

# HPSS System Admin Class

## Introductory Lecture

\* Please review disclosure statement on last slide

# Lecture 1: HPSS Overview

The slides in this presentation were delivered at the  
HPSS Overview lecture at the  
HPSS Admin Class in Houston in December, 2009  
Please review disclosure statement on last slide

## Contents

- A. Introduction**
- B. HSM Concepts**
- C. Storage Allocation Concepts**
- D. HPSS Architecture**
  - 1. Infrastructure Components**
  - 2. HPSS Server Components**
  - 3. Storage Subsystems**
  - 4. Client Interfaces**



## A. Introduction



- **What is HPSS?**

- Hierarchical storage manager (HSM).
- Scalable.
- Parallel.
- Network-centric (see next slide).
- Focus on high-performance access to data.
- Utilize parallel data transfer streams and/or striping across multiple servers to maximize performance.

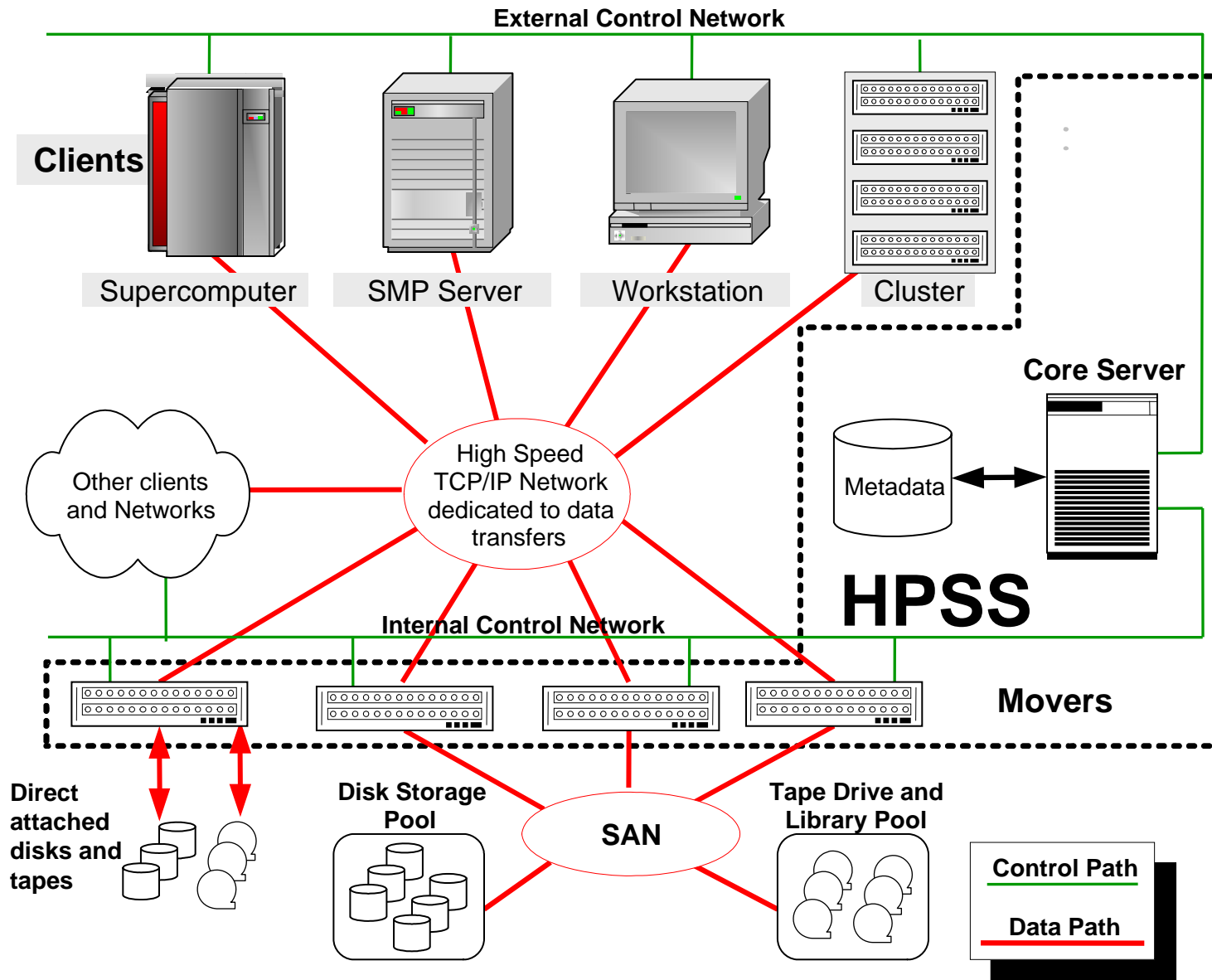
- **Why HPSS?**

- Requirements for fast per file and aggregate data transfers.
- Requirements for large data stores.
- Requirements for large single logical name spaces.
- Requirements for distributed, secure access to data.

- **Availability**

- Initially released on 7/1/96.
- Available from IBM Global Business Services - Houston as a service offering.

# A. Introduction



## A. Introduction

- **How do users access HPSS?**

- Standard FTP
  - **Allows convenient access from any client with a standard FTP client.**
- HPSS Parallel FTP (pftp\_client)
  - **An enhanced version of the standard FTP client with significantly improved performance.**
- HPSS Virtual File System (VFS) Interface Client
  - **Allows user to access HPSS as a Linux file system.**
- General Parallel File System (GPFS)/HPSS Interface (GHI)
  - **Provides GPFS users with the view of an infinite file system by migrating older files to HPSS and staging them to GPFS on demand.**
- Other HPSS Client API Applications:
  - **HSI**
    - **Provides Unix-style command interface in interactive, batch, and command modes.**
    - **Available from Gleicher Enterprises, Inc.**
  - **HTAR**
    - **Bundles small files into large files and store into HPSS.**
    - **Available from Gleicher Enterprises, Inc.**
  - **GridFTP enabled for HPSS**
    - **An extension of the standard FTP protocol for the GRID Computing environment.**
    - **Available from ANL.**
  - **Network File System (NFS V4)**
    - **Allows user to access HPSS as a native file system.**
    - **Available from CEA-DAM.**
  - **Site-specific**

## A. Introduction

- **HPSS Information Resources**

- Published Documentation

- **HPSS Installation Guide.**
- **HPSS Management Guide.**
- **HPSS Error Manual.**
- **HPSS User's Guide.**
- **HPSS Programmer's Reference.**
- **HPSS Conversion Guide.**

- Extensive set of web pages:

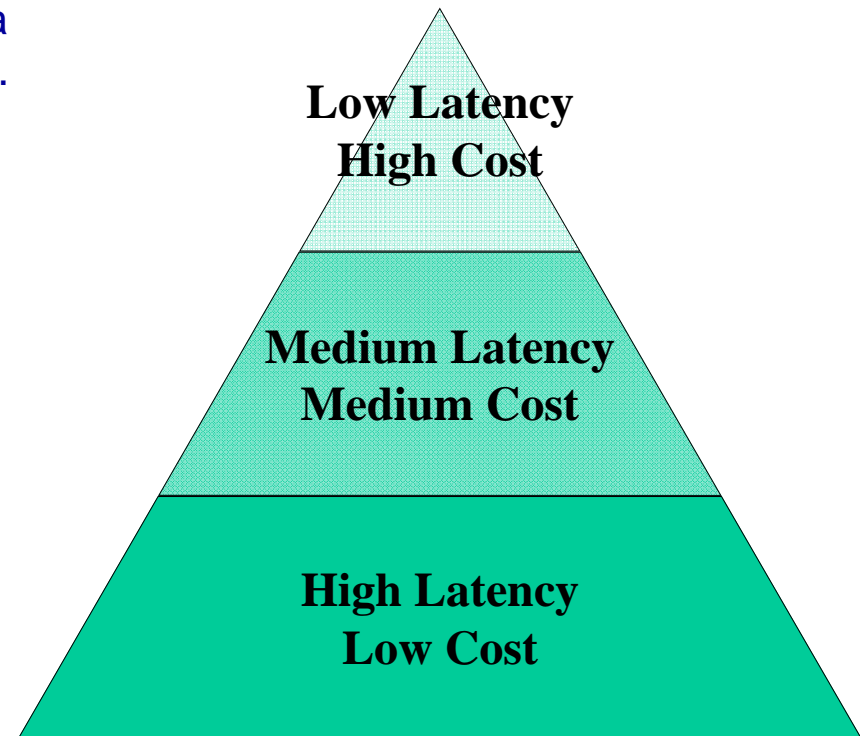
<http://www.hpss-collaboration.org/hpss/index.jsp>

- **Product information.**
- **Support Policy.**
- **Defect / Feature tracking and submission.**
- **Release information, support information, frequently asked questions, etc.**
- **Published documentation available for download in PDF and formats.**



## B. HSM Concepts

- **Hierarchical Storage Management**
  - Data management strategy that seeks to cost-effectively store large quantities of data by storing it on the lowest cost media available that will meet user requirements.
  - Implementations usually store the most recently accessed data on low latency, high cost storage while moving less recently accessed data to progressively higher latency, lower cost storage.
  - HPSS implements this strategy using:
    - **Storage classes.**
    - **Storage hierarchies.**
    - **Classes of service.**
    - **Migrate, purge, & stage operations.**

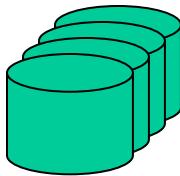


## B. HSM Concepts

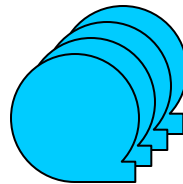
- **Storage Class**

- A logical grouping of similar storage media intended to store HPSS files with similar characteristics (i.e., file size, access frequency, tape technology, etc.)
- Examples:

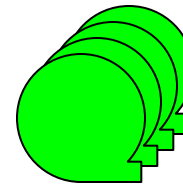
**Disk – Large File**



**LTO – Huge File**



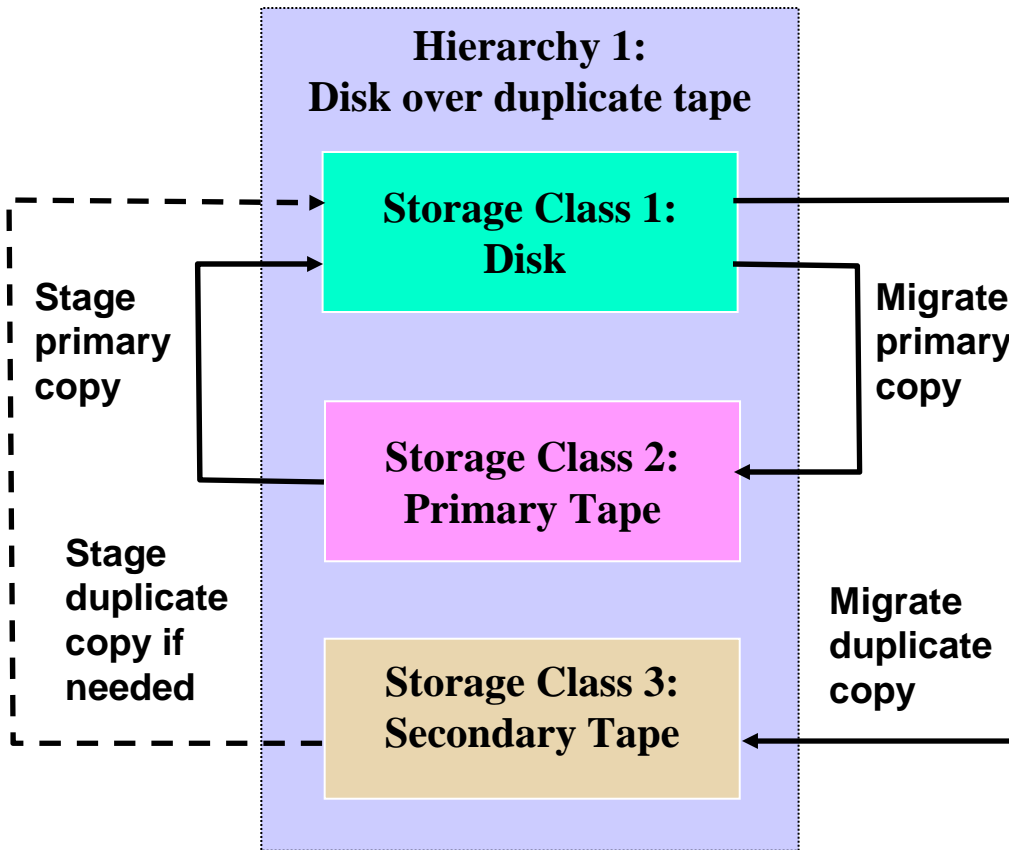
**STK 9940B – Deep Archive**



- **Storage Hierarchy**

- Consists of up to 5 levels, each of which is a separate Storage Class.
- New files are stored at the top of the hierarchy (level 1).
- Files are copied down the hierarchy via **migrate** operations.
- After migration to a lower level, file data may be deleted from a storage class via **purge** operations.
- Files are copied up the hierarchy via **stage** operations, usually when accessed after being purged from the top level of the hierarchy.

## B. HSM Concepts



### Migrate and Stage Operations

- File is migrated from disk to primary tape.
- Asynchronously, probably in parallel, file is migrated from disk to secondary tape.
- After some time of inactivity, and after migration is complete, file is purged from disk to free up space.
- When next referenced, file is staged back from primary tape to disk.
- If primary tape copy fails, file is staged from secondary tape back to disk, with operator approval.
- Many other storage hierarchies are possible, including
  - **disk only**
  - **Fast disk to high capacity disk to tape**
  - **tape only**
  - **disk to striped tape**
- Hierarchies may be up to five levels.

