Virtualization-based Bandwidth Management for Parallel Storage Systems

Yiqi Xu, Lixi Wang, Dulcardo Arteaga, Dr. Ming Zhao

School of Computing and Information Sciences

Florida International University, Miami, FL

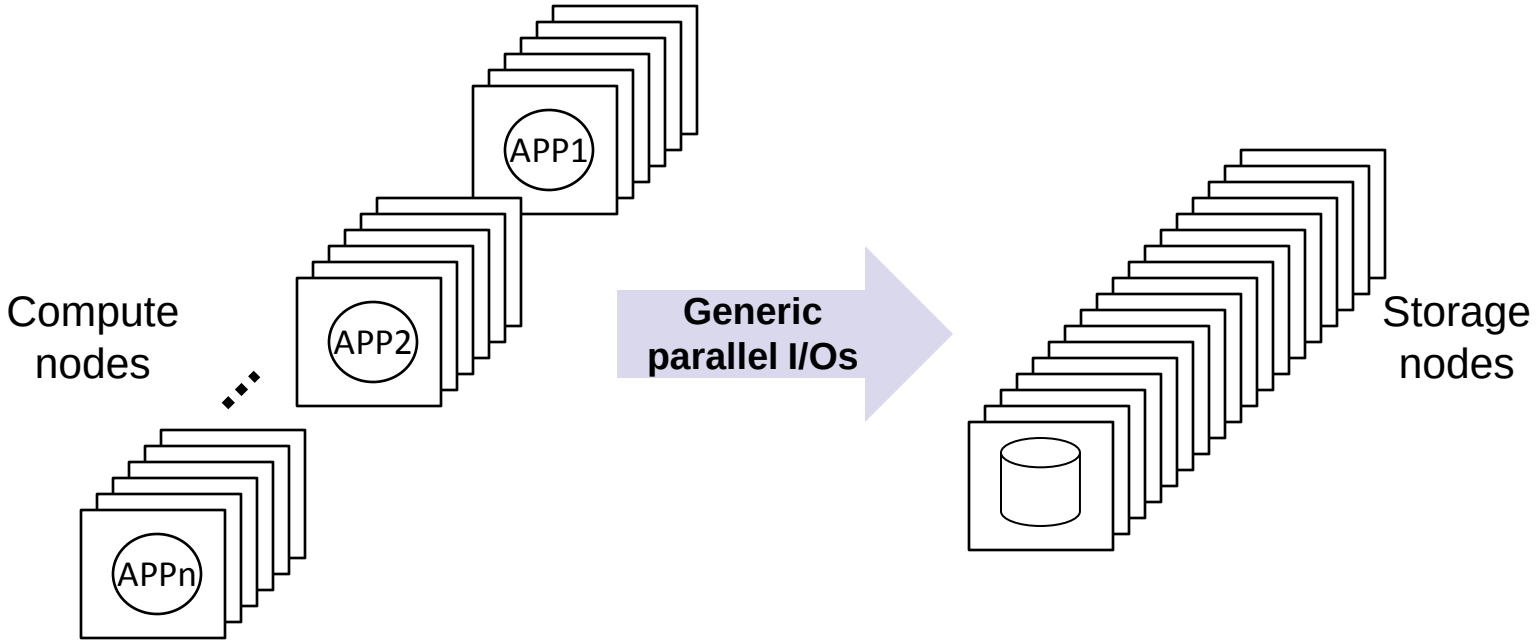
Yonggang Liu, Dr. Renato Figueiredo

Department of Electrical and Computer Engineering

University of Florida, Gainesville, FL

Motivation

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

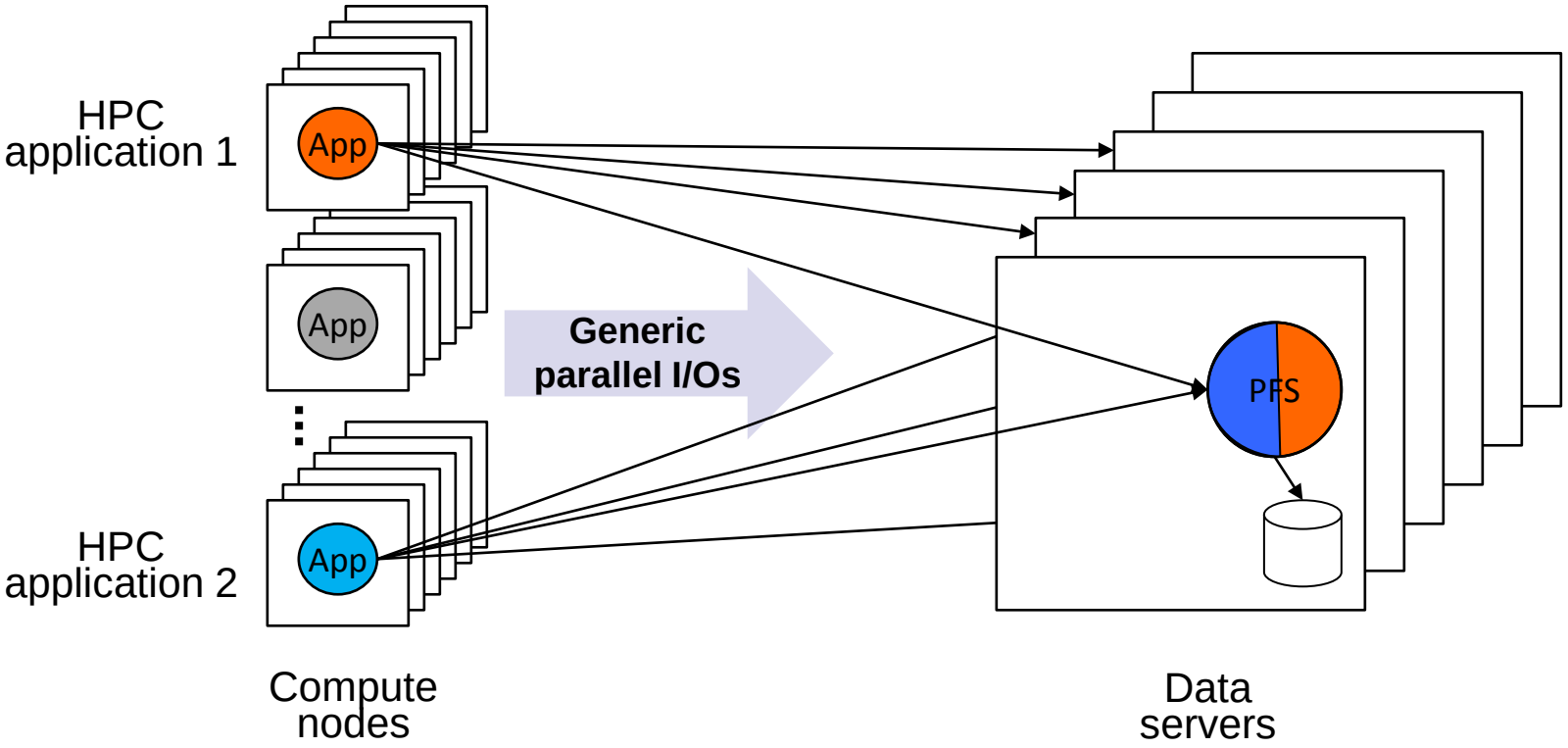
Objective

- Problem: Lack of per-application I/O bandwidth allocation
  - Static partition of storage nodes is inflexible
  - Compute nodes based partition is insufficient
