# Convergence of BigData Infrastructure for HPC and Internet Services

Garth Gibson
Carnegie Mellon University

http://www.istc-cc.cmu.edu/

# BigData meets Extreme Scale

The largest scale systems come in two flavors:

- Internet Services → Public Clouds
- High Performance Computing → Private Clouds

Have similar scales: 1000s to 10000s nodes

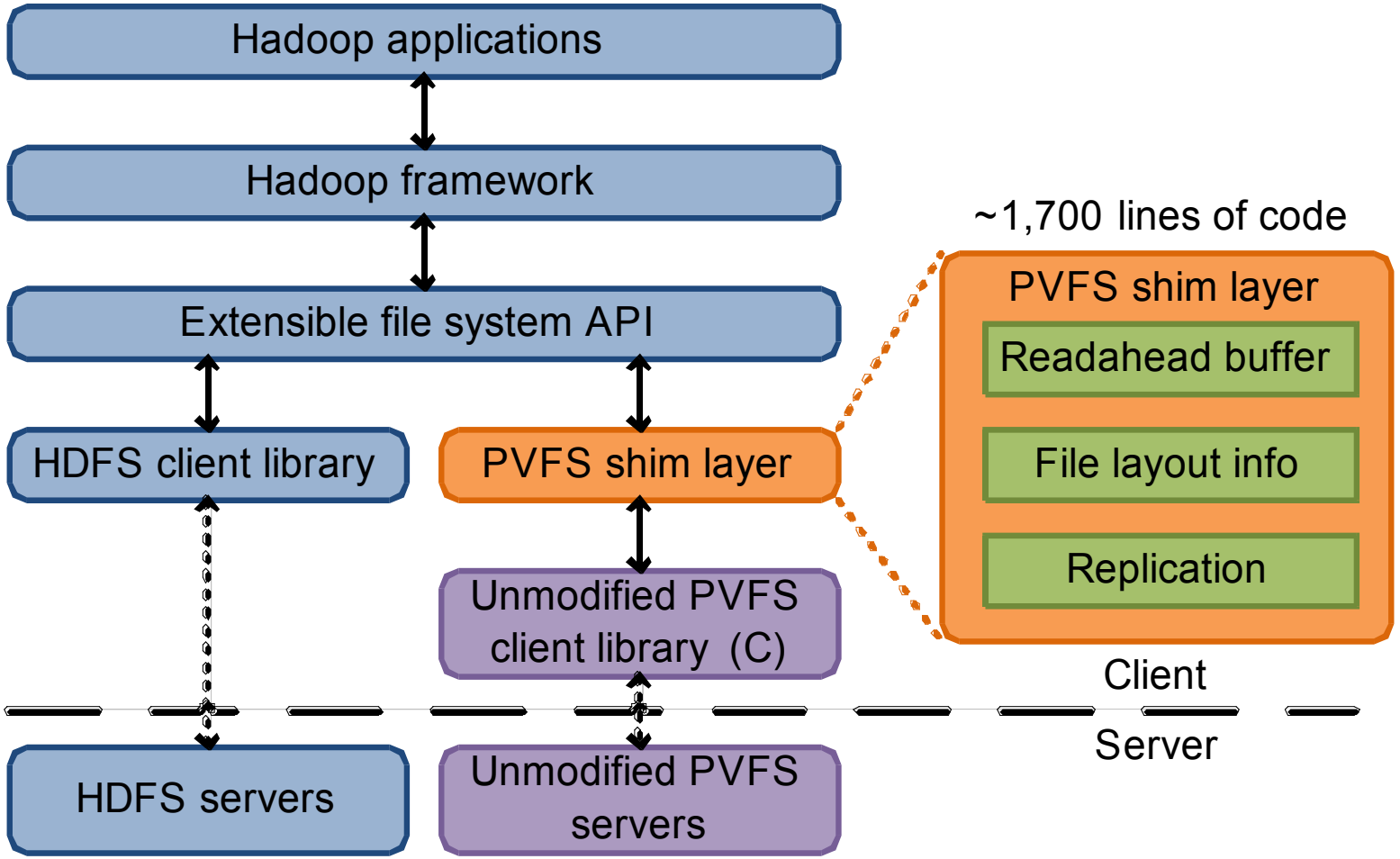
- But software was independently developed
- NIH is strong in both, but commonality is too

Lets look at issues in the storage stack

- Data
- Metadata

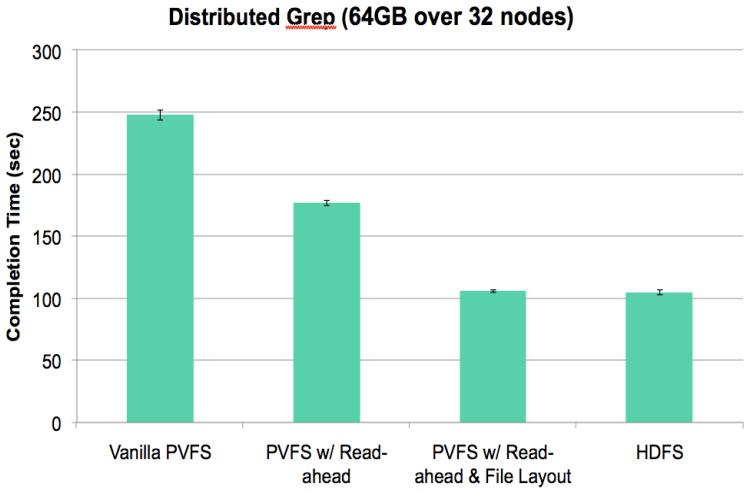
# Data

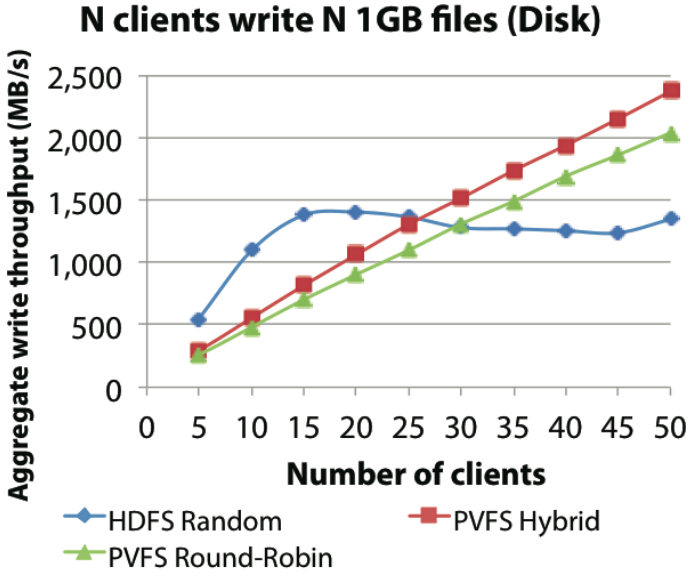
# Experiment: Replace CloudFS with HPC FS



