Scalable I/O Middleware and File System Optimizations for High-Performance Computing

Alok Choudhary, Professor
Director: Center for Ultra-Scale Computing and Security
Dept. of Electrical & Computer Engineering
And Kellogg School of Management
Northwestern University
choudhar@ece.northwestern.edu

Collaborators: Prof. M. Kandemir (Penn State) and Dr. R. Thakur (ANL)

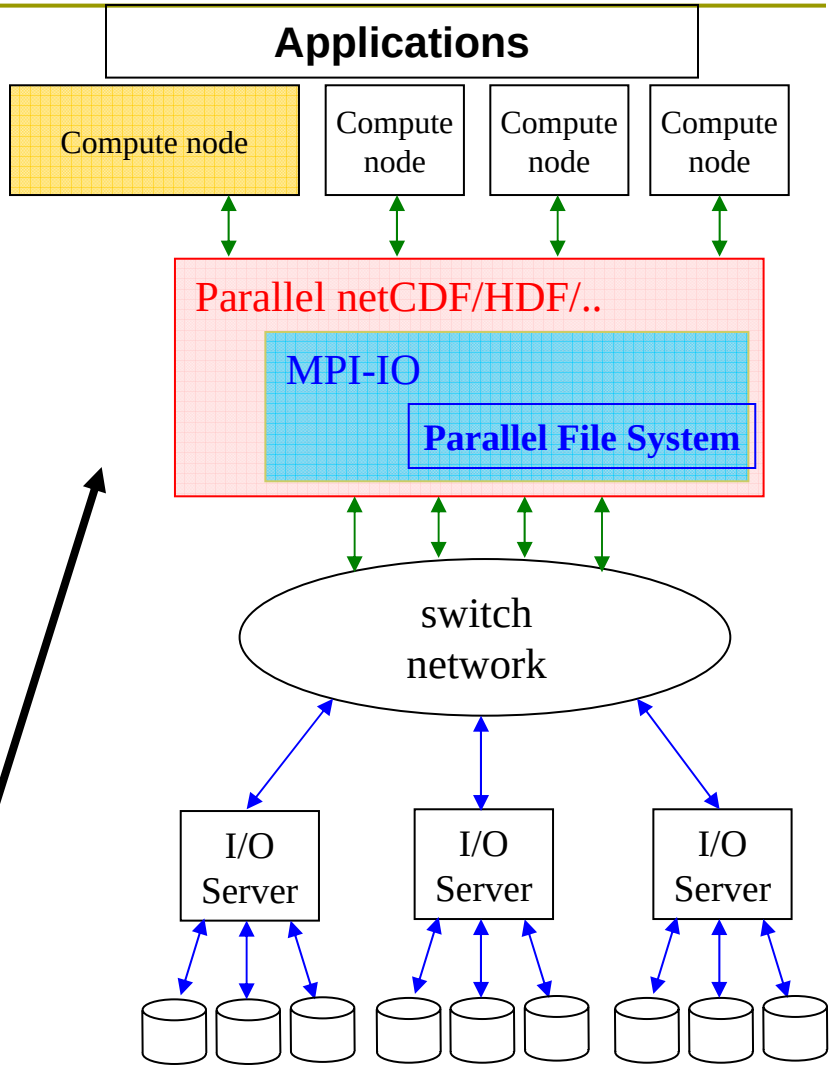
Objectives

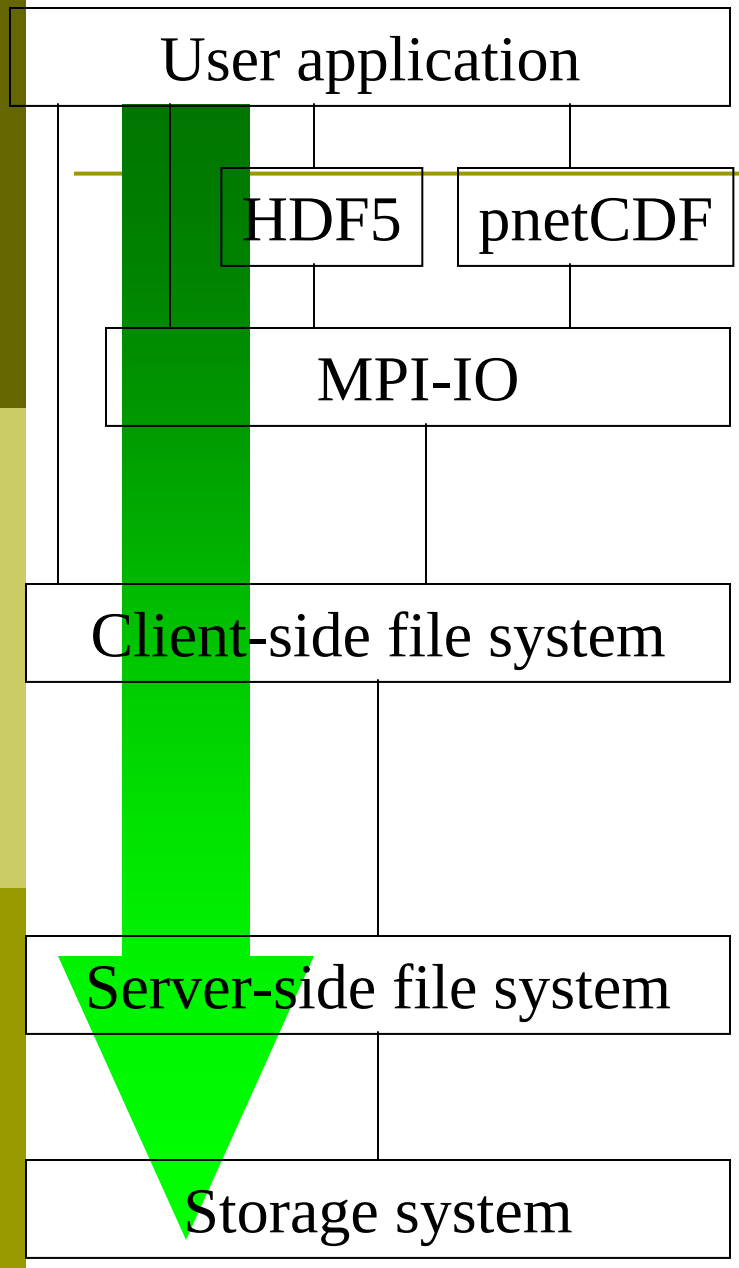
- Middleware Caching optimizations
- Application oriented benchmark kernels

Typical Software Layers for I/O in HEC

- Based on a lot of current apps
- High-Level
  - E.g., NetCDF, HDF, ABC
  - Applications use these
- Mid-level
  - E.g., MPI-IO
  - Performance experience
- Low Level
  - E.g., File Systems
  - Critical for performance in above

End-to-End Performance critical





Access patterns: shared files, individual files, data partitioning, check-pointing, data structures, inter-data relationship

Data types (byte-alignment), data structures (flexible dimensionality), hierarchical data model

Collectives, independents