- Proposed Solution: Per-application storage resource allocation
  - Parallel file system (PFS) virtualization
  - Per-application virtual PFSes

Outline

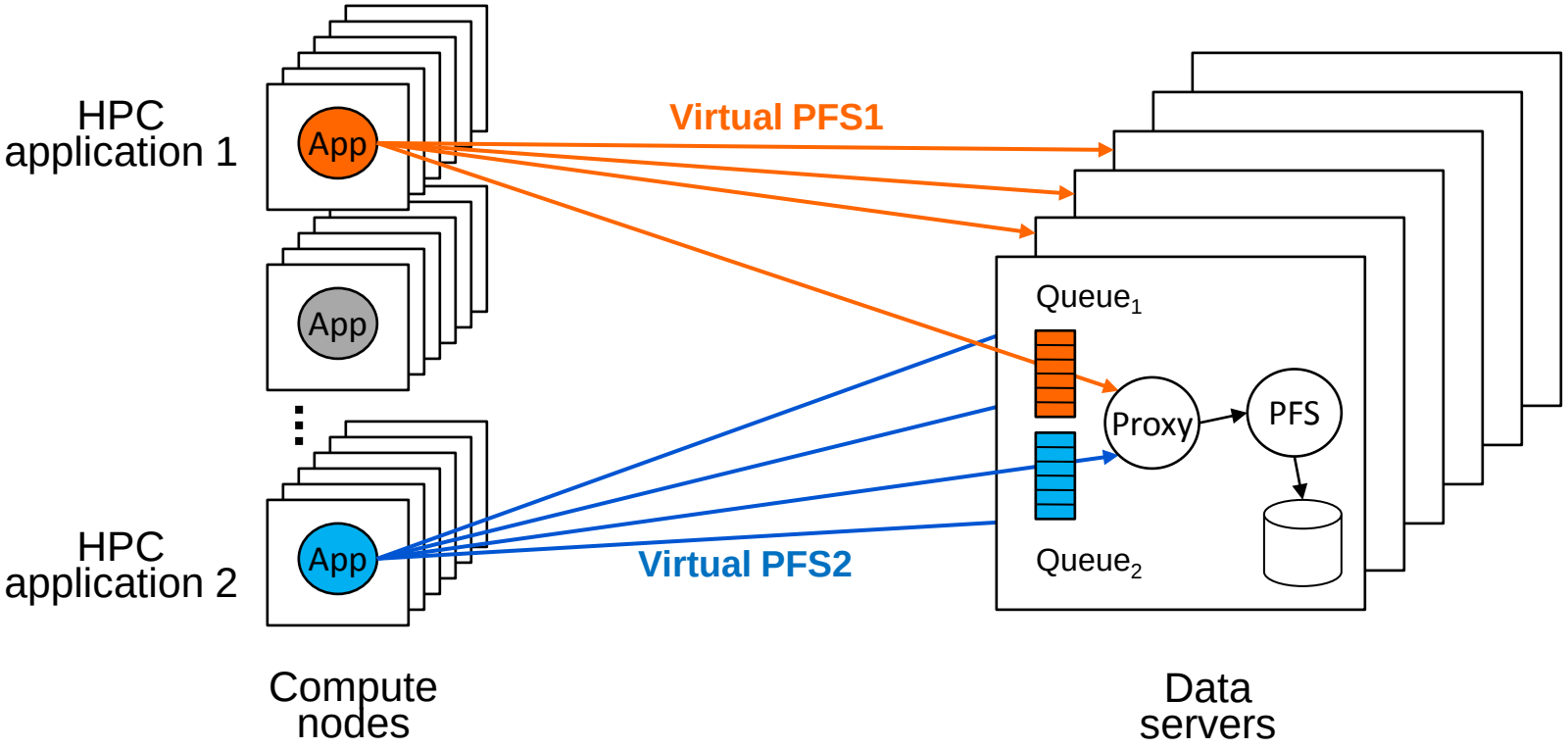
- Background
- Design
- Implementation
- Evaluation

Proxy-based PFS Virtualization (Before)



- Storage nodes are shared without any isolation
- No distinction of I/Os from different applications
- Lack of native scheduling support for bandwidth allocation

Proxy-based PFS Virtualization (After)



- Indirection of application I/O access
- Creation of per-application virtual PFS
- Dynamically spawned on the server

