QoS-driven Storage Management for High-end Computing Systems

Ming Zhao, Yiqi Xu, Dulcardo Clavijo

Computing & Information Sciences
Florida International University
zhaom@cis.fiu.edu

Renato Figueiredo, Yonggang Liu

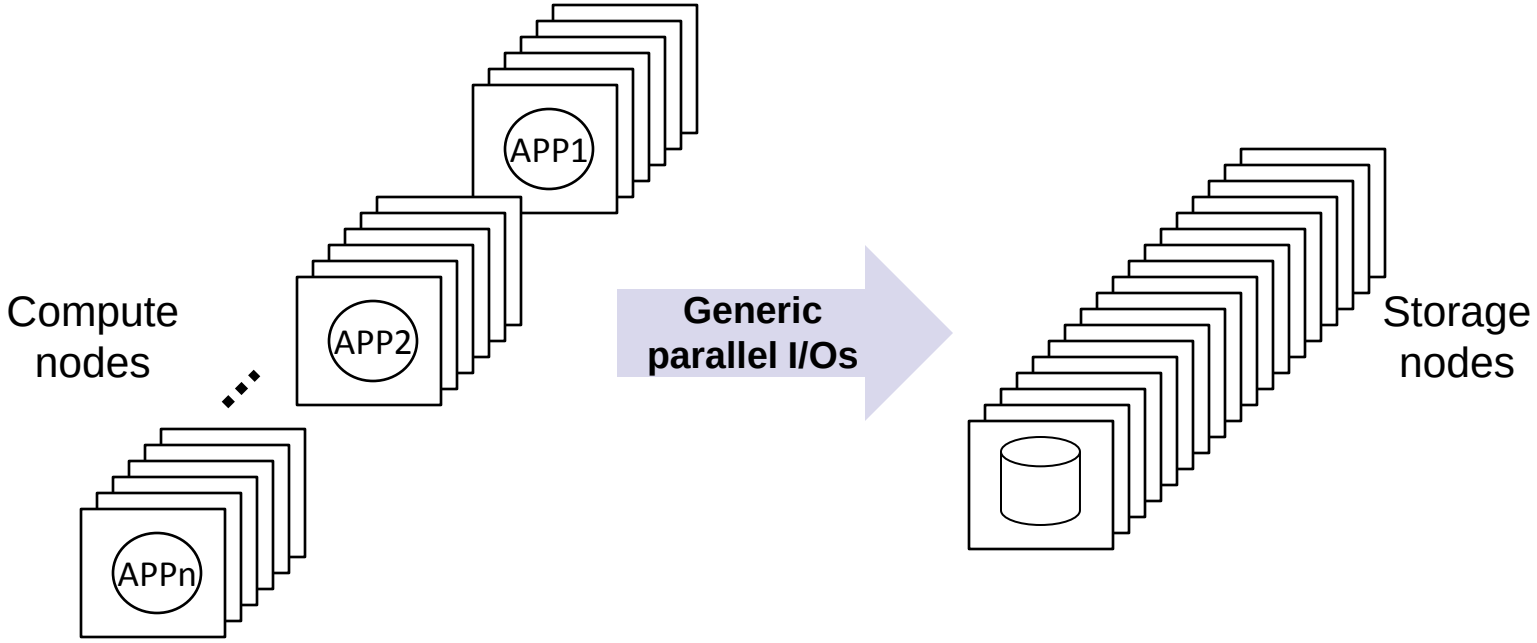
Electrical & Computer Engineering
University of Florida
renato@acis.ufl.edu

Outline

- Background
- Panel questions
- Our approach
- Current results
- Conclusions

Motivation

- The lack of QoS differentiation in HEC storage systems
  - Unable to recognize different application I/O workloads
  - Unable to satisfy users' different I/O performance needs



Motivation

- The need for different I/O QoS from HEC applications
  - Diverse I/O demands and performance requirements
  - Examples:
    - WRF: Hundreds of MBs of inputs and outputs
    - mpiBLAST: GBs of input databases
    - S3D: TBs of restart files on a regular basis
- This mismatch will become even more serious in future ultra-scale HEC systems

Panel Questions

- What types of QoS guarantees are useful to HEC application scientists?
  - Time
- What is the relationship between QoS infrastructure and other job scheduling infrastructure?
  - Integration
- Is the ratio of funded to productized work in QoS lower than in other areas? If so, why?
  - No!

Objectives

- Per-application storage resource allocation
  - Parallel file system (PFS) virtualization
  - Enable *transparent QoS-driven storage management*
- Efficient management of storage resource allocations
  - Storage management services
  - Allow *integration with job scheduling infrastructure*
- Automatic optimization of storage resources usage
  - Autonomic storage resource management
  - Support *application high-level QoS objectives*

