Two-level Throughput and Latency I/O Control for Parallel File Systems

Yiqi Xu, Ming Zhao

Florida International University

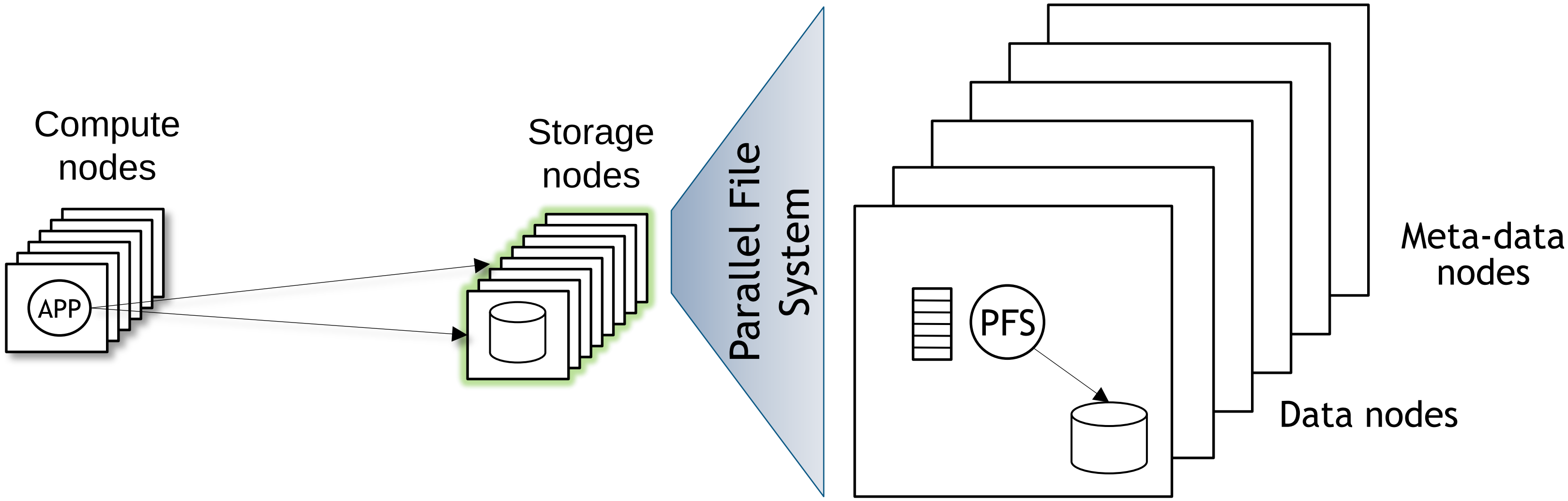
School of Computing and Information Sciences