# Once Tuned, Performance Equivalent

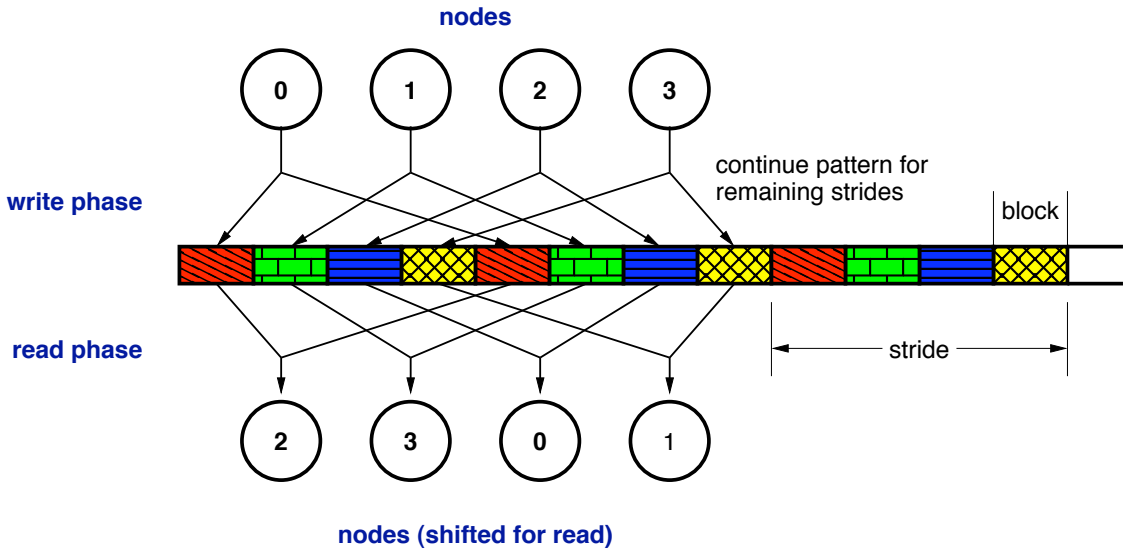
- HDFS: 64MB readahead
  - Tune PVFS to 4MB reads
- Expose PVFS layout
  - Most PFS have FS/Object layering exposed to client
- A few oddities:
  - HDFS on XFS does synch. chunk creates, while PVFS appends
  - PVFS readahead at storage server only, still slower
    - But other PFS do better



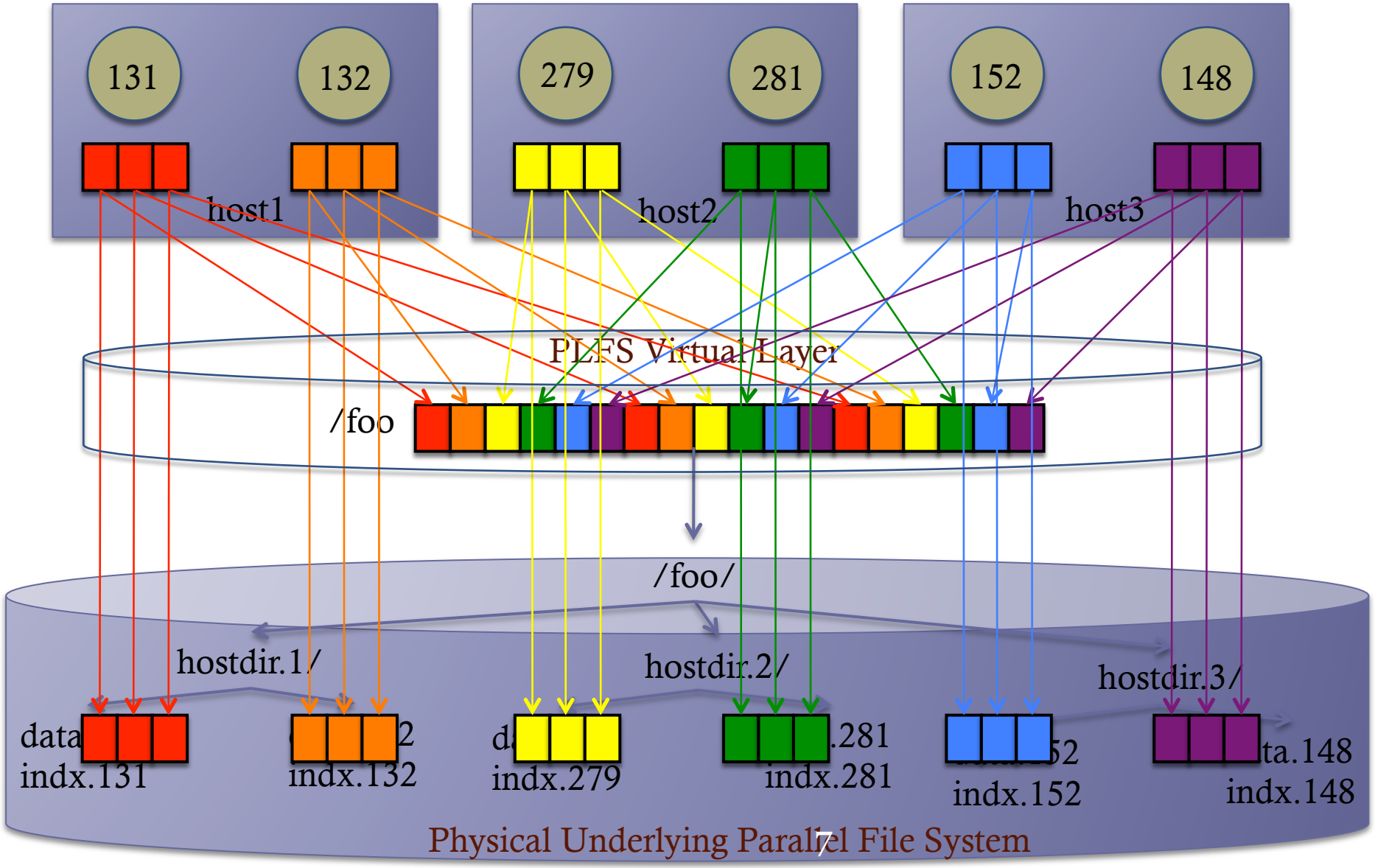



# Flip it around: HPC apps on Cloud Store

- HPC app wants to use HDFS formatted cluster
  - Uses single checkpoint file for fault tolerance
  - Adaptive mesh refinement: small strided writes
  - O(100,000) concurrent writes of 47,001 B each
- HDFS files are single writer, immutable ?
  - Apply indirection: HDFS as object store for PLFS lib

