

QoS-driven Storage Management for High-end Computing Systems

Ming Zhao

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

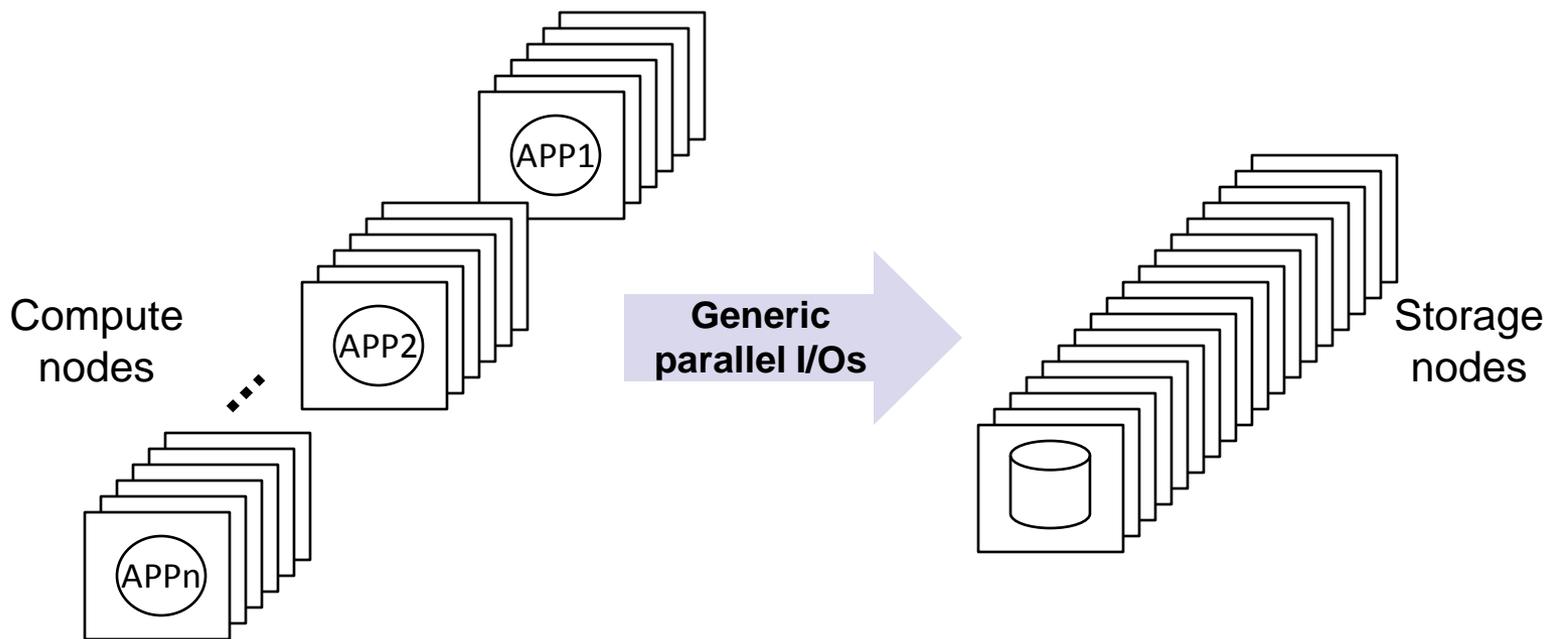
Renato Figueiredo

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



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

Objectives

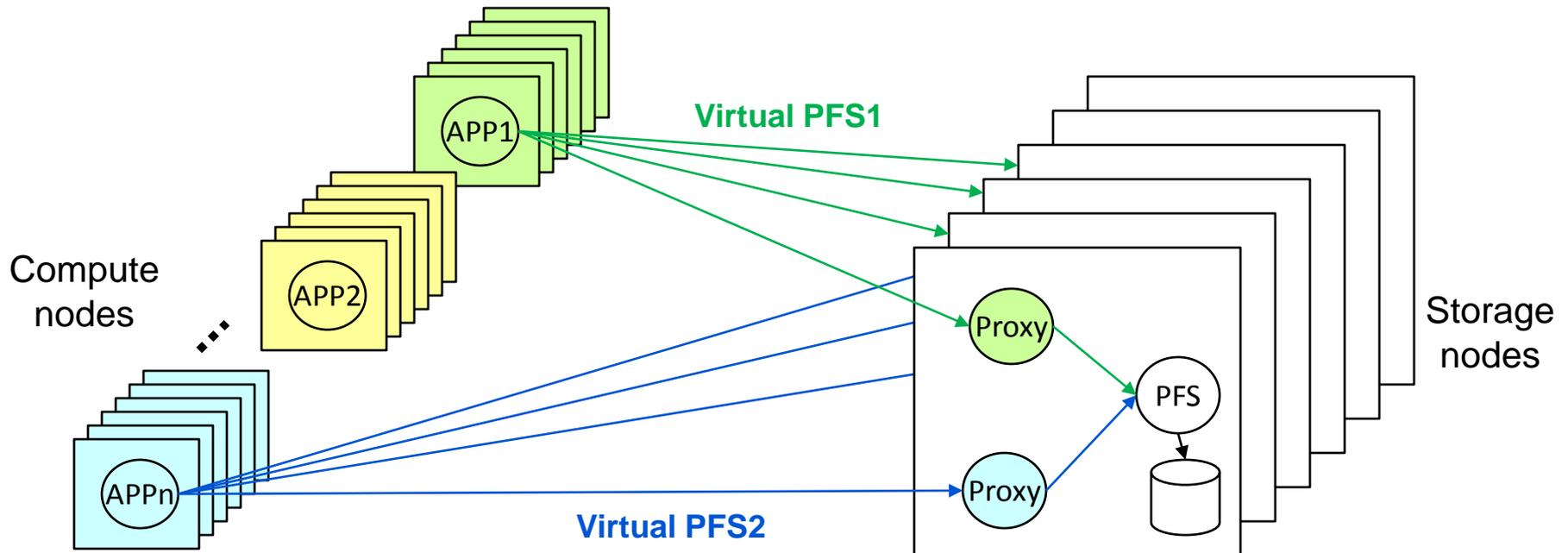
- Per-application storage resource allocation
 - Parallel file system (PFS) virtualization
- Efficient management of storage resource allocations
 - Storage management services
- Automatic optimization of storage resources usage
 - Autonomic storage resource management

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
 - Per-application virtual PFSs
 - Dynamically created and destroyed based on application lifecycles
 - Application-specific I/O bandwidth allocation per virtual PFS

