PVFS: Past, Present, Future

CLEMSON UNIVERSITY

What Is a Parallel File System?

- Distributes file data across multiple nodes on a parallel computer
  - RAID distributes across disks
  - Distributed FS distributes files across servers
- Support concurrent access by tasks in a parallel application

What is PVFS?

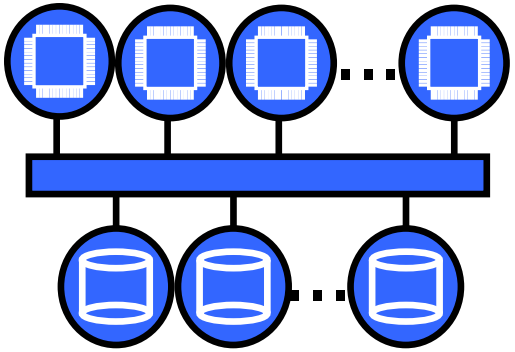
- Parallel Virtual File System
  - v0 1994
    - based on Vespa and PVM
  - v1 1996
    - strided non-contiguous access patterns
    - collective access reordering
  - v2 2002
    - code base rewritten for efficiency, portability, extensibility
    - MPI based non-contiguous access patterns
    - production ready features

PVFS Design Goals

- Scalable
  - Configurable file striping
  - Non-contiguous I/O patterns
  - Eliminates bottlenecks in I/O path
  - Does not need locks for metadata ops
  - Does not need locks for non-conflicting applications
- Usability
  - Very easy to install, small VFS kernel driver
  - Modular design for disk, network, etc
  - Easy to extend

I/O in HPC Systems

- HPC applications increasingly rely on I/O subsystems
  - Large input datasets, checkpointing, visualization
- Programmers desire interfaces that match their problem domain
  - Multidimensional arrays, typed data, portable formats
- Two issues to be resolved by I/O system
  - Performance requirements (concurrent access to HW)
  - Gap between app. abstractions and HW abstractions
- Software required to address both of these problems

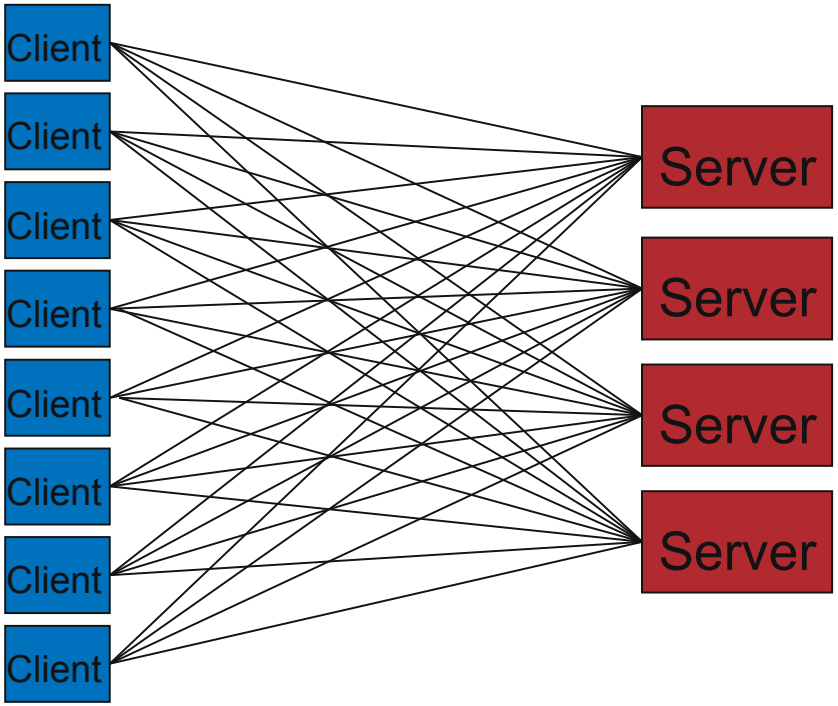


Clients running applications
(100s-10,000s)