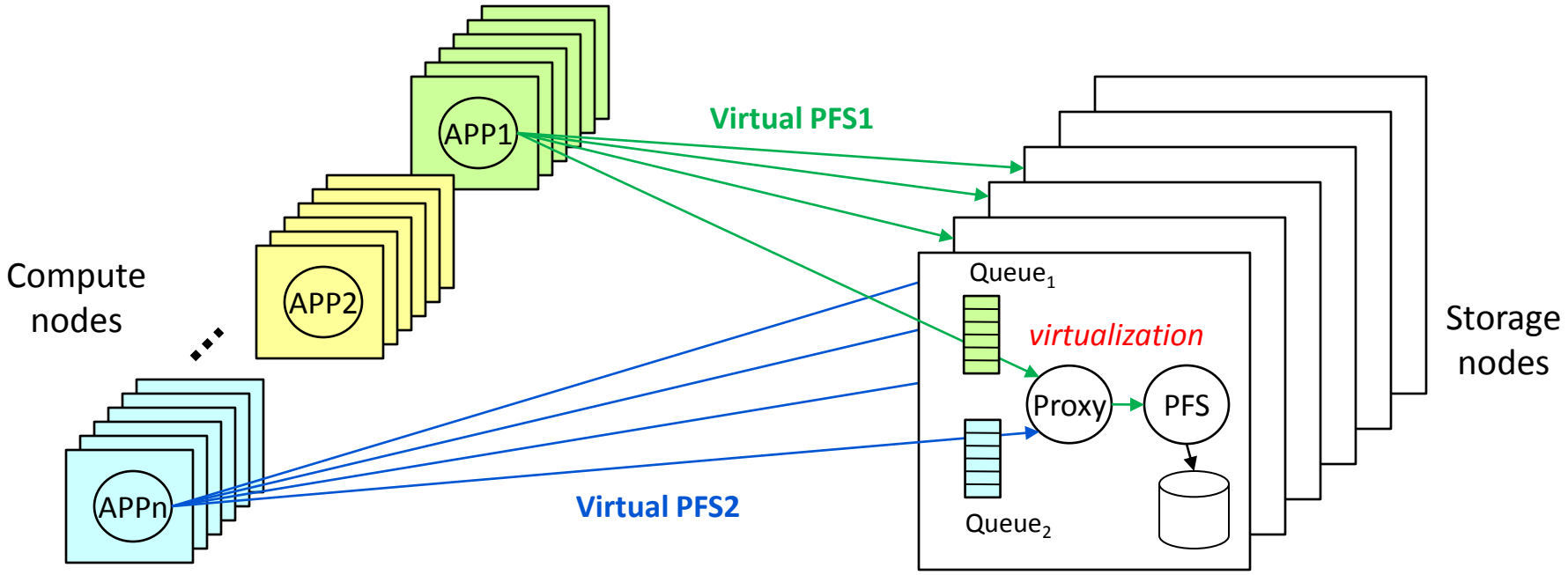
Per-application I/O Bandwidth Allocation

- Problem: Lack of per-application I/O bandwidth allocation
  - Static partition of storage nodes is inflexible
  - Compute nodes based partition is insufficient

- Proposed solution: PFS virtualization (*vPFS*)
  - Per-application virtual PFSs
    - Dynamically created and destroyed based on application lifecycles
  - Application-specific I/O bandwidth allocation per virtual PFS

PFS Virtualization

- Proxy-based PFS virtualization [*PDSW'10*]
  - Indirection of application I/O access
  - Creation of per-application virtual PFS



Prototype Implementation

- A PVFS2[1] (Parallel Virtual File System) proxy
  - Deployed on every data server
  - Intercept and forward PVFS2 messages
  - Identify and schedule I/Os per application
- SFQ[2]-based proportional bandwidth sharing
  - SFQ(D)[3]: proportional sharing of local storage bandwidth
  - DSFQ[4]: distributed scheduling for global fairness
  - Provide a generic interface for supporting other algorithms

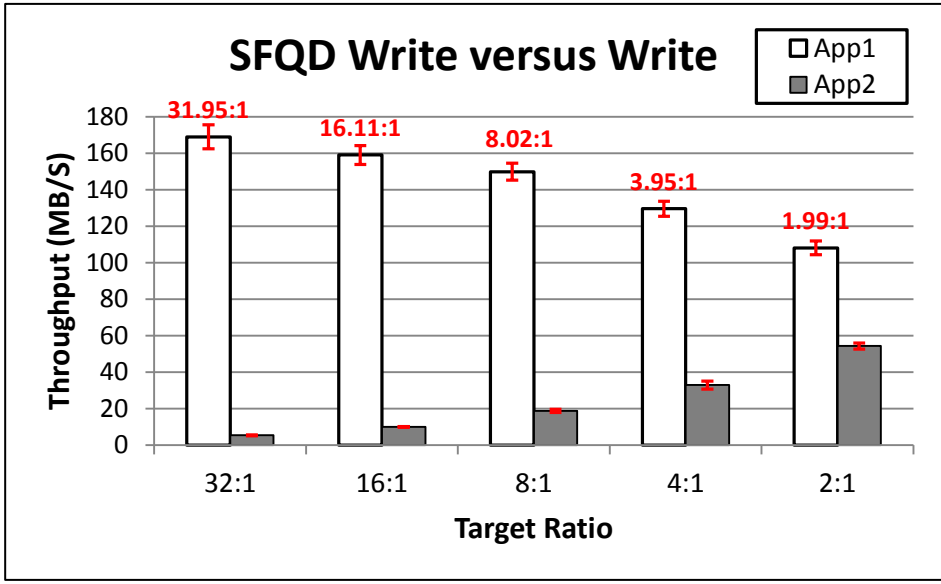
[1] PVFS2, http://www.pvfs.org/pvfs2/.
[2] "Start Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," IEEE/ACM Trans. Networking, vol. 5, no. 5, 1997.
[3] "Interposed Proportional Sharing For A Storage Service Utility," SIGMETRICS, 2004.
[4] "Proportional-share Scheduling for Distributed Storage Systems," FAST, 2007.