PFS Virtualization

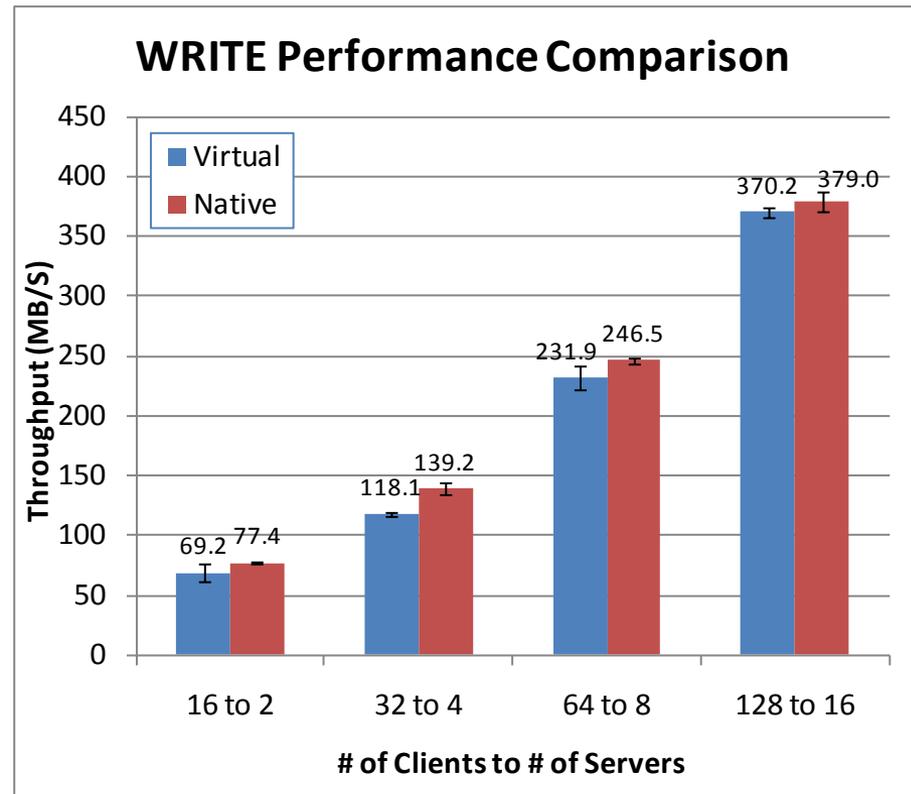
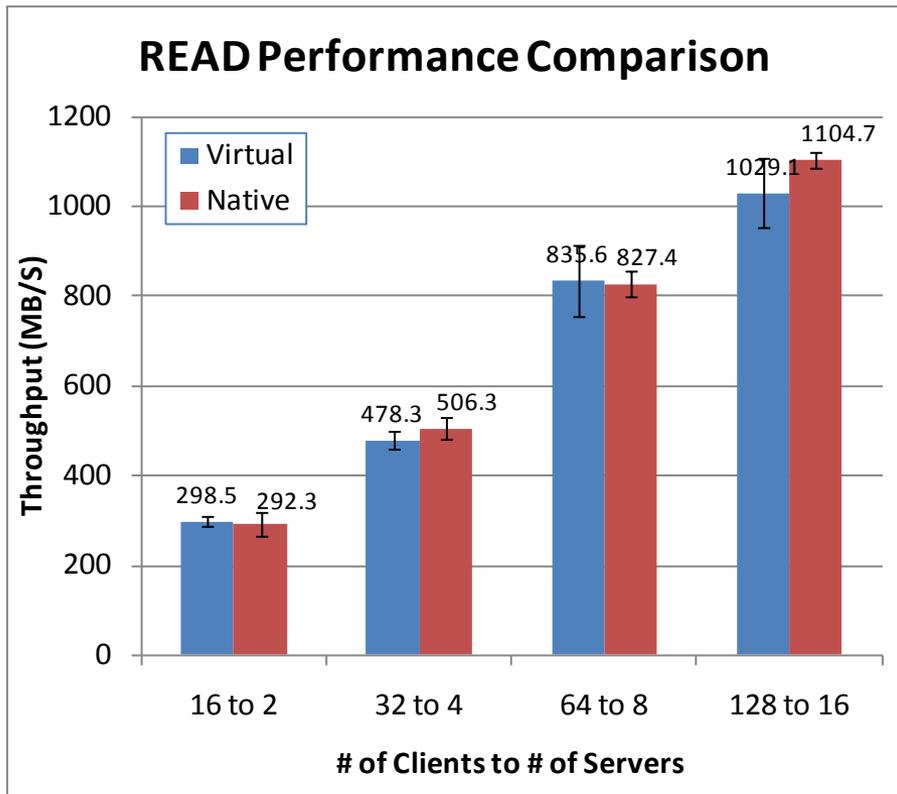
- Proxy-based PFS virtualization
 - Indirection of application I/O access
 - Creation of per-application virtual PFS



Implementation

- Prototype
 - A PVFS2 (Parallel Virtual File System) proxy
 - Intercept PVFS2 messages and virtualize PVFS2 deployment
- Evaluation
 - A virtual machine based testbed
 - Up to 128 PVFS clients and 16 PVFS servers
 - Benchmark: IOR2