## B. HSM Concepts

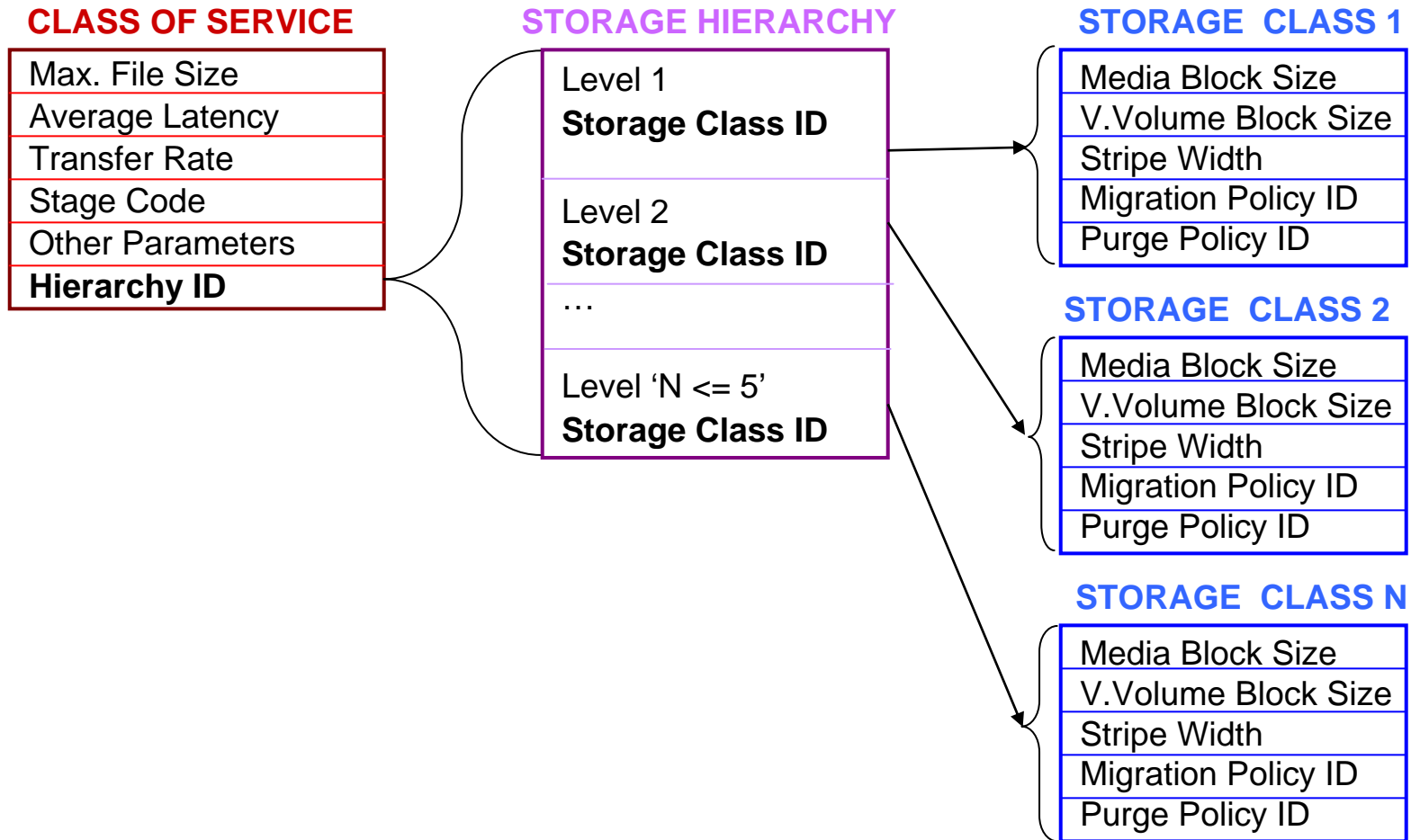
- **Class of Service (COS)**

- Associates a staging code, file size range, and other performance-related characteristics with a hierarchy.
- A given hierarchy may be associated with multiple COS, but each COS is associated with exactly one hierarchy.
- All HPSS files are assigned to a COS when created.
- Some HPSS interfaces allow the end user to specify the desired COS.

- **Migration, purge, and stage operations**

- Migration, purge, and stage operations are based on policies, usage patterns, and storage availability.
- Migration and purge policies are associated with storage classes.
- Stage behavior is determined by the class of service.

## B. HSM Concepts



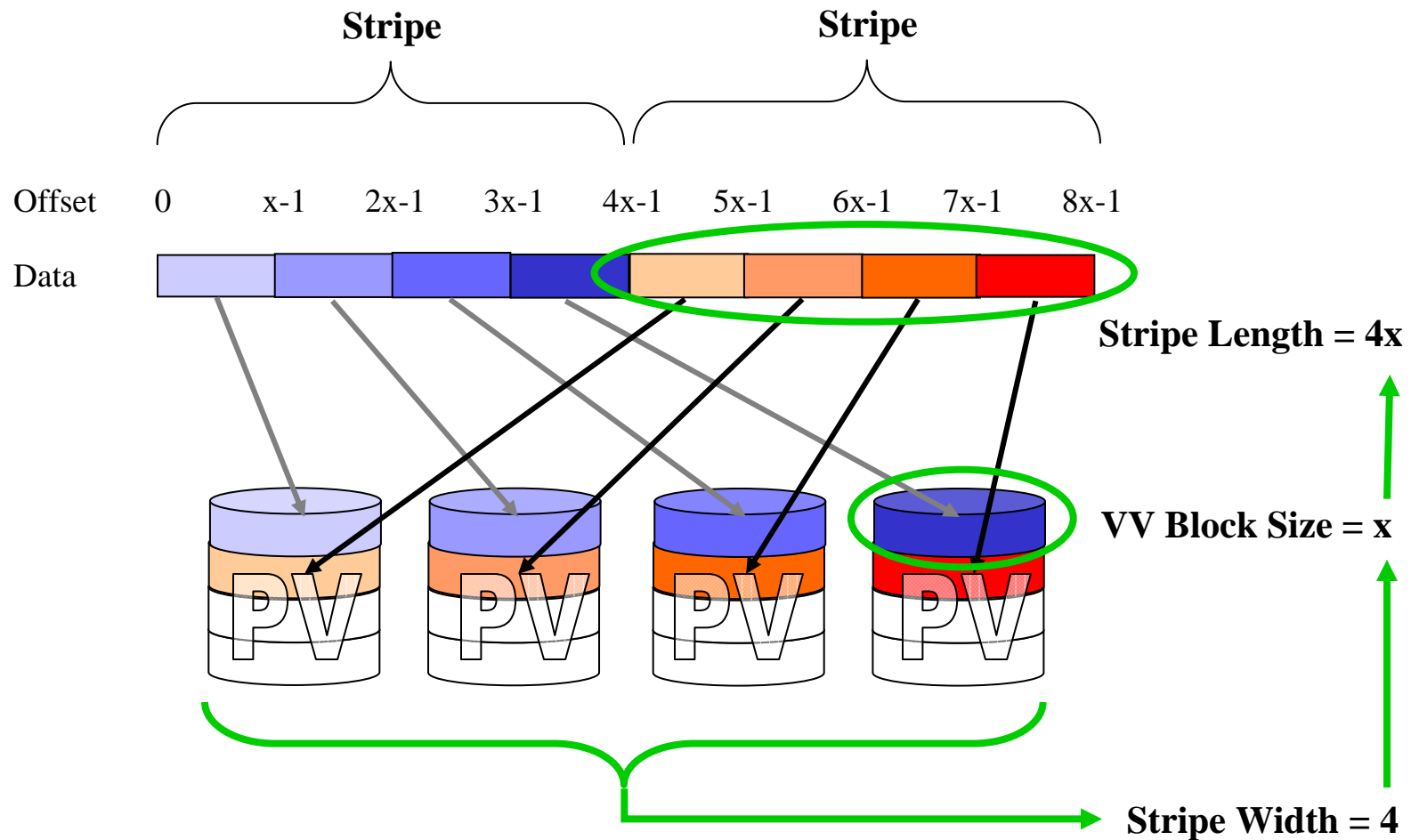
### Relationship of Class of Service, Storage Hierarchy, and Storage Class

