Asynchronous Object Storage with QoS for Scientific and Commercial Data

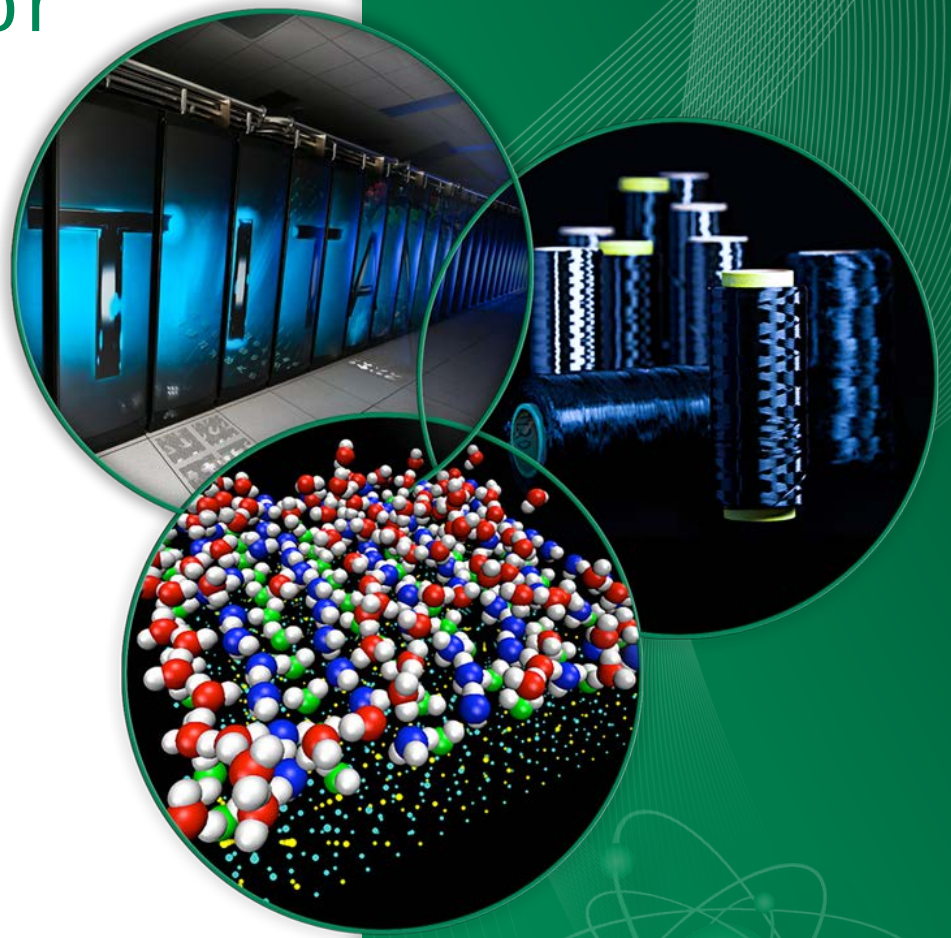
Michael J. Brim

David A. Dillow

Sarp Oral

Bradley W. Settlemyer*

Fieyi Wang



PDSW
Nov. 18, 2013

Introduction

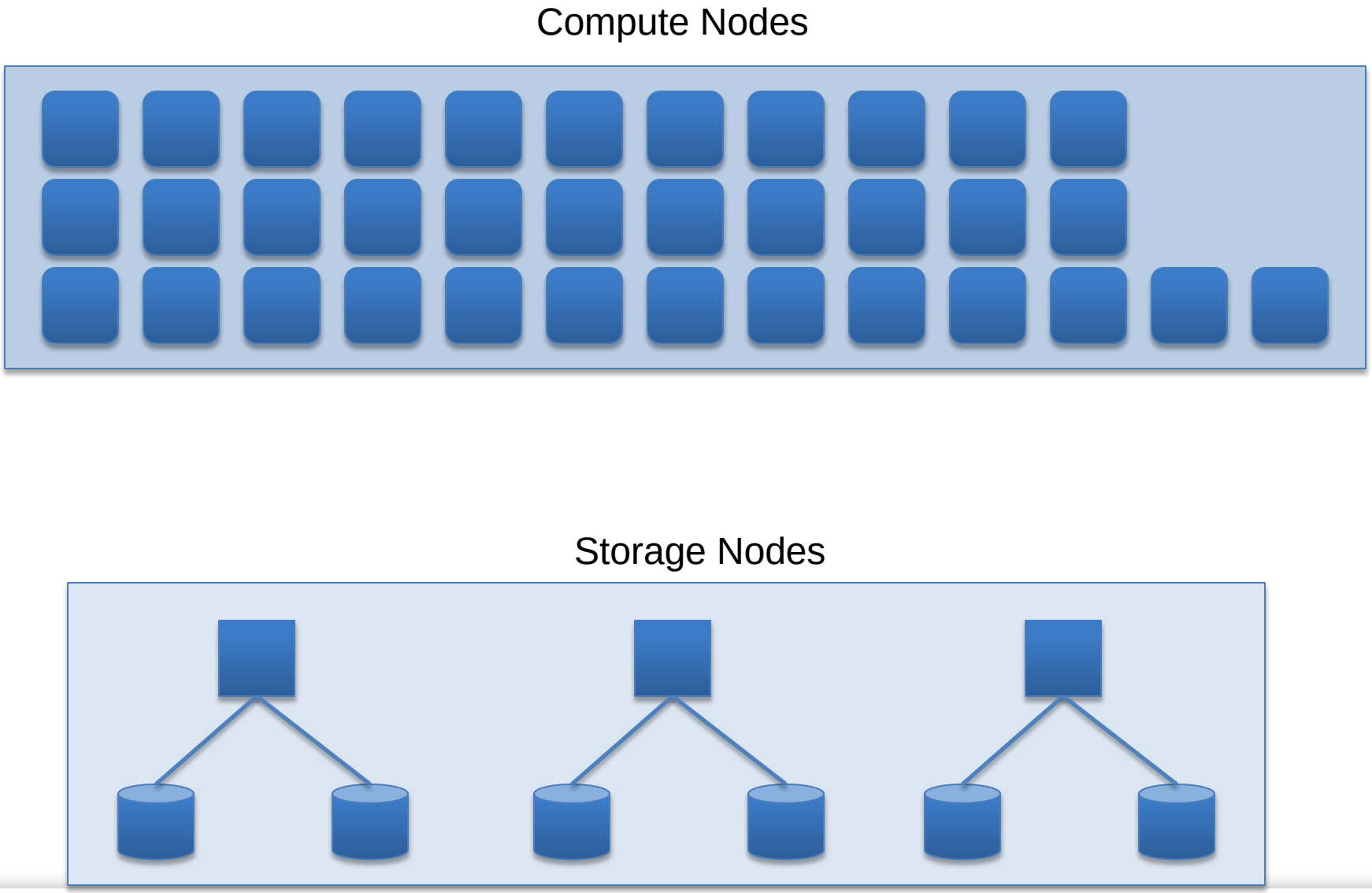
> "Scientific discovery in energy research and a wide range of other fields increasingly depends on effectively managing and searching large datasets for new insights."
>
> - Dr. Steven Chu, Secretary of Energy
> Big Data Research Initiative, March 29, 2012

