HPSS Basics

This document is edited from Chapter 2 of the HPSS Installation Guide for HPSS 7.3, available in the Documentation section of the HPSS web site www.hpss-collaboration.org.

1. Introduction

The High Performance Storage System (HPSS) provides hierarchical storage management and services for very large storage environments. HPSS may be of interest to organizations having present and future scalability requirements that are very demanding in terms of total storage capacity, file sizes, data rates, number of objects stored, and numbers of users. HPSS is part of an open, distributed environment based on remote procedure calls, Kerberos, LDAP directory systems and DB2 which form the infrastructure of HPSS. HPSS is the result of a collaborative effort by leading US Government supercomputer laboratories and industry to address real and urgent high-end storage requirements. HPSS is offered commercially by IBM.

HPSS provides scalable parallel storage systems for highly parallel computers as well as traditional supercomputers and workstation clusters. Concentrating on meeting the high end of storage system and data management requirements, HPSS is scalable and designed to store up to petabytes (10^15 bytes) of data and to use network-connected storage devices to transfer data at rates up to multiple gigabytes (10^9 bytes) per second.

HPSS provides a large degree of control for the customer to manage the hierarchical storage system. Using configuration information defined by the customer, HPSS organizes storage devices into multiple storage hierarchies. Based on policy information defined by the customer and actual usage information, data are then moved to the appropriate storage hierarchy and to appropriate levels in the storage hierarchy.

2. HPSS Capabilities

A primary goal of HPSS is to move large files between storage devices and parallel or clustered computers at speeds many times faster than today’s commercial storage system software products and to do this in a way that is more reliable and manageable than is possible with current systems. In order to accomplish this goal, HPSS is designed and implemented based on the concepts described in the following subsections.

2.1. Network-centered Architecture

The focus of HPSS is the network, not a single processor as in conventional storage systems. HPSS provides servers that can be distributed across a high performance network to provide scalability and parallelism. The basis for this architecture is the IEEE Mass Storage System Reference Model, Version

2.2. High Data Transfer Rate

HPSS achieves high data transfer rates by eliminating overhead normally associated with data transfer operations. In general, HPSS servers establish and control transfer sessions but are not involved in actual transfer of data.

2.3. Parallel Operation

The HPSS Client Application Program Interface (Client API) supports parallel or sequential access to storage devices by clients executing parallel or sequential applications. HPSS also provides a Parallel File Transfer Protocol. HPSS can even manage data transfers in a situation where the number of data sources differs from the number of destination sources. Parallel data transfer is vital in situations that demand fast access to very large files.

2.4. Based on Standard Components

HPSS runs on UNIX and is written in ANSI C and Java. It uses remote procedure calls, a selectable security service (Kerberos or UNIX), UNIX or LDAP for user configuration information, and DB2 as the basis for its portable, distributed, transaction-based architecture. These components are offered on many vendors' platforms.

The full HPSS system has been implemented on IBM AIX and LINUX platforms, and some components of HPSS have been ported to other platforms. Refer to Section 2.4: HPSS Hardware Platforms on page 34 and Section 3.3: Prerequisite Software Considerations on page 43 for specific information.

To aid vendors and users in porting HPSS to new platforms, HPSS source code is available.

2.5. Data Integrity Through Transaction Management

Transactional metadata management, provided by DB2, enables a reliable design that protects user data both from unauthorized use and from corruption or loss. A transaction is an atomic grouping of metadata management functions managed so that either all of them take place together or none of them takes place. Journaling makes it possible to back out any partially complete transactions if a failure occurs. Transaction technology is common in relational data management systems but not in storage systems. It is the key to maintaining reliability and security while scaling upward into a large, distributed storage environment.

2.6. Multiple Hierarchies and Classes of Services

Most other storage management systems support simple storage hierarchies consisting of one kind of disk and one kind of tape. HPSS provides multiple configurable hierarchies, which are particularly useful when inserting new storage technologies over time. As new disks or tapes are added, new classes of service can be set up. HPSS files reside in a particular class of service which users select based on parameters such as file size and performance. A class of service is implemented by a storage hierarchy which in turn consists of multiple storage classes, as shown in Figure 2. Storage classes are used to logically group storage media to provide storage for HPSS files. A hierarchy may be as simple as a single tape, or it may consist of two or more levels of disk and local tape. The user can even set up classes of service so that data from an older type of tape is subsequently migrated to a new type of tape. Such a procedure allows migration to new media over time without having to copy all the old media at once.

2.7. Storage Subsystems

Storage subsystems (or simply, “subsystems”) may be used to increase the scalability of HPSS in handling concurrent requests or to meet local political needs. Each subsystem contains a single Core Server. If migration and purge are needed for the subsystem, then it will also contain a Migration / Purge Server. In addition, if HPSS is to be used as a backing store for a 'linked' filesystem such as XFS, a DMAP Gateway will be required. All other servers are subsystem-independent.