Storage or System Network

I/O devices or servers
(10s-1000s)

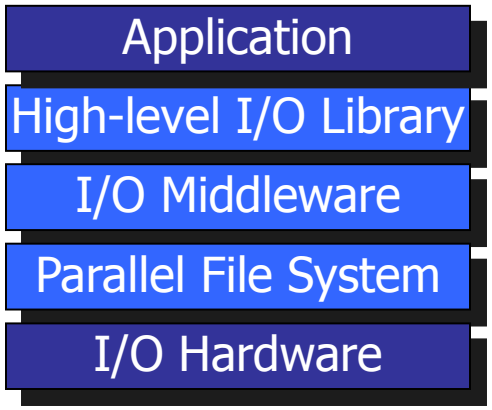
PVFS Architecture



- Components
  - clients library links to applications
  - network drivers (TCP/IP, Myrinet, Infiniband, etc.)
  - servers processes manage I/O devices

I/O Software Stack

- Computational science applications have complex I/O needs
  - Performance and scalability requirements
  - Usability (Interfaces!)
- Software layers combine to provide functionality
  - High-level I/O libraries provide useful interfaces
    - Examples: Parallel netCDF, HDF5
  - Middleware optimizes and matches to file system
    - Example: MPI-IO
  - Parallel file system organizes hardware and actually moves data
    - Examples: PVFS, GPFS, Lustre

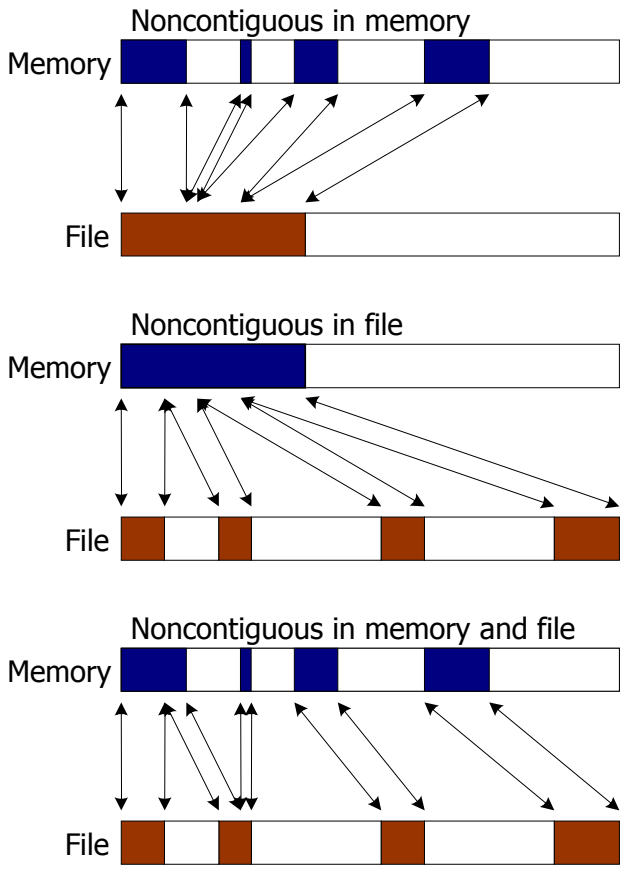


PVFS Interfaces

- PVFS client library designed for systems
  - exposes wide range of features and performance enhancing options
- ROMIO MPI-IO interface for high performance parallel programs
- Linux kernel interface for routine management (ls, rm, cp, chmod, etc.)
- Posix-like interface

Non-contiguous I/O

- Noncontiguous I/O operations are common in computational science applications
- Most PFSs available today implement a POSIX-like interface (open, write, close)
- POSIX noncontiguous support is poor:
  - readv/writev only good for noncontiguous in memory
  - POSIX listio requires matching sizes in memory and file
- Better interfaces allow for better scalability

