The Parallel Virtual File System: Overview and Usage

Phil Carns
pcarns@parl.clemson.edu
Parallel Architecture Research Laboratory (PARL)
Clemson University
http://www.parl.clemson.edu/

Outline of Presentation

- Overview of PVFS
- PVFS Architecture
- Application Access Options
- Real world use and examples (both good and bad)
- Current Research Topics
- Brief Analysis of Performance on Baby Blue

Why PVFS?

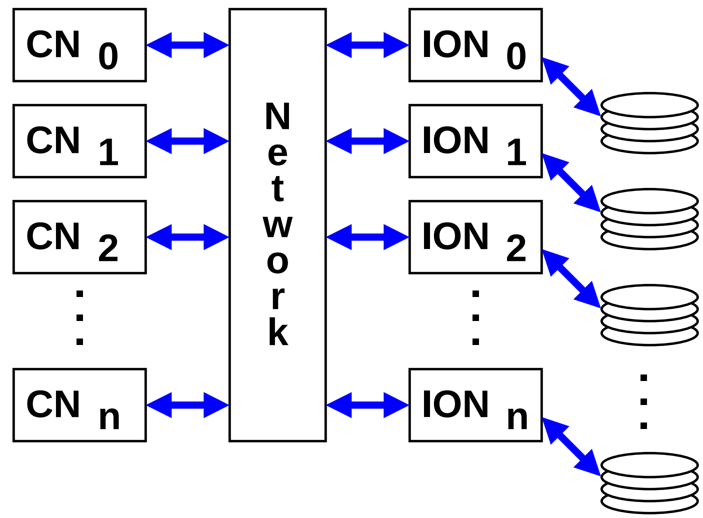
- Motivation:
  - As large scale scientific computational software grows, it is difficult for disk performance to keep up
  - Especially true for codes that spend a large fraction of time in I/O (a good example is satellite data processing)
- Obvious alternatives:
  - local disks on compute nodes: inconvenient for most apps
  - NFS: poor scalability and lack of parallel application features
  - Storage Area Networks: Requires custom hardware, may or may not scale

PVFS Approach

- Utilize N seperate I/O servers rather than one central server
  - Avoid single disk or disk array bottleneck
  - Attempt to distribute I/O load as evenly as possible
- Leverage commodity disks
- Commodity networking
- Provide convenient API's for parallel codes
- We are emphasizing *aggregate I/O performance*

System Architecture

- Allow many clients to access shared storage
- Each server maintains file data on its own local disk
- Clients communicate directly with I/O servers for data requests; no indirection through a centralized server



High level model

- Maintain metadata consistency with central manager for permissions, timestamps, etc.

Client Access

- Here is my application - where is the file system?

- Several possible interfaces:

  – Native PVFS library access

  – Kernel mode client access

  – MPI-IO library

- When is each appropriate?

Native PVFS library

- Part of the semi-portable "core file system code"
- Specific to PVFS; provides pvfs_open(), pvfs_read(), etc.
- Allows client tuning of file system parameters, such as stripe size and number of servers to use
- Very low overhead
- Includes a few advanced parallel file system features...
- No free ride - requires an application custom written for PVFS

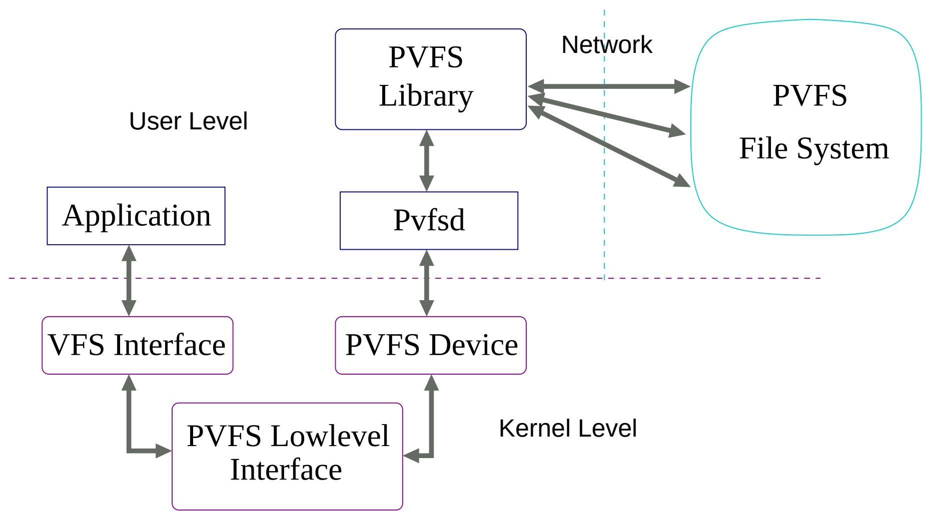
PVFS kernel module

- Allows users to mount PVFS file systems and use standard Unix I/O calls

- Recommended for file system maintenance and legacy applications

- Only available for Linux x86

- Serious performance penalty (ranging from 10% to 50%)