Data stored within HPSS is assigned to different subsystems based on pathname resolution. A pathname consisting of ‘/’ resolves to the root Core Server which is specified in the global configuration file.

However, if the pathname contains junction components, it may resolve to a Core Server in a different subsystem. For example, the pathname ‘/JunctionToSubsys2/mydir’ could lead to a fileset managed by the Core Server in subsystem 2. Sites which do not wish to partition their HPSS through the use of subsystems will run HPSS with a single subsystem.

3. HPSS Components

The components of HPSS include files, filesets, junctions, virtual volumes, physical volumes, storage segments, metadata, servers, infrastructure, user interfaces, a management interface, and policies. Metadata is control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in tables in a DB2 relational database. Media and file metadata are represented by data structures that describe the attributes and characteristics of storage system components such as files, filesets, junctions, storage segments, and volumes. Servers are the processes that control the logic of the system and control movement of the data. The HPSS infrastructure provides the services that are used by all the servers for operations such as sending messages and providing reliable transaction management. User interfaces provide several different views of HPSS to applications with different needs. The management interface provides a way to administer and control the storage system and implement site policy.

These HPSS components are discussed below in Sections 3.1 through 3.7.

3.1. HPSS Files, Filesets, Volumes, Storage Segments and Related

Metadata. The various metadata constructs used to describe the HPSS namespace and HPSS storage are described below:

Files (Bitfiles). Files in HPSS, called bitfiles in deference to IEEE Mass Storage Reference Model terminology, are logical strings of bytes, even though a particular bitfile may have a structure imposed by its owner. This unstructured view decouples HPSS from any particular file management system that host clients of HPSS might use. HPSS bitfile size is limited to264 -1 bytes. Each bitfile is identified by a machine-generated name called a bitfile ID. A bitfile may also have a human readable name. It is the job of the HPSS Core Server (discussed in Section 2.3.2) to map a human readable name to a bitfile's ID.

Filesets. A fileset is a logical collection of files that can be managed as a single administrative unit, or more simply, a disjoint directory tree. A fileset has two identifiers: a human readable name and a numeric fileset ID. Both identifiers are unique to a given realm.

Junctions. A junction is a Core Server object, much like a symbolic link to a directory, that is used to point to a fileset. This fileset may belong to the same Core Server or to a different Core Server. When pointing to a different Core Server, junctions allow HPSS users to traverse to different subsystems.

File Families. HPSS files can be grouped into families. All files in a given family are recorded on a set of tapes assigned to the family. Only files from the given family are recorded on these tapes. HPSS supports grouping files on tape volumes only. A family can be selected at the time a file is created by supplying the appropriate family id as one of the create parameters. All files created in the fileset belong to the family. When one of these files is migrated from disk to tape, it is recorded on a tape with other files in the same family. If no tape virtual volume is associated with the family, a blank tape is reassigned from the default family. The family affiliation is preserved when tapes are repacked.

Physical Volumes. A physical volume is a unit of storage media on which HPSS stores data. The media can be removable (e.g., cartridge tape, optical disk) or non-removable (magnetic disk). Physical volumes may also be composite media, such as RAID disks, but must be represented by the host OS as a single device. Physical volumes are not visible to the end user. The end user stores bitfiles into a logically unlimited storage space. HPSS, however, must implement this storage on a variety of types and quantities of physical volumes.

Virtual Volumes. A virtual volume is used by the Core Server to provide a logical abstraction or mapping of physical volumes. A virtual volume may include one or more physical volumes. Striping of storage media is accomplished by the Core Servers by collecting more than one physical volume into a single virtual volume. A virtual volume is primarily used inside of HPSS, thus hidden from the user, but its existence benefits the user by making the user's data independent of device characteristics. Virtual volumes are organized as strings of bytes up to (2^64-1) bytes in length that can be addressed by an offset into the virtual volume.

Storage Segments. A storage segment is an abstract storage object which is mapped onto a virtual volume. Each storage segment is associated with a storage class (defined below) and has a certain measure of location transparency. The Core Server (discussed in Section 3.2) uses both disk and tape storage segments as its primary method of obtaining and accessing HPSS storage resources. Mappings of storage segments onto virtual volumes are maintained by the HPSS Core Servers.

Storage Maps. A storage map is a data structure used by the Core Server to manage the allocation of storage space on virtual volumes.

Storage Classes. A storage class defines a set of characteristics and usage parameters to be associated with a particular grouping of HPSS virtual volumes. Each virtual volume and its associated physical volumes belong to a single storage class in HPSS. Storage classes in turn are grouped to form storage hierarchies (see below). An HPSS storage class is used to logically group storage media to provide storage for HPSS files with specific intended usage, similar size and usage characteristics.

