# 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*
**{yonggang,renato}@acis.ufl.edu, {darte003,yxu006,ming}@cis.fiu.edu**

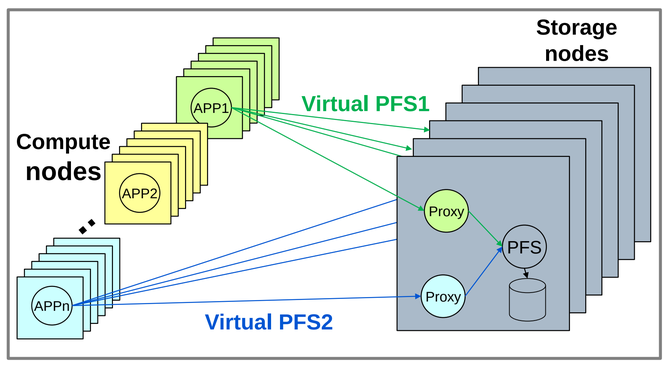
## Introduction

- As the size of parallel file systems (PFS's) in high-end computing (HEC) centers grows to peta- or exascale, to thoroughly research the data I/O scheduling mechanisms in such systems becomes very costly and disruptive.
- While parallel file system simulation frameworks have been proposed [1] [2], their emphasis has not been in the evaluation of the data I/O scheduling.
- We propose PFSsim, a simulator designed for the purpose of testing scheduling mechanisms for data I/O in PFS's.
- PFSsim is a trace-driven simulator based on the network simulation framework OMNeT++ and the disk device simulator DiskSim.

## Simulation Target: Virtualized Parallel File Systems

**Proxy-based PFS scheduling**

- Intercept the I/Os between the compute nodes and PFS storage nodes
- Create per-application virtual PFS's and enforce I/O resource allocation



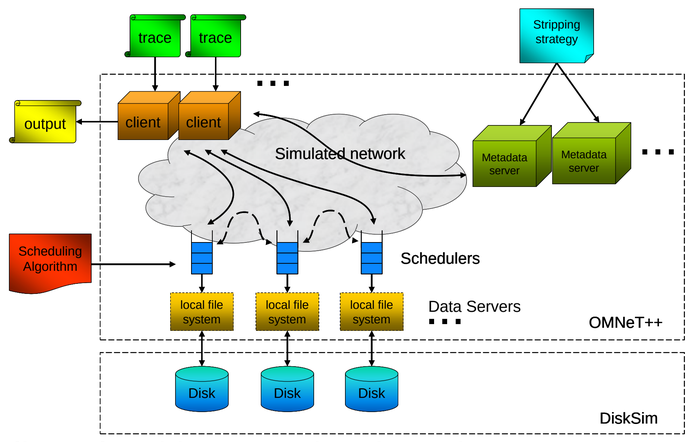

**Virtualized PFS's**

- Application-specific I/O bandwidth allocation for each virtual PFS
- Dynamically created and destroyed based on application lifecycles

## PVFSsim Architecture

**PFS and network layer**

- Based on discrete event simulation library OMNeT++ 4.0
- Network topology, file stripping strategy and scheduling algorithms are modeled
- Schedulers are deployed between clients and data servers, to apply the scheduling algorithms on the data I/O




**Disk device layer**

- Based on disk device simulator DiskSim 4.0
- Each DiskSim instance simulates one disk device. They synchronize with the data server modules in OMNeT++ through specific TCP connections

## Conclusions and Future Work

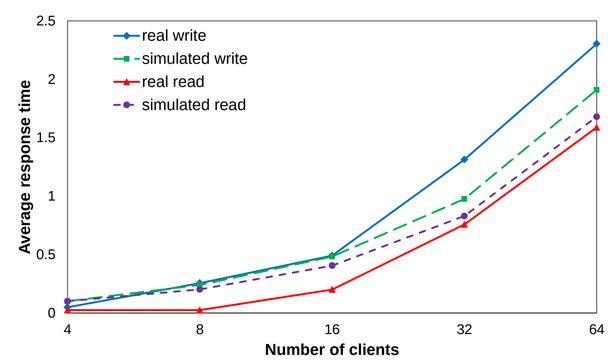
- Initial validation shows the system is capable of tracking relative performance trends based on workload and scheduling algorithms.
- Modularized system design facilitates the users to customize target file systems, as well as to implement new scheduling strategies.
- In the future, our work will proceed towards better simulation fidelity and faster simulation speed. We will refine the storage node design and TCP connection simulation, and deploy efficient synchronization schemes to cut down simulation overhead.

## References

[1] E. Molina-Estolano, C. Maltzahn, J. Bent and S. Brandt, "Building a parallel file system simulator", 2009, Journal of Physics: Conference Series 180 012050.
[2] P. Carns, B. Settlemyer and W. Ligon, "Using Server-to-Server Communication in Parallel File Systems to Simplify Consistency and Improve Performance", Proceedings of the 4th Annual Linux Showcase and Conference (Atlanta, GA) pp 317-327.
[3] W. Jin, J. Chase, and J. Kaur, "Interposed proportional sharing for a storage service utility", Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Jun 2004.

