Parallel I/O of Parallel Programs

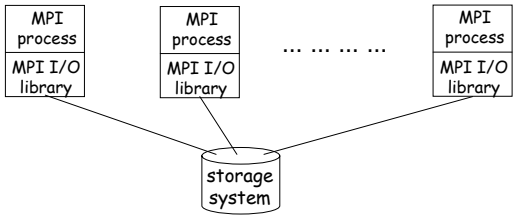
Kai Shen

Dept. of Computer Science, University of Rochester

I/O for Parallel Programs

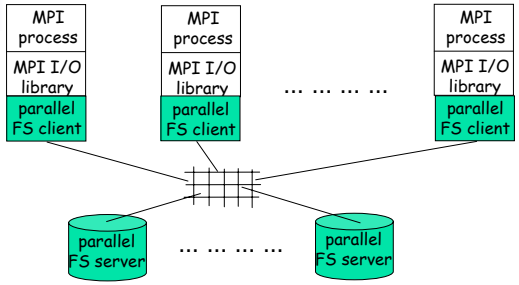
- Using a dedicated I/O node
  - all I/O is done at the dedicated I/O node; other nodes have to communicate with the I/O node for I/O
  - problem: scalability
- All nodes perform I/O simultaneously
  - synchronization
  - using special interface (e.g., MPI-I/O for MPI programs) to coordinate I/O from 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



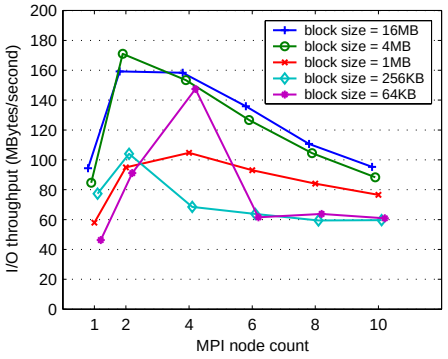
- Parallel file system
  - GPFS, PVFS
  - data striping, redundancy encoding, … …

Experimental Setup

- 14 machines (dual 2GHz Xeon, 2GB memory) connected with switched Gigabit Ethernet (TCP/IP roundtrip latency 80us)
  - 10 machines compute nodes
  - 4 machines as storage nodes (hard drive transfer rate at 33.8-66.0MB/sec)
- PVFS2
  - 64KB stripe block size
  - stock Linux 2.6 at each storage node
- MPICH2 (ROMIO)

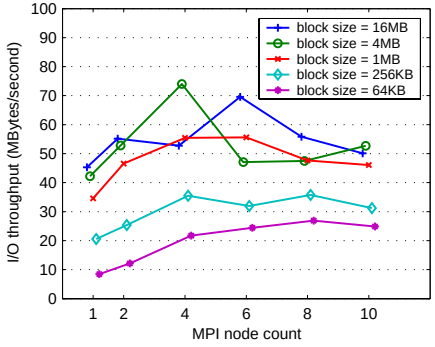
Read Throughput

- In an N-process MPI program, all processes open the same file and each process i (0<=i<N) reads blocks i, block N+i, block 2N+i, …



Synchronous Write Throughput

- In an N-process MPI program, all processes open the same file and each process i (0<=i<N) writes blocks i, block N+i, block 2N+i, …



Other Issues

- Coherence in parallel file system is difficult without cache snooping
  - probably provides no coherence support while letting applications to explicitly flush/invalidate
- Performance issues with meta-data management, non-bulk, and non-aligned data accesses
- Active storage
  - utilize the computation at storage nodes
  - reduce network I/O