http://visa.cis.fiu.edu

yxu006@cis.fiu.edu, ming@cis.fiu.edu

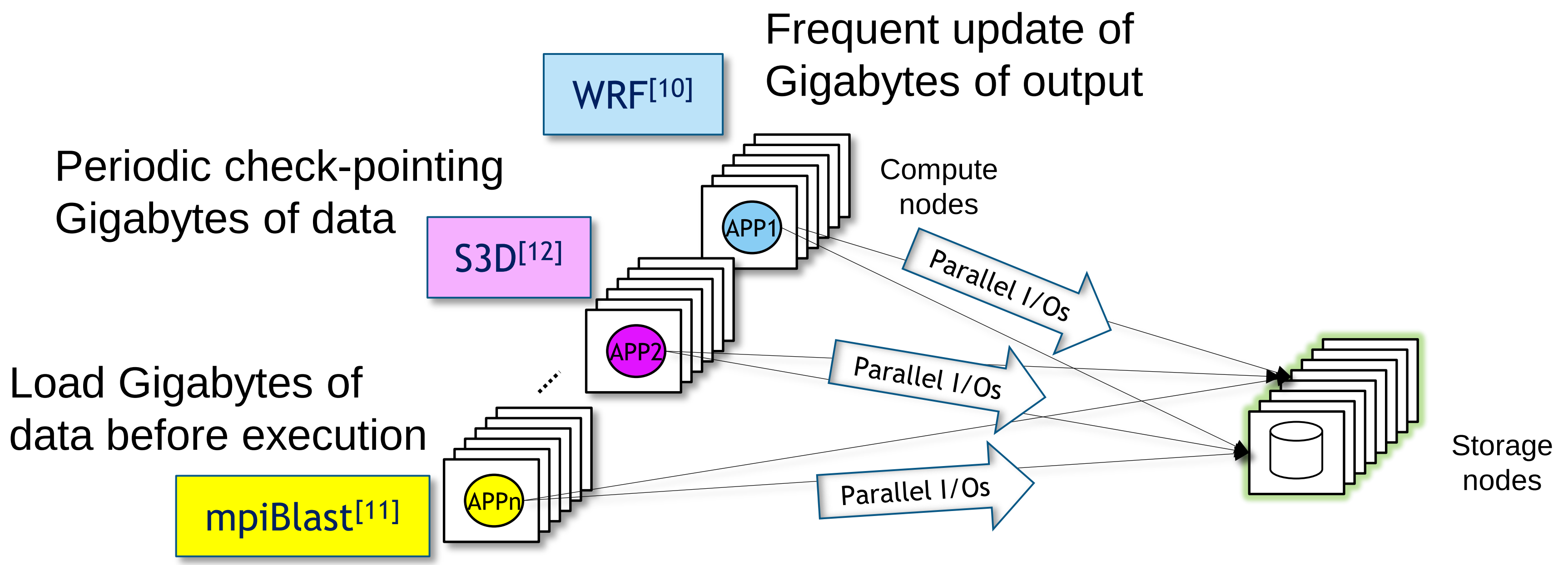
FLORIDA INTERNATIONAL UNIVERSITY

Background – Parallel Storages



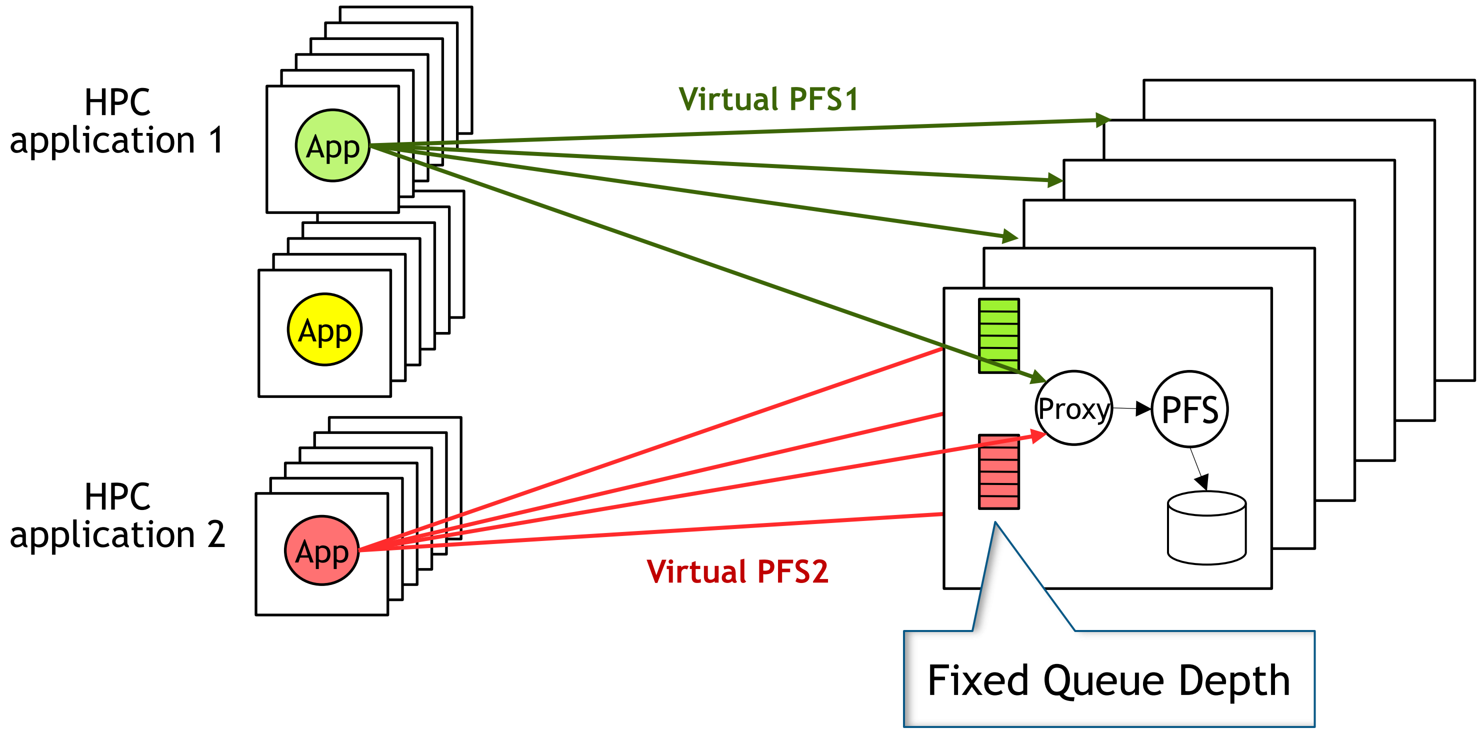
- Parallel File System in High Performance Computing
  - Distribute data on multiple storage nodes
  - Aggregate throughput from multiple, parallel storage nodes
- Components
  - Server side: data & meta-data server daemon
  - Client side: MPI library, client daemon

Motivation (1)



