Evaluating the Shared Root File System Approach for Diskless High-Performance Computing Systems

Christian Engelmann, Hong Ong, Stephen L. Scott

Computer Science and Mathematics Division

Oak Ridge National Laboratory

Oak Ridge, TN, USA

OAK RIDGE National Laboratory

Outline

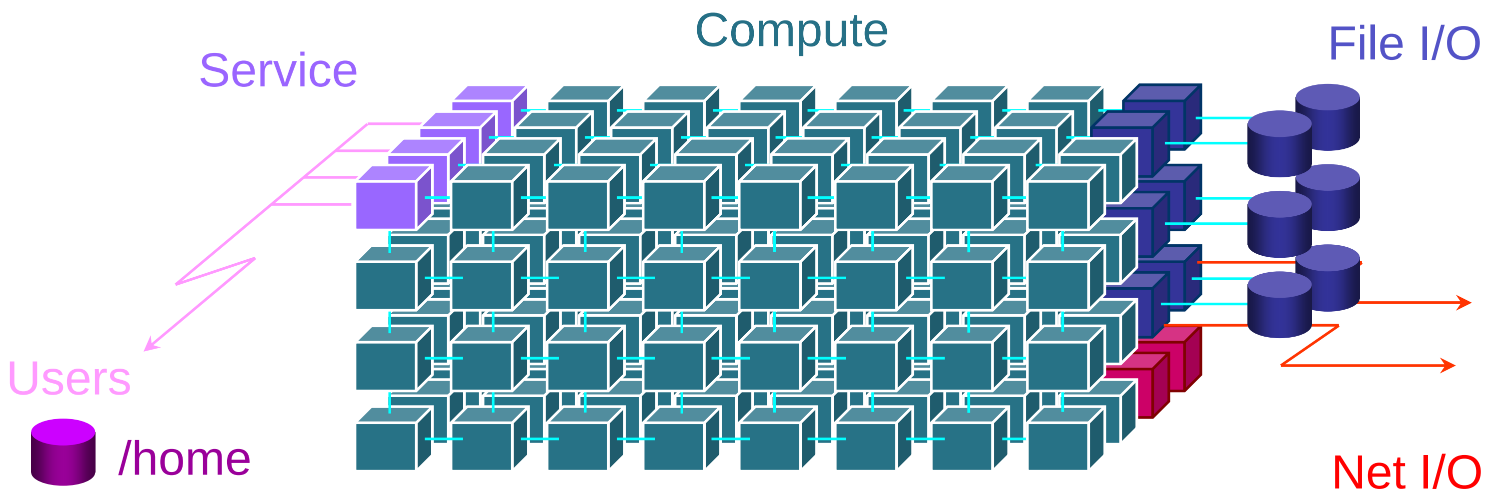
- **Motivation**
- **Architecture of Shared-Root File System**
- **Testing Environment**
  - **Hardware/Software**
- **Evaluation**
  - **Performance**
  - **Scalability**
  - **Availability**
- **Concluding Remarks and Future Work**

Motivation

- **Manageability, Scalability and Availability are key issues in large-scale HPC systems.**
- **Recent trend indicates HPC system architectures opt for diskless compute nodes.**
  - **Examples are Blue Gene/L, Cray XT, LANL Pink.**
  - **Utilise high-performance storage and high-speed network.**
  - **Removing disk drives significantly increases compute node reliability.**
- **However, typical diskless compute nodes require for a common root file system, e.g., Linux.**

Motivation (2)

- **Possible solutions to provide a common root file system for compute nodes are:**
  - **Remove the requirement and provide accesses to a networked, shared hierarchal storage for application.**
  - **Provide a common shared root file system via remote boot method.**
- **A cotemporary HPC architecture.**



Architecture of a Shared-Root File System

- **Three approaches:**
  - **Partition-wide sharing across compute nodes.**
  - **System-wide sharing across I/O service nodes.**
  - **Hybrid approach – combination of above two approaches.**
- **All approaches:**
  - **Root file system is mounted over the network by each compute node.**
    - **Mount root file system via NFS export points.**
  - **Configuration specific directories, such as /etc, are mounted over the network separately by each compute node.**