Storage Hierarchies. An HPSS storage hierarchy defines the storage classes on which files in that hierarchy are to be stored. A hierarchy consists of multiple levels of storage, with each level representing a different storage class. Files are moved up and down the hierarchy via migrate and stage operations based on usage patterns, storage availability, and site policies. For example, a storage hierarchy might consist of a fast disk, followed by a fast data transfer and medium storage capacity robot tape system, which in turn is followed by a large data storage capacity but relatively slow data transfer tape robot system. Files are placed on a particular level in the hierarchy depending upon the migration levels that are associated with each level in the hierarchy. Multiple copies are controlled by this mechanism. Also data can be placed at higher levels in the hierarchy by staging operations. The staging and migrating of data is shown in Figure 1: File Migration and Stage Operations.

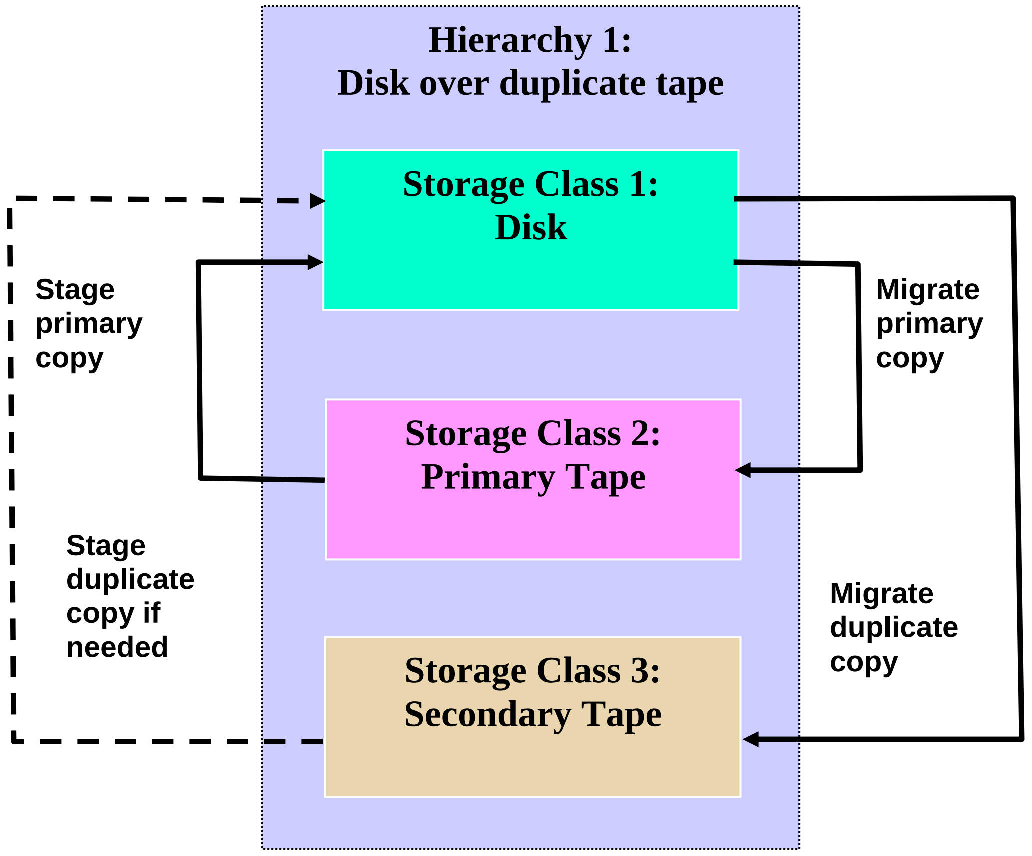

Figure 1. File Migration and Stage Operations

In Figure 1, the following sequence of events is represented:

- File is migrated from disk to primary tape.
- Asychronously, 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.

Class of Service (COS). Each bitfile has an attribute called Class Of Service. The COS defines a set of parameters associated with operational and performance characteristics of a bitfile. The COS results in the bitfile being stored in a storage hierarchy suitable for its anticipated and actual size and usage characteristics. Figure 2: Class of Service/Hierarchy/Storage Class shows the relationship between COS, storage hierarchies, and storage classes.