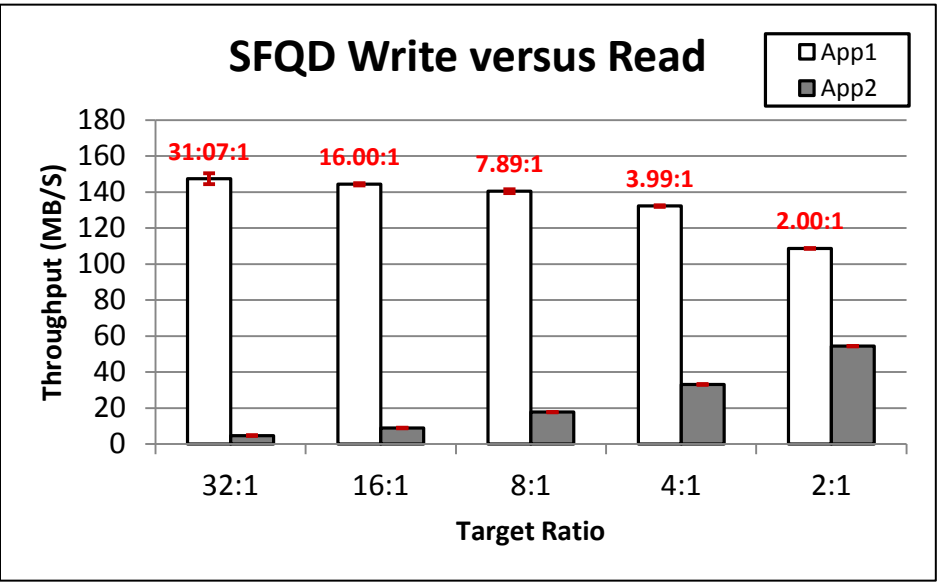
Evaluation

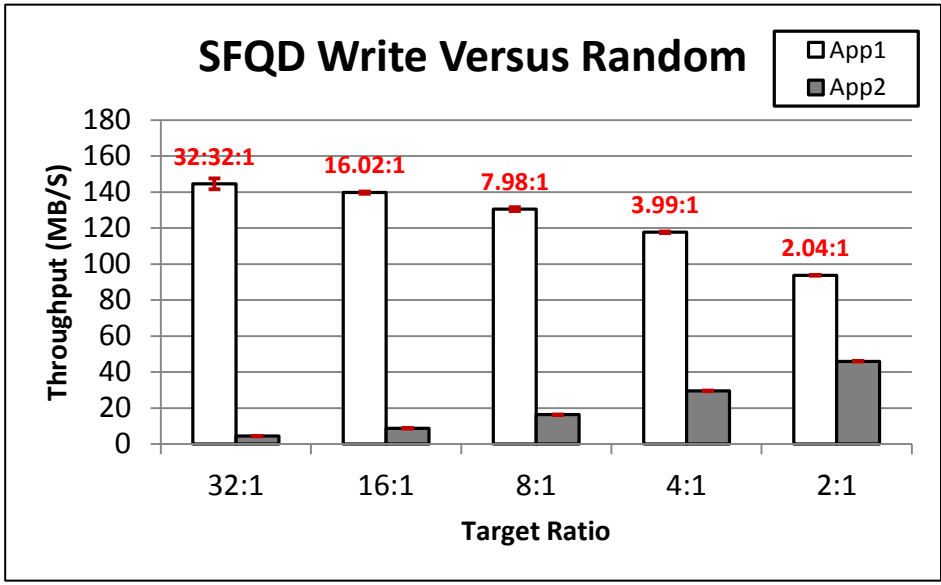
- Test bed
  - Compute nodes: a cluster of Opteron 2x6-core servers
  - Storage nodes: a cluster of Intel 2x6-core servers
  - Native PFS: PVFS version 2.8.2
- Benchmark: IOR version 2.10.3
  - Sequential reads/writes; also modified to issue random reads/writes *randomly*
  - Represent two competing parallel applications, each with 128 processes

Symmetric Sharing of Storage

- Two competing applications
  - Each with 128 processes
  - Sharing 8 PVFS2 servers
- Good proportional sharing can be achieved