Aims of the Study

- **Diskless HPC distributions offer NFS-based root file system.**
  - **Parallel file systems are solely for application data and check-pointing due to high scalability and performance.**
  - **Parallel file systems are perceived to rely on complex stack of kernel modules and system utilities.**
- **This study uses parallel file systems for the implementation of a shared root environment.**
  - **Aim to improve scalability and high availability.**
- **Methodology:**
  - **Tests on the various parallel file systems are to be made on the same hardware for reliable comparison.**
  - **Evaluate performance of parallel root file system.**
  - **Understand root I/O access pattern.**

Testing Environment

- **Hardware**
  - **A cluster of 30 nodes, interconnected via a HP Fast Ethernet switch.**
  - **Each node is equipped with:**
    - **A 2.66 GHz Intel Xeon, 512 Kbyte L2 Cache, 1 Gbyte RAM.**
    - **A 80 Gbytes Western Digital IDE at 7200 RPM, 2MB Cache.**
- **Software**
  - **OS: Debian GNU/Linux, kernel 2.6.15.6.**
    - **Kernel is configured with NFSv4, Lustre, PVFS2 FS.**
  - **IOR benchmark – a parallel program that performs concurrent writes and reads to/from a file using the POSIX and MPI-IO interfaces.**

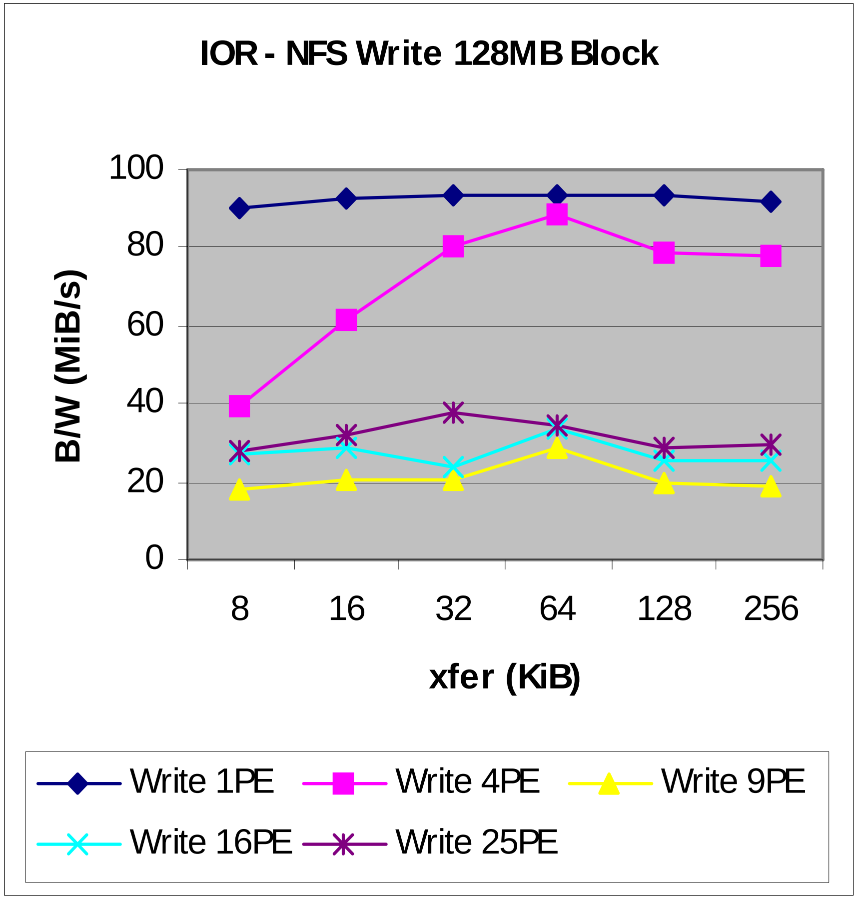
Software Infrastructure

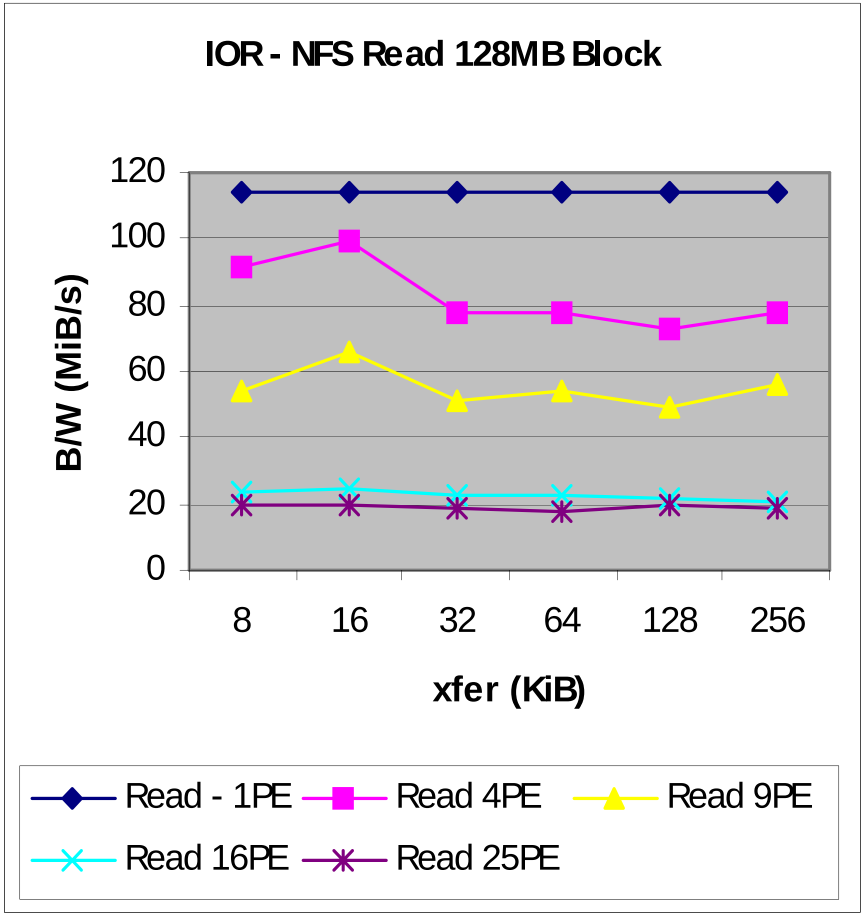
- **I/O servers:**