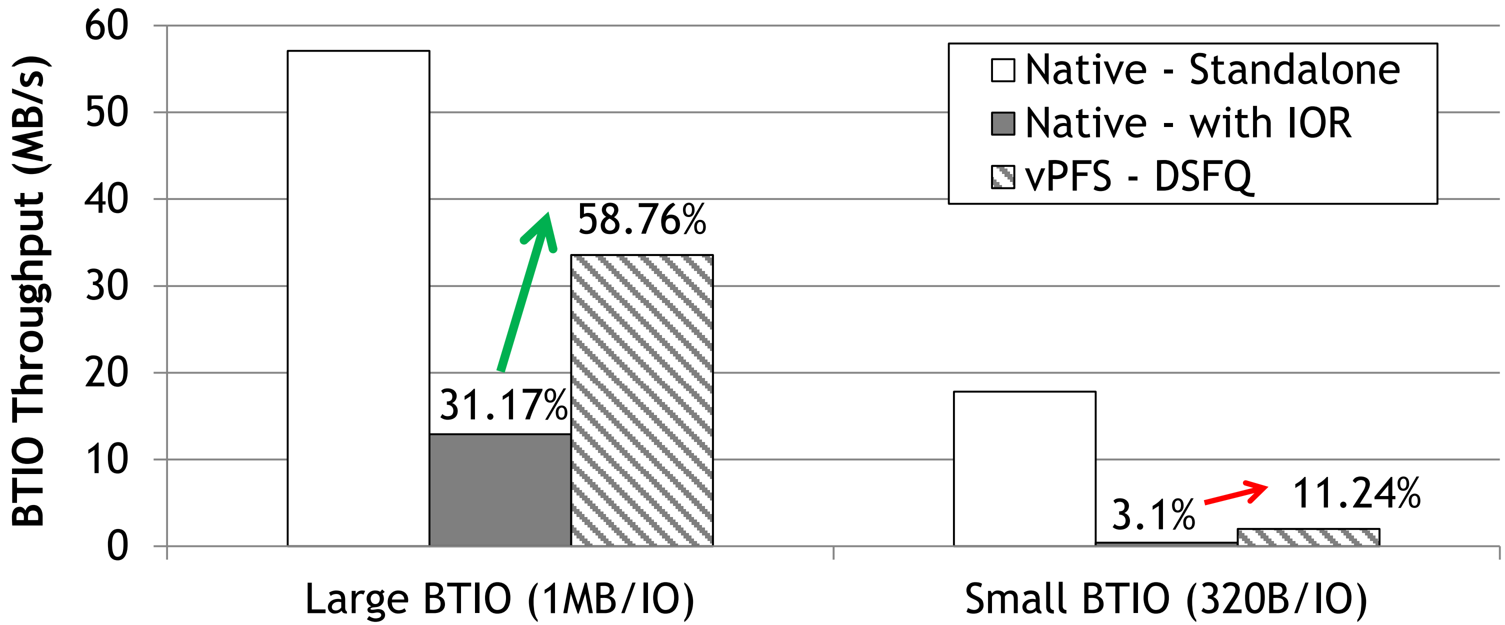
- Parallel storage is commonly shared
  - Applications have different I/O demands
  - Their I/Os interfere with each other

Background – vPFS



- Enhanced distributed SFQ scheduler
- Global bandwidth proportional sharing with low overhead

Motivation (2)



- Two representative parallel application: BTIO[9]/IOR[8]
- Limited performance improvement from vPFS[5]
- Throughput alone is not enough to satisfy applications' performance needs

Overview

- Problem
  - HPC applications requiring throughput or latency (or both) guarantees interfere with each other on the parallel storage
  - vPFS enforcement on bandwidth sharing is NOT enough to satisfy different applications' needs
- Solution
  - Use vPFS to create a new scheduler to recognize and regulate I/Os with awareness of both throughput and latency needs

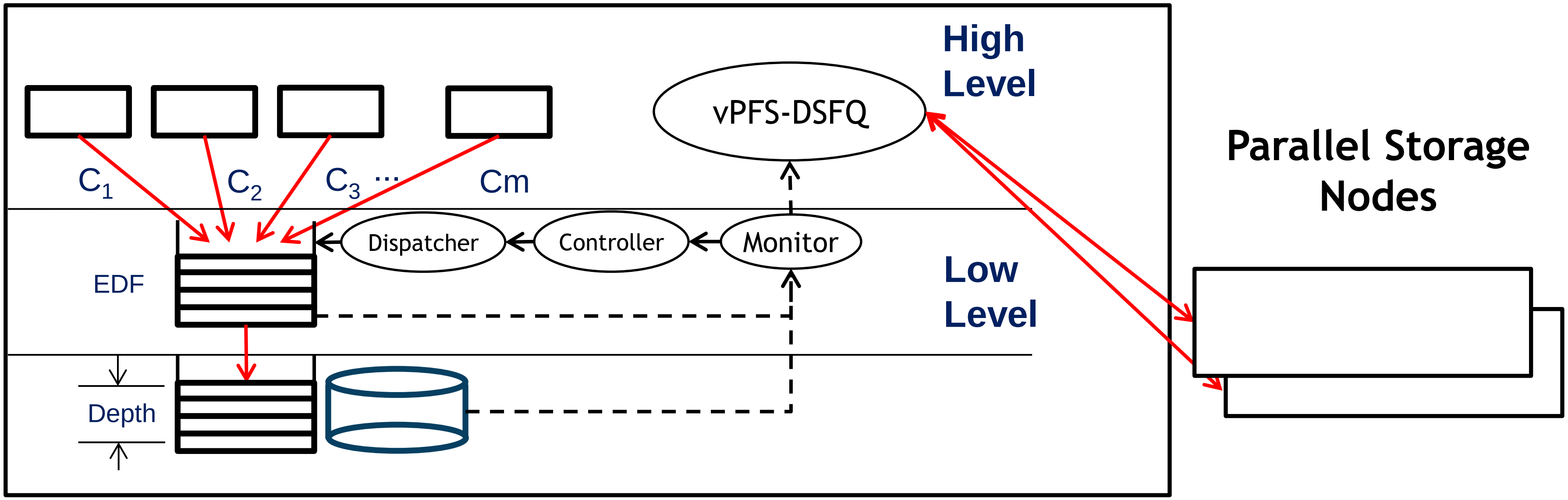
Outline

- Background, Motivation & Overview
- Two-Level Parallel I/O Scheduler
  - Architecture
  - Algorithm
- Experimental Evaluation
- Conclusions and Future Work