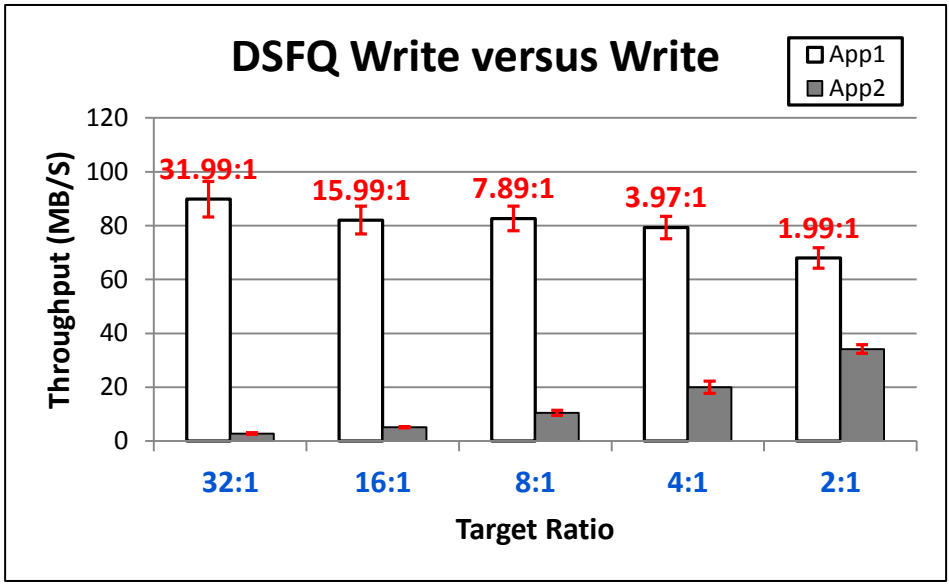


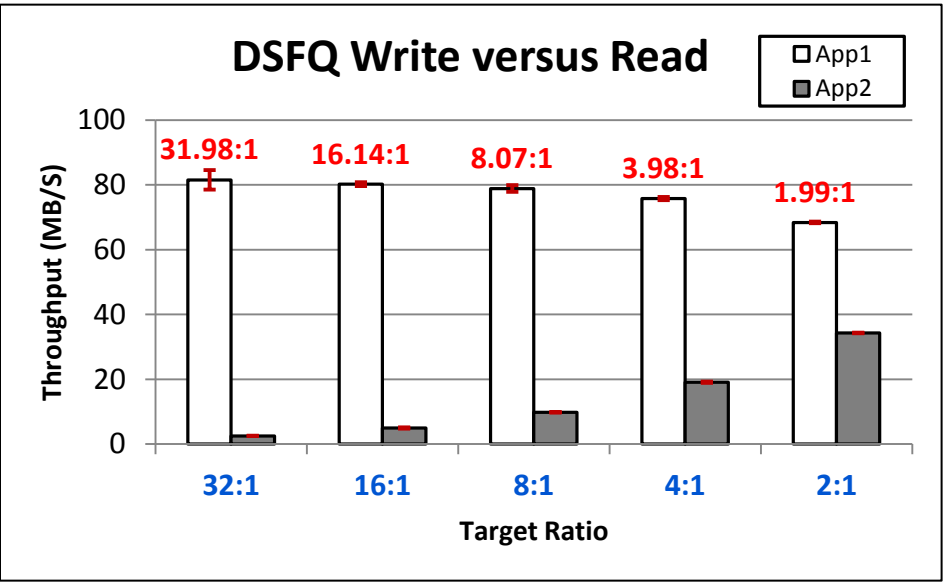


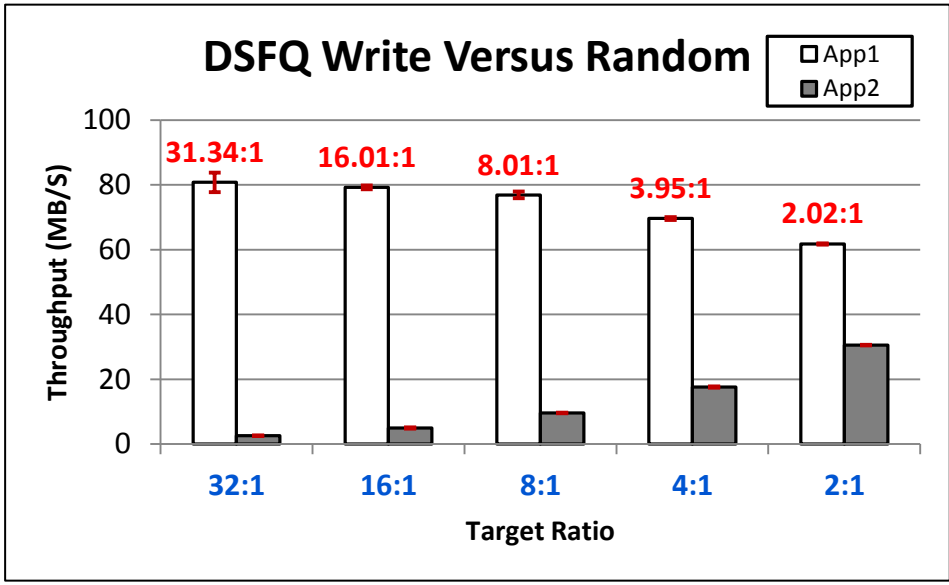


Asymmetric Sharing of Storage

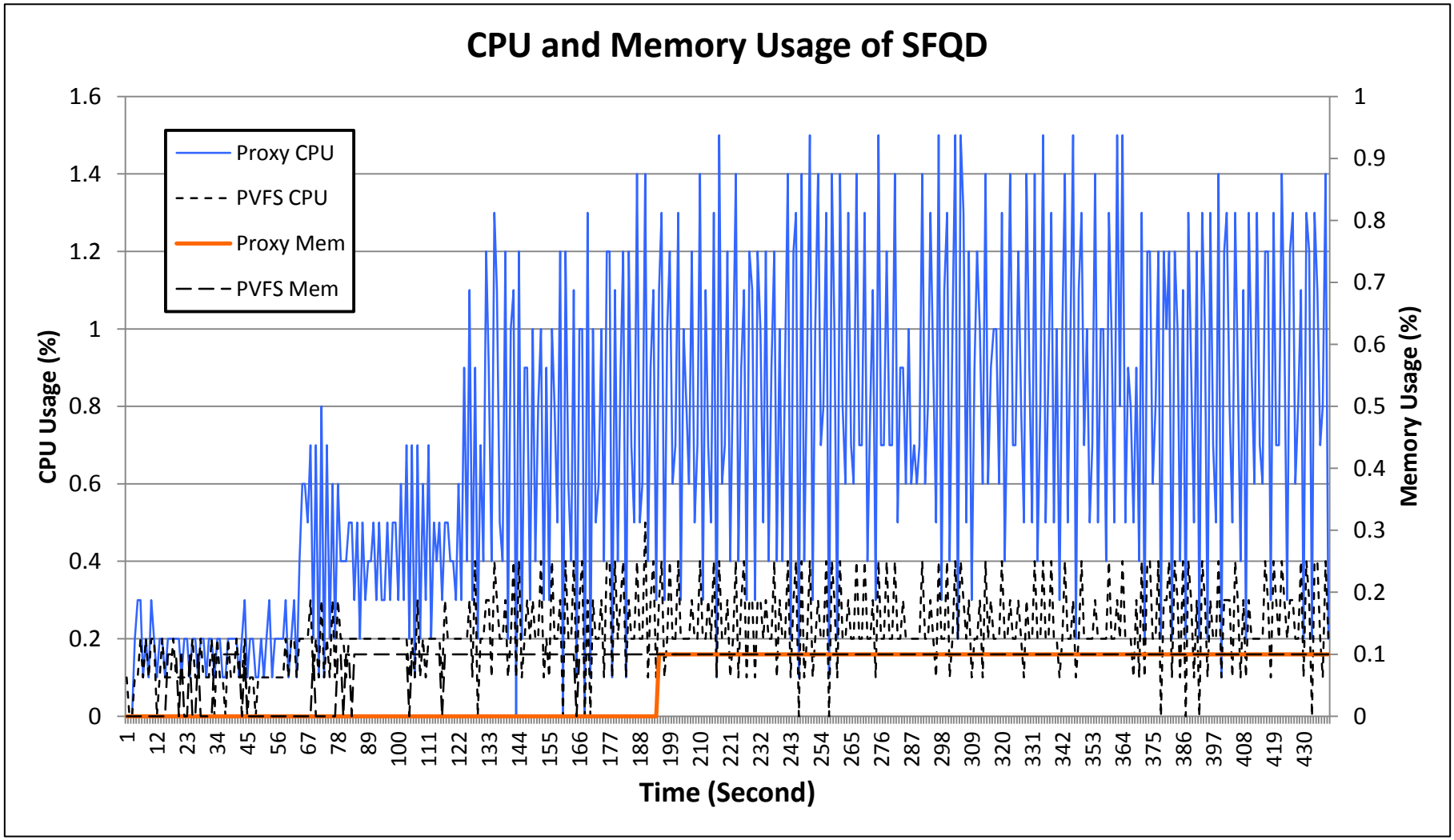
- Two competing applications
  - Each with 128 processes
  - App1 with only 4 PVFS2 servers
  - App2 with all 8 PVFS2 servers
- Threshold-driven global scheduling synchronization



