Towards Simulation of Parallel File System Scheduling Algorithms with PFSsim

Yonggang Liu, Renato Figueiredo

University of Florida

Gainesville, FL

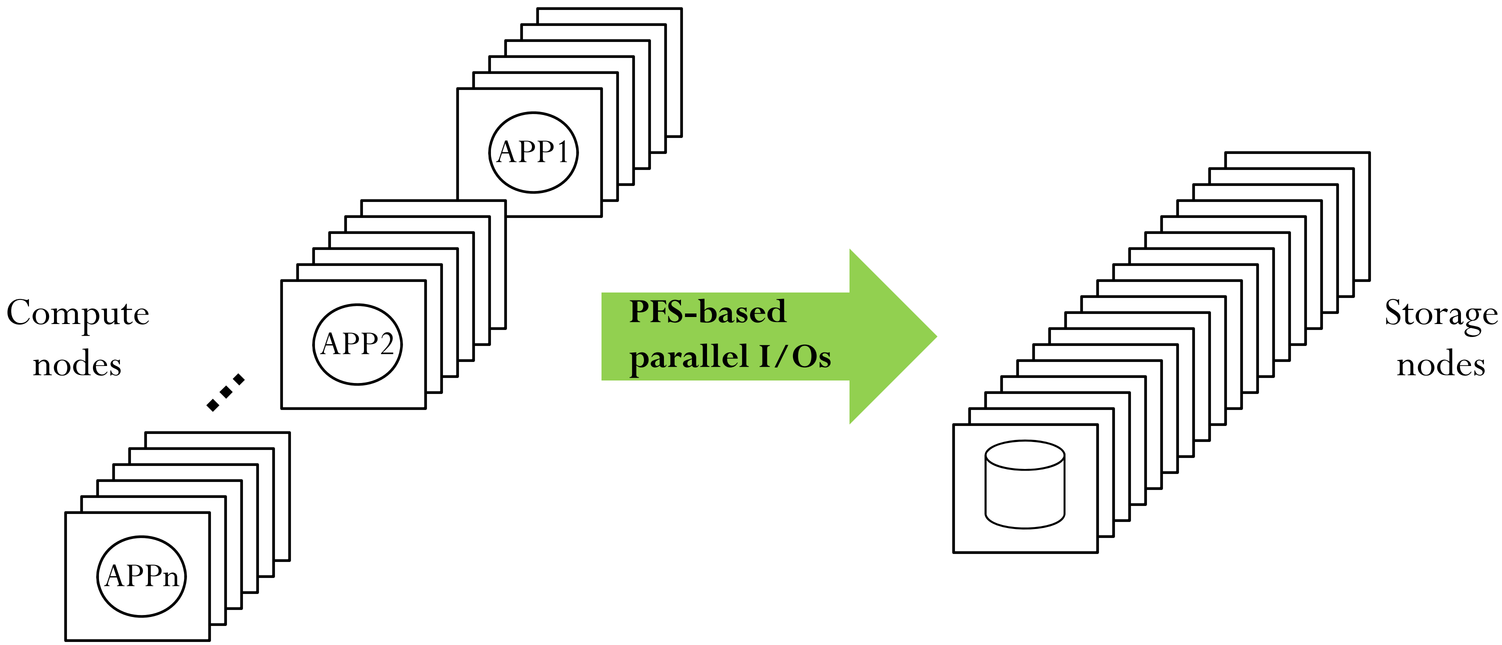
Dulcardo Clavijo, Yiqi Xu, Ming Zhao

Florida International University

Miami, FL

Introduction

- Parallel File Systems (PFSs) based storage
  - Widely used in high-performance computing systems
  - Examples: Lustre, PVFS2, PanFS, GPFS