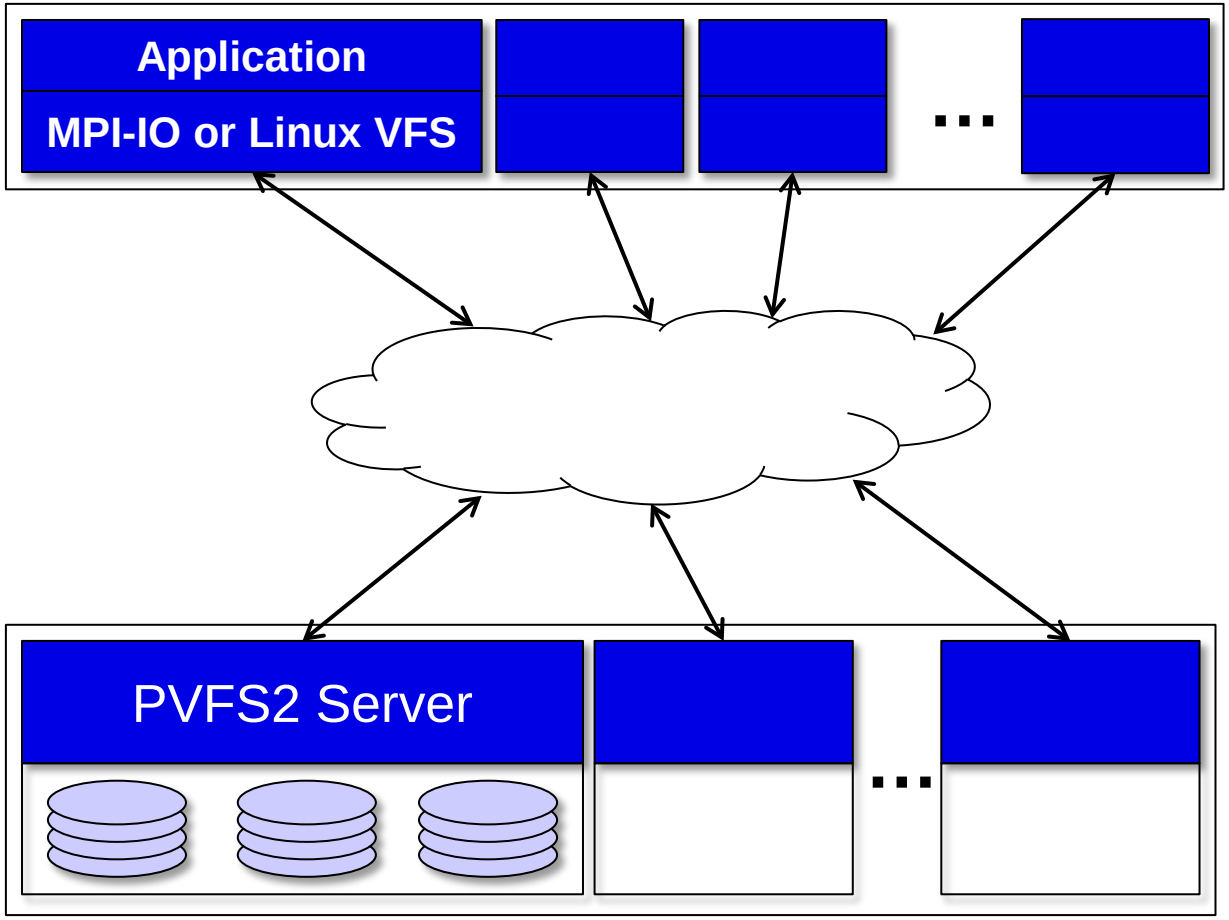
Virtualization Benefits and Costs

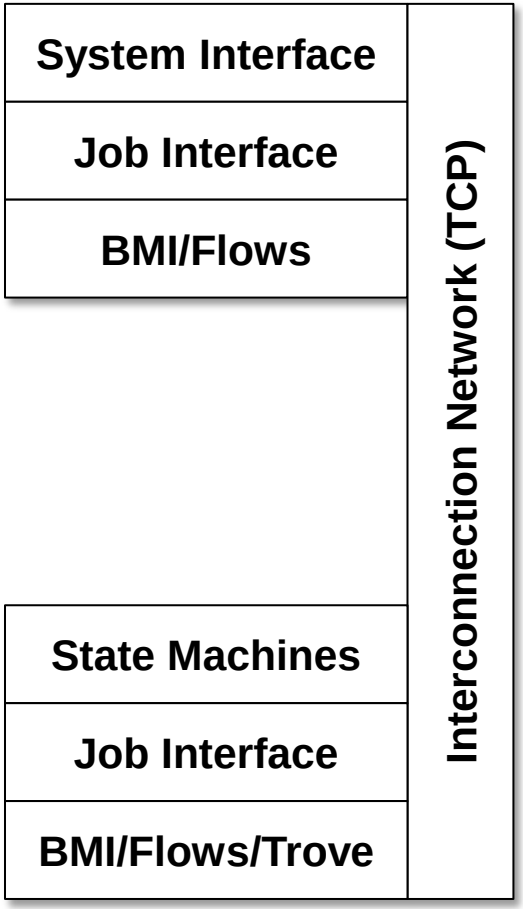
- Benefits
  - Enable various scheduling algorithms
    - SFQ(D) – proportional sharing algorithm
    - EDF – deadline based scheduling
  - Transparent to the existing PFSes
    - No change in existing implementation needed
    - Support different parallel storage systems
- Costs
  - Overhead involved in user-level proxy
  - Extra processing of data and communication