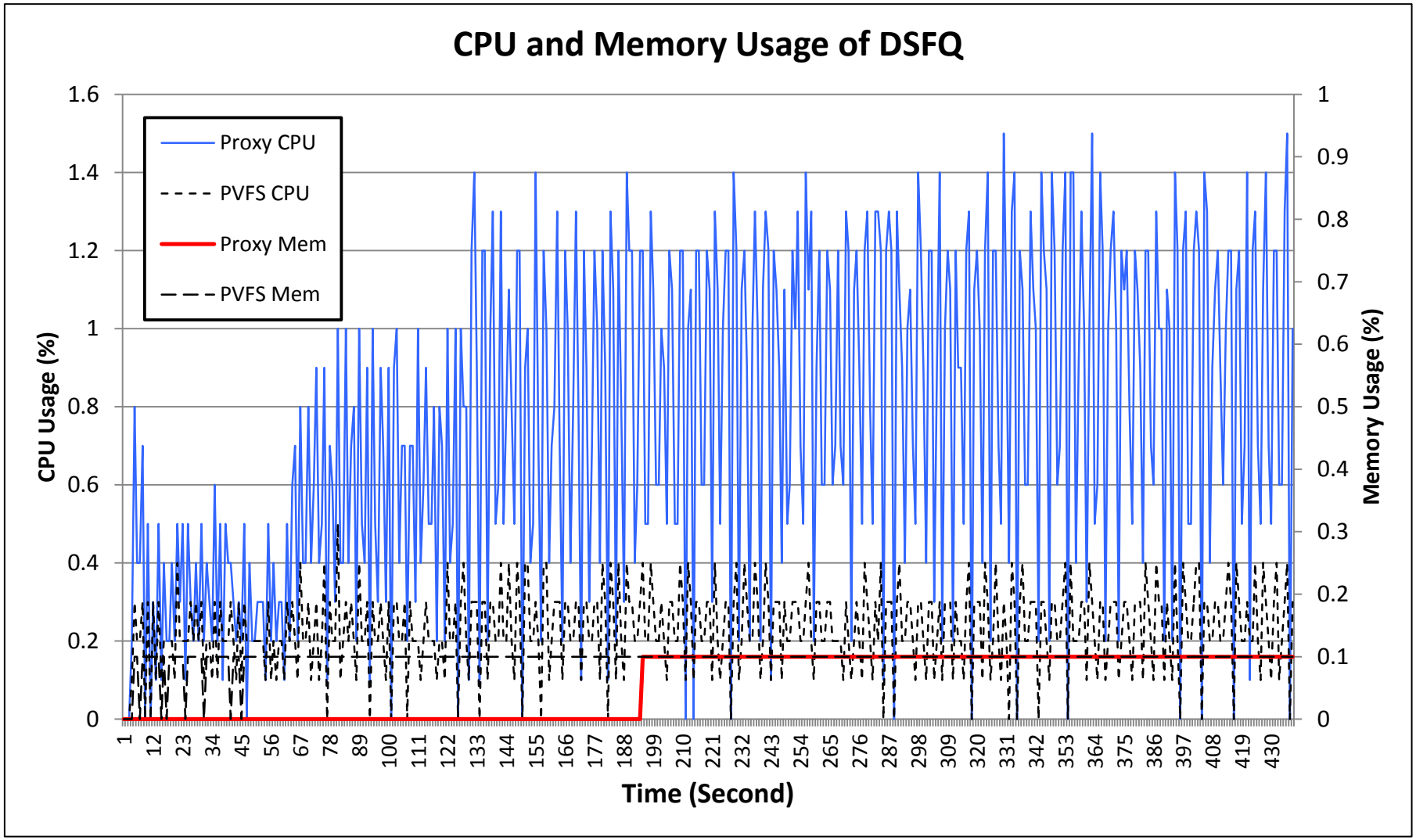




Virtualization Overhead



Virtualization Overhead

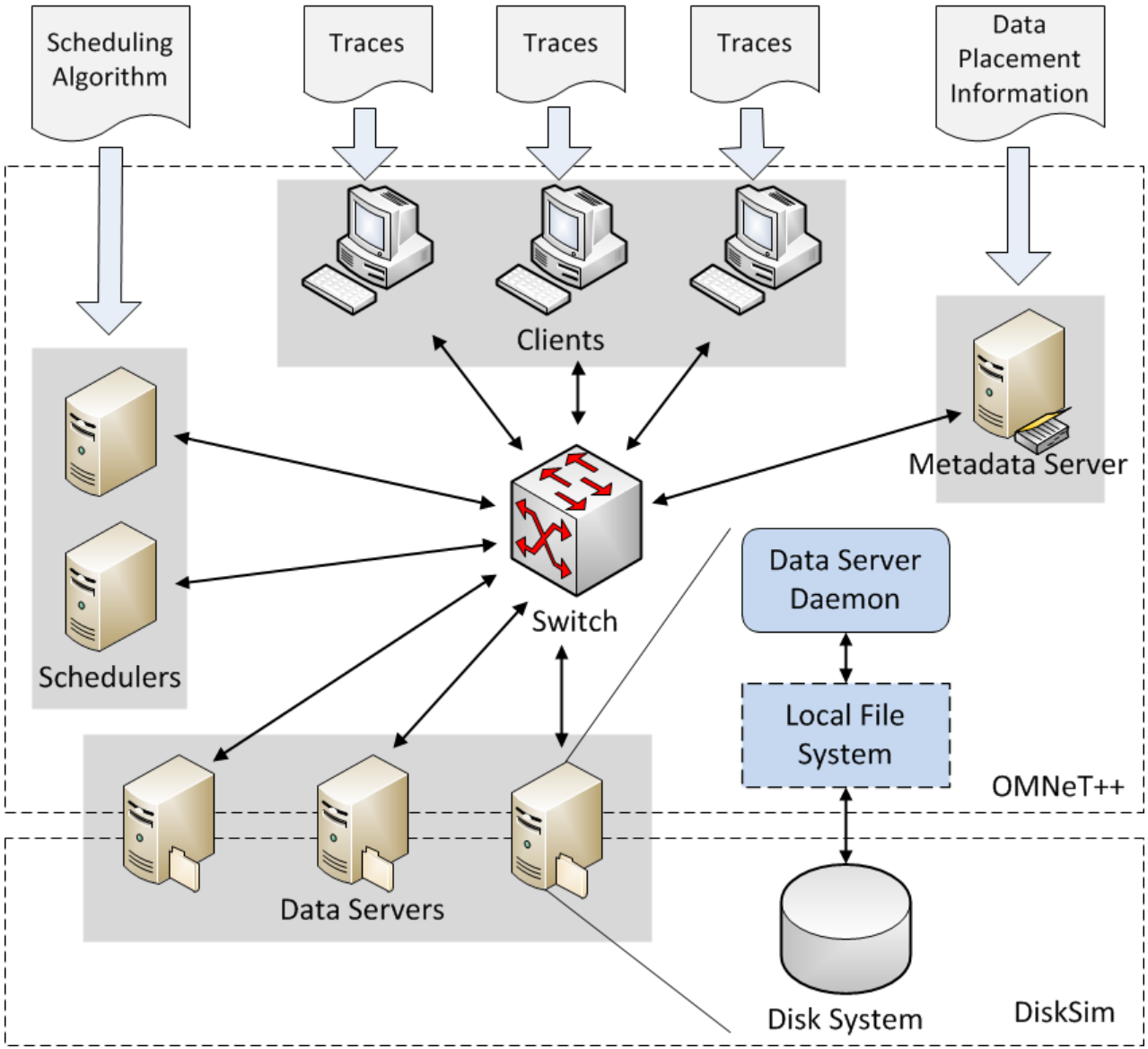


Simulation-based I/O Scheduling Study

- PFS simulator (*PFSsim*) [*SNAPI'11*]
  - To flexibly study parallel I/O scheduling algorithms
- Simulate PFS network
  - Use discrete event simulation library (OMNeT++ 4.0)
- Simulate PFS disks
  - Use DiskSim to simulate the disks
