

# High Performance Storage System Virtual File System (VFS) Interface



\* Please review disclosure statement on last slide



# Definition of a VFS

---

- A virtual file system (VFS) is a local or remote file system that has been mounted so that it is accessible to the local user.
- The HPSS VFS Interface enables HPSS to be accessible to the local user and to standard Linux applications.
- This is made possible by the Linux *virtual file system switch* that enables multiple file systems to appear as local file systems to a user.
- For example, one Linux computer might have all these file systems mounted using its virtual file system switch:
  - ext3 for local disk
  - IBM GPFS for local and remote disks on a cluster
  - NFS client for access to a network filer
  - CDFS to access a CD
  - HPSS VFS to access HPSS via LAN or SAN

# VFS – an Interface to HPSS

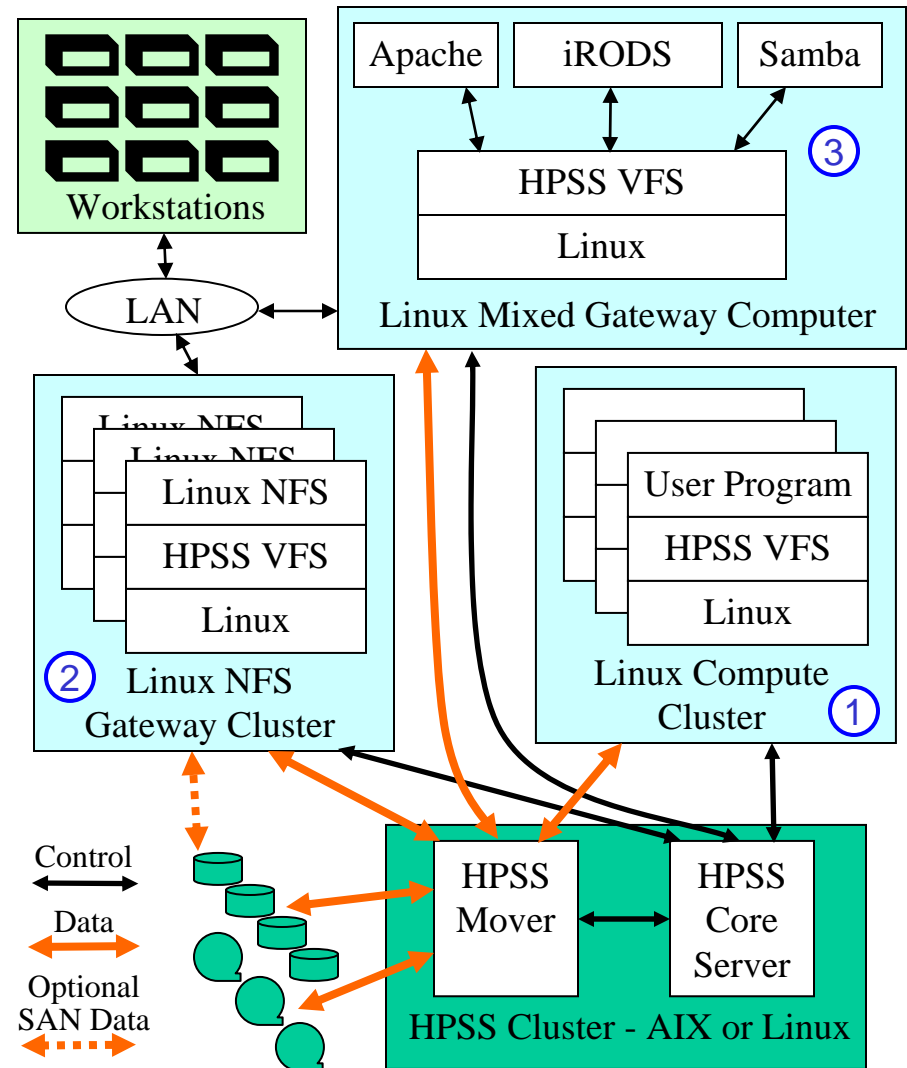


- The HPSS VFS Interface provides a standard POSIX interface for applications to use to access HPSS.
  - POSIX is an acronym for Portable Operating System Interface and is a trademark of the IEEE
  - Having a POSIX-compliant interface allows standard Linux commands, for example read( ), write( ), ls, cat, cp
  - Having a POSIX-compliant interface allows commercial applications to use HPSS for file space without modification
- The HPSS virtual file system should be used appropriately.
  - As long-term storage
  - For data that may migrate between disk and tape
  - For applications that can tolerate tape latencies
  - In other words, not as working storage