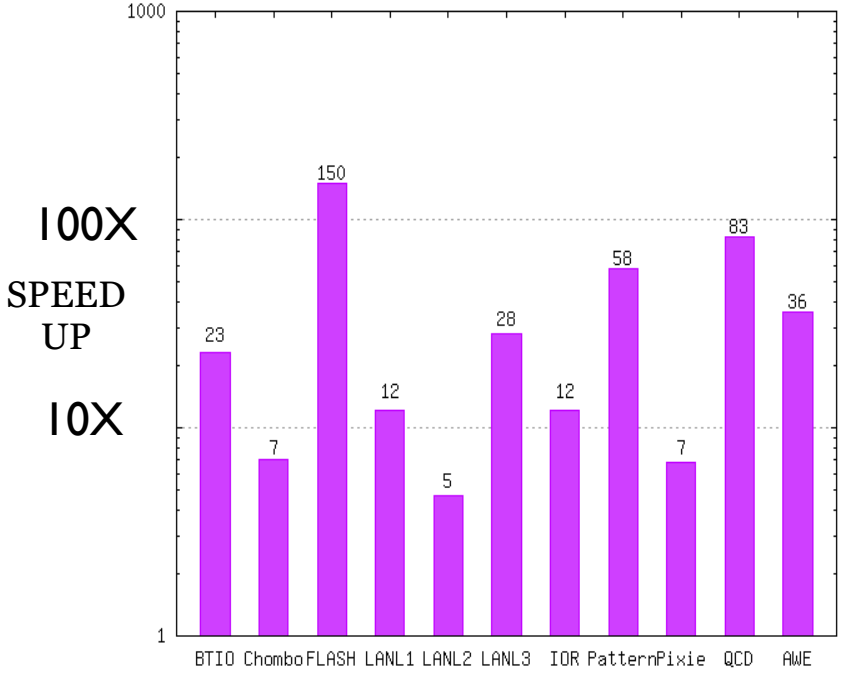


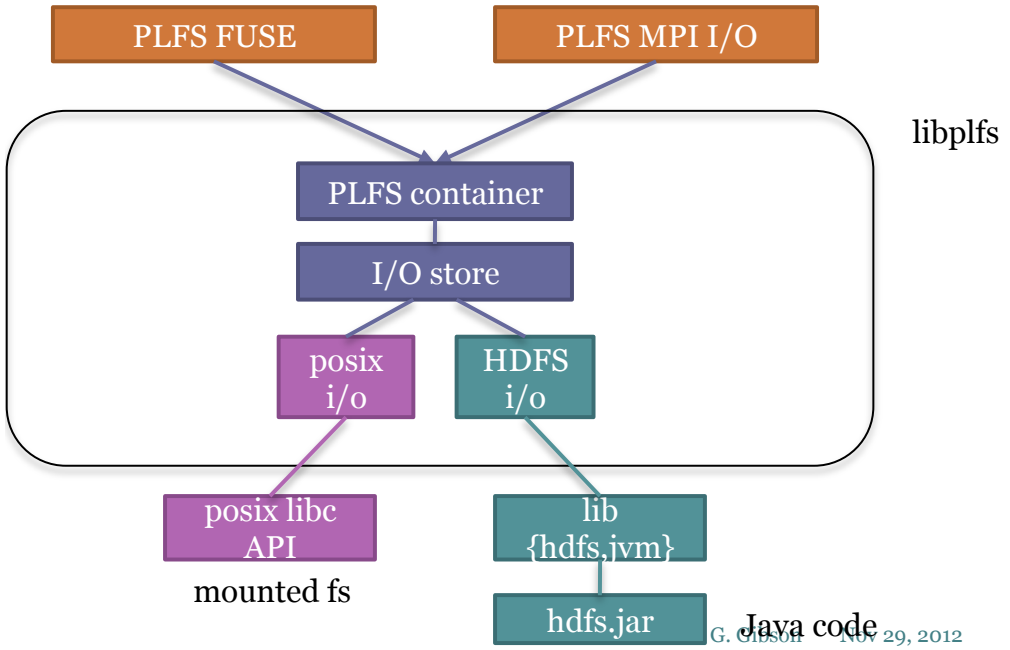
# PLFS Converts N-1 to N-N



# PLFS decouples concurrency & wins big

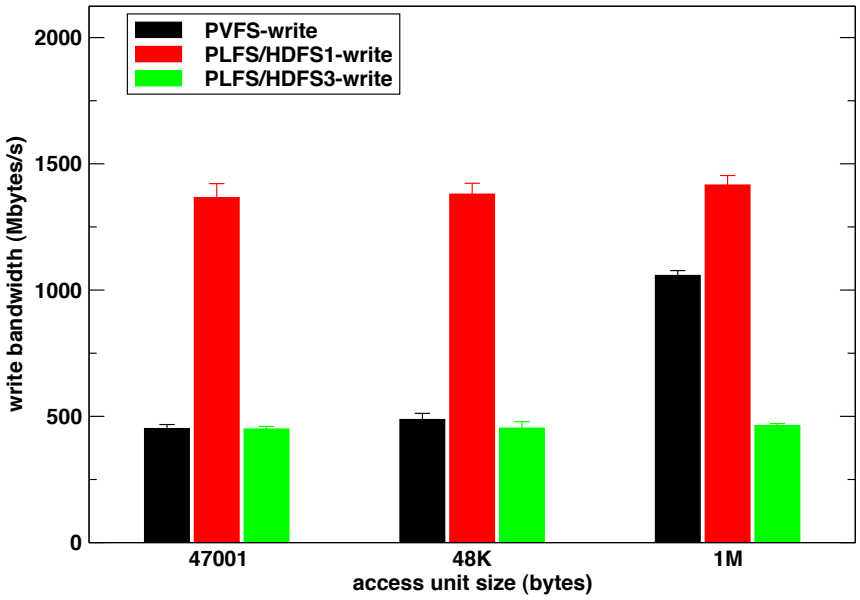
- HPC apps on HPC HW and HPC PFS: 10-100X
  - Gone into production at Los Alamos
  - Included in DOE Exascale SW stack
  - Supported by LANL, EMC, CMU
- Exploit decoupling to enable HDFS as substrate

