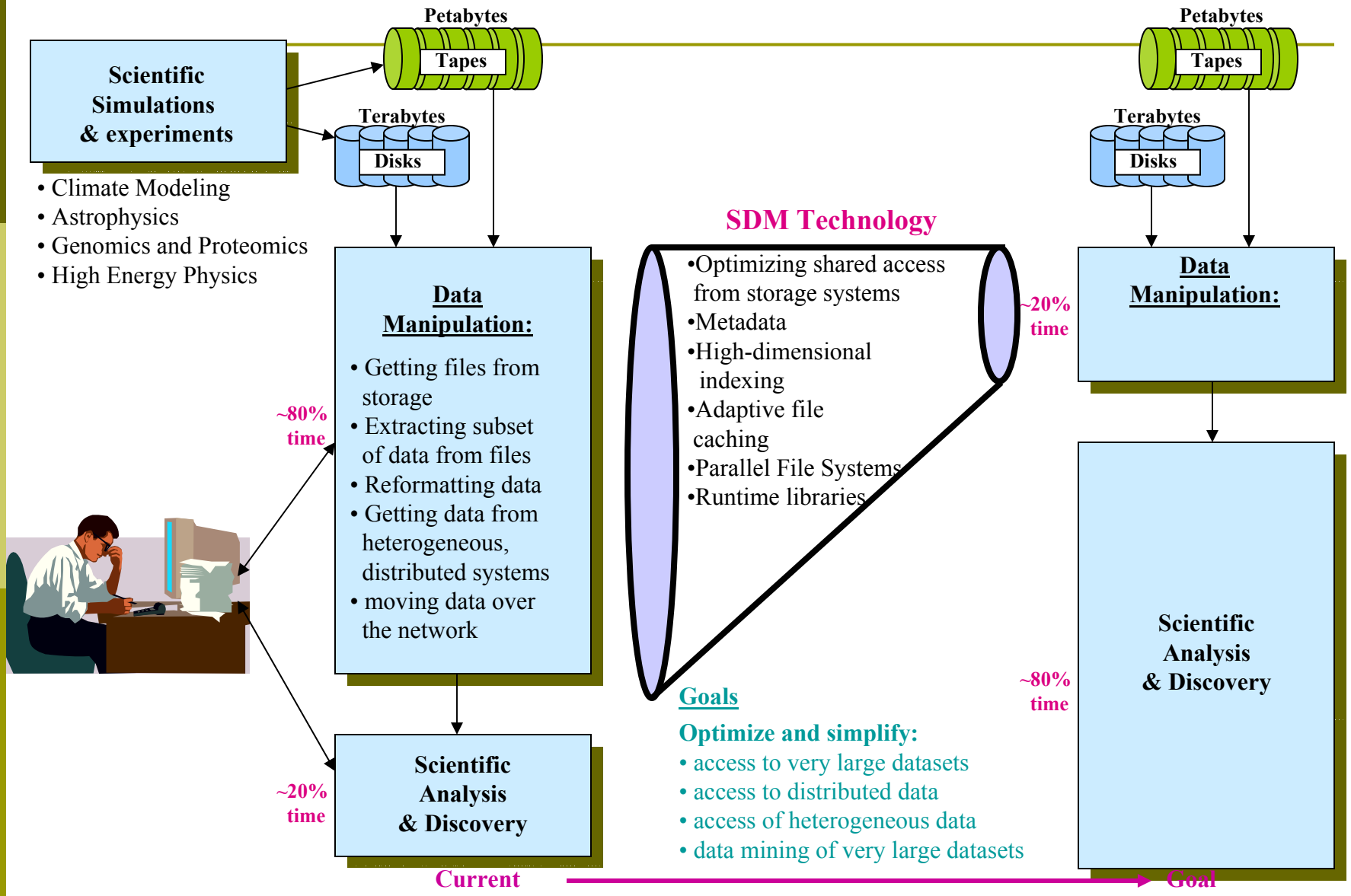
Scalable I/O: Research to Reach the Tipping Point

