Integrate Hadoop Distributed File System to Logistical Storage

- a cost efficient storage solution

Shunxing Bao
ACCRE, Vanderbilt University

Mar 25th, 2019

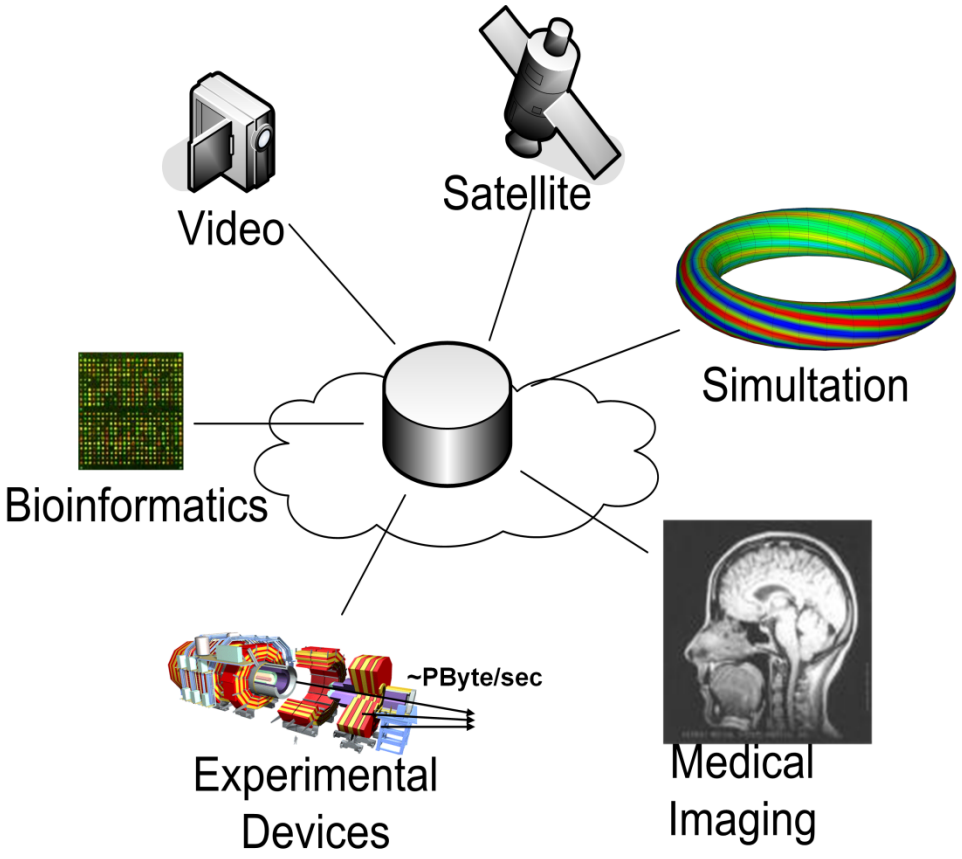
Overview

- Logistical Storage (LStore)
  - Logistical Networking
  - LStore Architecture
- LStore new feature
  - HDFS plugin

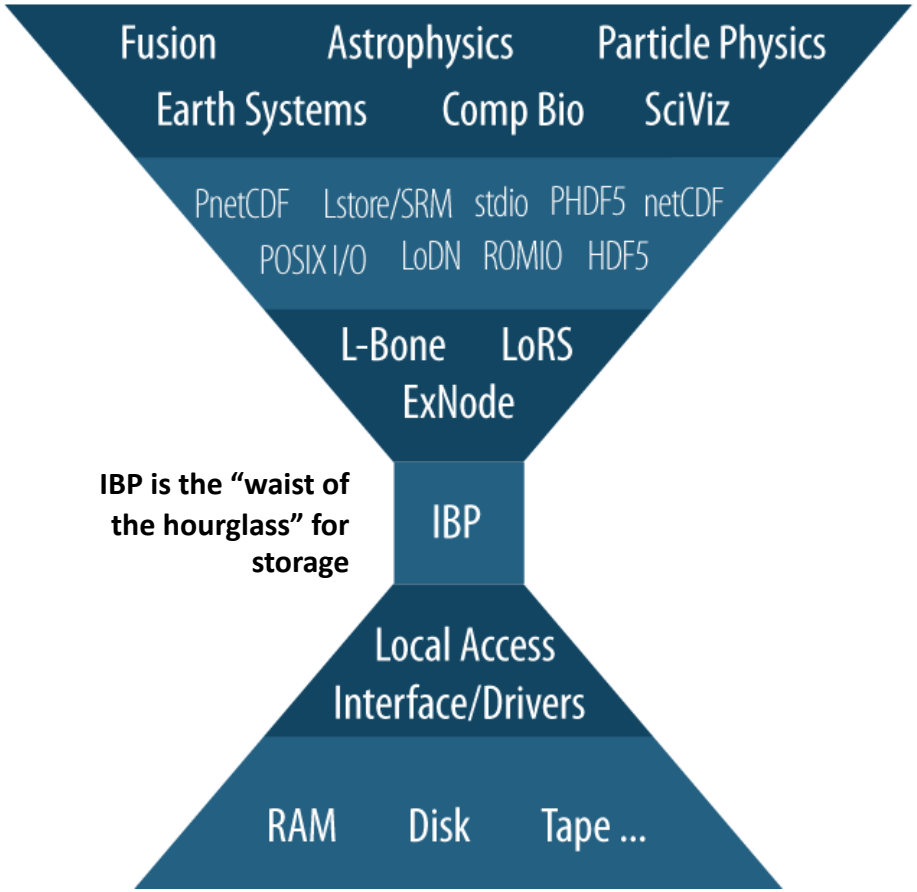
Logistical Networking (LN) provides a "bits are bits" Infrastructure

- Standardize on what we have an adequate common model for
  - Storage/buffer management
  - Coarse-grained data transfer
- Leave everything else to higher layers
  - End-to-end services: checksums, encryption, error encoding, etc.
- Enable autonomy in wide area service creation
  - Security, resource allocation, QoS guarantees
- **Gain the benefits of interoperability!**

One structure to serve them all!



LoCI Tools* : Logistical Computing and Internetworking Lab



- IBP Internet Backplane Protocol
  - Middleware for managing and using remote storage
  - Allows advanced space and **TIME** reservation
  - Supports multiple connections/depot
  - User configurable block size
  - Designed to support large scale, distributed systems
  - Provides global *"malloc()"* and *"free()"*
  - End-to-end guarantees
  - Capabilities