- What is Big Data
  - Volume, variety, and velocity
  - Good support for statistical inference/induction
    - As opposed to traditional descriptive statistics
  - Exposes some weaknesses in existing HPC storage systems

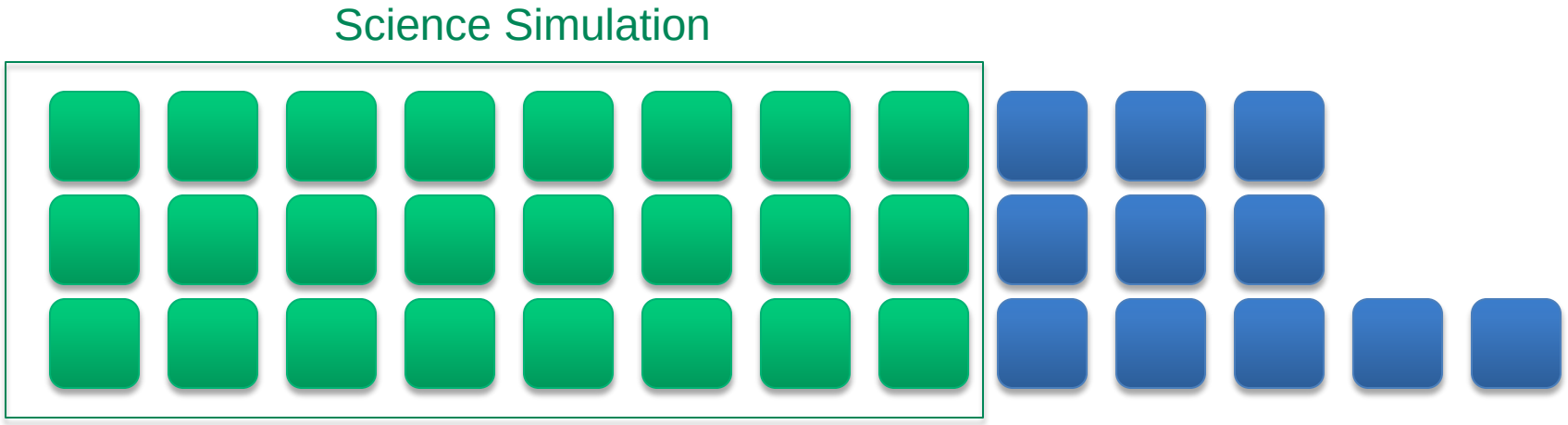
Big Data HPC Use Cases

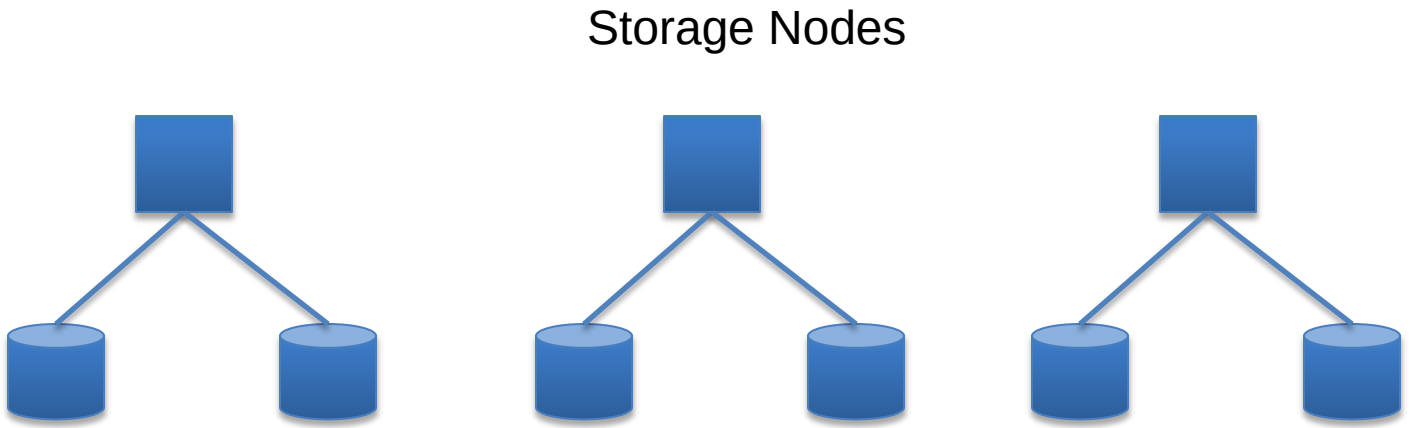
1. Scientific Application Checkpointing
   - Codes often time-step based simulation
   - Bursty I/O (write for 5 minutes, once an hour)
   - Almost entirely storage system write limited
   - Large fraction of the memory space of the entire application streaming to storage
2. Big Data Analysis
   - Data mining, parallel data analysis, statistical induction
   - Ideally, Map-Reduce
   - Large block storage system reads
   - Almost continuous storage system load