  - **PXE or ether boot.**
  - **Store kernel and initial ram disk images.**
    - **Initial ram disk contains an image of the whole root file system.**
    - **The root file system on compute nodes is memory resident.**
  - **Disks partitioned in 3 slices (NFS/PVFS/Lustre), managed by LVM2.**
  - **PVFS and Lustre see one multiple device partition.**
    - **40 GB x 3 disks = 120 GB for PVFS/Lustre.**
- **Compute nodes:**
  - **PXE or ether boot.**
  - **Kernel 2.6.15 and Lustre 1.4.6.4 patches.**
    - **PVFS2 does not require patches to the kernel.**
  - **/home are NFS-mounted from the login server.**
    - **This is not to waste the local disks of the file servers.**

Parallel Root Filesystems Testbed Configuration

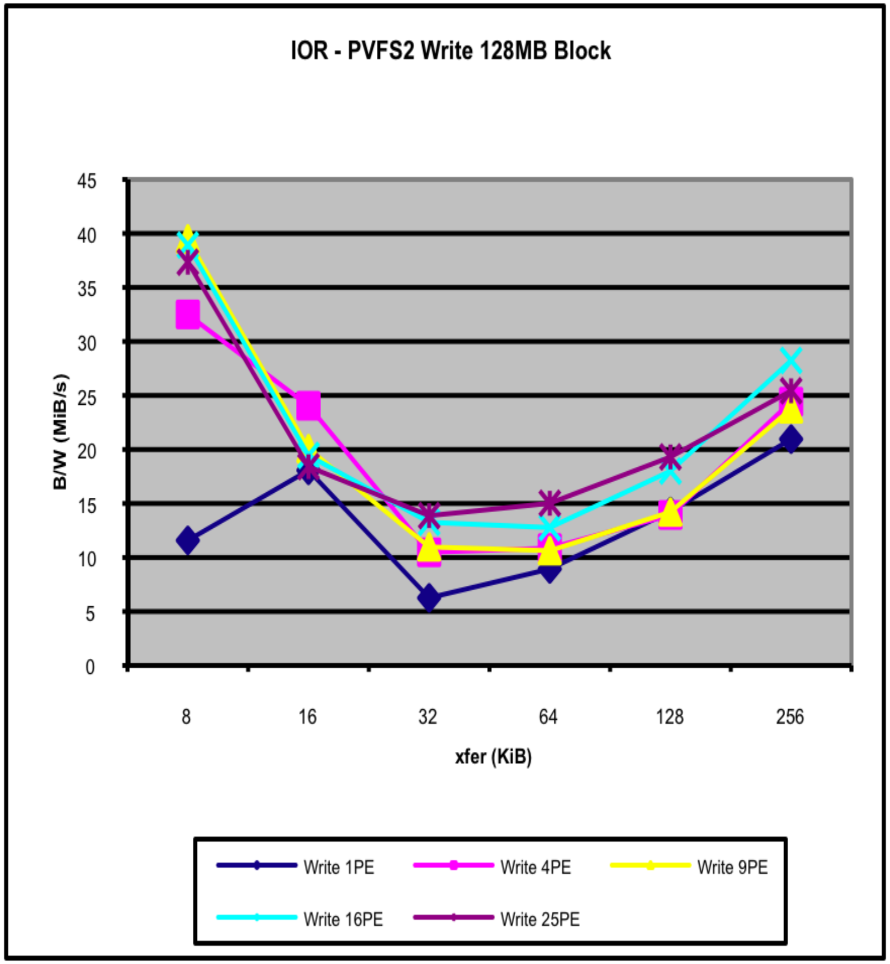
- **NFS RootFS.**
  - **Default configuration with 30 NFS servers pool.**
- **PVFS-2.**
  - **1 metadata server and 3 data servers.**
  - **Data and metadata stored on an ext3 partition.**
  - **Default stripe size 64k (but can be changed from file to file if using native calls).**
- **Lustre.**
  - **1 metadata server and 3 data servers.**
  - **Lustre relies on ext3 as underlying file system.**
    - **It can make a low level format of a physical device or access an already formatted device by pre-allocating a continuous slice of disk in a single file, using it as storage.**
  - **Default stripe size of 64k.**