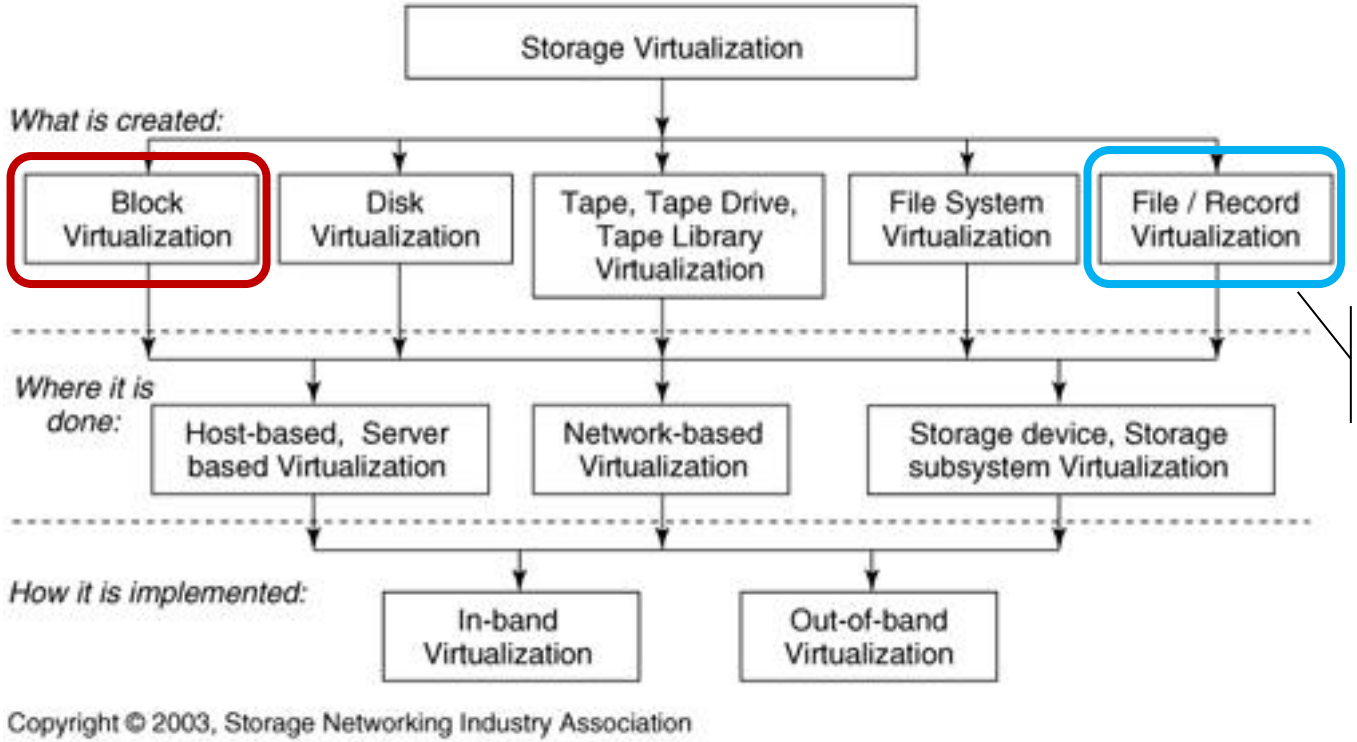
*http://loci.cs.utk.edu

What is a “capability”?

- Controls access to an IBP allocation
- 3 separate allocation keys
  - Read
  - Write
  - Manage - delete or modify an allocation
- Alias or Proxy allocations supported
  - End user never has to see true capabilities
  - Can be revoked at any time

What makes LN Different?

Based on a highly generic abstract block for storage (IBP)



Logistical Networking

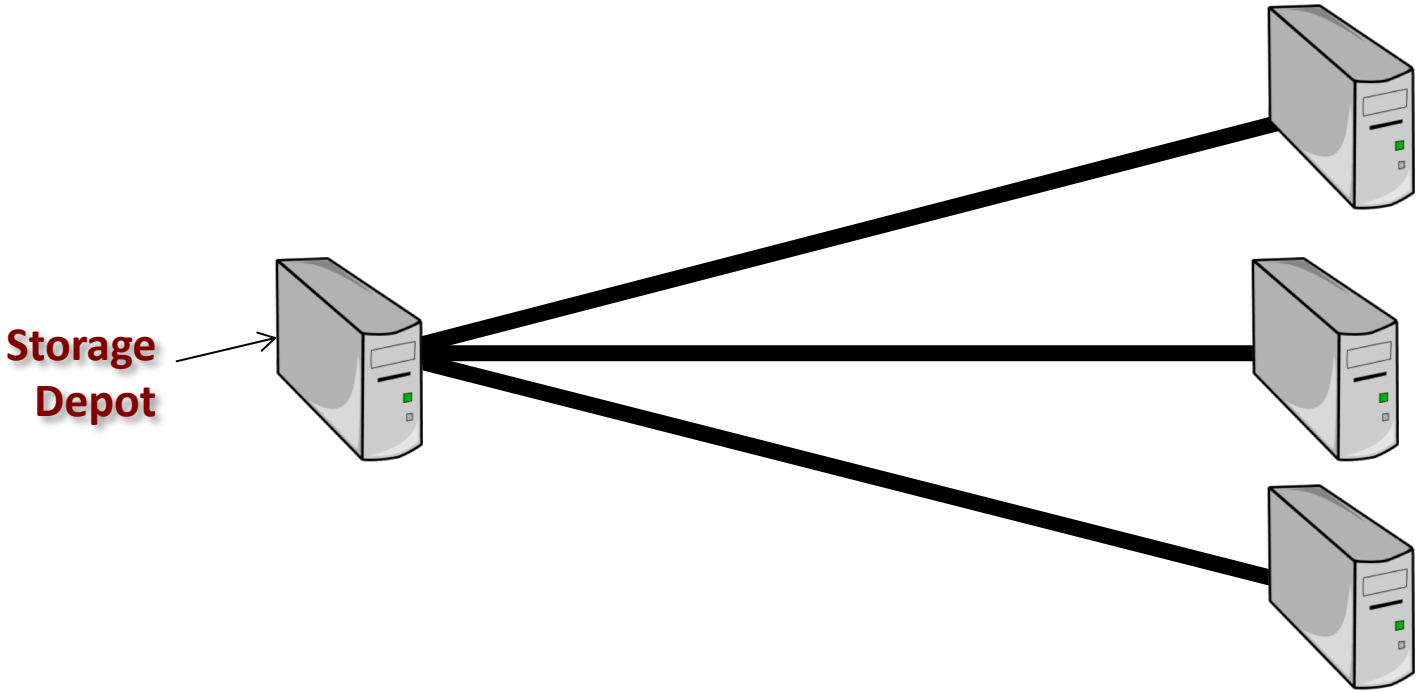
Traditional Approach

What makes LN Different?

Based on a highly generic abstract block for storage (IBP)

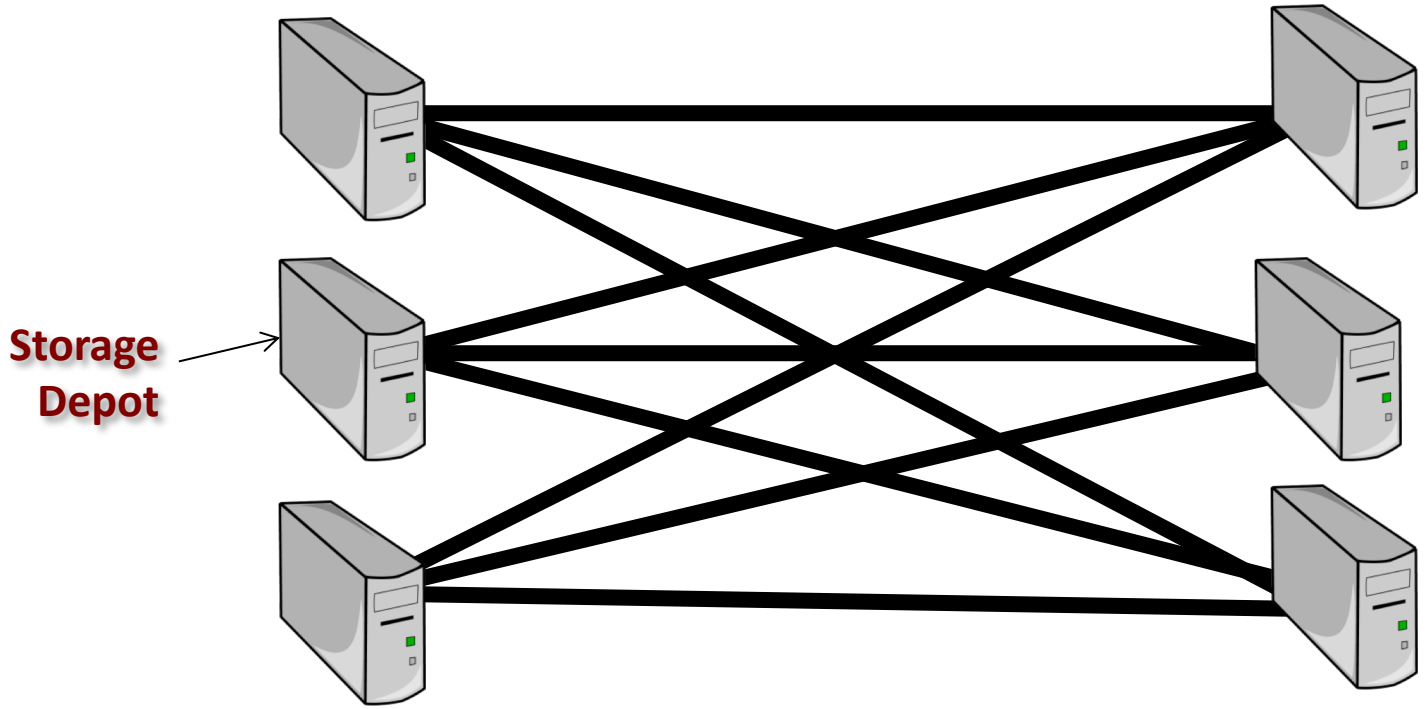
What makes LN Different?