Interference Scenario

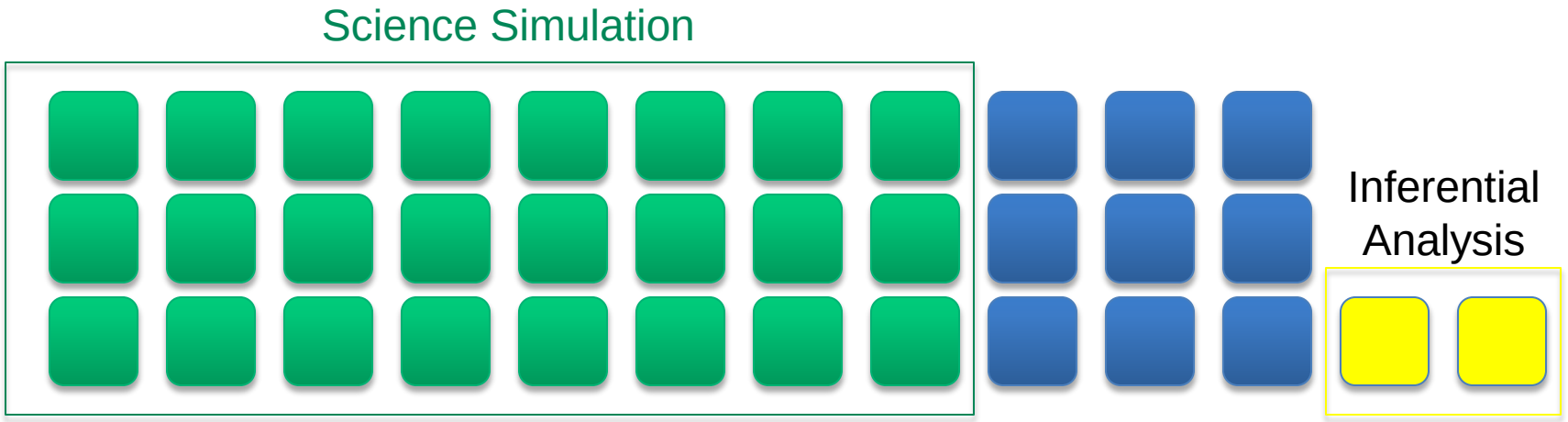


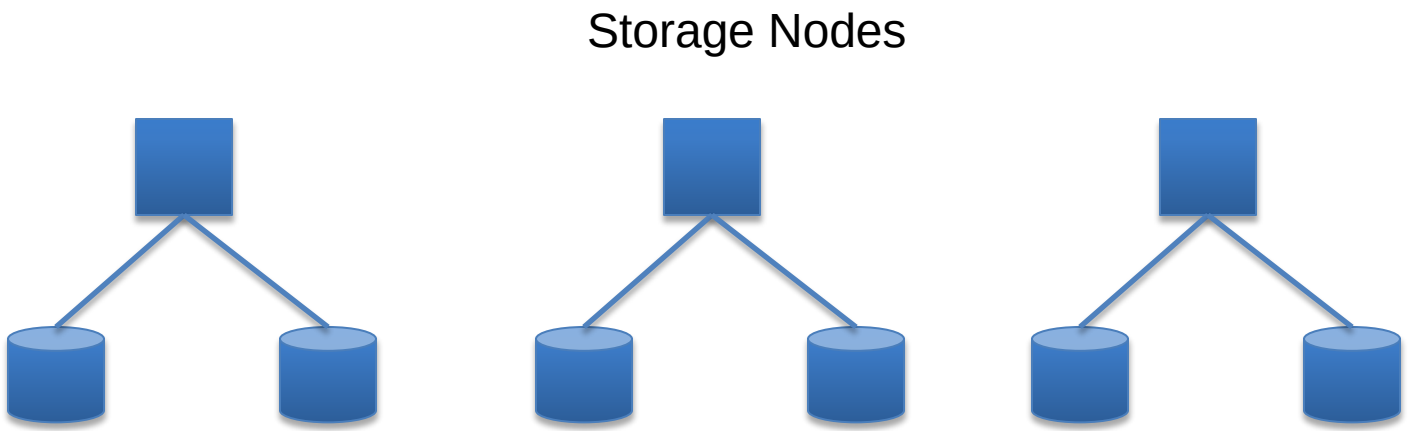
Interference Scenario



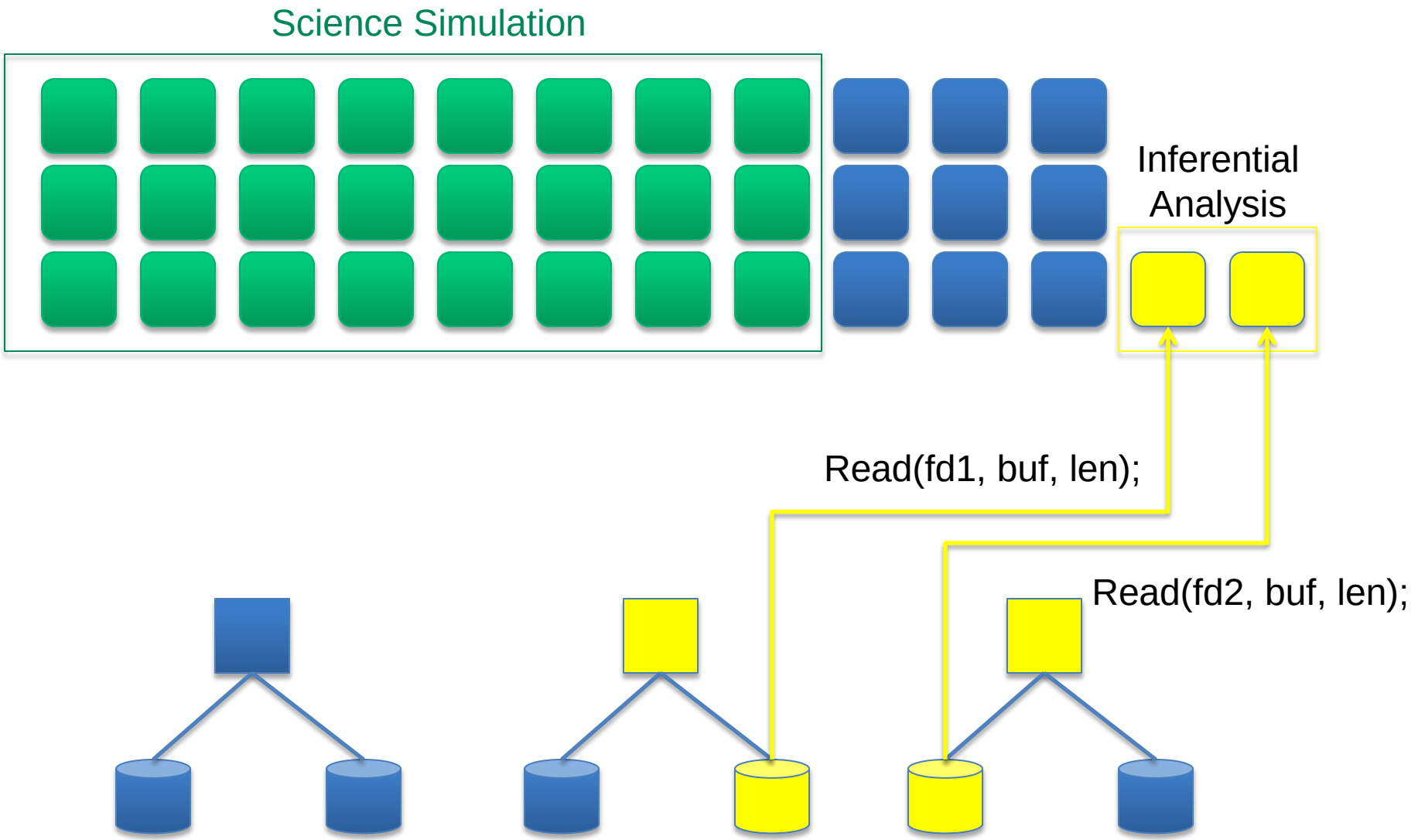


Interference Scenario

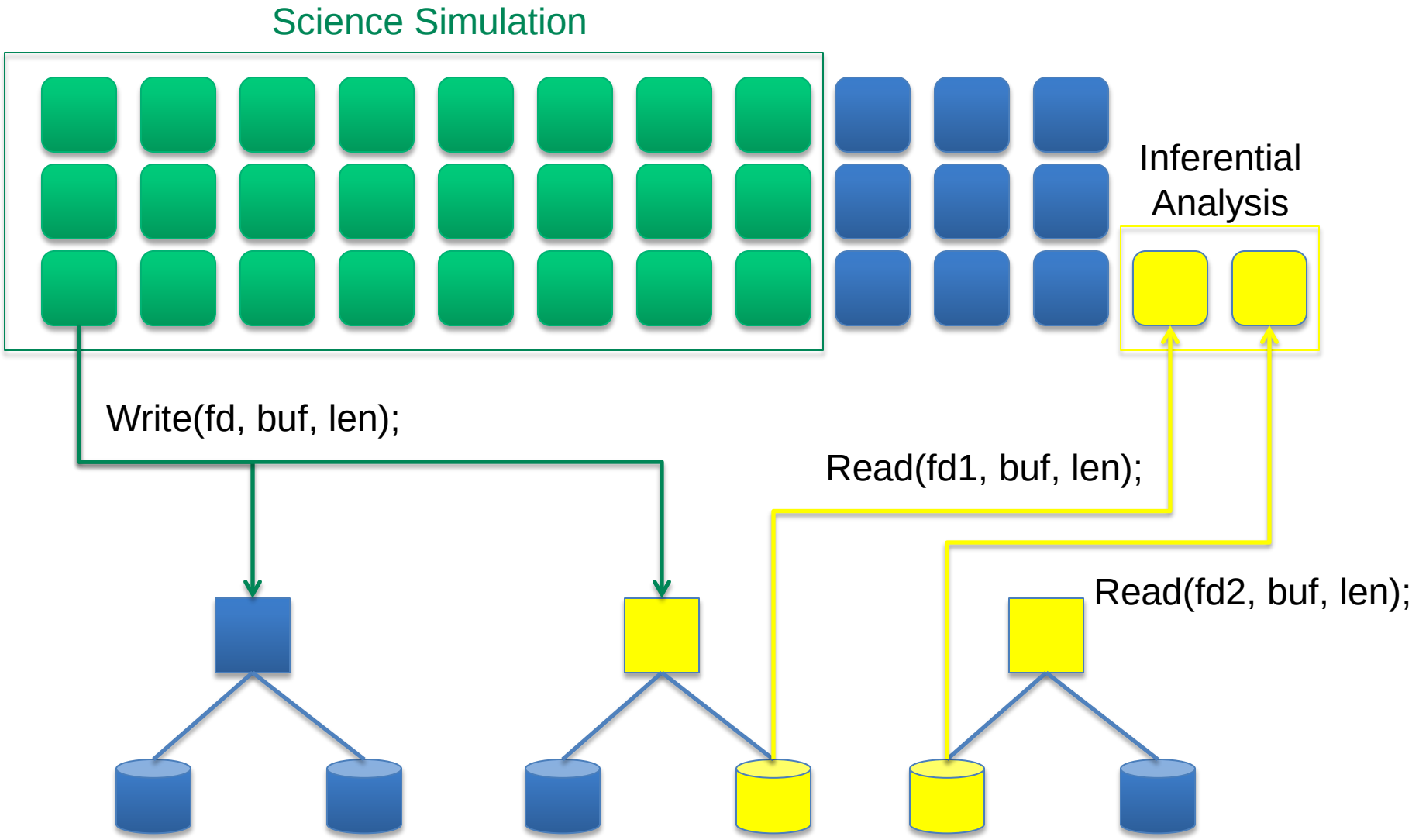




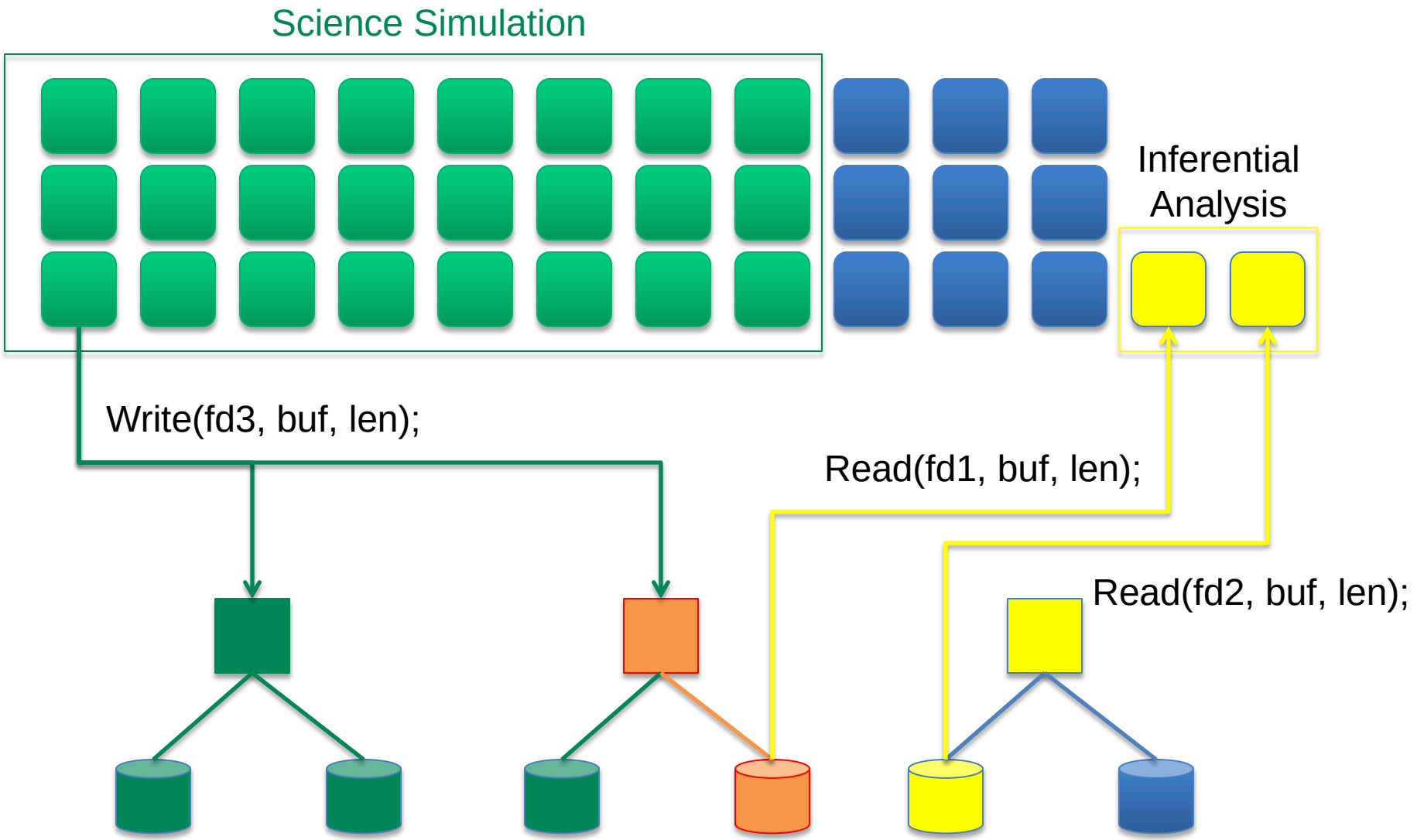
Interference Scenario



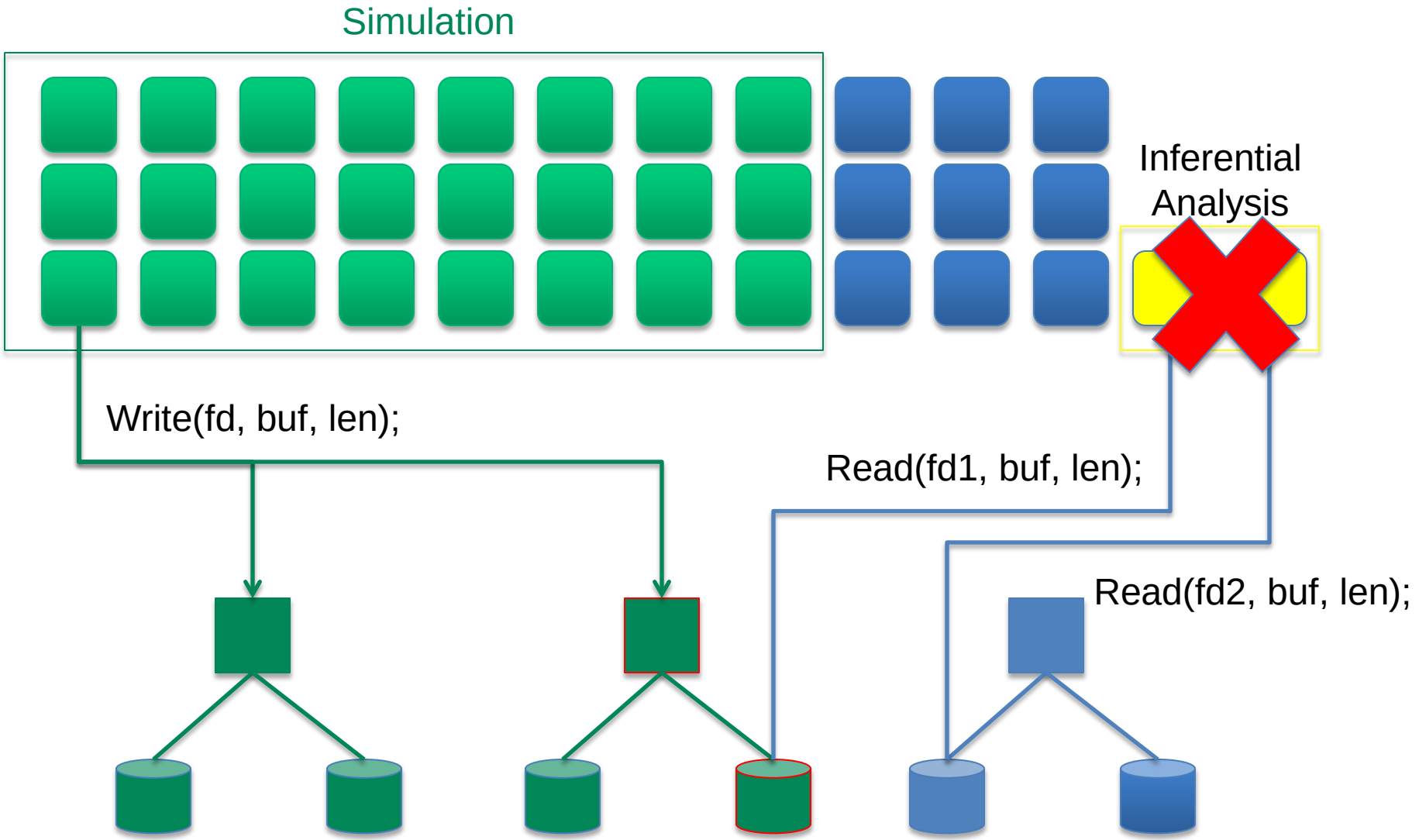
Interference Scenario



Interference Scenario



