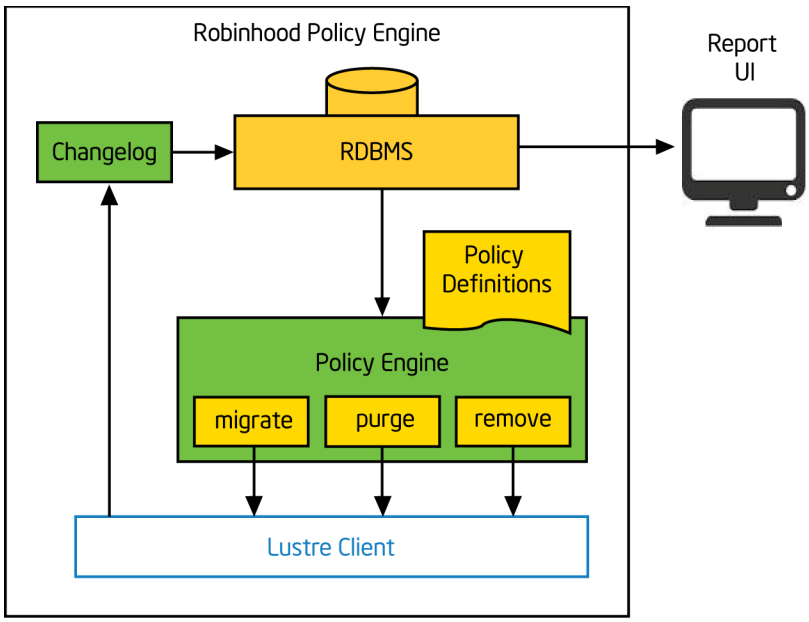


Figure 3. Robinhood Policy Engine – Single Box Architecture

The Robinhood policy engine software runs on a Lustre client computer. In addition to the normal Intel® EE for Lustre* client software "stack", the policy engine server requires a supported RDBMS and optionally, an HTTPD server configured with PHP.

Multiple file systems can be managed from a single server, with each file system having its own configuration and database instance. The policy engine host must have client access to each file system under consideration.

Figure 3 above shows the logical structure of the policy engine service running on a single server with one Lustre file system mounted (Lustre client interface). The Robinhood Policy Engine server runs as a single process with separate threads to manage the changelogs and policies. Policies are defined in text files read at startup (or after receiving a reload signal). Persistent storage for policy engine data is managed by a MySQL RDBMS instance. Queries for reports are executed by command line applications that interface directly with the database; there is no direct connection with the policy engine daemon to transact these queries.

Hardware Platform

The policy engine host is a Lustre file system client and has basic requirements similar to any other Lustre client. At a minimum, it is strongly recommended that the root file system is mirrored, preferably with a hardware RAID controller that can be monitored by the host OS. It is also recommended, although not required, that the database storage is separate from the operating system disks, and that the database disks are hardware-mirrored, using RAID.

In addition to providing extra protection for the data store, separating the OS from the database maximizes the storage performance available to the database. Storage capacity requirements for the database are reasonably modest, but will vary depending on the overall size of the Lustre file system being managed, the number of files, and the rate of change of the data stored therein. File systems with high rates of activity will generate more changelog data, which will, in turn, increase the amount of data recorded in the policy engine database.

The following configuration should suffice for the majority of client installations:

- 2 socket Intel® Xeon IA-64 server
- 16GB RAM
- 4 x SAS-2 HDD – 2 x OS disks, mirrored; 2 x data disks, mirrored
- High performance network interconnect for Lustre file system (e.g. InfiniBand or 10GbE)
- In-band management network for access via Web UI, SSH for management, monitoring

Please refer to manufacturers' documentation for instructions on hardware installation and configuration.

Software Platform

The policy engine software platform is based on RHEL 6.x or one of its derivatives, such as CentOS. In addition to the Robinhood policy engine software, the server will require MySQL and Intel® EE for Lustre* client software (or an Intel® HPDD supported equivalent). The Robinhood Web UI, while also optional, will require Apache HTTPD and PHP.

Authors

Malcolm Cowe, Systems Engineer, Intel® HPDD

More information

For more information, contact your Intel® Lustre reseller, or email the Lustre team at Intel® at: hpdd-info@intel.com.

References

1. Intel® Enterprise Edition for Lustre* http://lustre.intel.com
2. Lustre Community https://wiki.hpdd.intel.com
3. Robinhood http://robinhood.sf.net

Disclaimer and legal information

INTEL CONFIDENTIAL

Copyright 2014 Intel Corporation. All Rights Reserved.

Portions of the software contained or described herein and all documents related to said software ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material contains trade secrets and proprietary and confidential information of Intel or its suppliers and licensors. The Material is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.