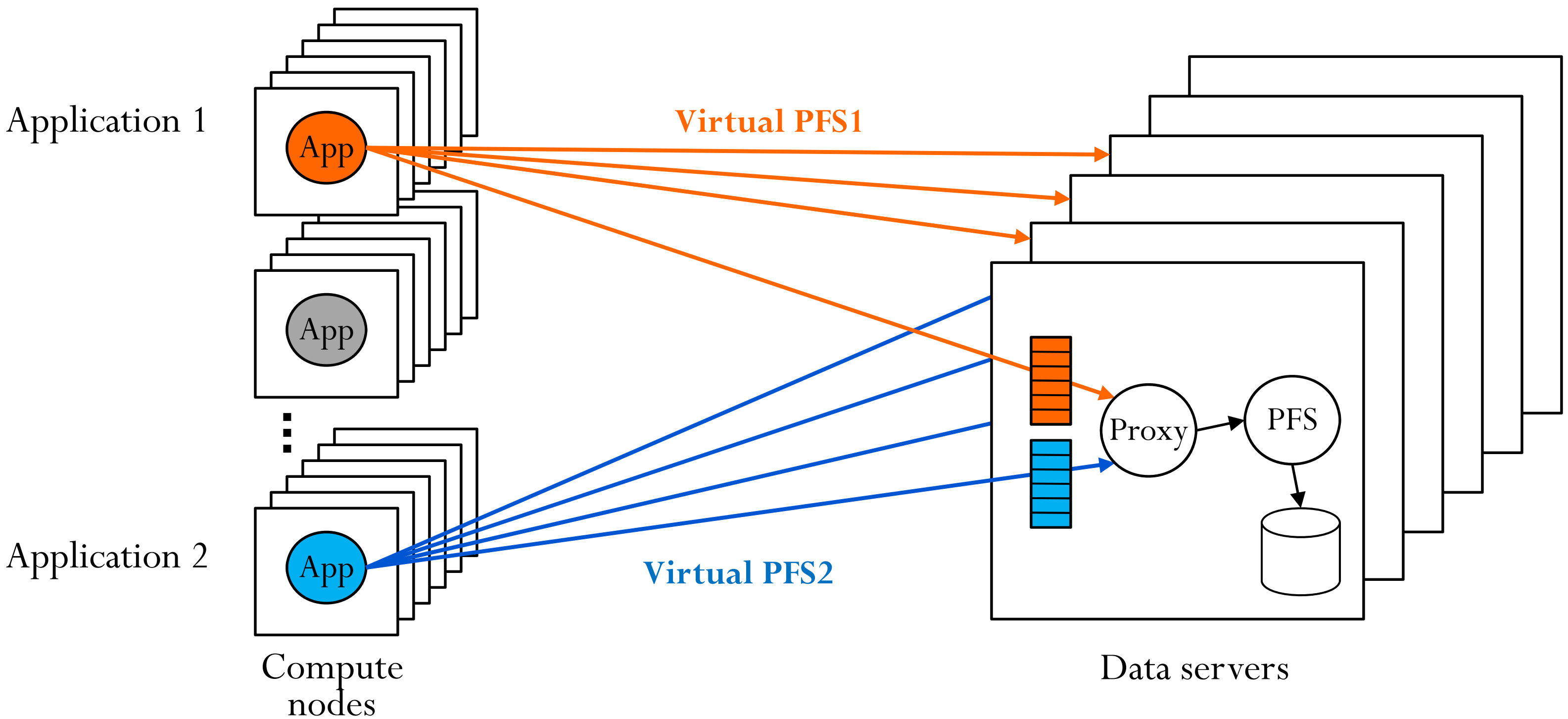
Challenge

- How to provide application-desired quality of service when the system has:
  - Many applications with large amount of I/O traffic
  - Diverse application access patterns
  - Diverse application QoS requirements
  - Examples: WRF, mpiBLAST, S3D

- This problem will only become even more serious as the scale HPC systems further increases

Virtualization-based Storage Management

- Creation of per-application virtual PFS*
- Ability to schedule I/Os on a per-application basis



*Y. Xu, et al., "Virtualization-based Bandwidth Management for Parallel Storage Systems", *PDSW'10*.

PFSsim

- Motivation
  - The need of evaluating parallel I/O scheduling algorithms
  - The need of a general-purpose parallel file system simulation framework

- Design goals
  - Easy-to-use
  - Flexible
  - Accurate
  - Scalable

Related Work

- IMPIOUS* by E. Molina-Estolano, et al.
  - Capable of fast evaluations of PFS designs
  - No simulation of metadata server and metadata operations
- The simulator developed by P. Carns, et al. **
  - Capable of evaluating the performance of I/O communications
  - Detailed simulation of network models
- SIMCAN*** by Alberto Núñez, et al.
  - Modulated design and statistical models
  - Complex system architecture
- No support for I/O scheduling simulations

* "Building a Parallel File System Simulator", *SciDAC'09*.

** "Using Server-to-server Communication in Parallel File System", *SC'08*.

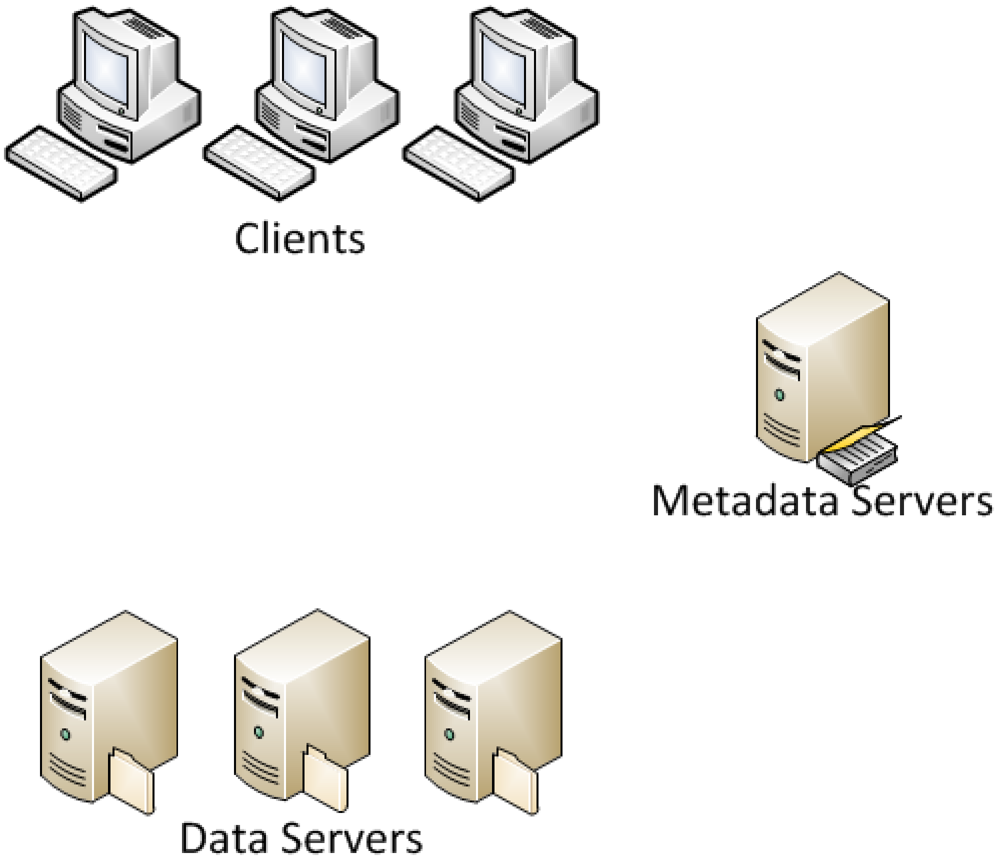
*** "SIMCAN: A Simulator Framework for Computer Architectures and Storage Networks", *OMNeT++'08*.

