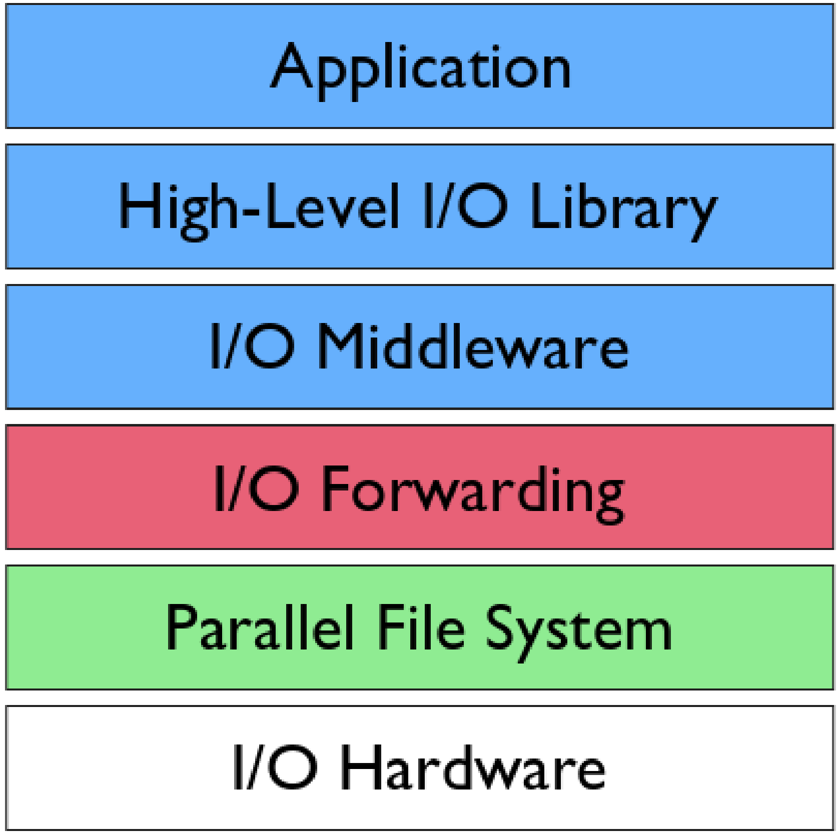
# I/O Stack

**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*



Application

High-Level I/O Library

I/O Middleware

I/O Forwarding

Parallel File System

I/O Hardware

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

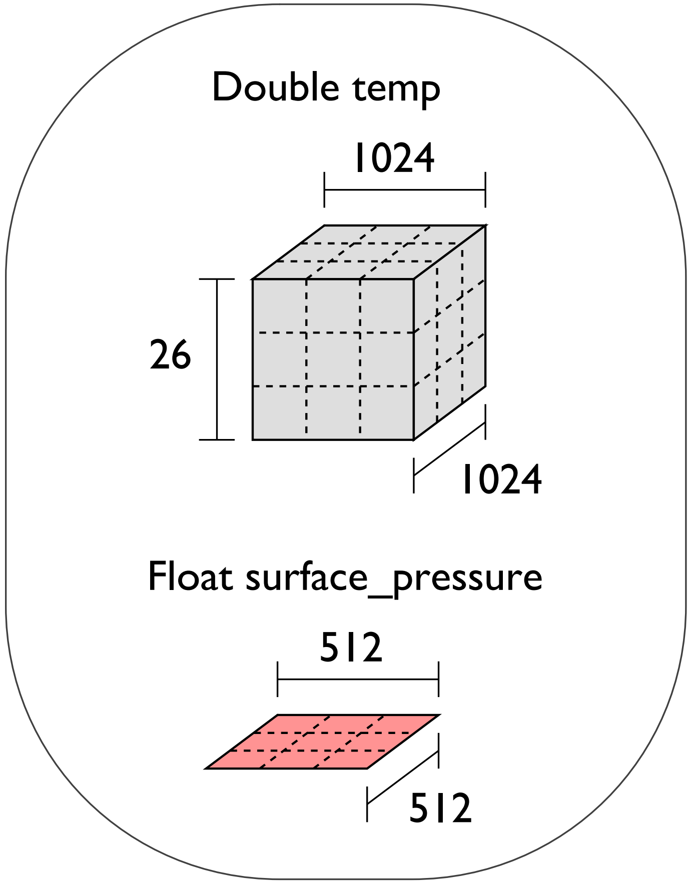
*MPI-IO*

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

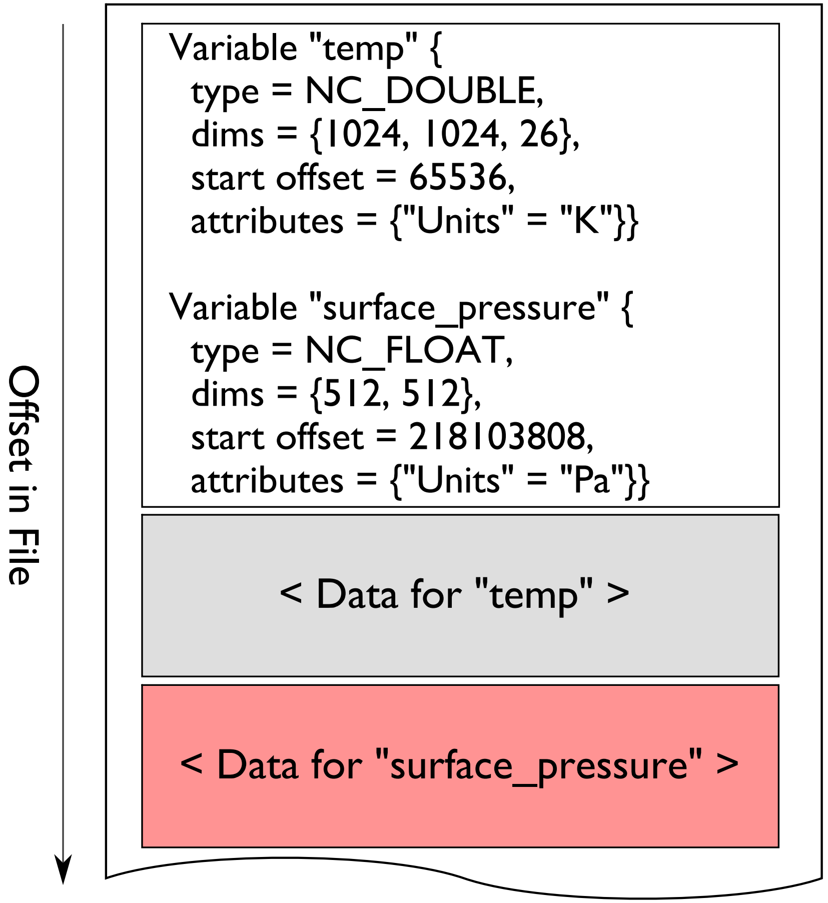
*PVFS, PanFS, GPFS, Lustre*

# High-level I/O Libraries: Structured Containers

## Application Data Structures



## netCDF File "checkpoint07.nc"



```