Two-Level QoS

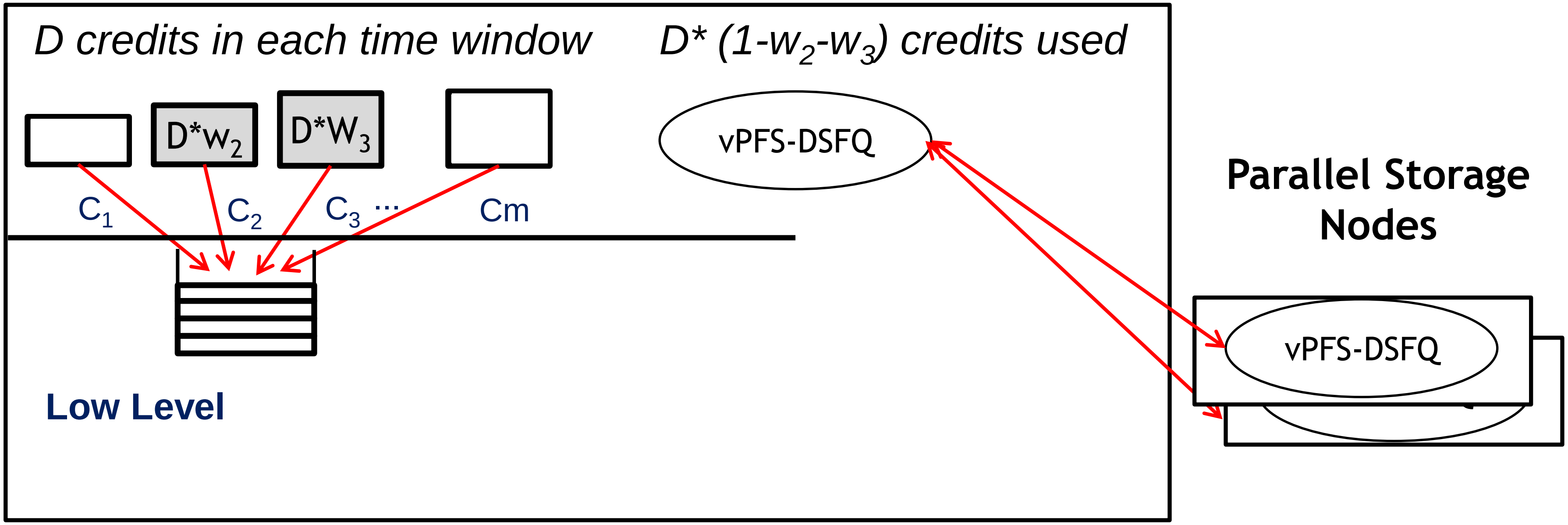
- (*T*, *D*): A tuple for both *T*hroughput and *L*atency
  - *T* is the agreed throughput upper bound limit from the application
  - *D* is the guaranteed the latency (deadline) upper bound from the storage
  - When *T* is violated, *D* is not guaranteed any more

Architecture



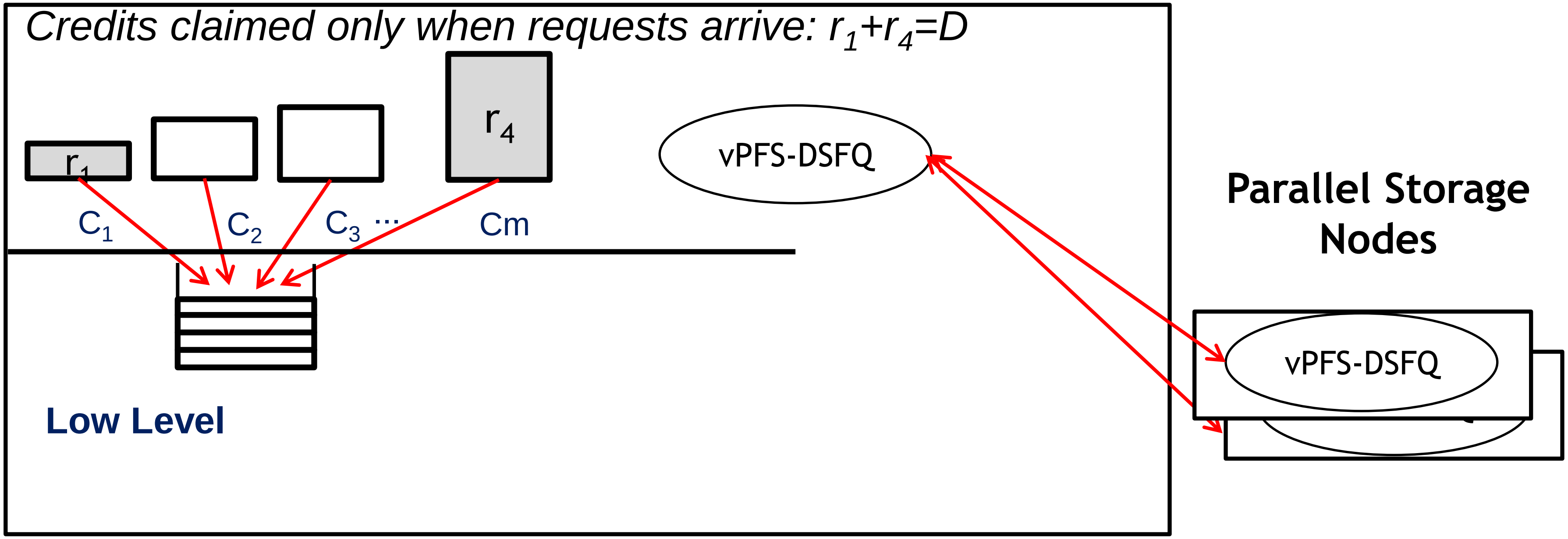
- High level provides throughput control as well as service synchronization
- Low level monitors the device and adjusts # outstanding requests of the device[13]