Outline

- Introduction
- Related Work
- Design and Implementation
- Validation and Evaluation
- Conclusion
- Future Work

PFSsim: Abstraction of PFSs

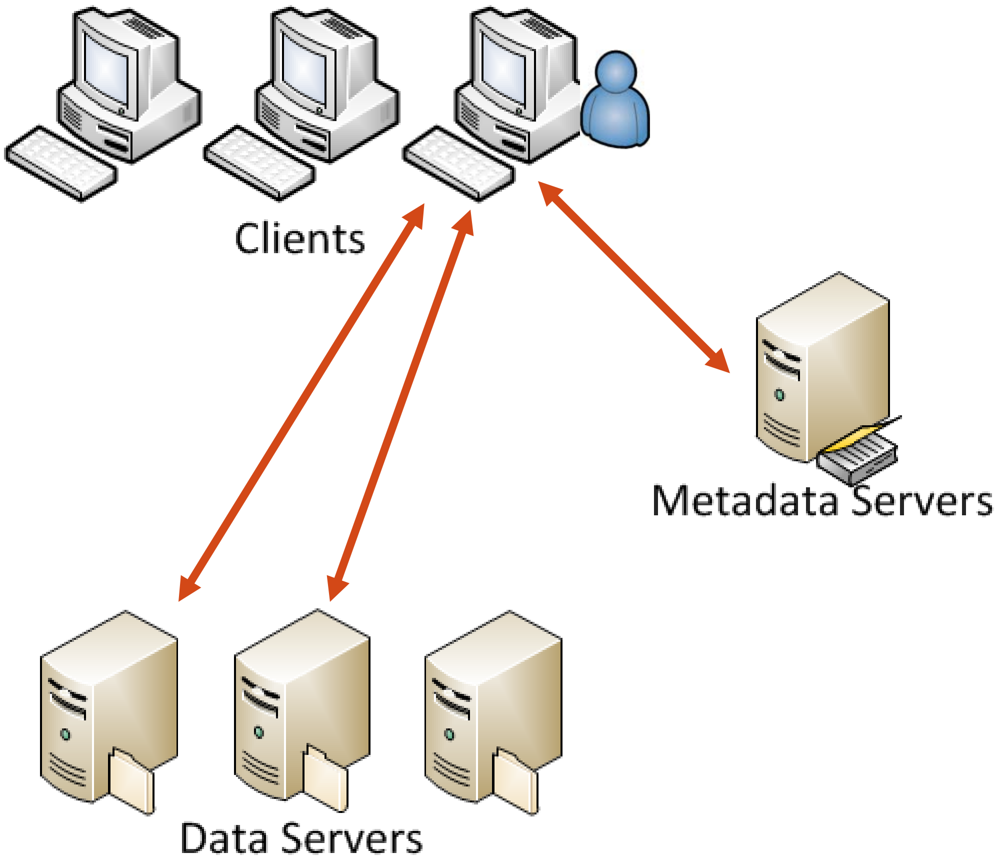
- Essential components and their functionalities



- Data Servers
  - Built based on the local file systems /block devices
  - Store application data in fixed-sized objects

Abstraction of PFSs

- A typical file data access (read/write) operation



- Application I/O request
  - {op, file_path, off, size} to the client

Four Important Aspects

- Metadata management
  - Can significantly impact application performance*
- Data placement strategy
  - Determine server load balance and I/O parallelism
- Data replication model
  - Writes can be slower due to updating multiple copies
- Data caching policy
  - Generally speed up data access, but consistency management also incurs overhead