Variable "temp" {
  type = NC_DOUBLE,
  dims = {1024, 1024, 26},
  start offset = 65536,
  attributes = {"Units" = "K"}}

Variable "surface_pressure" {
  type = NC_FLOAT,
  dims = {512, 512},
  start offset = 218103808,
  attributes = {"Units" = "Pa"}}
```

< Data for "temp" >

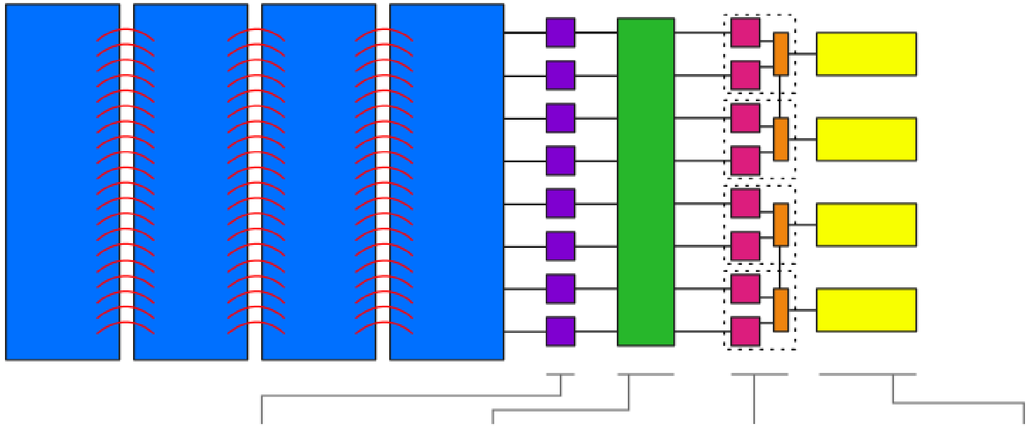
< Data for "surface_pressure" >

netCDF header describes the contents of the file: typed, multi-dimensional variables and attributes on variables or the dataset itself.