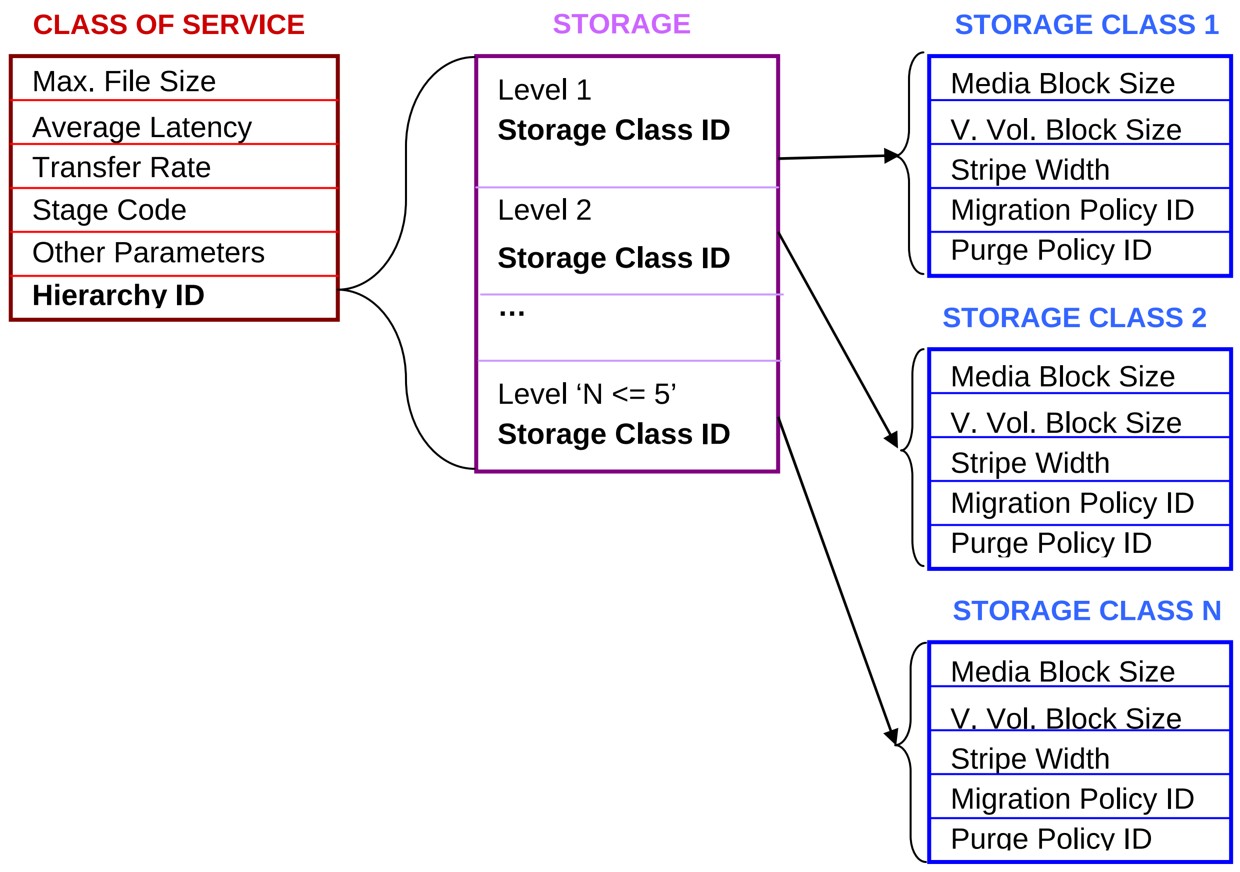

Figure 2, Relationship of Class of Service, Storage Hierarchy, and Storage Class.

User-defined Attributes (UDAs). User-defined Attributes in HPSS are client-created attributes containing additional metadata. UDAs can be associated with any namespace object. UDAs are represented internally as a well-formed XML document. The XML document size limit is 2GB. The maximum amount of data that can be returned at any one time is 512 bytes.

3.2. HPSS Servers

HPSS servers include the Core Server, Migration/Purge Server, Gatekeeper, Location Server, Log Client, Log Daemon, Physical Volume Library, Physical Volume Repository, Mover, Storage System Management System Manager, and Startup Daemon.

Core Server. The Core Server provides several key sets of functionality. First, the Core Server provides translation between human-oriented names and HPSS object identifiers. Name space objects managed by the Core Server are filesets, junctions, directories, files, hard links, and symbolic links. The Core Server provides access verification to objects and mechanisms for manipulating access to these objects via a Portable Operating System Interface (POSIX) view of the name space. This name space is a hierarchical structure consisting of directories, files, and links. These name space objects may exist within filesets that are connected via junctions.

Second, the Core Server provides the abstraction of logical bitfiles to its clients. A bitfile is identified by a Core Server generated name called a bitfile ID. Clients may reference portions of a bitfile by specifying the bitfile ID and a starting address and length. The Core Server supports random access to files and sparsely written files. It supports parallel reading and writing of data to bitfiles and performs the mapping of logical portions of bitfiles onto physical storage devices. The Core Server supports the migration, purging, and staging of data in a storage hierarchy (though the migration/purge policies are implemented through the Migration/Purge Server, a client to the Core Server).

Third, the Core Server provides a hierarchy of storage objects: storage segments, virtual volumes, and physical volumes. The Core Server translates storage segment references into virtual volume references and then into physical volume references, handles the mapping of physical resources into striped virtual volumes to allow parallel I/O to that set of resources, and schedules the mounting and dismounting of removable media through the Physical Volume Library (see below).

Migration/Purge Server (MPS). The MPS allows a site to implement its storage management policies by managing the placement of data on HPSS storage media using site-defined migration and purge policies. By making appropriate calls to its Core Server, an MPS copies data to lower levels in the hierarchy (migration), removes data from the current level once copies have been made (purge), or moves data between volumes at the same level (lateral move). Based on the hierarchy configuration, MPS can be directed to create duplicate copies of data when it is being migrated from disk or tape. This is done by copying the data to multiple lower levels in the storage hierarchy. There are three types of migration: disk migration, tape file migration, and tape volume migration. The designation disk or tape refers to the type of storage class that migration is running against. See Section 3.7.2: Migration/Purge Server on page 71 for a more complete discussion of the different types of migration.