- Compared to other simulators[5][6][7], our focus is in parallel I/O scheduling
  - Simulate enough details necessary for scheduling study but with an acceptable simulation time

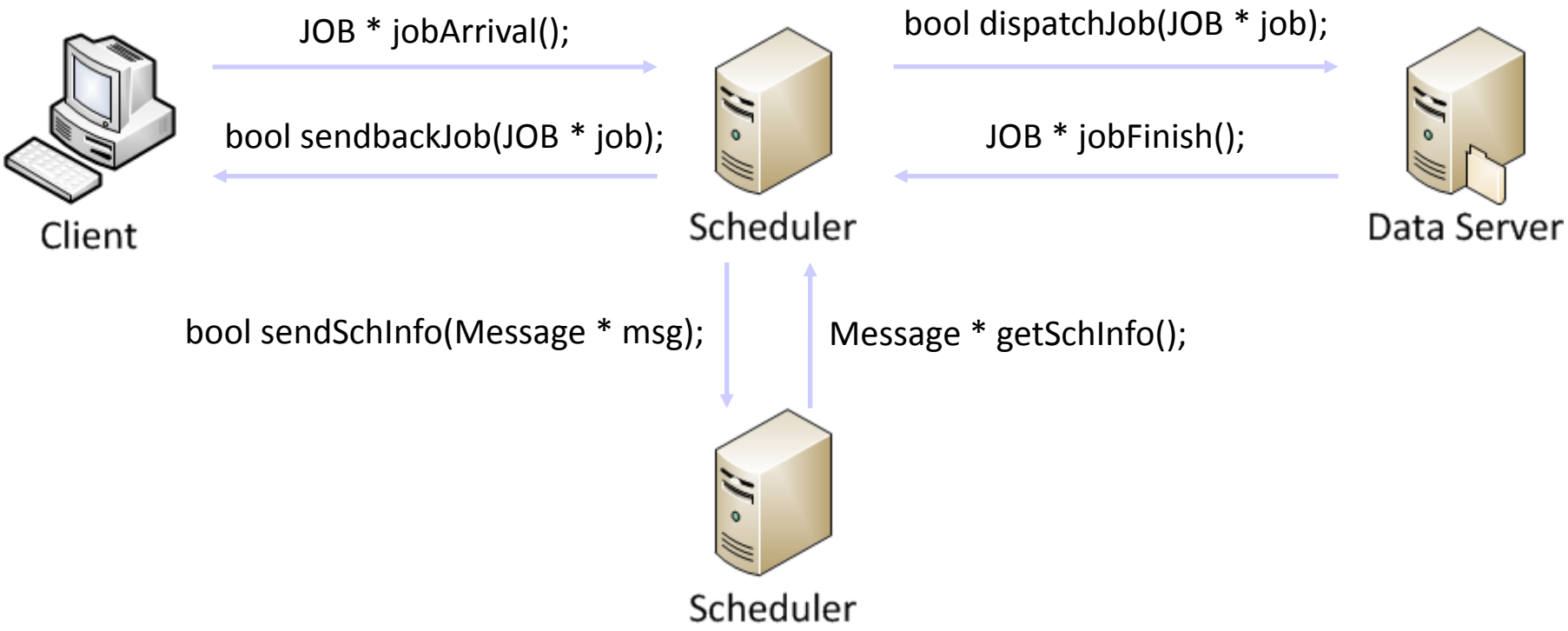
[5] "Building a Parallel File System Simulator," SciDAC'09.
[6] "Using Server-to-server Communication in Parallel File System," SC'08.
[7] "SIMCAN: A Simulator Framework for Computer Architectures and Storage Networks," OMNeT++'08.