Data for variables is stored in contiguous blocks, encoded in a portable binary format according to the variable's type.

# How Things Go Wrong (1/4): Poorly Distributed Work

- Just like an application, load balance is key
  - Network
  - Storage devices
- Storage system doesn't know enough to do the right thing on its own
  - e.g., Lustre distributes files based on capacity utilization, not upcoming I/O
- Application may need to take extra steps to ensure work is distributed over resources
  - e.g., setting striping parameters to engage OSTs
  - Middleware can help – encodes this knowledge



**Gateway nodes** run parallel file system client software and forward I/O operations from HPC clients.

**Commodity network** primarily carries storage traffic.

**Storage nodes** run parallel file system software and manage incoming FS traffic from gateway nodes.

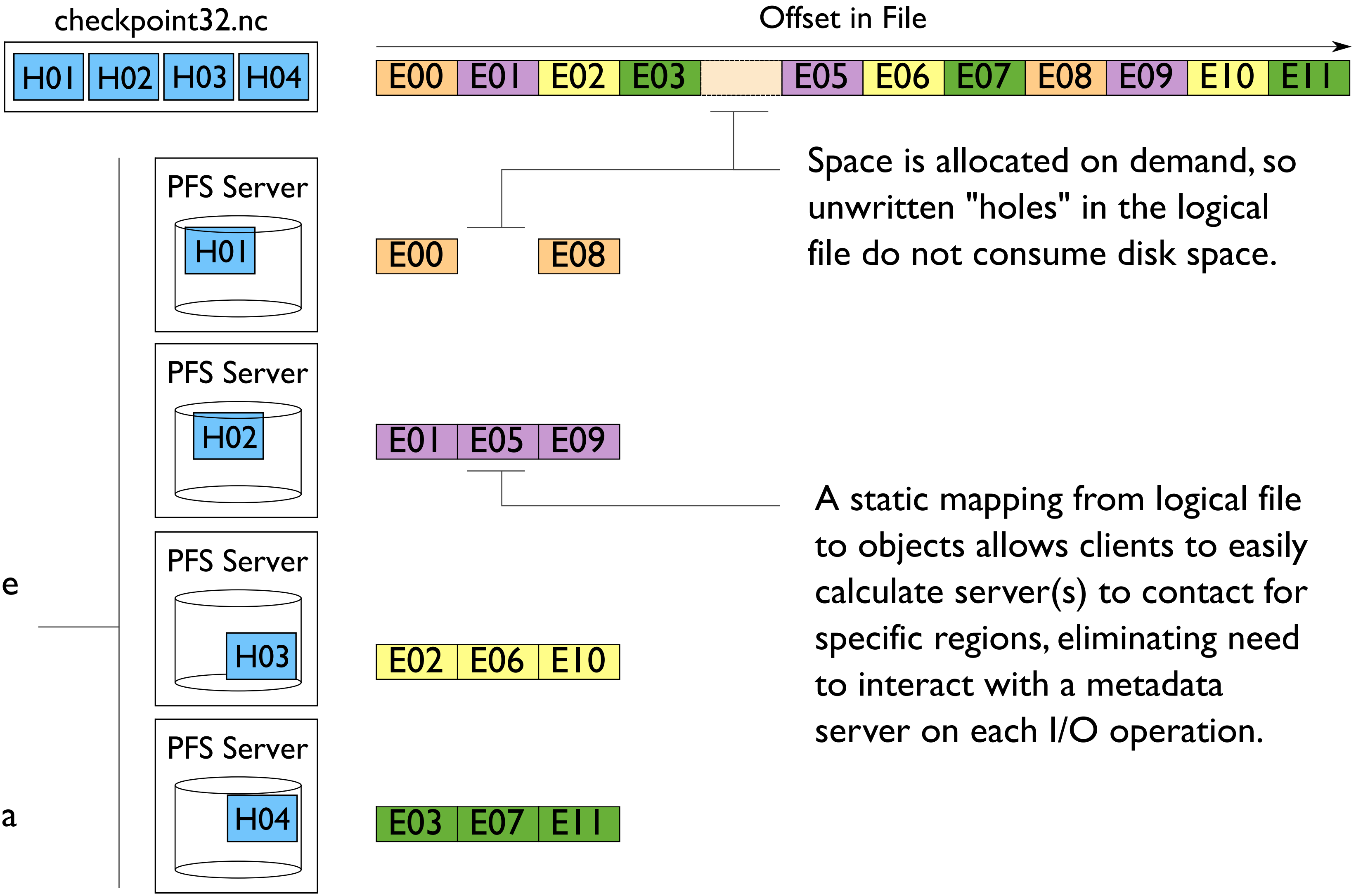
**Enterprise storage** controllers and large racks of disks are connected via InfiniBand.