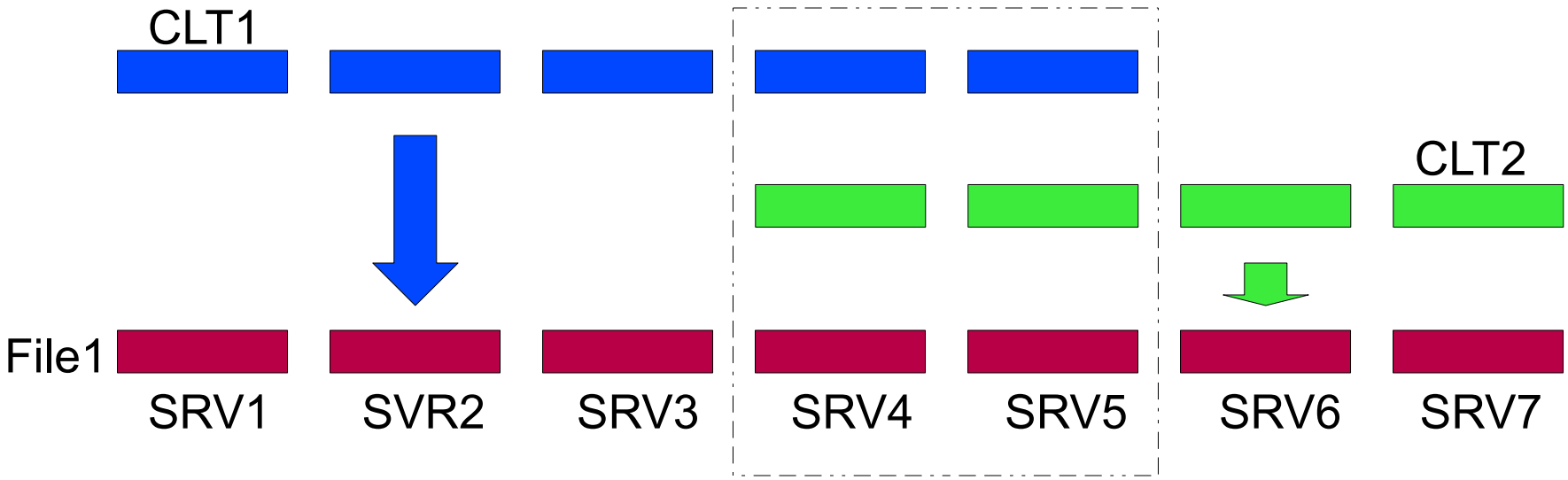


Semantics

- Sequential Consistency
  - the "gold" standard
  - specified by Posix
- BUT …
  - Expensive to implement for performance and scalability
  - Not needed if applications well behaved
- PVFS uses a weaker consistency model
  - Indistinguishable from SC for many programs
  - Provides much better performance/scalability

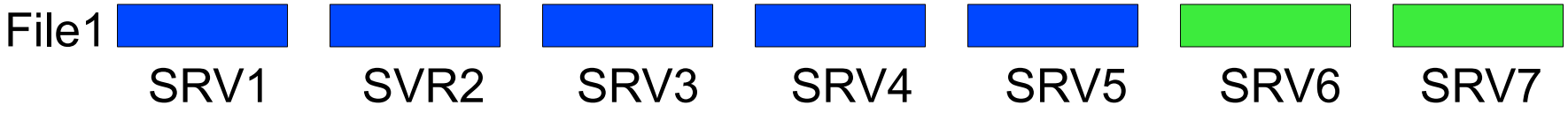
Semantics Example

Semantics Example(1)



Semantics Example(2)

Acceptable!

