A Look at PVFS, a Parallel File System for Linux

Amy Apon

Yuheng Du

Talk originally given by

Will Arensman and Anila Pillai

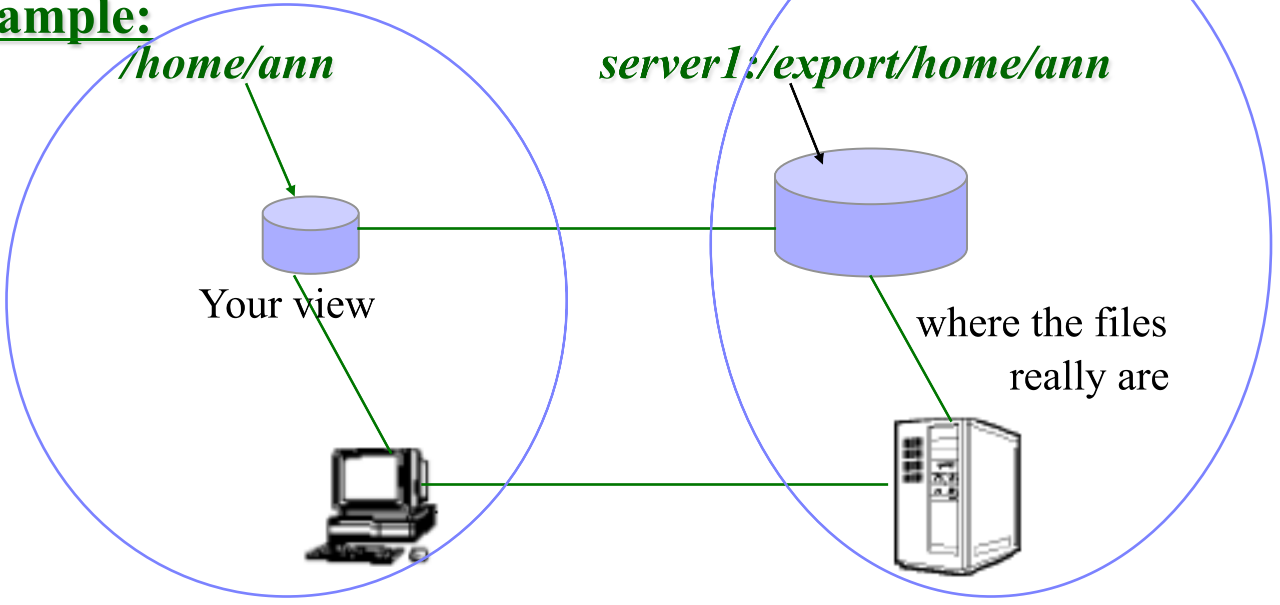
Overview

- Network File Systems (NFS)
- Drawbacks of NFS
- Parallel Virtual File Systems (PVFS)
- Using PVFS
- Conclusion
- References

1. Network File System (NFS)

- NFS is a client/server application developed by Sun Microsystems
- It lets a user view, store and update files on a remote computer as though the files were on the user's local machine.
- The basic function of the NFS server is to allow its file systems to be accessed by any computer on an IP network.
- NFS clients access the server files by mounting the servers exported file systems.

For example:



2. Drawbacks of NFS

- **Having all your data stored in a central location presents a number of problems:**
  - **Scalability: arises when the number of computing nodes exceeds the performance capacity of the machine exporting the file system; could add more memory, processing power and network interfaces at the NFS server, but you will soon run out of CPU, memory and PCI slots; the higher the node count, the less bandwidth (file I/O) individual node processes end up with**
  - **Availability: if NFS server goes down all the processing nodes have to wait until the server comes back into life.**
- **Solution: Parallel Virtual File System (PVFS)**