High Level Throughput Control



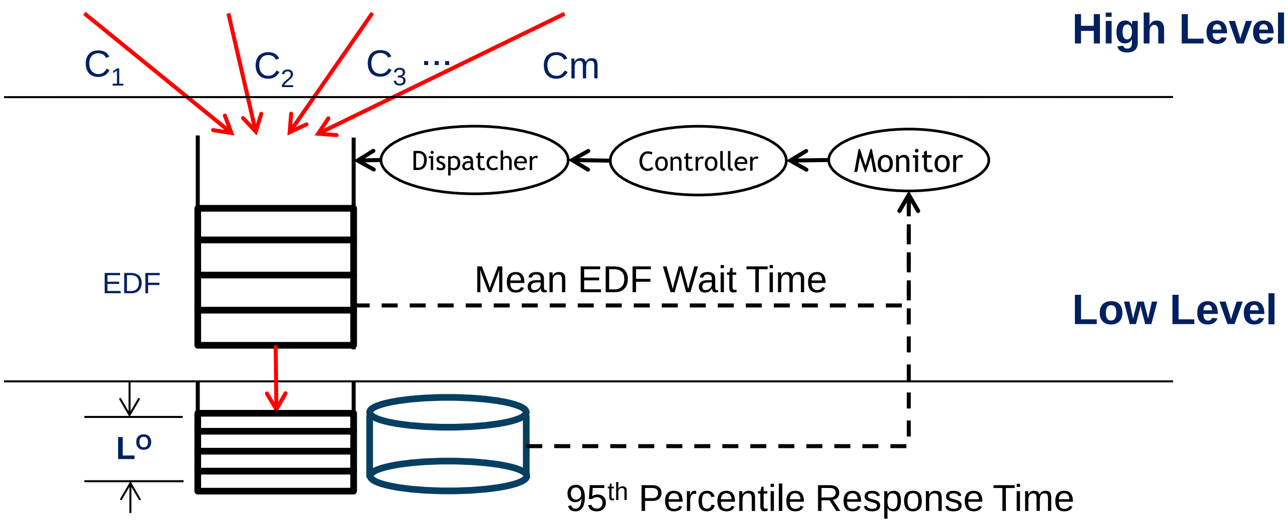
- Efficient parallel storage synchronization: total-service proportional sharing of bandwidth
- Strict fair sharing using SFQ-based algorithm: better utilization

High Level Throughput Control



- Total-service proportional sharing: parallel storage synchronization
- Strict fair sharing of using SFQ-based algorithm: better utilization

Low Level Latency Control

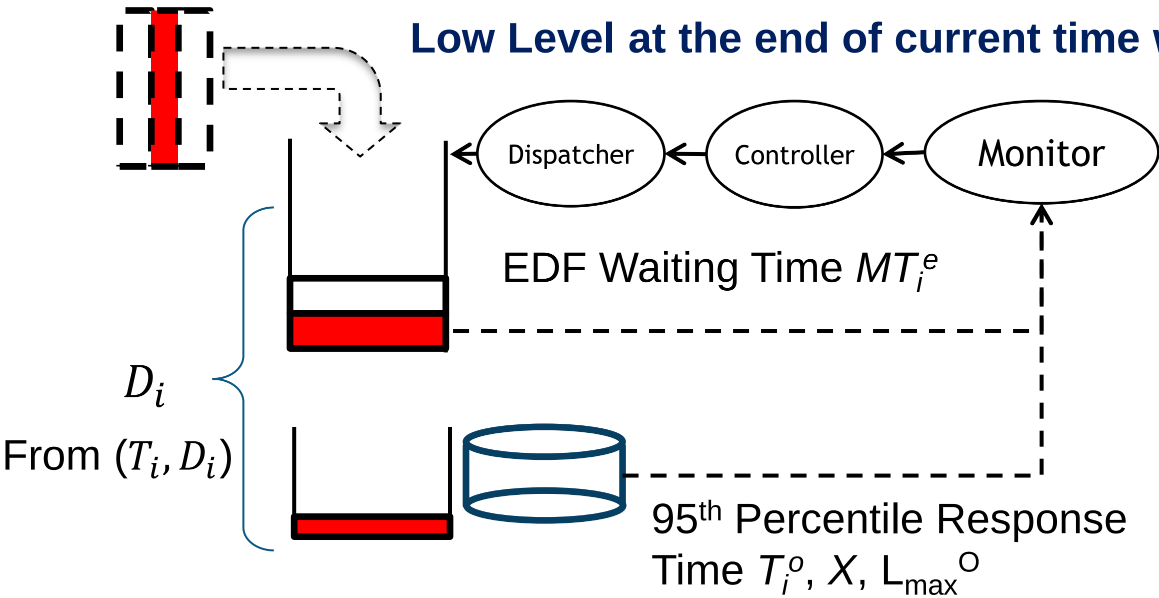


- Final dispatching of requests to storage device
- A feedback-control loop for adjusting the device depth

Low Level Bounds and Terms

- Three bounds to predict the future
  - For class i, the maximum depth L_{RT}^{O} allowed without violating the deadline
  - The lower bound depth L_{l}^{O} to ensure any request whose deadline is in the current time window is completed
  - The upper bound depth L_{u}^{O} if the latency need is continuously met and utilization should be raised
- Terms
  - X – # requests completed in last time window
  - L^{O} – current window queue depth
  - L^{O}_{max} – maximum # outstanding requests in current window