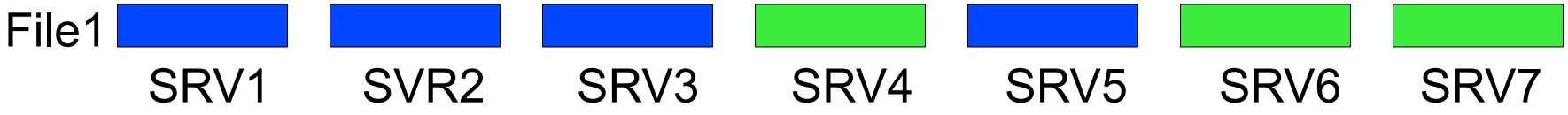


Semantics Example(3)

Acceptable!

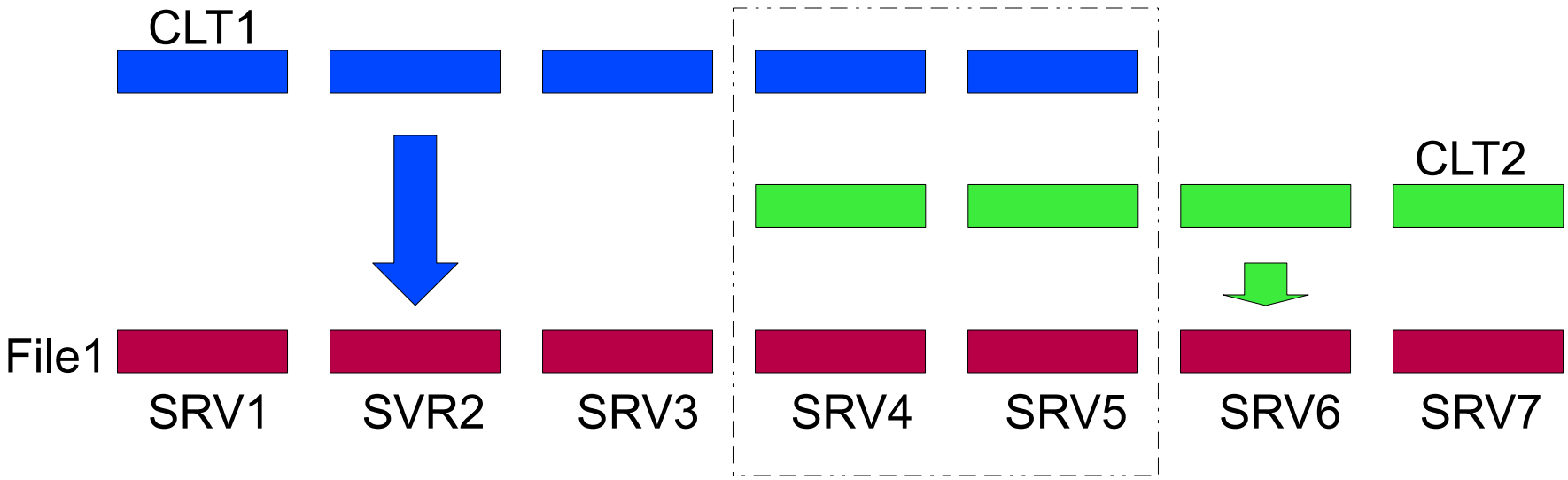
Semantics Example(4)

NOT Acceptable!!!



Semantics Example(5)

Most application NEVER do this!
Use synchronization!



PVFS Is Good For

- Large files (>1GB)
- Large accesses (>1MB)
- Large number of clients
- Large number of servers
- Not so good for
  - small files
  - lots of files (1M files in a dir)
  - small accesses
  - interactive use

NOT a replacement for NFS!

What is wrong with small?

