Parallel Virtual File System Version 2

Clemson University and
Argonne National Laboratory
May 2002

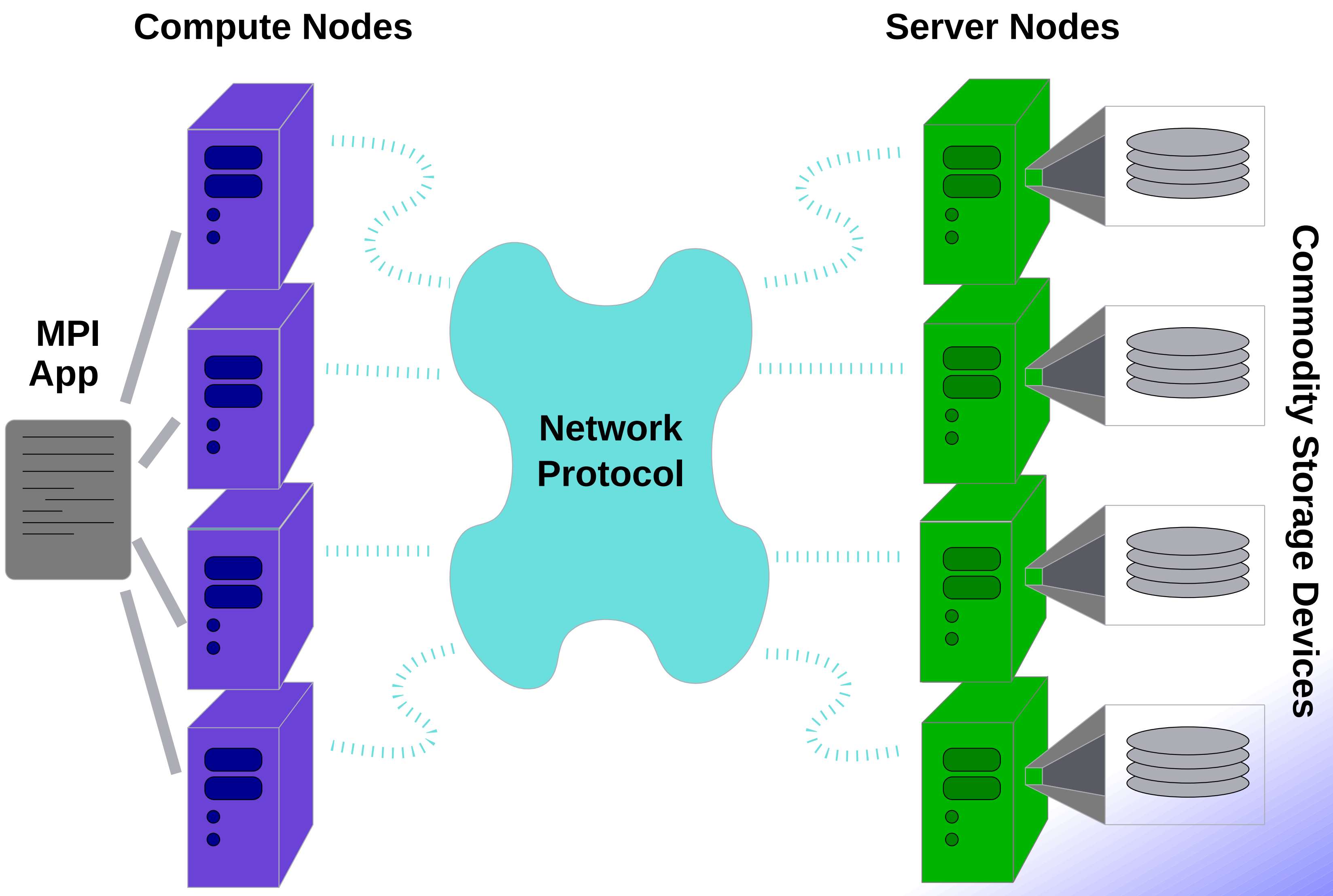
Contact Information

- PVFS Web Site:
  - http://www.parl.clemson.edu/pvfs
- PVFS Mailing Lists
  - http://www.parl.clemson.edu/pvfs/pvfs-lists.html
- Project contacts
  - Walt Ligon walt@parl.clemson.edu
  - Rob Ross rross@mcs.anl.gov
  - Phil Carns pcarns@parl.clemson.edu

PVFS V2 Architecture

- Set of user level servers provide file system services
  - Each server can handle meta data or file I/O requests
- Client access to the servers through:
  - User level libraries
  - Kernel level driver
- Pluggable modules for many features
  - I/O transports
  - Scheduling
  - Data Distributions

PVFS System Overview



Buffered Message Interface

- Low level networking abstraction
- Allows the use of multiple protocols for PVFS v2 communication
- Simple, streamlined API
- Nonblocking
- Operates on message buffers rather than streams
- Allows optimizations for user level networking
- Currently supports TCP/IP and Myricom's GM protocol

