PVFS: A Parallel File System for Linux Clusters

- Is a joint project between Clemson University and the Mathematics and Computer Science Division at Argonne National Laboratory.

Why ?

- High Performance I/O.
- It can also be used as tool for pursuing further research in parallel I/O.
- PVFS is being used at a number of sites. Argonne National Laboratory, NASA, Oak Ridge National Laboratory.

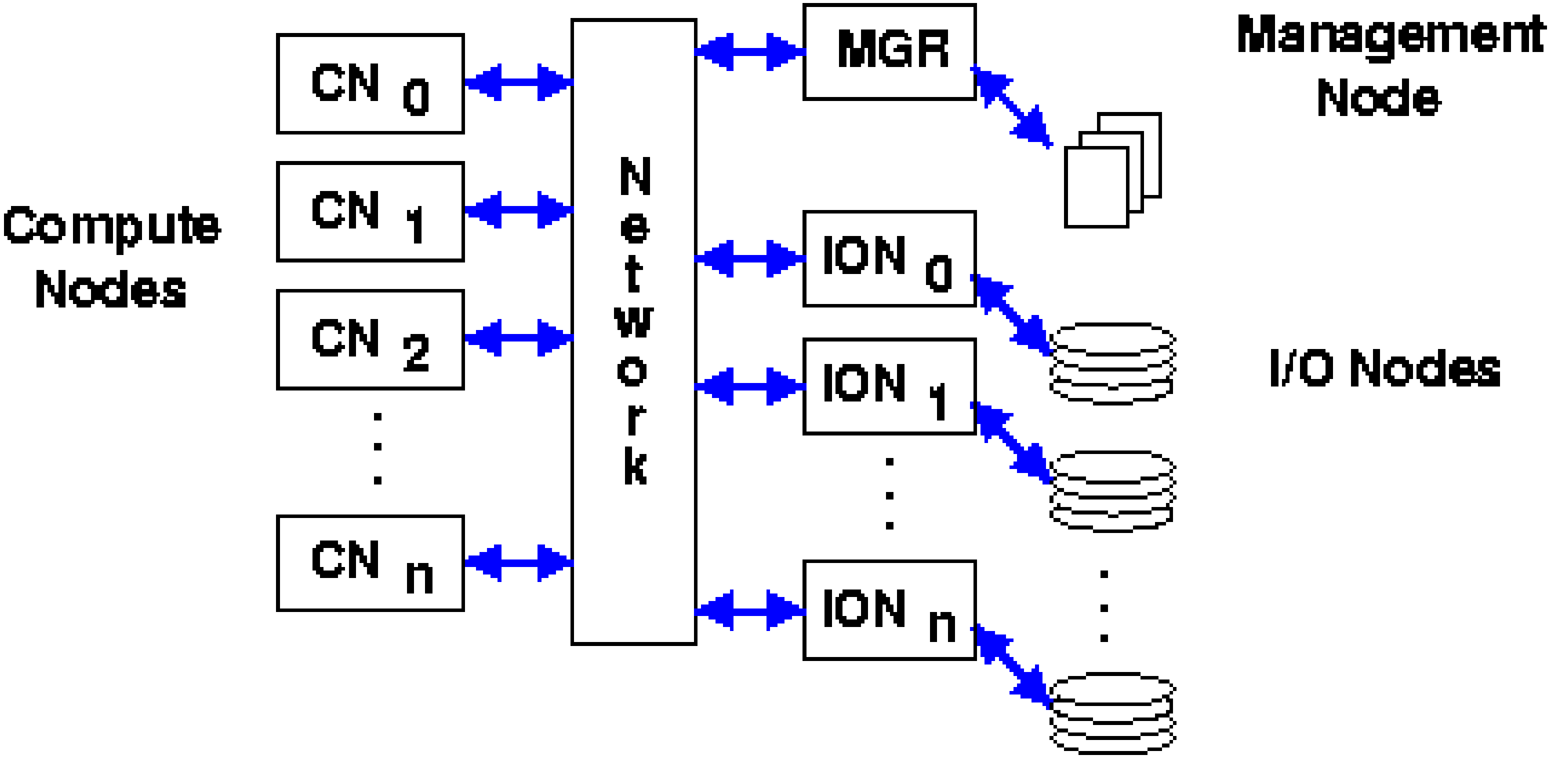
Design Goals

- High bandwidth for concurrent read/write.
- Support for multiple API's.
- Support for common UNIX shell commands.
- Access PVFS with out recompiling.
- Robust and scalable.
- Ease of use.

Related Work

- Commercial Parallel file systems.
- Distributed File Systems.
- Research Projects.

