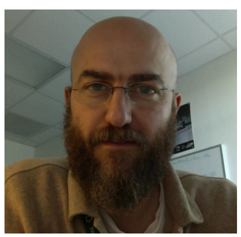
LADS: Optimizing Data Transfers using Layout-Aware Data Scheduling

Youngjae Kim

Scott Atchley

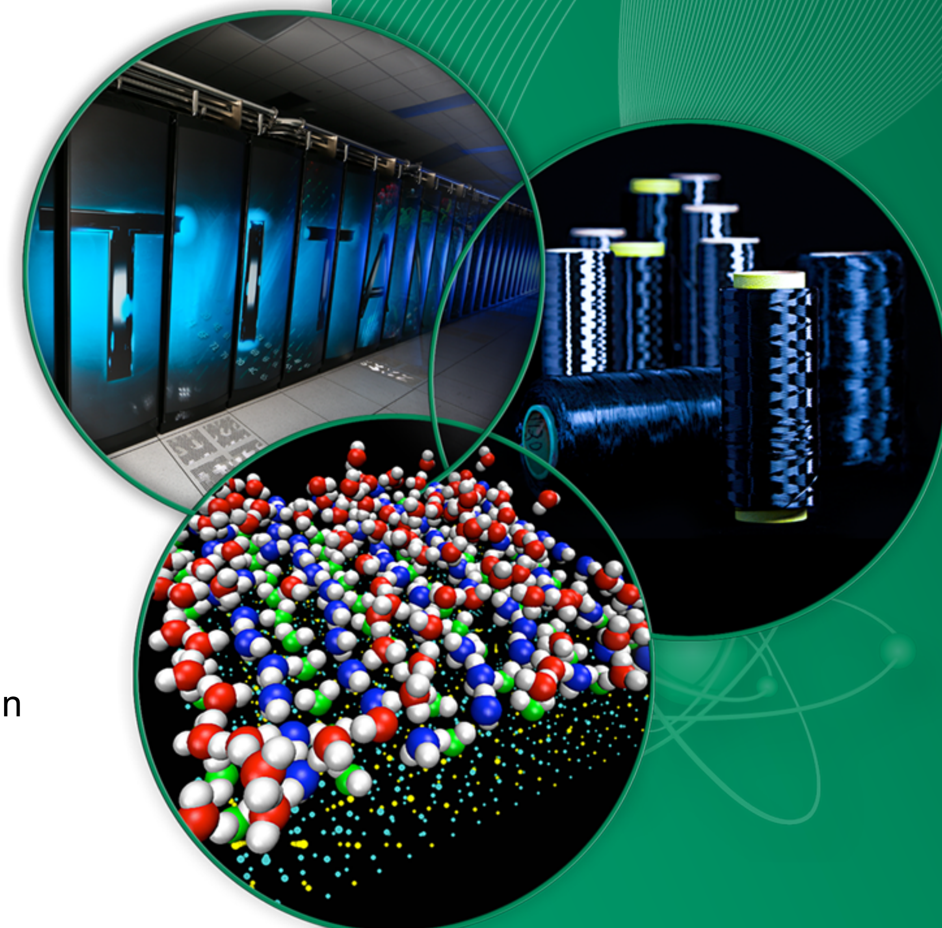


Geoffroy Vallee

Galen M. Shipman

Oak Ridge National Laboratory

* Galen is currently with Los Alamos National Lab.

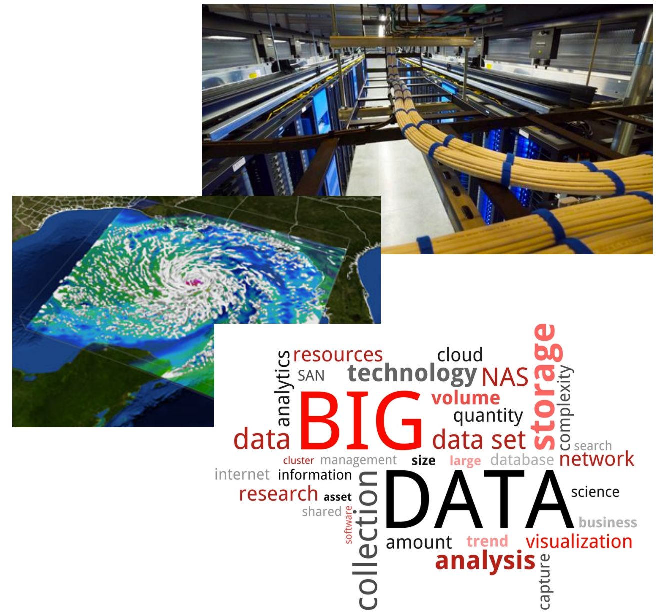


Outline

- Background
- Motivation
- LADS: Layout-Aware Data Scheduler
- Problem Definition
- Design
- Evaluation
- Conclusion

Leading Research Requires the Use of Extreme-scale Resources across DOE

- Big Data Challenges in Science Domains
- Extreme-scale Resources
  - Computational facilities – ALCF, NERSC, and OLCF
    - 1 exabyte generated per year by 2018
- Coupling data
  - Is to combine two different data sets physically stored on different institutes to use for big data analysis purpose
- Many examples of coupling data today:
  - Nuclear interaction datasets generated at NERSC needed at the OLCF for Petascale simulation
  - Climate simulations run at ALCF and OLCF validated with BER data sets at ORNL data centers



Unfortunately data-sets do not exist in isolation!

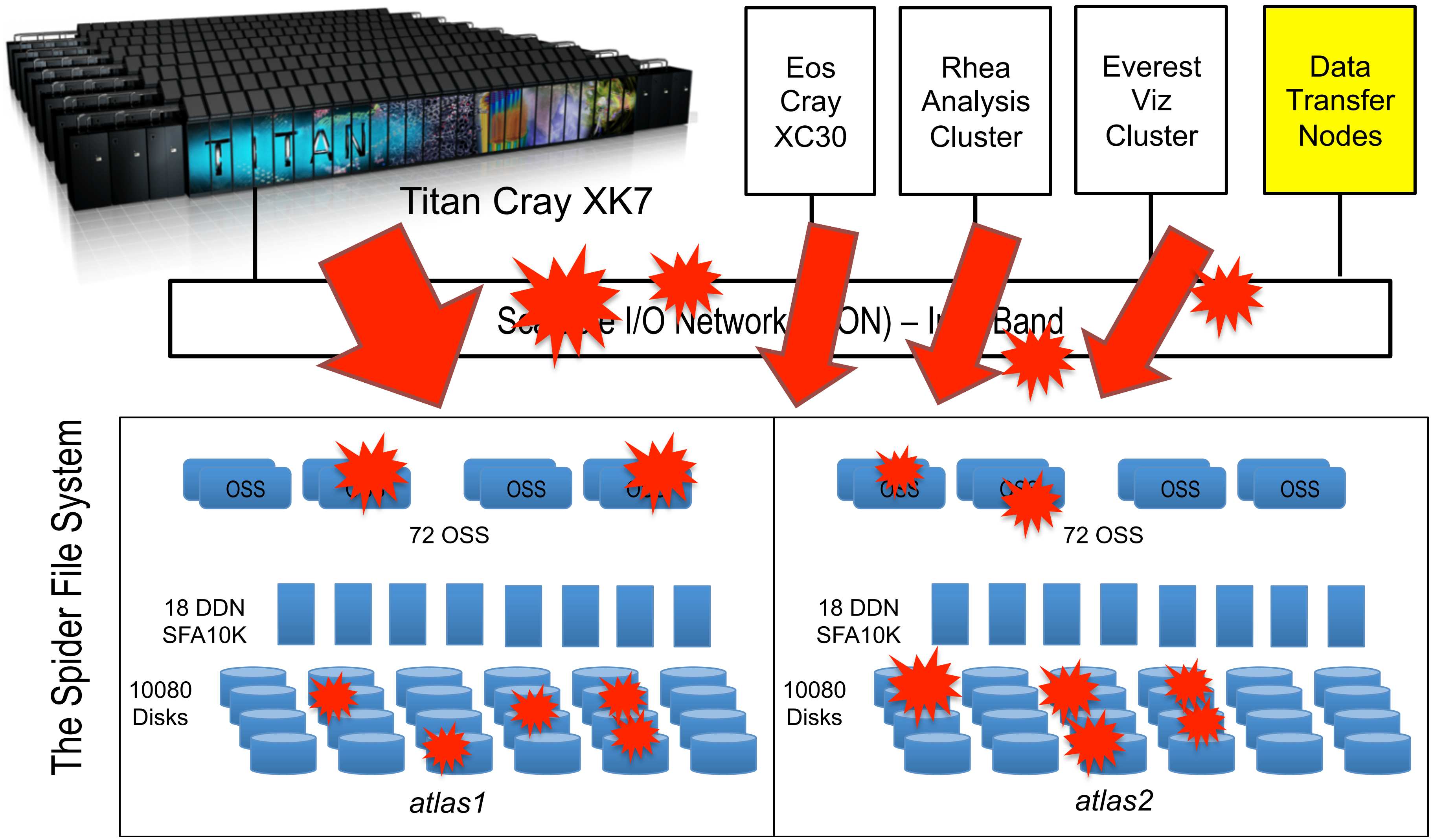
Enabling Network Technology

- DOE's Energy Science Network (Esnet)
  - Network infrastructure between many DOE facilities
- Improved data transfer rate
  - Currently at 100Gb/s, mostly likely to support 400Gb/s
  - 1Tb/s in near future

However, this network improvement only contributes the network transfer rate.