MPS runs migration on each storage class periodically using the time interval specified in the migration policy for that class. See Section 2.3.7: HPSS Policy Modules on page 32 for details on migration and purge policies. Migration runs can be started automatically when the warning or critical space thresholds for the storage class are exceeded. In addition, migration runs can be started manually by an administrator.

Purge runs are started automatically on each storage class when the free space in that class falls below the percentage specified in the purge policy. Purge runs may also be started manually.

Disk Migration/Purge. The purpose of disk migration is to make one or more copies of disk files to lower levels in the hierarchy. The number of copies depends on the configuration of the hierarchy. For disk, migration and purge are separate operations. It is common for disk storage classes which have been configured for migration to also be configured for purge as well. Once a file has been migrated (copied) downwards in the hierarchy, it becomes eligible for purge, which subsequently removes the file from the higher level and allows the disk space to be reused.

Tape File Migration. The purpose of tape file migration is to make an additional copy (or multiple additional copies) of a file, in a tape storage class, to a lower level in the hierarchy. It is also possible to move files downwards instead of copying them. In this case there is no duplicate copy maintained. There is no separate purge component to tape file migration. Empty volumes must be reclaimed using the reclaim utility.

Tape Volume Migration. The purpose of tape volume migration is to free tape volumes for reuse. Tape volumes are selected based on being in the EOM map state and containing the most unused space (caused by users overwriting or deleting files). The remaining segments on these volumes are either migrated downwards to the next level in the hierarchy, or are moved laterally to another tape volume at the same level. This results in empty tape volumes which may then be reclaimed. Note that there is no purge component to tape volume migration. All of the operations use a move instead of a copy semantic.

Gatekeeper (GK). The Gatekeeper provides two main services:

It provides sites with the ability to schedule the use of HPSS resources using the Gatekeeping Service.

It provides sites with the ability to validate user accounts using the Account Validation Service.

Both of these services allow sites to implement their own policy.

The default Gatekeeping Service policy is to not do any gatekeeping. Sites may choose to implement a policy for monitoring authorized callers, creates, opens, and stages. The Core Server will call the appropriate GK API depending on the requests that the site-implemented policy is monitoring.

The Account Validation Service performs authorizations of user storage charges. A site may perform no authorization, default authorization, or site-customized authorization depending on how the Accounting Policy is set up and whether or not a site has written site-specific account validation code. Clients call this service when creating files, changing file ownership, or changing accounting information. If Account Validation is enabled, the Account Validation Service determines if the user is allowed to use a specific account or gives the user an account to use, if needed. The Core Server also calls this service to perform an authorization check just before account-sensitive operations take place.

Location Server (LS). The Location Server acts as an information clearinghouse to its clients through the HPSS Client API to enable them to locate servers and gather information from both local and remote HPSS systems. Its primary function is to allow a client to determine a server's location and, by knowing other information about the server such as its object UUID, determine its server type or its subsystem id. This allows a client to contact the appropriate server. Usually the Location Server is only used by the Core Server or the Gatekeeper.

Physical Volume Library (PVL). The PVL manages all HPSS physical volumes. It is in charge of mounting and dismounting sets of physical volumes, allocating drive and cartridge resources to satisfy mount and dismount requests, providing a mapping of physical volume to cartridge and of cartridge to Physical Volume Repository (PVR), and issuing commands to PVRs to perform physical mount and dismount actions. A primary function of the PVL is the support for atomic mounts of sets of cartridges for parallel access to data. Atomic mounts are implemented by the PVL, which waits until all necessary cartridge resources for a request are available before issuing mount commands to the PVRs.

Physical Volume Repository (PVR). PVRs manage HPSS cartridges. Though an HPSS system may contain multiple PVRs, each cartridge is managed by only one. PVRs provide APIs for clients to request cartridge mounts and dismounts and query the status of cartridges. For convenience, PVRs are often configured in one-to-one correspondence to tape libraries. For information on the types of tape libraries supported by HPSS PVRs, see Section 3.4.2: Robotically Mounted Tape on page 48.

An Operator PVR is provided for cartridges not under control of a robotic library. These cartridges are mounted on a set of drives by operators.

Mover. The purpose of the Mover is to transfer data from a source device to a sink device. A device can be a standard I/O device with geometry (e.g., tape, disk) or a device without geometry (e.g., network, memory). The Mover's client (typically the Core Server) describes the data to be moved and where the data is to be sent. It is the Mover's responsibility to actually transfer the data, retrying failed requests and attempting to optimize transfers. The Mover supports transfers for disk devices, tape devices and a mover protocol that can be used as a lightweight coordination and flow control mechanism for large transfers.