Prototype Implementation

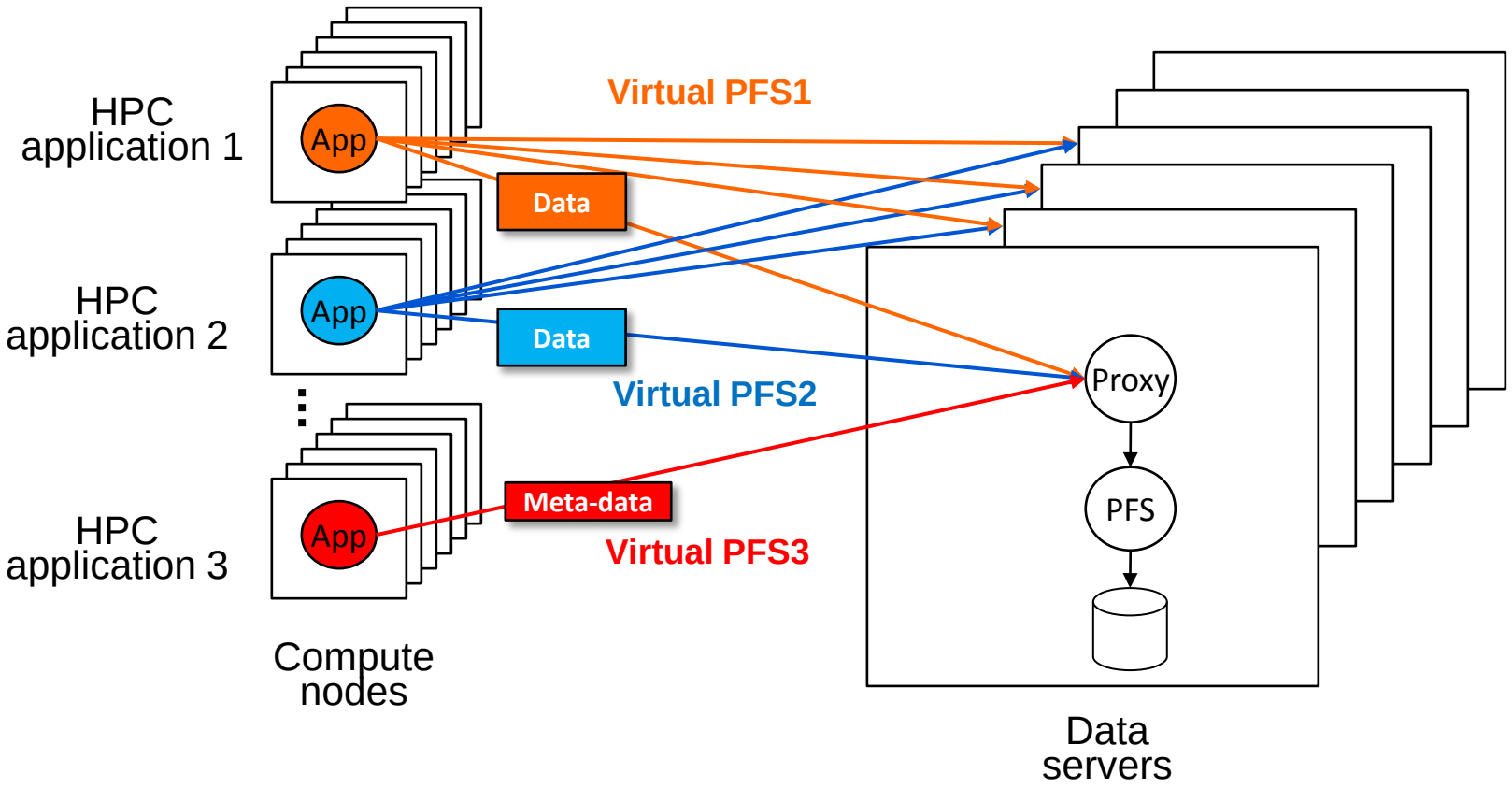
- A PVFS 2.8.2 (Parallel Virtual File System) proxy
  - Deployed on every data server
  - Intercepts and forwards PVFS2 messages
  - Asynchronous I/O for less overhead
  - Identifies I/Os from different applications
  - Dynamically configured by a configuration file
- Proxy implements SFQ(D) scheduling
  - Supports a generic scheduling interface for other algorithms

PVFS2 Background





PVFS2 Proxy



- Non-I/O messages are not scheduled
- Extra processing for I/O messages at proxy side

