

# Adaptive Techniques for Achieving End-to-End QoS in the I/O Stack on Petascale Multiprocessors

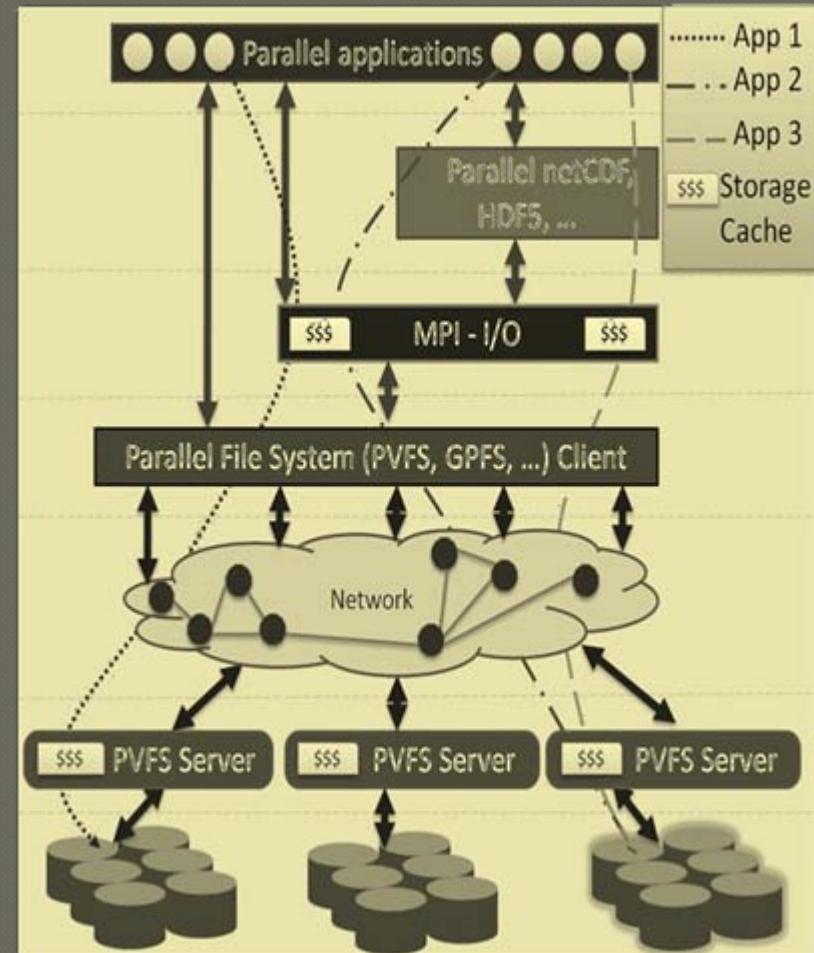
PIs: M. Kandemir<sup>†</sup>, P. Raghavan<sup>†</sup>, Q. Wang<sup>†</sup>, J. Dennis<sup>‡</sup>

<sup>†</sup>Pennsylvania State University

<sup>‡</sup>National Center for Atmospheric Research

# I/O Stack

- I/O stack is critical for data-intensive applications that target emerging systems of thousands of processors, I/O nodes, and disks
- Resources managed by the I/O stack (e.g., client/server caches, disk caches, network bandwidth, disk bandwidth, etc) are dynamically shared among various parallel applications, leading to possible destructive interferences