I/O hints: access style (read_once, write_mostly, sequential, random, …), collective buffering, chunking, striping

Caching, fault tolerance, read-ahead, write-behind, I/O load balance, wide-area, heterogeneous FS support, thread-safe

Open mode (O_RDONLY, O_WRONLY, O_SYNC), file status, locking, flushing, cache invalidation
Machine dependent: data shipping, sparse access, double buffering

application-aware caching, pre-fetching, file grouping, "vector of bytes", flexible caching control, object-based data alignment, memory-file layout mapping, more control over hardware, Shared file descriptors,

Read-ahead, write-behind, metadata management, file striping, security, redundancy

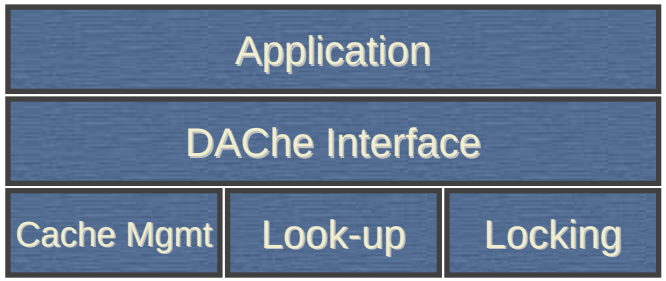
Group locks, flexible locking control, scalable metadata management, zero-copying, QoS, Shared file descriptors,

Access base on : file blocks, objects Scheduling, aggregation

Active storage: data filtering,object-based/hierarchical storage management, indexing, mining, power-management

Middleware Caching: Direct Access Cache System (DACche)