## Simulator Validation

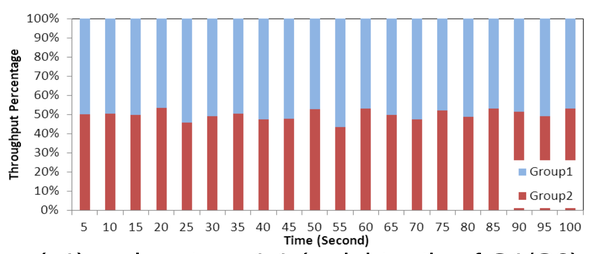
- **Simulated System** Parallel Virtual File System (v2), containing 1 metadata server and 4 object storage devices.
- **Benchmark** IOR, each client continuously issues 400 I/O requests (1MB each, sequential read/write from/to one file); the data is evenly distributed onto 4 object storage devices (OSDs).
- **Platform** A cluster of virtual machines, each with 2.4GHz AMD CPU and 1 GB RAM. EXT3 local file systems are deployed.



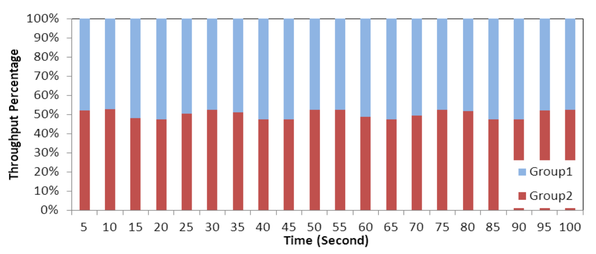

***We observed …***

- Even though the simulator is not able to predict the absolute values with accuracy for the real system, relative values can still be used to infer the performance trade-offs.
- For the lower read workload (smaller client number), the simulated results do not match the real results very well. The possible reason is the real local file system EXT3 provides read pre-fetching, which promotes the reading speed when recent accessed data are adjacent in locality.

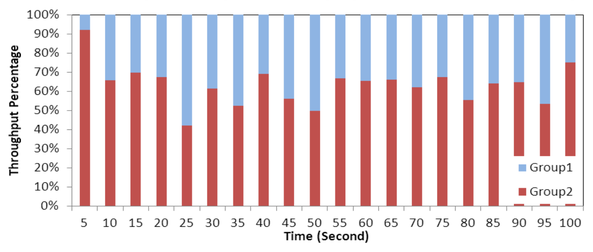
## Scheduler Validation



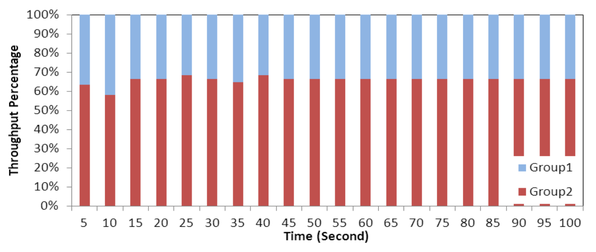

(a1) real system, 1:1 (weight ratio of G1/G2), 50.00% (average throughput ratio G2 takes)



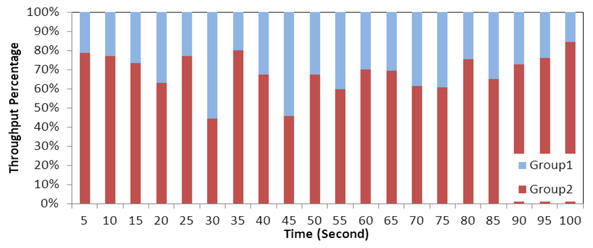

(a2) simulator, 1:1, 50.23%



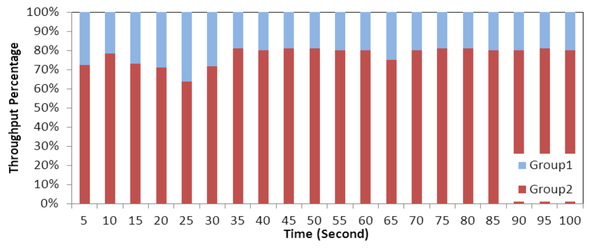

(b1) real system, 1:2, 63.41%




(b2) simulator, 1:2, 66.19%




(c1) real system, 1:4, 68.58%




(c2) simulator, 1:4, 77.67%

- **Simulated System** Servers are the same as last experiment. Clients are divided into 2 groups; each group has 16 clients.
- **Scheduling algorithm** Start-time Fair Queuing algorithm[3] with depth=4. Conduct multiple tests by assigning different weight ratios to Group1 and Group2 (shown in picture tittles).
- **Benchmark** The same as last experiment, expect that only sequential write I/Os are issued by the clients.

***We observed …***

- Average throughput shows that PFSsim is able to model first-order performance effects of proportional-sharing algorithms.
- The oscillation in real results is due to many other factors, such as TCP timeout or caching. These factors are not yet modeled.

## Acknowledgement

This material is sponsored by the National Science Foundation under HECURA collaborative awards 0937973 and 0938045. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the sponsors.