Virtualization Overhead

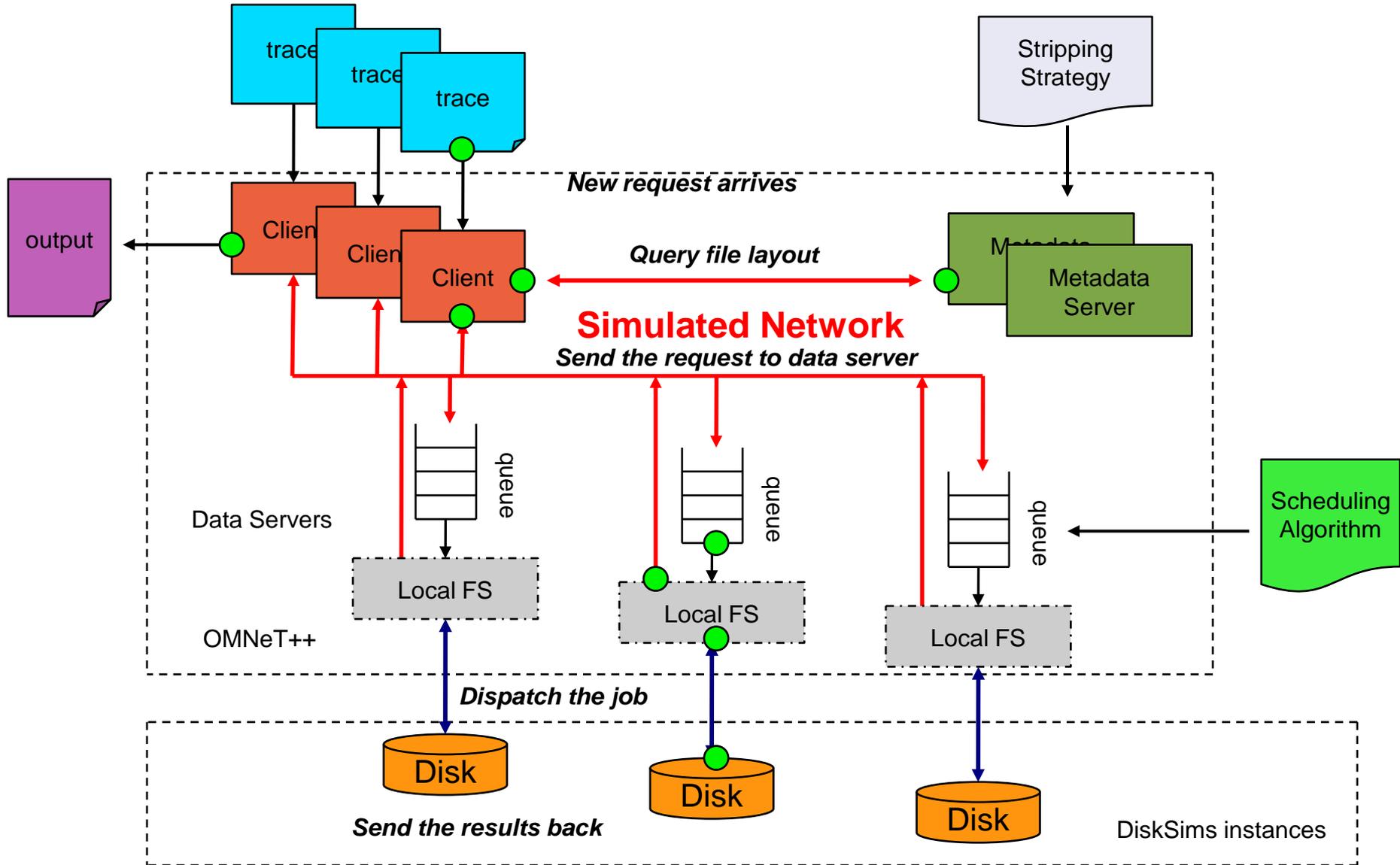


- Proxy CPU and memory usages are both very low

Simulation-based I/O Scheduling Study

- PFS simulator
 - 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
- Our focus is in parallel I/O scheduling
 - Simulate enough details necessary for scheduling study but with an acceptable simulation time

Simulator Architecture



Implementation of Scheduling Algorithms

```
/* SFQ algorithm, at each data server */
```

```
systemtime = 0
```

```
waitQ.initiate()
```

```
while(!simulation_end) {
```

```
  if reqArrive(), then:
```

```
    R = getReq()
```

```
    R.start_tag = min { R.getPrevReq().finish_tag, systemtime }
```

```
    R.finish_tag = R.start_tag + R.cost / R.getFlow().weight
```

```
    pushReq(R, waitQ)
```

```
  if diskHasSlot(), then:
```

```
    R = popReqwithMinStartTag(waitQ)
```

```
    systemtime = R.start_tag
```

```
    dispatch(R)
```

```
}
```

The request from client arrives at the data server.

DiskSim tells OMNet++ that it still has free slot.