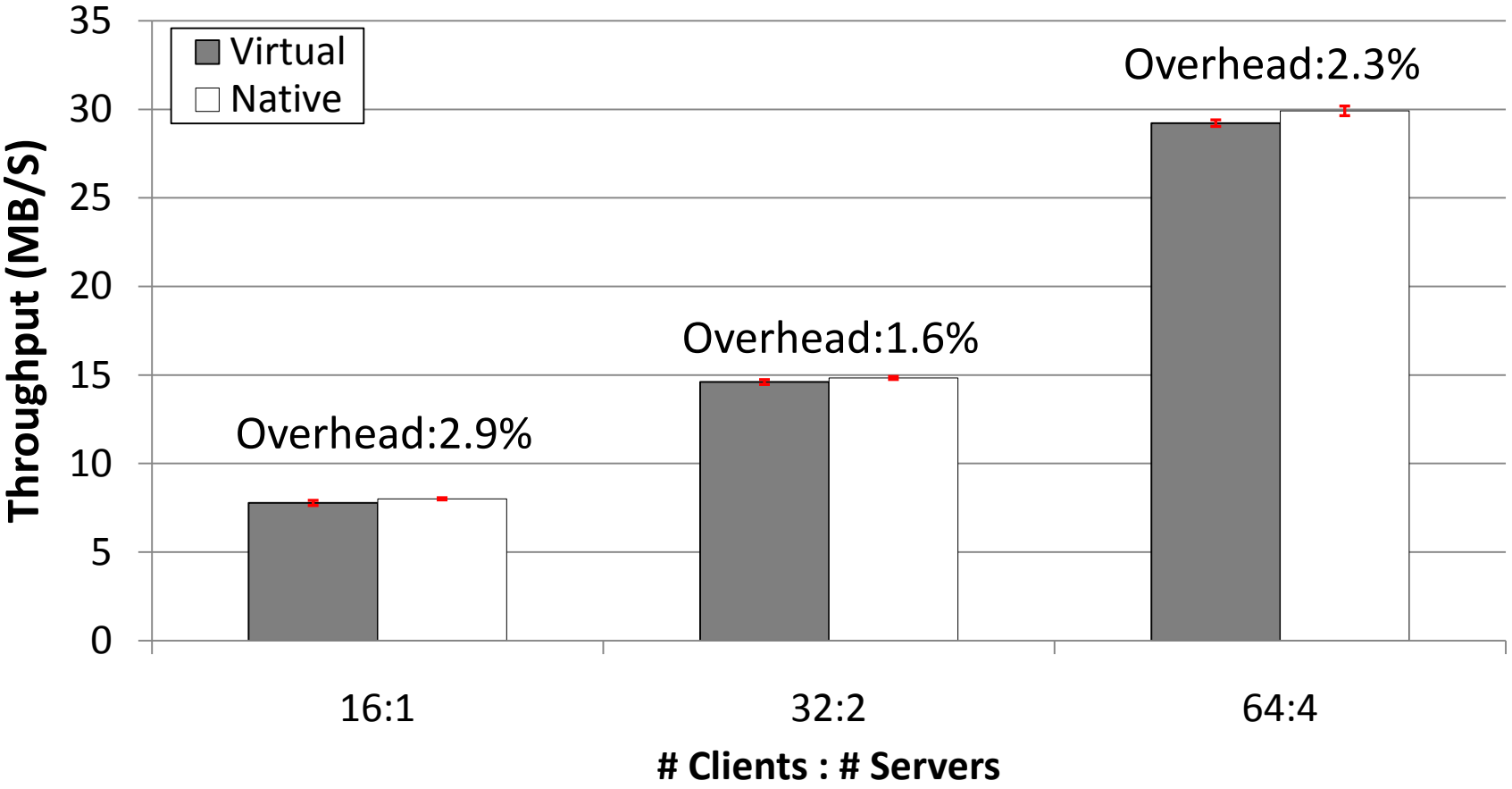
SFQ – Start Time Fair Queuing

- SFQ
  - Originally designed for network packet scheduling
  - Work-conserving, proportional sharing-based scheduling
  - Adapts to variation in server capacity
- SFQ(D)
  - Extended from SFQ
  - Adds depth to allow and control concurrency
  - Multiplexes storage bandwidth and enhance utilization

Evaluation

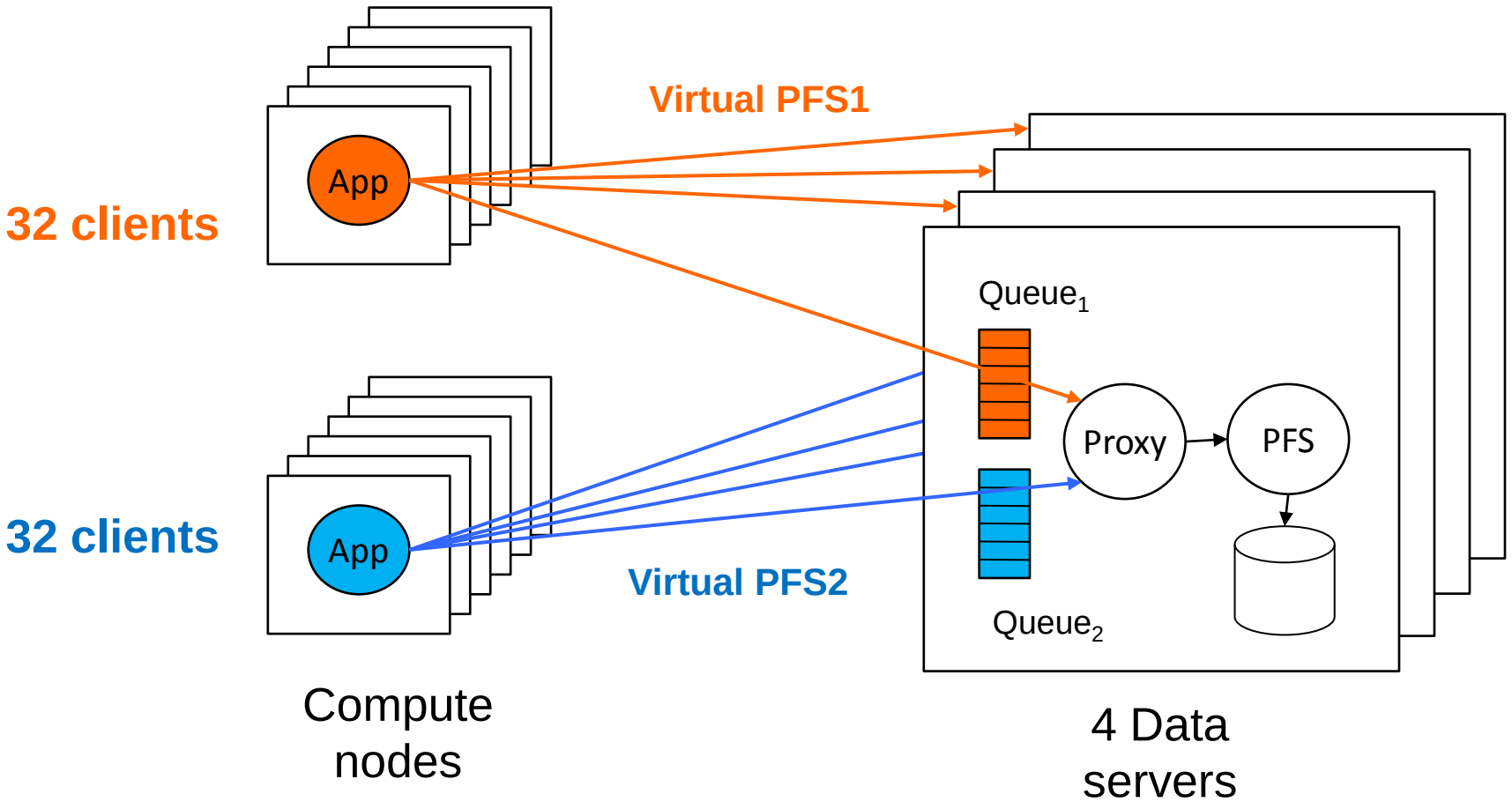
- A virtual machine based testbed
  - A cluster of 8 DELL PowerEdge 2970 servers
  - Ubuntu 8.04, Kernel 2.6.18.8
  - Up to 64 PVFS2 clients and 4 PVFS2 servers (version 2.8.2)
  - 2 competing parallel applications
- Benchmark: IOR version 2.10.3 with sequential writes
- Proxy implements performance monitoring
- Experiments:
  - Virtualization overhead
  - Effectiveness of proportional sharing