- PVFS does not cache data on the client
  - the overhead of I/O is large when data is small
- But … we are developing client caching
  - caching on the client is hard
  - consistency issues become even more complex
  - we plan to continue our weaker consistency model to allow efficient client caching

Development and Support

- Argonne National Laboratory
  - Rob Ross, Phil Carns, Rob Latham, Sam Lang
- Clemson University
  - Academic research team
  - Professional development team
- Ohio Supercomputer Center
  - Pete Wyckoff, Troy Baer, Ananth Devulapalli
- PVFS Community
  - Northwestern, CMU, Ohio State, Oregon, Michigan, Heidelberg (Gr)
  - Ames, Sandia
  - Acxiom, Myricom

CLEMSON UNIVERSITY

Current Development

- Small file support
- Scalable metadata ops
- Security enhancements
- Redundancy
- BlueGene P

Small File Support

- Stuffed files
  - small files on one server w/metadata
- Lazy stripe allocation
  - small files only use a few stripes
- Client caching
  - small accesses cached, latency reduced
  - various consistency models available
    - read-only
    - non-overlapping write
    - weaker consistency
    - sequential consistency

Scalable Metadata Ops

- Server-to-server collective comm
  - scalable operations on large number of servers
- Distributed directories
  - very large (1M files) directories accessed in parallel
- Pre-allocated stripes
  - reduce communication at create time
- Readdir plus
  - aggregate multiple reads