## C. Storage Allocation Concepts

- **Physical Volume**
  - Storage media on which HPSS stores data.
  - For disk, this is an OS-level raw device (e.g., /dev/rhdisk5).
  - For tape, this is a cartridge (e.g., A00950).
- **Virtual Volume**
  - Striped group of 1 or more Physical Volumes.
  - StripeWidth=1 is equivalent to no striping, but VVs exist nevertheless.
  - Each Virtual Volume is assigned to a Storage Class.
- **Data Striping**
  - Distributing sequential data across multiple volumes.
  - Used to improve performance by taking advantage of hardware parallelism.
- **Virtual Volume Block Size**
  - Amount of data written to each PV in a stripe before switching to the next PV.
- **Stripe Width**
  - Number of Physical Volumes grouped together to form a Virtual Volume.
- **Stripe Length**
  - Number of bytes written to span all Physical Volumes of a Virtual Volume.
  - Equals Stripe Width times the Virtual Volume Block Size.

## C. Storage Allocation Concepts

- Physical layout of a VV in a 4-way disk storage class:



## C. Storage Allocation Concepts

- **Bitfile**
  - Logical sequence of bits that constitutes the data for a single file.
- **Storage Map**
  - A data structure used by the Core Server to manage the allocation of storage space for a VV.
    - **Keep track of used and available space for a VV.**
    - **Metadata is read into memory for faster operations.**
- **Storage Segment**
  - Space allocated for a file from using a storage map.
  - **Disk**
    - **A cluster is the smallest allocation unit assigned to a disk volume.**
    - **An extent is a contiguous set of clusters on a single volume.**
    - **A disk storage segment can have up to eight non-contiguous extents.**
    - **The disk storage segment size for a file can be of fixed or variable length, depending on the configured allocation method.**
  - **Tape**
    - **Tape storage segments are created with variable lengths to satisfy requests.**

## C. Storage Allocation Concepts

- **Sparse Tapes**

- Over time, some tapes become sparse as results of file deletions.
- Tape becomes sparse because we can only append to tapes.
- When tape segments are “deleted” or become inactive, this segment space cannot be reused.
- Need to repack to move active segments off of sparse tapes and then reclaim the newly empty tapes for reuse.

- **Repack**

- Moves active segments from sparse VVs to free VVs in the same storage class.
- Invoke via SSM GUI or command line utility.
- Can automatically select candidate VVs for a specified storage class.

- **Reclaim**

- Resets the state of empty tape VVs so they can be reused.
- Invoke via SSM GUI or command line utility.
- Can automatically select candidate VVs.

## C. Storage Allocation Concepts

- **Retire**
  - When media needs to be removed from HPSS, it can be 'retired'
  - This prevents new storage segments from being allocated
  - Volume can be removed after it becomes empty via repack or attrition
- **Remove**
  - Remove retired-and-empty volumes and export them from HPSS.
- **Shelf Tape**
  - Allow specified tapes to be ejected from tape library to manually move to "shelves".
  - HPSS displays a tape mount request when a shelved tape needs to be mounted.
  - HPSS automatically mounts a requested tape once it is placed back in the library.

## C. Storage Allocation Concepts

- **Fileset**

- All name space objects are in filesets.
- Every Core Server has a default root fileset.
- All other filesets are disjoint directory subtrees, administered as a unit, that can be mounted in the global name space.
- Similar to a UNIX file system.

- **Junction**

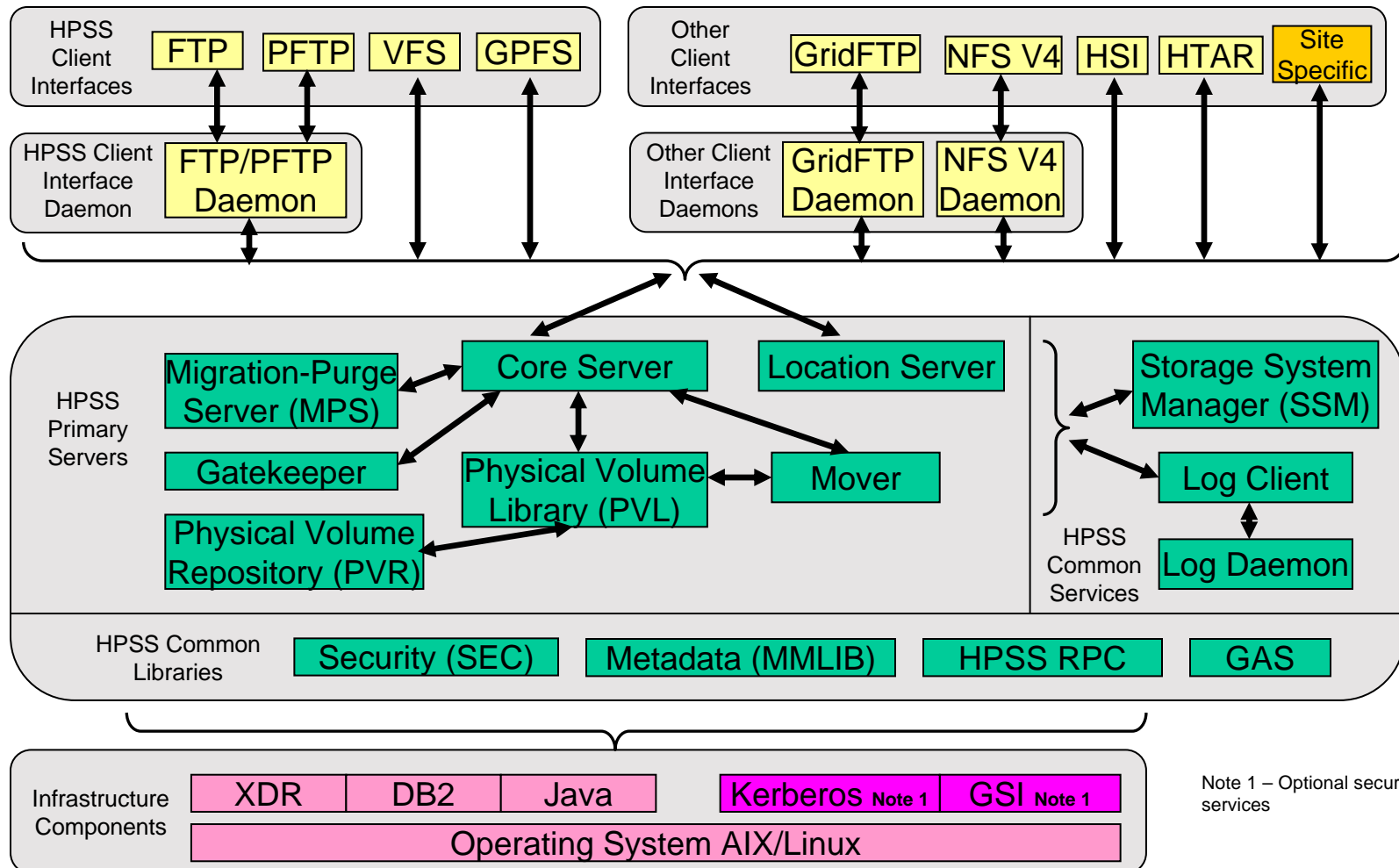
- Name space object used to attach a fileset to a specified point in the HPSS name space.
- Similar to a UNIX mount point.