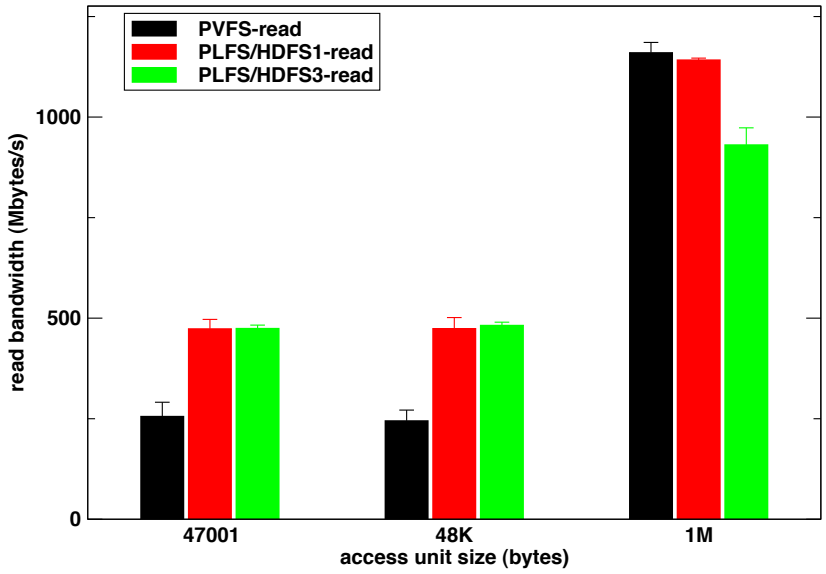


# PLFS/HDFS vs PVFS for HPC N-1 Apps

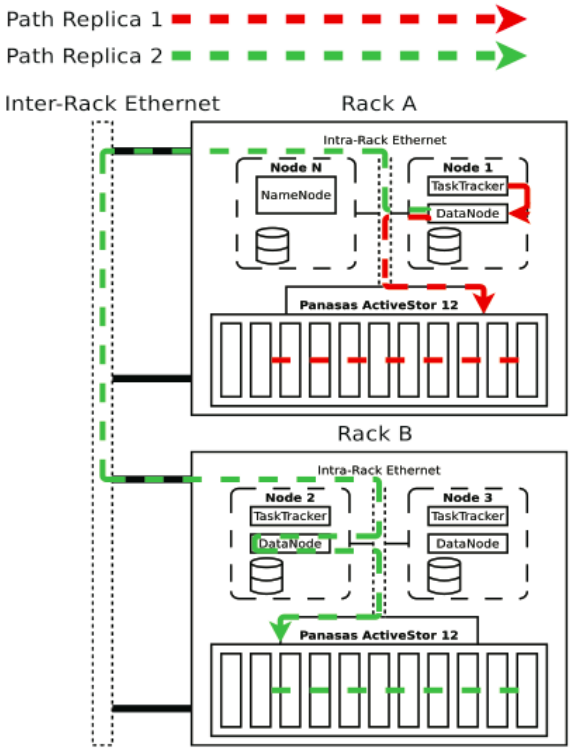
- Develop PLFS IOStore backend & API for HDFS
- Compare to PVFS on 64 node HW (no RAID)
  - Use 47001B, 48KB, 1MB strided writes
- It works, has batching benefits, less perfect balance

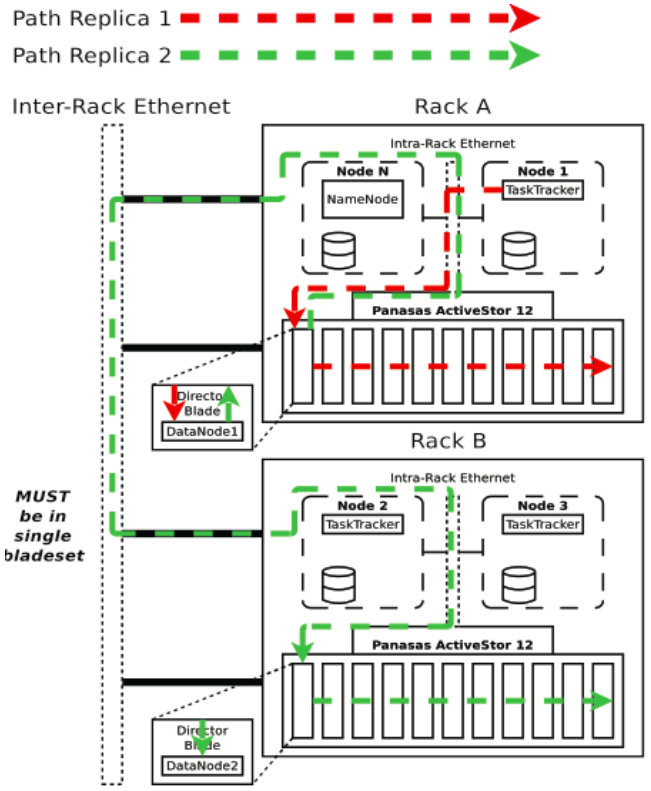


# How to layer HDFS on external NAS?

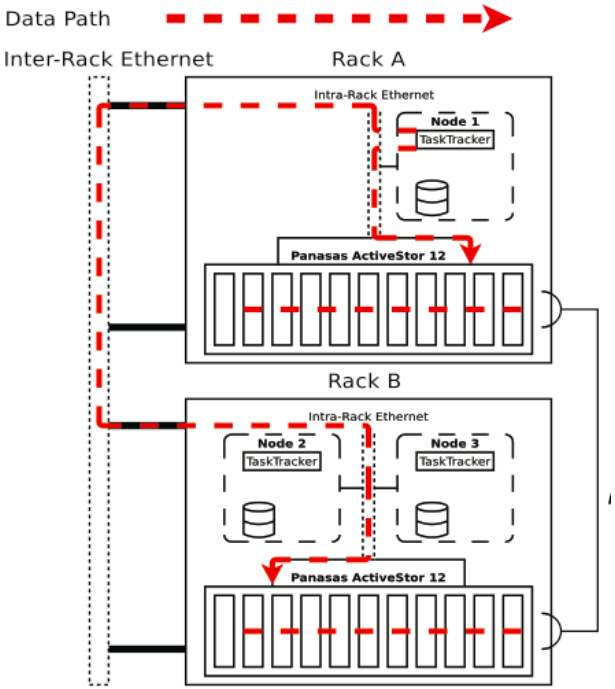
- External RAID as object server (NetApp) [left]
  - SAN-based local disk model
- HDFS as a NAS wire protocol (Isilon) [right]
  - HDFS implementation entirely in NAS box

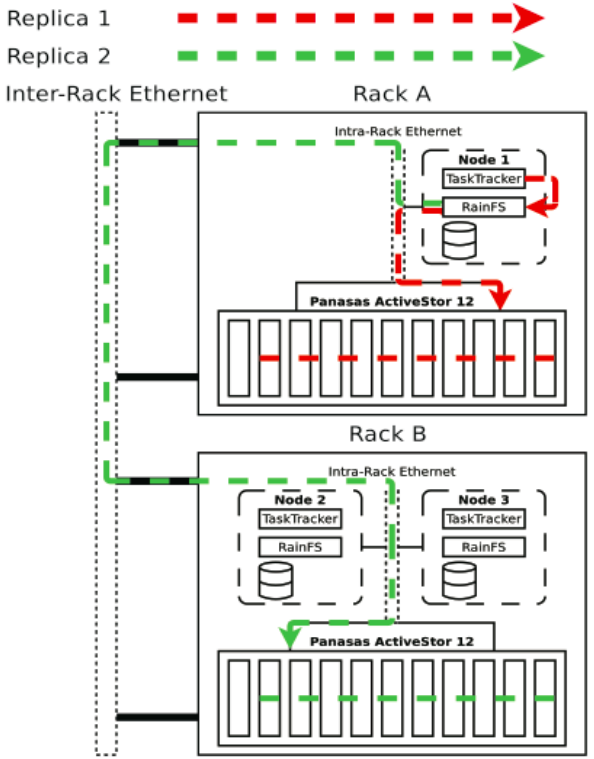


# Simplify away one file system (HDFS)

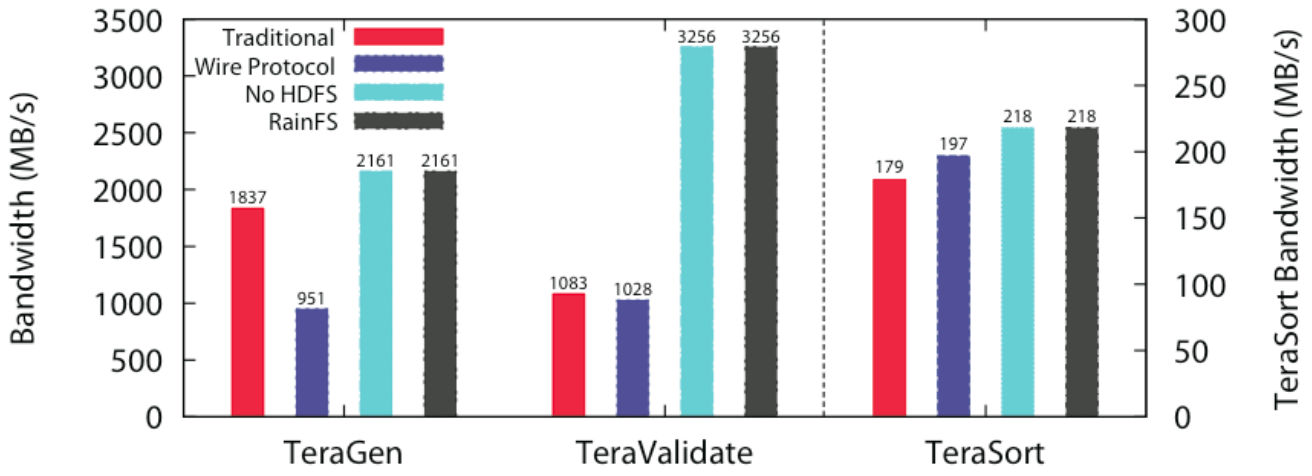
- Replace HDFS w/ PFS (mount PFS path)[left]
  - Scaling of PFS mount limits scaling
- Client library (RainFS) for rep control [right]
  - Client-based synchronous mirroring