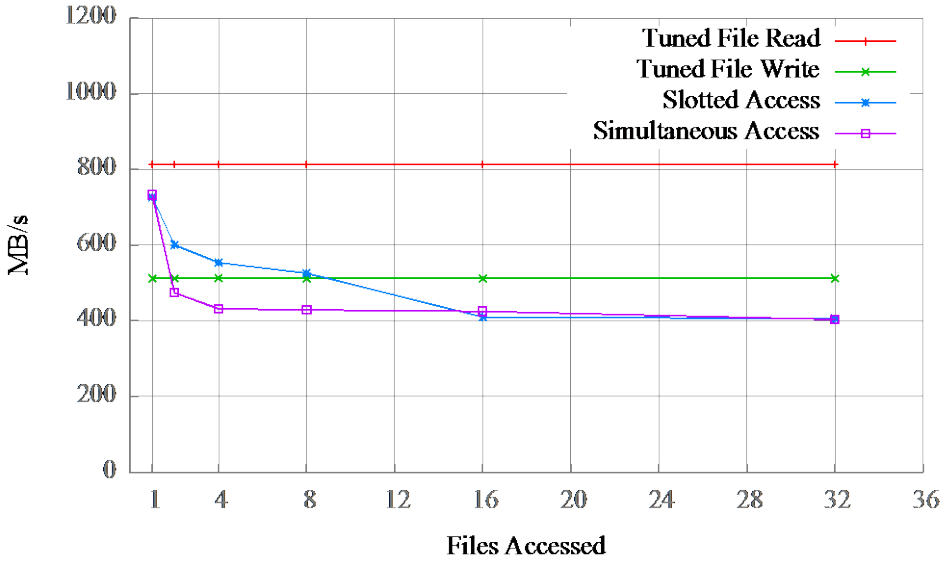
Interference Scenario

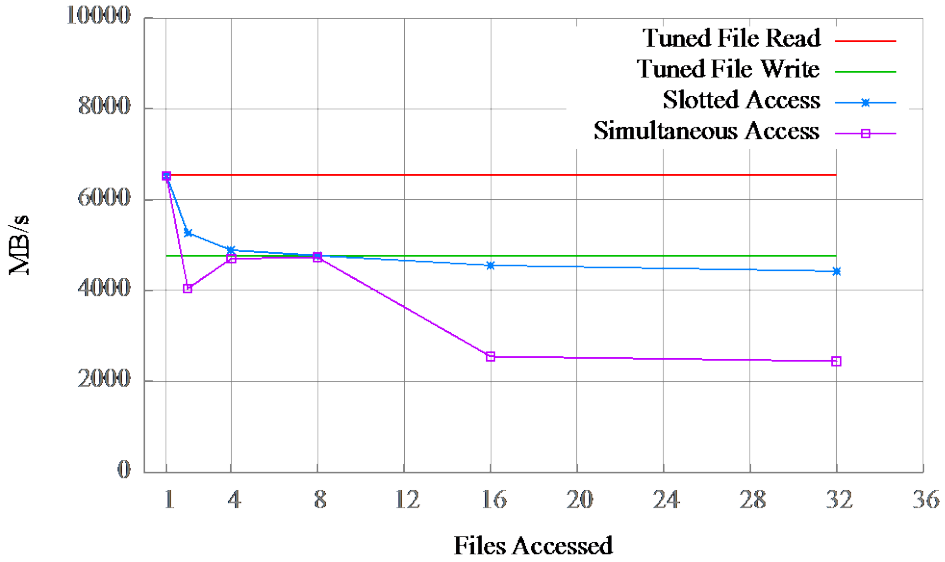


I/O Interference Reality

- Why can't scientific applications just overlap computation with I/O?
  - A time-step based simulation doesn't checkpoint after every time-step, just after some time-steps
  - Next program state depends on previous program state
  - Significant memory pressure