MPI-IO

- Portion of MPI 2.0 specification providing advanced I/O interface, including:
  - Derived datatypes (noncontiguous access for file and memory)
  - Collective I/O (coordinated aggregate operations)
  - Application hints (application level tuning parameters)
  - Consistency semantics
- PVFS is fully supported in ROMIO MPI-IO implementation from Argonne National Laboratory

MPI-IO benefits

- Included by default with MPICH, but may be used with other MPI implementations

- Portable across different file systems and architectures

- Uses native PVFS library for performance

- Provides many optimizations

What PVFS does not provide

- Data redundancy and fault tolerance
  - I/O server crashes -> file system does not recover
  - Raid may be used on each file server to protect against disk failure, but not against overall machine failure
- Caching and prefetching
  - Caching only done at individual server level
  - No client side caching

What PVFS does not provide

- Locking
  - No flock(), fcntl(), or POSIX style locking
  - No MPI-IO atomic mode
- Symbolic links
- Small operation latency

Good examples of PVFS use

- Parallel applications that can utilize parallel bandwidth
- Run time storage for computation data: “scratch space”
- Staging application data to nodes (even if jobs are not parallel)

Bad examples of PVFS use

- Long term archival
  - Remember redundancy?
- User home directories
  - No optimizations for this workload
  - Poor metadata latency in kernel module
- Non parallel applications with frequent small requests
  - Such as typical web server load (unless you intend to stream multimedia)

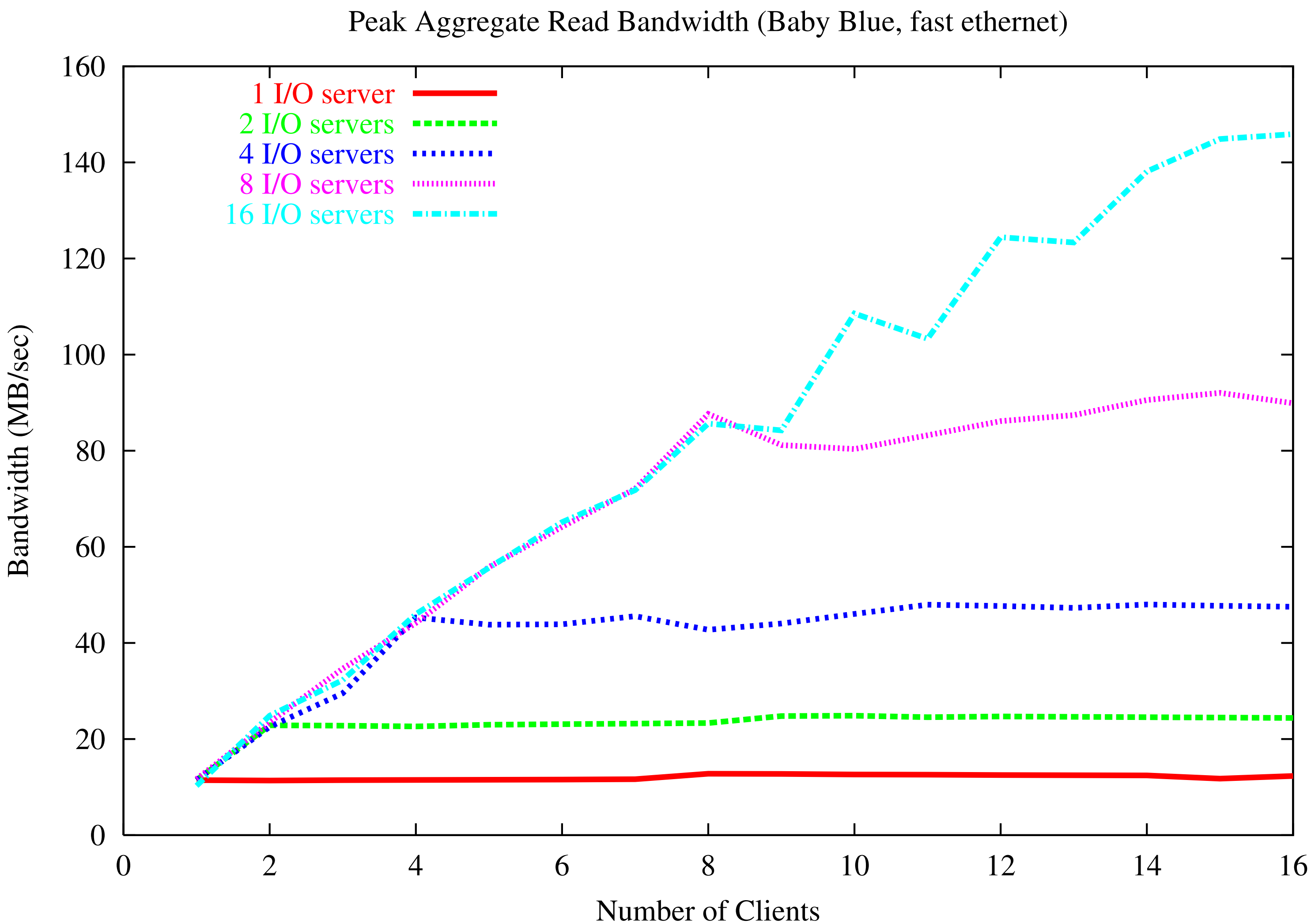
Where are we going?