Based on a highly generic abstract block for storage (IBP)



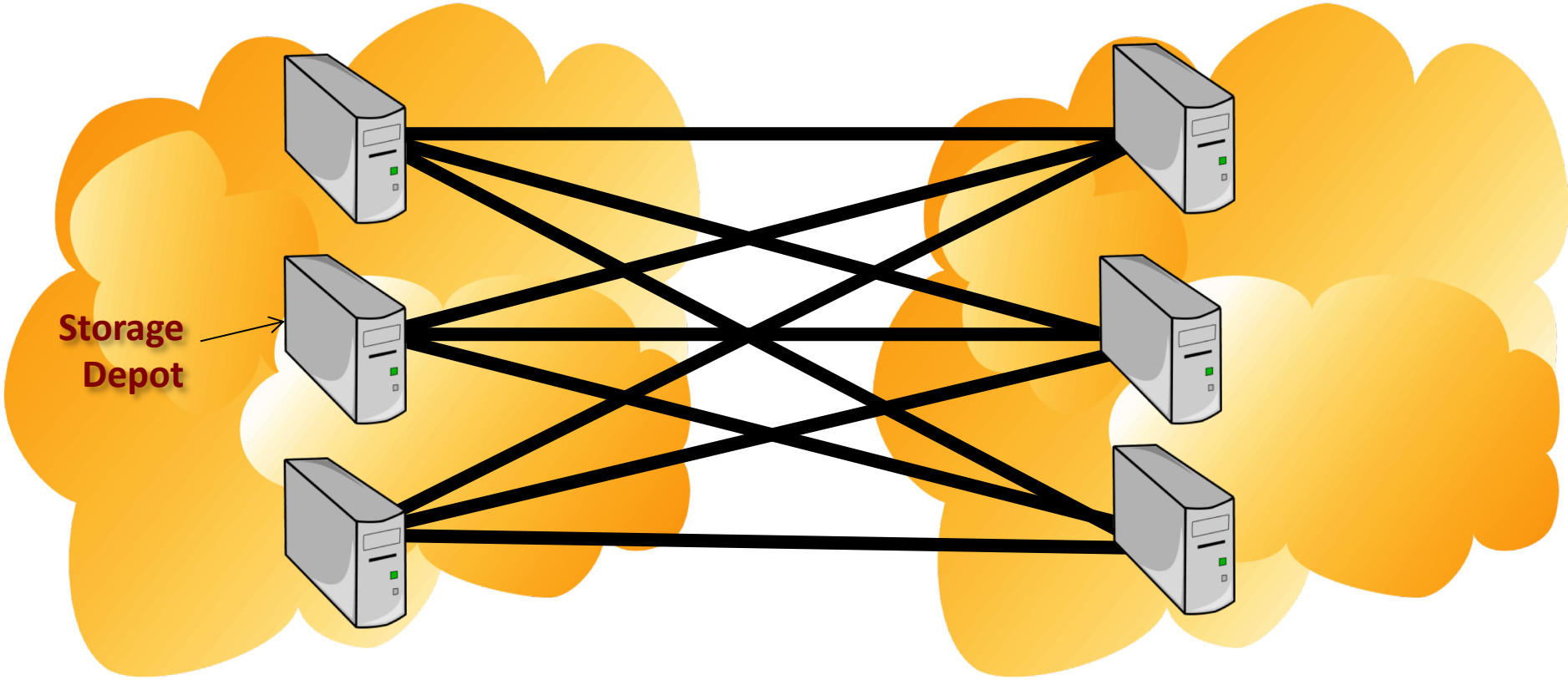
What makes LN Different?

Based on a highly generic abstract block for storage (IBP)



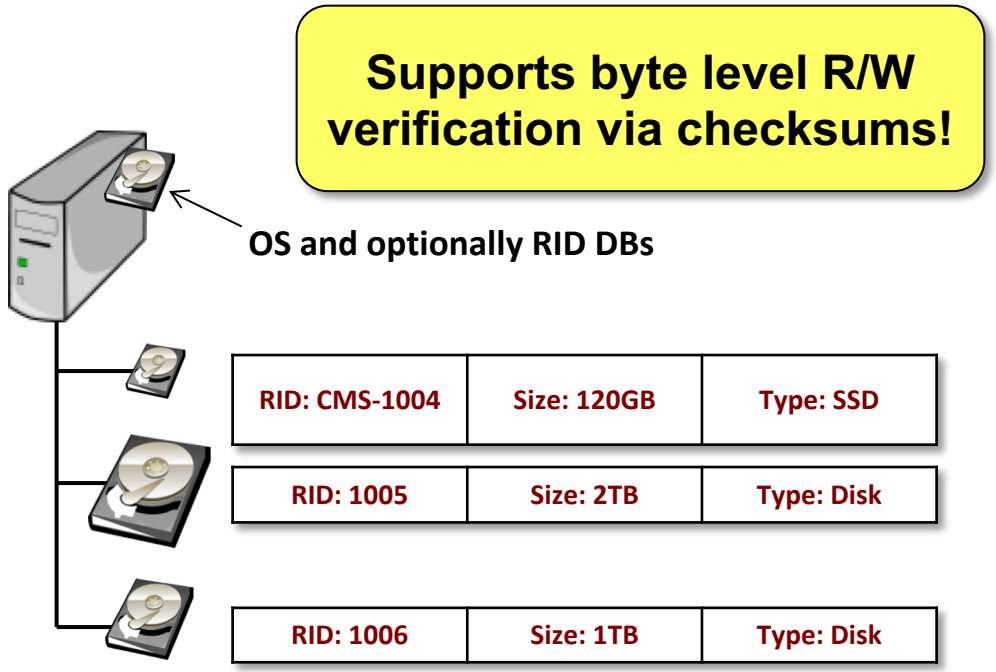
What makes LN Different?

Based on a highly generic abstract block for storage (IBP)



IBP Server (Depot)

- Runs the ibp_server process
- Resource
  - Unique ID
  - Separate data and metadata partitions
  - Optionally can import metadata to SSD
- Typically JBOD disk configuration
- Heterogeneous disk sizes and types
- Don’t have to use dedicated disks