Log Client (LOGC). The Log Client receives log messages from each HPSS server running on its node and filters those messages based on the configured log policies. Log messages which pass the filter are written to a local log and sent on to the Log Daemon.

Log Daemon (LOGD). The Log Daemon enters log messages into the central log file and sends those messages which are to be displayed in the Alarms and Events SSM window to the SSM

Startup Daemon. The Startup Daemon is called by the SSMSM to start each HPSS server except the SSMSM, the Startup Daemon itself, and the remote portion of the Mover. A Startup Daemon is required on each node where any HPSS Server executes, with the exception that no Startup Daemon is required on nodes where the only HPSS Server is the remote portion of the Mover.

3.3. HPSS Storage Subsystems

The goal of storage subsystems (or just "subsystems") is to increase the scalability of HPSS by allowing multiple Core Servers to be used within a single HPSS system. Every HPSS system is partitioned into one or more subsystems. Each subsystem contains a single Core Server. If migration and purge are needed, then the subsystem should contain a single Migration/Purge Server. Each Core Server and each Migration/Purge Server must exist within a storage subsystem. Each subsystem may optionally be serviced by a Gatekeeper which performs site-specific user-level scheduling of HPSS storage requests or account validation. Each Gatekeeper may service multiple subsystems. If a subsystem wishes to utilize XDSM filesets, a DMAP Gateway must also be configured. Only one DMAP Gateway is needed for multiple subsystems. All other servers exist independently of storage subsystems. Sites which do not need multiple Core Servers use a single storage subsystem.

The computer that runs the Core Server for subsystem X is referred to as the “Subsystem X node”, while the computer running the Root Core Server is known as the “Root Subsystem Node”.

Each HPSS system consists of two types of DB2 databases. The global database contains subsystem- independent data, and a subsystem database contains subsystem-dependent data. An HPSS system has exactly one global database and one or more subsystem databases.

The definitions of classes of service, hierarchies, and storage classes apply to the entire HPSS system (they are subsystem-independent). All classes of service, hierarchies, and storage classes are known to all storage subsystems within HPSS. The level of resources dedicated to these entities by each storage subsystem may differ. It is possible to disable selected classes of service within given storage subsystems. Although the class of service definitions are global, if a class of service is disabled within a storage subsystem then the Core Server in that subsystem never selects that class of service.

Since the Migration/Purge Server (MPS) is contained within the storage subsystem, migration and purge operate independently in each subsystem. Each Migration/Purge Server is responsible for migration and purge for those storage class resources contained within its particular storage subsystem. Migration and purge runs are independent and are not synchronized. Migration and purge for a storage class may be configured differently for each storage subsystem. It is possible to set up a single migration or purge policy which applies to a storage class across all storage subsystems (to make configuration easier), but it is also possible to control migration and purge differently in each storage subsystem.

Similarly, storage class thresholds may be configured differently for each storage subsystem. It is possible to set up a single set of thresholds which apply to a storage class across all storage subsystems, but it is also possible to control the thresholds differently for each storage subsystem.

3.4. HPSS Infrastructure

The HPSS infrastructure items are those components and services used by the various HPSS servers. The HPSS infrastructure components common among servers are discussed below.

Remote Procedure Calls (RPC). Most HPSS servers, with the exception of the MVR, PFTPD, and logging services (see below), communicate requests and status (control information) via RPCs. HPSS does not use RPCs to move user data. RPCs provide a communication interface resembling simple, local procedure calls.

Thread Services. HPSS uses a threads package for multitasking. The threads package is vital for HPSS to serve large numbers of concurrent users and to enable multiprocessing of its servers.

Transaction Management. Requests to perform actions, such as creating bitfiles or accessing file data, result in client-server interactions between software components. The

HPSS Core Server performs most of the HPSS metadata changes using the transaction management tools provided by DB2. For the most part, these metadata transactions are managed entirely within the Core Server. Other servers such as MPS and PVL modify their metadata transactionally, and those transactions are entirely contained within those servers. A very small number of rarely performed operations require distributed transaction management, and these are handled by DB2 as well. Transactional integrity to guarantee consistency of server state and metadata is required in HPSS in case a particular component fails. HPSS metadata updates utilize the transactional capability of DB2. The selection of DB2 was based on functionality and vendor platform support. It provides HPSS with an environment in which a job or action completes successfully or is aborted completely.

DB2 provides a full suite of recovery options for metadata transactions. Recovery of the database to a consistent state after a failure of HPSS or DB2 is automatic. A full suite of database backup and maintenance tools is provided as well.

Security. HPSS security software provides mechanisms that allow HPSS components to communicate in an authenticated manner, to authorize access to HPSS objects, to enforce access control on HPSS objects, and to issue log records for security-related events. The security components of HPSS provide authentication, authorization, enforcement, and audit capabilities for the HPSS components. Customer sites may use the default security policy delivered with HPSS or define their own security policy by implementing their own version of the security policy module.

