Crossing the Chasm: Sneaking a parallel file system into Hadoop

Wittawat Tantisiriroj

Swapnil Patil, Garth Gibson

PARALLEL DATA LABORATORY

Carnegie Mellon University

In this work …

- Compare and contrast large storage system architectures
  - Internet services
  - High performance computing

- Can we use a parallel file system for Internet service applications?
  - Hadoop, an Internet service software stack
  - HDFS, an Internet service file system for Hadoop
  - PVFS, a parallel file system

Today's Internet services

- Applications are becoming data-intensive
  - Large input data set (e.g. the entire web)
  - Distributed, parallel application execution
- Distributed file system is a key component
  - Define new semantics for anticipated workloads
    - Atomic append in Google FS
    - Write-once in HDFS
  - Commodity hardware and network
    - Handle failures through replication

The HPC world

- Equally large applications
  - Large input data set (e.g. astronomy data)
  - Parallel execution on large clusters

- Use parallel file systems for scalable I/O
  - e.g. IBM's GPFS, Sun's Lustre FS, PanFS, and Parallel Virtual File System (PVFS)

Why use parallel file systems?

- Handle a wide variety of workloads
  - High concurrent reads and writes
  - Small file support, scalable metadata
- Offer performance vs. reliability tradeoff
  - RAID-5 (e.g., PanFS)
  - Mirroring
  - Failover (e.g., LustreFS)
- Standard Unix FS interface & POSIX semantics
  - pNFS standard (NFS v4.1)

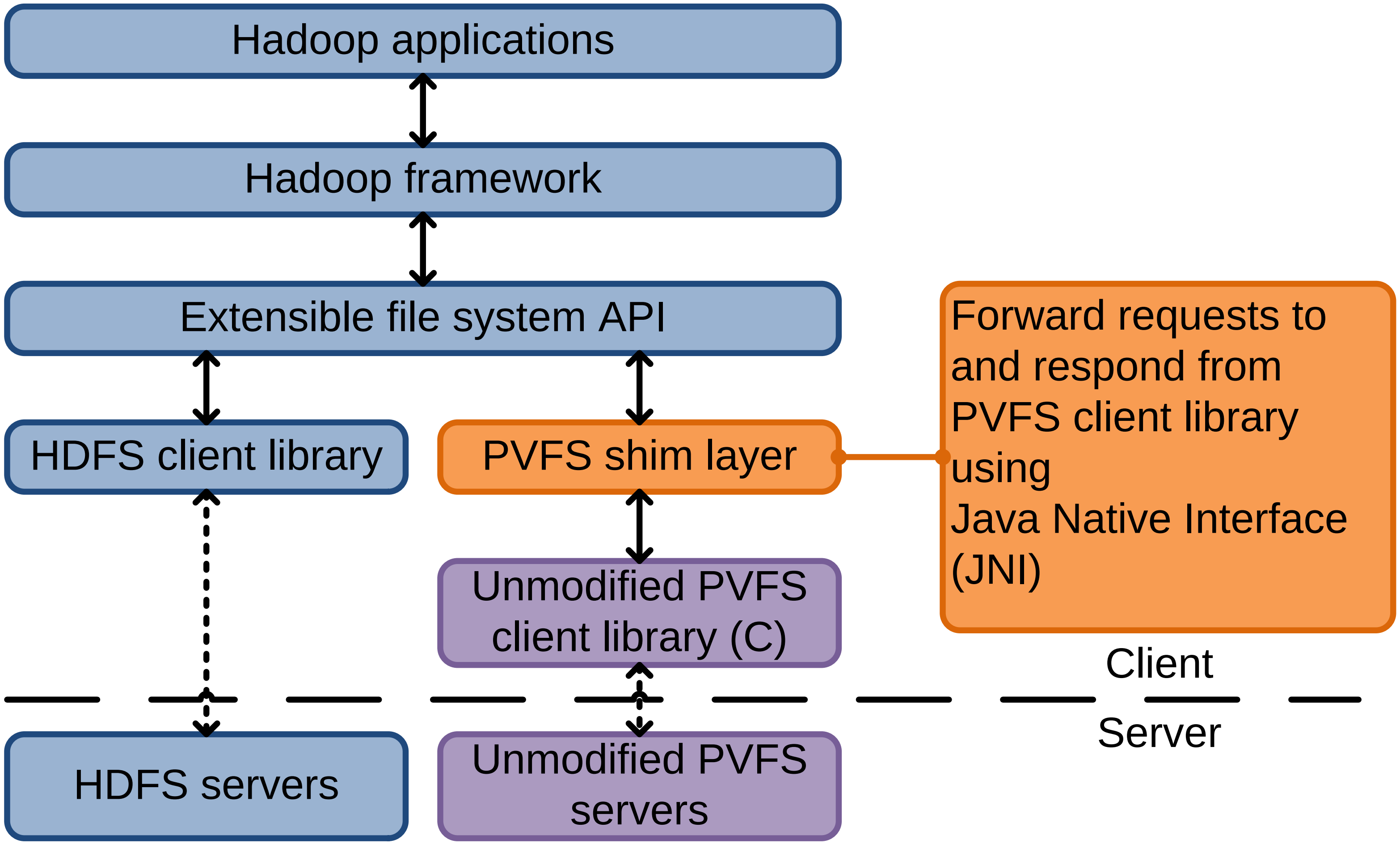
Outline

- ➢ A basic shim layer & preliminary evaluation
- Three add-on features in a shim layer
- Evaluation