- **File Families**

- A Family ID may be assigned to a fileset when the fileset is created.
- A Family ID can be assigned to a file when it is created.
- Data for files in a family will be segregated to tapes assigned to the file family.
- How are tapes assigned to a family?
  - **The Core Server allocates free space on tapes already associated with the requested family if possible. Otherwise, it dynamically assigns tapes from the set of free tapes with no assigned family (family 0).**
  - **Reclaim resets the Family Id of tapes back to the default (family 0).**

# D. HPSS Architecture

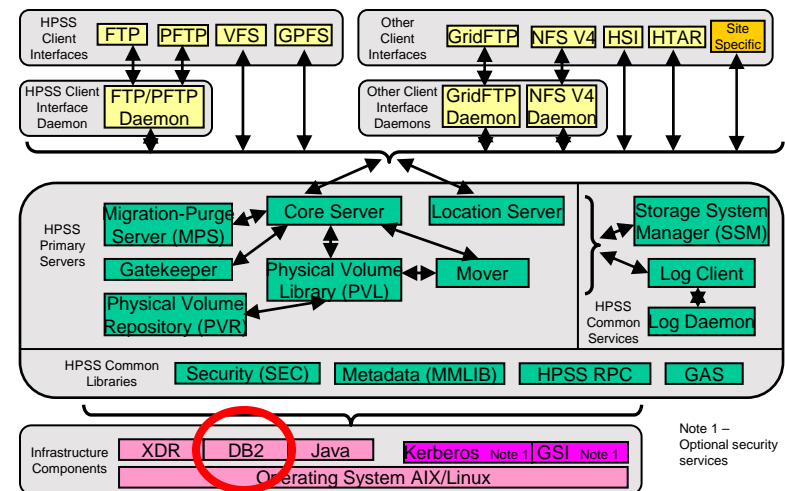
- Shows interaction between HPSS components.
- Communication occurs via remote procedure calls (RPCs).



## D. HPSS Architecture - Infrastructure Components

- DB2

- Manages HPSS metadata.
- Provides transactional integrity.
  - **Guarantee consistency of metadata in case of component failures.**
- Allows use of standard DB2 backup/restore features to backup/restore HPSS metadata.

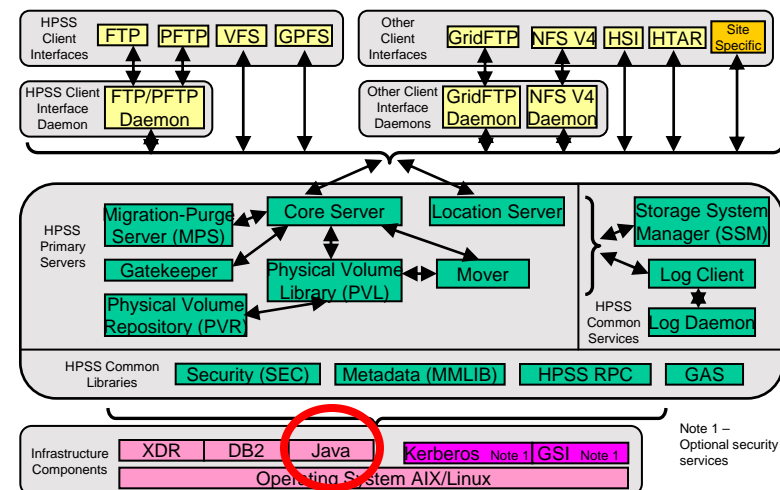


## D. HPSS Architecture - Infrastructure Components

- Java

- Requires to run:

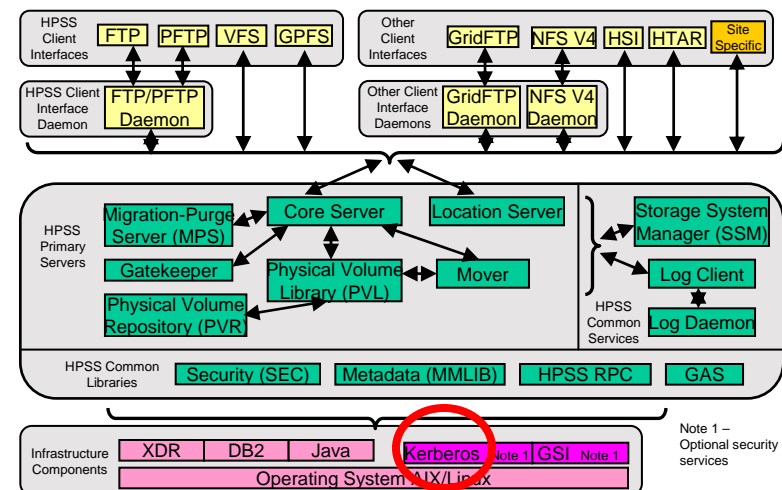
- SSM graphical user interface (hpssgui).
- SSM command line interface (hpssadm).
- mkhpss utility.



## D. HPSS Architecture - Infrastructure Components

- Kerberos (Optional)

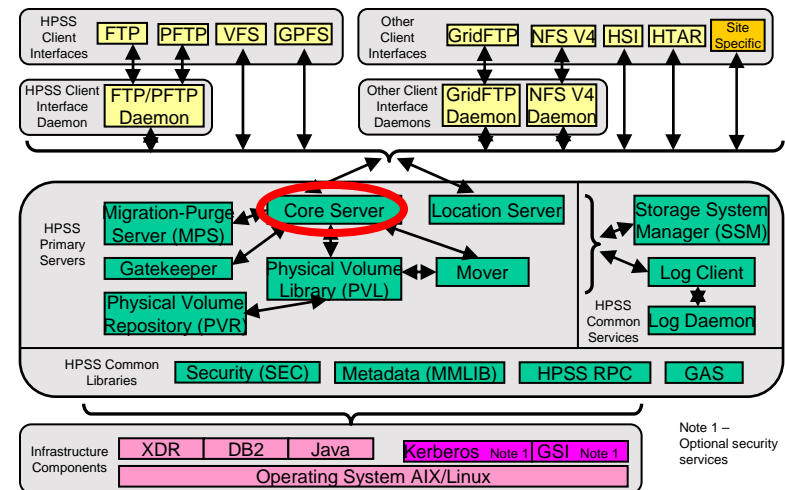
- Provides Kerberos authentication.
- Not require if using UNIX authentication.