3. Parallel Virtual File System(PVFS)

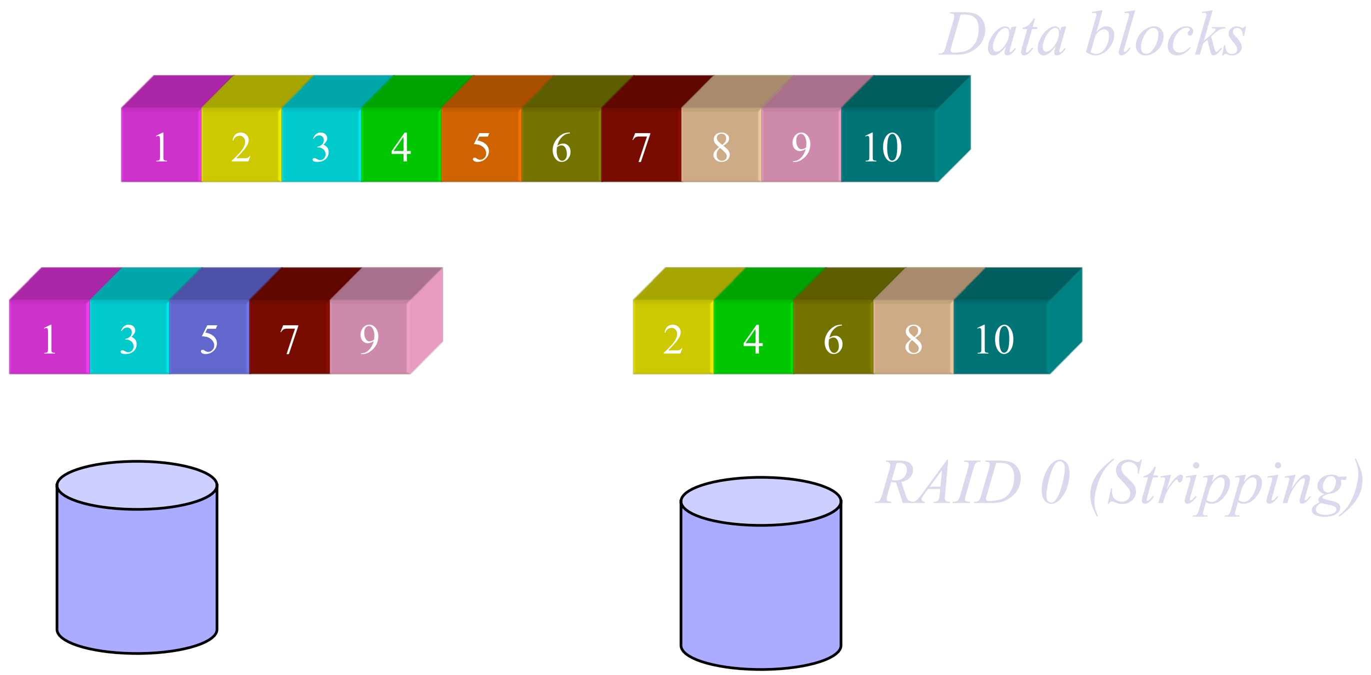
- Parallel Virtual File System (PVFS) is an open source implementation of a parallel file system developed specifically for Beowulf class parallel computers and Linux operating system
- PVFS is now as distributed as OrangeFS through Omnibond, a startup from Clemson
- PVFS has been released and supported under a GPL license since 1998

3. Parallel Virtual File System(PVFS)

- File System – allows users to store and retrieve data using common file access methods (open, close, read, write)
- Parallel – stores data on multiple independent machines with separate network connections
- Virtual – exists as a set of user-space daemons storing data on local file systems

PVFS…

- Instead of having one server exporting a file via NFS, you have N servers exporting portions of a file to parallel application tasks running on multiple processing nodes over an existing network
- The aggregate bandwidth exceeds that of a single machine exporting the same file to all processing nodes.
- This works much the same way as RAID 0 – file data is striped across all I/O nodes.