HDFS & PVFS: high level design

- Meta-data servers
  - Store all file system metadata
  - Handle all metadata operations
- Data servers
  - Store actual file system data
  - Handle all read and write operations
- Files are divided into chunks
  - Chunks of a file are distributed across servers

PVFS shim layer under Hadoop

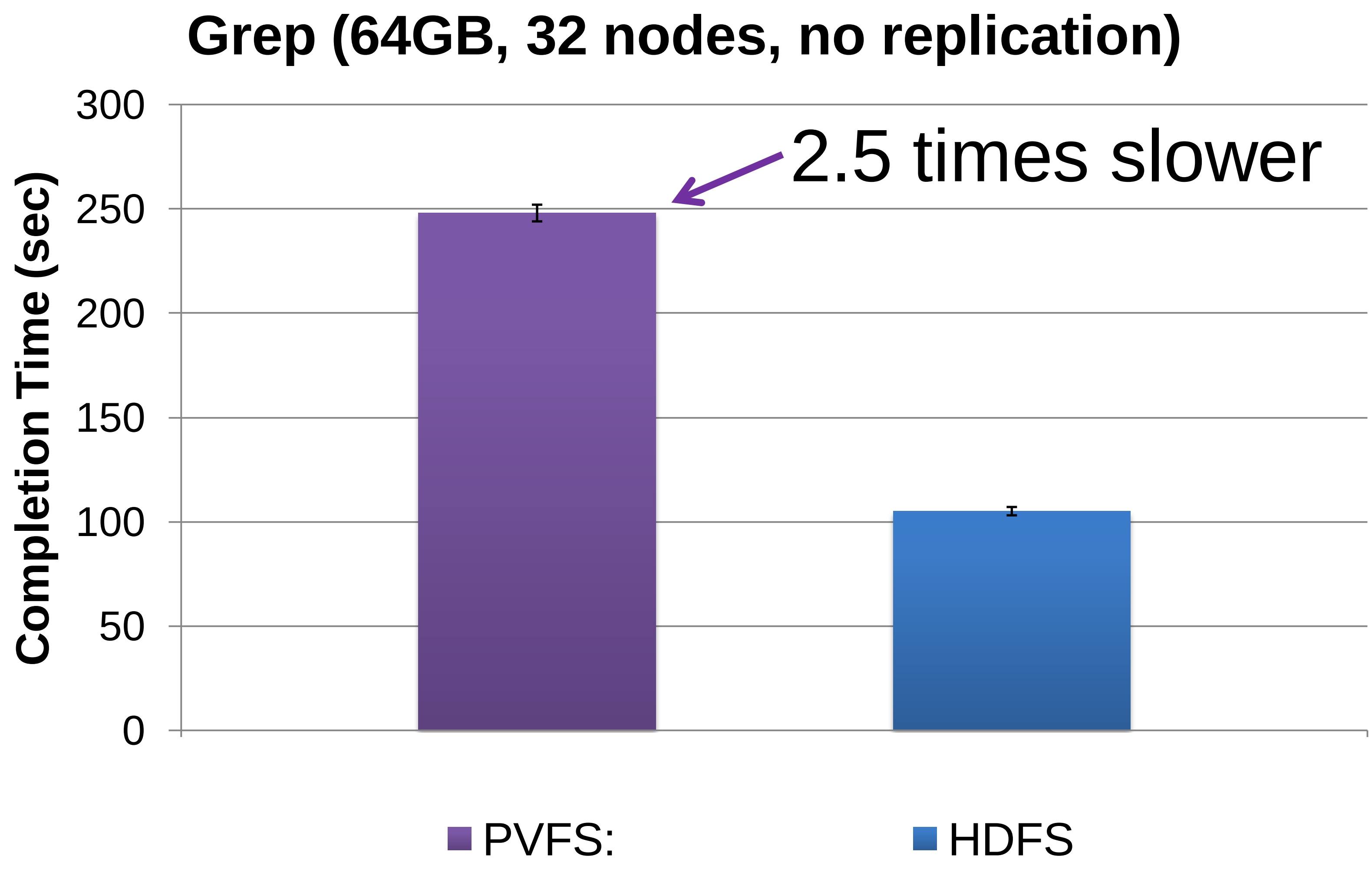


Preliminary Evaluation

- Text search ("grep")
  - common workloads in Internet service applications

- Search for a rare pattern in 100-byte records
  - 64GB data set
  - 32 nodes
  - Each node serves as storage and compute nodes

Vanilla PVFS is disappointing …



Outline

- A basic shim layer & preliminary evaluation
- ➢ Three add-on features in a shim layer
  - ✓ Readahead buffer
  - File layout information
  - Replication
- Evaluation