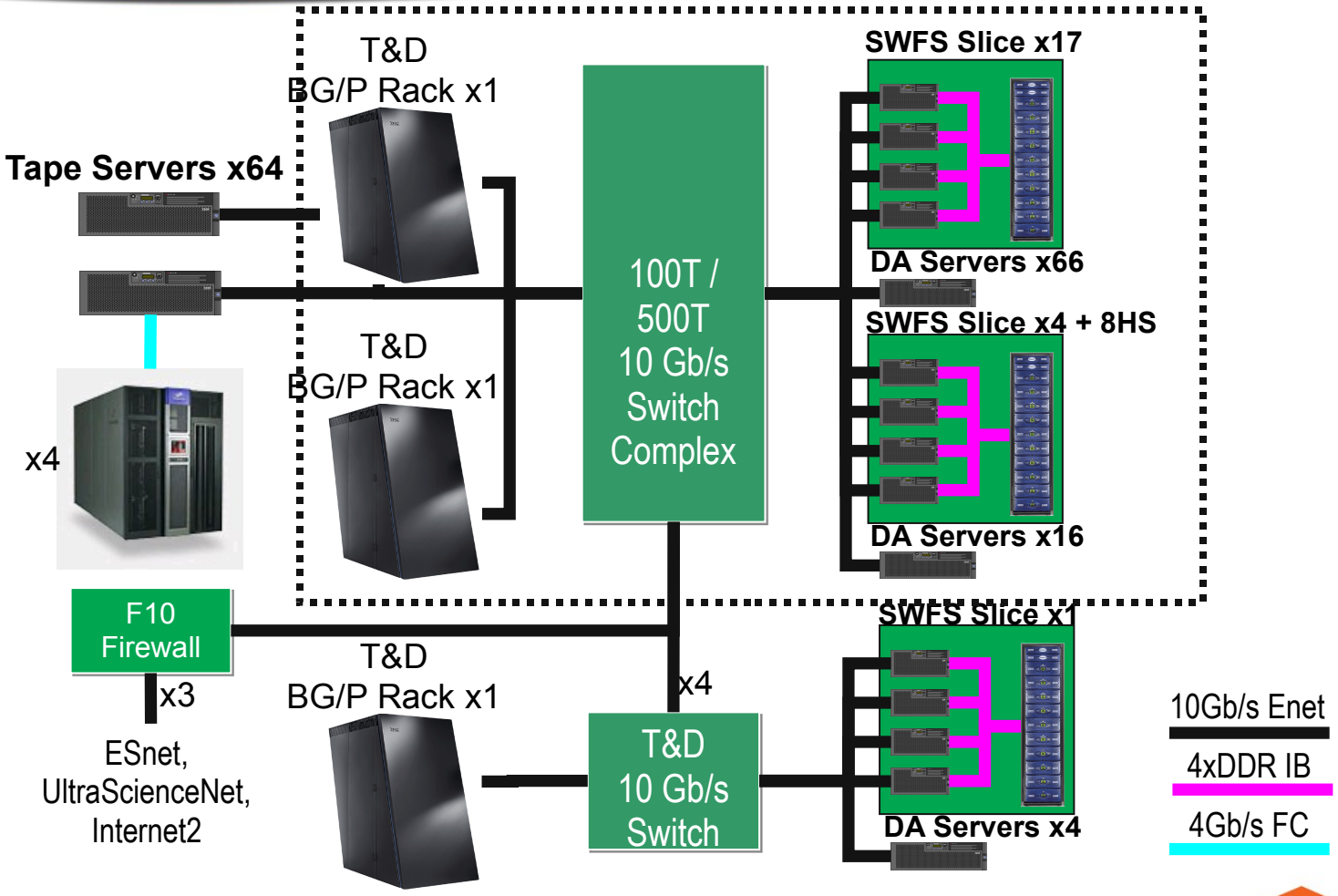
Security Enhancements

- Capability based security
  - signed structure transfers access control
  - timeouts, revoke lists, sequencing
- Certificate based authentication
  - conforms with existing authentication
- Unix and ACS style access control
  - familiar and flexible

Redundancy

- Hardware based high availability
  - most effective
  - best performance
- FS supported mirroring
  - working prototype
  - may see on checkpointing system
- FS supported parity redundancy
  - depends on demand