PVFS…

- PVFS provides the following features in one package:
  - allows existing binaries to operate on PVFS files without the need for recompiling
  - enables user-controlled striping of data across disks on the I/O nodes
  - robust and scalable
  - provides high bandwidth for concurrent read/write operations from multiple processes to a common file

PVFS…

- PVFS provides the following features in one package:
  - ease of installation
  - easily used - provides a cluster wide consistent name space,
  - PVFS file systems may be mounted on all nodes in the same directory simultaneously, allowing all nodes to see and access all files on the PVFS file system through the same directory scheme.
  - Once mounted PVFS files and directories can be operated on with all the familiar tools, such as ls, cp, and rm.

PVFS Design and Implementation

- Roles of nodes in PVFS:

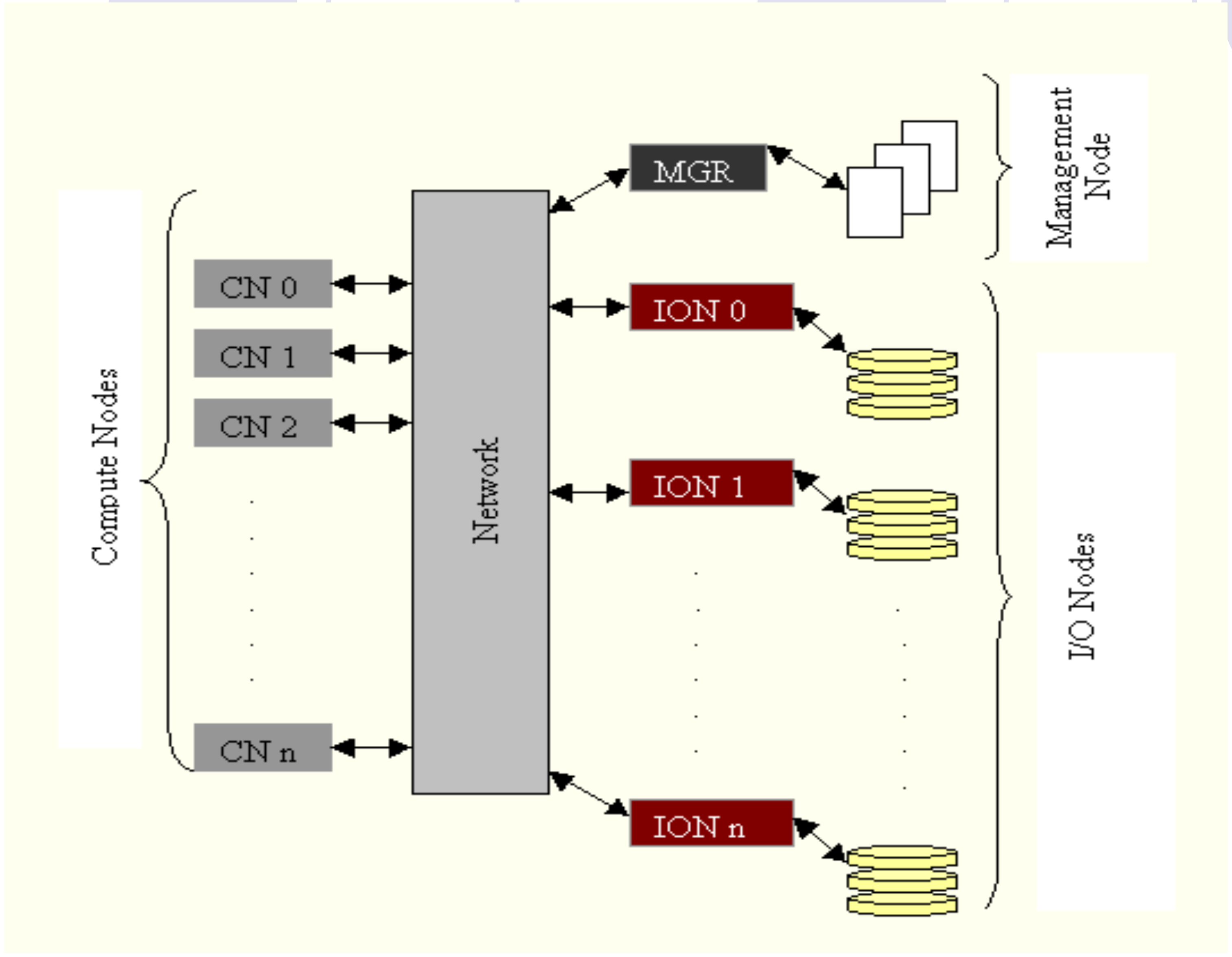
1. COMPUTE NODES - on which applications are run,
2. MANAGEMENT NODE - which handles metadata operations
3. I/O NODES - which store file data for PVFS file systems.