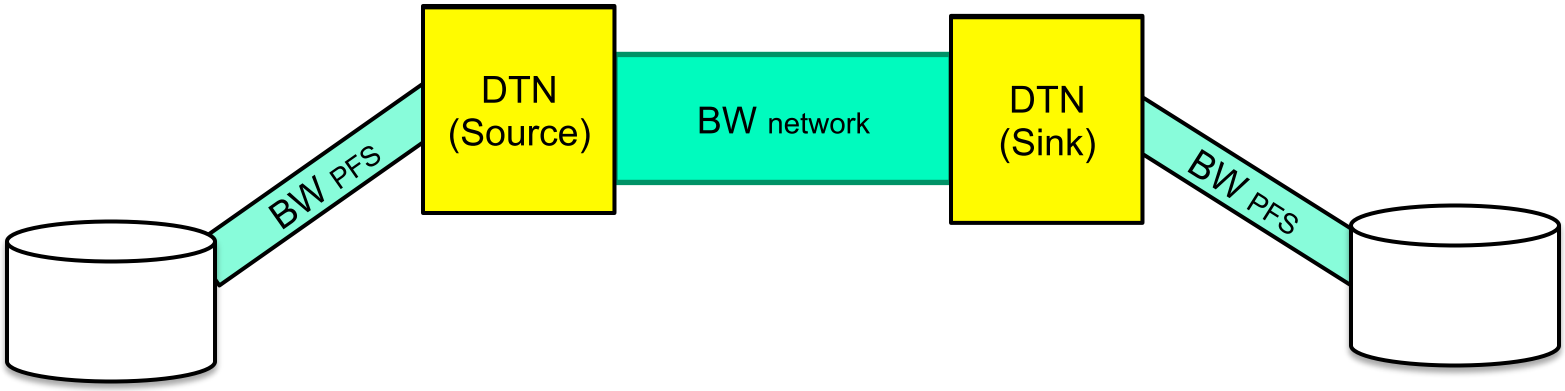
Data sets are stored at slow storage systems.

OLCF center-wide PFS and clients



The PFS can become the bottleneck for data transfers.

Problem and Challenge



$BW_{PFS} < BW_{network}$

Data transfer nodes (DTNs) are a focal point for impedance match between the **faster networks** and the relatively **slower storage systems** (PFS).

Key Question

How to improve the underutilized PFS bandwidth for big data transfers?

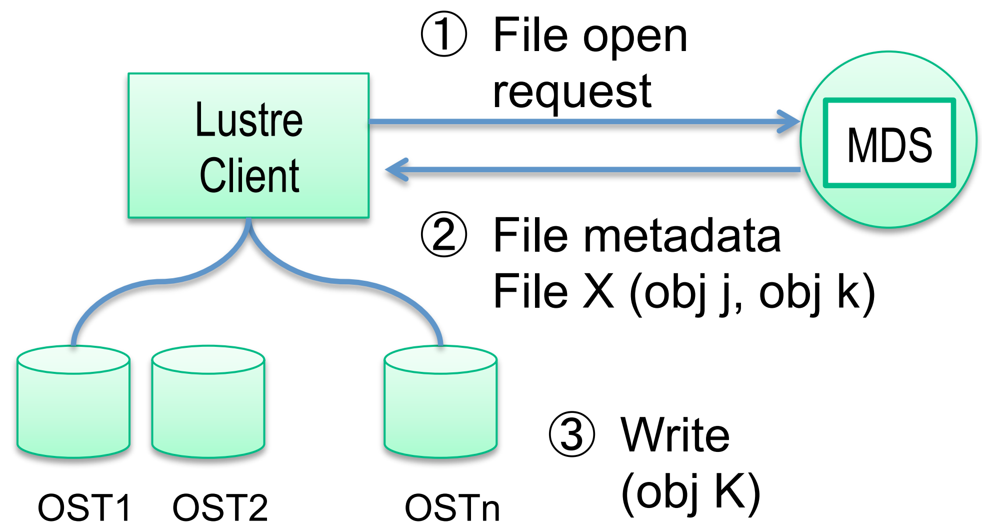
Outline

- Background
- Motivation
- LADS: Layout-Aware Data Scheduler
- Problem Definition
- Design
- Evaluation
- Conclusion

General Parallel File Systems

- Parallel File Systems
  - Lustre, IBM's GPFS, PVFS, PanFS
- Lustre
  - Two types of servers
    - MDS and OSS
  - Metadata server (MDS)
    - Holds the directory tree
    - Stores metadata about each file (except for size)
    - Once file is opened, I/O to file does not involve the MDS
  - Object storage server (OSS)
    - Manages OSTs (disk/LUN)
    - OSTs hold stripes of the file contents
      - Think RAID0
    - Maintains the locking for the file contents it is responsible for

File open and file I/O in Lustre



Observation form the PFS

PFS is viewed as a single name space.

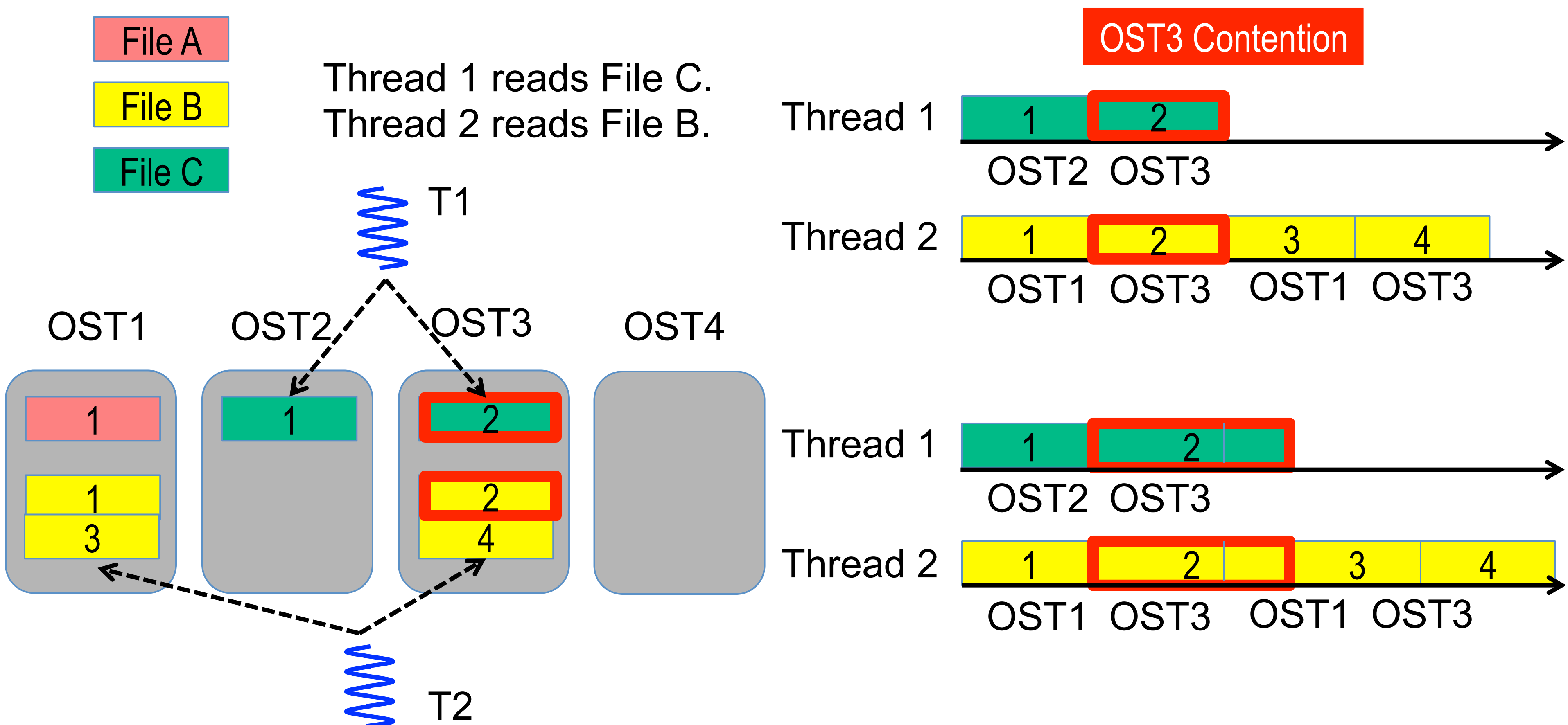
However it is not a single disk, but the arrays of multiple disks with servers.

PFS has been designed for parallel I/Os.

However, traditional data transfer tools do not fully utilize the available parallelism on the PFS for I/Os.