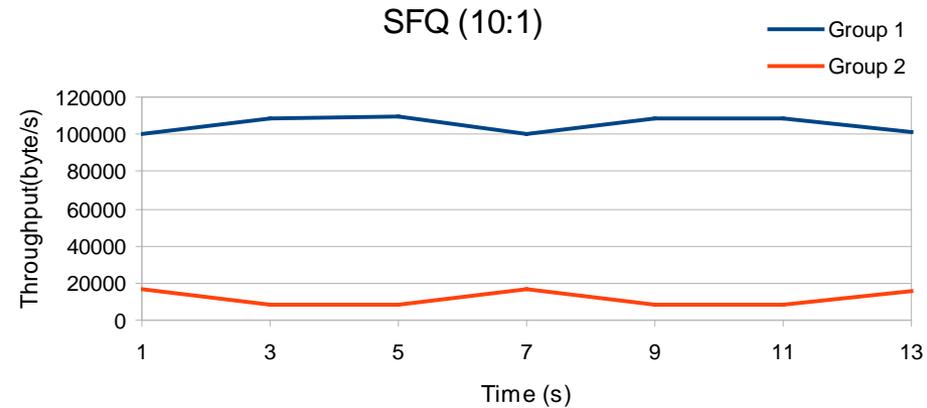
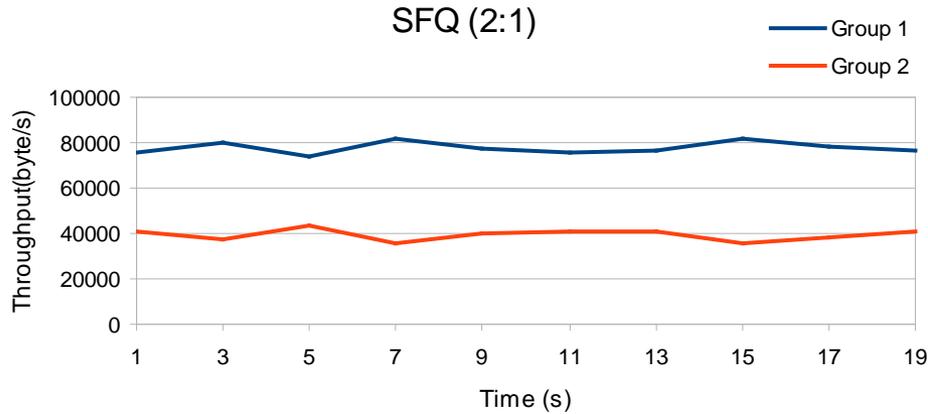
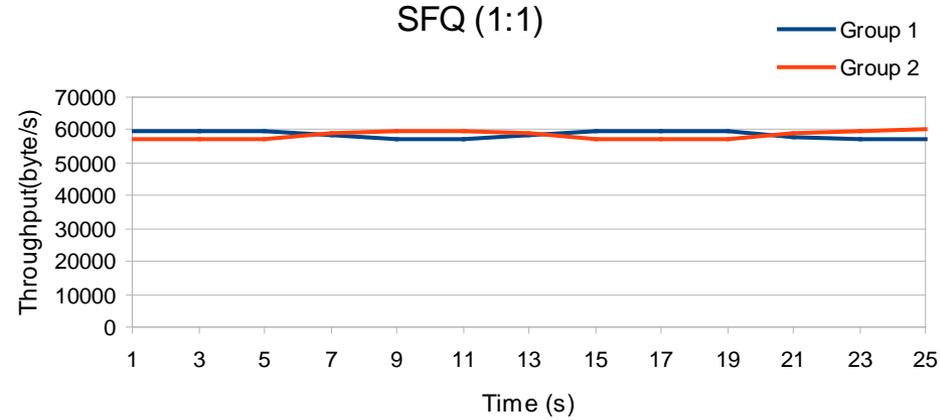
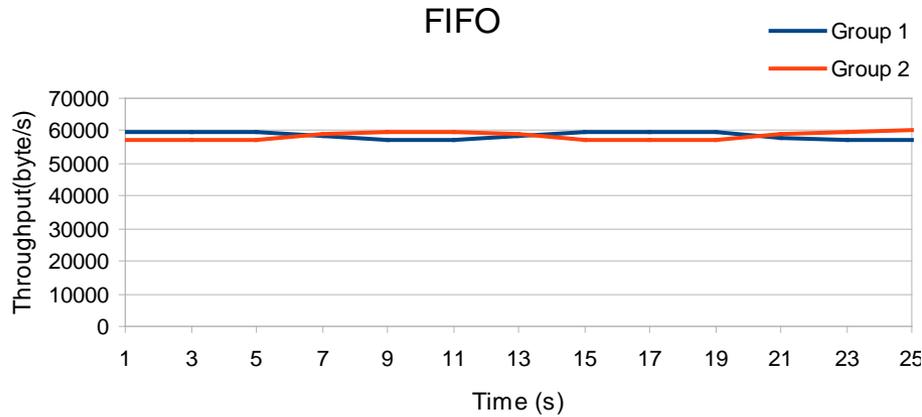
OMNet++ dispatches the request to DiskSim.