# Problems with the I/O Stack

---

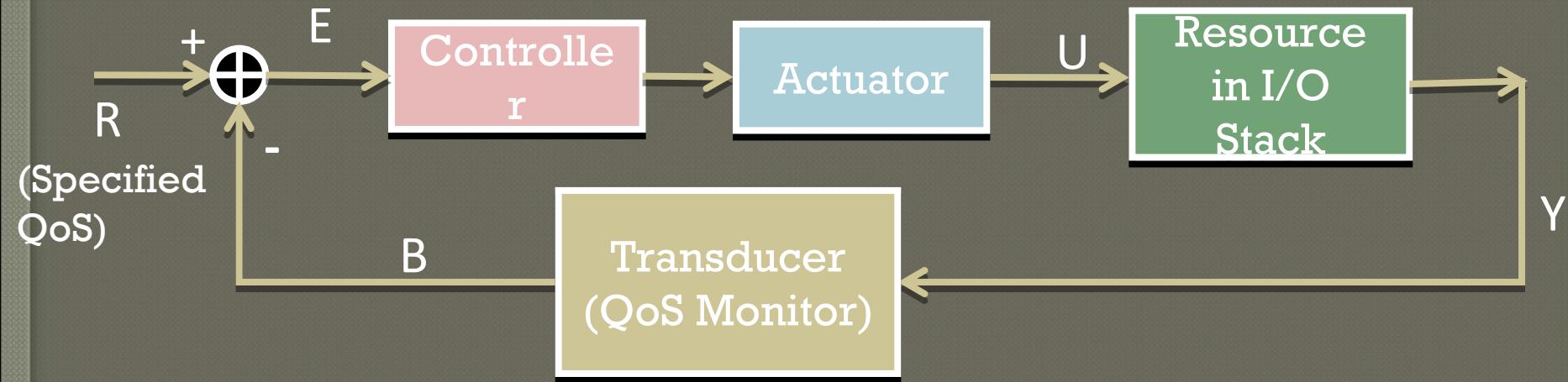
- No available mechanism that accommodates I/O stack-wide, application level QoS specification and management
  - A QoS metric specified at a high level needs to be mapped to sub-QoSs at lower layers
- No mechanism for monitoring QoS across different layers in the I/O stack
  - Such monitoring may be instrumental in identifying performance bottlenecks and pinpointing the reasons why the specified QoS cannot be satisfied
- No performance guarantees
  - Challenging since any layer can accommodate shared resources that serve different applications
- Not clear how one can best allocate available resources to maximize a global metric
  - The key here is to identify which applications can make better use of available shared resources, as compared to others

# Our Proposal

---

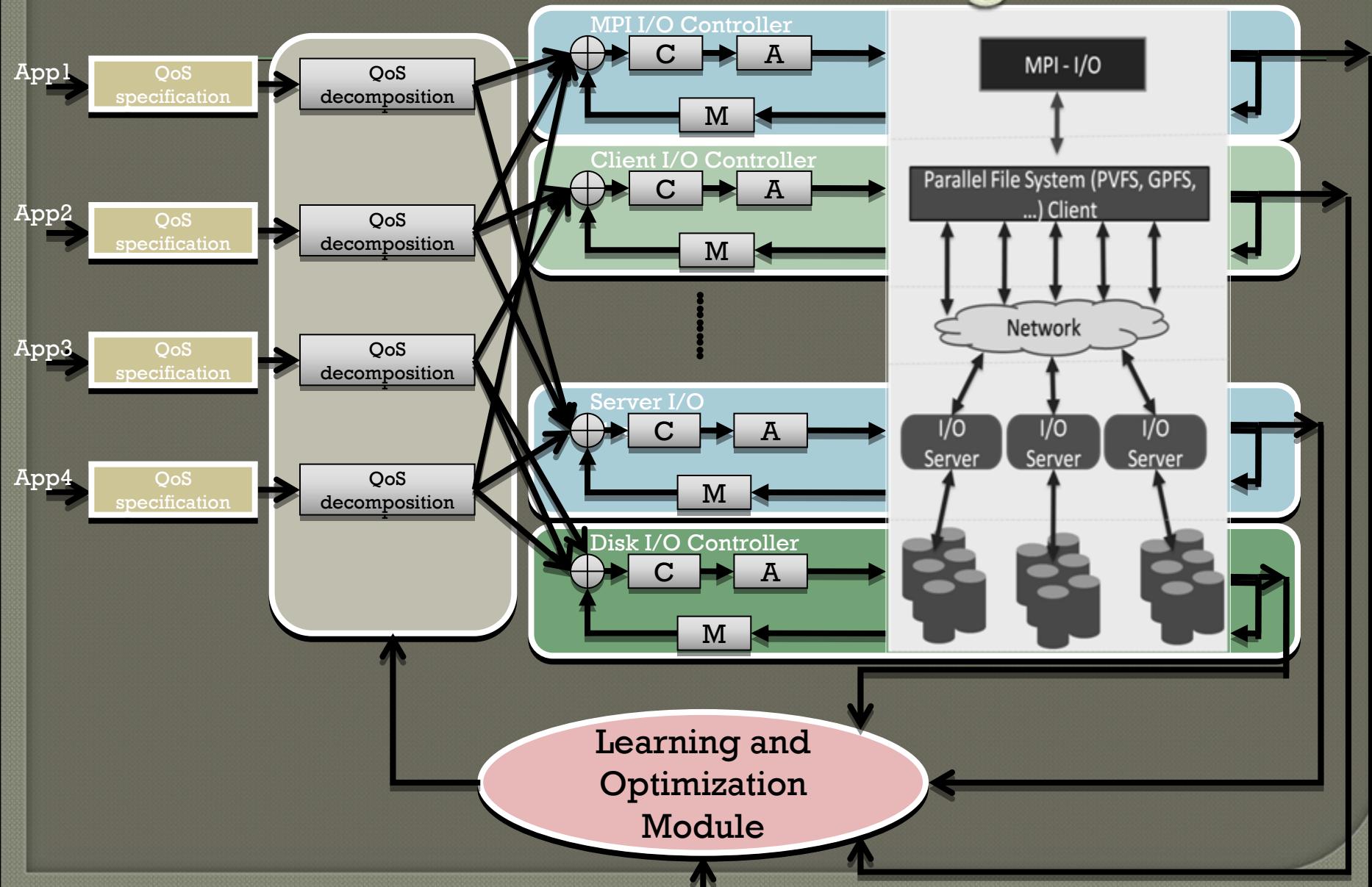
- A novel approach to the management of the I/O stack using feedback control theory, machine learning, and optimization. Specifically, we use
  - machine learning and optimization to determine the best decomposition of application-level QoS to sub-QoSs and also to distribute remaining resources across competing applications once the specified QoSs for the applications are satisfied
  - feedback control theory to allocate shared resources managed by the I/O stack such that the specified QoSs are satisfied throughout the execution.