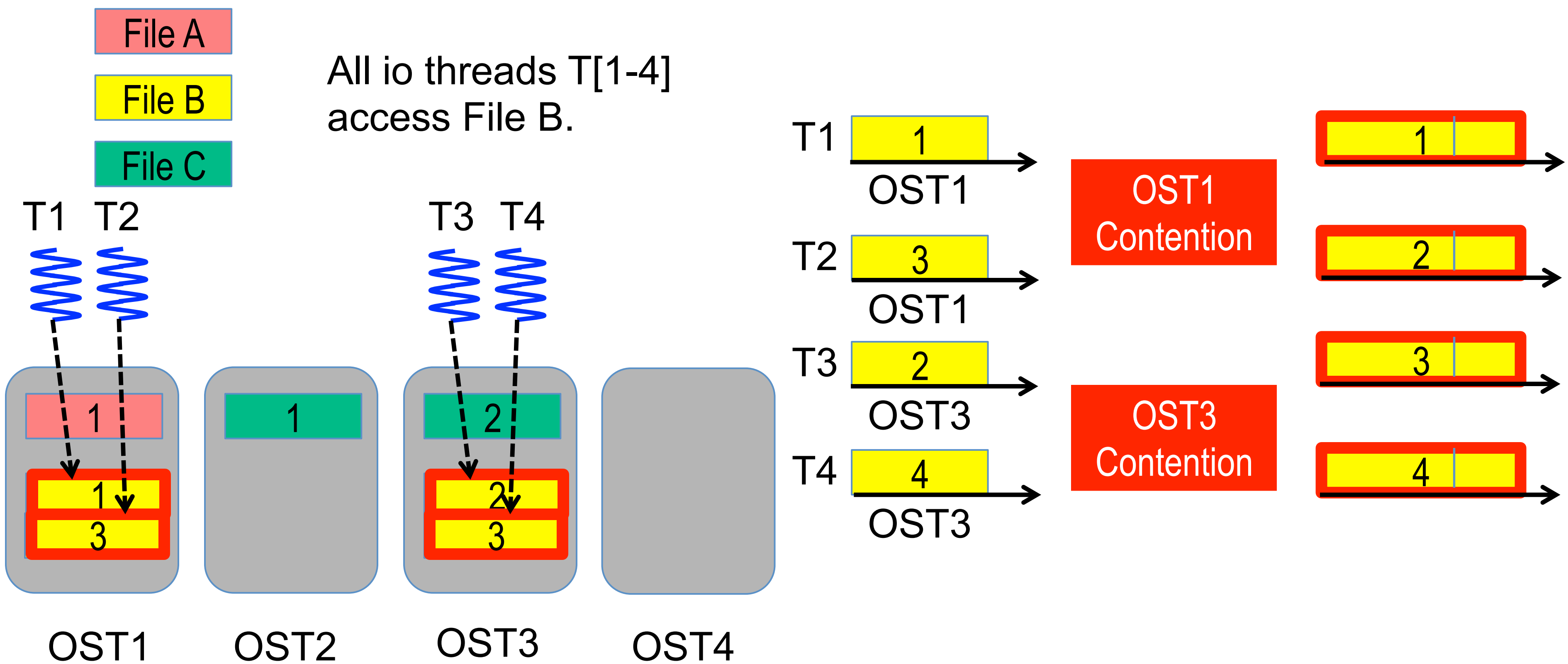
Problem(1): Traditional File Based Approach

- Ignoring file layout information
- A complete file can be assigned to each thread, and each thread works on the file until the file is read.
  - Multiple threads can interfere each other on accessing the same OST.



Problem(2): Traditional File Based Approach

- Ignoring file layout information
- Multiple threads can work on a single file.
  - The parallelism can be limited by a stripe width of a file in the PFS.



Bulk Data Movement Software

- GridFTP
  - Requires Globus Toolkit
  - Supports multiple I/O threads, but it implements a file based approach
- bbcp
  - The most popular data transfer tool for convenience reason, not for performance
  - Implements a file based approach
- RFTP [SC'11]
  - It is a file based approach, not fully utilizing underlying object layouts on the PFS
- SCP
  - A single thread secure copy tool

None of these tools are optimized for fully utilizing the parallelism on PFS for big data transfers.

Traditional file transfers tools employ a logical view of files.

On the other hand,

> LADS uses the **physical view of files**, instead of a **logical view of files**.

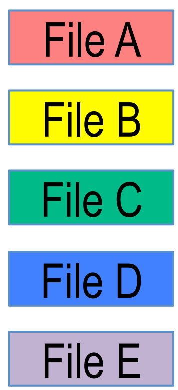
LADS can understand

- The physical layout of files in which files are composed of data objects
- The set of storage targets that hold the objects
- The topology of storage servers and targets

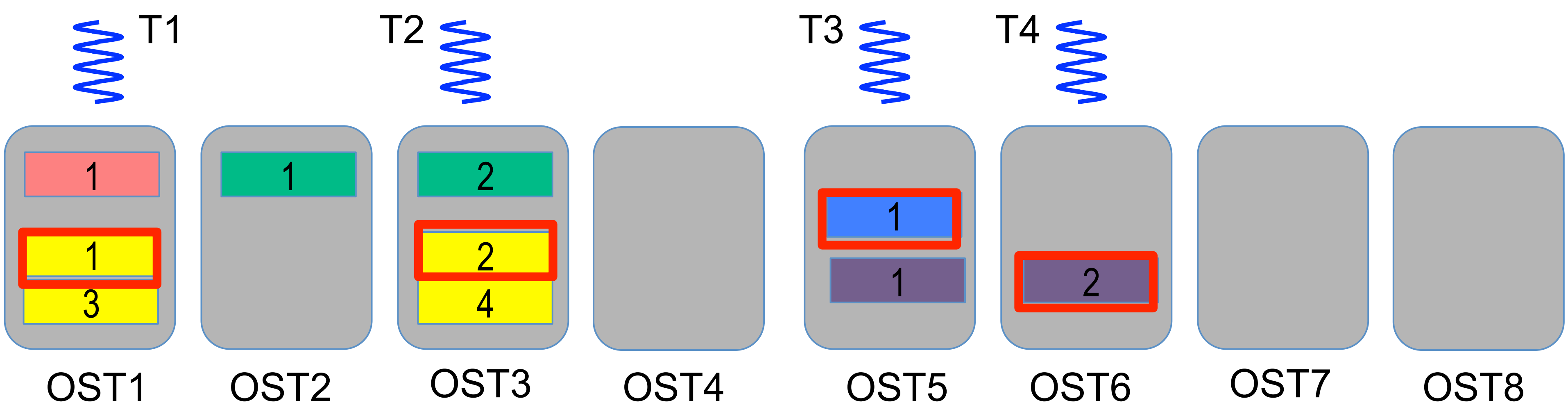
Solution: Object Based Approach

- Aware of file layout information
- A thread can work on any slice (object) on any OST.

None of I/O threads contend for OSTs.
Parallelism = 4 (# of threads)



Thread 1 reads obj 1 of File B.
Thread 2 reads obj 2 of File B.
Thread 3 reads obj 1 of File D.
Thread 4 reads obj 2 of File E.



LADS: Design Goals

Parallelism on Multi-core CPUs

Portability for Modern Network Technologies

Parallelism on PFS

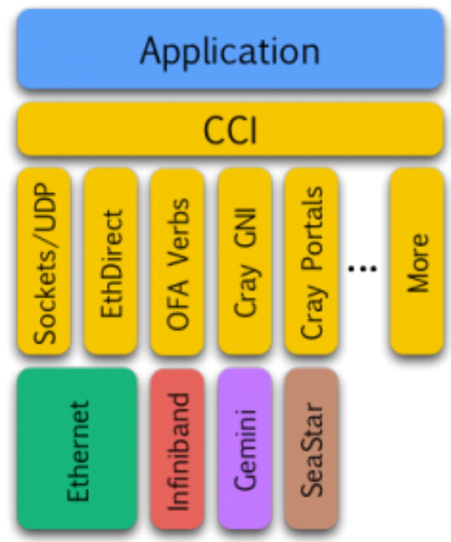
HotSpot/Congestion Avoidance

End-to-End Data Transfer Optimization

Solving the **impedance mismatch problem** between the **faster network** and **slower storage system**

Common Communication Interface

- A new generic, communication abstraction layer
  - A network **Application Programming Interface (API)** for inter-process communication