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

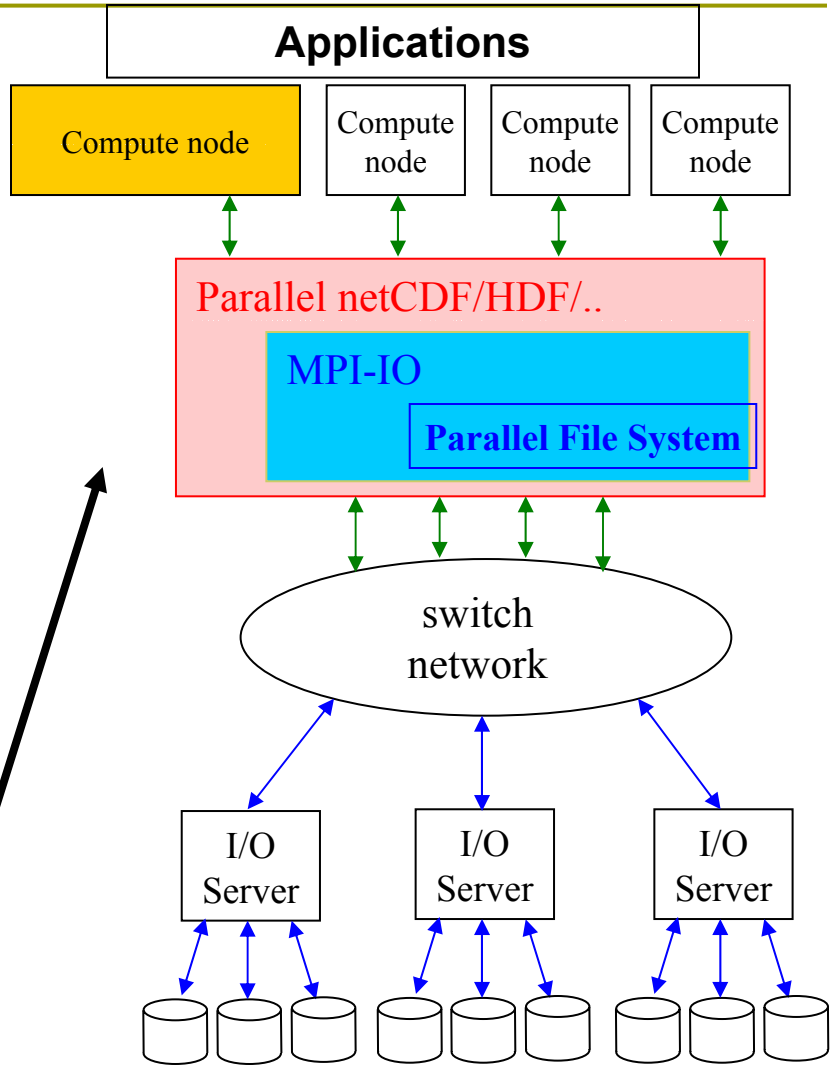
Consider Productivity and Performance



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

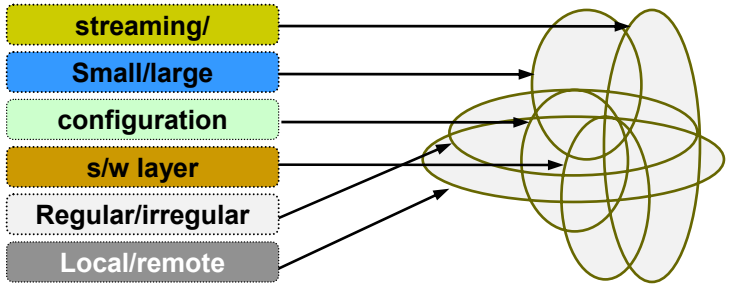
End-to-End Performance critical



I/O – Complex Optimization Space

User specifies how

Complex non-portable optimization space



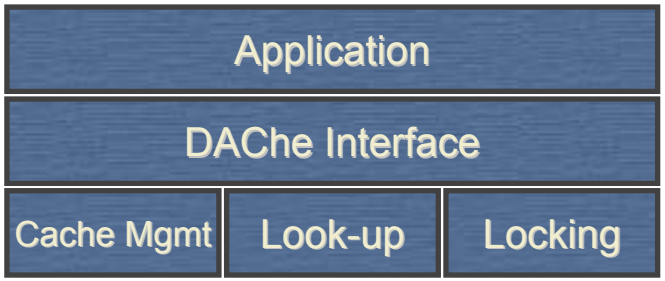
- **user burdened**