## D. HPSS Architecture - HPSS Server Components

- Core Server

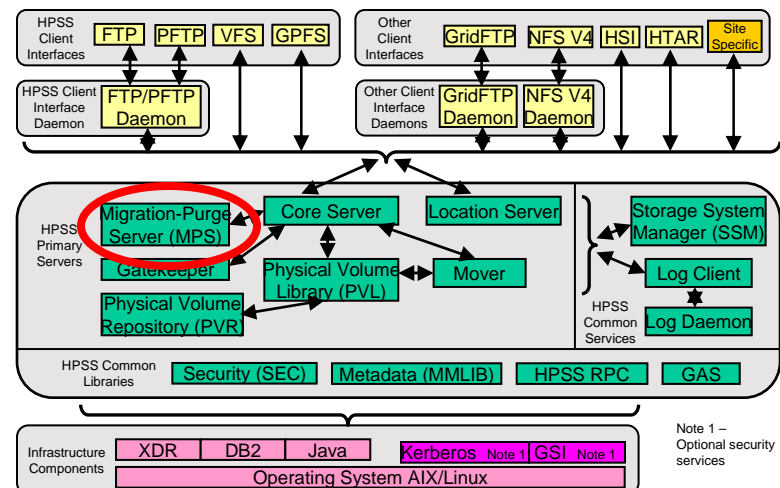
- Manages the HPSS name space.
- Enforces security (e.g., file access permissions).
- Handles I/O and name space requests from HPSS clients.
- Performs file staging as requested by HPSS clients.
- Manages HPSS storage and storage allocation.
- Performs file migration and purge as requested by MPS.
- Directs the PVL and Movers to fulfill client requests.



## D. HPSS Architecture - HPSS Server Components

- Migration / Purge Server (MPS)

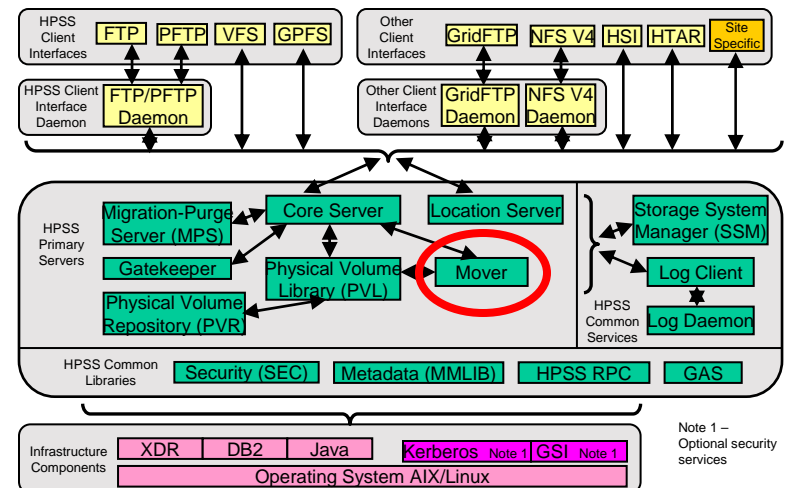
- Executes defined migration and purge policies to manage HPSS storage space.
- Submits migration and purge requests to the Core Server when policies dictate.



## D. HPSS Architecture - HPSS Server Components

- **Mover (MVR)**

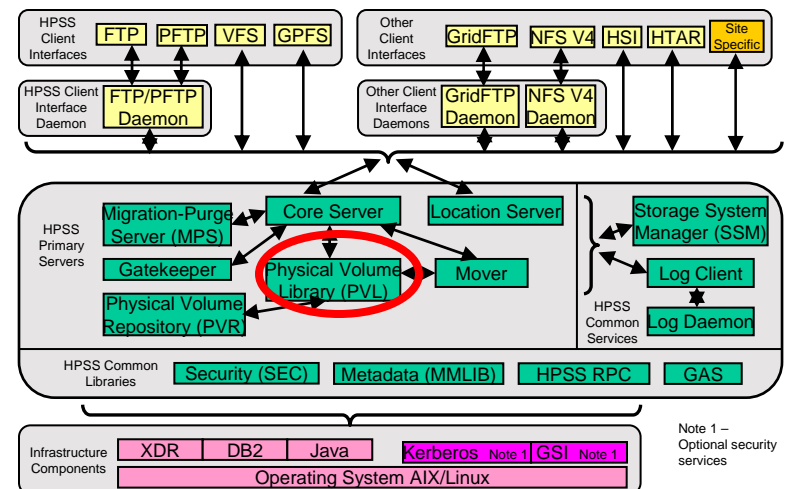
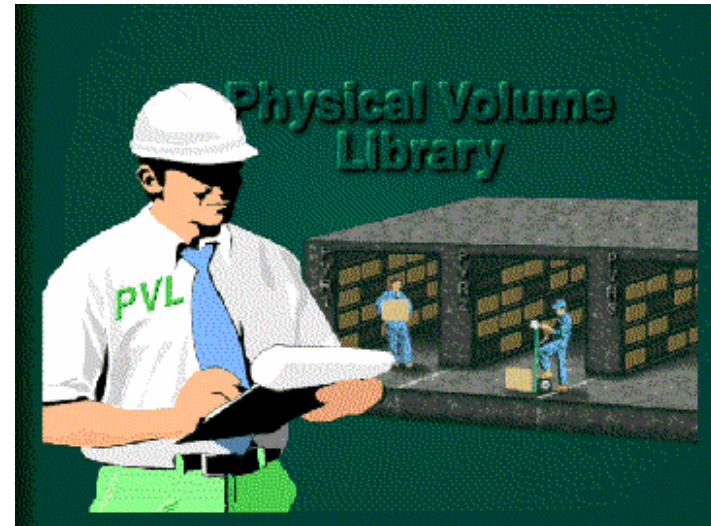
- Transfers data from one location to another.
  - **A location may be a standard I/O device (e.g., disk or tape), a network address, or a shared memory location.**
- A single Mover can handle I/O for up to 64 disk and/or tape devices on a single node.
- Writes and verifies HPSS media labels.



