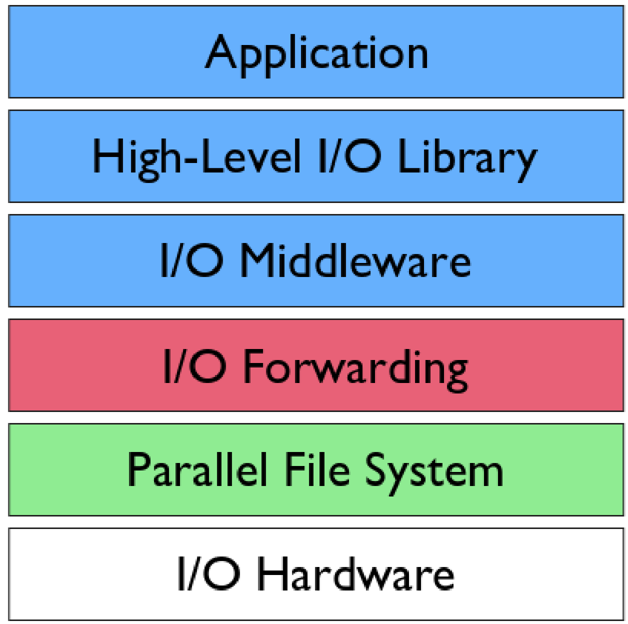
I/O for Computational Science

High-Level I/O Library
maps application abstractions onto storage abstractions and provides data portability.

HDF5, Parallel netCDF, ADIOS

I/O Forwarding
bridges between app. tasks and storage system and provides aggregation for uncoordinated I/O.

IBM ciod, IOFSL, Cray DVS



I/O Middleware
organizes accesses from many processes, especially those using collective I/O.

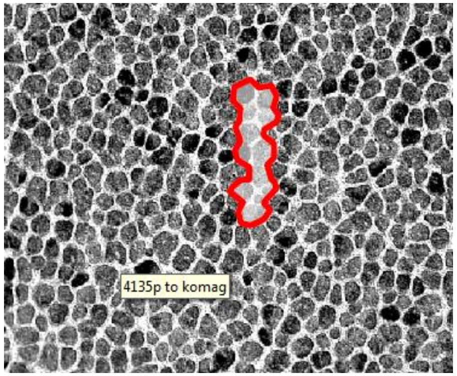
MPI-IO

Parallel File System
maintains logical space and provides efficient access to data.

PVFS, PanFS, GPFS, Lustre

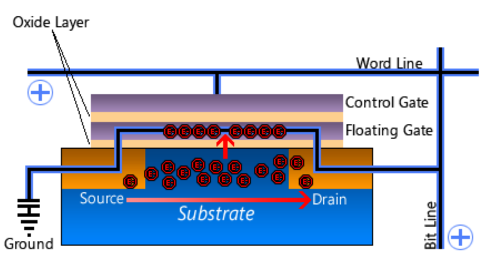
Additional I/O software provides improved performance and usability over directly accessing the parallel file system. Reduces or (ideally) eliminates need for optimization in application codes.

I/O Hardware



Magnetic or Solid State storage bits

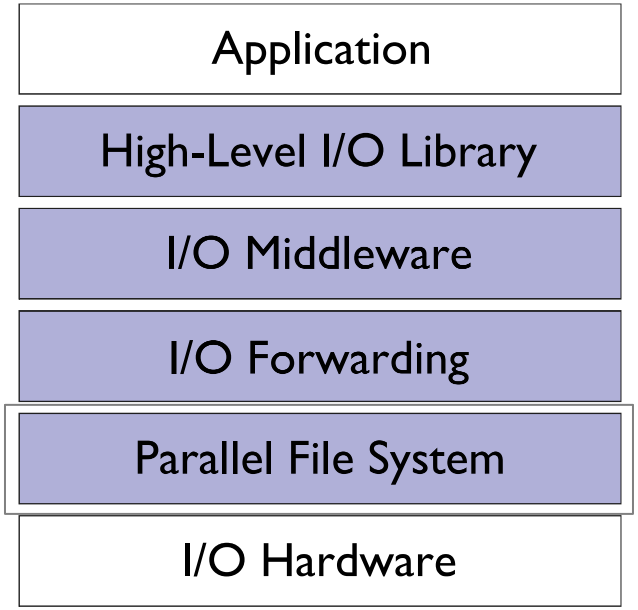
Storage Devices



Application
High-Level I/O Library
I/O Middleware
I/O Forwarding
Parallel File System
I/O Hardware