# Feedback Control Loop



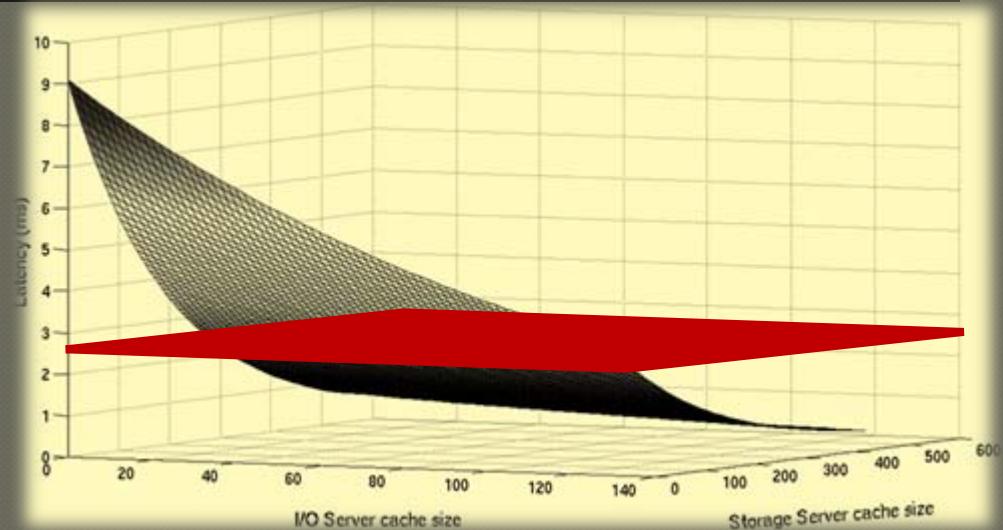
Controller makes the policy and Actuator enforces that policy using a mechanism such as resource allocator