Carnegie Mellon
Parallel Data Laboratory

Read operation in Hadoop

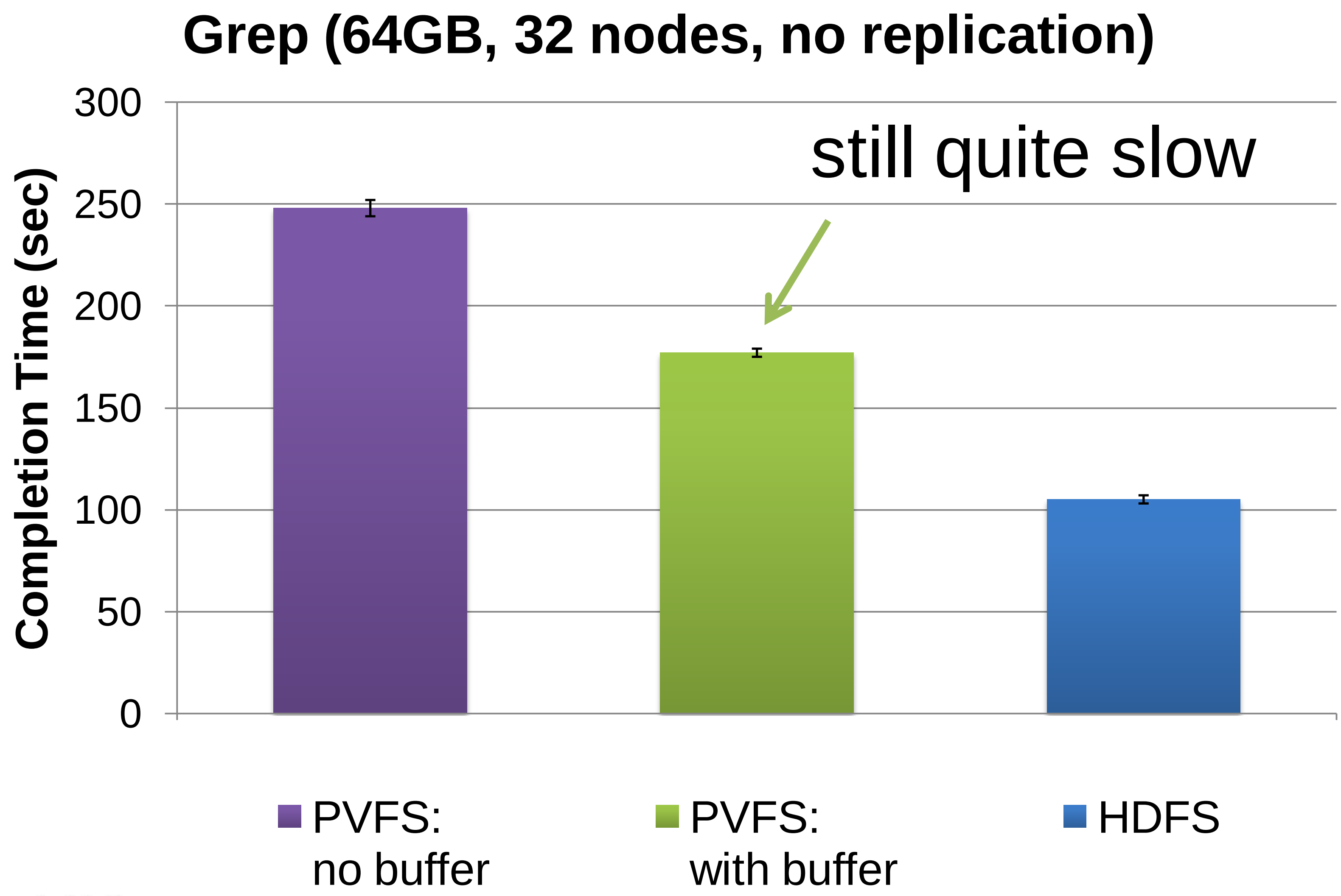
- Typical read workload:
  - Small (less than 128 KB)
  - Sequential through an entire chunk

- HDFS prefetches an entire chunk
  - No cache coherence issue with its write-once semantic

Readahead buffer

- PVFS has no client buffer cache
  - Avoid a cache coherence issue with concurrent writes
- Readahead buffer can be added to PVFS shim layer
  - In Hadoop, a file can become immutable after it is closed
  - No need for cache coherence mechanism

PVFS with 4MB buffer



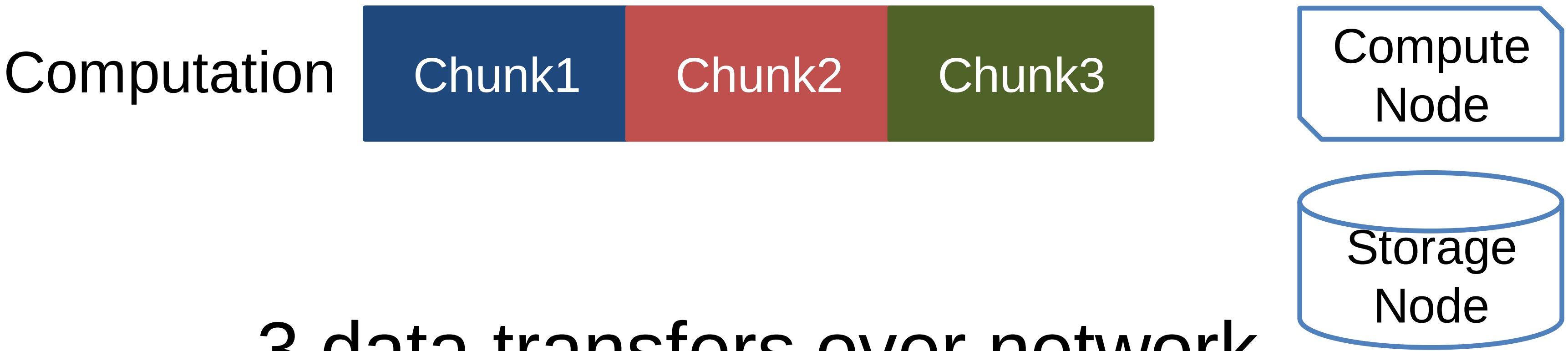
Outline

- A basic shim layer & preliminary evaluation
- ➢ Three add-on features in a shim layer
  - Readahead buffer
  - ✓ File layout information
  - Replication
- Evaluation