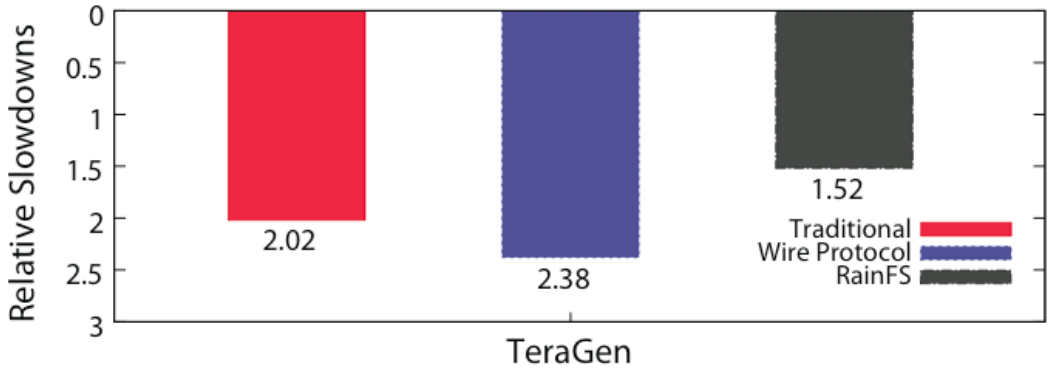



# HDFS DataNode code in the way

- Comparing – 50 VMs/nodes, 1GE, 5 PanFS arrays (20 disks, 20GE, RAID5), Terasort
  - 1 copy

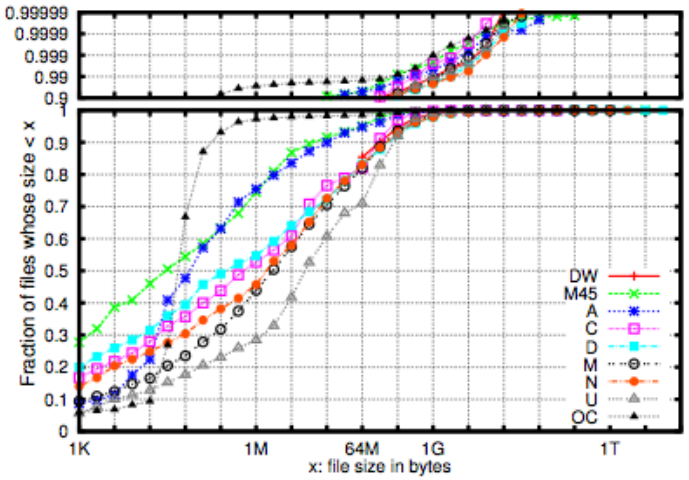



  - 2 copies

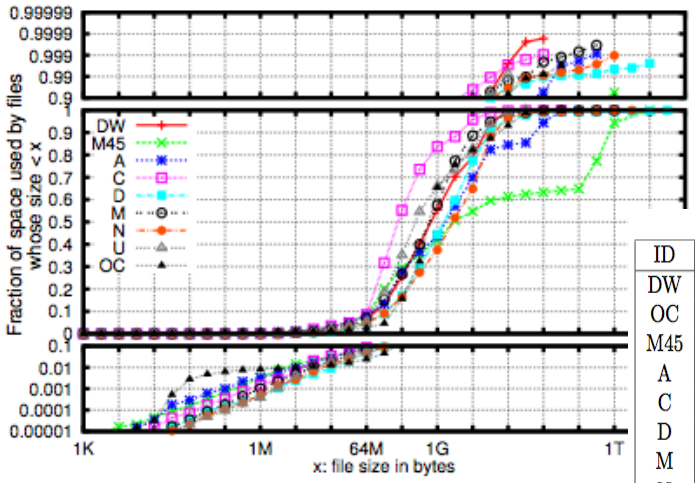


# Metadata

# Metadata in the BigData World



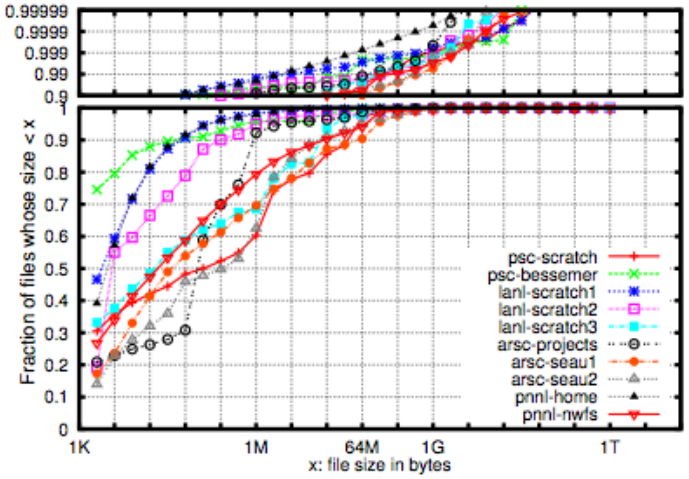

(a) DISC file systems: fraction of files $\leq x$ bytes



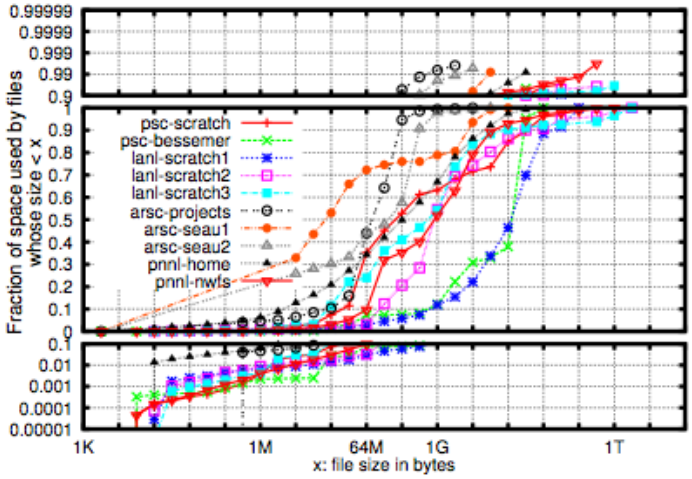

(b) DISC file systems: fraction of storage used by files $\leq x$ bytes

File Distribution Traces

| ID | Description | # nodes | Raw Capacity |
|---|---|---|---|
| DW | unknown | 2000 | 21 PB |
| OC | Research | 64 | 0.25 PB |
| M45 | Research | 400 | 1.5 PB |
| A | Sandbox | 1800 | 3.8 PB |
| C | Research | 800 | 3.5 PB |
| D | Production | 3000 | 6.5 PB |
| M | Research | 2000 | 6.3 PB |
| N | Research | 3500 | 10.6 PB |
| O | Production | 1000 | 7.5 PB |
| U | Production | 1700 | 13.0 PB |



