Parallel I/O of Parallel Programs

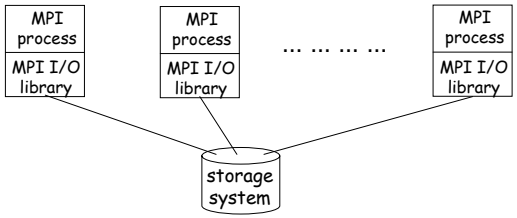
Kai Shen

Dept. of Computer Science, University of Rochester

I/O for Parallel Programs

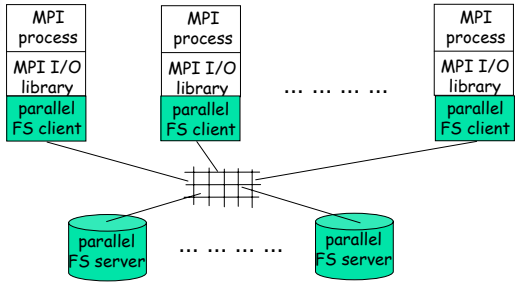
- Using a dedicated I/O node
  - all I/O is done at the dedicated I/O node; other nodes have to communicate with the I/O node for I/O
  - problem: scalability
- All nodes perform I/O simultaneously
  - synchronization
  - using special interface (e.g., MPI-I/O for MPI programs) to coordinate I/O from multiple nodes
- MPI-I/O
  - filetype, view, offset, displacement
  - http://www.mpi-forum.org/docs/mpi-20-html/node173.htm#Node173

System Architecture (as of now)



- ROMIO – one MPI I/O implementation (part of MPICH2)
  - Data sieving and collective I/O: addressing the inefficiency of non-sequential accesses on storage devices

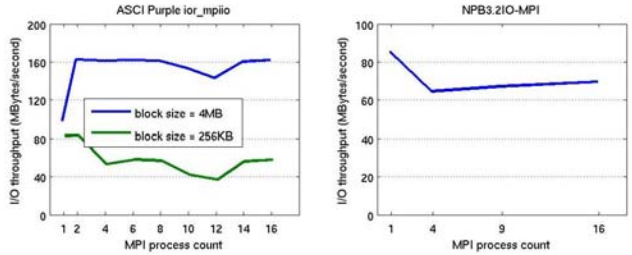
Parallel Storage and Parallel FS



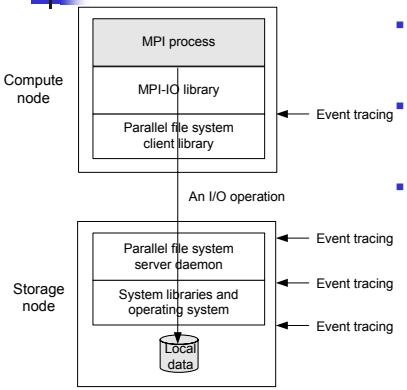
- Parallel file system
  - GPFS, PVFS
  - data striping, redundancy encoding, … …

A Quantitative Example

- 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)



Problem Diagnosis - I/O Trace Collection

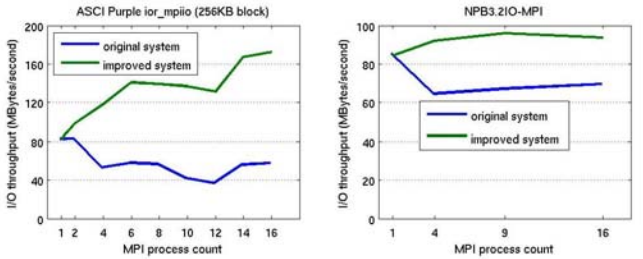


- Many layers of software present possible sources of problems
- Tracing at multiple layer boundaries of I/O stack help pinpoint problem sources
- Traced I/O characteristics
  - Sequentiality of I/O access pattern
  - I/O request size/granularity
  - I/O workload concurrency/idleness
  - I/O throughput
  - Wait between causal I/O events

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 [Iyer & Druschel 2001] 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

Performance Results After Problem Fixes



- Resolved anomalous performance degradation under concurrent I/O
- 39-156% throughput improvement for four applications

Summary

- Parallel I/O may deliver scalable performance
  - constrained by underlying storage device and networking performance
- Also, 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

Other Issues

- Coherence in parallel file system is difficult without cache snooping
  - probably provides no coherence support while letting applications to explicitly flush/invalidate
- Active storage
  - utilize the computation at storage nodes
  - reduce network I/O