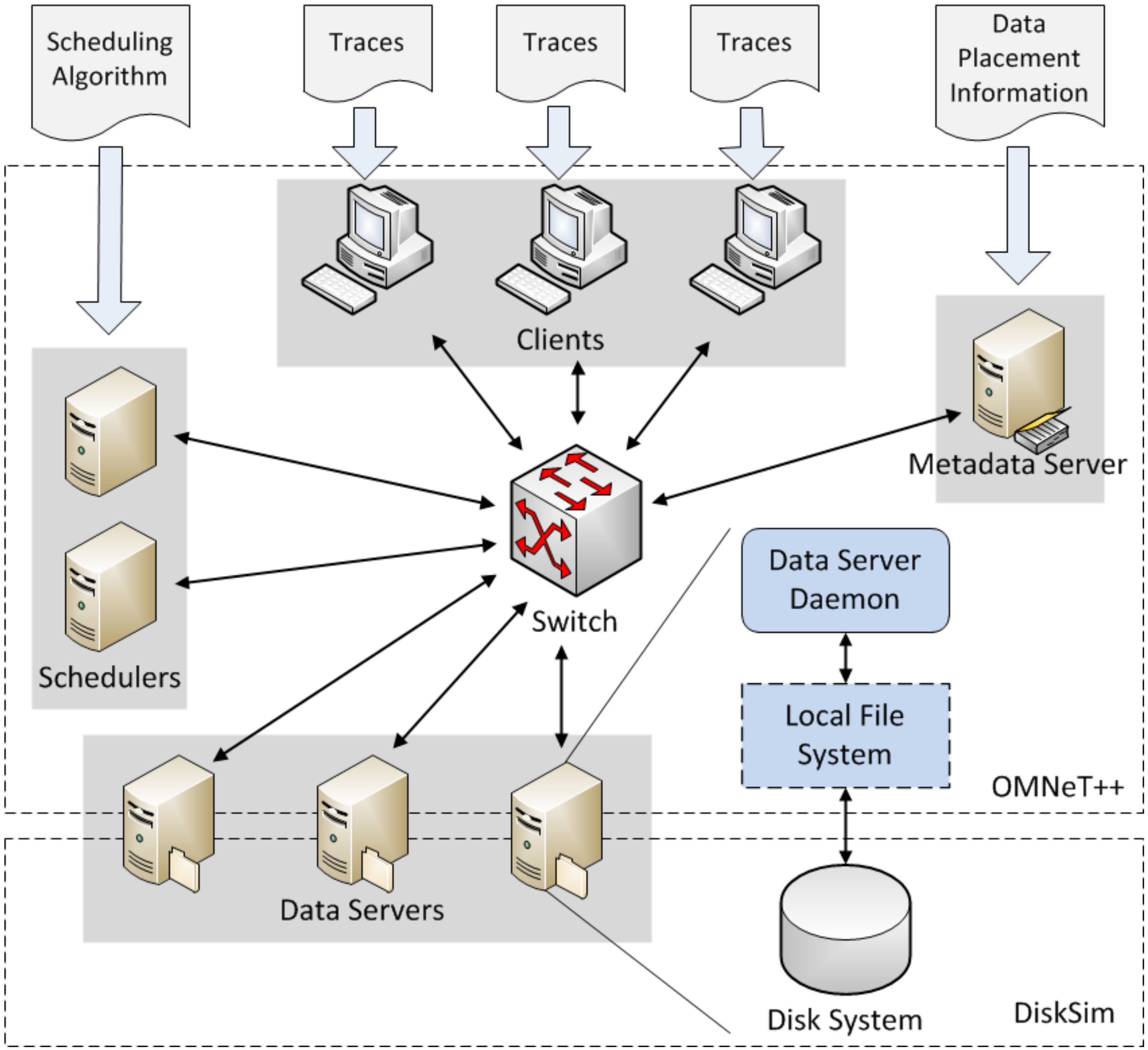
* R. Oldfield, et al. "Modeling the Impact of Checkpoints on Next-Generation Systems", MSST'07.

Abstraction of PFS Schedulers

- Schedulers in storage systems are deployed in different ways:
  - On the gateways/proxies/data servers
  - Centralized/decentralized

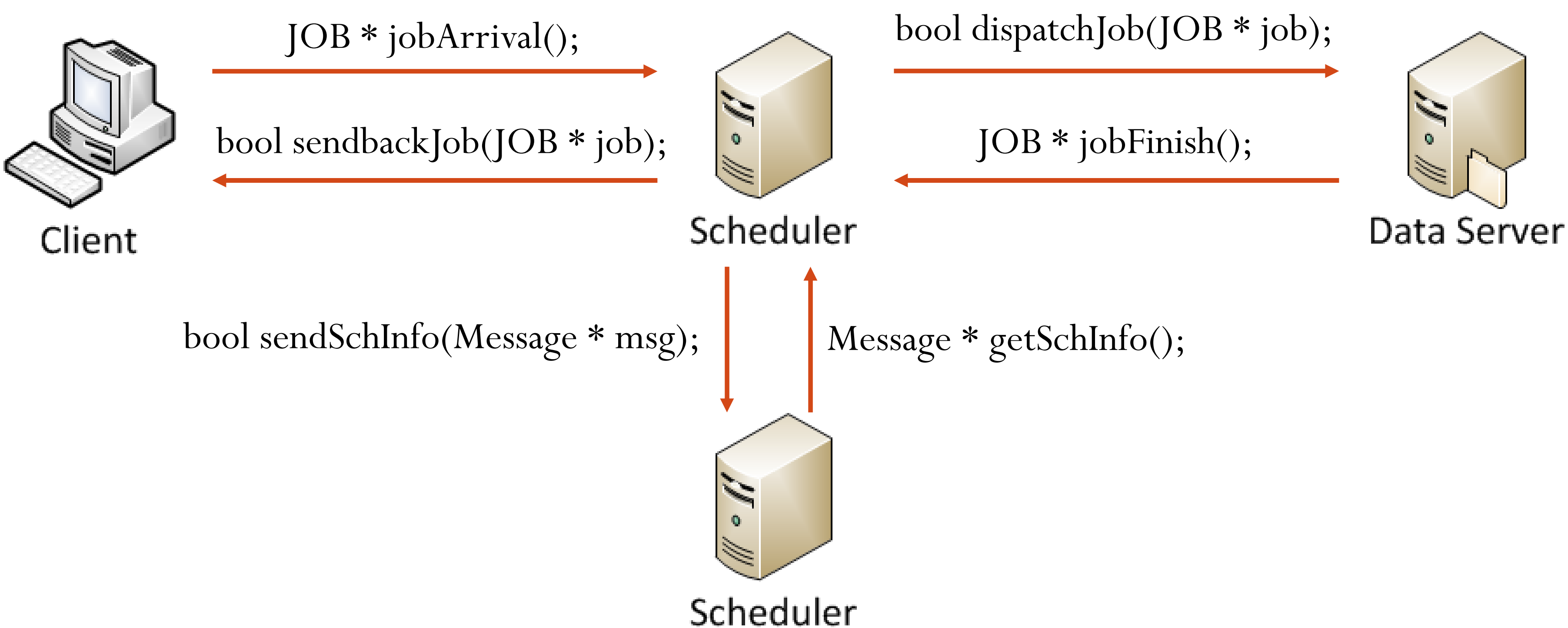
- In PFSsim, the schedulers can be modeled flexibily:
  - Stand-alone/coupled with the network entities
  - Inter-scheduler communications are supported