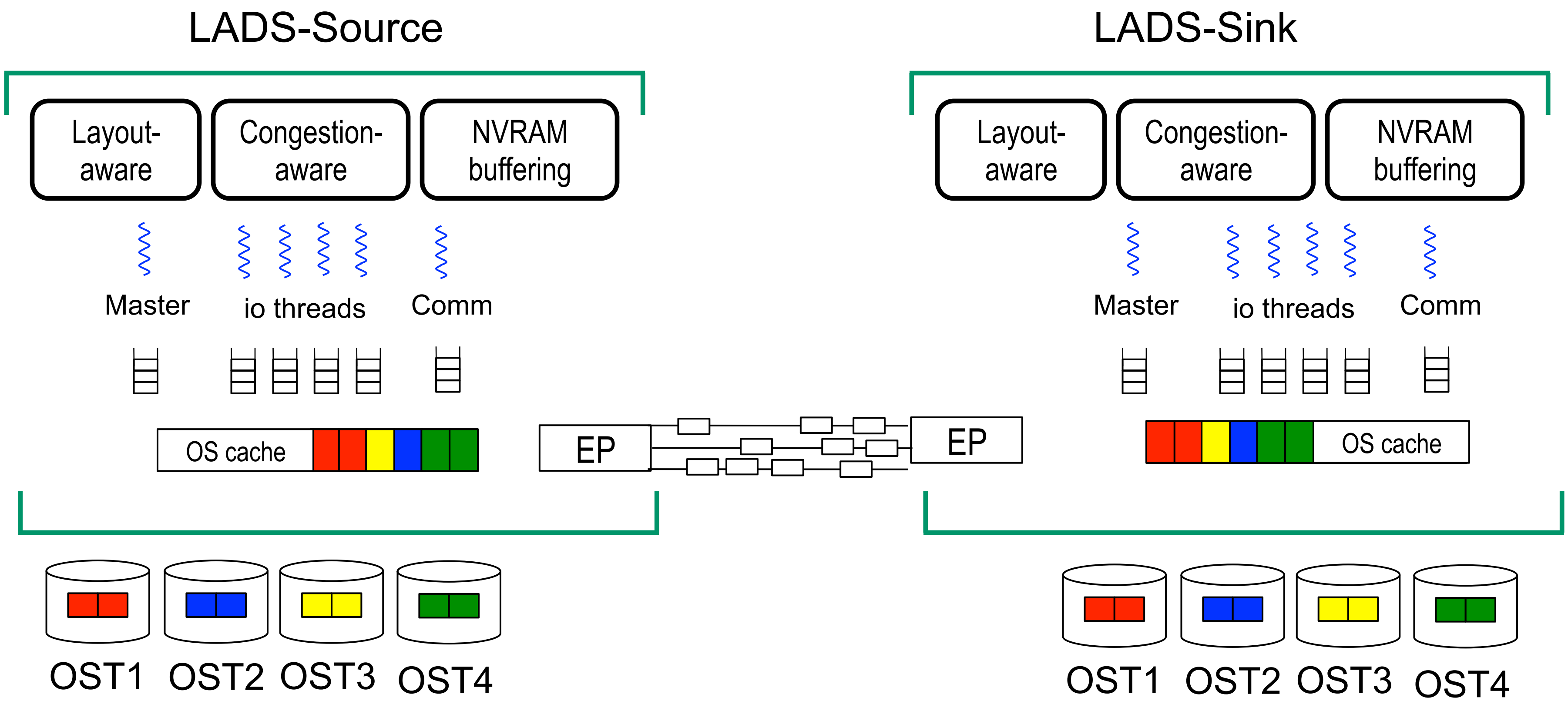
- Design goals
  - Portability, Scalability, Performance, Robustness, Simplicity

- Network solutions that CCI is currently supporting
  - Sockets (TCP, UDP), verbs (InifiBand, RoCE, iWarp), Cray uGNI, and a high-performance kernel-level Ethernet

[CCI HOTI'11] *S. Atchley, D. Dillow, G. Shipman, P. Geoffray, J. Squyres, G. Bosilca and R. Minnich, "The Common Communication Interface (CCI)"*
In the Proceedings of 19th IEEE Symposium on High Performance Interconnects (HOTI), 2011.

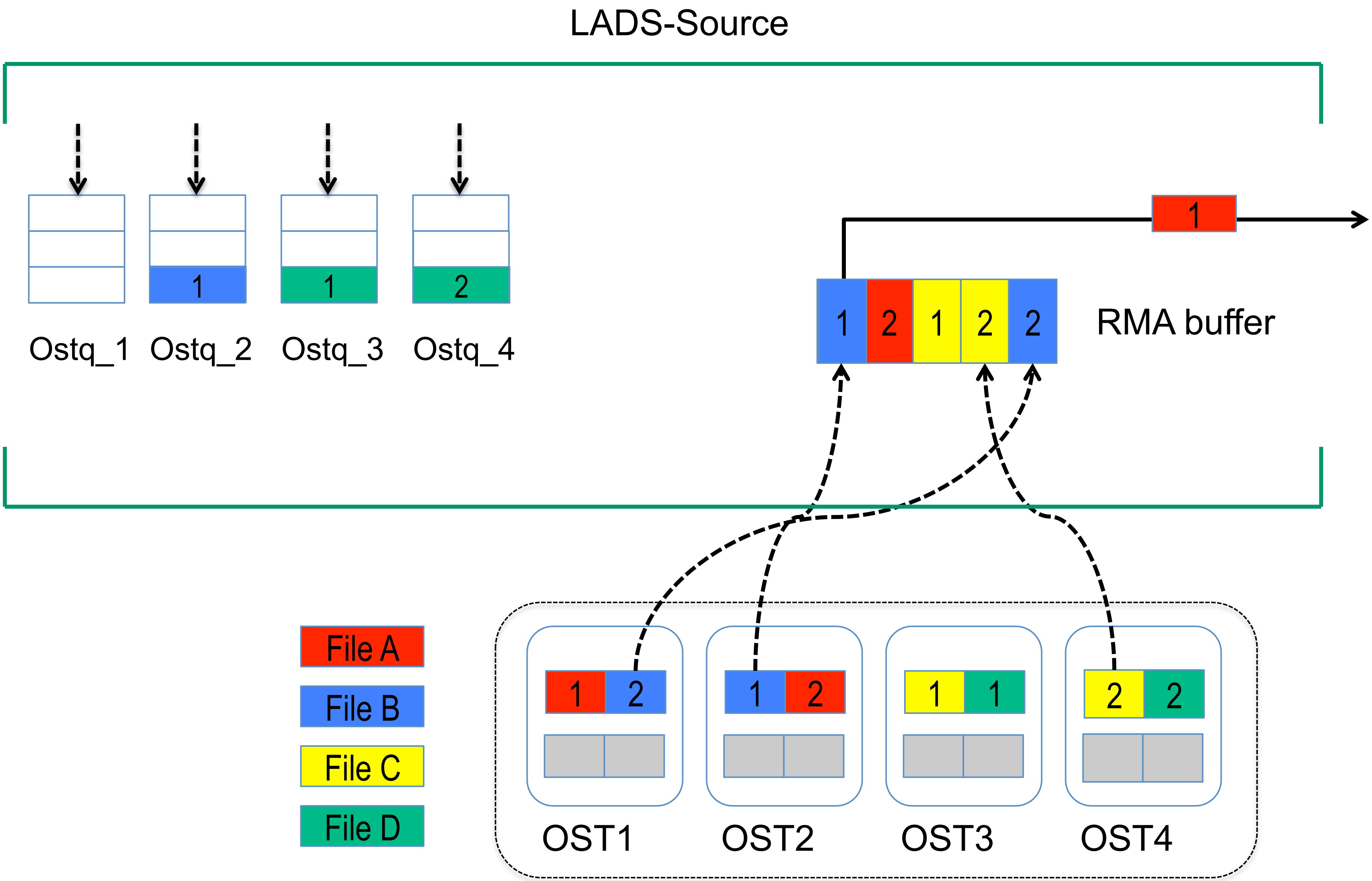
CCI Website: http://cci-forum.com`

LADS Architecture Overview



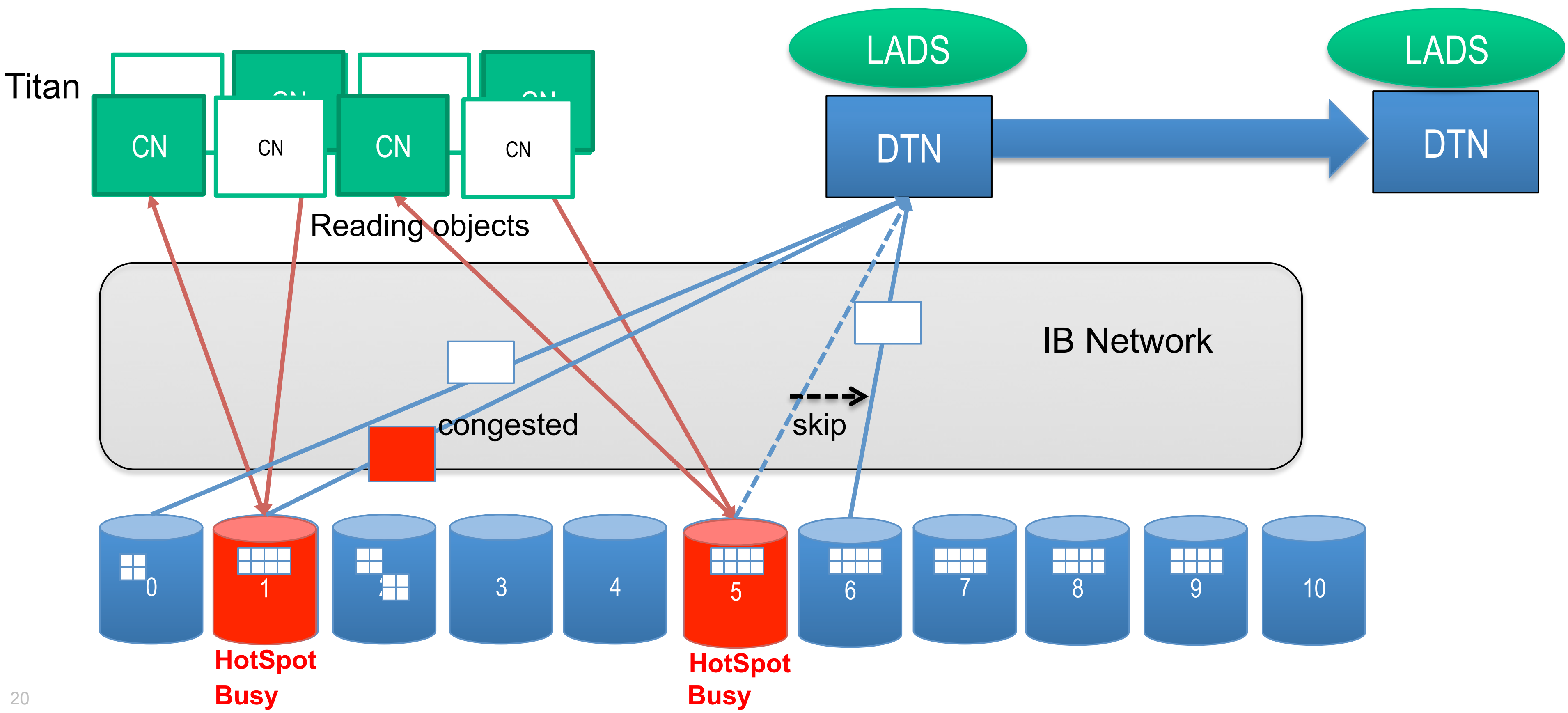
- CCI implementation
  - Construct endpoints (virtual device) and pre-register RMA buffers
- Threads Implementation
  - Queue based implementation with Master, I/O and Comm threads