  - Coordination across multiple compute nodes makes these problems worse rather than better
- Doesn't CFQ scheduling make this a non-issue?
  - The storage servers use the same PID
- Alternative solutions like ADIOS circumvent FS services

Read/Write I/O interference



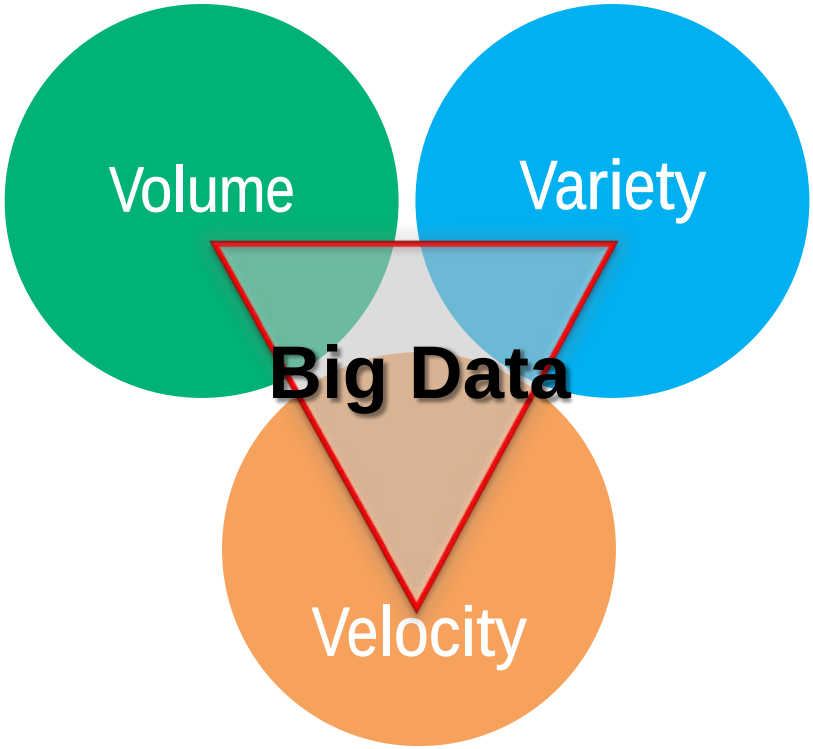


- Low-end storage array
- Simultaneous access falls off immediately

- High-end solid state storage (FusionIO Octal)