Architecture of a Simulated System



Scheduler Implementation

- The schedulers are implemented by inheriting a base class with several essential methods:



Network Implementation

- Network links are simulated by the channel components in OMNeT++
  - Configurable bandwidth/latency/bit error rate
- Detailed real-world network protocols are omitted
  - Can be extended with the INET framework*
- Basic wired network devices are simulated
  - Such as switches, routers
  - Can be customized or extended by users

* "http://inet.omnetpp.org"

Local File System Implementation

- Memory component is simulated for data caching/buffering
  - Configurable memory size and page replacement policies

- Files are mapped to disk blocks in a contiguous manner
  - Real-world disk block management schemes are hard to simulate, dependent on many factors (file system, file size and disk usage)*
  - Possible simulation using statistical models in future work

* A. Nunez, "New Techniques for Modeling File Data Distribution on Storage Nodes"

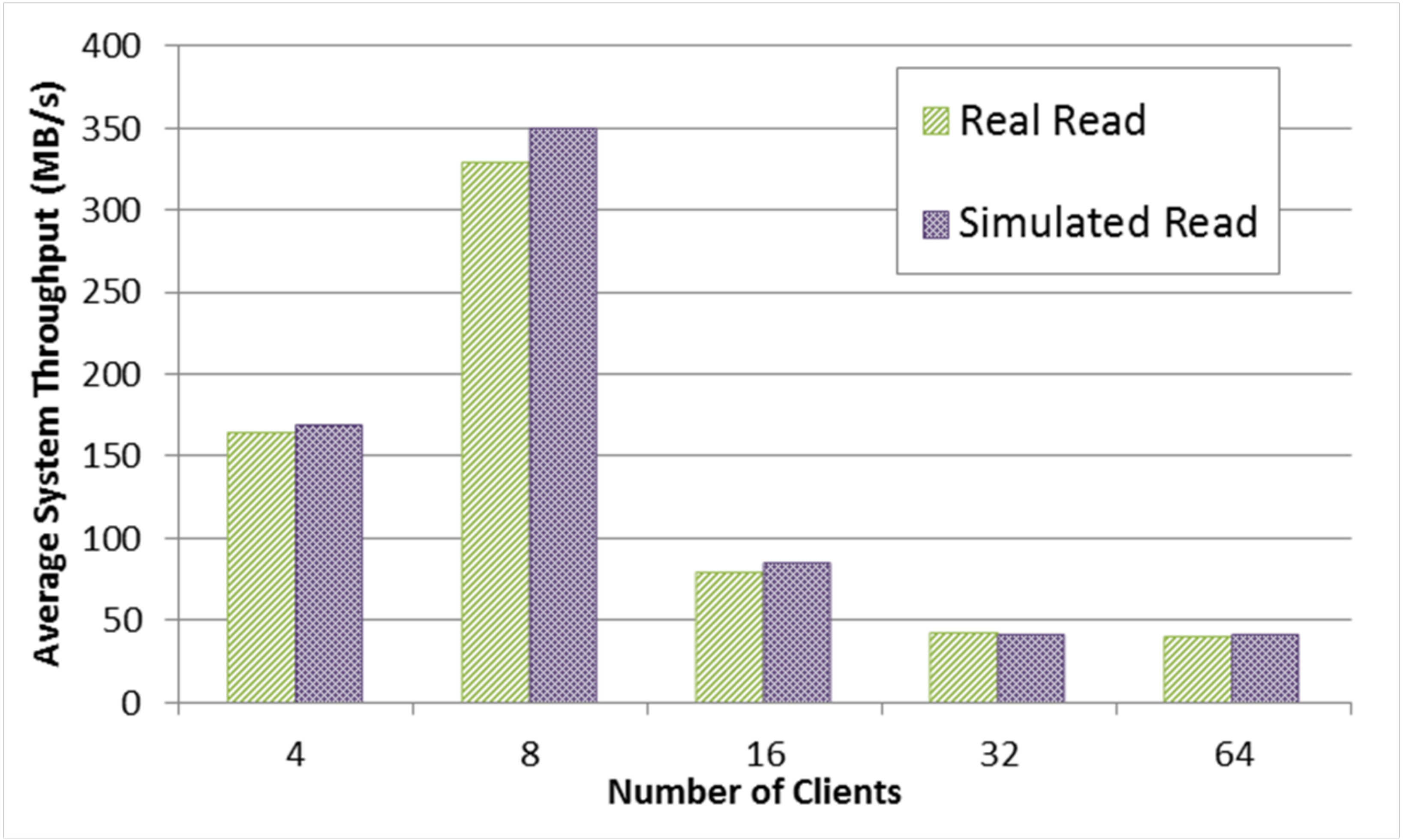
Outline

- Introduction
- Related Work
- Design and Implementation
- Validation and Evaluation
- Conclusion
- Future Work