# Data Distribution in Parallel File Systems

Logically a file is an extendable sequence of bytes that can be referenced by offset into the sequence.

Metadata associated with the file specifies a mapping of this sequence of bytes into a set of objects on PFS servers.

Extents in the byte sequence are mapped into objects on PFS servers. This mapping is usually determined at file creation time and is often a round-robin distribution of a fixed extent size over the allocated objects.

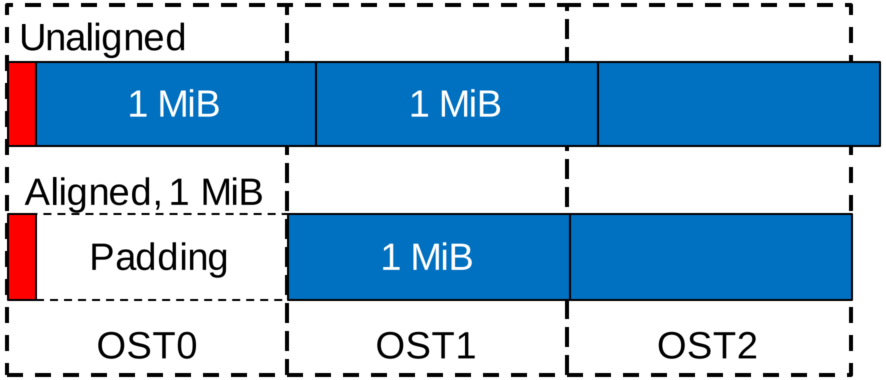


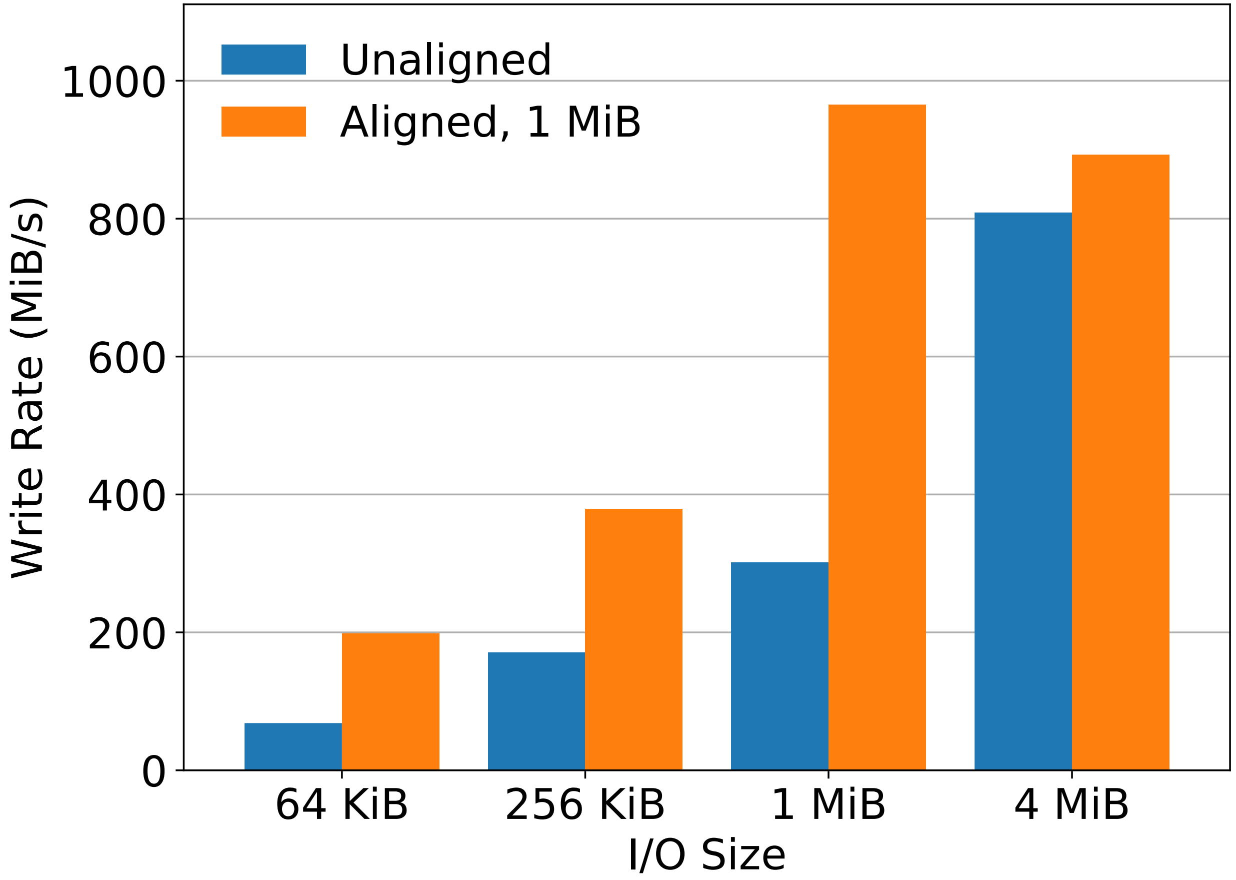
Space is allocated on demand, so unwritten "holes" in the logical file do not consume disk space.