- Other scheduling algorithms, e.g., FIFO, Distributed SFQ (DSFQ), MinSFQ, are also implemented

Simulation Example 1

- Simulating a PVFS2 setup
 - 2 parallel applications (each with 16 clients)
 - 4 data servers and 1 metadata server
 - All files striped to all servers (stripe size: 256KB)
- Trace files are generated by IOR2
 - Each client does a 100MB checkpoint operation
- I/O scheduling algorithms
 - FIFO
 - Local SFQ with different weight assignments (1:1, 2:1, 10:1)

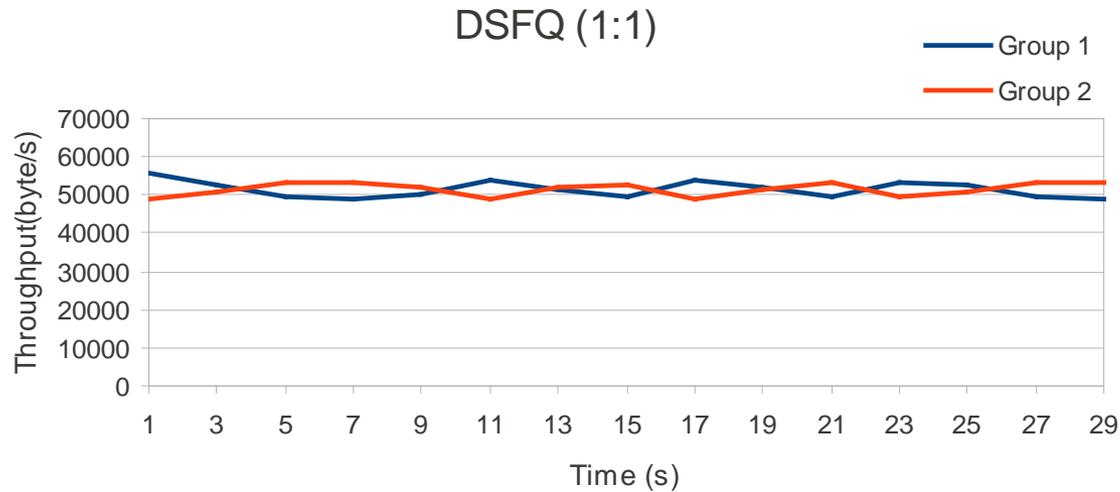
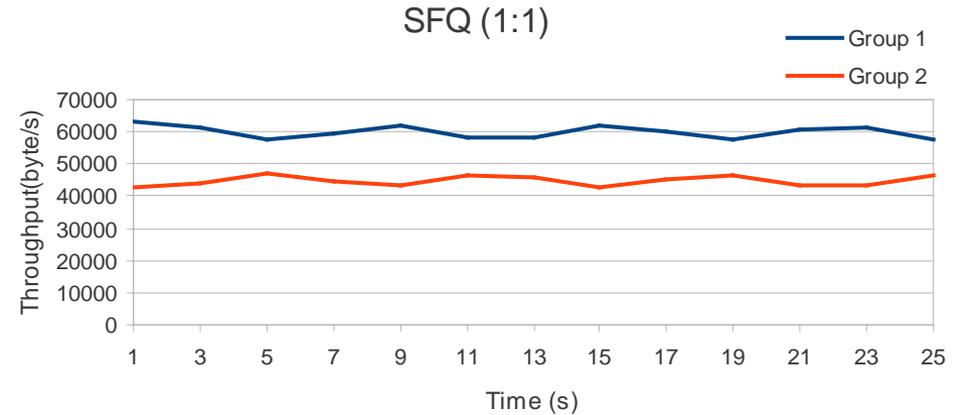
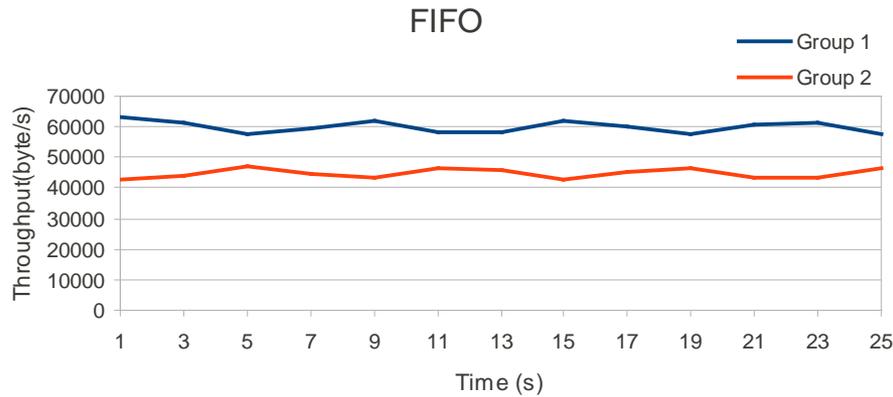
Throughput Results