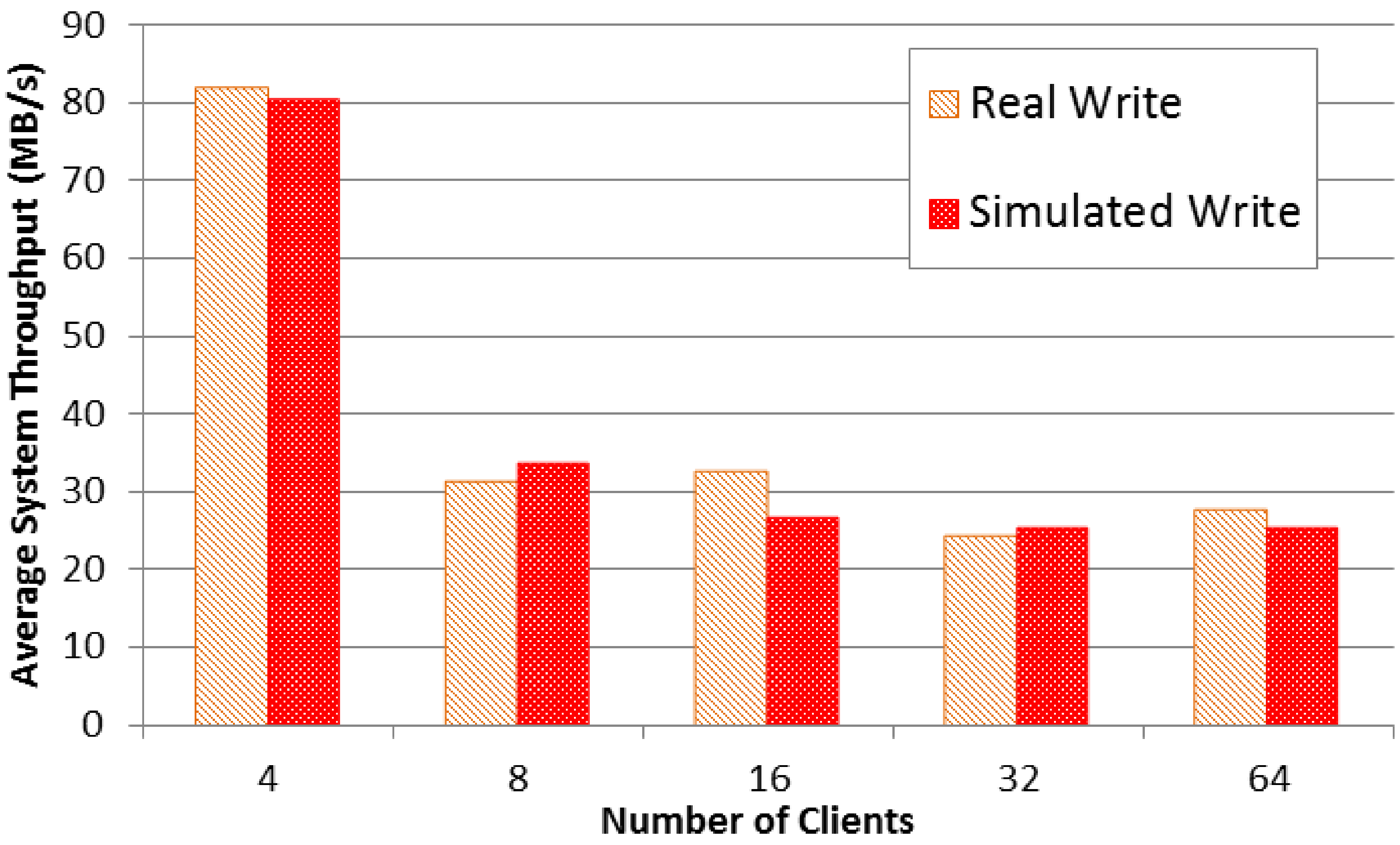
PFSsim Validation

- Validate the I/O throughput and latency under different workloads
- Benchmark system
  - PVFS2: 4 data servers/1 metadata server/varying number of clients
  - Each client/server has one 2.4GHz 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