- **Ineffective interfaces**
- **Non-communicating layers**
- **Reactive**

"Currently data handling, I/O, storage (speed), analysis is the main bottleneck. It is known how to scale computations based on processing power and memory. Subsequent phases are a bottleneck due to h/w and s/w infrastructure" An App scientist

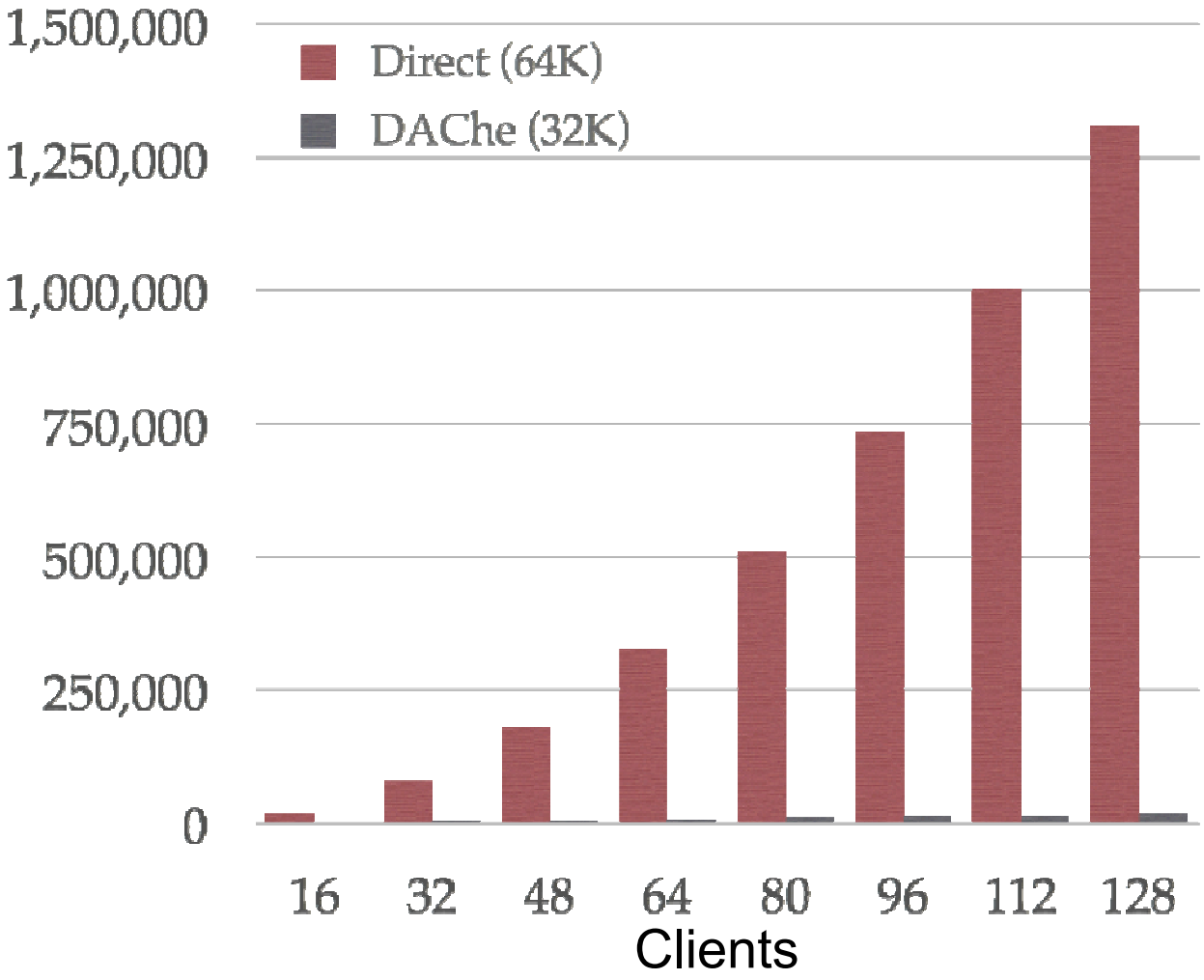
Current Research Example: Direct Access Cache System (DACche)