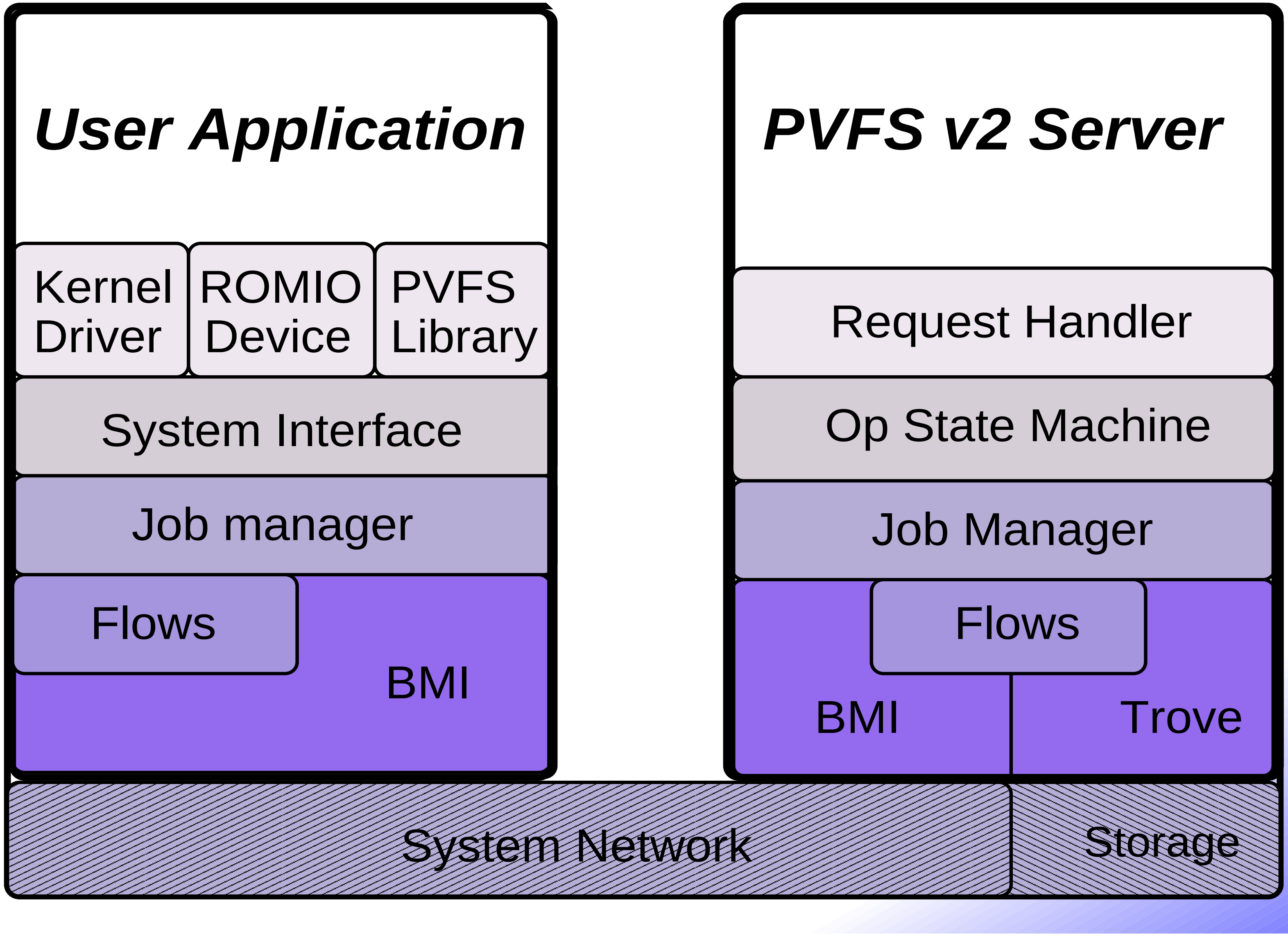
Trove

- Low level storage abstraction
- Allows the use of multiple storage devices and API's with PVFS v2
- Supports noncontiguous byte stream access
- Supports simple database operations
- Includes hooks for consistency semantics
- Nonblocking
- Prototype runs on top of Berkeley DB and Unix files