A static mapping from logical file to objects allows clients to easily calculate server(s) to contact for specific regions, eliminating need to interact with a metadata server on each I/O operation.

# How Things Go Wrong (2/4): Inefficient use of HW

- Block devices like, well, blocks…
- Networks have latency
- Can be difficult to differentiate from a software protocol issue (and often it doesn't really matter which it is)

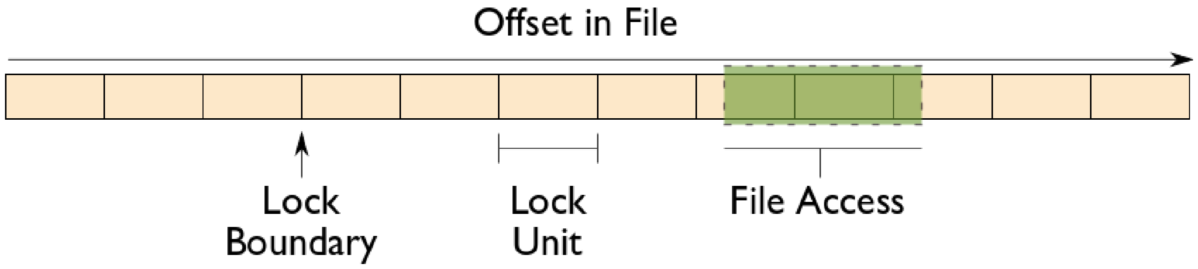




Writing 32 GiB of data with a 26.75 KiB header from 8 nodes (16 ppn) to one file on one Lustre OST