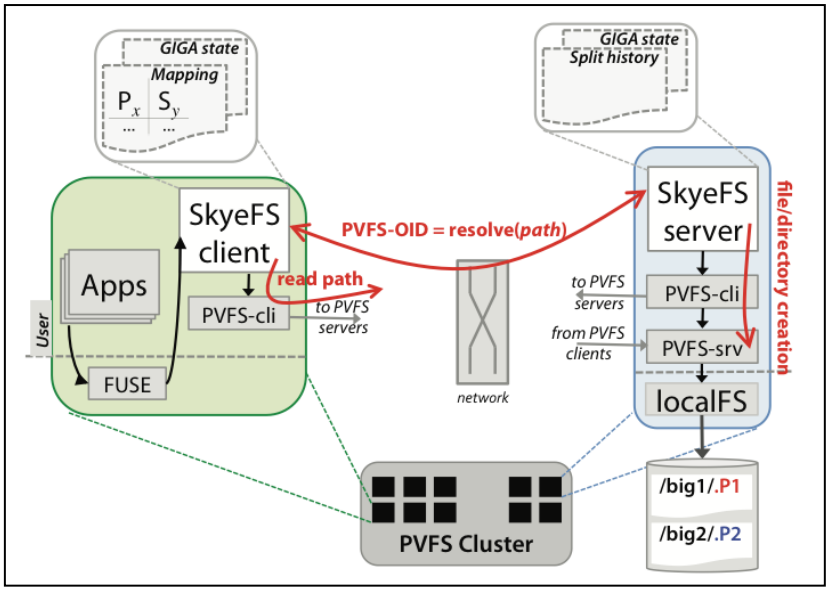

(c) HPC file systems: fraction of $\leq x$ bytes

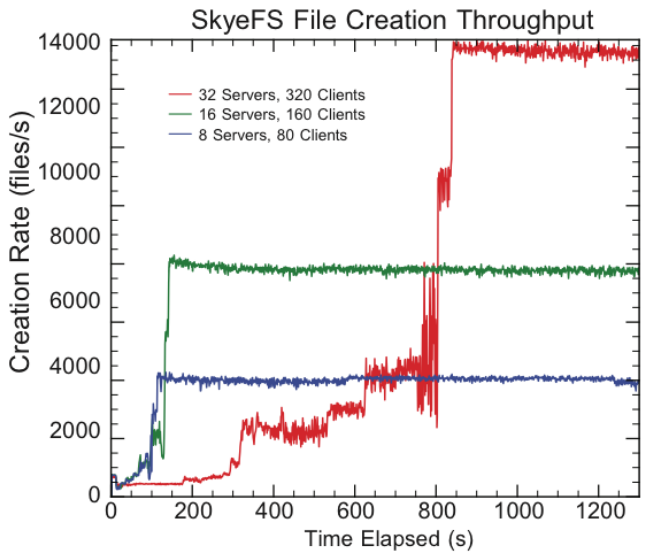



(d) HPC file systems: fraction of storage used by files $\leq x$ bytes

# Scalable Metadata Rare in HPC

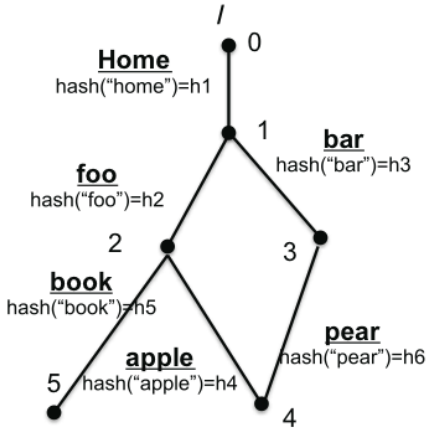
- Closest metadata scalable PFS is PVFS (OrangeFS)
  - Distribution unit is entire directory
  - Apply Giga+ directory rehashing to PVFS: SkyFS
    - Binary split directories, EC cache of server address
    - Depends on PVFS atomic rename across servers



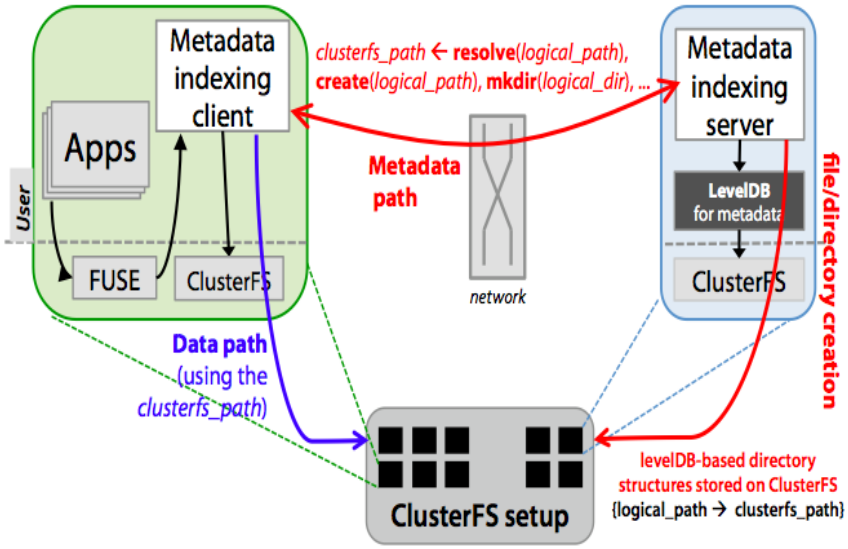
# Reimplement to use Global Object Name

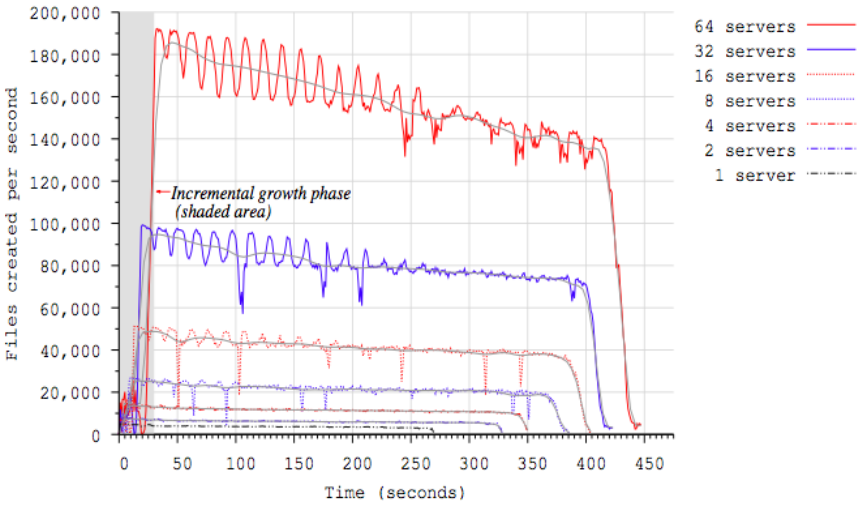
- Don't use PFS dirs
- Merge dirs & inodes into LevelDB table
  - Scalable creation rate for 0B files