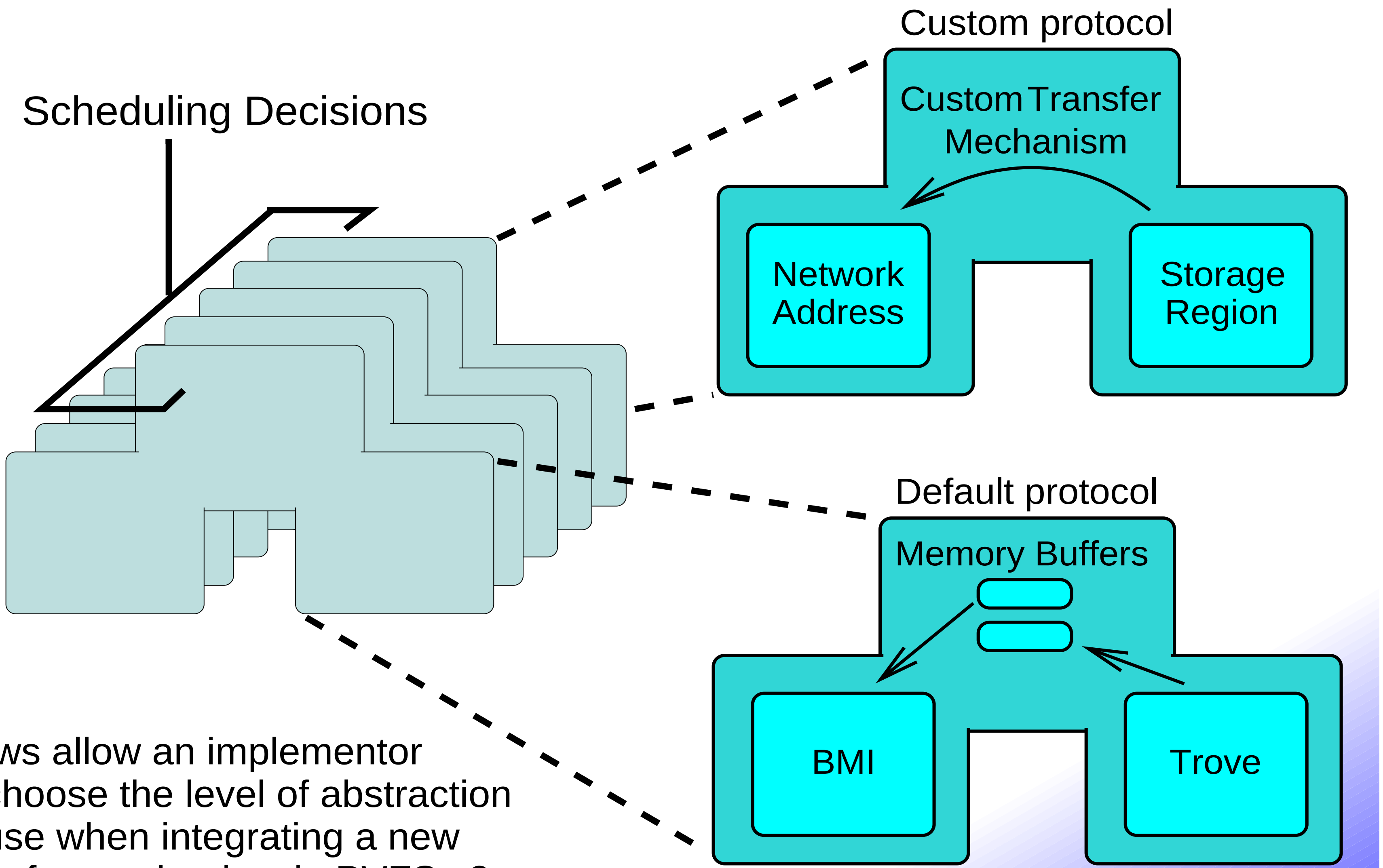
Flows

- High level I/O abstraction
- Moves data between a source and destination
  - Network, Storage device, or Memory region
- Scheduling decisions based on combined knowledge of network and storage behavior
- Initial implementation uses Trove and BMI for I/O, but these could be bypassed with optimized ``flow protocols''
- Handles converting I/O request descriptions and and file distribution information into simple low level operations

Interface Overview



Flow Diagram



Flows allow an implementor
to choose the level of abstraction
to use when integrating a new
transfer mechanism in PVFS v2

Data Distributions

- Pluggable distribution mechanisms
- Support more than just RAID 0 style striping
- More complex patterns, such as block-cyclic or Hilbert curves
- Useful for applications that could benefit from alternative data locality properties
- Express distributions using a set of generic functions and parameters; hide the mathematics from the I/O mechanism

Redundancy

- Plan for the future
- Frequently requested user feature
- Could support several approaches:
  - ``Lazy'' redundancy (computing parity on file close)
  - Mirroring of I/O data
  - Advanced schemes that make use of Trove consistency semantics
- Some of these approaches are being evaluated in PVFS v1 right now

I/O Request Descriptions

- I/O request should be able to express more than simple strided and contiguous operations
- Flexible and compact representation
- Convenient mapping to MPI-IO style datatypes (for more efficient integration with ROMIO)
- Similar to existing approaches in MPI
- Supports incremental ``unrolling'' of descriptions as needed by the I/O subsystem

Client Interfaces

- Native user level PVFS library
  - pvfs_open(), pvfs_read(), etc.
- Linux kernel integration
  - For administration and compatibility
- MPI-IO
  - Through ROMIO MPI-IO implementation
- All built on top of a unified ``system interface'' that provides the common primitives needed by all three interfaces
- System interface will provide a clean entry point for future additions (such as grid interfaces, etc.)

Distributed Metadata

- PVFS v2 will support the use of multiple metadata servers
- Fast client side hashing to determine the metadata server for a given file
- Avoids recursive path lookups
- Improved latency for interactive and metadata intensive applications
- Same server code supports both metadata and I/O operations for easier configuration and installation