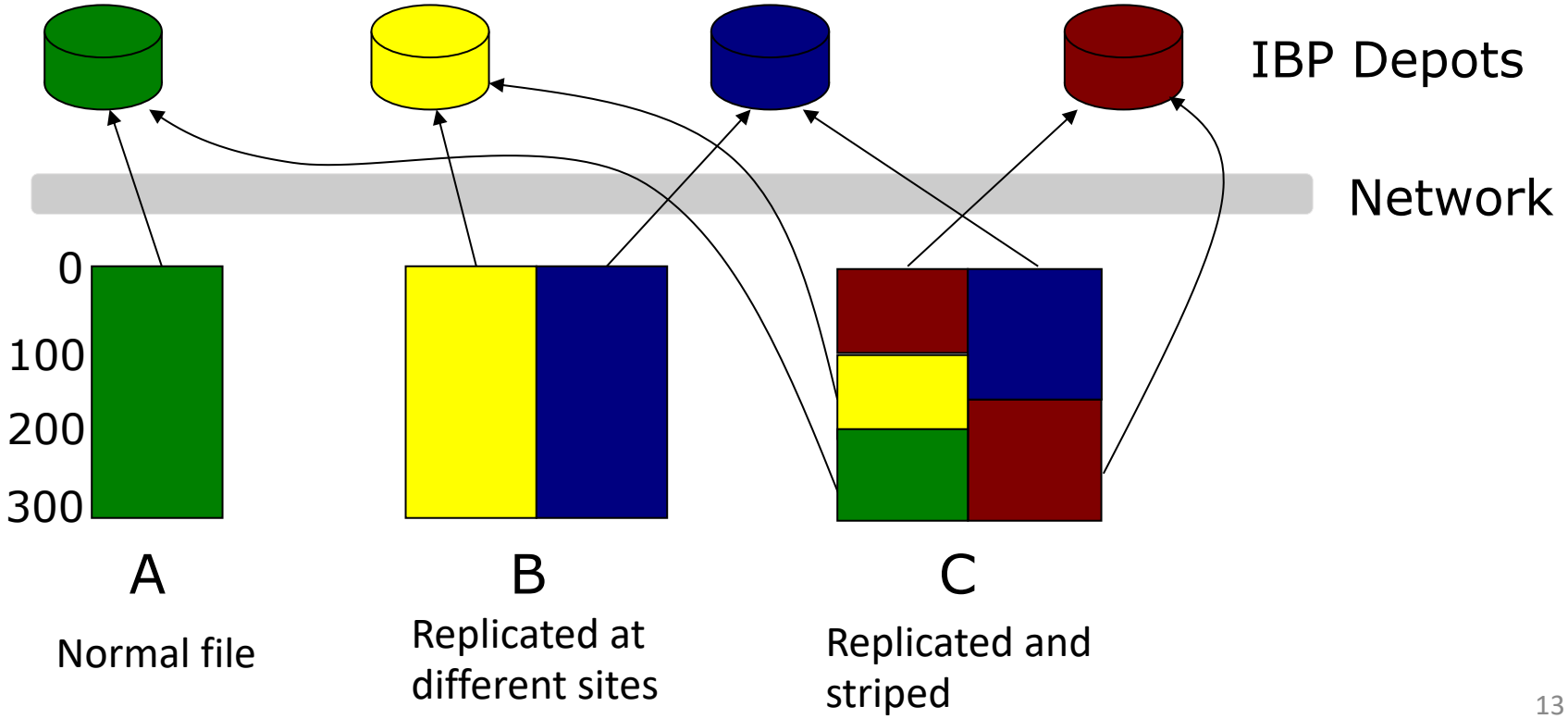
- **Current depots have 36 10TB drives and can sustain 15Gb/s (disk check summing to protect against bit rot) to 20Gb/s (no disk checksum)**
- **150Gb/s is the currently highest sustained transfer rate and was network limited.**

Overview

- Logistical Storage (LStore)
  - Logistical Networking
  - LStore Architecture
- LStore new feature
  - HDFS plugin

LStore Architecture

- **Exnode : File metadata object**
  - A data structure for aggregation, analogous to the Unix inode
  - Allocations
  - How to assemble file
  - Fault tolerance encoding scheme
  - Encryption keys
- **LStore File System**
  - Metadata server (exNodes)



LStore Architecture

- **Exnode**
  - Collection of data blocks and segments to provide different views of data
  - Different segments can be used for versioning, replication, optimized access (row vs. column), etc.
  - **Segment**
    - Collection of blocks with a predefined structure.
    - Type: Linear, LUN, RAID5, Generalized Reed-Solomon, Log, caching, etc.
    - Can be stacked with other segments
  - **Resource Service**
    - Data placement(stripe across depots vs. across disks) and lookup
    - Boolean query expression
  - **Data Service**
    - Performs the actual data operations
    - Currently only IBP is supported

- **Object Service**
  - Metadata operations
  - Full support for streaming operations to minimize latency effects. Most operations can be implemented with a single call (ls -l, mkdir, find, etc)

LStore Access Interface

- **LStore command line tools**
  - **The Logistical Input/Output (LIO) command line tools are designed to replicate the normal Linux File System Tools**
- **Linux FUSE client**

Linux

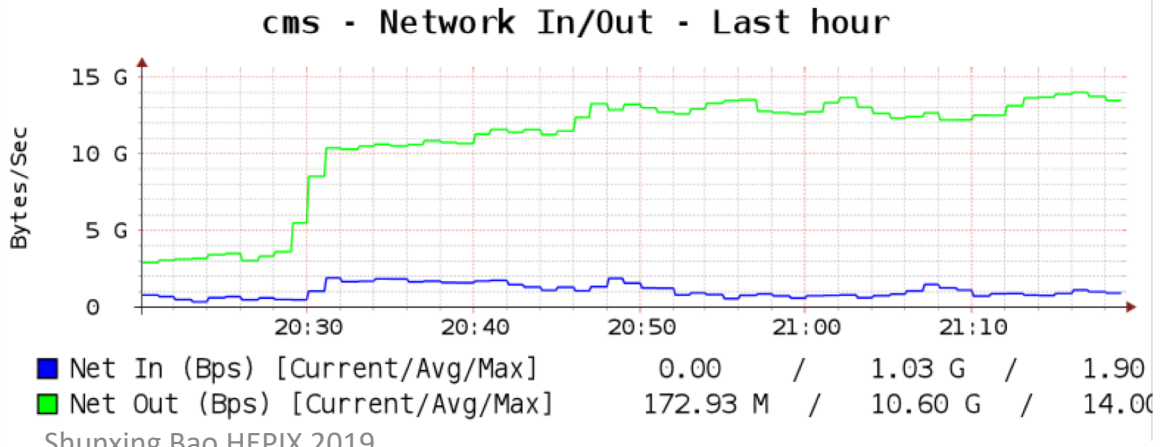
lio_cp	lio_mkdir
lio_du	lio_mv
lio_find	lio_rm
lio_fsck	lio_rmdir
lio_ln	lio_touch
lio_setattr	lio_getattr
lio_ls	

LIO Specific

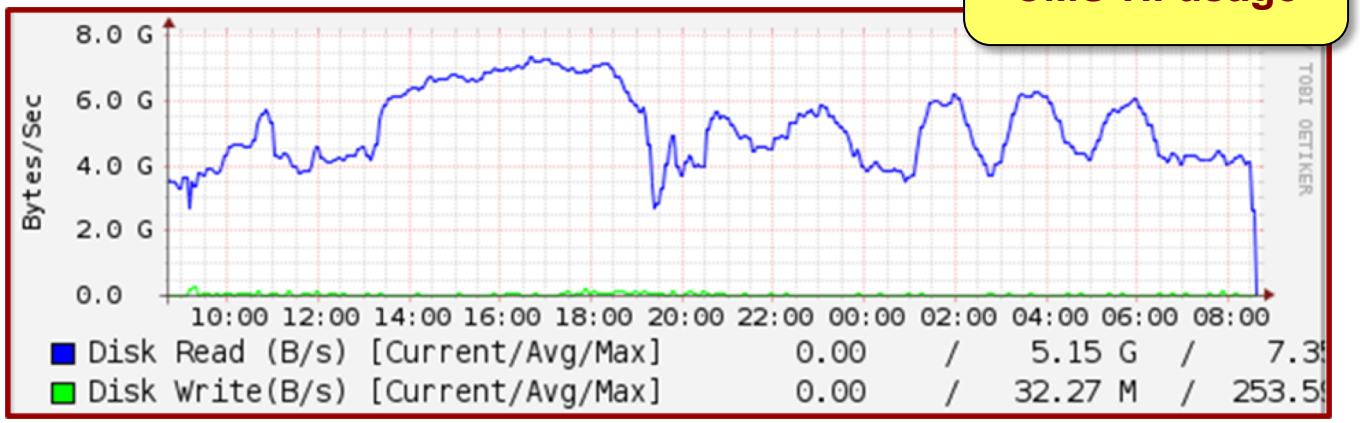
lio_fuse	lio_get
lio_inspect	lio_put
lio_signature	lio_warm

LStore Usage at VU

- CMS Heavy-Ion has 10PB of native disk space
  - Routinely read over 1PB/week using the FUSE mount for production jobs and sustain 120Gb/s read rates.
- 600TB of shared space at VU
  - Vanderbilt TV News Archive has 400+ TB of space