Note:- nodes may perform more than one role

PVFS I/O Nodes

- In order to provide high-performance access to data stored on the file system by many clients, PVFS spreads data out across multiple cluster nodes I/O nodes
- By spreading data across multiple I/O nodes, applications have multiple paths to data through the network and multiple disks on which data is stored.
- This eliminates single bottlenecks in the I/O path and thus increases the total potential bandwidth for multiple clients, or aggregate bandwidth.

PVFS System Architecture



PVFS Software Components

- There are four major components to the PVFS software system:
  1. Metadata server (mgr)
  2. I/O server (iod)
  3. PVFS native API (we will use MPIIO instead)
  4. PVFS Linux kernel support
- The first two components are daemons (server types) which run on nodes in the cluster

PVFS Components

1. The metadata server (or mgr)

- File manager; it manages metadata for PVFS files.
- A single manager daemon is responsible for the storage of and access to all the metadata in the PVFS file system
- Metadata - information describing the characteristics of a file, such as permissions, the owner and group, and, more important, the physical distribution of the file data

PVFS Version 1 I/O Node Configuration

- Data in a file is striped across a set of I/O nodes in order to facilitate parallel access.
- The specifics of a given file distribution are described with three metadata parameters:
  - base I/O node number
  - number of I/O nodes
  - stripe size
- These parameters, together with an ordering of the I/O nodes for the file system, allow the file distribution to be completely specified

PVFS Components

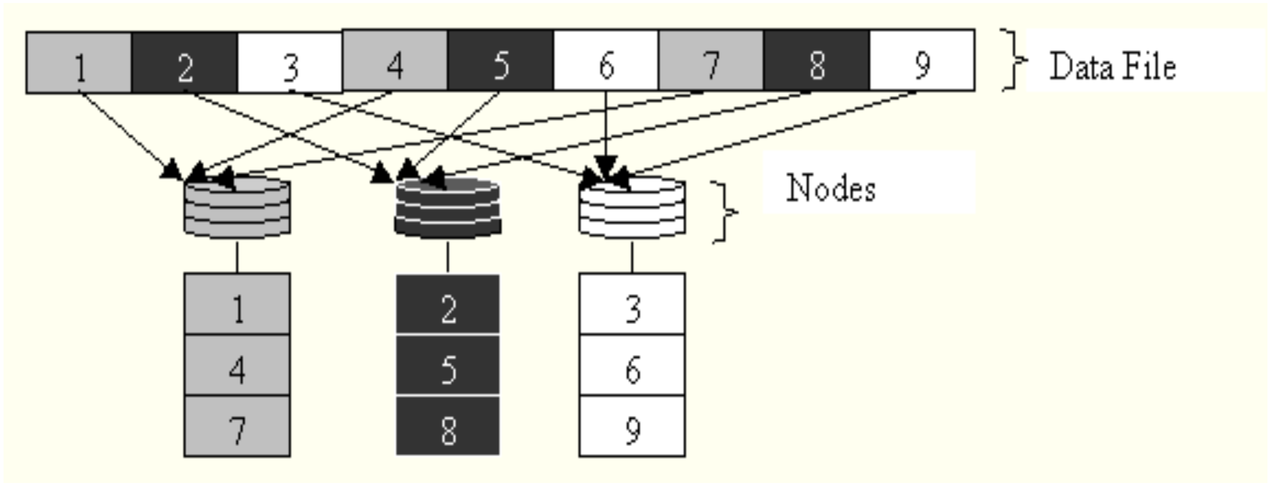
- Example:
  - pcount - field specifies that the the number of I/O nodes used for storing data
  - base - specifies that the first (or base) I/O node (is node 2 here)
  - ssize - specifies that the stripe size--the unit by which the file is divided among the I/O nodes—here it is 64 Kbytes
  - The user can set these parameters when the file is created, or PVFS will use a default set of values