- **Authentication** — is responsible for guaranteeing that a principal (a customer identity) is the entity that is claimed, and that information received from an entity is from that entity.
- **Authorization** — is responsible for enabling an authenticated entity access to an allowed set of resources and objects. Authorization enables end user access to HPSS directories and bitfiles.
- **Enforcement** — is responsible for guaranteeing that operations are restricted to the authorized set of operations.
- **Audit** — is responsible for generating a log of security-relevant activity. HPSS audit capabilities allow sites to monitor HPSS authentication, authorization, and file security events. File security events include file creation, deletion, opening for I/O, and attribute modification operations. HPSS components that communicate with each other maintain a joint security context. The security context for both sides of the communication contains identity and authorization information for the peer principals as well as an optional encryption key.

Access to HPSS server interfaces is controlled through an Access Control List (ACL) mechanism. Membership on this ACL is controlled by the HPSS administrator.

Logging. A logging infrastructure component in HPSS provides an audit trail of server events. Logged data includes alarms, events, requests, security audit records, status records, and trace information. The Log Client, which may keep a temporary local copy of logged information, communicates log messages to a central Log Daemon, which in turn maintains a central log. Depending on the type of log message, the Log Daemon may send the message to the SSM for display purposes. When the central HPSS log fills, messages are sent to a secondary log file. A configuration option allows the filled log to be automatically archived to HPSS. A delog function is provided to extract and format log records from a central or archived log file. Delog options support filtering by time interval, record type, server, and user.

Accounting. The HPSS accounting system provides the means to collect usage information in order to allow a particular site to charge its users for the use of HPSS resources. It is the responsibility of the individual site to sort and use this information for subsequent billing based on site-specific charging policies. For more information on the HPSS accounting policy, refer to Section 3.7: HPSS Policy Modules.

3.5. HPSS User Interfaces

HPSS provides the user with a number of transfer interfaces as discussed below.

File Transfer Protocol (FTP). HPSS provides an industry-standard FTP user interface. Because standard FTP is a serial interface, data sent to a user is received serially. This does not mean that the data within HPSS is not stored and retrieved in parallel; it means that the Parallel FTP Daemon within HPSS must consolidate its internal parallel transfers into a serial data transfer to the user. HPSS FTP performance in many cases will be limited not by the speed of a single storage device but by the speed of the data path between the HPSS Parallel FTP Daemon and the user’s FTP client.

Parallel FTP (PFTP). The PFTP supports standard FTP commands plus extensions and is built to optimize performance for storing and retrieving files from HPSS by allowing data to be transferred in parallel across the network media. The parallel client interfaces have a syntax similar to FTP but with numerous extensions to allow the user to transfer data to and from HPSS across parallel communication interfaces established between the PFTP client and the HPSS Movers. This provides the potential for using multiple client nodes as well as multiple server nodes. PFTP supports transfers via TCP/IP. The PFTP client establishes a control connection with the HPSS Parallel FTP Daemon and subsequently establishes TCP/IP data connections directly with HPSS Movers to transfer data at rates limited only by the underlying media, communications hardware, and software.

Client Application Program Interface (Client API). The Client API is an HPSS-specific programming interface that mirrors the POSIX.1 specification where possible to provide ease of use to POSIX application programmers. Additional APIs are also provided to allow the programmer to take advantage of the specific features provided by HPSS (e.g., storage/access hints passed on file creation and parallel data transfers). The Client API is a programming level interface. It supports file open/create and close operations; file data and attribute access operations; file name operations; directory creation, deletion, and access operations; and working directory operations. HPSS users interested in taking advantage of parallel I/O capabilities in HPSS can add Client API calls to their applications to utilize parallel I/O. For the specific details of this interface see the HPSS Programmer’s Reference Guide.

HPSS VFS Interface. The HPSS VFS Interface presents a standard POSIX I/O interface to a user application. This obviates the need for a user application to be rewritten against the HPSS Client API and hence can be used “out of the box” as long as the user application is POSIX compliant. A portion of an HPSS directory tree can be mounted on a client machine as if it were a local POSIX-compliant filesystem.

3.6. HPSS Storage System Management

The HPSS Storage System Manager provides operators and administrators access by both a command line interface and a graphical user interface. Monitoring capabilities include the ability to query the values of important management attributes of storage system resources and the ability to receive notifications of alarms and other significant system events. Controlling capabilities include the ability to start up and shut down servers and the ability to set the values of management attributes of storage system resources and storage system policy parameters. Additionally, SSM can request that specific operations be performed on resources within the storage system, such as adding and deleting logical or physical resources.

Graphical User Interface. HPSS provides a graphical user interface, the SSM hpssgui, for HPSS administration and operations GUI. The hpssgui simplifies the management of HPSS by organizing a broad range of technical data into a series of easy-to-read graphic displays. The hpssgui allows monitoring and control of virtually all HPSS processes and resources from windows that can easily be added, deleted, moved, or overlapped as desired.