# VFS Conceptual Architecture



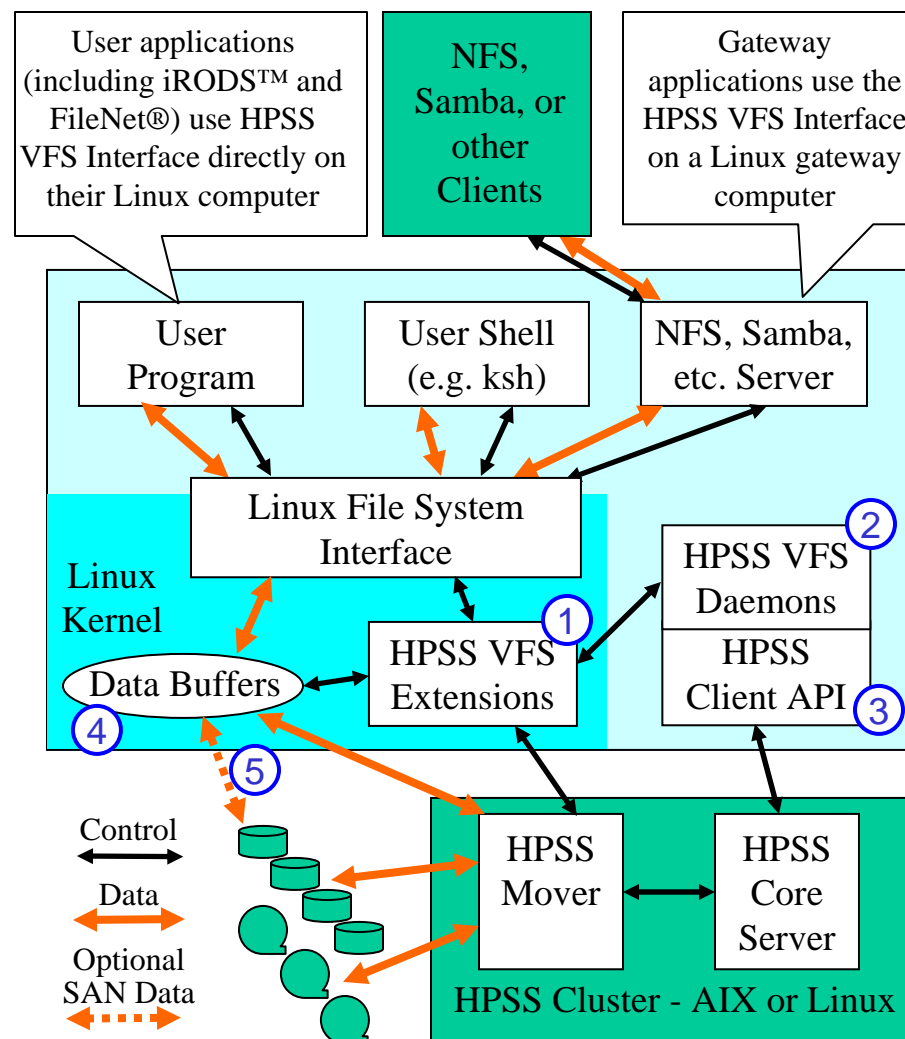
- HPSS VFS Interface provides a direct POSIX read( ) write( ) interface for a Linux computational cluster. ①
- HPSS VFS Interface enables commercial and open source Linux-based gateway applications. Examples are NFS, Secure FTP, Apache, and Samba.
  - Other examples are iRODS™ and FileNet®, providing content management for the user's files.
  - HPSS supports multiple gateway computers ② for scalable throughput.
  - HPSS supports multiple gateway applications ③ on one computer. Each gateway sees all permitted HPSS files.
- Clients may be connected by LAN or WAN.
- When using gateways, the gateways must provide authentication and security as with any other gateway service.



# VFS Conceptual Architecture



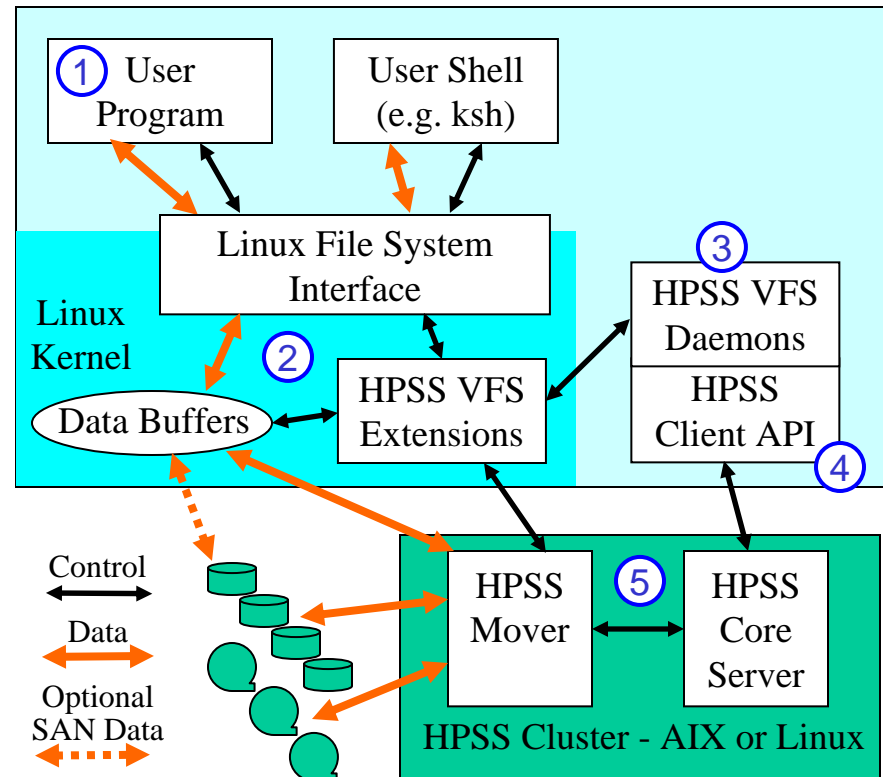
1. HPSS VFS Extensions interface with the Linux File System as would other VFS types.
2. HPSS VFS Daemons issue the attribute or I/O requests on behalf of the kernel.
3. HPSS VFS uses the HPSS Client API that underlies all HPSS I/O services to communicate with HPSS itself.
4. The HPSS Client API uses data buffers that are an extension of Linux kernel memory, allowing buffering.
5. The HPSS Client API supports the separation of command and data paths, and when appropriately configured, can take advantage of LAN-less data transfer over a SAN.