Lexicographic order

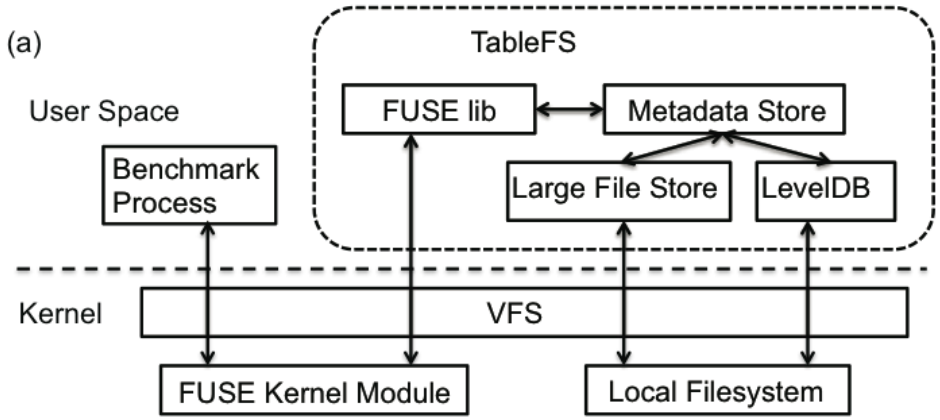
| Key | Value |
|---|---|
| <0,h1> | 1, "home", struct *stat* |
| <1,h2> | 2, "foo", struct *stat* |
| <1,h3> | 3, "bar", struct *stat* |
| <2,h4> | 4, "apple", hard link |
| <2,h5> | 5, "book", struct *stat*, inline small file (<4KB) |
| <3,h6> | 4, "pear", hard link |
| <4,null> | 4, struct *stat*, large file pointer (> 4KB) |

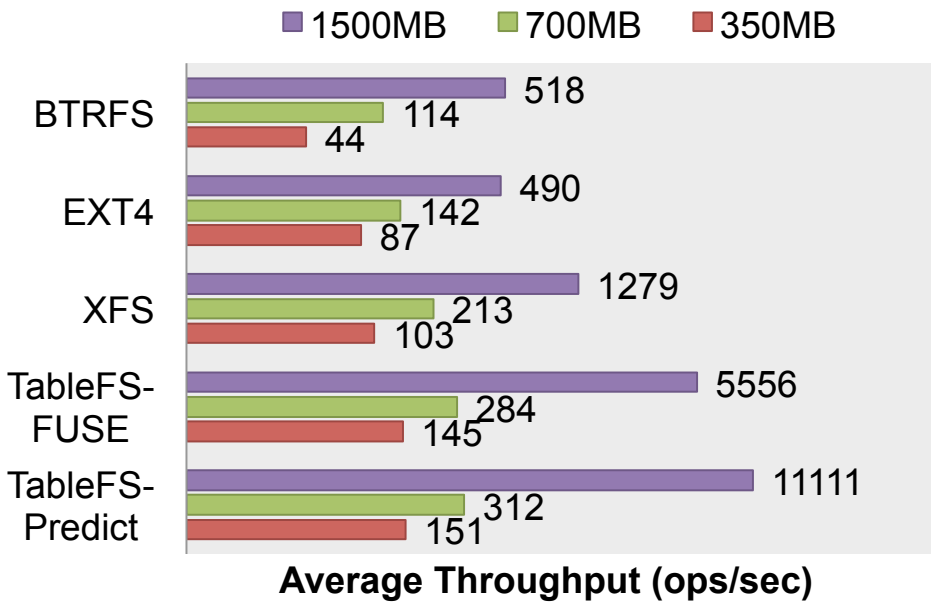


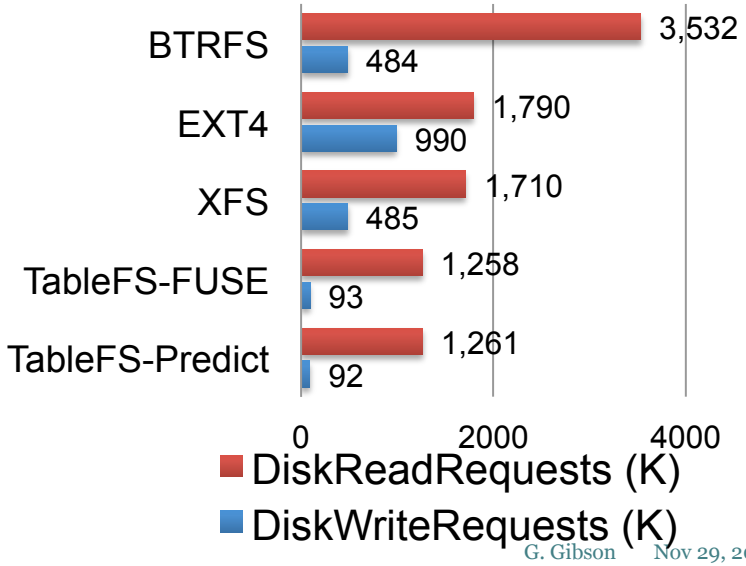



# Merged Dirs for Local FS: TableFS

- Surprisingly good benefits from packing
- Beats best localFS by up to 10X
  - Rand update inodes