inode	*1092157504*
base	*2*
pcount	*3*
ssize	*65536*

Meta data example for file

PVFS Components

PVFS file striping done in a round-robin fashion



- The file is striped across three I/O nodes.
- Each I/O daemon stores its portion of the PVFS file in a file on the local file system on the I/O node.
- The name of this file is based on the inode number that the manager assigned to the PVFS file (in our example, 1092157504).

PVFS Components

2.0 The I/O server (or iod)

It handles storing and retrieving file data stored on local disks connected to the node.

- when application processes (clients) open a PVFS file, the PVFS manager informs them of the locations of the I/O daemons
- the clients then establish connections with the I/O daemons directly

PVFS Components

- the daemons determine when a client wishes to access file data, the client library sends a descriptor of the file region being accessed to the I/O daemons holding data in the region
- what portions of the requested region they have locally and perform the necessary I/O and data transfers.

PVFS Components

3. PVFS native API (libpvfs)

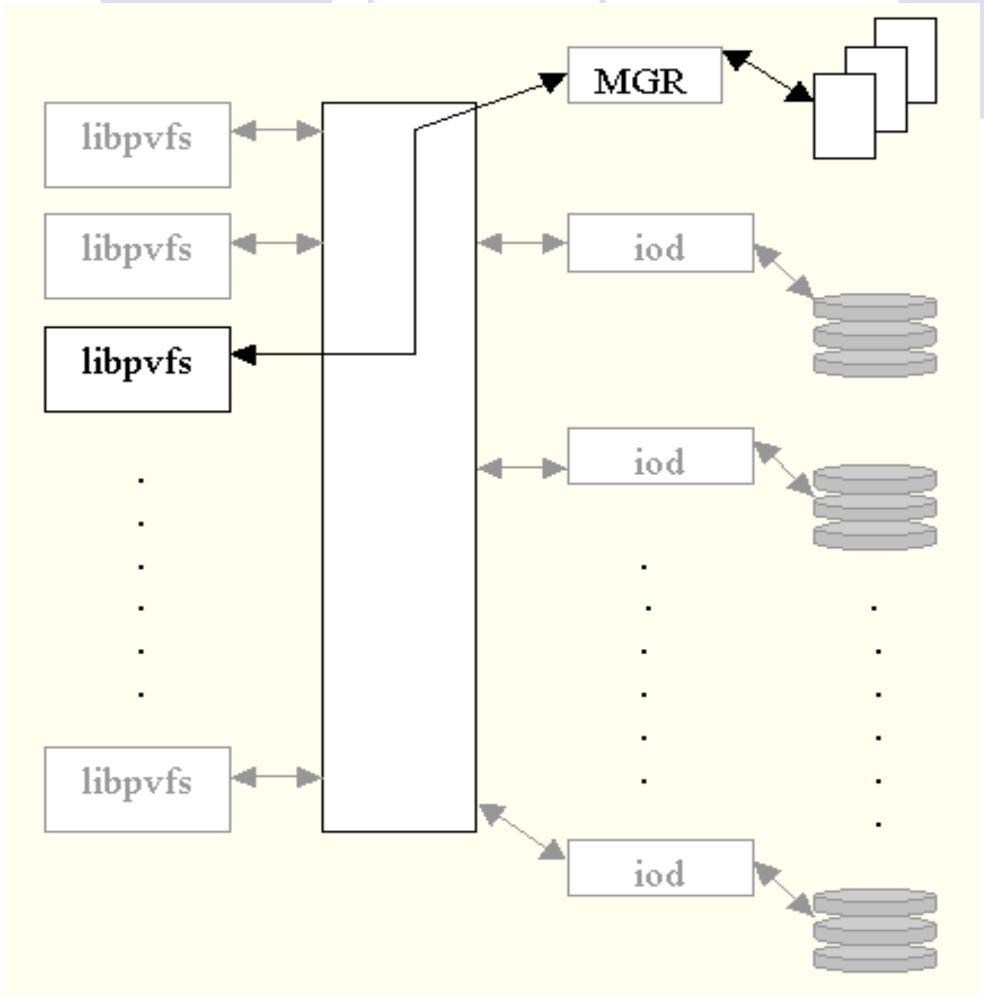
- It provides user-space access to the PVFS servers
- This library handles the scatter/gather operations necessary to move data between user buffers and PVFS servers, keeping these operations transparent to the user
- For metadata operations, applications communicate through the library with the metadata server