# VFS – How it works, part 1



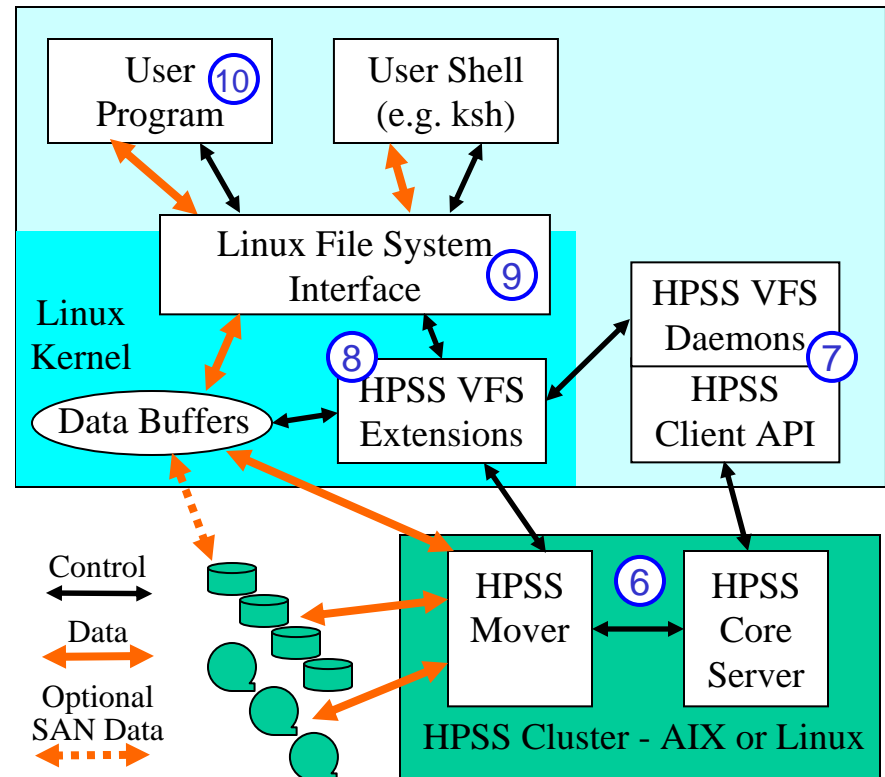
1. Application issues open() system call to open a file.
2. Kernel VFS layer provides common file system functionality, then passes control to HPSS VFS module for HPSS specific functionality.
3. HPSS VFS formats HPSS open request and forwards to HPSS Daemon via a device interface..
4. HPSS Daemon receives request and uses HPSS client api to interface with HPSS.
5. HPSS performs the open request. If permissions, path, and attributes valid, the file is created.



# VFS – How it works, part 2



6. HPSS returns success / failure and file attributes to VFS Daemon.
7. The HPSS Daemon returns information from HPSS Client API back to the HPSS Kernel Extension.
8. The HPSS Kernel extension returns the information back to the Linux VFS Layer. Data transfers have additional step of passing actual data using sockets communicating with HPSS Mover nodes.
9. The Linux VFS layer returns to the application system call.
10. Application continues with either an open file or an error code indicating the failure.



# VFS – How it works, part 3

---



- File data transfers include additional steps where HPSS Movers exchange data directly to the HPSS VFS component using sockets.
  - The Linux VFS caching feature may be able to resolve data requests without requiring HPSS VFS involvement.
  - For serial data reads, HPSS VFS provides a read-ahead algorithm to attempt to cache data before requested.
- An exception to the above is when HPSS is configured for SAN access.
  - Data is transferred directly from client system buffers to the SAN disks (as shown on an earlier slide).
- File attribute information may also be cached by the Linux VFS layer, minimizing request latency.

# HPSS Contacts

---



Jim A. Gerry – [jgerry@us.ibm.com](mailto:jgerry@us.ibm.com)

Harry Hulen – [hulen@us.ibm.com](mailto:hulen@us.ibm.com)

Patrick Schaefer – [pschaef@us.ibm.com](mailto:pschaef@us.ibm.com)

Bob Coyne – [coyne@us.ibm.com](mailto:coyne@us.ibm.com)

# Disclaimer



- 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.