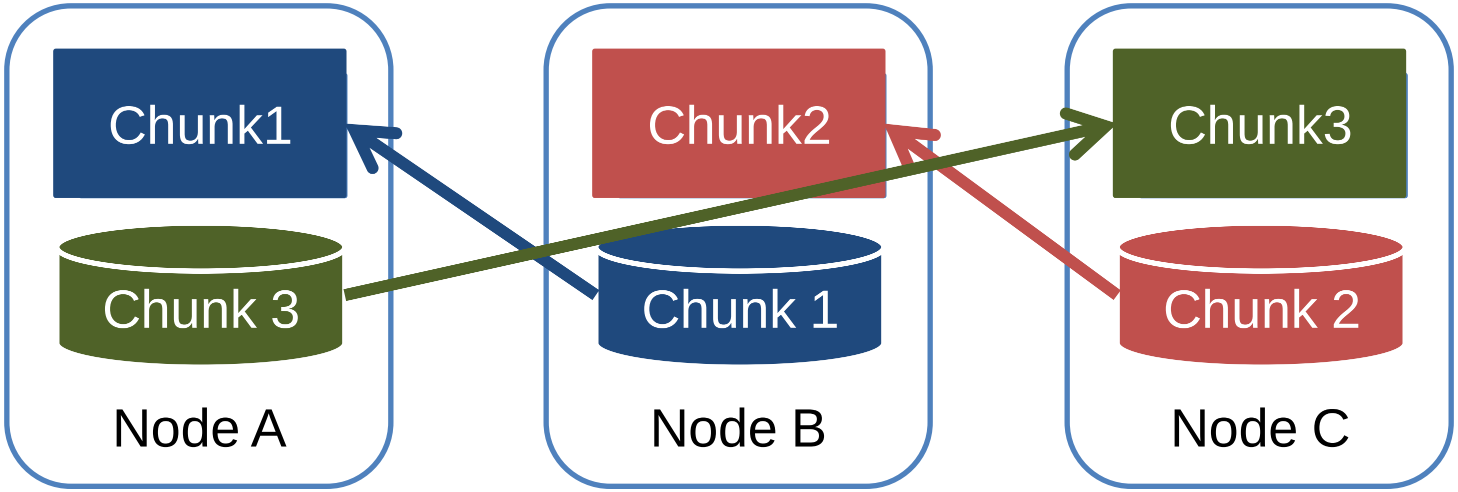
Collocation in Hadoop

- File layout information
  - Describe where chunks are located
- Collocate computation and data
  - Ship computation to where data is located
  - Reduce network traffic