ISC2005

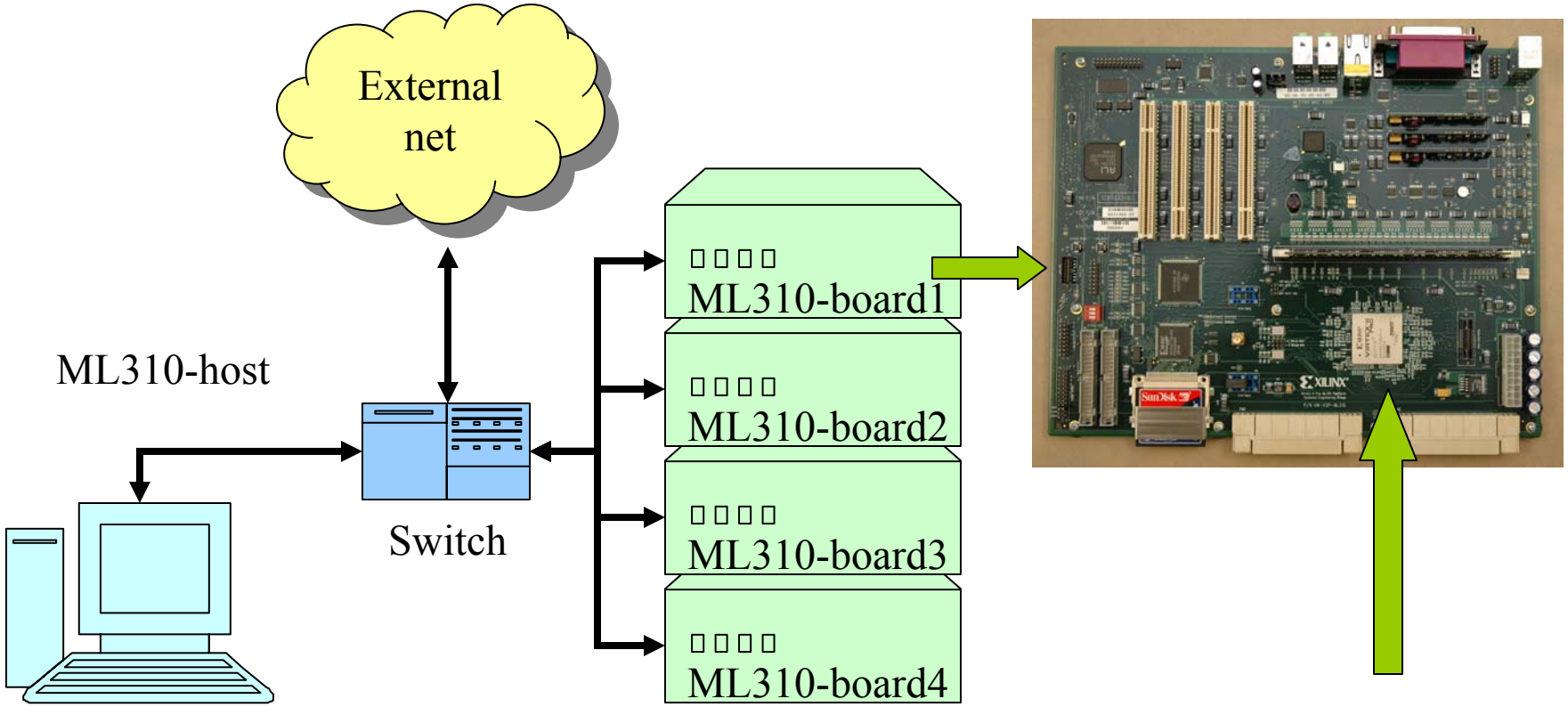
- Main Idea: Runtime Cache in user space to capture small, irregular accesses
- Portable
- 4 main subsystems
  - I/O interface and protocol
  - Cache Management
  - Look-up management
  - Locking Subsystem