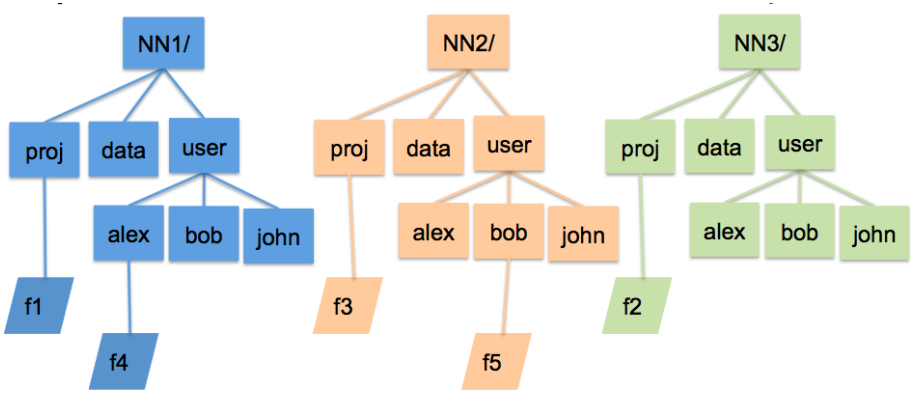


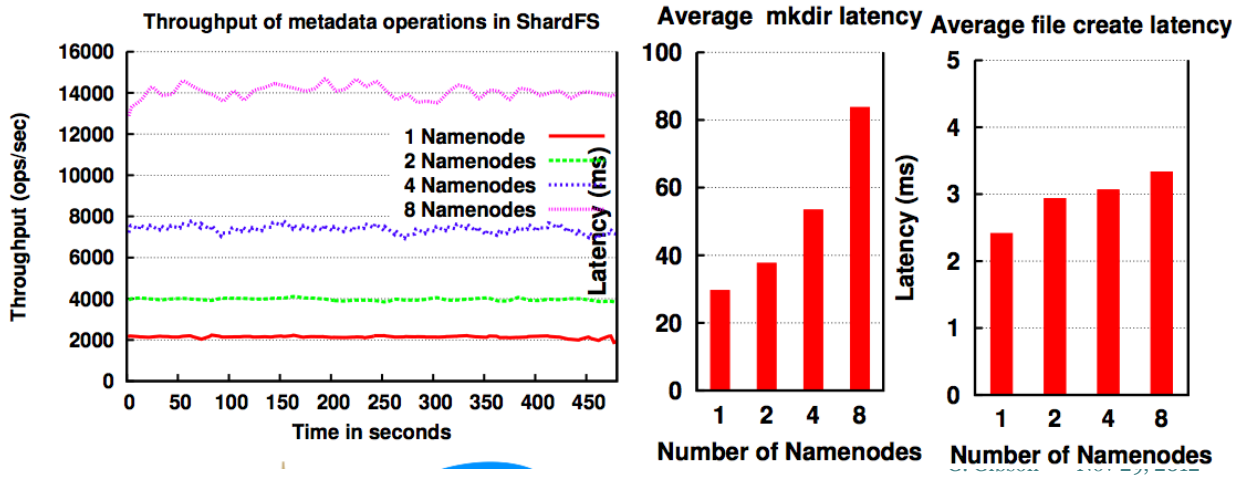



# What about CloudFS metadata

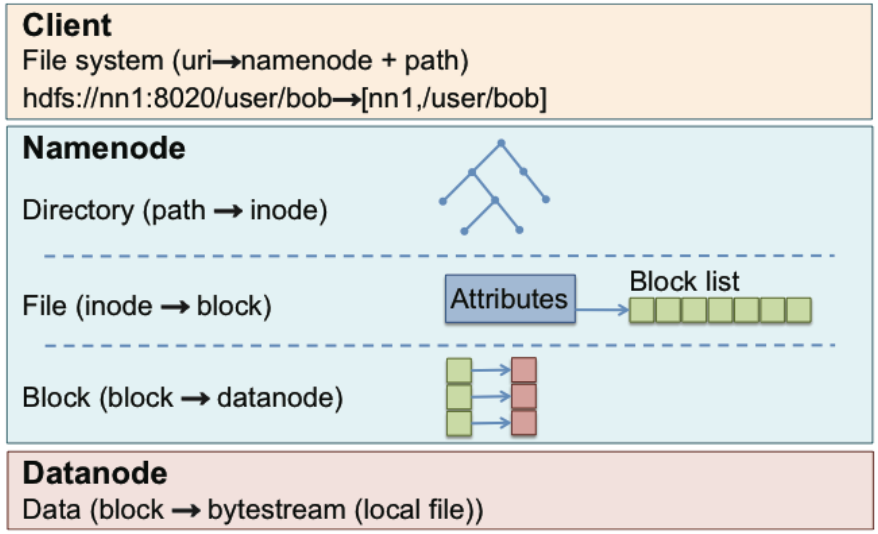
- Start w/ client lib: ShardFS
  - Federated DataNodes
  - Replicated namespace
  - Sharded files
  - Zookeeper for (slow) namespace transaction
  - Optimistic file ops, on failure try blocking on locks

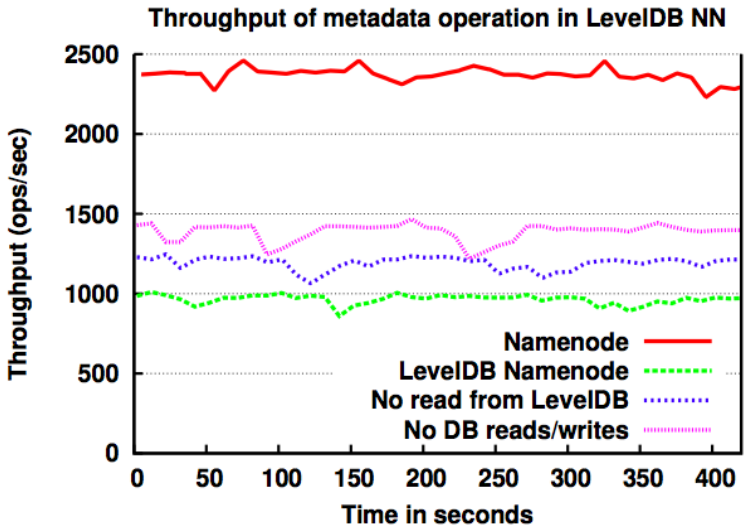


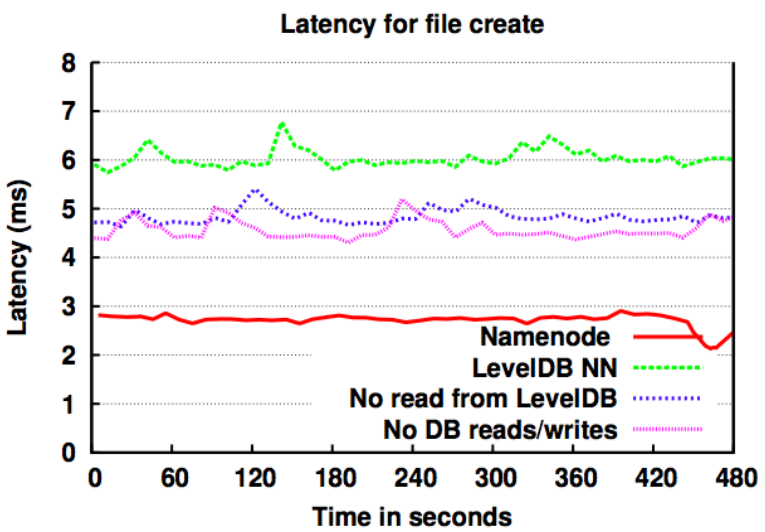



# First step to distributed table for HDFS

- First represent Namenode metadata in a table (LevelDB)
- Remove limit on # files in NN
- But NN is clean/fast
  - Code path to generic cache to LevelDB hurting



# Next steps for Scalable HDFS

- LevelDB is local out-of-core table
  - Move to global table: Hbase, etc
  - Or, stay with LevelDB, implement splitting in NN
    - Like Giga+TableFS
  - How much benefit in app specific load balancing?
    - Giga+ trick is to only split if load balancing needed
    - Hbase and friends split on growth at all times
- Fault tolerance for MDS in PFS is simple failover
  - Primary manages backup-of-logs
  - Quorum consensus replicated db manages roles
  - Move "all" of MDS processing into replicated db?
    - Primary benefit would be fast failover

# Student and Staff Partners

- Students/staff on this project
  - Lin Xiao, Kai Ren, Kartik Kulkarni, Chuck Cranor, Swapnil Patil, Carnegie Mellon
  - Ellis Wilson, Penn State
- Other advised ISTC-CC students
  - Jiri Simsa, Ben Blum, Jin Kyu Kim, Jinliang Wei, Samantha Gottlieb, Carnegie Mellon
- Organizational Partners
  - IRHPIT, Los Alamos National Lab, (PDSI, DOE)
  - Qloud, CMU Qatar
  - NSA Research