Low Level Feedbacks



$$e_i = \frac{D_i - MT_i^{\,e}}{T_i^{\,O}}$$

$$l = \frac{\underline{L}^O}{X}, \text{ e.g. } \frac{3}{X},$$

$$u = \frac{\overline{L}^O}{X}, \text{ e.g. } \frac{6}{X}$$

- L^O scaled by 3 coefficients to derive 3 threshold bounds
  - $e_i : \frac{average\ time\ left\ to\ complete\ future\ requests}{95th\ percentile\ latency\ to\ complete\ current\ requests}$ (about latency)
    - The more time left to complete a request, the larger e_i
    - The smaller actual device latency, the larger e_i
  - $l : \frac{\underline{future\ demand}}{current\ capability}$ or u: $\frac{\overline{future\ demand}}{current\ capability}$ (about throughput)

FIU FLORIDA INTERNATIONAL UNIVERSITY

Controlling L^O :Underloaded Case

- $L_{RT}{}^{O} = e_i \times L^O$; $L_l{}^{O} = l \times L^O$; $L_u{}^{O} = u \times L^O$

$If\ L_{RT}{}^{O} \leq L_l{}^{O}$ or $If\ L_u{}^{O} \leq L_{RT}{}^{O}$

$Then\ \infty$ $Then\ L_u{}^{O}$

$If\ L^{O}{}_{max} < L^O \leq L_{RT}{}^{O}$

$Then\ L^O$

$If\ L_{RT}{}^{O} < L^O$

$Then\ L_{RT}{}^{O}$

$If\ L^O \leq L^{O}{}_{max}$

$Then\ L_{RT}{}^{O}$

Controlling L^O :Overloaded Case

- $L_{RT}{}^{O} = e_i \times L^{O}{}_{max}$ver all classes, a minimum of all selected queue threshold is chosen

$If\ X < \bar{L}^{O}$

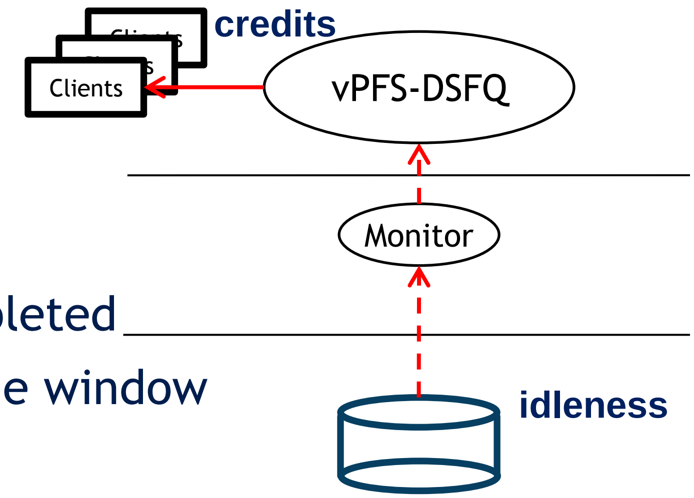
$Then\ \infty$

$\underline{L}^{O} \leq X$

$Then\ max(\underline{L}^{O}, L_{RT}{}^{O})$

- Over all classes, a minimum of all selected queue threshold is chosen

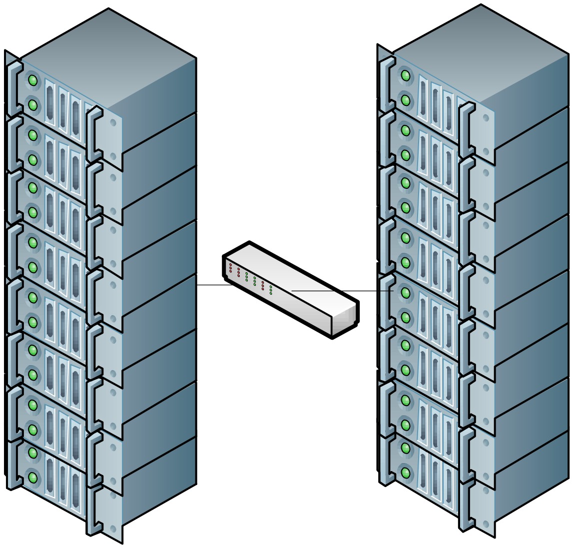
Cooperation between Two Levels



- Low level idleness detection
  - If $L_{curr} \leq L^{O} \times 0.9$
  - Idleness updated on the lower level:
    - When a request is dispatched or completed
    - At the beginning of an overloaded time window
- High level credit replenishment
  - When the lower level reports idleness
  - When no remaining credits
    - But new requests query and find the idleness
  - When credit replenishment time window elapsed