Virtualization Overhead

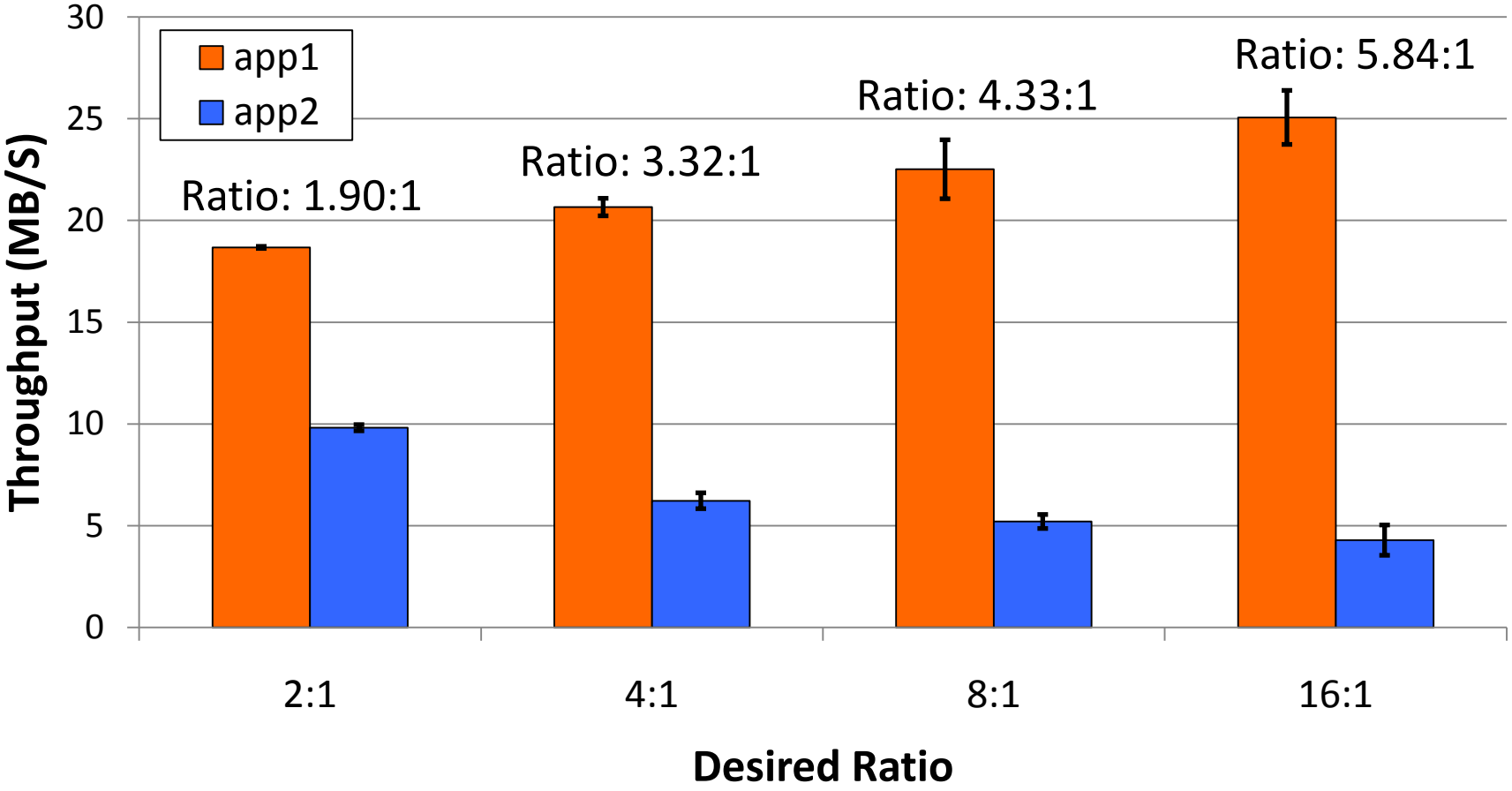


- Throughput overhead is small
- About 3% Proxy CPU and 1MB RAM usage on each node