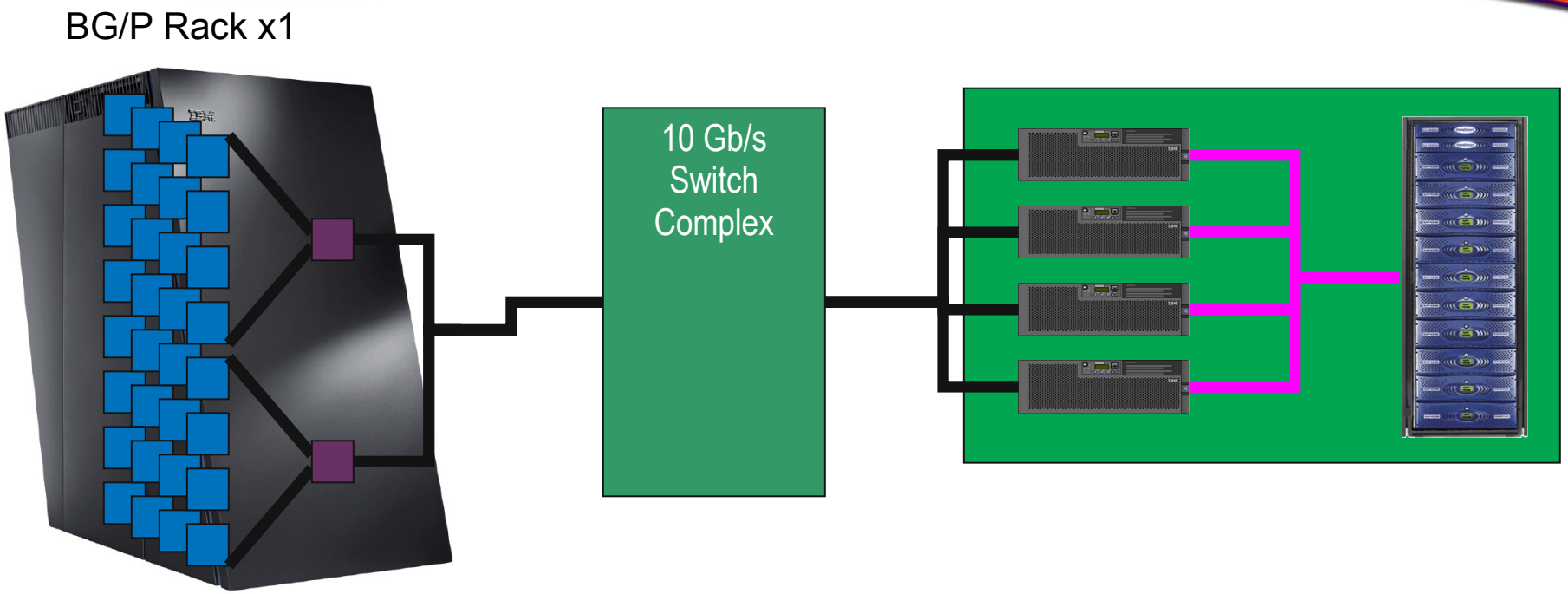
BlueGene/P Systems at ANL



CLEMSON UNIVERSITY

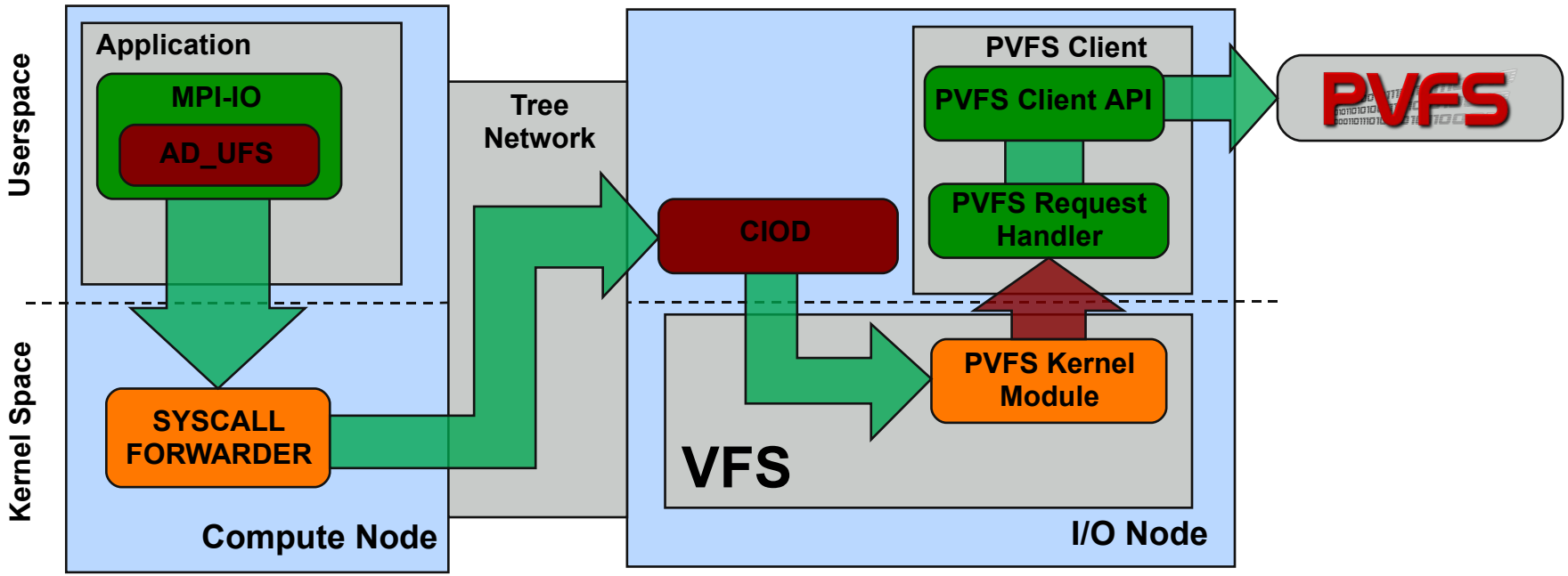
SC³ • South Carolina Computing Consortium