Hadoop without collocation

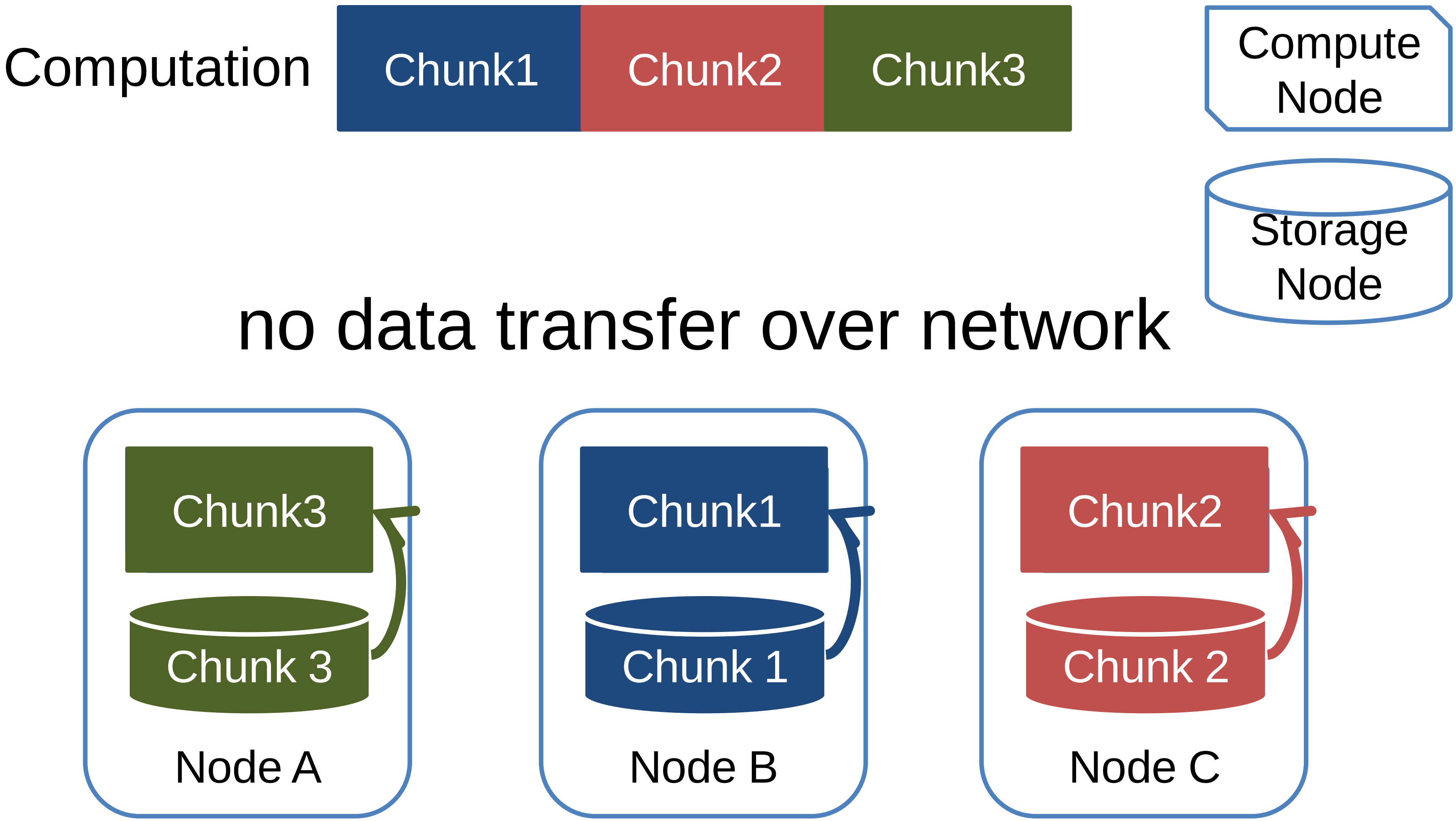


3 data transfers over network



Carnegie Mellon
Parallel Data Laboratory

Hadoop with collocation



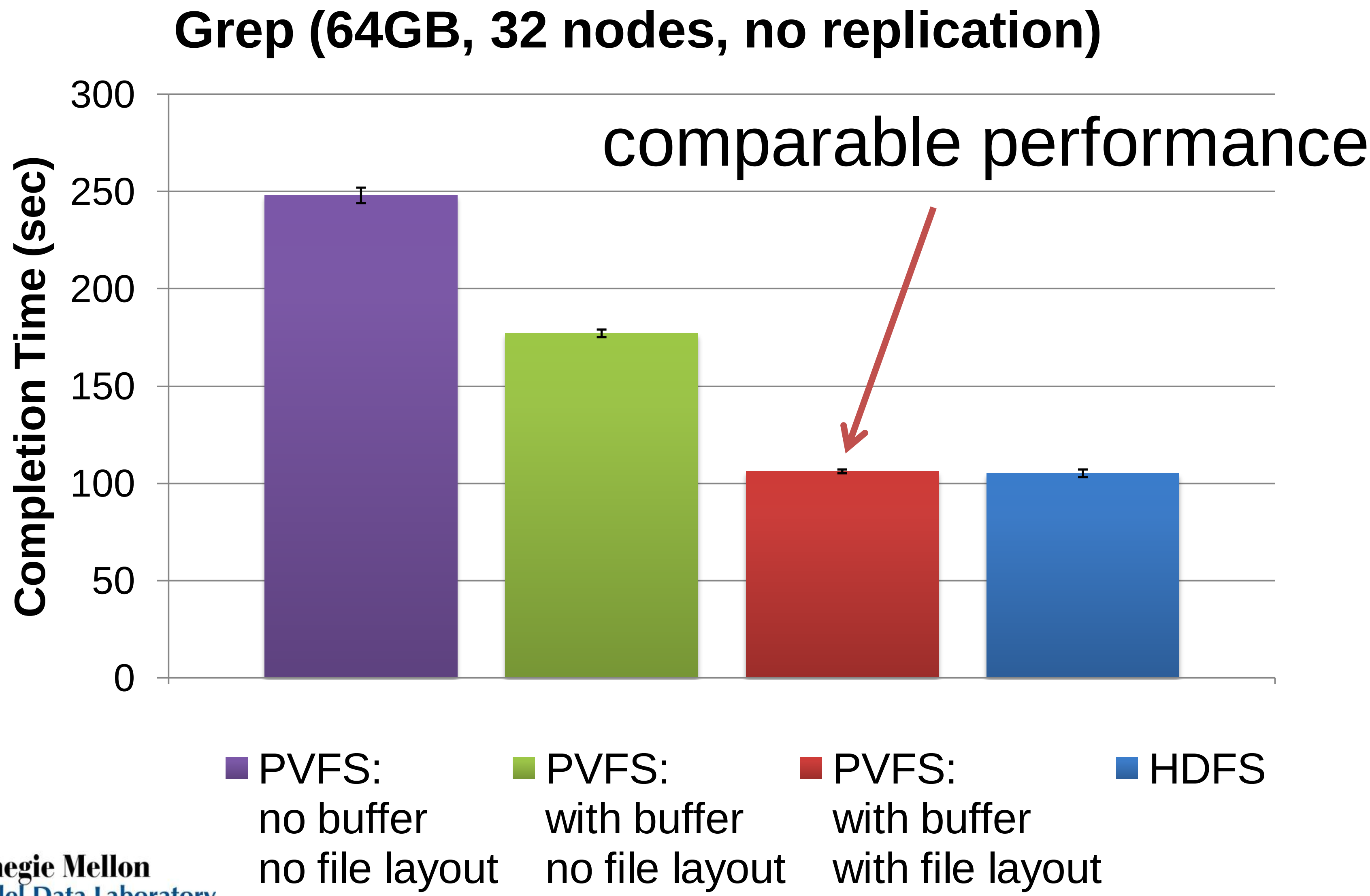
Carnegie Mellon
Parallel Data Laboratory