Proportional Sharing in a Symmetric Setup

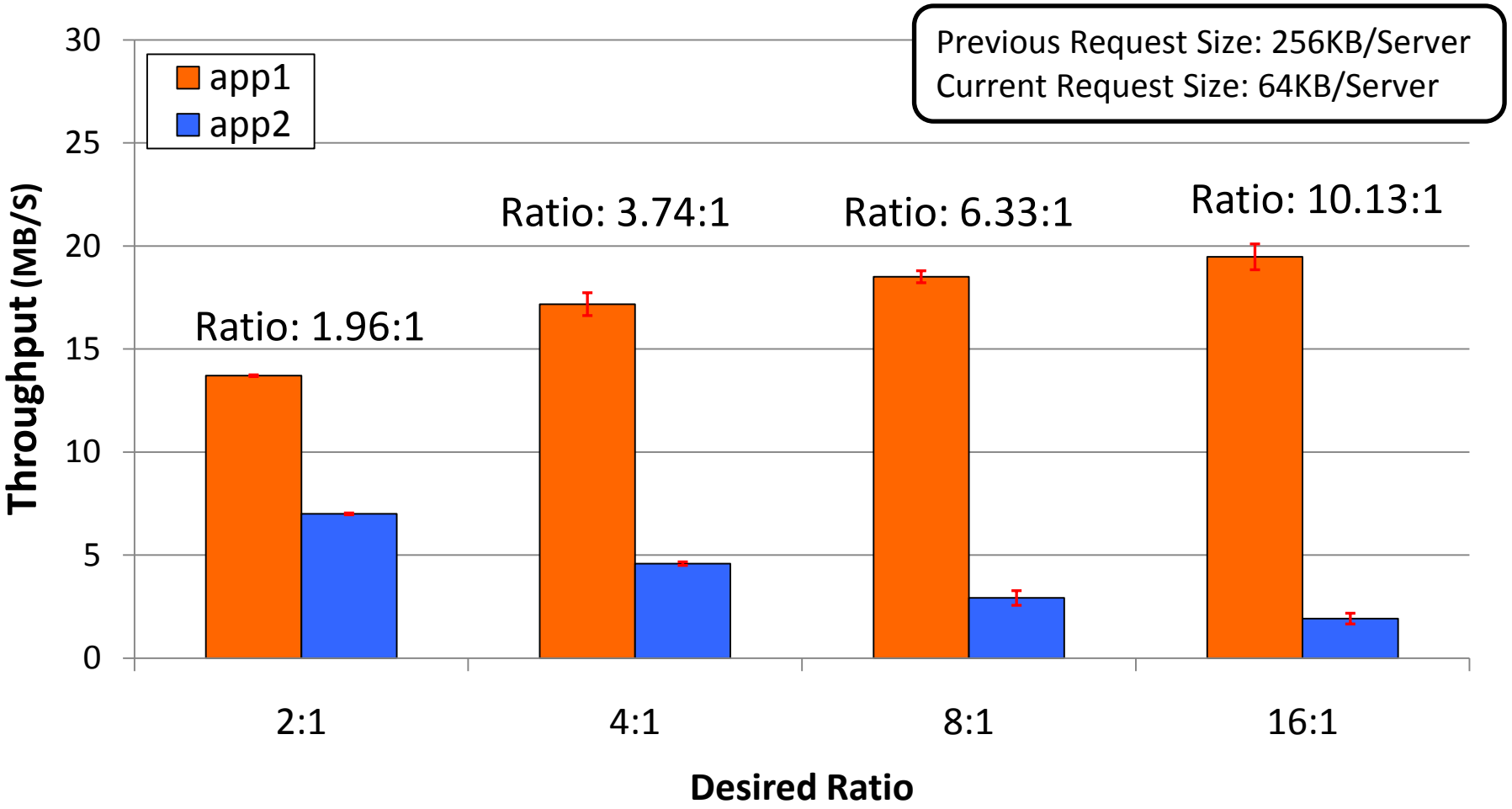


Proportional Sharing with Varying Ratios



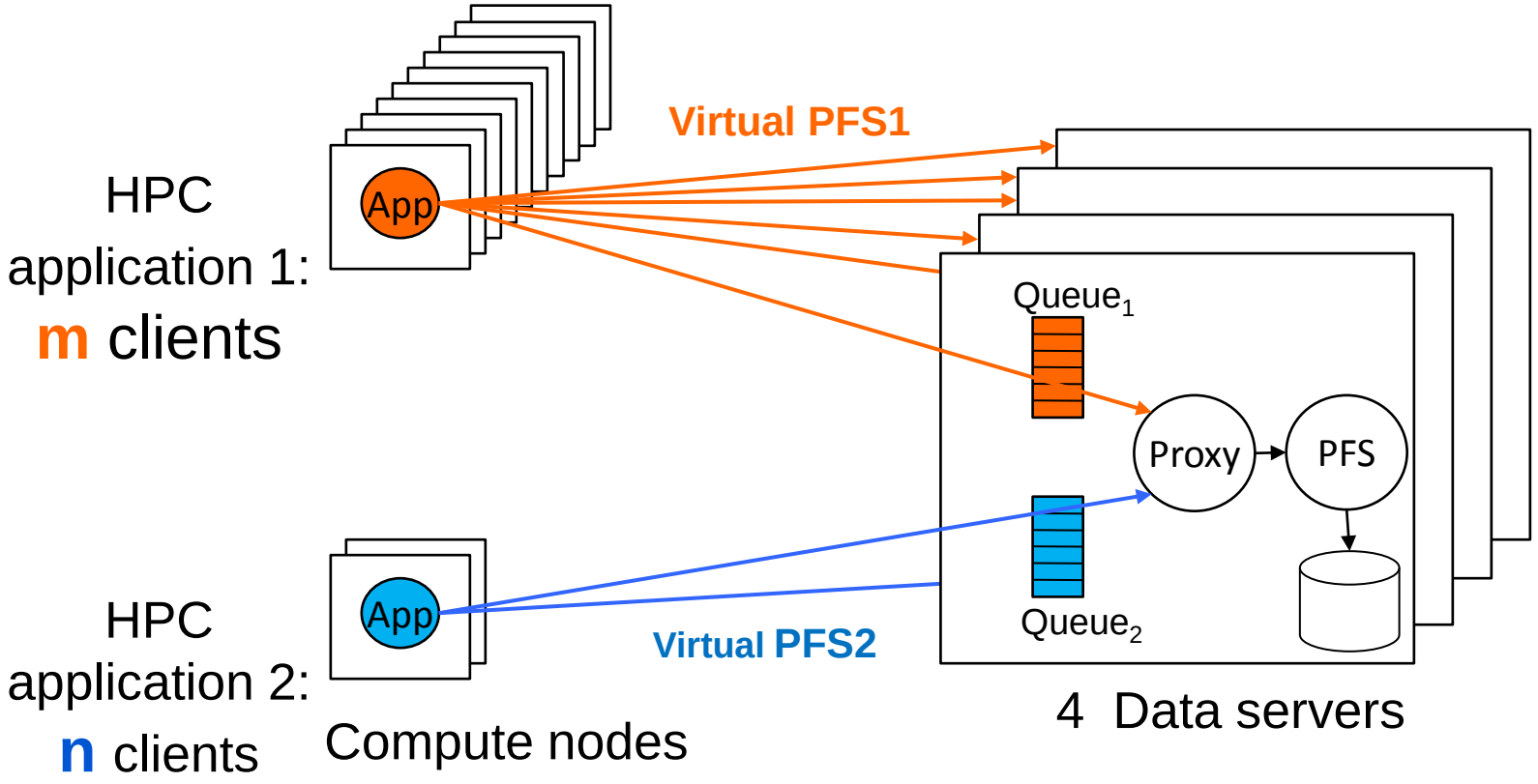
- Good proportional sharing can be achieved
- The actual ratio drops when the desired ratio is high