LADS: Transferring Data at Source



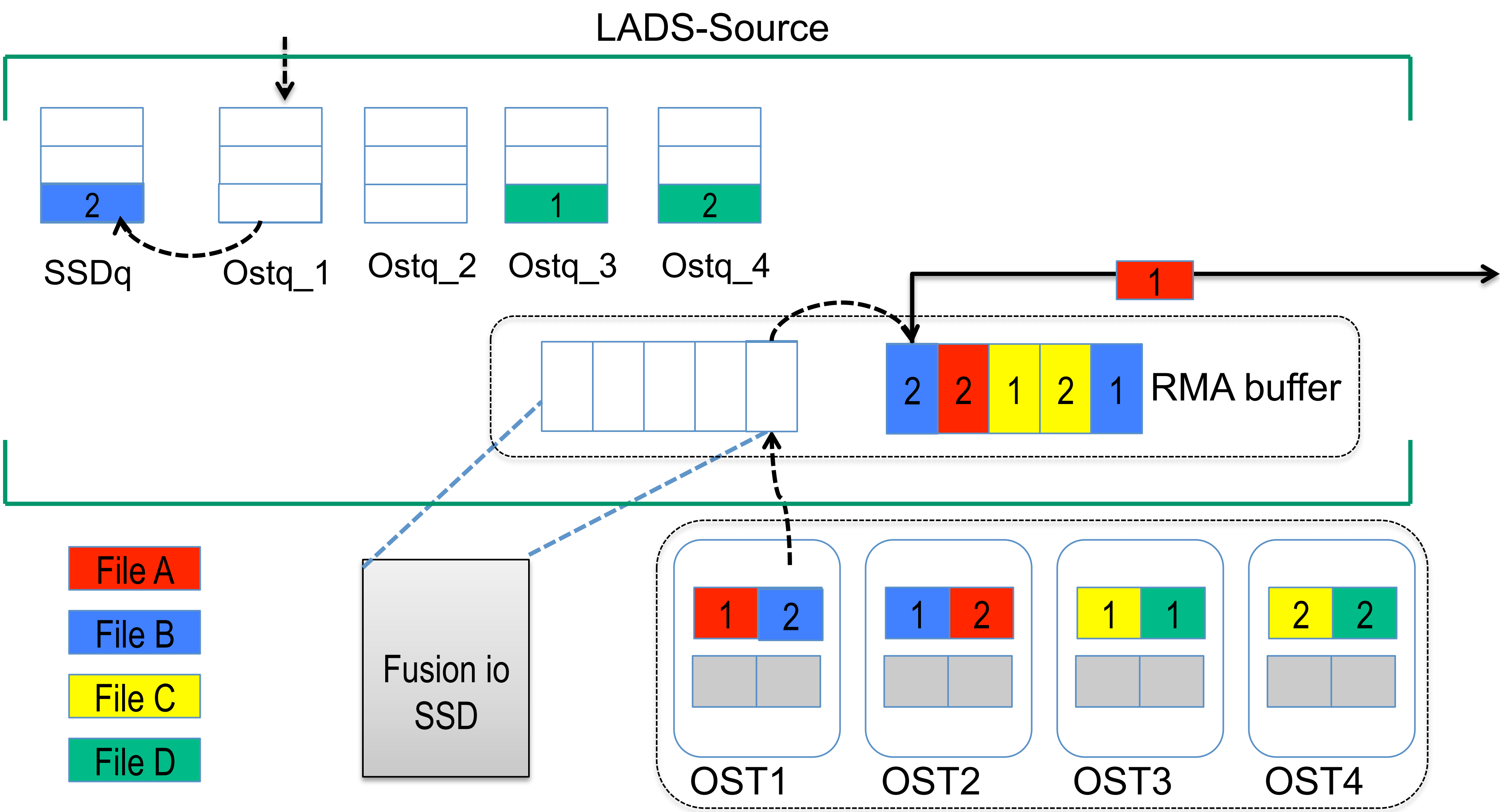
LADS: Congestion-Aware Algorithm

- Minimizing impact of intermittent congestion on storage servers
- Implemented a threshold-based reactive algorithm using
  - a preset value (object reading or writing time) for determining congested server
  - the number of skips on the congested servers

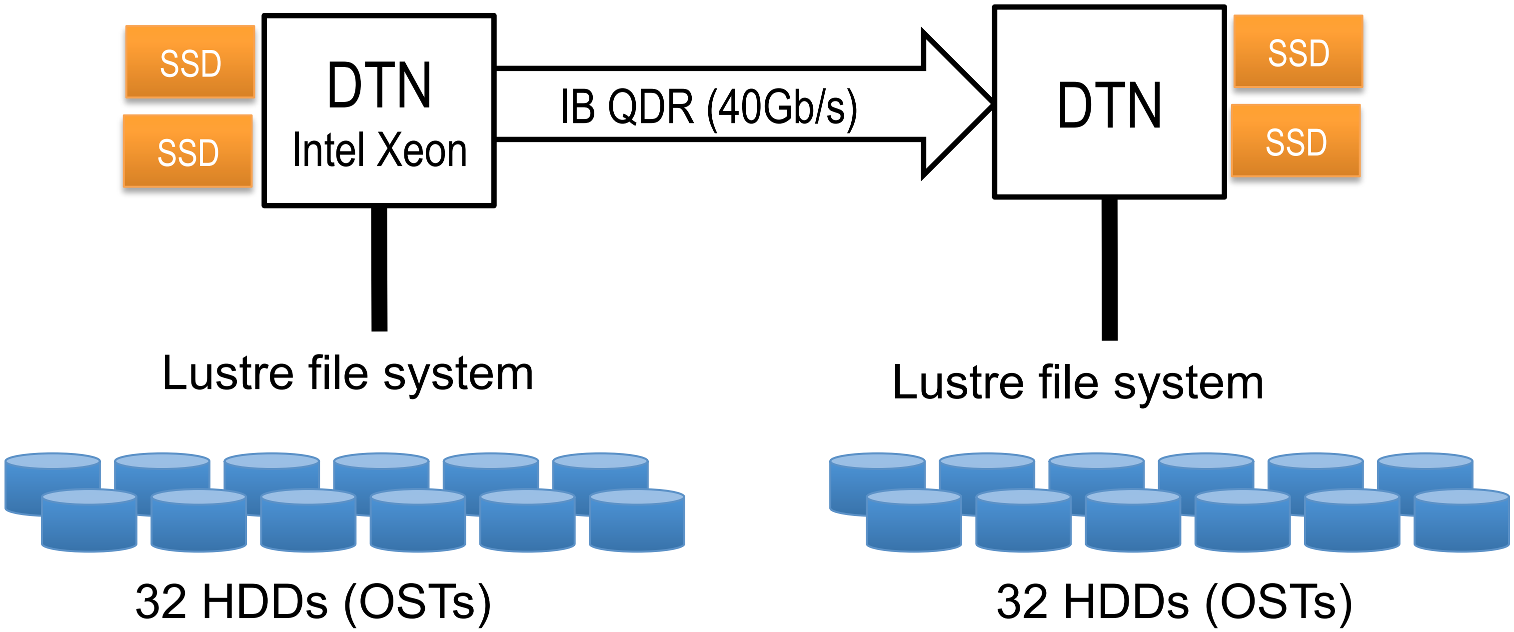


LADS: Source-based Buffering on NVM

- Source's RMA buffer full
  - The RMA buffer at source can be full if the sink is experiencing wide-spread congestion.