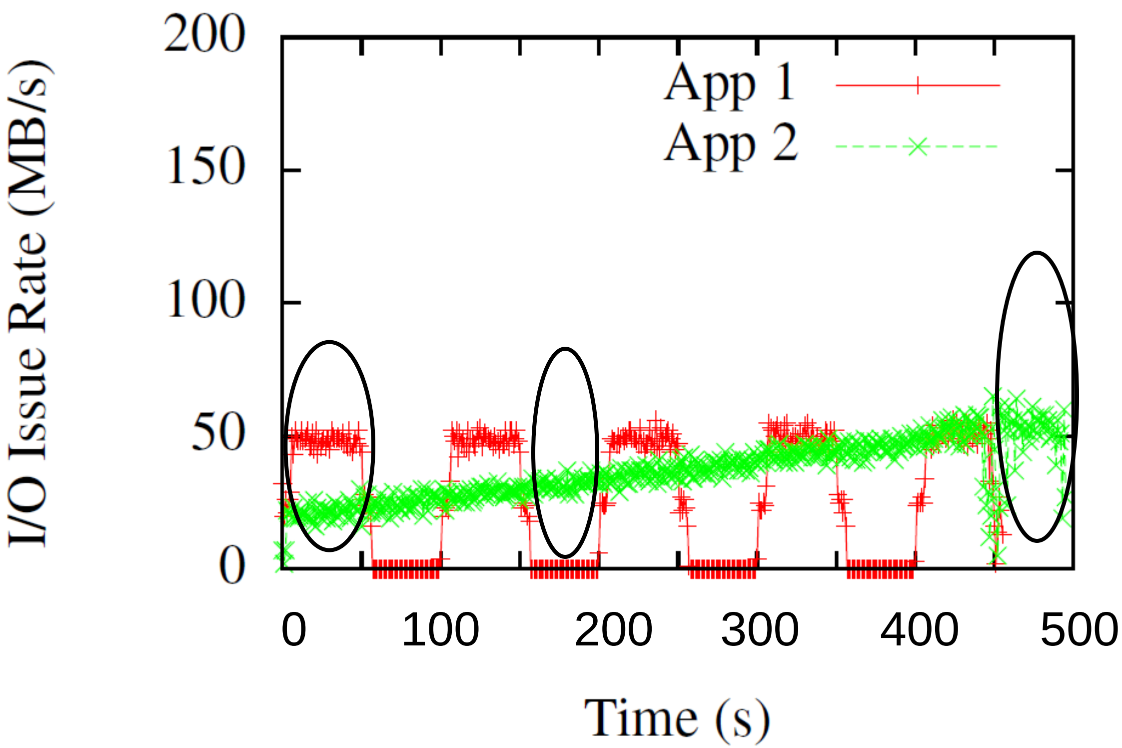
Evaluation

- Hardware
  - 1 Client with 64 processes
  - 1 Server
  - One gigabit switch
- Software
  - PVFS 2.8.2
  - IOR 2.10.3
- Experiments
  - Adaptation of storage queue size
  - Handling of overloaded storage