File System Calls



Active Storage System (reconfigurable system)



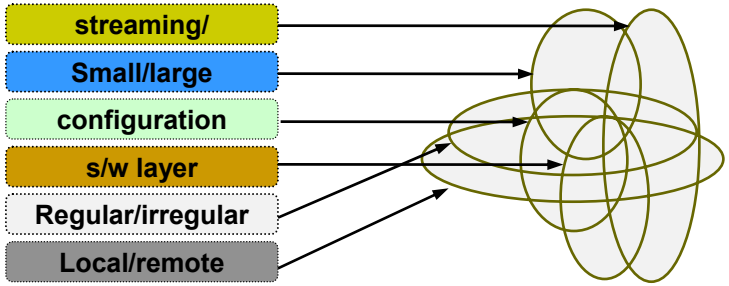
- Software:
  - Data Mining
  - Encryption
  - Functions and runtime libs
  - Linux micro-kernel

- Xilinx XC2VP30 Virtex-II Pro family
  - 30,816 logic cells (3424 CLBs)
  - 2 PPC405 embedded cores
  - 2,448 Kb (136 18 Kb blocks) BRAM
  - 136 dedicated 18x18 multiplier blocks

Research Directions

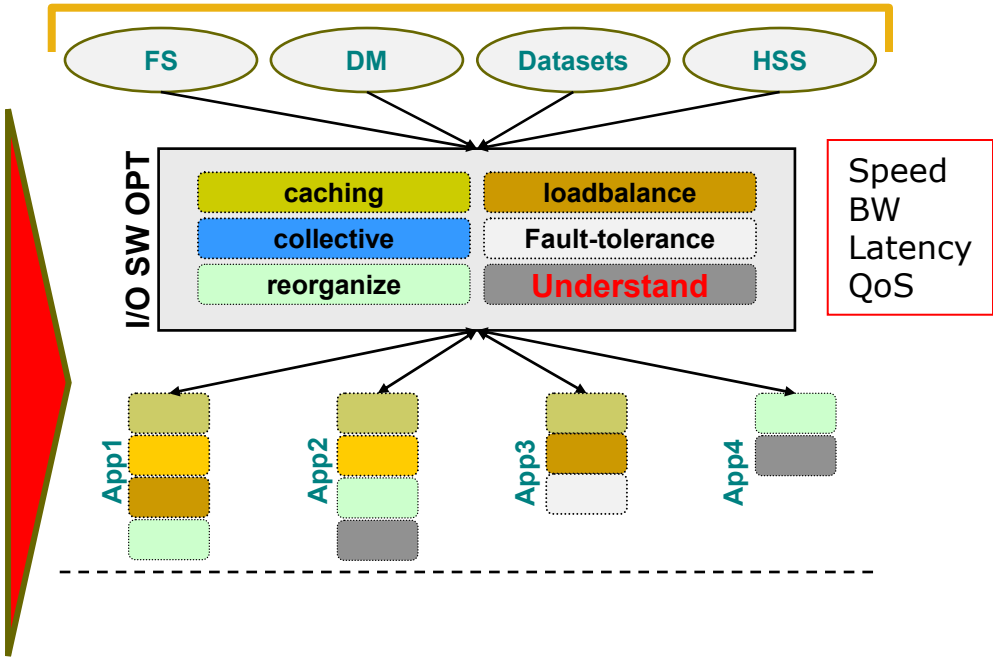
Decouple "What" from "How" and Be Proactive