Expose file layout information

- File layout information in PVFS
  - Stored as extended attributes
  - Different format from Hadoop format

- A shim layer converts file layout information from PVFS format to Hadoop format
  - Enable Hadoop to collocate computation and data

PVFS with file layout information



Outline

- A basic shim layer & preliminary evaluation
- Three add-on features in a shim layer
  - Readahead buffer
  - File layout information
  - ✓ Replication
- Evaluation

Carnegie Mellon
Parallel Data Laboratory

Replication in HDFS

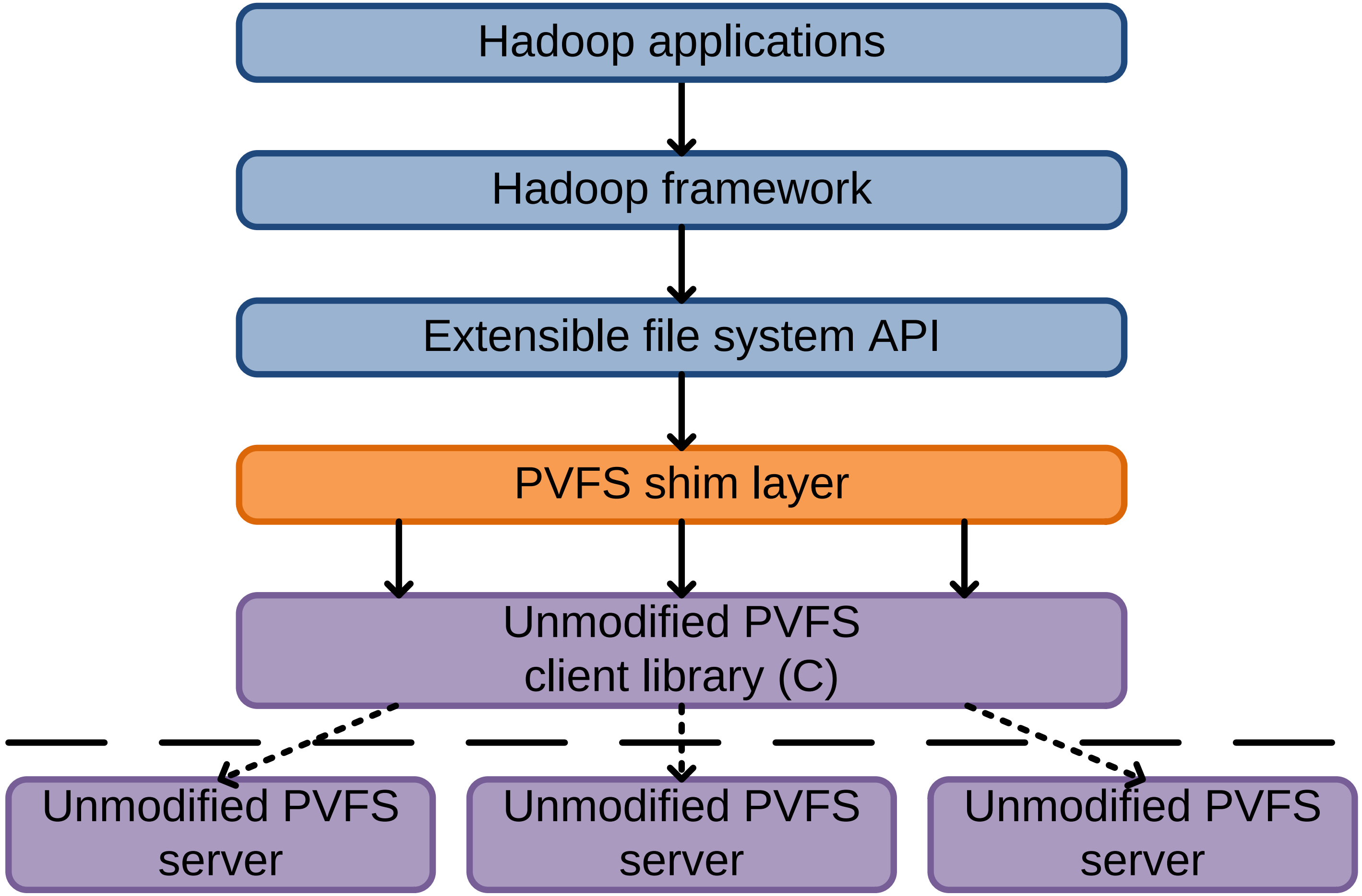
- Rack-awareness replication
  - By default, 3 copies for each file (triplication)
  1. Write to a local storage node
  2. Write to a storage node in the local rack
  3. Write to a storage node in the other rack