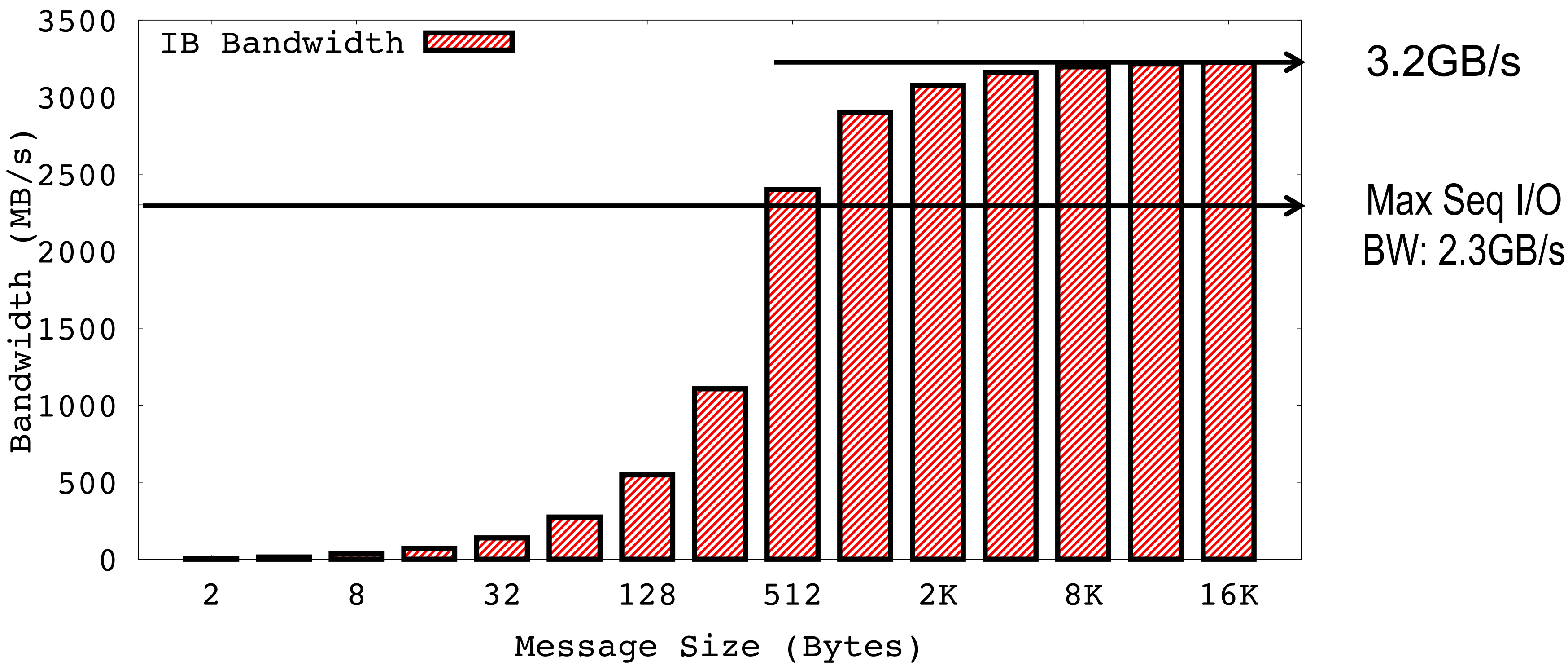


Test-bed Configuration



Validation of Test-bed

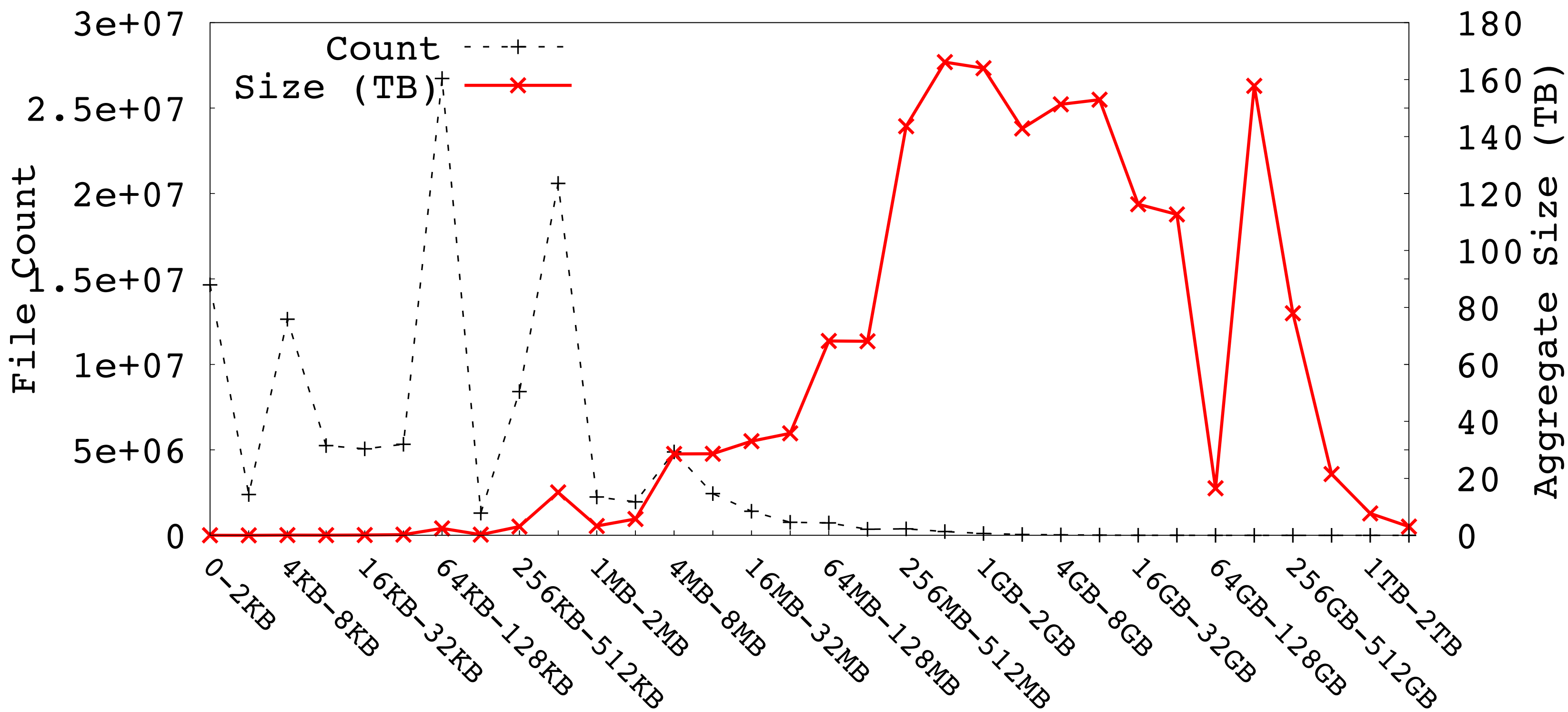
- Comparing IB bandwidth vs. Storage bandwidth



The storage server bandwidth is not over-provisioned with respect to network bandwidth.

Workloads

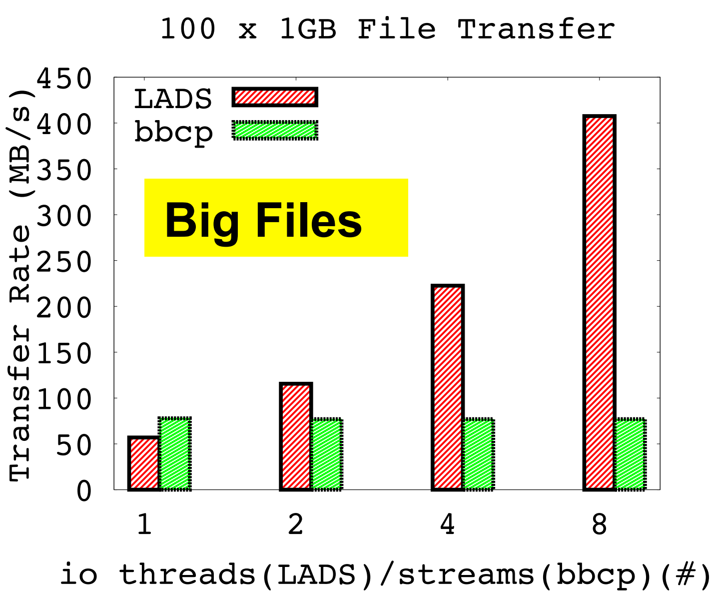
- File size distribution for a snapshot taken from the Spider-I file system

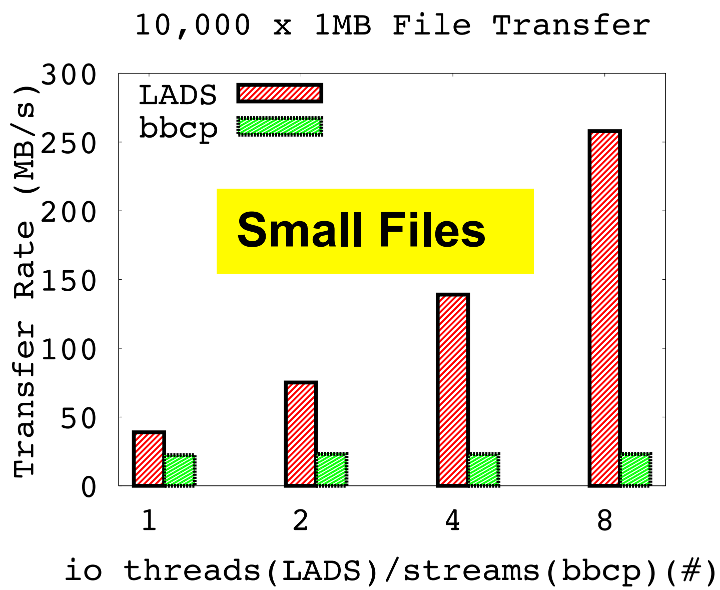


- 85% of the files are less than 1MB and less than 15% of the files are greater than 1MB.
- The larger files occupy most of the file system space.