Design


Management Node
MGR
Compute Nodes
CN 0
CN 1
CN 2
CN n
Network
ION 0
ION 1
ION n
I/O Nodes

Design (Cont..)

- Intelligent Server Architecture.
- Consistent Name space.
- Use of existing binaries without recompilation.

PVFS Manager

- A single manager is responsible of the metadata.
- The distribution information includes both the file location and the location of the disk in the cluster.
- The location of the file is specified with three parameters base I/O node number, number of I/O nodes, stripe size.

Example...

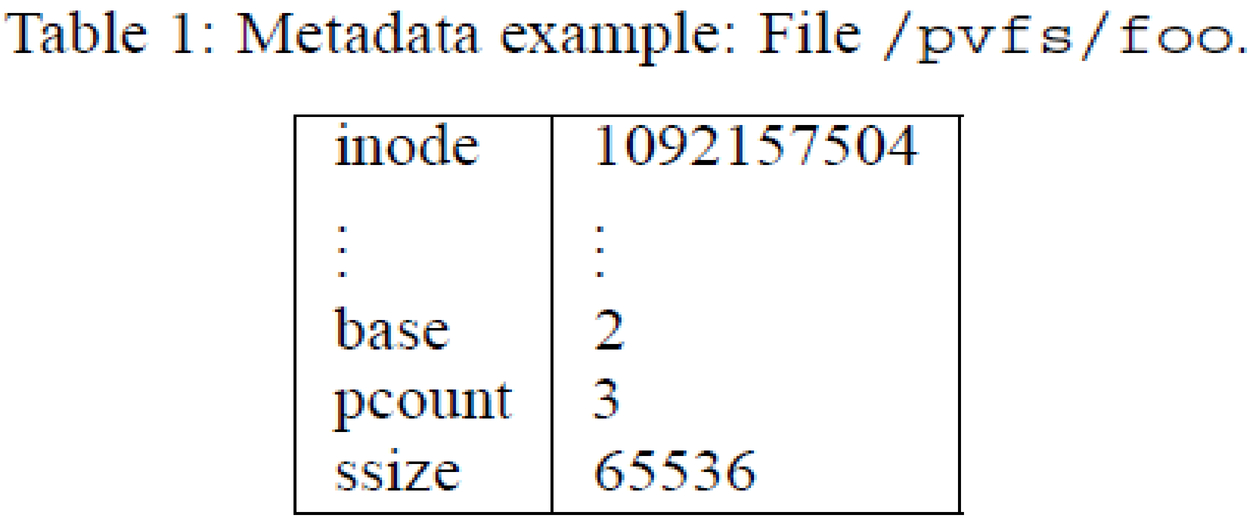
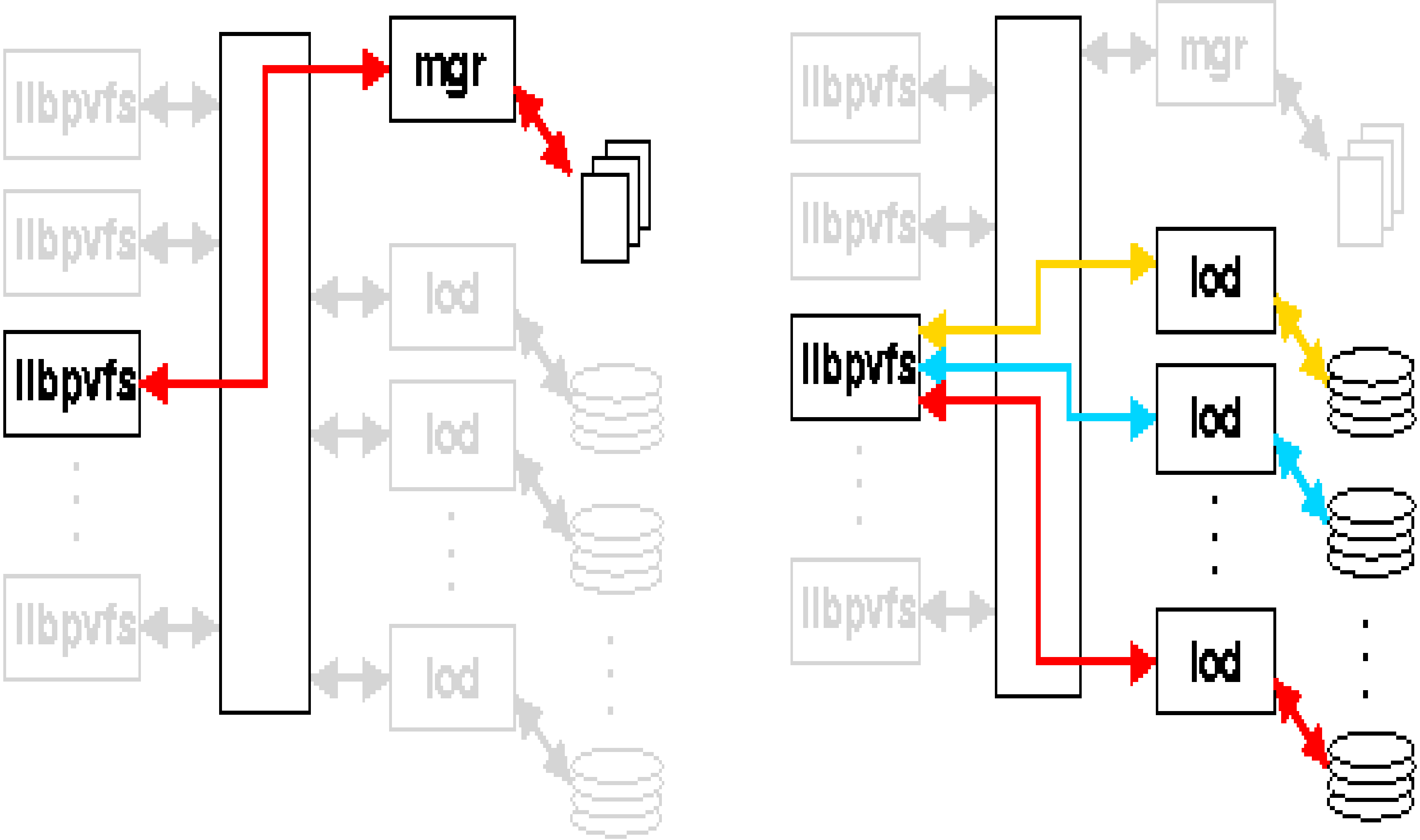
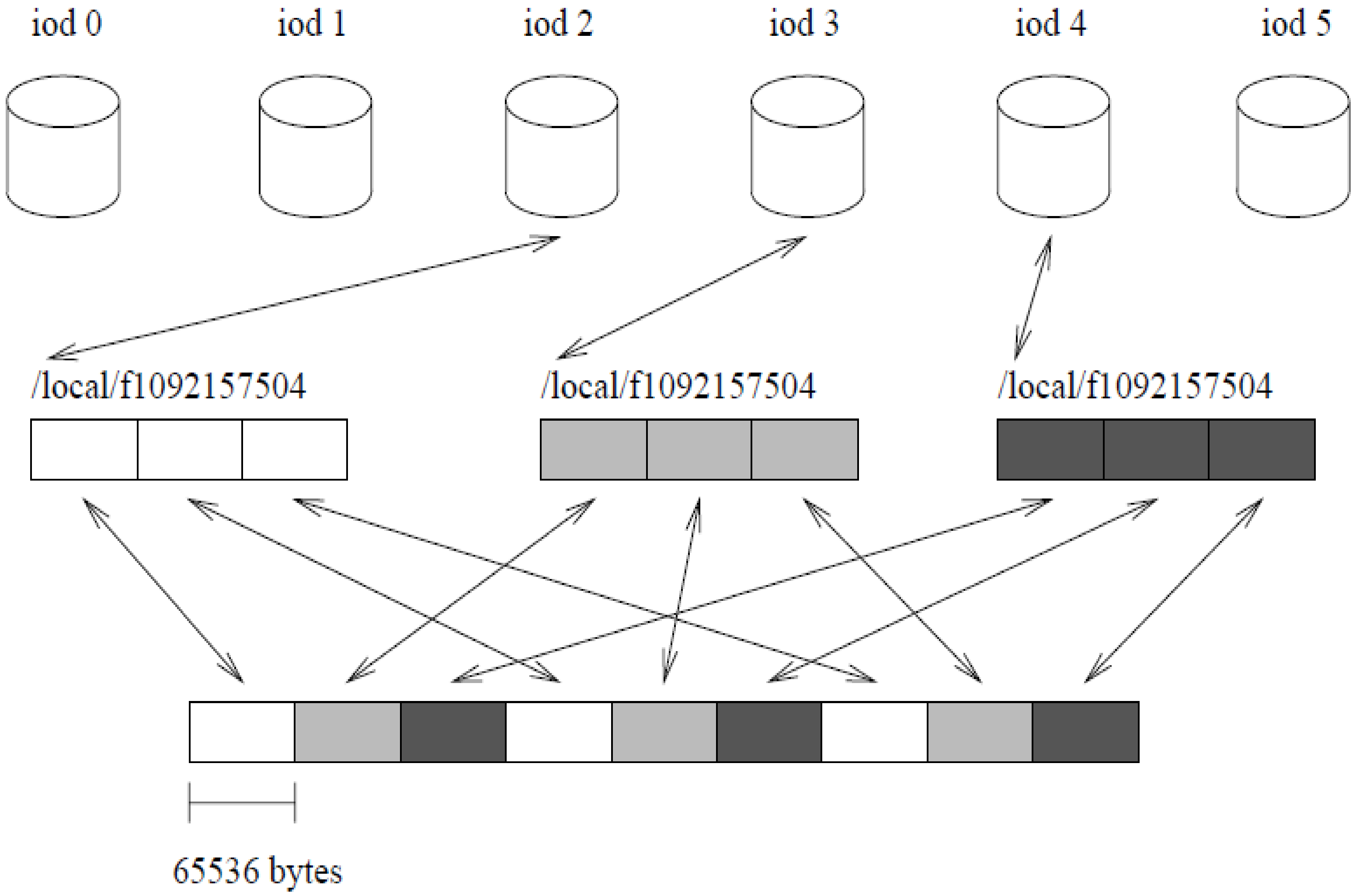


