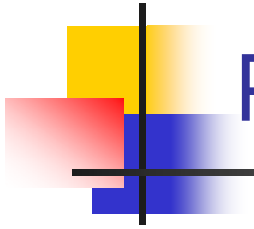
Performance Anomaly Identification for Cluster-based Parallel I/O Systems

Kai Shen

University of Rochester

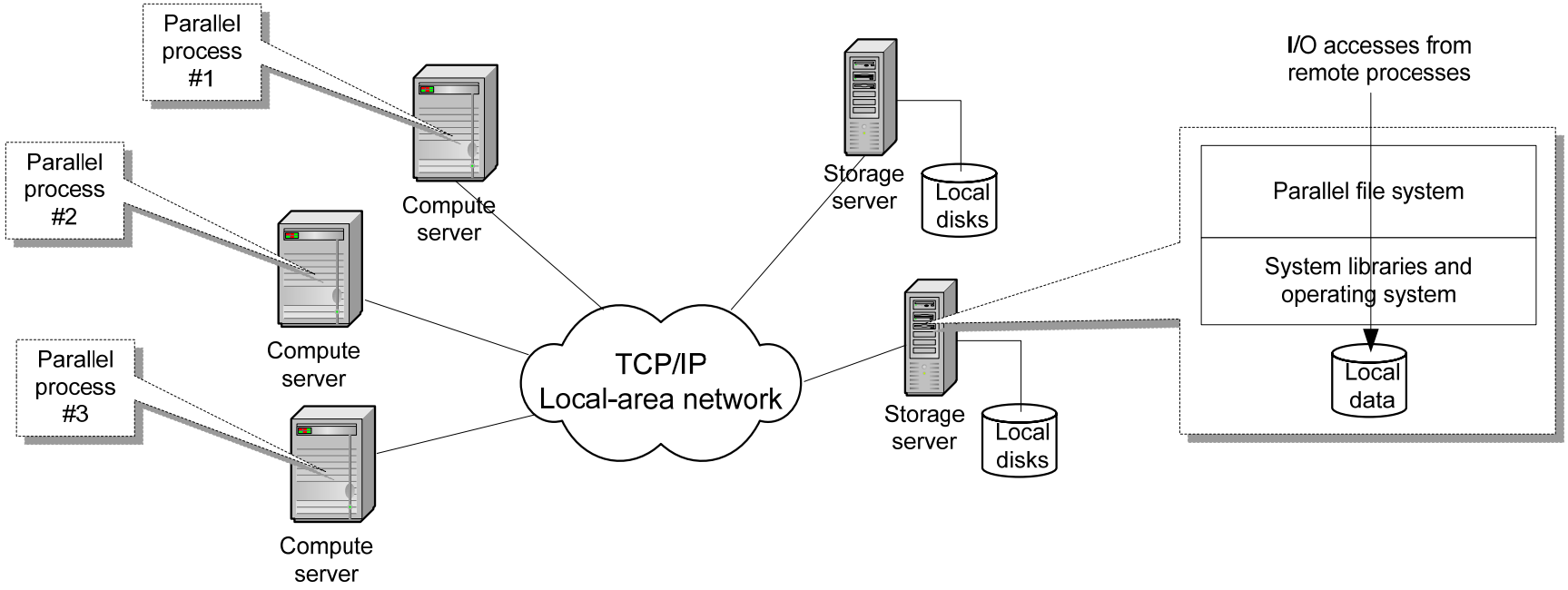


Project Overview

- Data-intensive high-end applications demand scalable I/O capability
- Parallel I/O systems are complex
  - multiple layers of system components
  - independent parallelism for I/O and computing