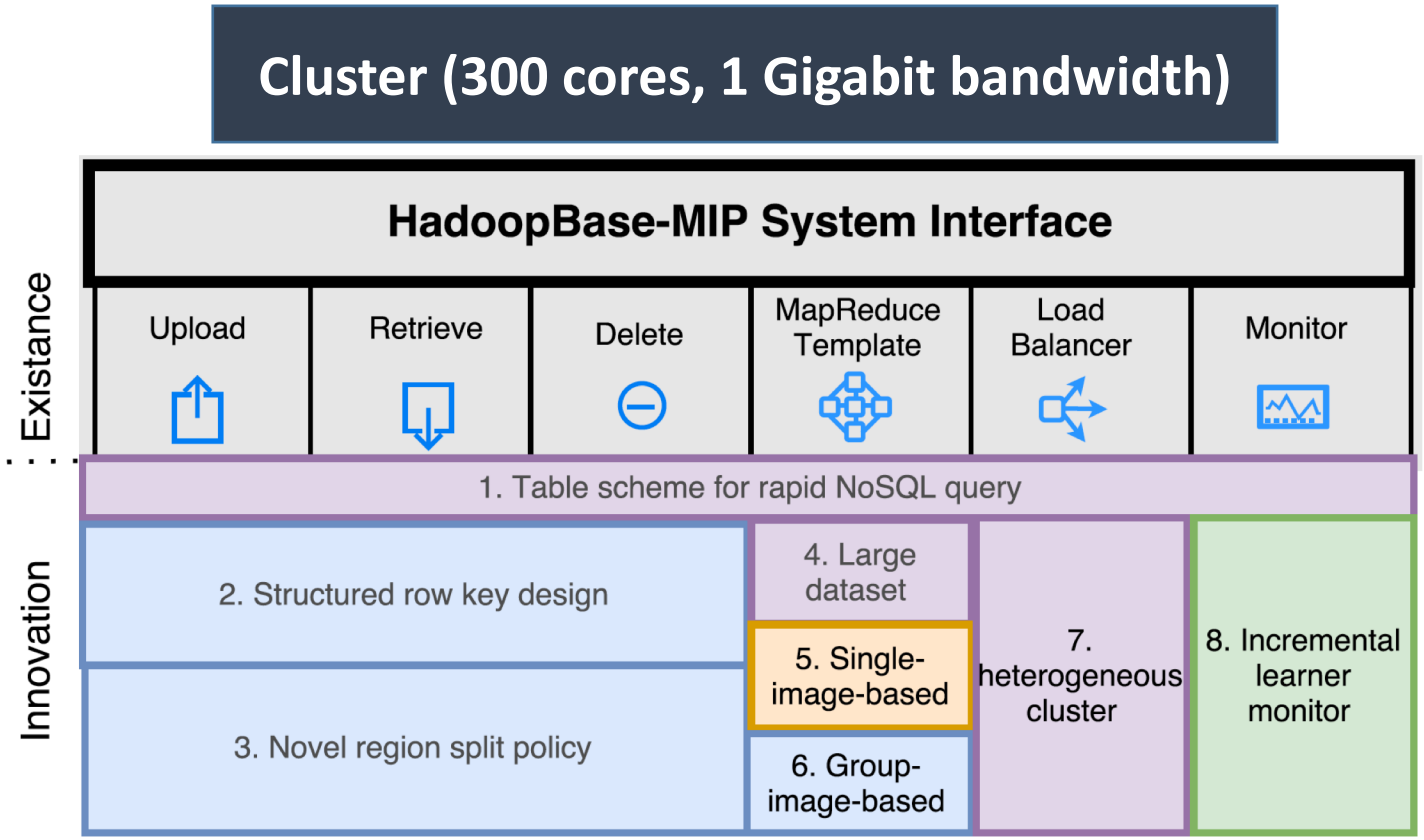
24 hour plot of CMS-HI usage

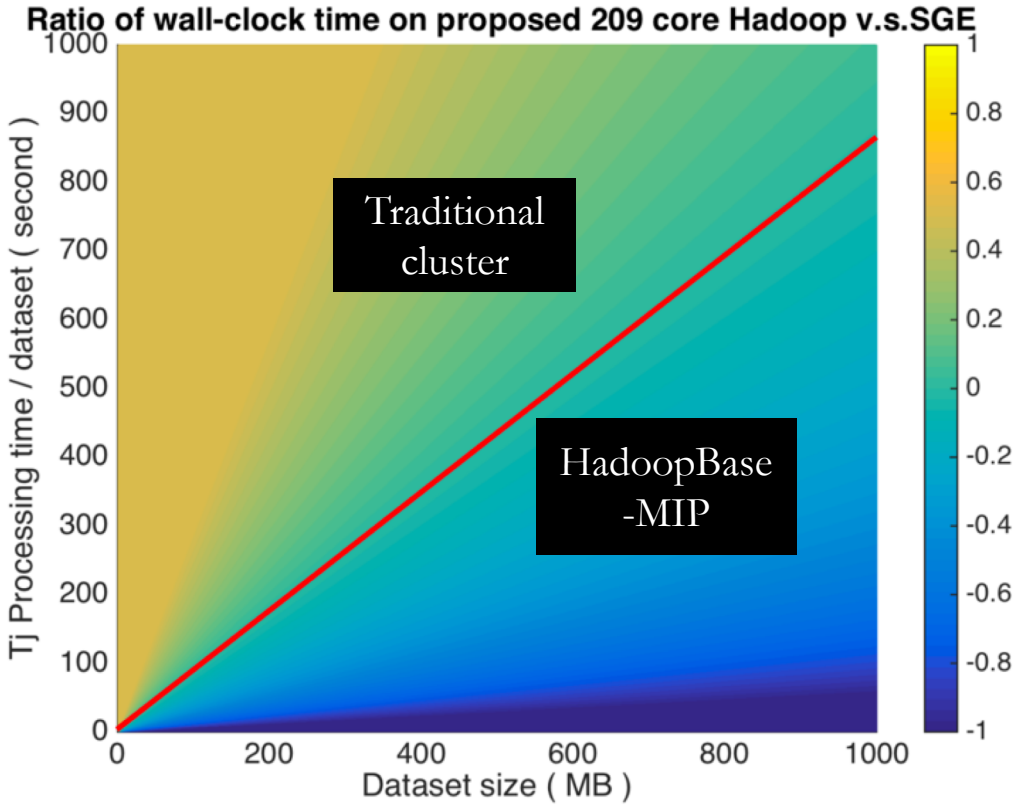


Overview

- LStore
  - Logistical Networking
  - LStore Architecture
- LStore new feature
  - HDFS plugin