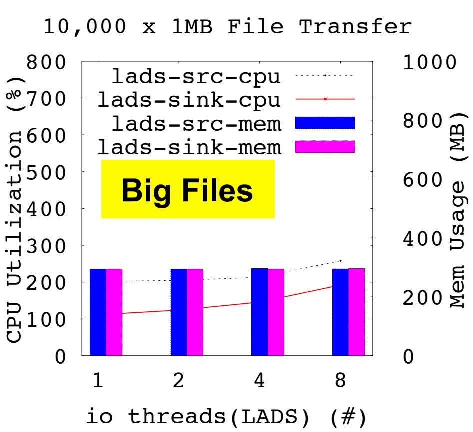
Comparison of LADS and bbcp

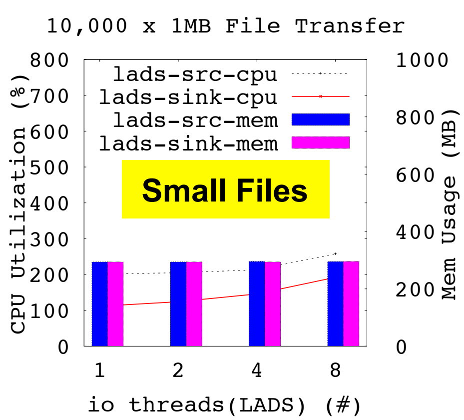
- Performance





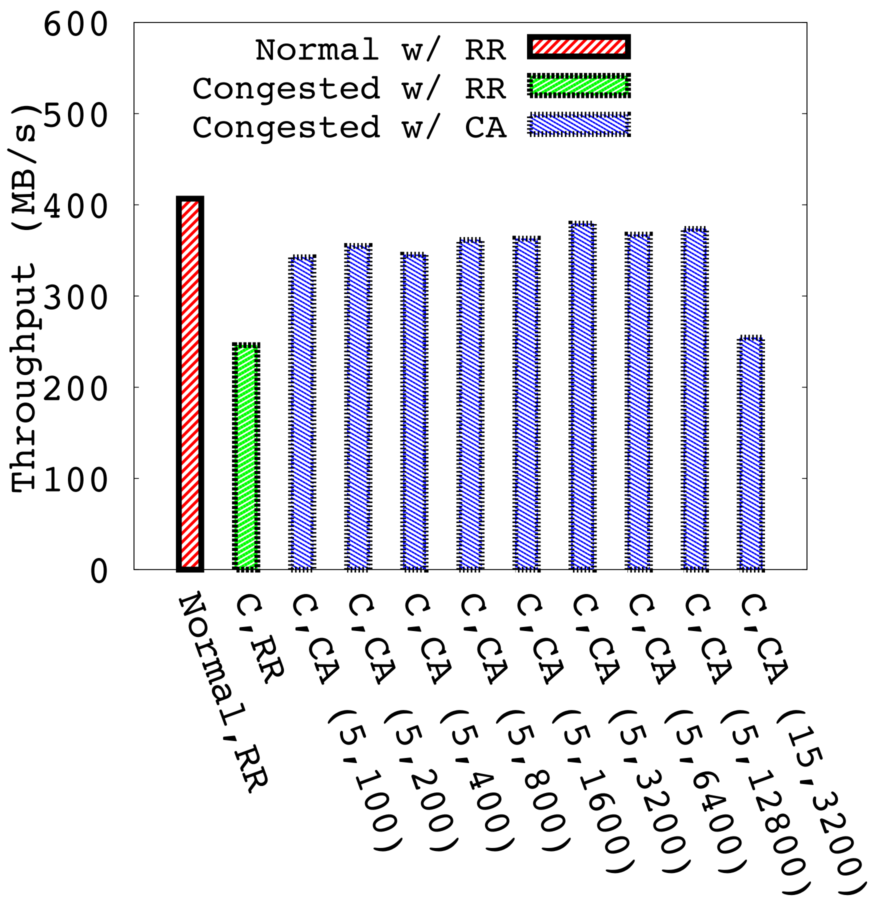
- Resource Utilization



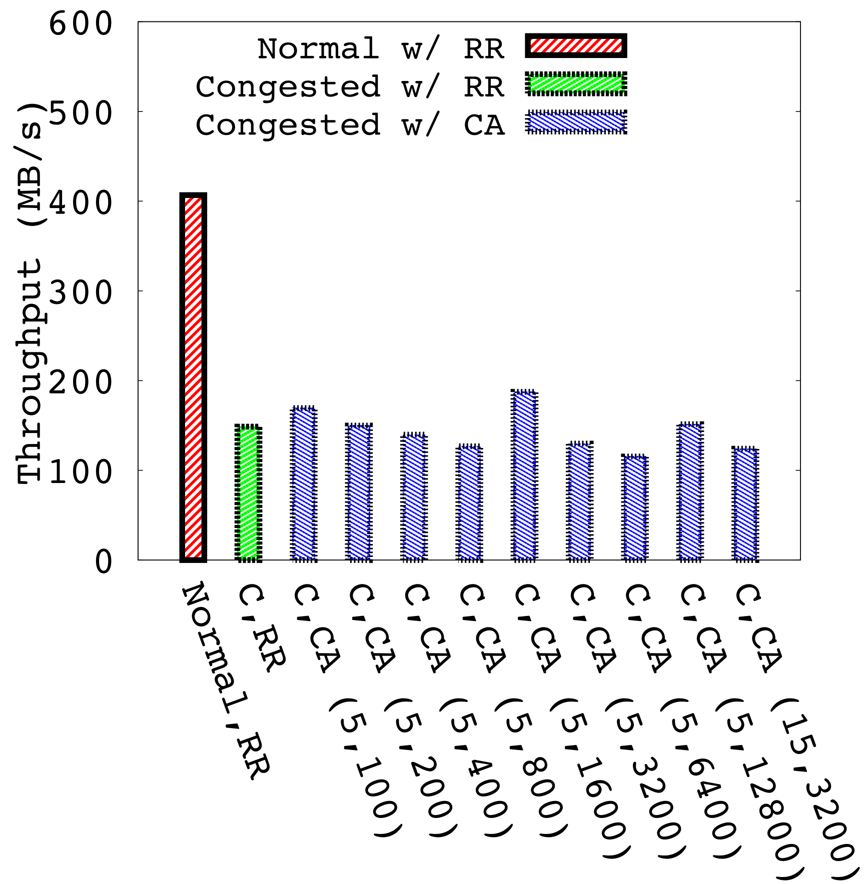


LADS with Storage in Contention

- Evaluation of storage congestion-aware algorithm in LADS



(a) Source storage congested



(b) Sink storage congested

More Experiments - Summary

- Effectiveness of the use of flash buffering at source
  - Throughputs increases as the available memory for communication at the source increases.
  - Doubling the size of DRAM is very expensive and the same throughput could be achieved using flash memory cheaper than DRAM.
- Evaluation between DTNs at ORNL
  - For this experiment, both LADS and bbcp uses Sockets (LADS uses a CCI setup to use its TCP transport).
  - LADS shows 6.8 times higher data transfer rate than bbcp.
  - bbcp shows slightly higher in throughput than LADS for a single thread.
  - In bbcp, I/O parallelism is limited to a stripe width of a file in Lustre (which is four in our evaluation).