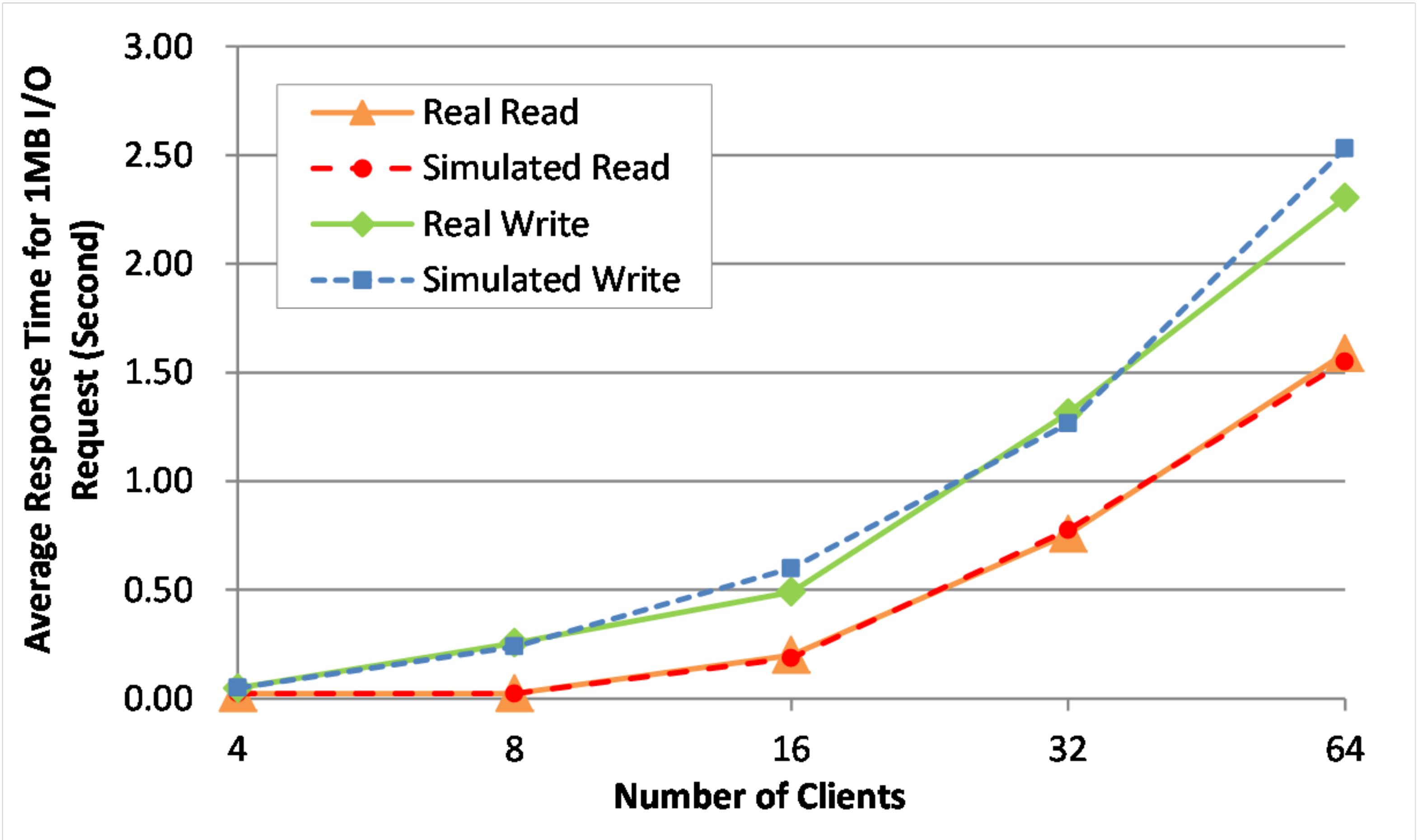
Read Throughput



Write Throughput



Response Time



Scheduler Validation

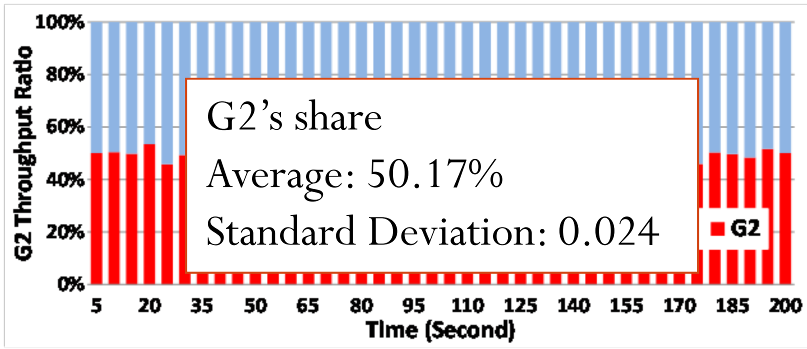
- Validate SFQ(D)* algorithm with different proportional sharing ratios
- Benchmark system and traces
  - PVFS2: 4 data servers/1 metadata server
  - 16 clients in Group1(G1) / 16 clients in Group2(G2)
  - SFQ(D) is deployed on each scheduler (D=4)
  - One scheduler per data server
  - Each client sequentially writes to 400MB, 1MB per write
- Varying sharing ratio between G1 and G2

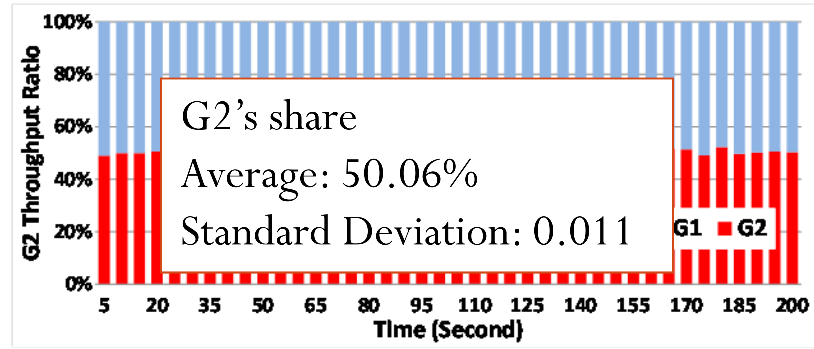
* W. Jin, et al., "Interposed Proportional Sharing For A Storage Service Utility", SIGMETRICS'04.