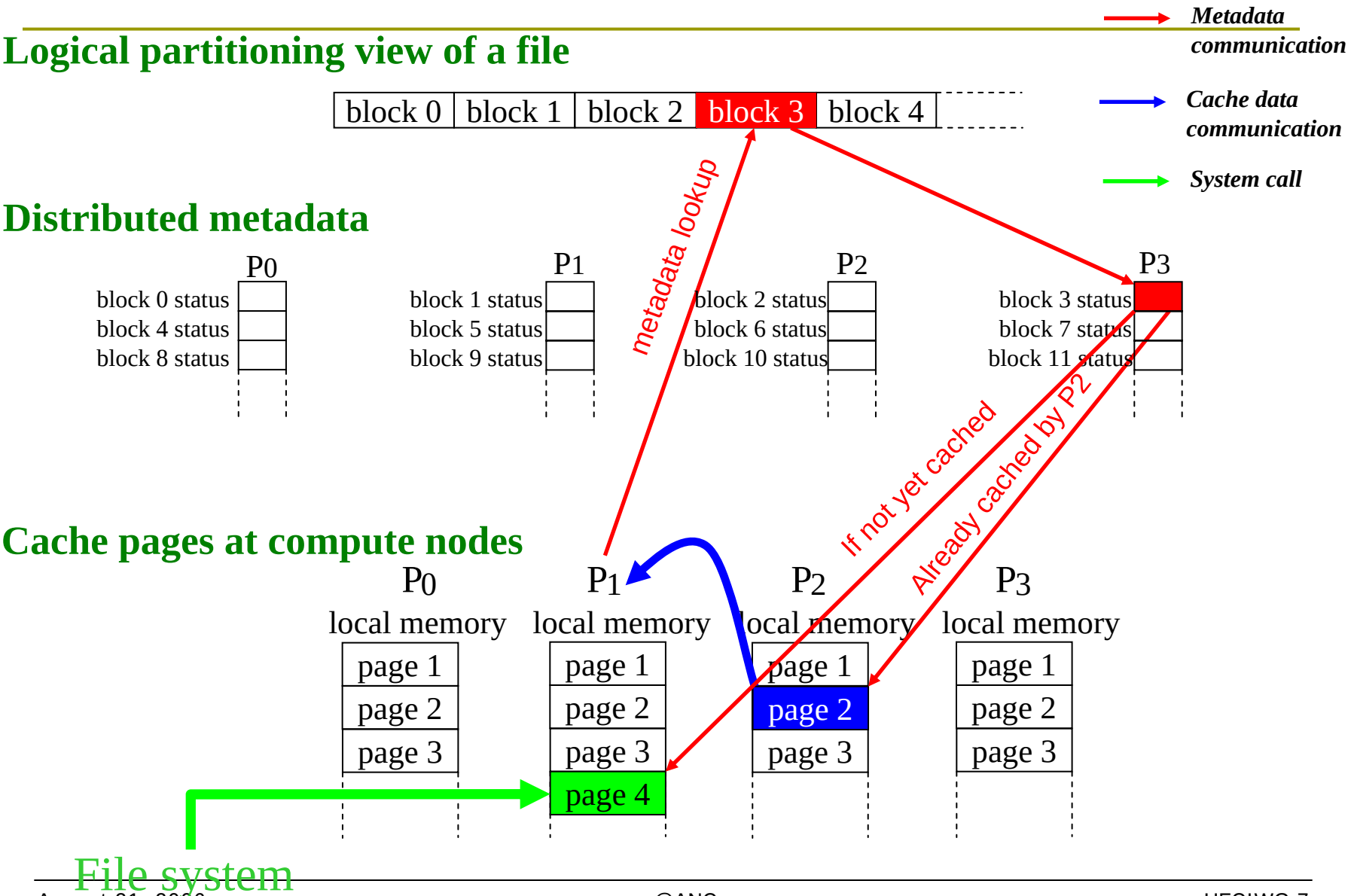
- Main Idea: Runtime Cache in user space to capture small, irregular accesses
- Portable
- 4 main subsystems
  - I/O interface and protocol
  - Cache Management
  - Look-up management
  - Locking Subsystem



Example Design & Implementation

- Global cache metadata management
  - A file is logically divided into blocks
  - Metadata of blocks are distributed across all MPI clients
  - Block-based file locking is used for consistency control
- Local cache page management
  - At most one copy of file data can be cached at any time
  - Handling local/remote requests to locally cached data
  - Page eviction is based on local reference entirely
  - Page migration is triggered by global metadata reference