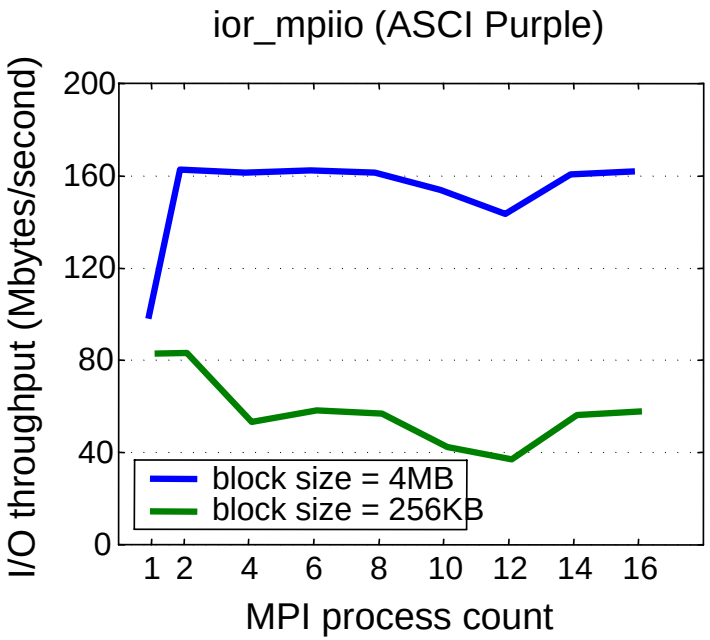
- Project goal:
  - recognize performance problems/anomalies in these systems
  - develop automatic techniques/approaches to identify the causes
- Focus on open systems
  - commonly available hardware (commodity storage devices and communication networks)
  - open-source (mostly general-purpose) software components

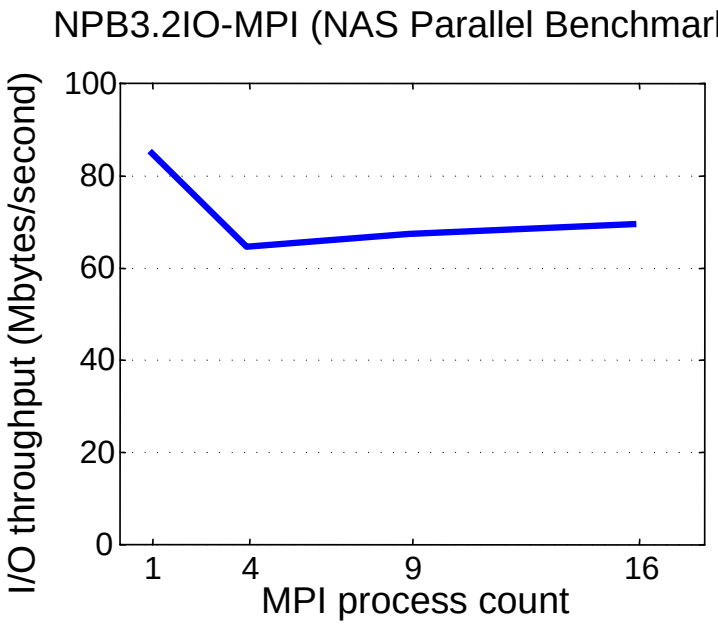
Architecture



- I/O workloads are transformed through many software layers
- Two independent flavors of parallelism leading to complex concurrent I/O issues