Architecture of a Simulated System



Scheduler Implementation

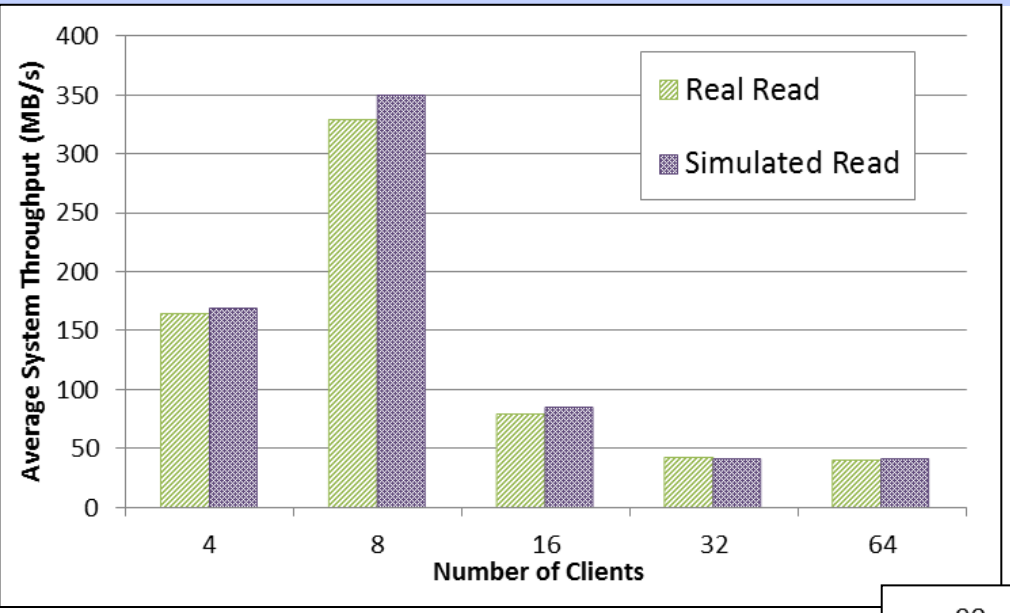
- Schedulers implemented by inheriting a base class
  - With the following essential methods defined:

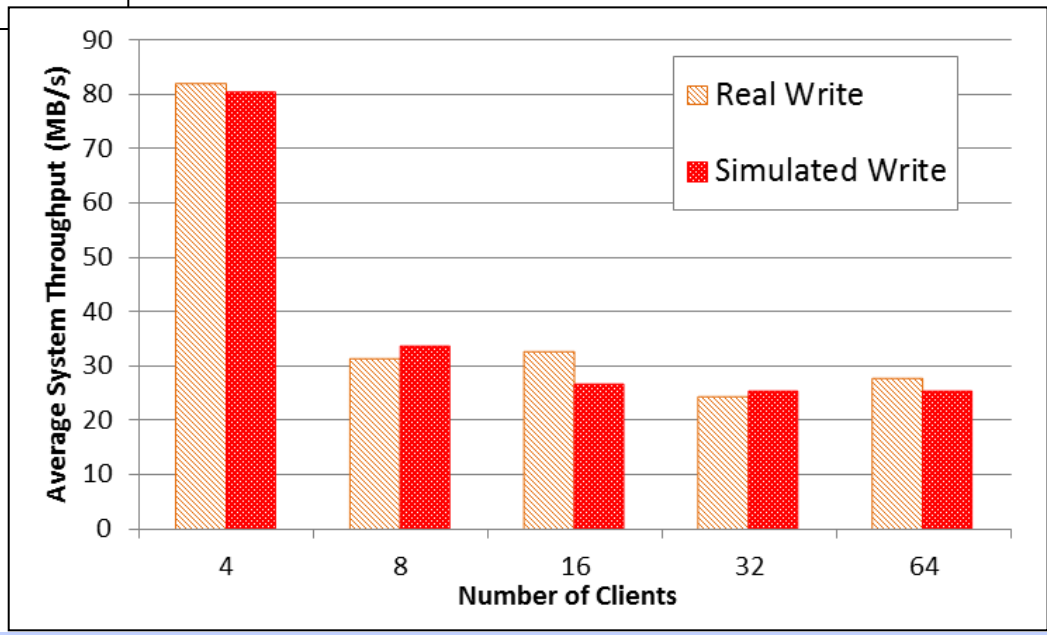


PFSsim Validation

- Validate the I/O throughput and latency under different workloads
- Benchmark system
  - 4 data servers/1 metadata server/varying number of clients
  - Each client/server has 2.4GHz AMD CPU/1GB RAM
  - PVFS2, stripe size set to 256KB, round-robin distribution
- Traces
  - Each client sequentially writes 400MB, 1MB per write
  - Each client sequentially reads 400MB, 1MB per read
  - Reads are conducted on the same files right after write

Read/Write Throughput





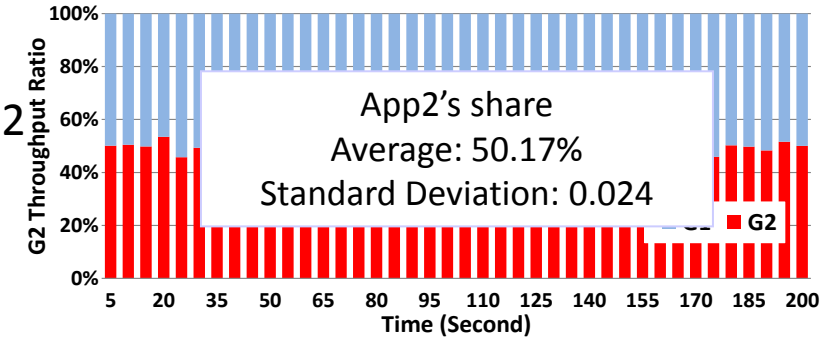
Scheduler Validation

- Validate SFQ(D) algorithm with different sharing ratios
- Benchmark system and traces
  - 4 data servers / 1 metadata server
  - 16 clients in App1 / 16 clients in App2
  - SFQ(D) is deployed on each scheduler (D=4)
  - One scheduler per data server
  - Each client sequentially writes to 400MB, 1MB per write
- Varying sharing ratio between App1 and App2