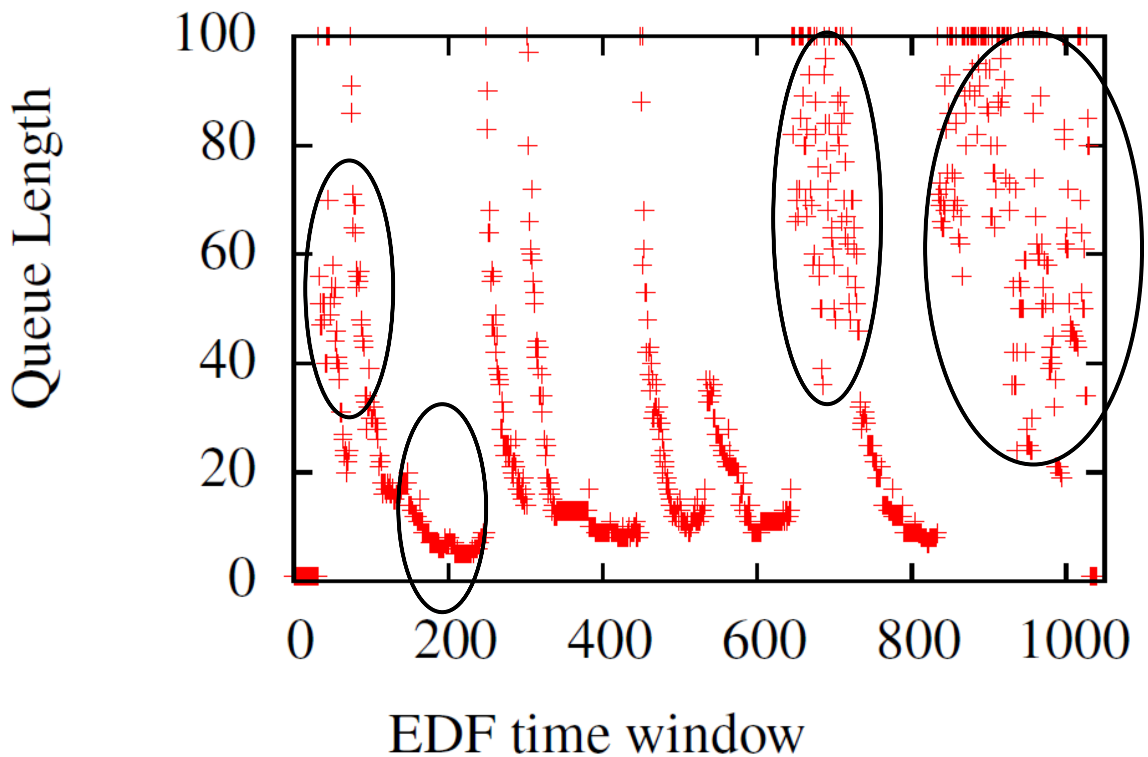
FLORIDA INTERNATIONAL UNIVERSITY

IORs' Issue Rates



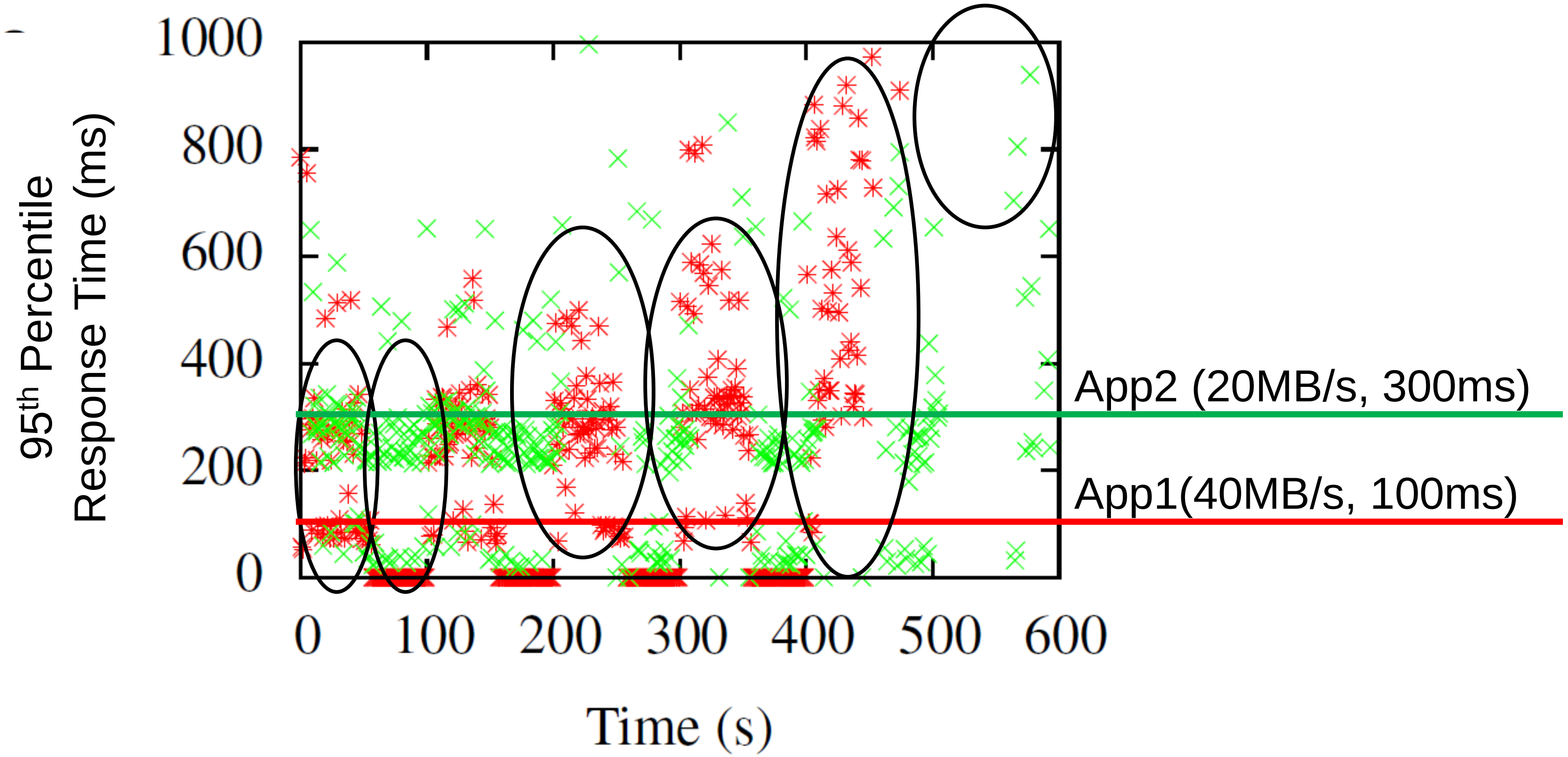
- One on-off pattern, with one constantly increasing
- Storage capacity is about 50MB/s
- App1 QoS: (40MB/s, 100ms); App2 QoS: (20MB/s, 300ms)

Adaptation of Queue Length



- Accurate transition between over- and under-load
- Good depth obtained for adequate throughput

Latency Differentiation



- Storage is overloaded when both Apps are on
  - App1 conforms to 100ms 10 times than App2
  - App1's overall 95th percentile latency is smaller than App2

Conclusions & Future Work

- Two-level I/O control for parallel storage
  - Two-level scheduler can effectively respect the latency needs of different applications
  - Latency can be managed using a feedback-control loop for a black box storage device
- Future work
  - Manage I/Os of different sizes
  - Create distributed versions of EDF

References

[1] PVFS2. http://www.pvfs.org/pvfs2/.
[2] PanFS. http://www.panasas.com .
[3] GPFS. http://www.ibm.com/systems/software/gpfs .
[4] Lustre. http://www.lustre.org .
[5] Y. Xu, D. Arteaga, M. Zhao, Y. Liu, R. Figueiredo, S. Seelam, "vPFS: Bandwidth Virtualization of Parallel Storage Systems", IEEE 28th Symposium on Mass Storage Systems and Technologies, 2012
[6] Yin Wang and Arif Merchant, "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.
[7] W. Jin, J. S. Chase, and J. Kaur, "Interposed Proportional Sharing For A Storage Service Utility," SIGMETRICS, 2004.
[8] IOR HPC Benchmark, http://sourceforge.net/projects/ior-sio/.
[9] NASA Parallel Benchmark, http://www.nas.nasa.gov/publications/npb.html .
[10] P. Welsh, P. Bogenschutz, "Weather Research and Forecast (WRF) Model: Precipitation Prognostics from the WRF Model during Recent Tropical Cyclones," Interdepartmental Hurricane Conference, 2005.
[11] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST," ClusterWorld Conf. and Expo, 2003.
[12] R. Sankaran, et al., "Direct Numerical Simulations of Turbulent Lean Premixed Combustion," Journal of Physics Conference Series, 2006.
[13] J. Zhang, A. Sivasubramaniam, Q. Wang, A. Riska, and E. Riedel, "Storage performance virtualization via throughput and latency control". Trans. Storage 2 (August 2006), 283–308.

Acknowledgement

- Sponsor: National Science Foundation
- VISA lab: http://visa.cs.fiu.edu
- More information: http://visa.cis.fiu.edu/hecura

Backup Slides