Replication in PVFS

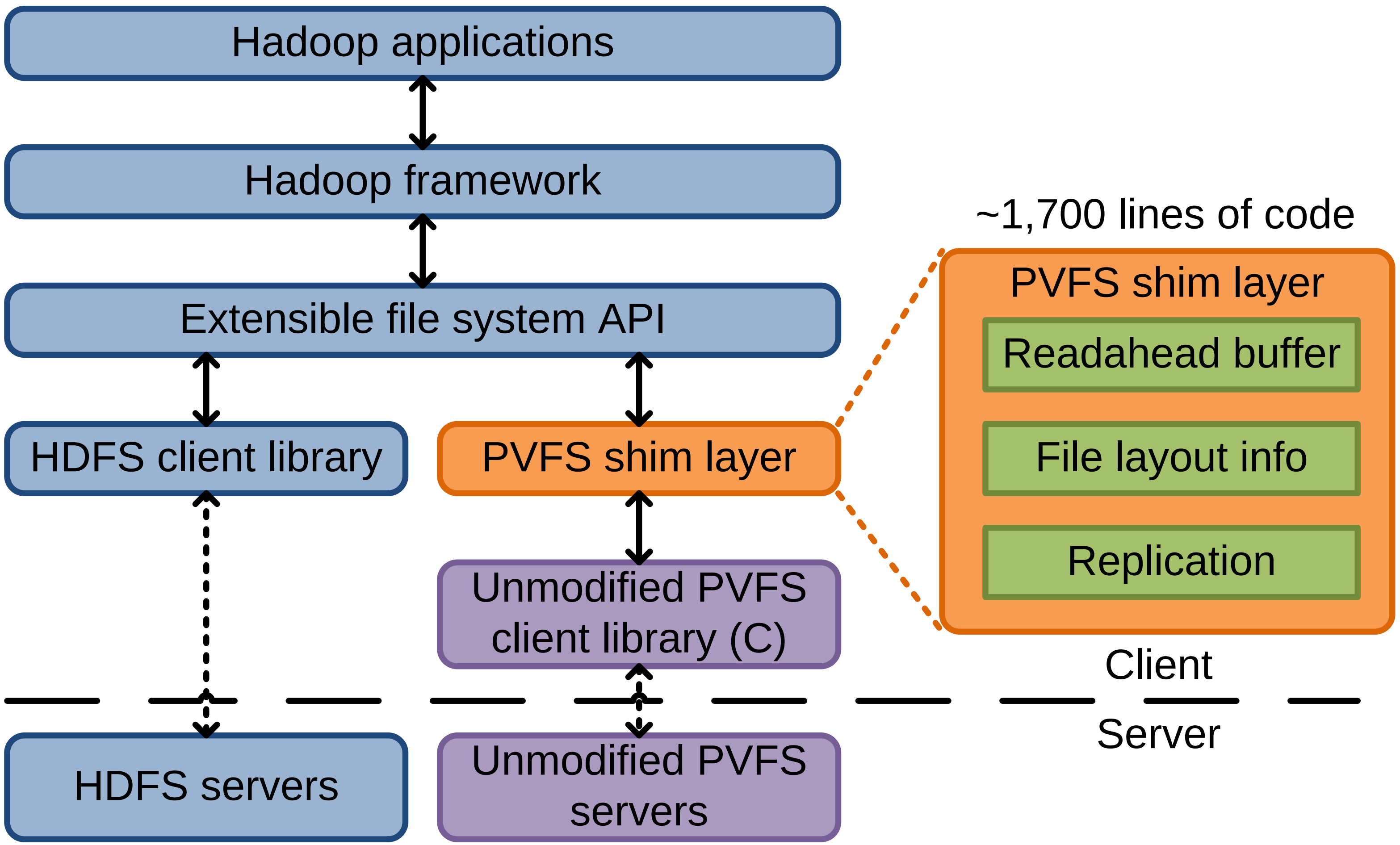
- No replication in the public release of PVFS
- Rely on hardware based reliability solutions
  - Per server RAID inside logical storage devices

- Replication can be added in a shim layer
  - Write each file to three servers
  - No reconstruction/recovery in the prototype