# How Things Go Wrong (3/4): Software Protocols

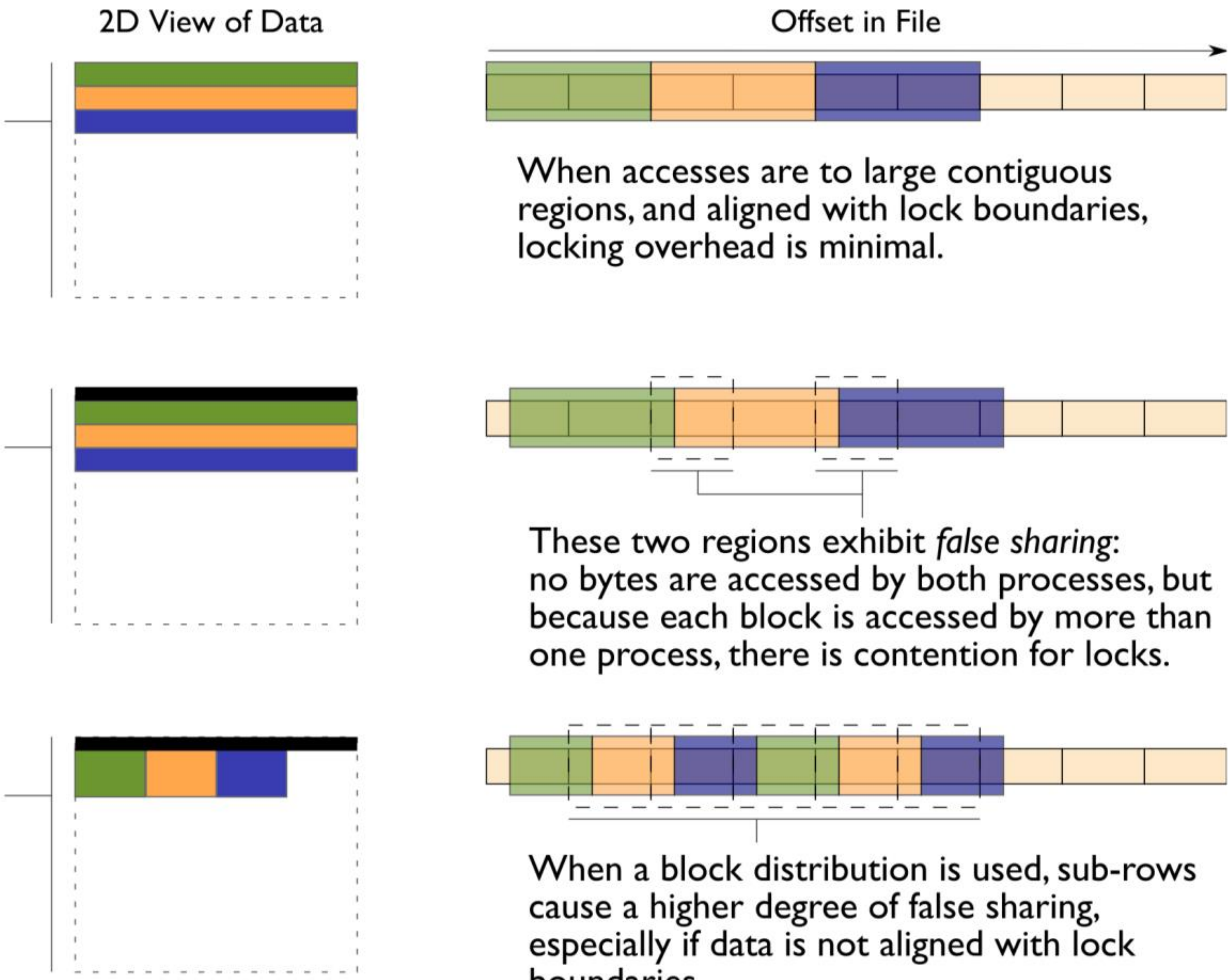
- Protocols are the language used for interaction between services and applications
- Specifics of I/O protocols can have unexpected results
- These can be visible inconsistencies or performance anomalies
    - Unusual behavior of directories in NFSv3
    - Very slow performance in POSIX
- Because protocols vary, it's difficult to work around these issues in a portable way



One example of a protocol that can cause trouble is file system locking. Implemented to enforce POSIX semantics, locks are handed to clients along specific boundaries. File accesses require appropriate read/write locks to proceed.

A typical problem in HPC I/O is that accesses which are unaligned with these lock boundaries exhibit false sharing, significantly degrading performance.

# Locking and Concurrent Access



The left diagram shows a row-block distribution of data for three processes. On the right we see how these accesses map onto locking units in the file.

When accesses are to large contiguous regions, and aligned with lock boundaries, locking overhead is minimal.

In this example a header (black) has been prepended to the data. If the header is not aligned with lock boundaries, false sharing will occur.