Scheduler Validation

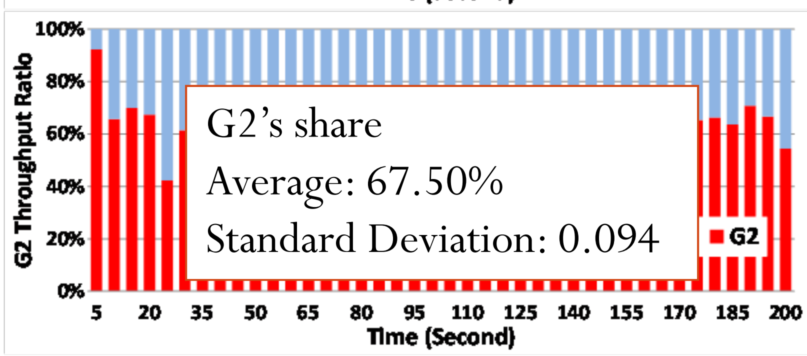
Real System Results | Simulated Results

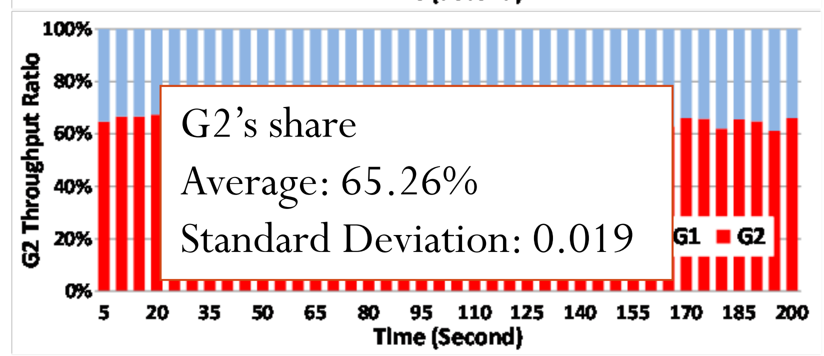
G1:G2 =1:1



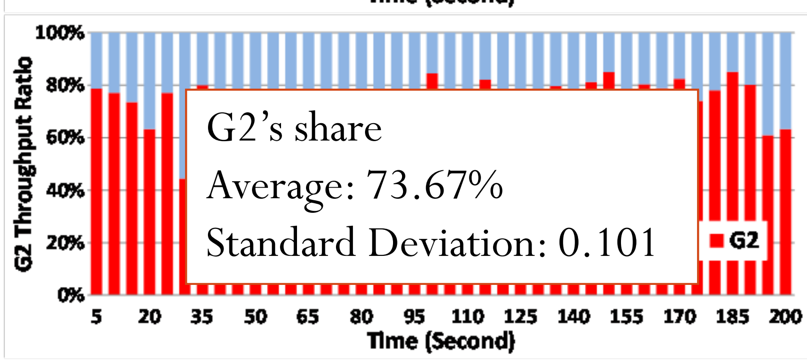


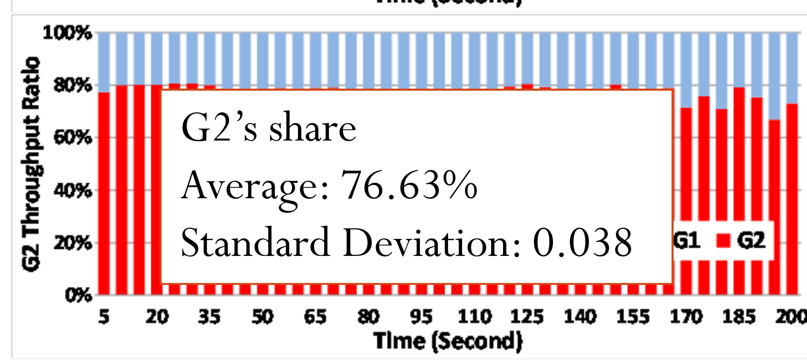
G1:G2 =1:2





G1:G2 =1:4





Outline

- Introduction
- Related Work
- System Modeling
- Simulator Implementation
- Validation and Evaluation
- Conclusion
- Future Work

Conclusion

- Progress towards the four basic design goals
  - Easy-to-use
    - Modular system design, object-oriented code
  - Flexible
    - Highly tunable parallel file system configuration, scheduler parameters, and network topology
  - Accurate
    - Good simulation accuracy shown in the validation results
  - Scalable
    - Able to simulate 512 clients and 32 servers in half an hour on a PC with 2.13GHz Intel i3 CPU and 2GB RAM

Outline

- Introduction
- Related Work
- Design and Implementation
- Validation and Evaluation
- Conclusion
- Future Work

Future Work

- Validate PFSsim against more realistic benchmarks
- Integrate a synthetic trace generator
- Simulate disk block management using statistical models
- Explore ways to support the simulation of very large scale systems

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
- Sponsor: NSF HECURA CCF-0937973/CCF-0938045
- More information:
  - http://visa.cis.fiu.edu/hecura
  - https://github.com/myidpt/PFSsim

Thank You!