Background



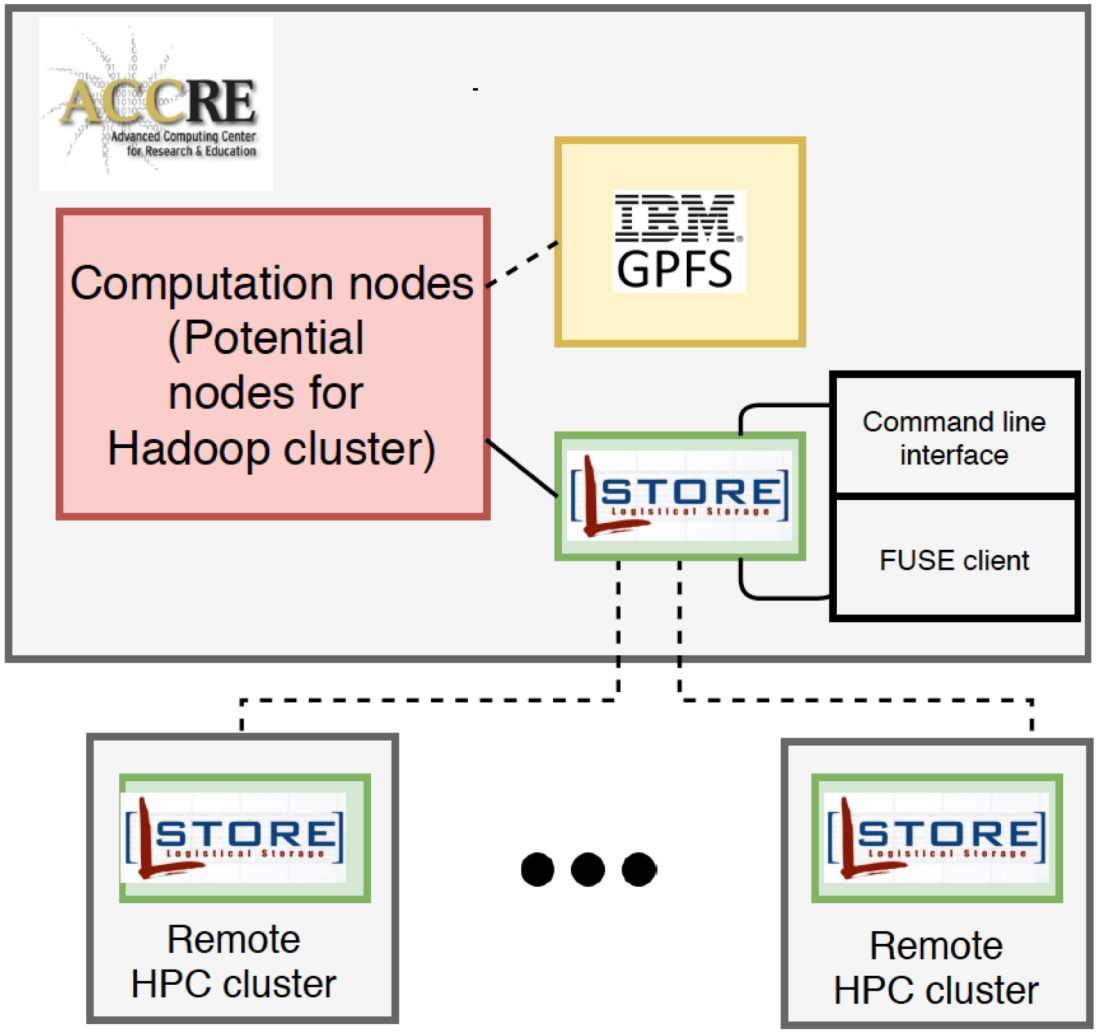


HadoopBase-MIP

Hadoop & HBase-based toolkit for medical image processing

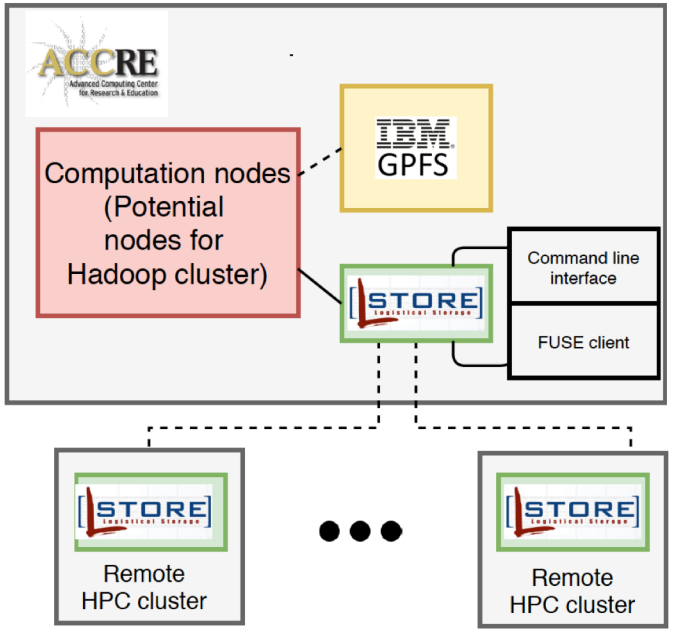
Background

- Vanderbilt ACCRE Datacenter
  - IBM GPFS
  - LStore
- Medical image processing at ACCRE
  - 557,986 raw image scans (T1, fmri, diffusion)
  - Average file size is 10 MB including the processed image.
  - Approximately 100 TB data in total
  - 96,103 medical image processing jobs, 15 seconds to 9 days. (2011-2016)



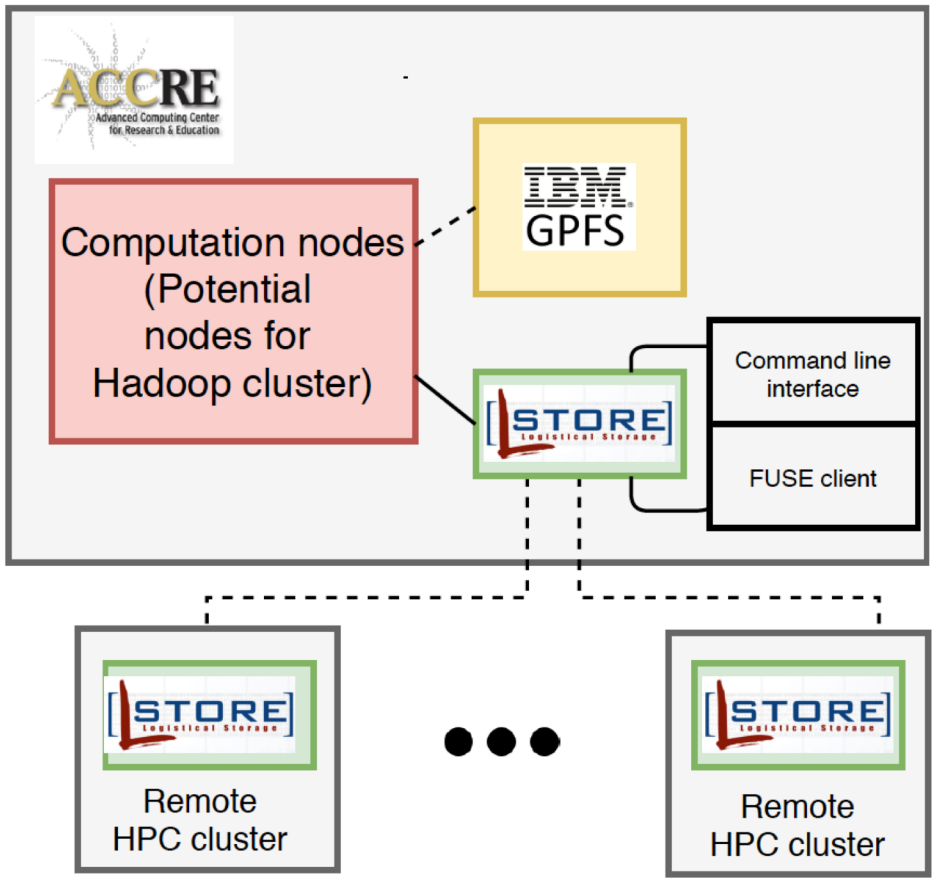
Step back: Motivation and challenge of integrating HDFS to LStore

- Case study: Integrate HDFS to LStore
  - Cannot get rid of this existing framework.
  - LStore is much cheaper when compare it with GPFS
    - RAID-6
  - Data redundancy (if data already in LStore, no need to import to HDFS)
  - Create HDFS on the fly
- Ultimate goal
  - **Generalize** the integration: HDFS to any HPC storage