Threads (#)	1	2	4	8
LADS	58.71	116.30	228.38	407.02
bbcp	59.91	58.46	57.85	59.49

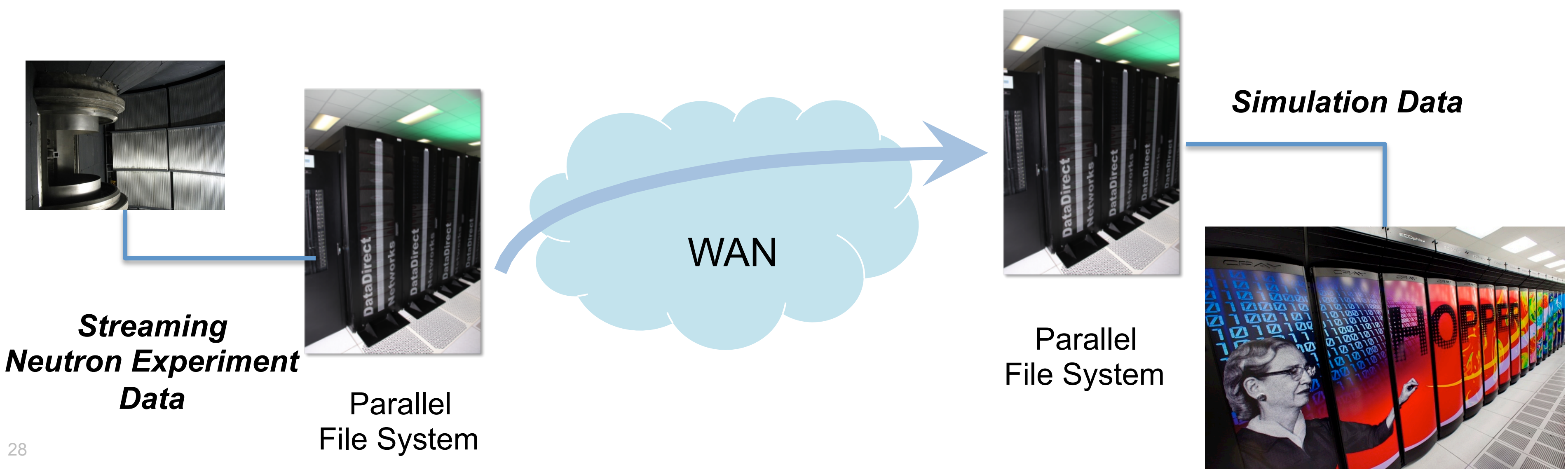
Throughput comparison (MB/s)

Summary

I. We identified multiple bottlenecks that exist along the end-to-end data transfer from source and sink host systems.

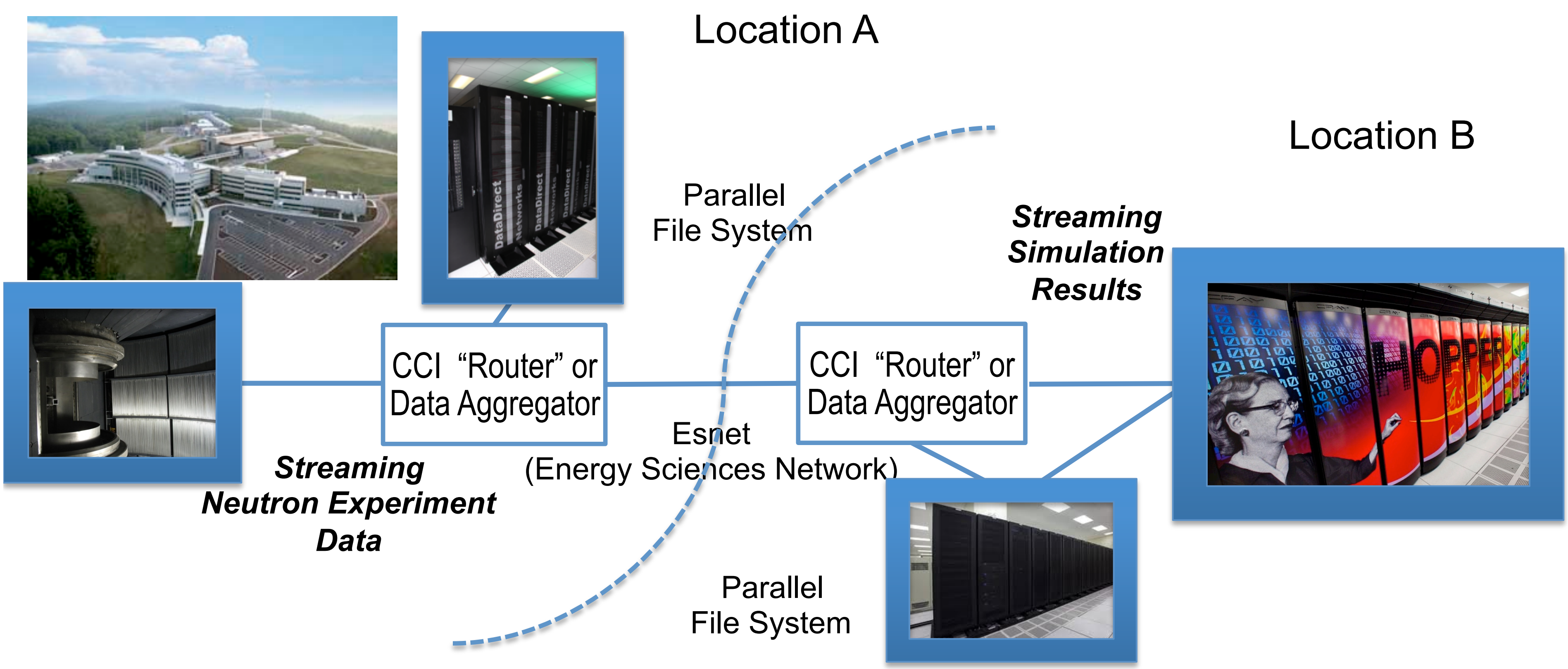
II. We developed LADS to demonstrate techniques that can alleviate some end-to-end bottlenecks while at the same time, negatively impact the use of the PFS by other resources.

III. We investigated three I/O optimization techniques: I/O slicing, layout-aware and congestion-aware I/O scheduling, and source-side SSD buffering.



Our Vision

- *An optimized end-to-end virtual path from any source to any sink*



A variety of data sources and sinks must be supported to transition from traditional data movement to streaming experiment/simulation data

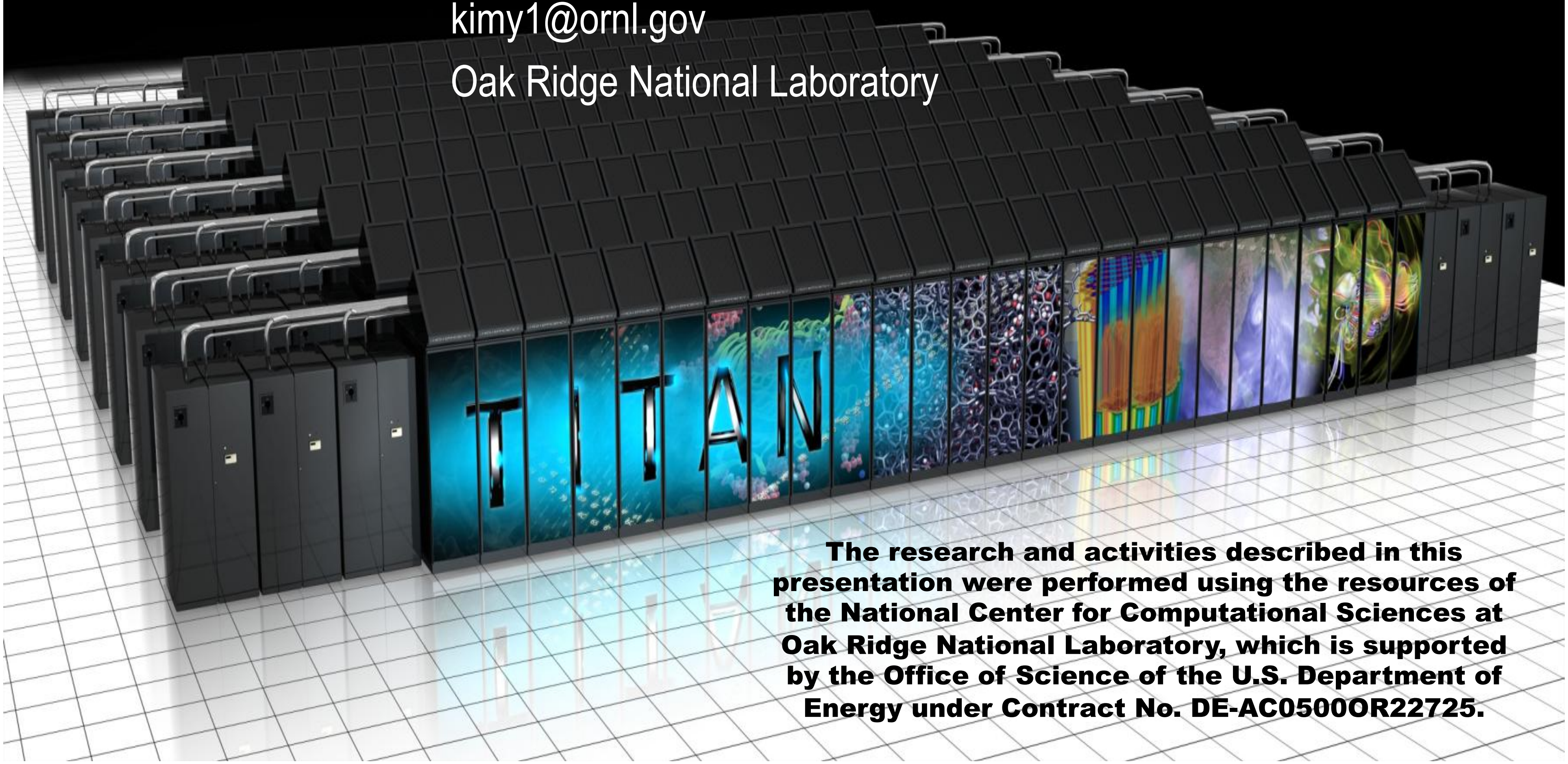
Questions?

Contact info

Youngjae Kim (PhD)

kimy1@ornl.gov

Oak Ridge National Laboratory



The research and activities described in this presentation were performed using the resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC0500OR22725.