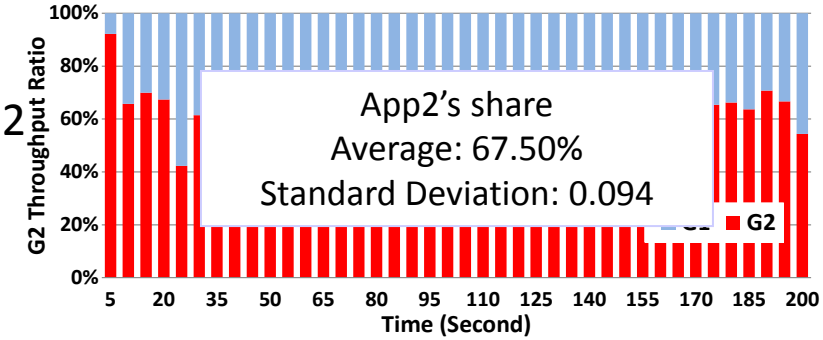
Scheduler Validation

Real System Results

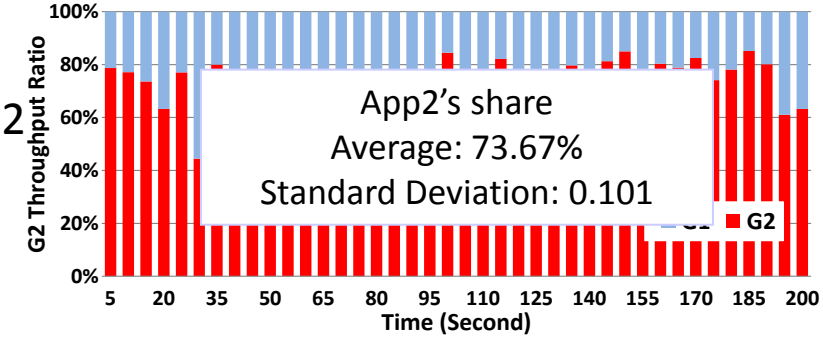
App1:App2 =1:1



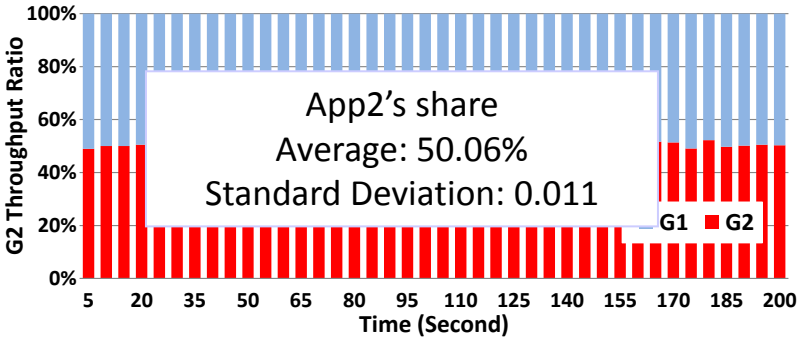
App1:App2 =1:2

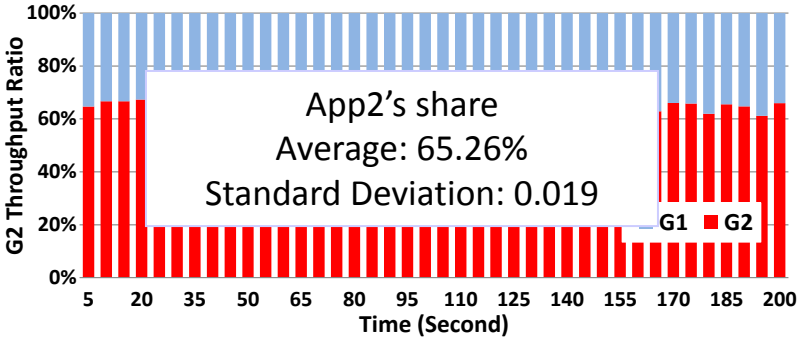


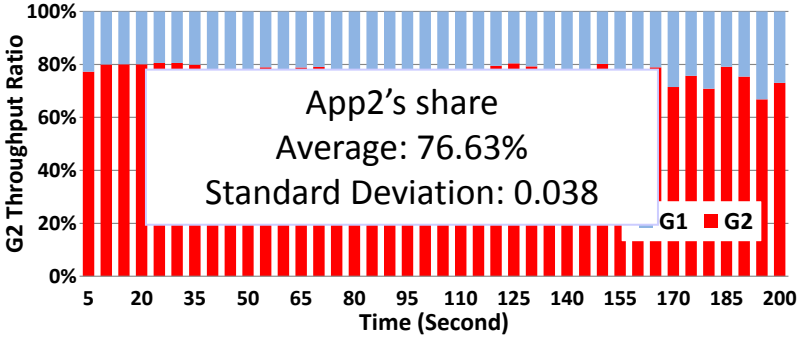
App1:App2 =1:4



Simulated Results







Conclusions and Future Work

- Virtualization-based parallel storage management
  - Overhead is small in terms of throughput & resource usage
  - Can achieve good proportional bandwidth sharing
  - TODO:
    - Implement more scalable global schedule synchronization
    - Consider other scheduling objectives
- Simulation-based parallel I/O scheduling study
  - Support fast, flexible scheduler implementation & evaluation
  - Can achieve good accuracy in terms of throughput & latency
  - TODO:
    - Validate at larger scale
    - Study new algorithms considering both long- and short-term fairness

Acknowledgement

- FIU VISA lab
  - Dulcardo Clavijo, Yiqi Xu, Ming Zhao
- UF ACIS lab
  - Yonggang Liu, Renato Figueiredo
- Industry collaborator:
  - Seetharami Seelam (IBM T.J. Watson)

- Sponsor:
  - NSF HECURA CCF-0937973/CCF-0938045

- More information: http://visa.cs.fiu.edu/hecura
  - Software source code & appliances available for download