PVFS Components

3. <u>PVFS native API (libpvfs)</u>

- For data access the metadata server is eliminated from the access path and instead I/O servers are contacted directly
- This is key to providing scalable aggregate performance
- The figure shows data flow in the PVFS system for metadata operations and data access

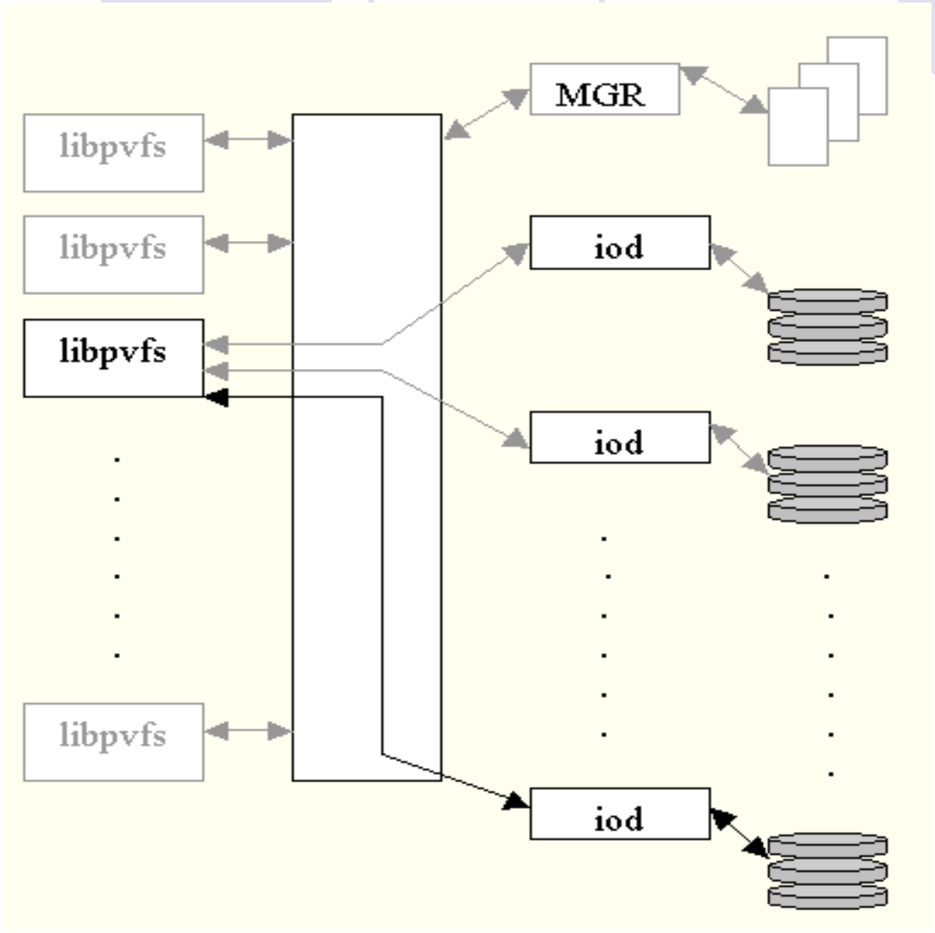
PVFS Components



Metadata access

For metadata operations applications communicate through the library with the metadata server