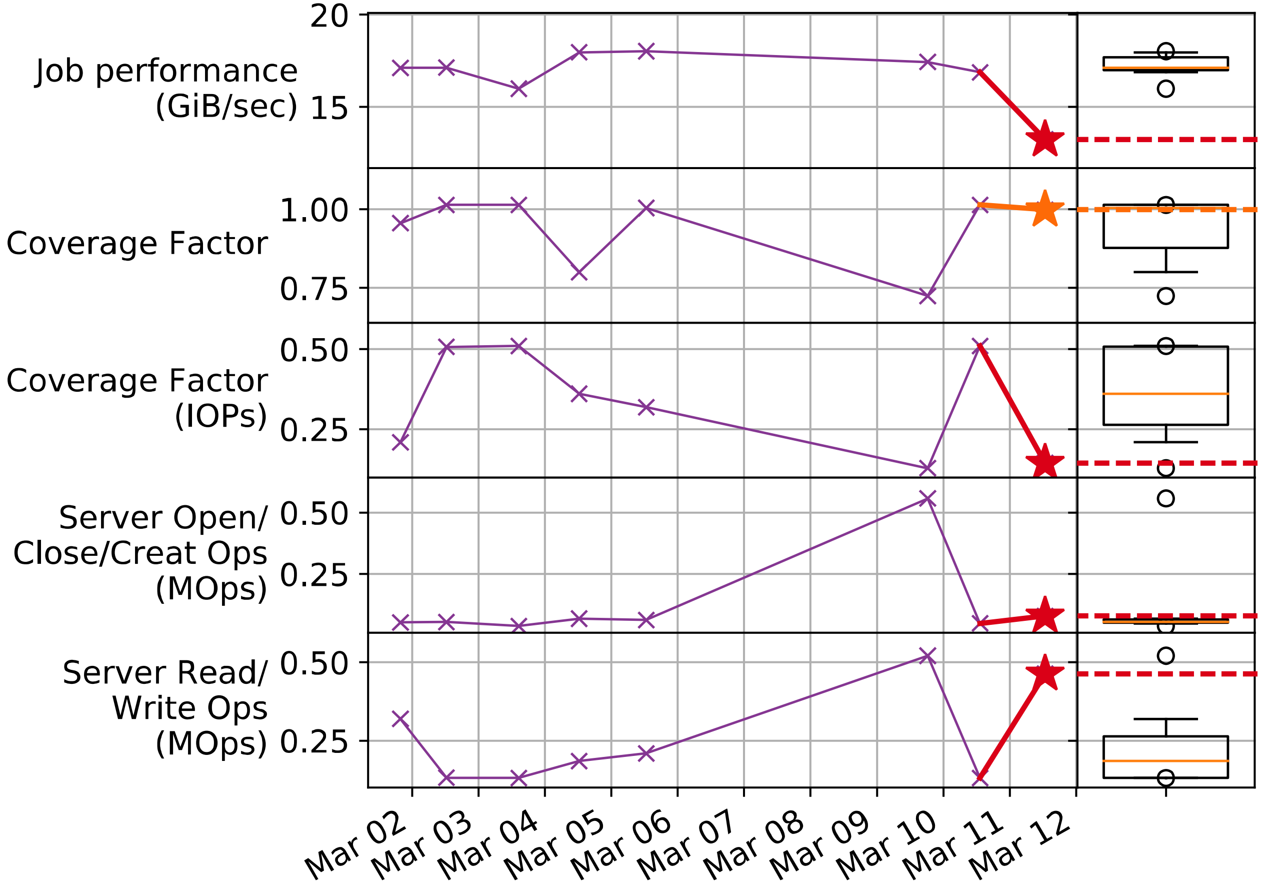
These two regions exhibit *false sharing*: no bytes are accessed by both processes, but because each block is accessed by more than one process, there is contention for locks.

In this example, processes exhibit a block-block access pattern (e.g. accessing a subarray). This results in many interleaved accesses in the file.

When a block distribution is used, sub-rows cause a higher degree of false sharing, especially if data is not aligned with lock boundaries.

# How Things Go Wrong (4/4): Interference from Others

- Storage systems are usually a shared resource
- Network is also a shared resource
- Traffic from other jobs can impact performance of I/O for your job
- Quality of Service (QoS) doesn't really exist for HPC storage at this time
  - Yes that's a good idea!
- Main thing one can do here is to try to not be a source of disruptive traffic



Performance of VPIC-IO runs on Mira platform over 10 days, significant I/O degradation (top graph) resulted from background readdir() traffic generated elsewhere.

G. Lockwood et al. UMAMI: a recipe for generating meaningful metrics through holistic I/O performance analysis. PDSW-DISCS. November 2017.