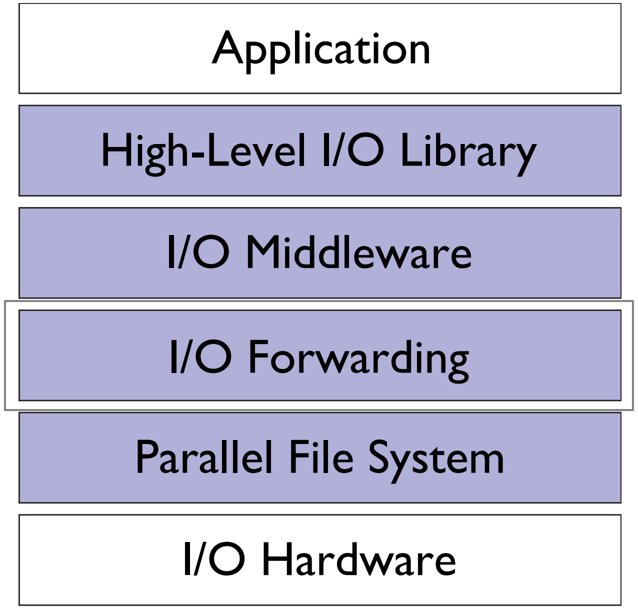
Characteristics of Storage Devices affect performance, reliability, and system design

Parallel File System



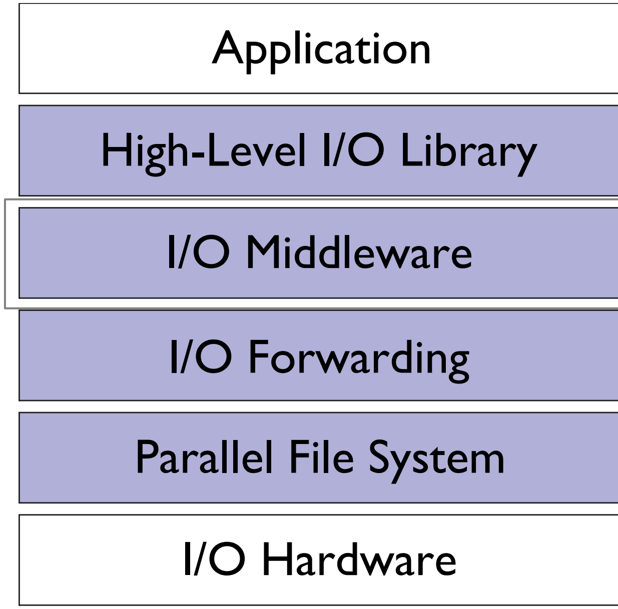
- Manage storage hardware
  - Present single view
  - Stripe files for performance
- In the I/O software stack
  - Focus on concurrent, independent access
  - Publish an interface that middleware can use effectively
    - Rich I/O language
    - Relaxed but sufficient semantics

I/O Forwarding



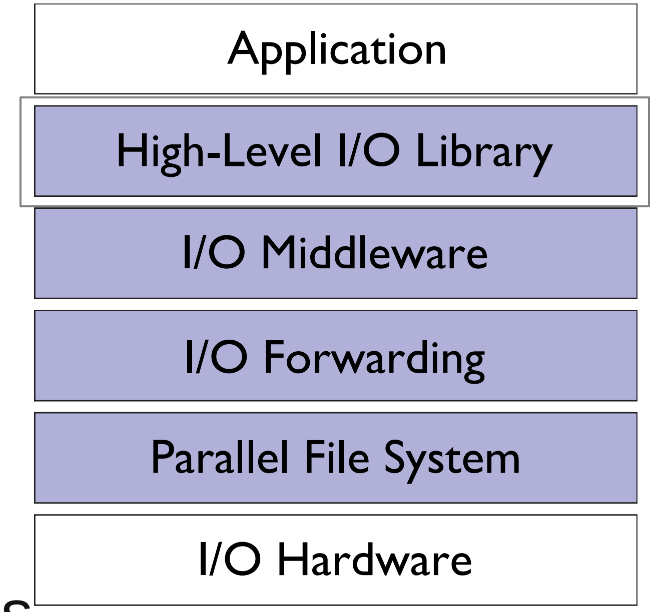
- Present in some of the largest systems
  - Provides bridge between system and storage in machines such as the Blue Gene/P
- Allows for a point of aggregation, hiding true number of clients from underlying file system
  - Also allows in-situ processing
- Poor implementations can lead to unnecessary serialization, hindering performance

I/O Middleware



- Match the programming model (e.g. MPI)
- Facilitate concurrent access by groupsof processes
  - Collective I/O
  - Atomicity rules
- Expose a generic interface
  - Good building block for high-level libraries
- Efficiently map middleware operations into PFS ones
  - Leverage any rich PFS access constructs, such as:
    - Scalable file name resolution
    - Rich I/O descriptions

High Level Libraries



- Match storage abstraction to domain
  - Multidimensional datasets
  - Typed variables
  - Attributes
- Provide self-describing, structured files
- Map to middleware interface
  - Encourage collective I/O
- Implement optimizations that middleware cannot, such as
  - Caching attributes of variables
  - Chunking of datasets

What we've said so far…

- Application scientists have basic goals for interacting with storage
  - Keep productivity high (meaningful interfaces)
  - Keep efficiency high (extracting high performance from hardware)
- Many solutions have been pursued by application teams, with limited success
  - This is largely due to reliance on file system APIs, which are poorly designed for computational science
- Parallel I/O teams have developed software to address these goals
  - Provide meaningful interfaces with common abstractions
  - Interact with the file system in the most efficient way possible