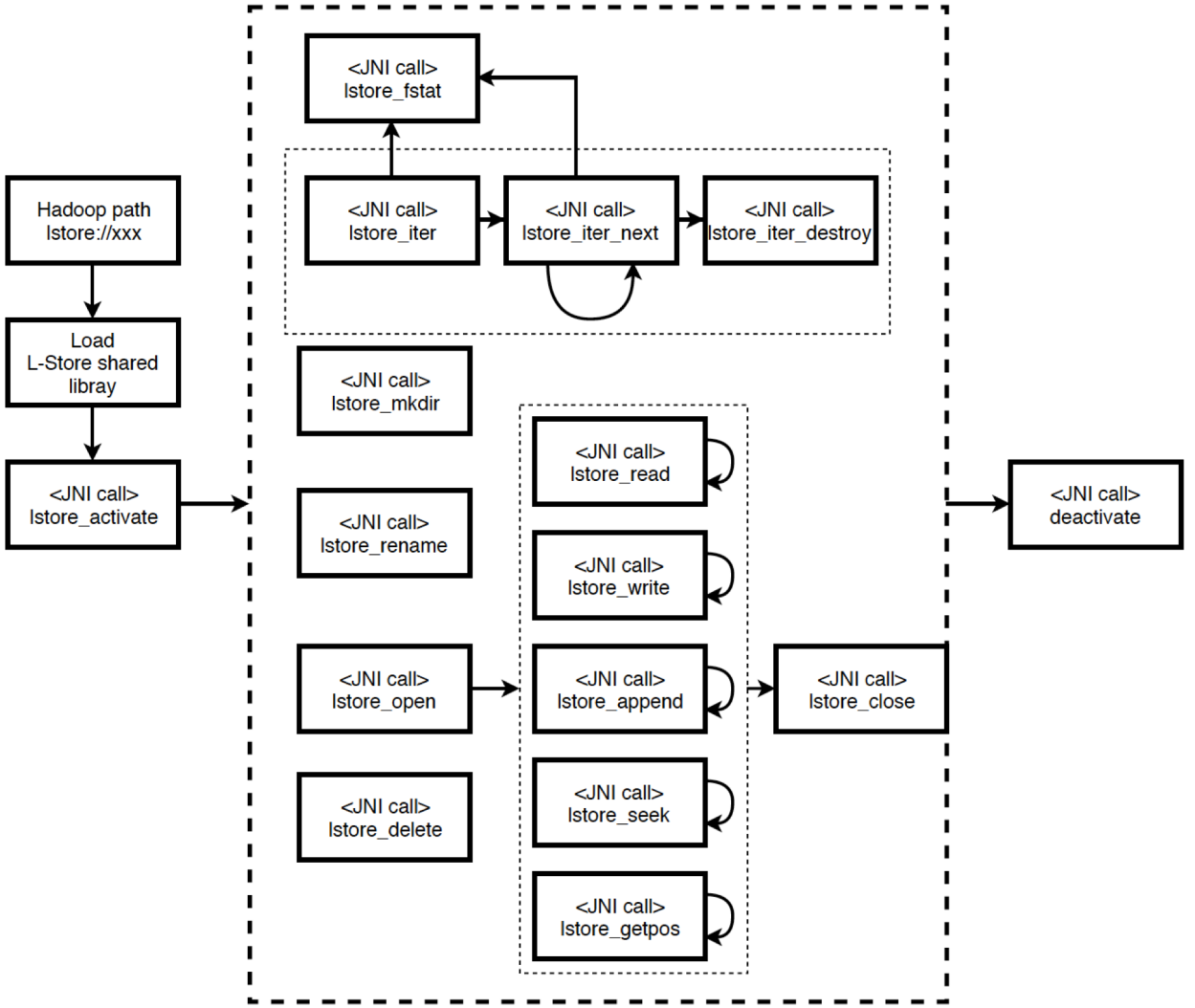
Challenge of integrating HDFS to ACCRE

- Challenge - How to make HDFS utilize LStore storage for large dataset transfer?
  - LStore Access Command line interface
  - LStore FUSE client - limit buffer size for read / write (128 KB/ request). HDFS and LStore command line: buffer size is tunable (e.g. > 80 MB /request)
- Goal
  - HDFS-LStore ≈ LStore command line tool
  - HDFS-LStore >> LStore FUSE
  - Alternative solution rather than GPFS



Our solution - system design

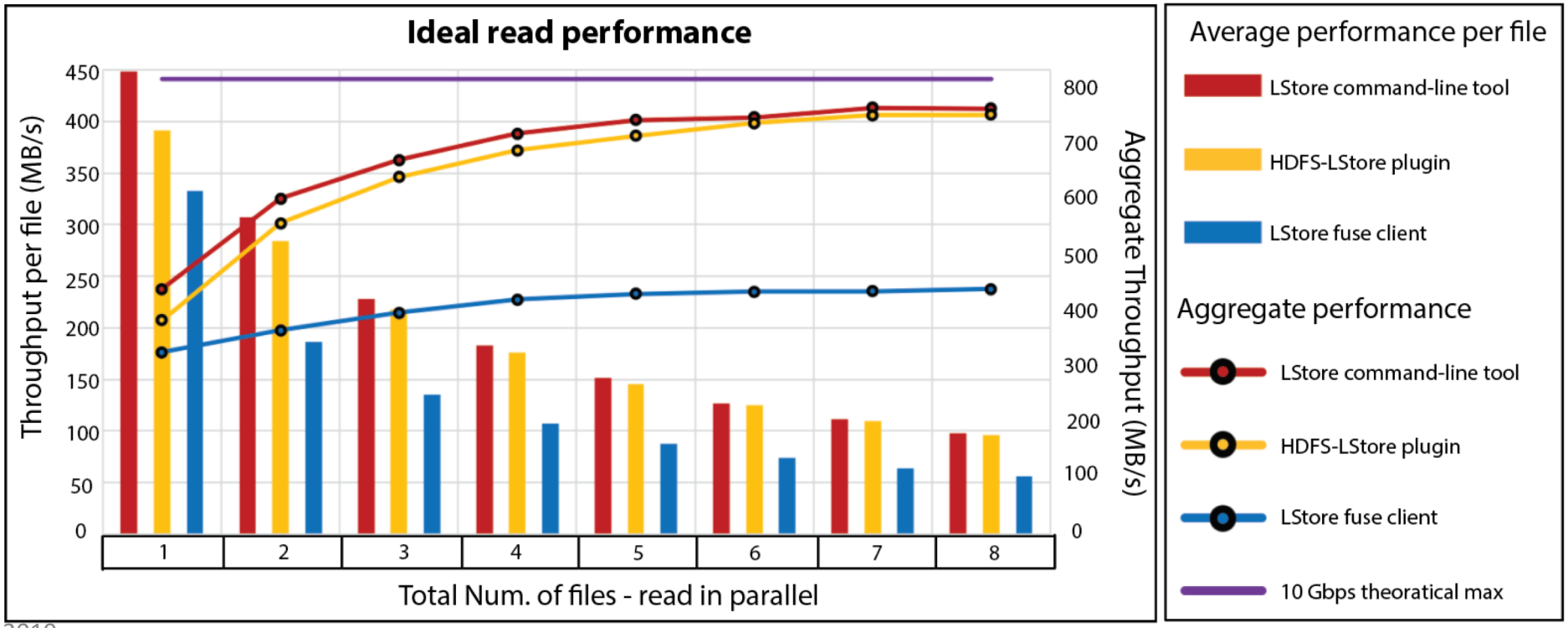
- Bypass FUSE client
- LStore side
  - Command line tool C shared wrapper library
- HDFS side
  - Java native interface
  - LStore schema (lstore://)
  - org.apache.hadoop.fs.FileSystem.LStoreFileSystem
  - class LstoreBaseFile - open / close / seek / read / write / append
  - MapReduce integration