PVFS Components



Data access

Metadata server is eliminated from the access path; instead I/O servers are contacted directly; libpvfs reconstructs file data from pieces received from iods

PVFS Components

4. PVFS Linux kernel support

- The PVFS Linux kernel support provides the functionality necessary to mount PVFS file systems on Linux nodes
- This allows existing programs to access PVFS files without any modification
- This support is not necessary for PVFS use by applications, but it provides an extremely convenient means for interacting with the system

PVFS Components

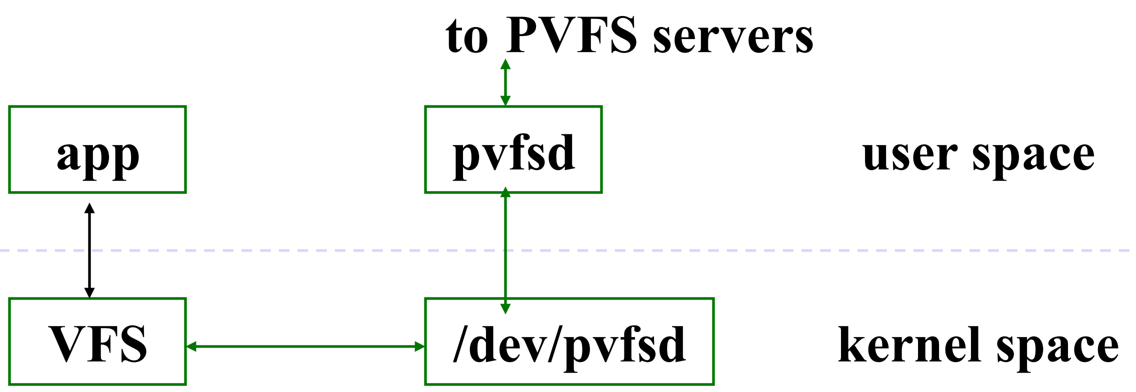
4. PVFS Linux kernel support

- The PVFS Linux kernel support includes:
  - a loadable module
  - an optional kernel patch to eliminate a memory copy
  - a daemon (pvfsd) that accesses the PVFS file system on

    behalf of applications
- It uses functions from libpvfs to perform these operations.

PVFS Components

Data flow through kernel



- The figure shows data flow through the kernel when the Linux kernel support is used
- Operations are passed through system calls to the Linux VFS layer. Here they are queued for service by the pvfsd, which receives operations from the kernel through a device file
- It then communicates with the PVFS servers and returns data through the kernel to the application

PVFS Application Interfaces

- **ROMIO MPI-IO interface:-**

  ROMIO implements the MPI2 I/O calls in a portable library. This allows parallel programmers using MPI to access PVFS files through the MPI-IO interface

7. Conclusions

- Pros:
  - Higher cluster performance than NFS.
  - Many hard drives to act a one large hard drive.
  - Works with current software.
  - Best when reading/writing large amounts of data
- Cons:
  - Multiple points of failure.
  - Not as good for “interactive” work.

8. References

1. *The Parallel Virtual File System, Available at: http:// www.parl.clemson.edu/pvfs/*
2. *P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, ``PVFS: A Parallel File System For Linux Clusters'', Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, October 2000, pp. 317-327*
3. *Thomas Sterling, "Beowulf Cluster Computing with Linux", The MIT Press, 2002*
4. *W. B. Ligon III and R. B. Ross, ``An Overview of the Parallel Virtual File System''*, **Proceedings of the 1999 Extreme Linux Workshop,** *June, 1999.*
5. *Network File System, Available at: http://www.redhat.com/ docs/manuals/linux/RHL-7.2-Manual/ref-guide/ch-nfs.html*
6. *http://www.linuxuser.co.uk/articles/issue14/lu14-All_you_need_to_know_about-PVFS.pdf*