Compute to storage path



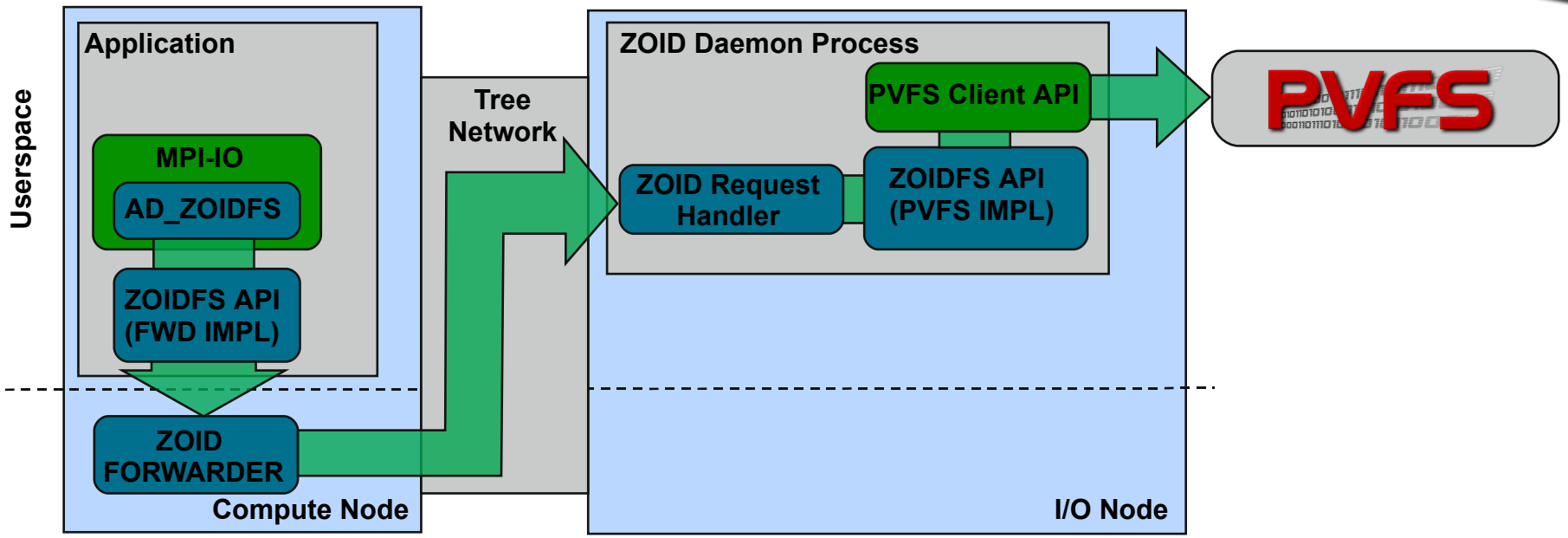
- 1024 compute nodes ( ) in a rack communicate with IO nodes ( ) via a tree network
  - Ratio of CNs to IONs is 64:1, each CN talks with one specific ION
- IONs communicate with file servers via 10GigE
- Per-CN I/O bandwidth is ~20 MB/sec (1.25 GBps / 64) if all are active

Software in Storage Path



- IBM software marshals arguments and forwards system calls to I/O node
  - "I/O forwarding" layer, CIOD is the component on the I/O node
- CIOD replays system calls on behalf of application process
- IBM compute node kernel and I/O forwarding software make convoluted path to storage
  - True for GPFS as well

Radix Solution



- Radix team is implementing replacement kernel (using Linux) and I/O forwarding (called ZOID) for BG/P
- ZOID I/O protocol is more efficient than IBM version (allows aggregation)
- ZOID implementation hooks directly to PVFS libraries, eliminating copies

Conclusion

- PVFS has a long history
- PVFS today is used at both small and large sites, industry and research
- There are issues to be aware of
- Development to overcome these is going strong
- www.pvfs.org for downloads and mailing lists