Read Example

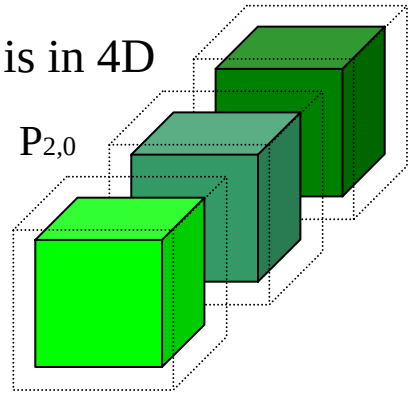


Illustrative Performance Evaluation

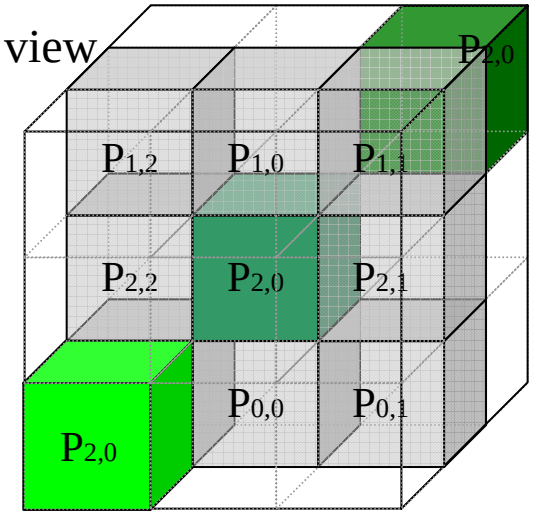
- Platforms
  - Tungsten, a Linux cluster @ NCSA
    - Lustre parallel file system
  - Mercury, an IBM Linux cluster @ NCSA
    - GPFS parallel file system
- MPICH
  - Caching added at ADIO layer

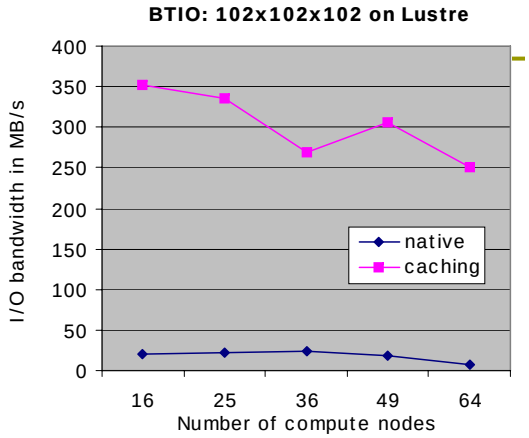
BTIO

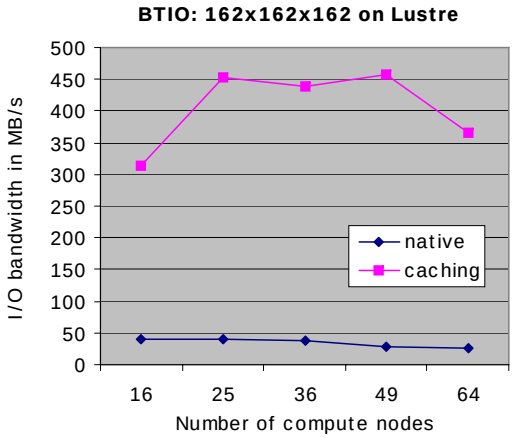
Local array is in 4D

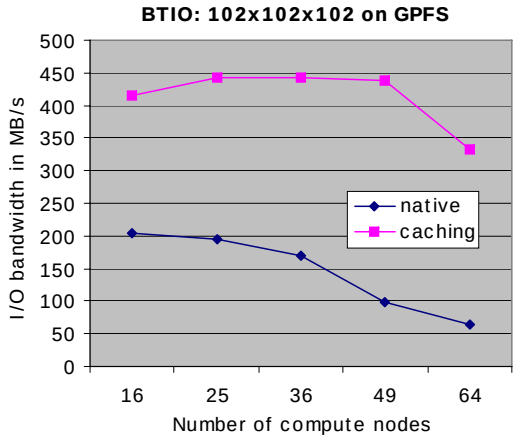


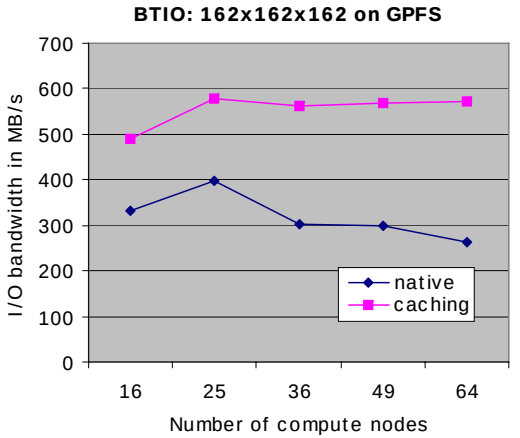
File view











- Block tri-diagonal array partitioning
- writes followed by reads
- I/O amount:
  - Class B (102^3) is 1.6 GB
  - Class C (162^3) is 6.5 GB

Aligning +Caching Illustration