Table 1: Metadata example: File `/pvfs/foo`.

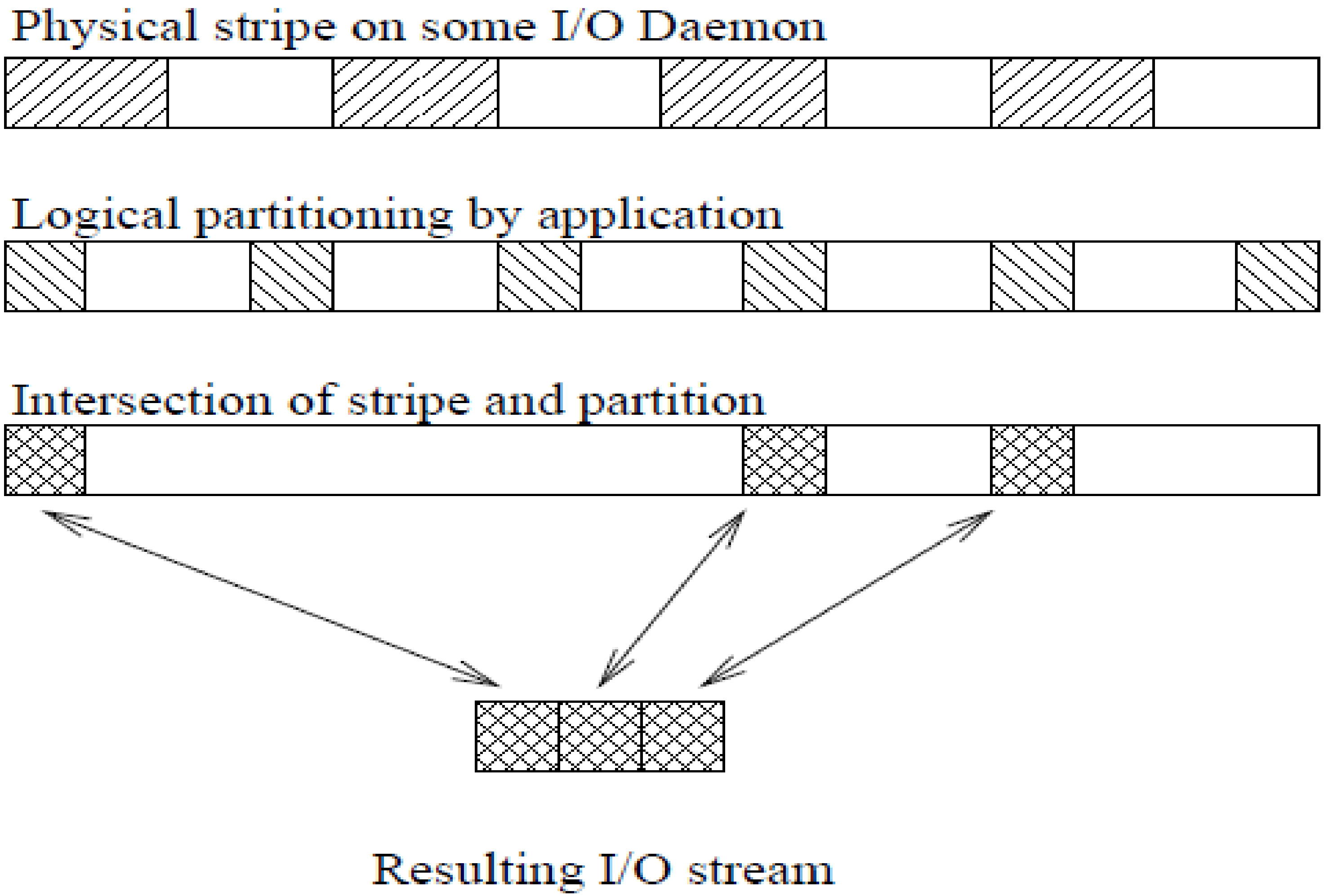
inode	1092157504
⋮	⋮
base	2
pcount	3
ssize	65536