https://github.com/cerndb/hadoop-xrootd

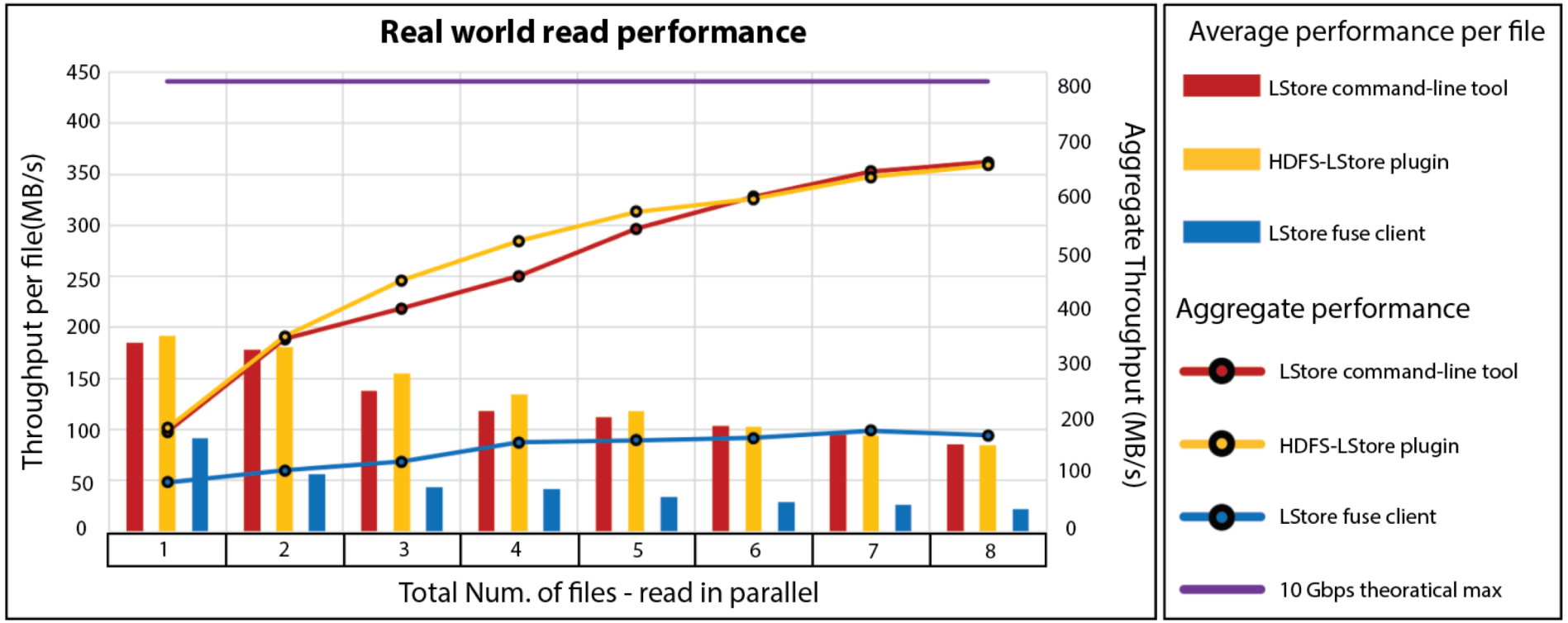
Validation for HDFS & LStore integration - read performance

- Ideal scenario – read to single client
  - Each file: 10GB
  - 10 Gbps LAN connection
  - Single user dedicate environment
  - HDFS / LStore command line buffer size: 80MB



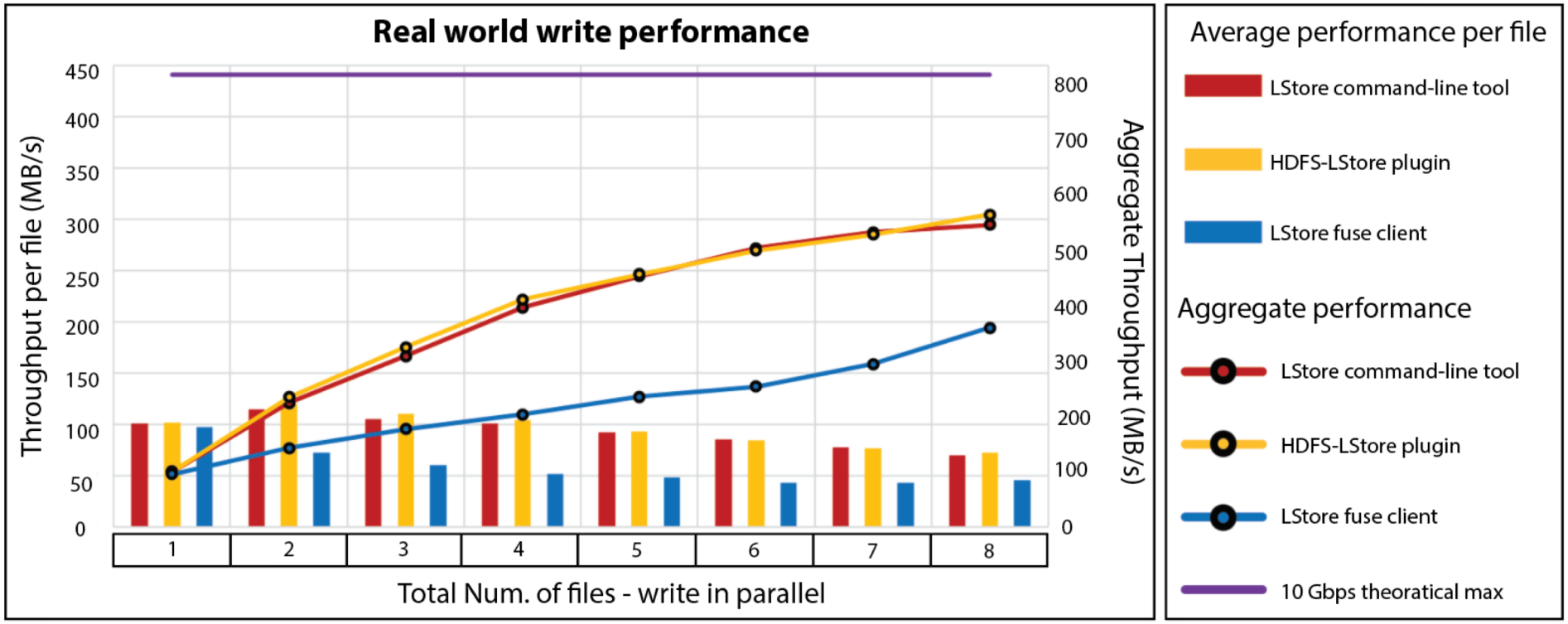
Validation for HDFS & LStore integration - read performance cont.

- Real world scenario – read to single client
  - ACCRE's inside LStore
  - 10 Gbps LAN connection
  - Multi-user share environment (5000+ active jobs)



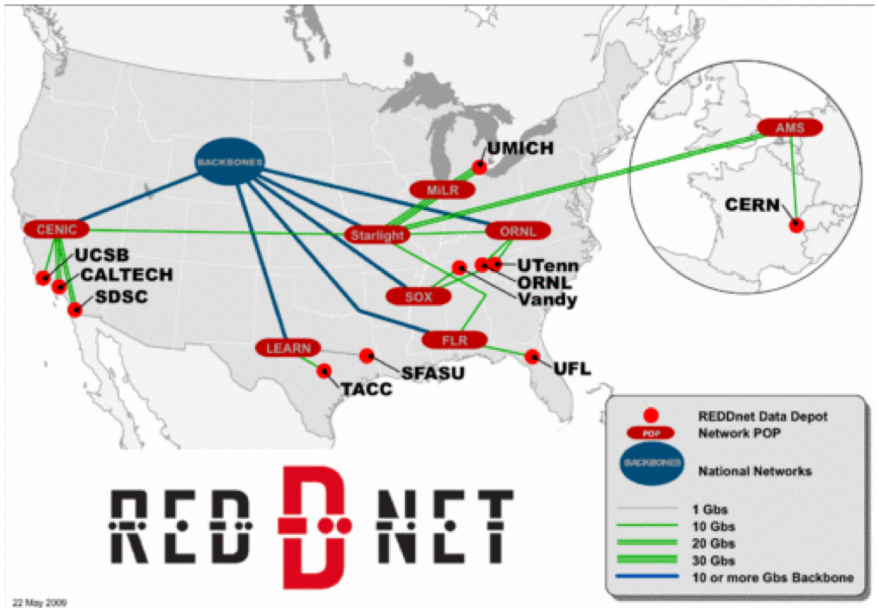
Validation for HDFS & LStore integration - write performance

- Real world scenario – write to single client
  - ACCRE's inside LStore
  - 10 Gbps LAN connection
  - Multi-user share environment (5000+ active jobs)



Conclusion and future work

- LN provides a generic block level storage abstraction
- LStore provides a highly scalable, fault tolerant file system via ExNode
- LStore's new feature - HDFS
  - Data loading efficiency in HPC multi-user environment.
- Performance over WAN
  - REDDNET
- HBase, small file



Question?