Command line. HPSS also provides a command line SSM interface, hpssadm. This tool does not provide all the functionality of the hpssgui, but does implement a subset of its frequently used features, such as some monitoring and some control of servers, devices, storage classes, volumes, and alarms. It is useful for performing HPSS administration from remote locations where network traffic is slow or difficult. Additionally, hpssadm provides some rudimentary mass configuration support by means of the ability to issue configuration commands from a batch script.

Utilities. In addition to SSM, HPSS provides a number of command line utilities for specialized management purposes, such as listing the volumes managed by a particular PVR or core server. See the HPSS Management Guide Chapter 16: Management Tools for more information. See also the HPSS man pages for descriptions of these utilities.

3.7. HPSS Policy Modules

There are a number of aspects of storage management that probably will differ at each HPSS site. For instance, sites typically have their own guidelines or policies covering the implementation of accounting, security, and other storage management operations. In order to accommodate site-specific policies, HPSS has implemented flexible interfaces to its servers to allow local sites the freedom to tailor management operations to meet their particular needs.

HPSS policies are implemented using two different approaches. Under the first approach, used for migration, purge, and logging policies, sites are provided with a large number of parameters that may be used to implement local policy. Under the second approach, HPSS communicates information through a well-defined interface to a policy software module that can be completely replaced by a site. Under both approaches, HPSS provides a default policy set for users.

Migration Policy. The migration policy defines the conditions under which data is copied from one level in a storage hierarchy to one or more lower levels. Each storage class that is to have data copied from that storage class to a lower level in the hierarchy has a migration policy associated with it. The MPS uses this policy to control when files are copied and how much data is copied from the storage class in a given migration run. Migration runs are started automatically by the MPS based upon parameters in the migration policy. Note that the number of copies which migration makes and the location of these copies is determined by the definition of the storage hierarchy and not by the migration policy.

Purge Policy. The purge policy defines the conditions under which data that has already been migrated from a disk storage class can be deleted. Purge applies only to disk storage classes. It is common, but not necessary, for disk storage classes which have a migration policy to also have a purge policy. Purge runs are started automatically by the MPS based upon parameters in the purge policy.

Logging Policy. The logging policy controls the types of messages to log. On a per server basis, the message types to write to the HPSS log may be defined. In addition, for each server, options to send Alarm, Event, or Status messages to SSM may be defined.

Security Policy. Security policy defines the authorization and access controls to be used for client access to HPSS. HPSS security policies are provided to control access (authentication) from FTP and/or Parallel FTP using Username/Password, Ident, or Kerberos credentials. HPSS provides facilities for recording information about authentication and object (file/directory) creation, deletion, access, and authorization events. The security audit policy for each server determines the records that each individual server will generate. All servers can generate authentication records.

Accounting Policy. The accounting policy provides runtime information to the accounting report utility and to the Account Validation service of the Gatekeeper. It helps determine what style of accounting should be used and what level of validation should be enforced. The two types of accounting are site-style and UNIX-style. The site-style approach is the traditional type of accounting in use by most mass storage systems. Each site will have a site- specific table (Account Map) that correlates the HPSS account index number with their local account charge codes. The UNIX-style approach allows a site to use the user identifier (UID) for the account index. The UID is passed along in UNIX-style accounting just as the account index number is passed along in site-style accounting.

Account Validation allows a site to perform usage authorization of an account for a user. It is turned on by enabling the Account Validation field of the Accounting Policy configuration screen. If Account Validation is enabled, the accounting style in use at the site is determined by the Accounting Style field. A site policy module may be implemented by the local site to perform customized account validation operations. The default Account Validation behavior is performed for any Account Validation operation that is not overridden by the site policy module.

Location Policy. The location policy defines how Location Servers at a given site will perform, especially in regards to how often server location information is updated. All local, replicated Location Servers update information according to the same policy.

Gatekeeping Policy. The Gatekeeper provides a Gatekeeping Service along with an Account Validation Service. These services provide the mechanism for HPSS to communicate information though a well-defined interface to a policy software module that can be written by a site. The site policy code is placed in well-defined shared libraries for the gatekeeping policy and the accounting policy (/opt/hpss/lib/libgksite.[a|so] and /opt/hpss/lib/libacctsite.[a|so] respectively) which are linked to the Gatekeeper. The Gatekeeping policy shared library contains a default policy which does NO gatekeeping. Sites will need to enhance this library to implement local policy rules if they wish to monitor and load-balance requests.

4. HPSS Hardware Platforms

The following matrix illustrates which platforms support HPSS interfaces.

Table 1. HPSS Client Interface and Mover Platforms

Platform	Core Server	Mover	Client API	PFTP Client	VFS Client	GPFS	Generic FTP
IBM AIX	X	X	X	X		X	X
Linux (x86 or Power)	X	X	X	X	X	X	X
Sun Solaris (Big Endian ONLY)			X	X			X
Silicon Graphics IRIX (32-bit)		X		X			X