libpvfs
libpvfs
libpvfs
libpvfs
mgr
iod
iod
iod
libpvfs
libpvfs
libpvfs
libpvfs
mgr
iod
iod
iod

I/O Daemons

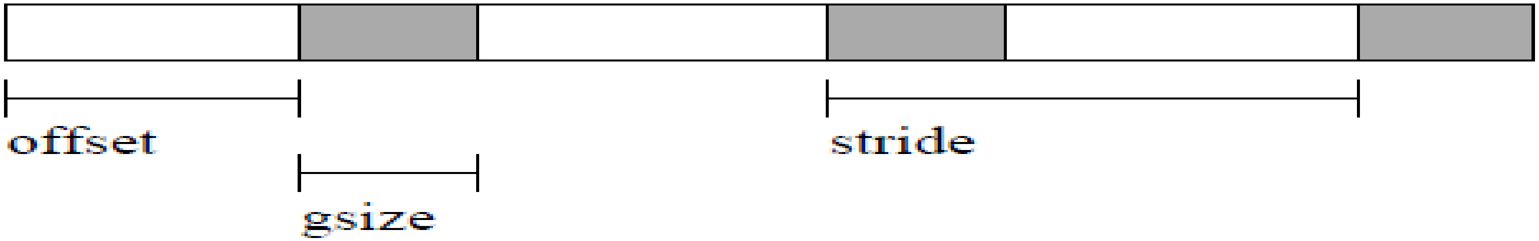

iod 0
iod 1
iod 2
iod 3
iod 4
iod 5
/local/f1092157504
/local/f1092157504
/local/f1092157504
65536 bytes

I/O Stream.