- PVFS 2 design and implementation is underway
- More flexibility and use of modern technology
- Long term project
- Full file system rewrite

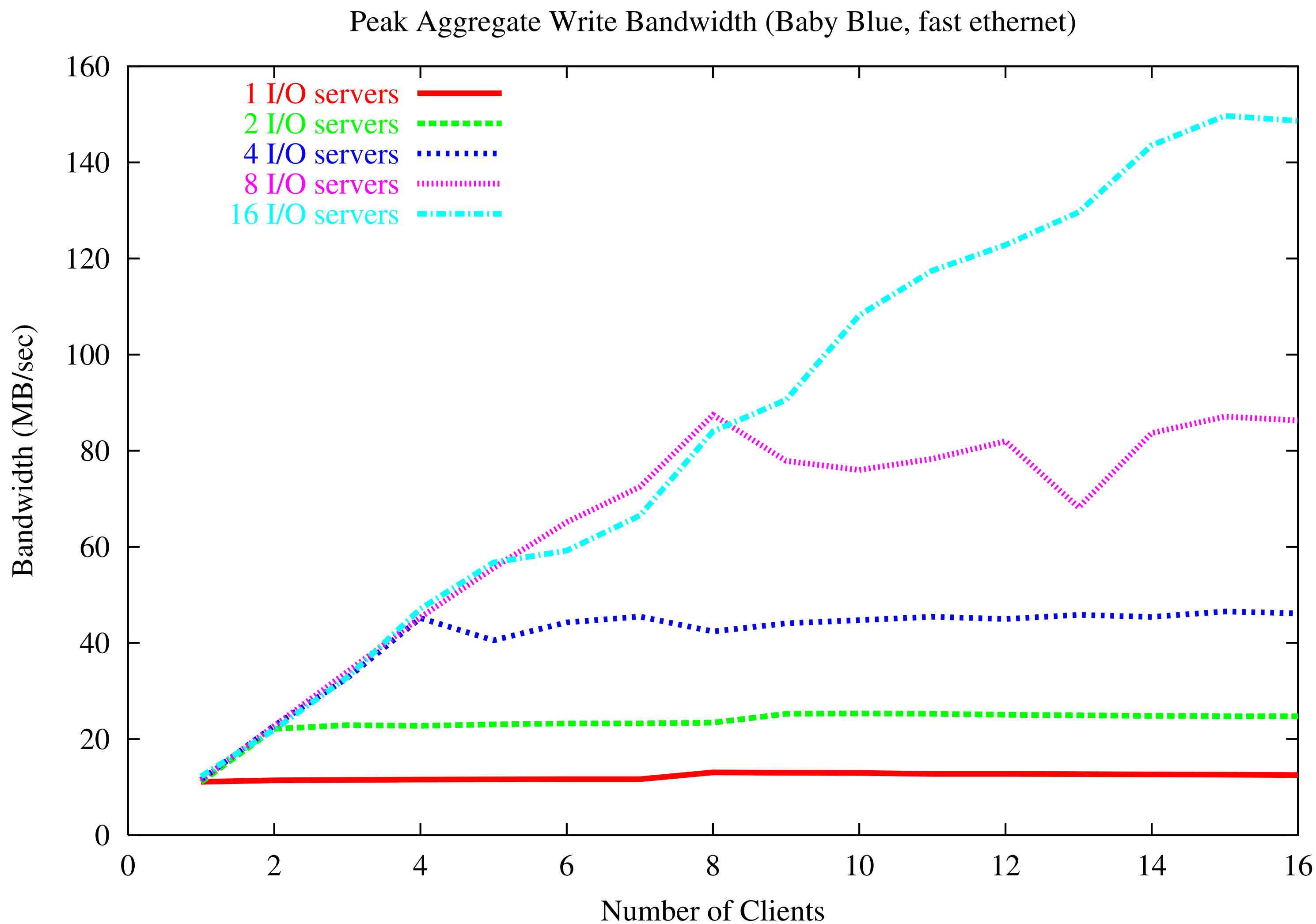
PVFS 2 highlights

- Modular use of alternative network protocols
- Modular use of alternative storage mechanisms
- Advanced data distributions (beyond striping)
- Better scheduling hooks
- Multiple metadata servers
- More expressive interface for better MPI-IO support
- Extended metadata attributes

Performance



Performance



Performance