Simulation Example 2

- Similar to Simulation 1, except that workloads are not evenly distributed across the data servers
 - Application 1's files are striped to all servers [1, 2, 3, 4]
 - Application 2's files are striped to only 3 servers [1, 2, 3]
- I/O scheduling algorithms
 - FIFO
 - Local SFQ
 - Distributed SFQ (all servers share the global scheduling information)

Throughput Results



Conclusion and Future Work

- Proxy-based PFS virtualization is feasible
 - Its throughput overhead and resource usage overhead are not significant
 - TODO: implement optimized I/O schedulers upon proxy
- Simulation-based PFS scheduling study is valuable
 - Its results can guide the design of real I/O schedulers
 - TODO: improve the scale and realism of simulation

Acknowledgement

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

Backup slides

Parallel File System (PFS) Background

- At the core of storage resource management
 - Components: PFS clients, data servers, metadata servers
 - Examples: GPFS, IBRIX, Lustre, PanFS, PVFS etc.
- Designed for general parallel applications
 - No differentiation of different application I/Os
- Fine-tuned for overall system throughput
 - Not for specific application I/O QoS requirements

Different virtualization approaches

- Proxy based virtualization
 - Applicable to different PFS protocols
 - Seamless integration with existing HEC storage systems
 - Non-negligible overhead due to extra layer of indirection
- PFS extension based virtualization
 - Modifications on existing PFS protocols
 - Support for per-application I/O identification and handling

Service-based Storage Management

- Problem:
 - Management of I/O bandwidth allocations for a large number of applications in a ultra-scale HEC system
- Proposed solution:
 - Service-based middleware for managing virtual PFSs
 - Storage resource scheduling
 - Storage resource monitoring

Service-based Storage Management

- Storage resource scheduling
 - Support for per-application reservation of I/O bandwidth
 - Integration with typical HEC job schedulers (e.g., PBS, Torque, LoadLeveler)