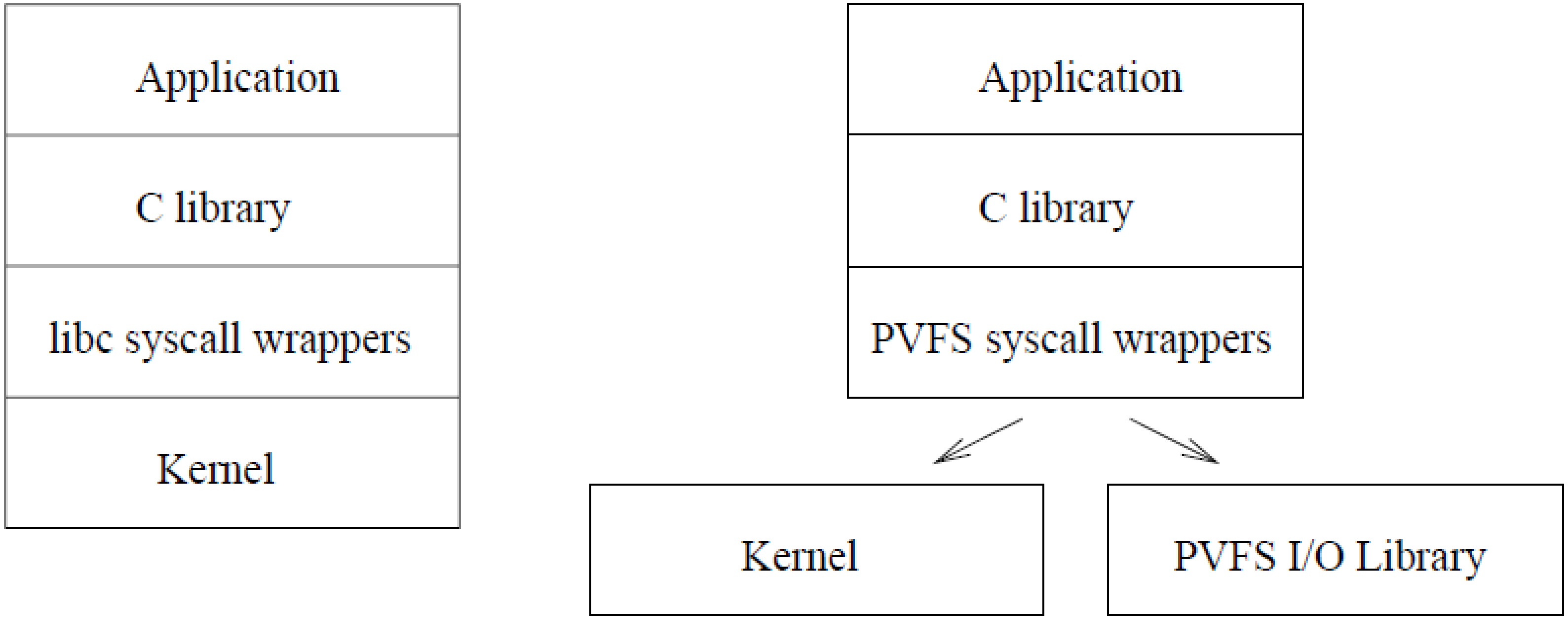

Physical stripe on some I/O Daemon
Logical partitioning by application
Intersection of stripe and partition
Resulting I/O stream

API

- Native PVFS api.
- UNIX/POSIX api.
- MPI – I/O api.
- Native api also include support for partitioned-file interface, which supports simple strided access.