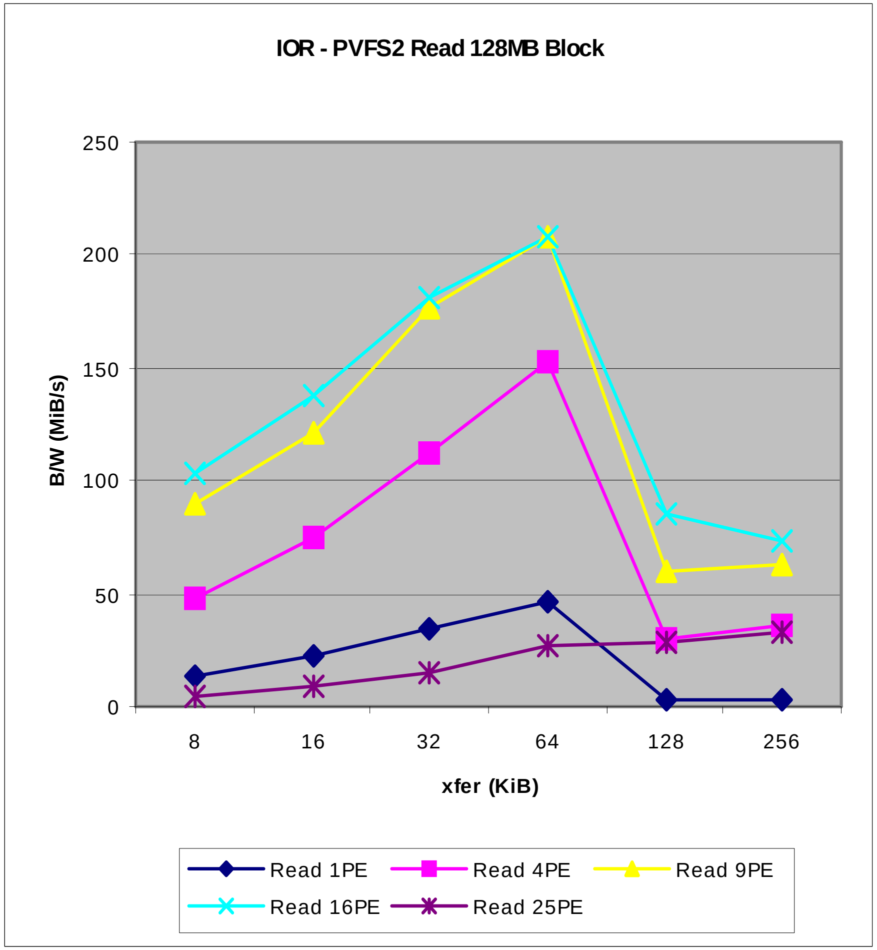
Performance – NFSv4 Read/Write



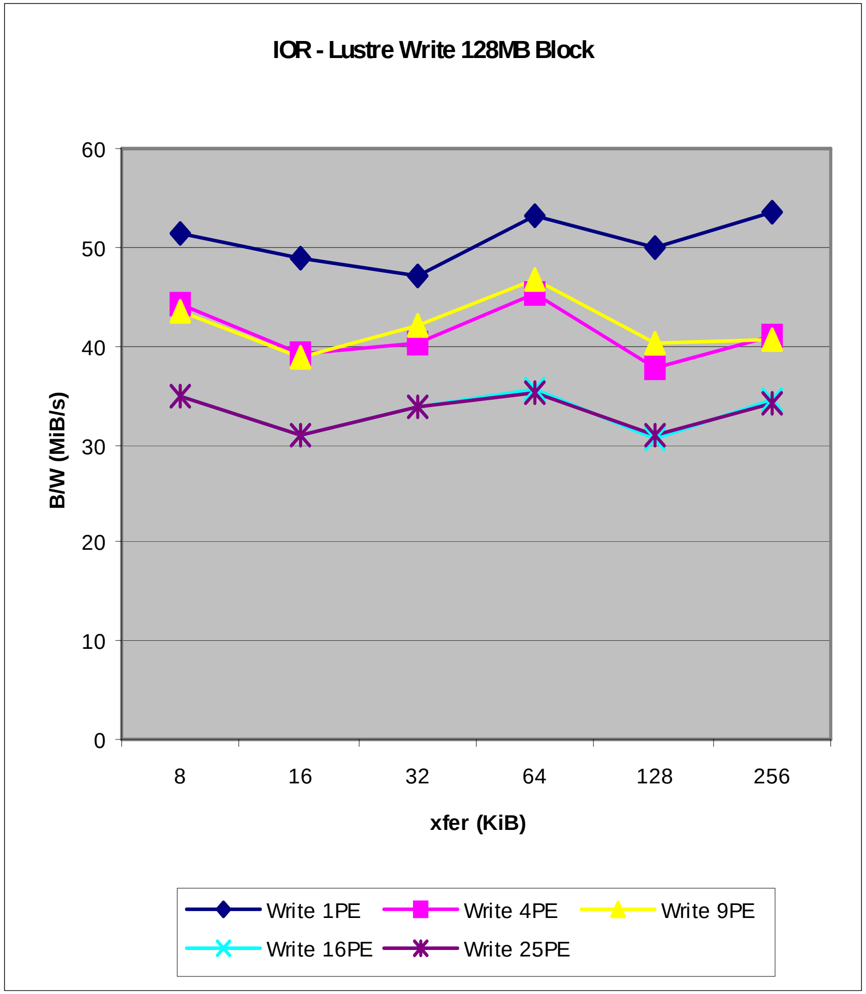


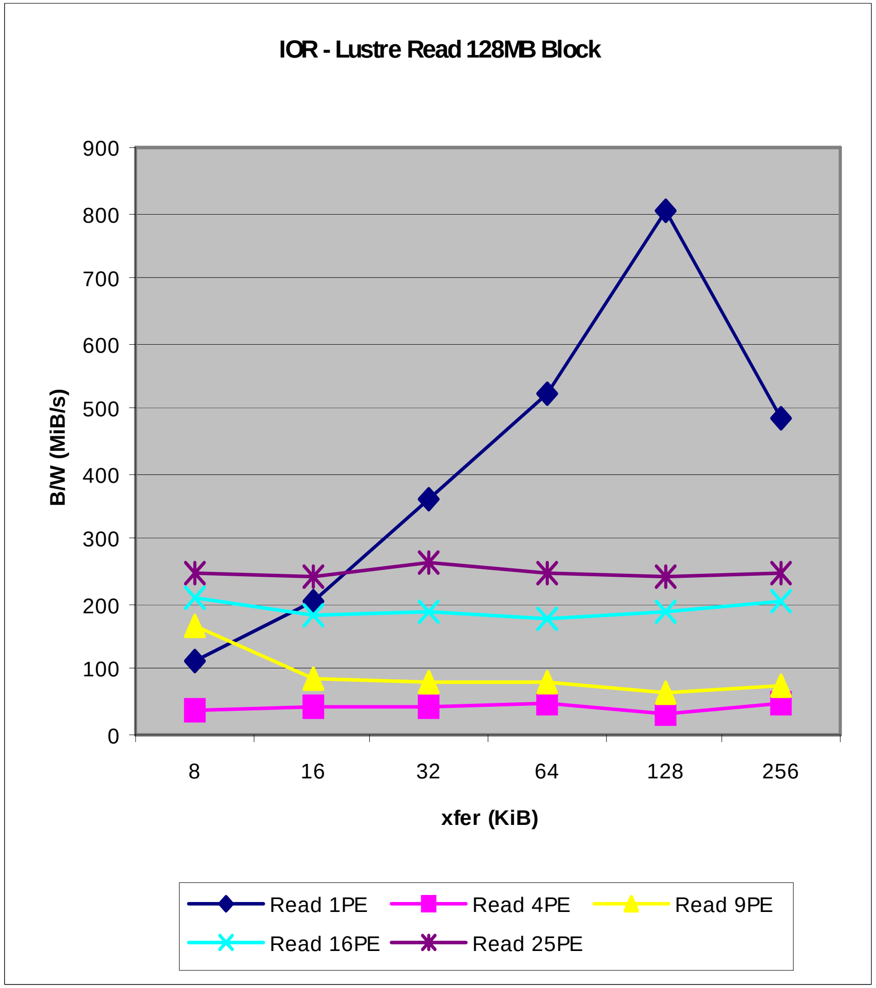
Performance – PVFS2 Read/Write



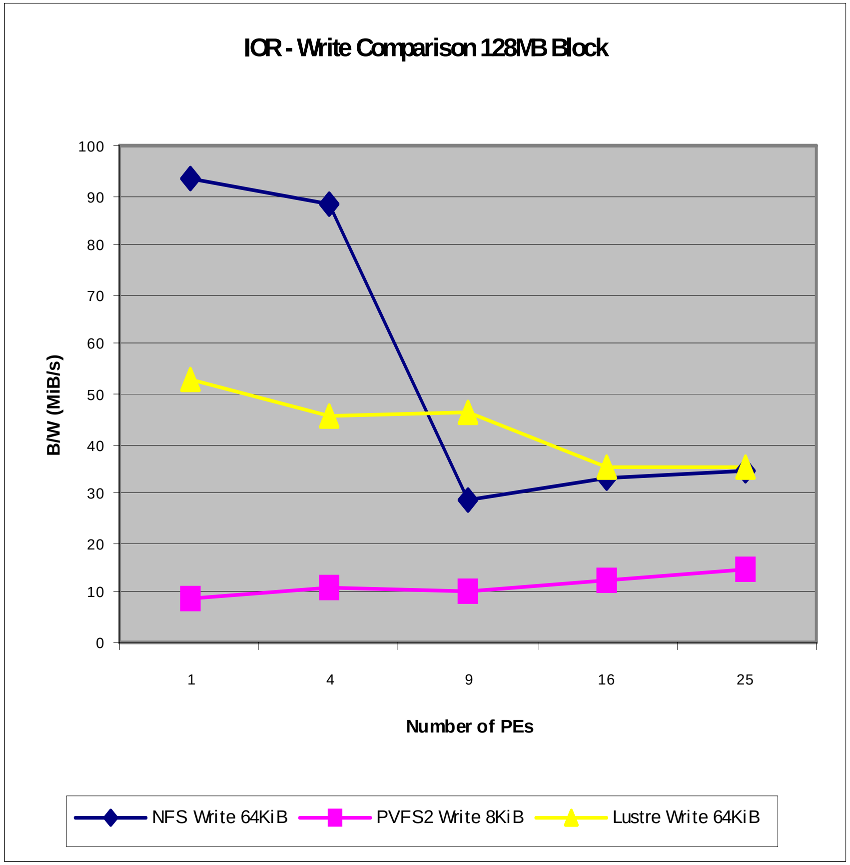


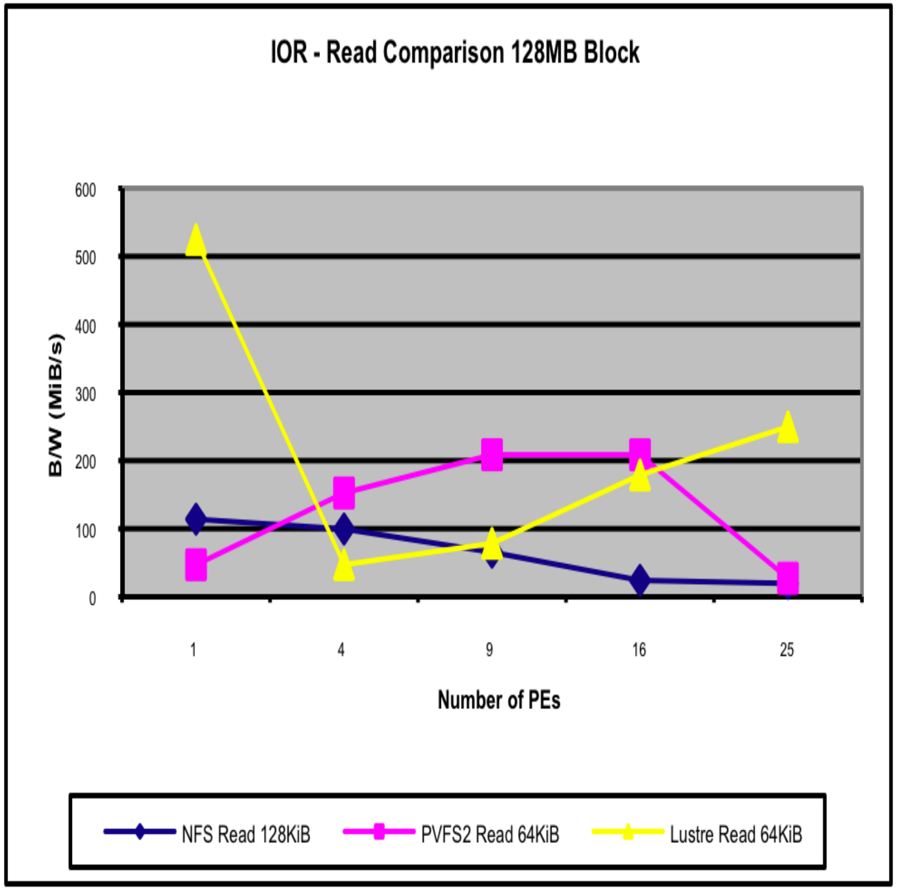
Performance – Lustre Read/Write





Scalability





- Lustre and PVFS2 scale reasonably well as the number of clients increase.
- Lustre and PVFS2 does not perform well for small reads/writes.
- NFSv4 read/write performance and scalability are limited by its single server architecture.

Shared Root FS Availability

- **Possible drawback to address w.r.t diskless.**
  - **The absence of a disk swap area essentially means that the job memory demand must strictly fit into the RAM, otherwise the job could be abruptly terminated.**