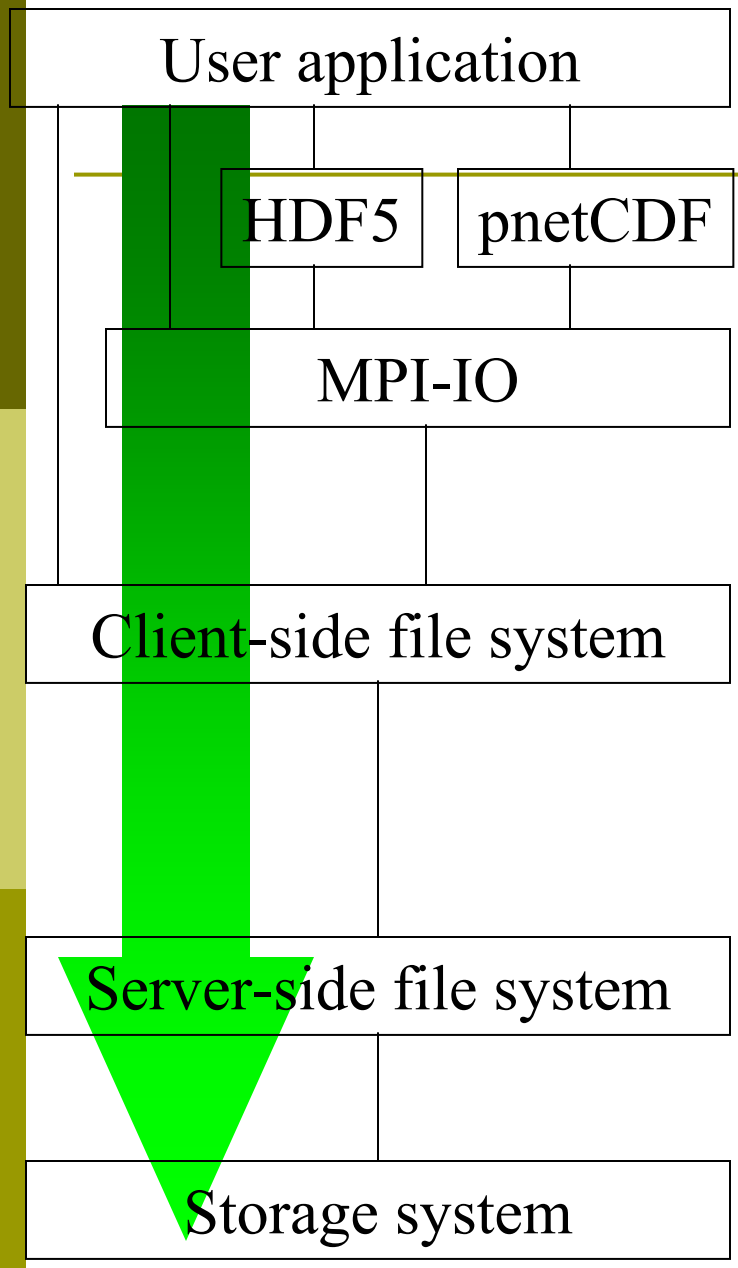
Current



- **user burdened**
- **Ineffective interfaces**
- **Non-communicating layers**

Goal





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,

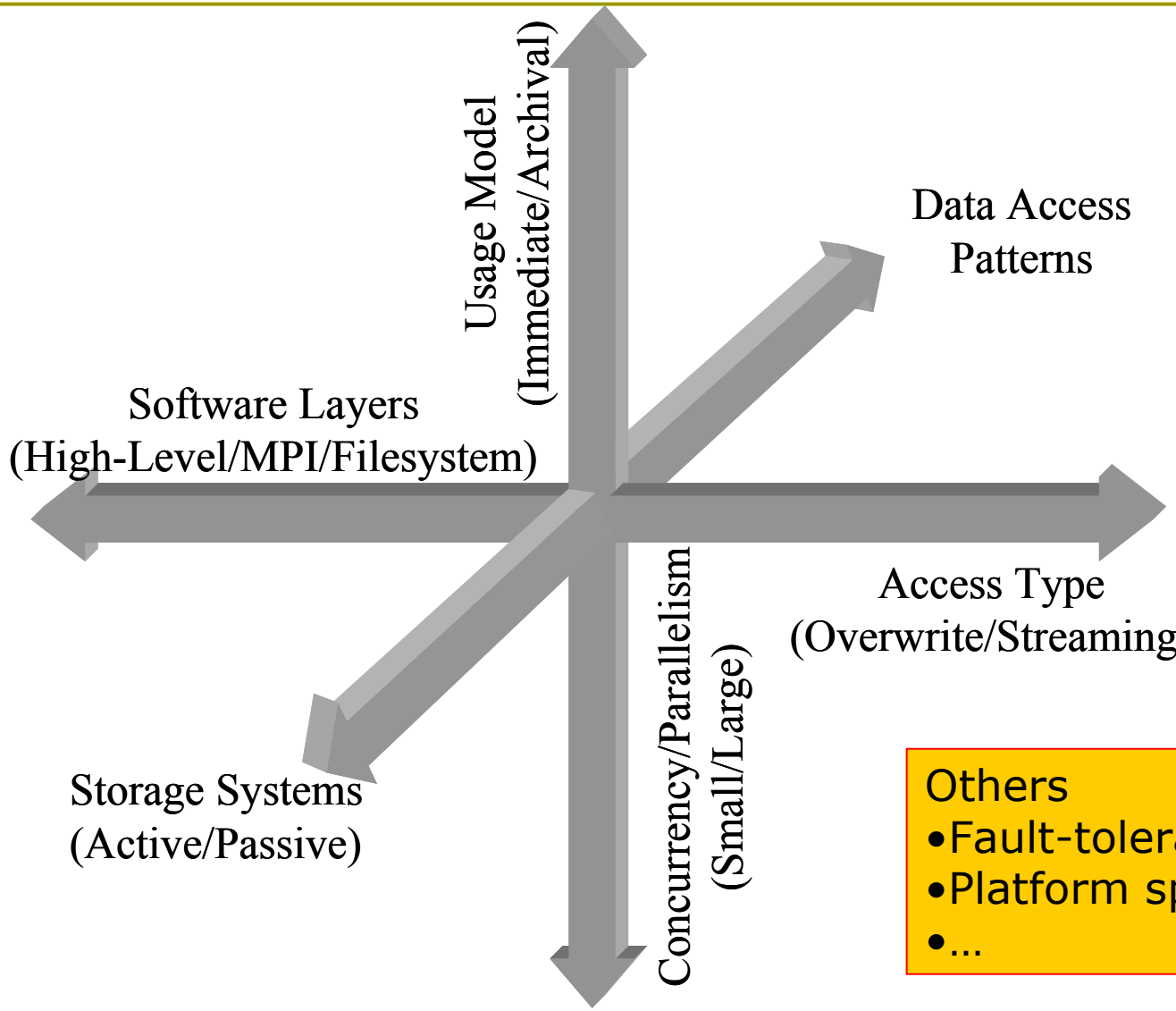
Read-ahead, write-behind, metadata management, file striping, security, redundancy

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

Some Complexity Dimensions



Others
- Fault-tolerance
- Platform specific
- …