## D. HPSS Architecture - HPSS Server Components

- **Physical Volume Library (PVL)**

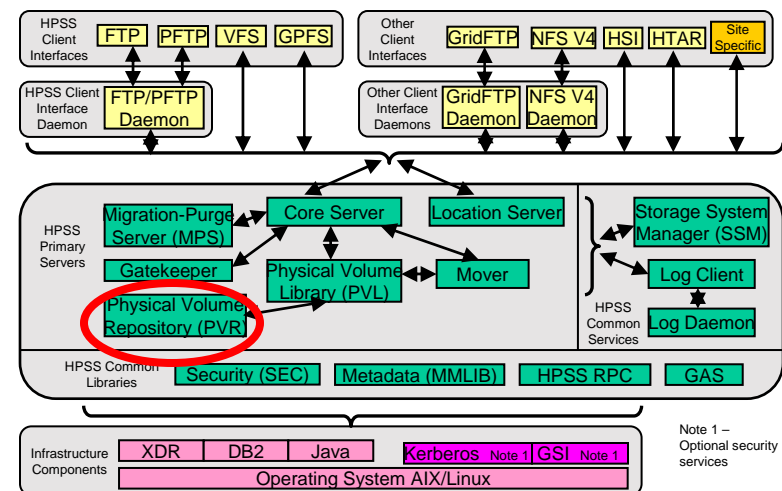
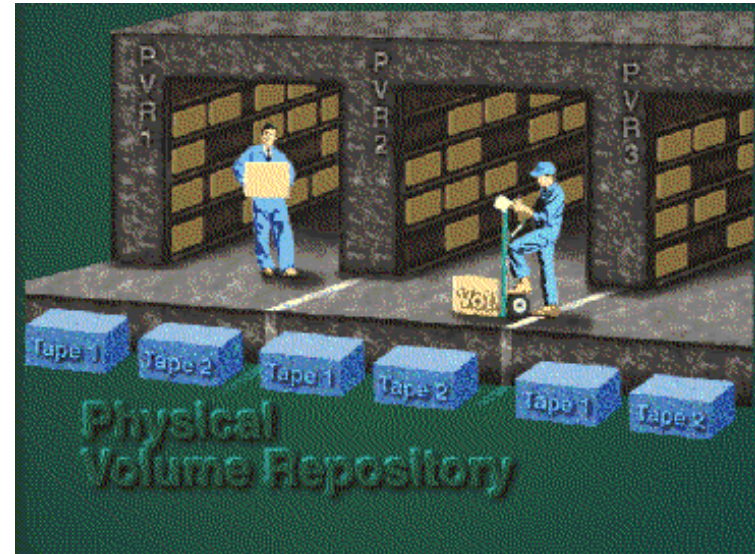
- Manages all HPSS disk and tape physical volumes.
- Coordinates all configured PVRs to mount/dismount/eject tape cartridges as directed by Core Server.
- Directs Movers to verify HPSS media labels prior to allowing I/O operations.



## D. HPSS Architecture - HPSS Server Components

### • Physical Volume Repository (PVR)

- Manages tape cartridges within a particular robotic library.
- Controls all robotic movements within the library (mount, dismount, etc...).
- Each library class requires a unique PVR. Currently, the following PVR are supported:
  - **3494 PVR:** Support 3494 library.
  - **STK PVR:** Support ACSLS libraries.
  - **ADIC AML:** Support AML and S10K library - by special bid only.
  - **SCSI PVR:** Support IBM (3584/3582/TS3500), STK (L40/SL500/SL8500), ADIC i500, and Spectra Logic libraries.
  - **Operator PVR:** Support operator mounted drives not under the control of a robotics device.

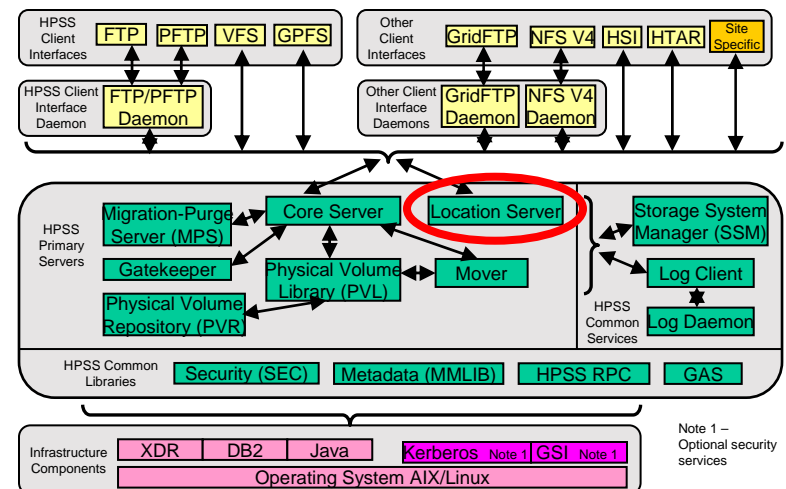


## D. HPSS Architecture - HPSS Server Components

- **Location Server (LS)**

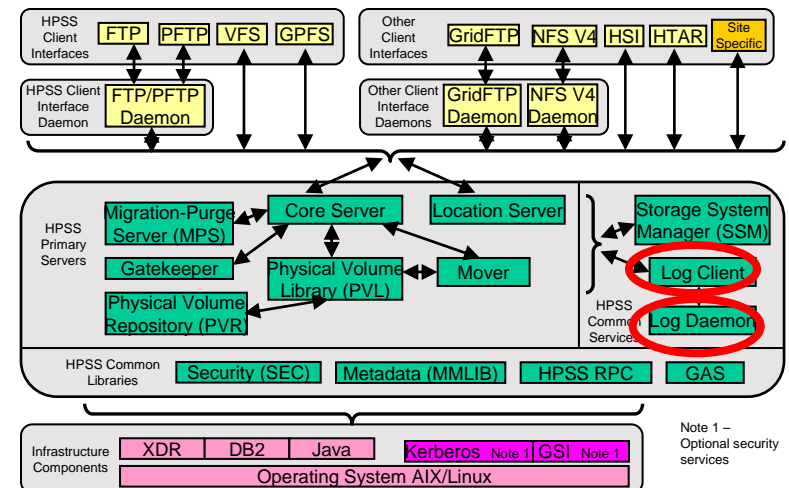
- Serves as an information clearinghouse:

- **Allows user interfaces to contact the Core Server.**



## D. HPSS Architecture - HPSS Server Components

- **Log Daemon (LOGD)**
  - Maintains the central HPSS logs (binary format).
  - Optionally archives the central HPSS logs into HPSS namespace.
- **Log Client (LOGC)**
  - Receives log messages from all HPSS servers on a particular node.
  - Applies each server's log policy
    - **Decides which message types are logged in HPSS log files.**
    - **Decides which message types are sent to SSM to be displayed.**
  - Writes appropriate messages to the local ASCII log file.
  - Forwards appropriate messages to the LOGD for central logging.
  - Forwards appropriate log messages to SSM for display.