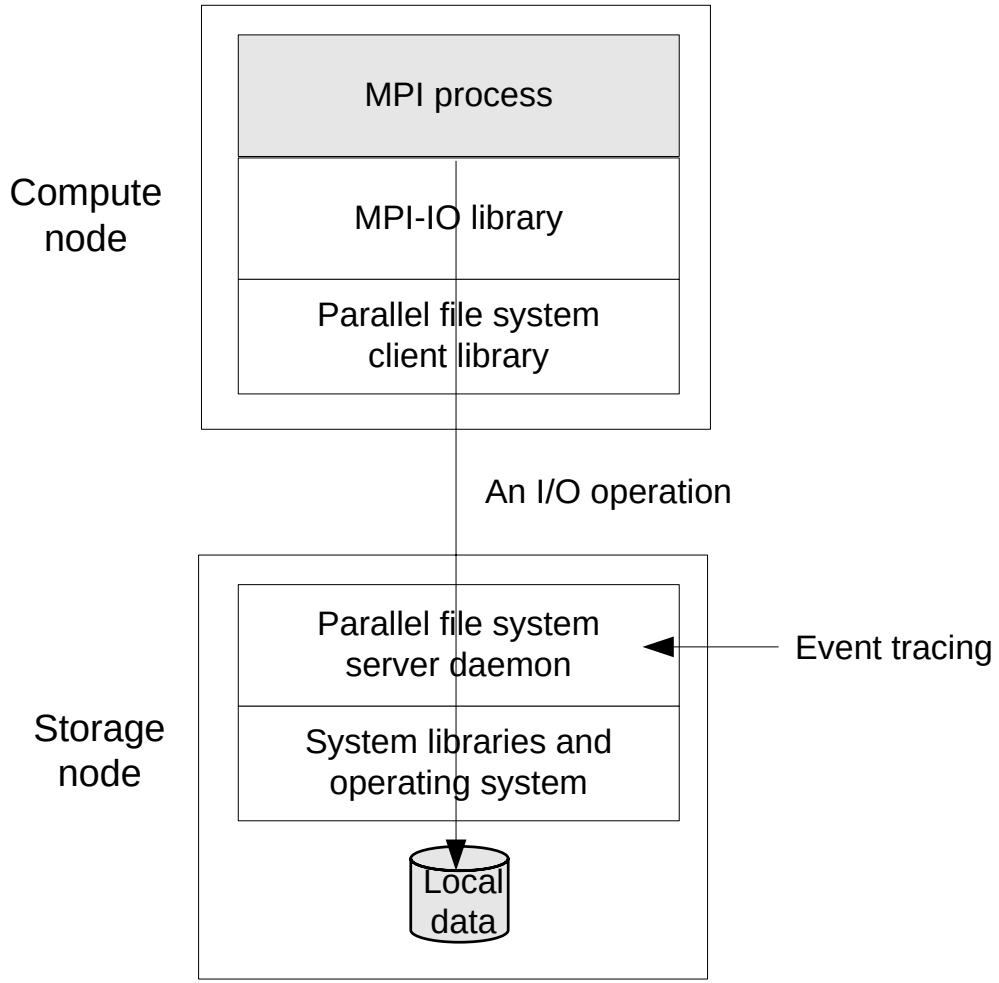
A Quantitative Example (I/O Read Throughput)



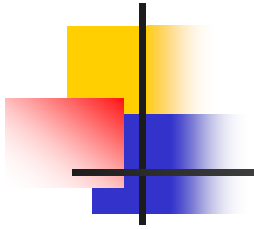


- Up to 16 compute nodes running MPICH2 (MPI-IO)
- 6 striped storage nodes running PVFS2; each run Linux 2.6.12
- Gigabit Ethernet (~80us TCP/IP roundtrip latency)

Anomaly Identification Through I/O Trace Analysis

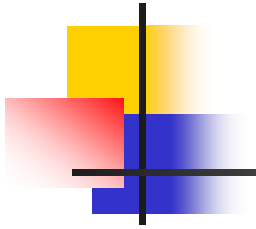


- Many layers of software present possible sources of problems
- In our ad hoc approach, we trace I/O events in PVFS server daemon
  - arrival of client I/O requests
  - issuance of I/O operations to lower layer
  - completion of I/O operations



Results of Trace Analysis

- Result #1: interleaved I/O under concurrent operations
- Further analysis within the operating system
  - prefetching in general-purpose OS is insufficient
  - anticipatory I/O scheduling does not work properly due to the lost of remote process context at storage nodes
- Result #2: slow return of I/O that should hit the cache
- Further analysis within the C library
  - PVFS uses one open file to issue I/O operations on the same file
  - all asynchronous I/O operations using the same open file are serialized by the C library



Summary and Future Direction

- Performance problems exist in parallel I/O systems
  - multiple layers of software interacting in complex ways
  - performance semantics are not well exposed through layer interface
  - problems often relate to parallelism and concurrency in the system
- Tracing and analysis at one software layer is insufficient

- Future direction
  - collecting system I/O traces at multiple layers and automatically extracting performance-sensitive characteristics:
    - access pattern, parallelism/concurrency, idleness, ...
  - across-layer synthesis of these characteristics may pinpoint problematic software layer