- **Possible drawback to address w.r.t high availability.**
  - **In general this is still a gray area.**
  - **NFSv4 has a single point of failure for the entire system.**
  - **MDS is a single point of failure for PVFS2 and Lustre.**
    - **Storage servers can utilise data replication to provide high availability.**

High Availability for Share-root Environment

- **NFSv4, PVFS2, and Lustre do not have built-in high-availability support.**
- **Typical solution uses active/standby or active/active configuration.**
  - **For example, SLURM and DRDB.**
  - **Both methods require heartbeat monitor mechanism.**
    - **MTTR depend on the heartbeat interval, may vary between a few seconds to several minutes.**
- **Our previous work on symmetric active/active replication could be a solution (see citations in paper).**
  - **Basically, it uses multiple redundant service nodes running in virtual synchrony via a state-machine replication mechanism.**
    - **It does not depend on fail-over to backup.**
  - **Attained 26ms latency for PVFS MDS writes.**

Concluding Remarks

- **Multiple options are available for attaching storage to diskless HPC.**
- **Our study showed that parallel file systems are viable option for serving a common root.**
- **NFS-based FS is sufficient for lightly I/O loads.**
  - **May not be able to scale to the volume of data/clients on large HPC systems.**
  - **NFS has a single point of failure and control.**
- **Parallel FS is efficient for heavier I/O loads.**
  - **Offer highest performance and lowest overall cost for accesses to data storage.**
    - **Illustrated that Lustre is a viable solution.**
- **Parallel FSs lack of efficient out-of-the-box solution for supporting high-availability.**

Future Works

- **Detailed study of each parallel filesystem w.r.t how the filesystems work internally and identify the best tunings on a larger scale system.**
  - **Study the time dependence of the throughputs**
  - **Study the filesystems scheduling and caching mechanisms.**
- **Perform measurements with an high-end storage system.**
- **Perform measurements with an high-speed network, e.g., InfiniBand.**

Questions?

Thank you