- Performance loss due to large queue depths

A Scalable Object Store

- HPC Storage + Analytics
- SOS Goals:
  1. 100% Asynchronous access
  2. Provide storage quality of service (QoS)
  3. Object resilience with multi-tiered storage
  4. In transit object data transformation

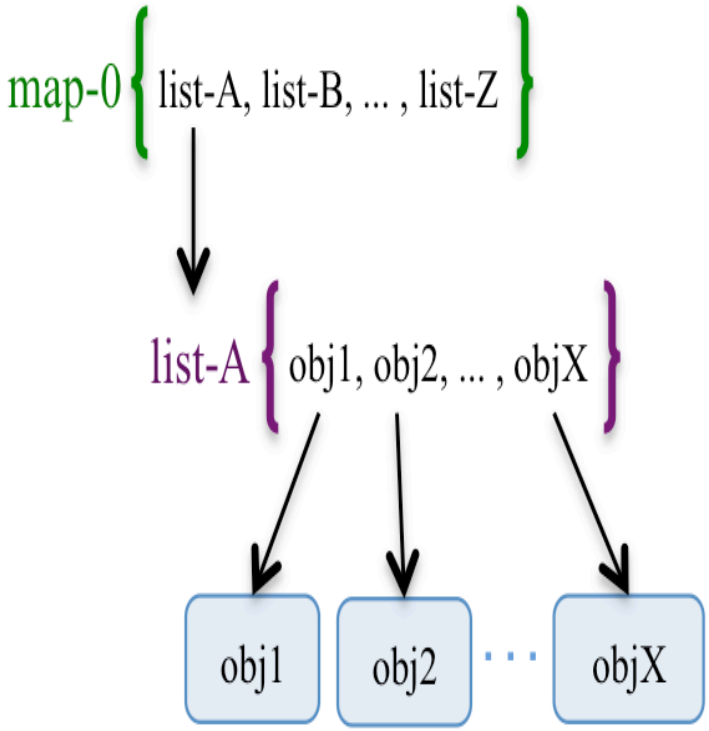


Quality of Service

- Two factors make QoS tractable:
  - SOS I/O is fundamentally async and server-directed
  - Object semantic takes away consistency between two clients writing to same store
- Lot's of existing work on how to do this for a single server with multiple clients
  - No magic, we just take a slotted approach to multiple access (like ALOHA)
  - Don't try to make it perfect, if an IOP runs past the end of the slot, complete it anyway
- Clients request QoS via reservations
  - Similar to network RSVP protocols