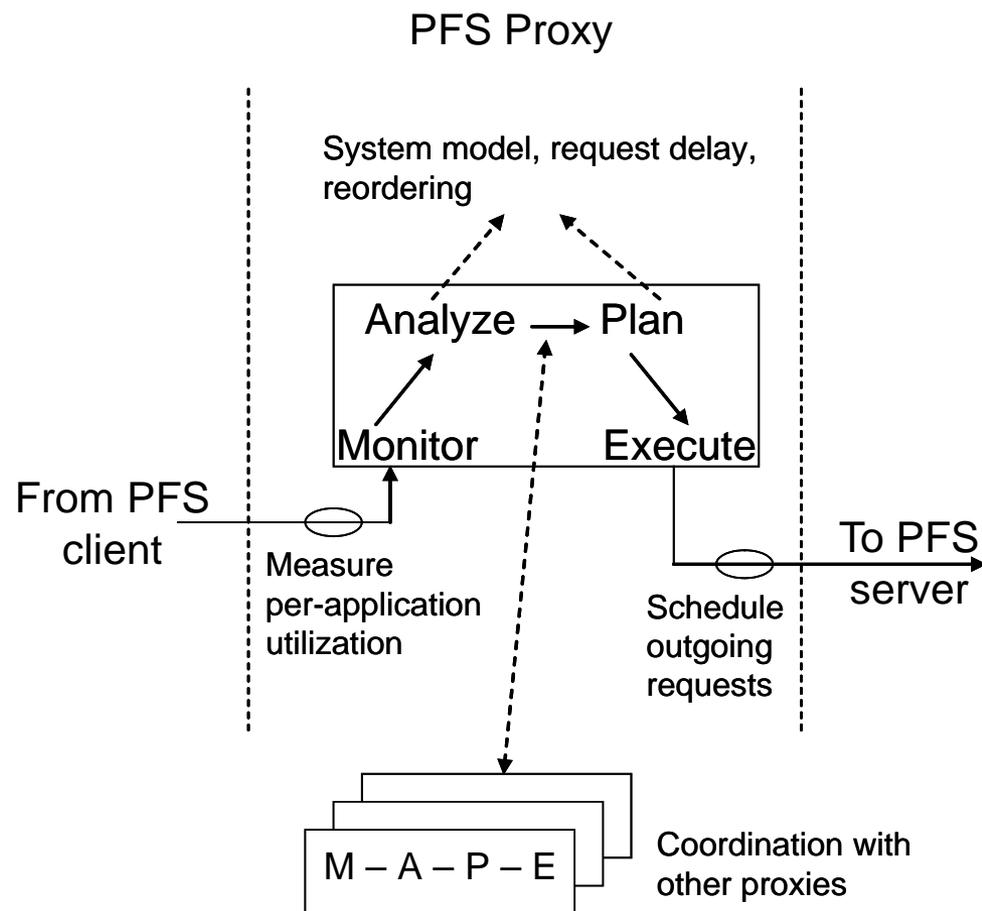
- Storage resource monitoring
 - Support for per-application tracking of bandwidth usage
 - Integration with typical cluster monitoring frameworks (e.g., Ganglia, ClusterMon)

Autonomic Storage Resource Optimization

- Problem:
 - Dynamic resource scheduling for fair sharing of storage resources
 - Automatic optimization of I/O bandwidth utilization
- Proposed research:
 - Autonomic storage resource management upon the virtualized PFS infrastructure

Autonomic Storage Resource Optimization

- Proxy-based autonomic I/O control loop
- Dynamic scheduling algorithms (e.g., SFQ)
- Optimization based on coordinated scheduling



Existing PFS simulators

- The IMPIOUS simulator, by E Molina-Estolano, *al. et*[1].
 - It does not model the metadata server.
 - The scheduler modules are lacking, so scheduling algorithms are hard to model.
- The simulator developed in PVFS improvement paper by Carns P. H., *al. et*[2].
 - It over simulates the network, which extends the simulation time.
 - It uses real PVFS in simulation, which introduces too much details, while not flexible.

Simulator Details

- Use the discrete event simulation library OMNeT++ 4.0 to simulate the network.
 - It is capable of simulating the network topology with bandwidth and delay.
- Use DiskSim to simulate the data server disks.
 - DiskSim accurately estimates the time for data transactions on the physical disks.
 - DiskSim allows users to extract disk characteristics from real disks.

References

- [1] E Molina-Estolano, C Maltzahn, J Bent and S A Brandt, “Building a parallel file system simulator”, *Journal of Physics: Conference Series 180 (2009) 012050*.
- [2] Carns P H, Ligon W B, *al. et.* “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] Yin Wang and Arif Merchant, “Proportional Share Scheduling for Distributed Storage Systems”, *File and Storage Technologies (FAST’07)*, San Jose, CA, February 2007.
- [4] W. Jin, J. S. Chase and J. Kaur, “Interposed Proportional Sharing For A Storage Service Utility”, *SIGMETRICS*, E. G. C. Jr., Z. Liu, and A. Merchant, Eds. ACM, 2004, pp. 37-48.