# D. HPSS Architecture - HPSS Server Components

## Java Based

- **Storage System Manager (SSM)**

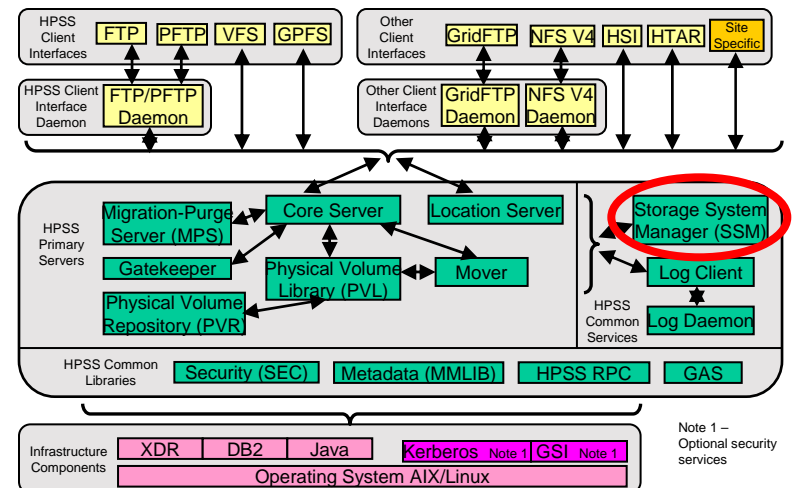
- Allows administrators to interactively configure, monitor and control HPSS.
- Can be administered via either of two interfaces.

- **Graphical User Interface (GUI) – hpssgui.**

- **Command line interface - hpssadm .**

- **Same capability as for hpssgui, minus the following configuration options:**

- » Hierarchies.
- » Storage Subsystems.
- » Logging Policies.
- » Migration Policies (Disk/Tape).
- » Purge Policies.
- » Subsystem-specific Thresholds.



## D. HPSS Architecture - Storage Subsystems

- **Storage Subsystems**

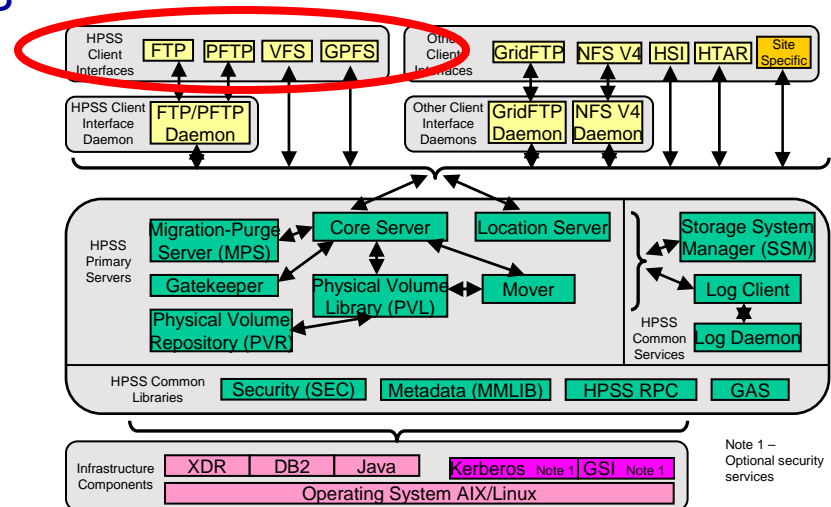
- An HPSS system consists of one or more storage subsystems (or subsystems).
- Each subsystem is serviced by its own Core Server and MPS.
- Each subsystem's metadata is stored in its own DB2 instance.
- Each subsystem has its own pool of VVs for each storage class.
- Files are assigned to a particular subsystem by their location in the HPSS name space.
- A subsystems is not required to support all of the configured COS.
- Migration and purge policies can be modified to act differently depending on the subsystem.

## D. HPSS Architecture - Storage Subsystems

- **Why use multiple subsystems?**
  - To distribute the database processing overhead between multiple nodes.
    - For WAN-based HPSS systems.
    - For systems where DB2 would otherwise become a bottleneck.
  - To segregate HPSS resources (nodes, media, networks, etc.) among multiple user groups.
    - When users supply their own nodes, networks, media, etc.
    - When users do not want to “step on” one another when putting the system under heavy load.

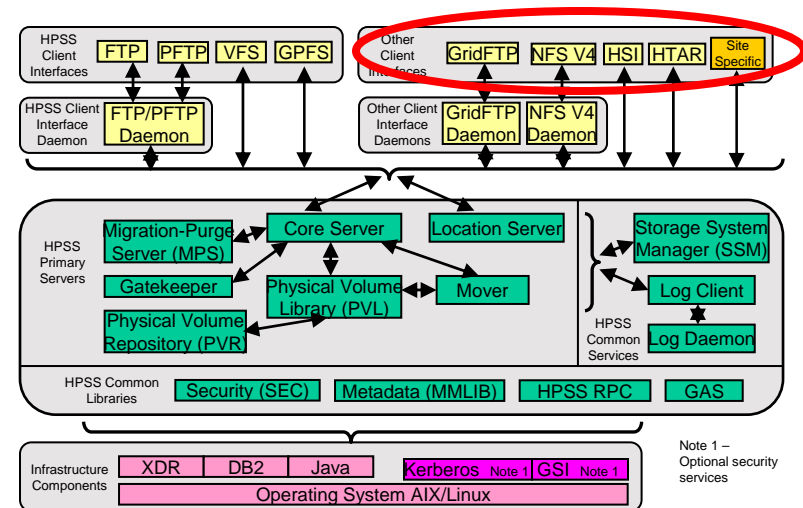
## D. HPSS Architecture - Client Interfaces

- **FTP**
  - Implements standard FTP command set.
  - No special client code required.
- **Parallel FTP (pftp\_client)**
  - Augments command set to support parallel data paths for HPSS transfers.
  - Transfers data directly from Mover to client without routing through the FTP Daemon.
- **HPSS VFS Interface Client**
  - Enables HPSS to appear as a local file system on Linux machines.
- **GPFS/HPSS Interface (GHI)**
  - Presents GPFS users with an infinite file system.
  - HPSS will archive GPFS data and stage it on demand.



## D. HPSS Architecture - Client Interfaces

- **HSI**
  - Provides Unix-style command interface in interactive, batch, and command modes.
  - Available from Gleicher Enterprises, Inc.
- **HTAR**
  - Bundles small files into large files and store into HPSS.
  - Available from Gleicher Enterprises, Inc.
- **GridFTP enabled for HPSS**
  - An extension of the standard FTP protocol for the GRID Computing environment.
  - Available from ANL.
- **Network File System (NFS V4)**
  - Allows user to access HPSS as a native file system.
  - Available from CEA-DAM.
- **Site-specific**
  - Developed and owned by HPSS sites.



## Disclosure

- **Please obtain and read product documentation before deciding to acquire HPSS.**
  - Documentation includes the HPSS License Agreement, the Statement of Work, and HPSS and other product manuals.
  - In case of conflict between information herein and product documentation, the documentation shall take precedence.
- **Forward looking information including schedules and future product capabilities reflect current planning that may change and should not be taken as commitments by IBM.**
- **IBM may at its sole discretion discontinue, add, or change HPSS features and function without notice.**