SOS Prototype Data Organization Model

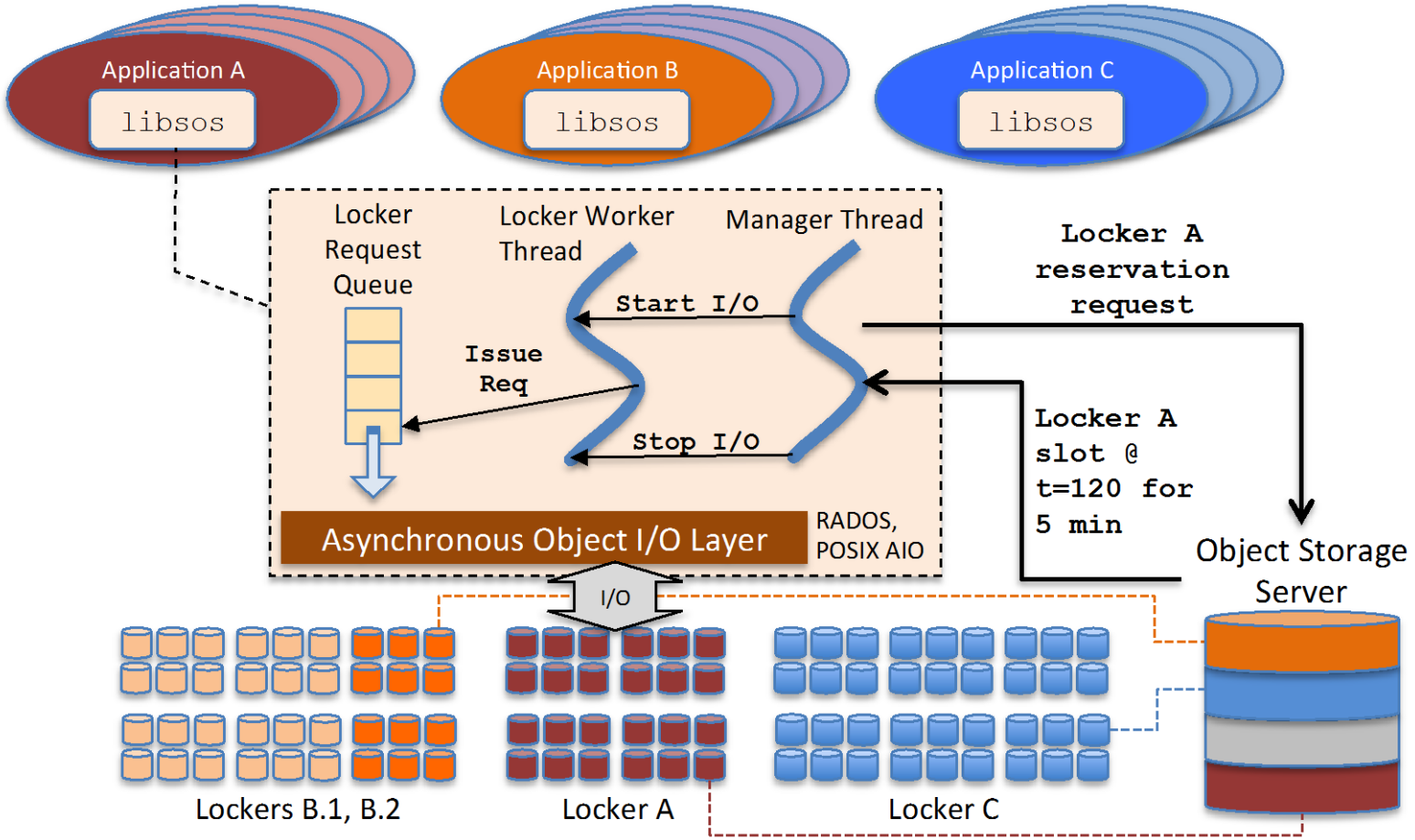
- Objects
  - Named data buffers
- Object Collections
  - Lists – named collection of objects
  - Maps – named collections of lists
- Provides naming hierarchy, that we believe is compatible with use cases



SOS Prototype Storage Organization Model

- Object Lockers
  - A dynamically provisioned allocation of storage resources
  - Provides opaque private namespace
  - Returned via a reservation request
  - Lockers provide the QoS scheme
- Prototype implementation is based on Ceph
  - Lockers provided by RADOS pools
  - Asynchronous Object Placement with CRUSH is weird
  - Leverage the lessons of POSIX AIO and Linux AIO

SOS Prototype



Ongoing work

- Server-directed object placement
  - Ceph uses CRUSH which is client directed placement
  - Exploring organizing locker's out of Chord Rings
- Scalable reservation protocols
  - Currently dedicating a server interval slot to handling reservations
  - Simply dropping reservation requests when a reservation for a new locker cannot be satisfied
  - Reservations acquired in order for existing lockers (to avoid dining philosophers problem)
  - Fault tolerant runtimes are still rare

Future Use Cases

- Scalable Fault Tolerance Backplane (for runtimes)
  - Storage systems are and should continue to be far more reliable than large compute resources
- Visualization and performance traces
  - Tricky because these are user-guided, and performance is dominated by small, unaligned data access
- Key-value support
  - Important to whole classes of Cloud applications
  - E.g. HTTP Session Data

Acknowledgements

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

This work also used resources at the Extreme Scale Systems Center, located at Oak Ridge National Laboratory and supported by DoD.

Thanks!

Questions?

Asynchronous I/O Interfaces

- HPC networks moving steadily toward asynchrony simply to support scale
- Many existing asynchronous I/O API's problematic
- The only exception I am aware of is Linux AIO
  - Has proven very useful in scenarios such as FS scan
  - Effective because it improves disk access performance!
- SOS attempts to build on lessons of Linux AIO
  - HPC networks likely natively async
  - Improve disk performance via server-directed I/O
  - Improve client predictability QoS