Proportional Sharing with Smaller I/Os

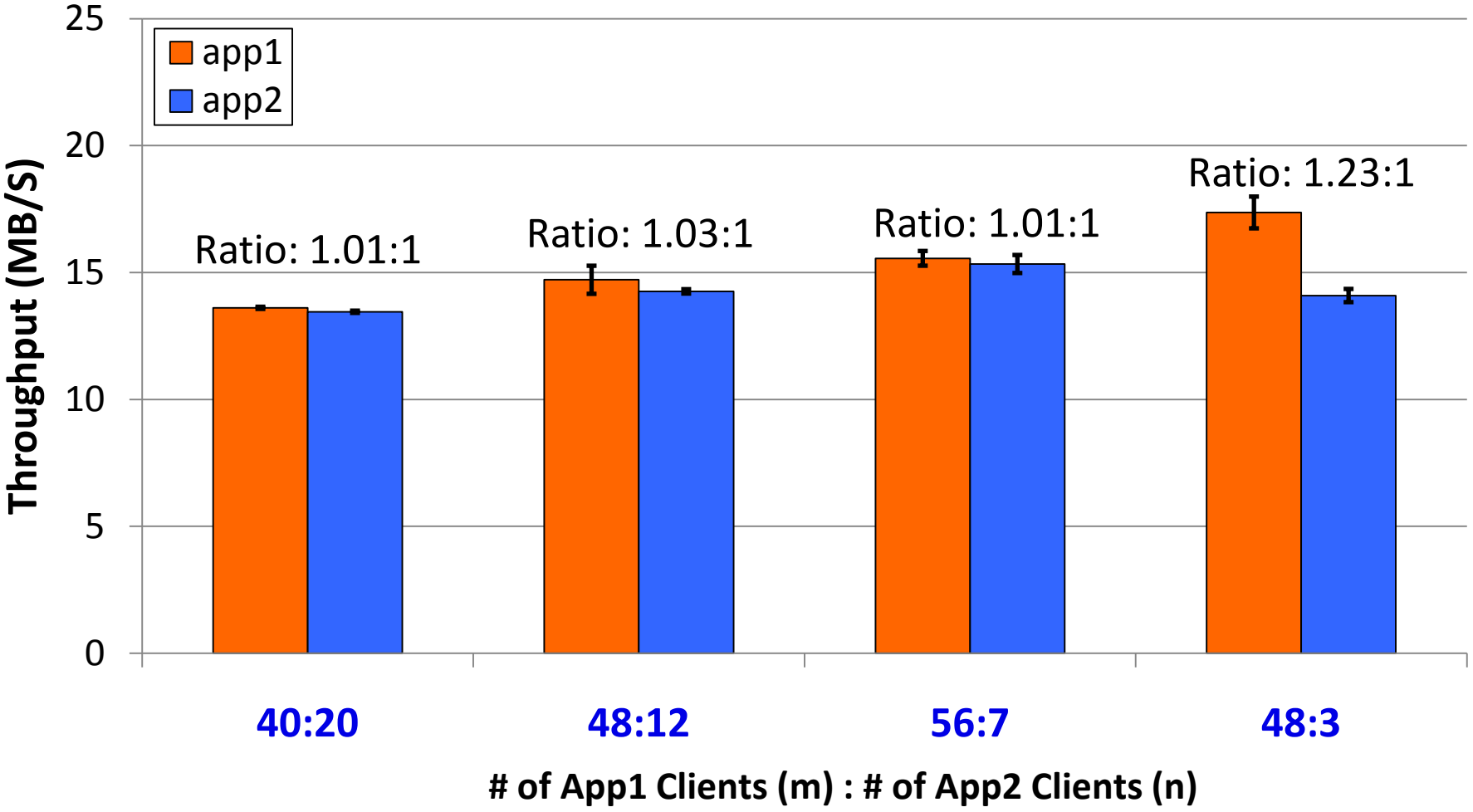


- Increasing request rate improves the actual ratio
- Can be further improved by increasing number of clients

Proportional Sharing in an Asymmetric Setup



Proportional Sharing in an Asymmetric Setup



- Almost perfect fairness can be achieved

Conclusions and Future Work

- Conclusions
  - Proxy-based PFS virtualization is feasible with negligible overhead
  - Effective proportional bandwidth sharing using SFQ(D)
- Future work
  - More scheduling algorithms
    - Global proportional sharing
    - Deadline based scheduling
  - Evaluate on applications' access patterns and I/O requirements with more diversities

References

[1] PVFS2. URL: http://www.pvfs.org/pvfs2/.

[2] P. Goyal, H. M. Vin, and H. Cheng, — Start Time Fair Queuing: A Scheduling Algorithm For Integrated Services Packet Switching Networks, IEEE/ACM Trans. Networking, vol. 5, no. 5, 1997.

[3] Yin Wang and Arif Merchant. 2007. Proportional-share scheduling for distributed storage systems. In Proceedings of the 5th USENIX conference on File and Storage Technologies (FAST '07). USENIX Association, Berkeley, CA, USA, 4-4.

[4] W. Jin, J. S. Chase, and J. Kaur, "Interposed Proportional Sharing For A Storage Service Utility", SIGMETRICS, 2004.

[5] IOR HPC Benchmark, http://sourceforge.net/projects/ior-sio/.

[6] P. Welsh, P. Bogenschutz, —Weather Research and Forecast (WRF) Model: Precipitation Prognostics from the WRF Model during Recent Tropical Cyclones, Interdepartmental Hurricane Conference, 2005.

[7] A. Darling, L. Carey, and W. Feng, —The Design, Implementation, and Evaluation of mpiBLAST, ClusterWorld Conf. and Expo, 2003.

[8] R. Sankaran, et al., —Direct Numerical Simulations of Turbulent Lean Premixed Combustion, Journal of Physics Conference Series, 2006.

Acknowledgement

- Research team
  - VISA lab at FIU
    - Yiqi Xu, Dulcardo Clavijo, Lixi Wang, Dr. Ming Zhao
  - ACIS lab at UF
    - Yonggang Liu, Dr. Renato Figueiredo
  - Industry collaborator
    - Dr. Seetharami Seelam (IBM T.J. Watson)
- Sponsor: NSF HECURA CCF-0937973/CCF-0938045
- More information: http://visa.cis.fiu.edu/hecura

Thank You!