Trapping UNIX I/O Calls



Application

C library

libc syscall wrappers

Kernel

a) Standard operation

Application

C library

PVFS syscall wrappers

Kernel

PVFS I/O Library

b) With PVFS library loaded

Performance Results

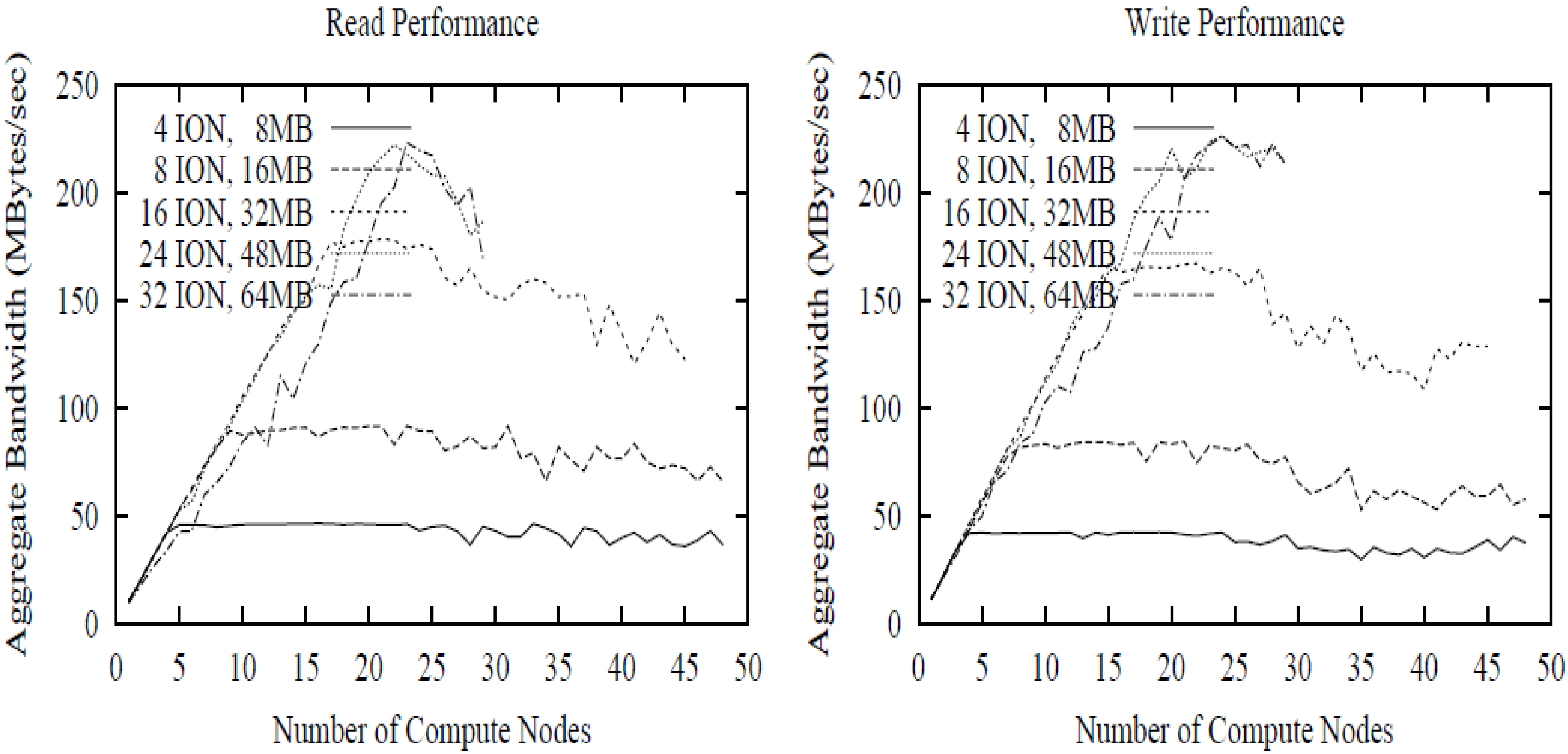


Figure 5: PVFS performance with fast ethernet

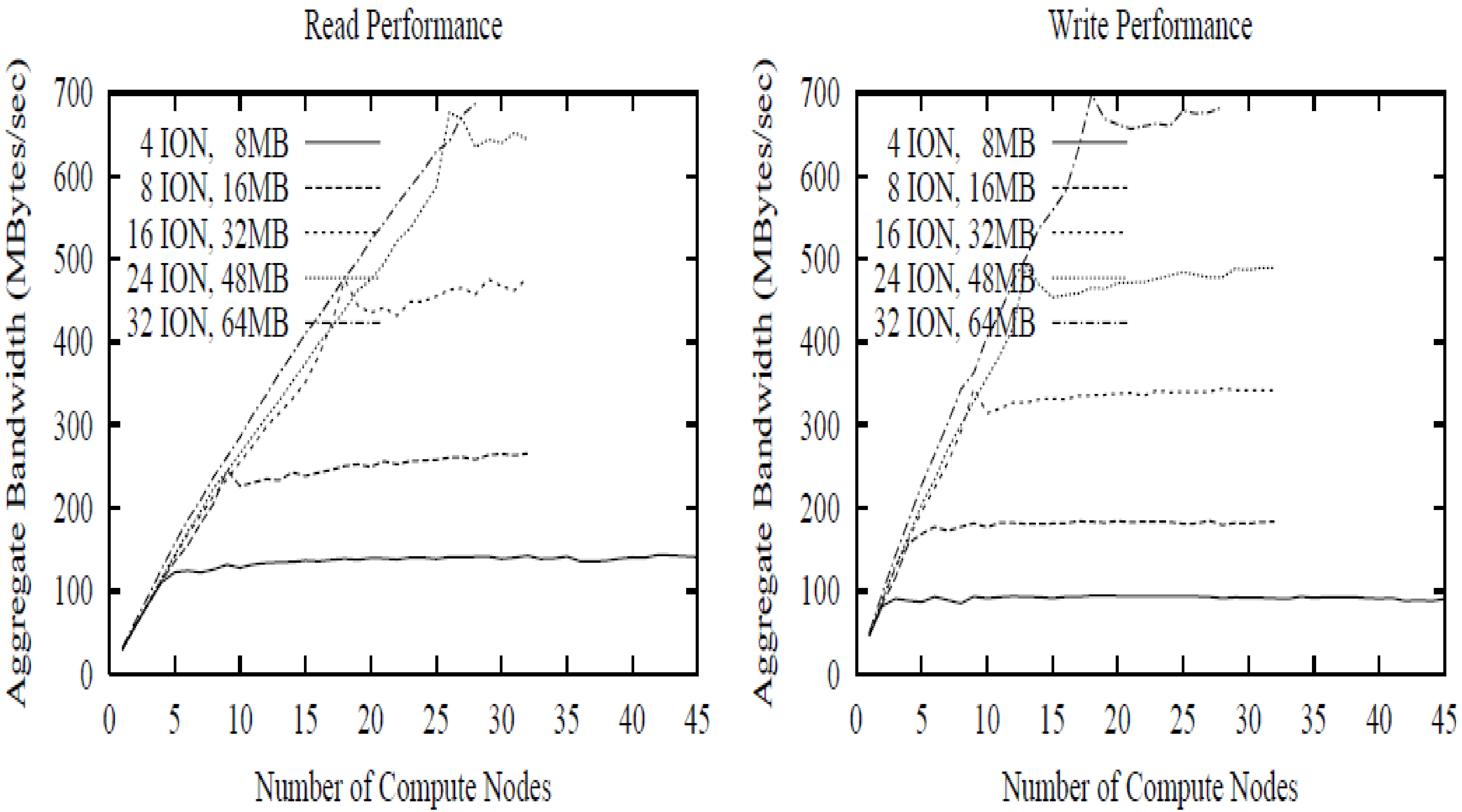


Figure 6: PVFS performance with Myrinet

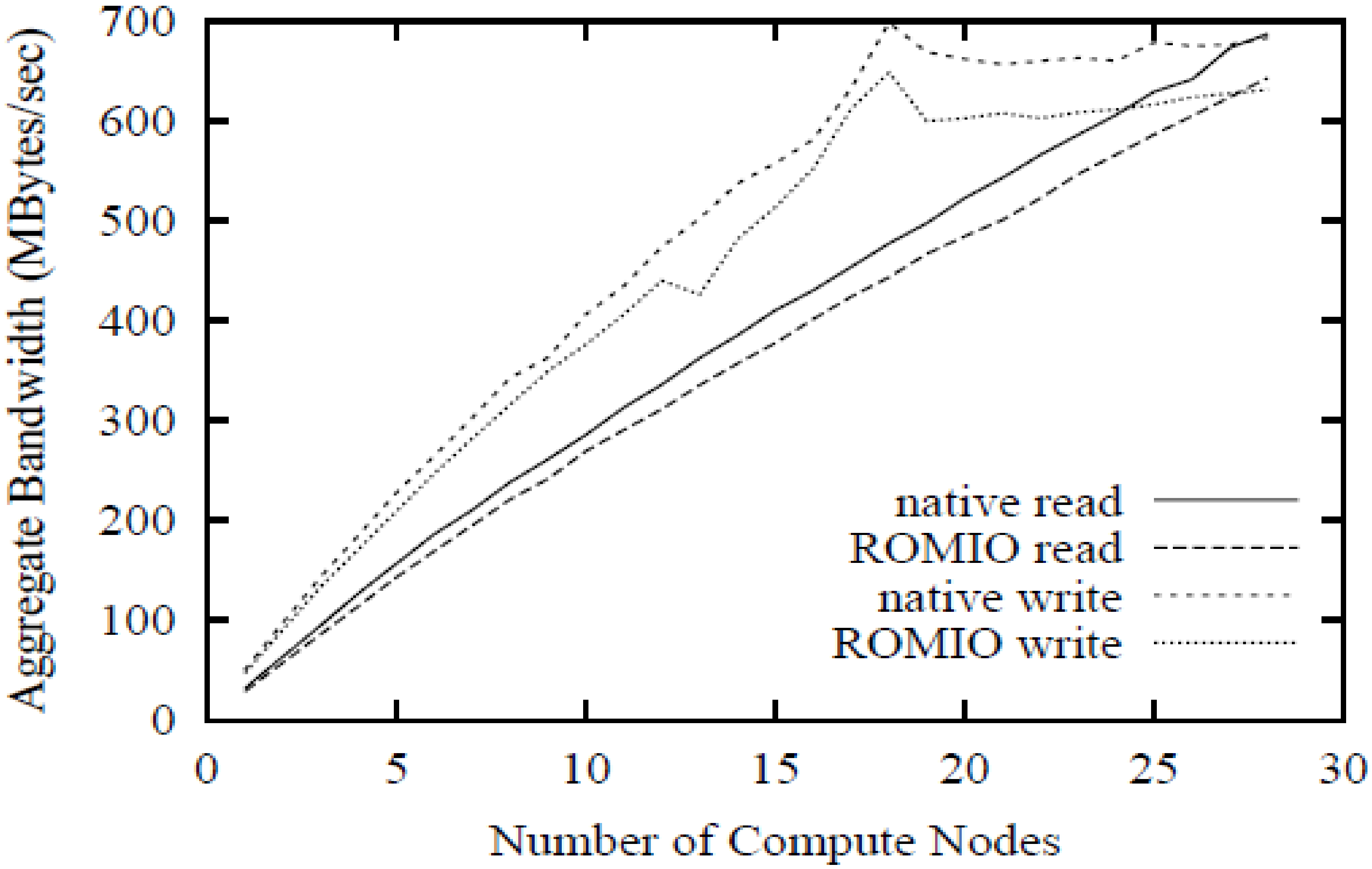


Figure 7: ROMIO versus native PVFS performance with Myrinet and 32 I/O nodes

Future Work....

- Support for faster communication mechanisms.
- Scope for tuning.
- General file partitioning interface.
- Design an new internal I/O description format that is more flexible.
- Adding redundancy support.
- Better scheduling algorithms for use in I/O daemons.

PVFS2

- Supporting New Hardware Technologies (Buffered Messaging Interface).
- System Monitoring.
- Data Migration.