# Big Picture

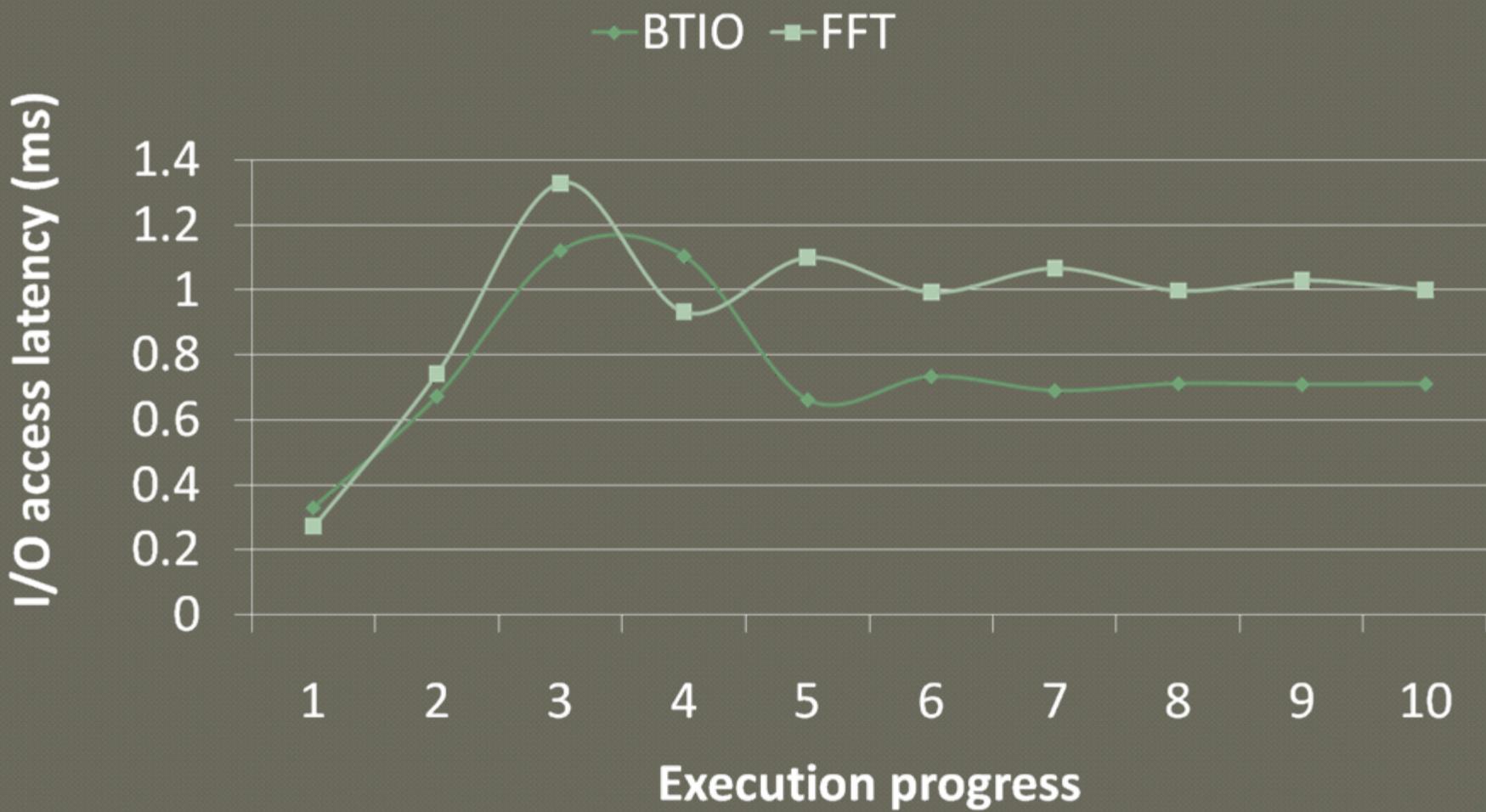


# Learning

- 3-D I/O latency regression-model
  - Per application
  - I/O server cache
  - Storage server cache
- Updated using real measurements
  - Captures dynamic changes in resource requirements (phase changes)
- SLO fulfillment
  - All points on the plane cutting the surface
  - All points below are also valid but limited to one application



# Preliminary Results



# Research Questions

---

- Resource requirement characterization from application end (UCAR, LBNL)
  - Different ways of satisfying a given requirement
- QoS distribution strategies across the I/O stack
  - Horizontal, vertical, hybrid
- High-level design issues
  - Number and types of control loops, assigning control loops to entities, (applications, metrics, physical resources), connections among loops
  - Where to implement control loops
  - Dynamic adaptation of control loops
- Controller design
  - PID, model predictive, custom, ...
- Actuator and transducer designs
  - Choice of knobs, collection of runtime statistics
- Learning and optimization
  - Choice of learning algorithms
  - Selection of global (inter-application) metrics and their optimization
  - Interactions between learning and control components