PVFS with replication



PVFS shim layer under Hadoop



Carnegie Mellon
Parallel Data Laboratory

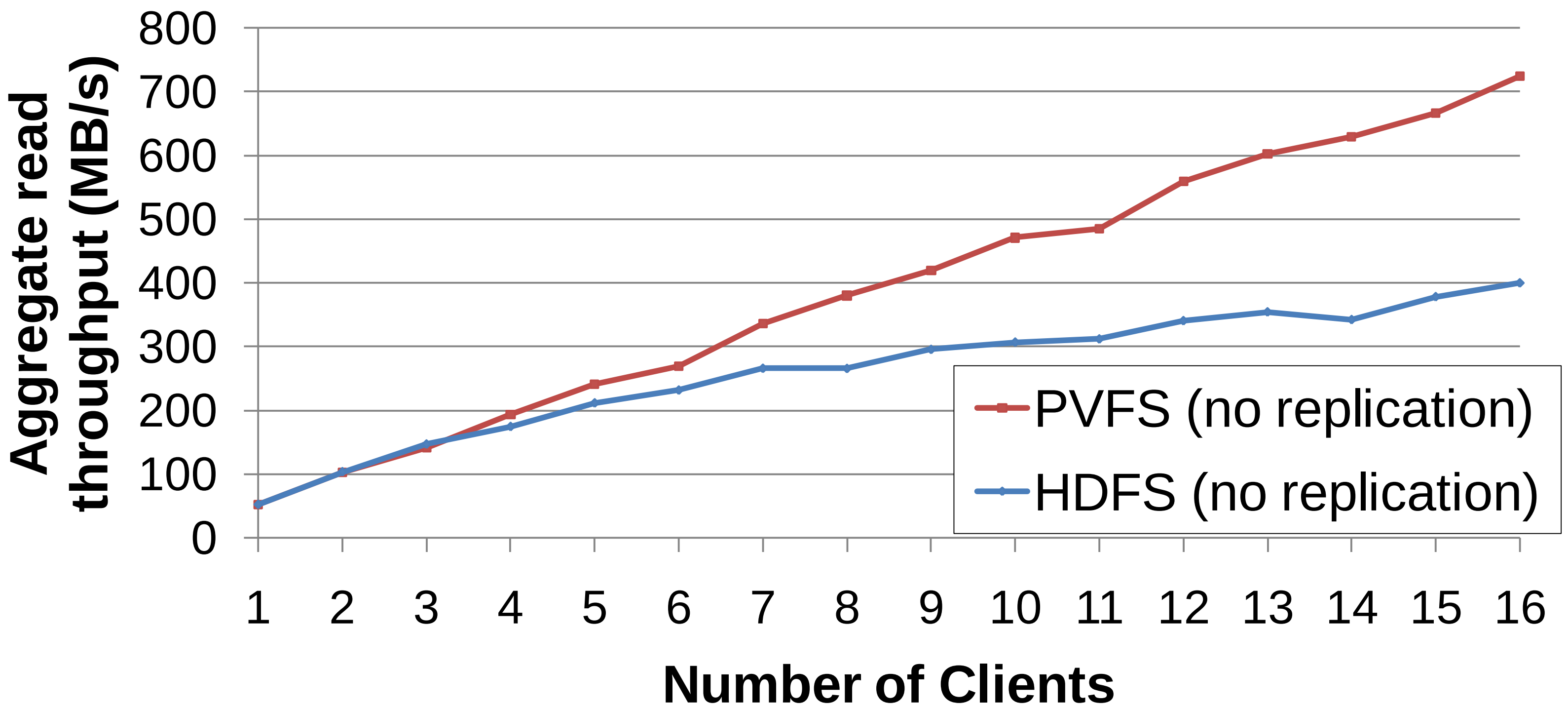
Outline

- A basic shim layer & preliminary evaluation
- Three add-on features in a shim layer
- ➢ Evaluation
  - ✓ Micro-benchmark (non MapReduce)
  - MapReduce benchmark

Micro-benchmark

- Cluster configuration
  - 16 nodes
  - Pentium D dual-core 3.0GHz
  - 4 GB Memory
  - One 7200 rpm SATA 160 GB (8 MB buffer)
  - Gigabit Ethernet
- Use file system API directly without Hadoop involvement

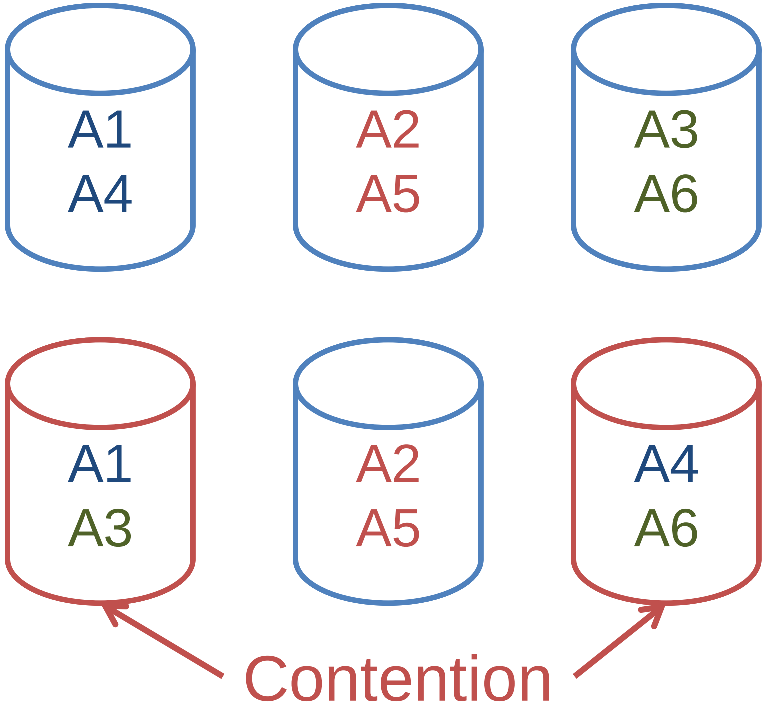
N clients, each reads 1/N of single file



- Round-robin file layout in PVFS helps avoid contention

Why is PVFS better in this case?

- Without scheduling, clients read in a uniform pattern
  - Client1 reads A1 then A4
  - Client2 reads A2 then A5
  - Client3 reads A3 then A6
- PVFS
  - Round-robin placement
- HDFS
  - Random placement



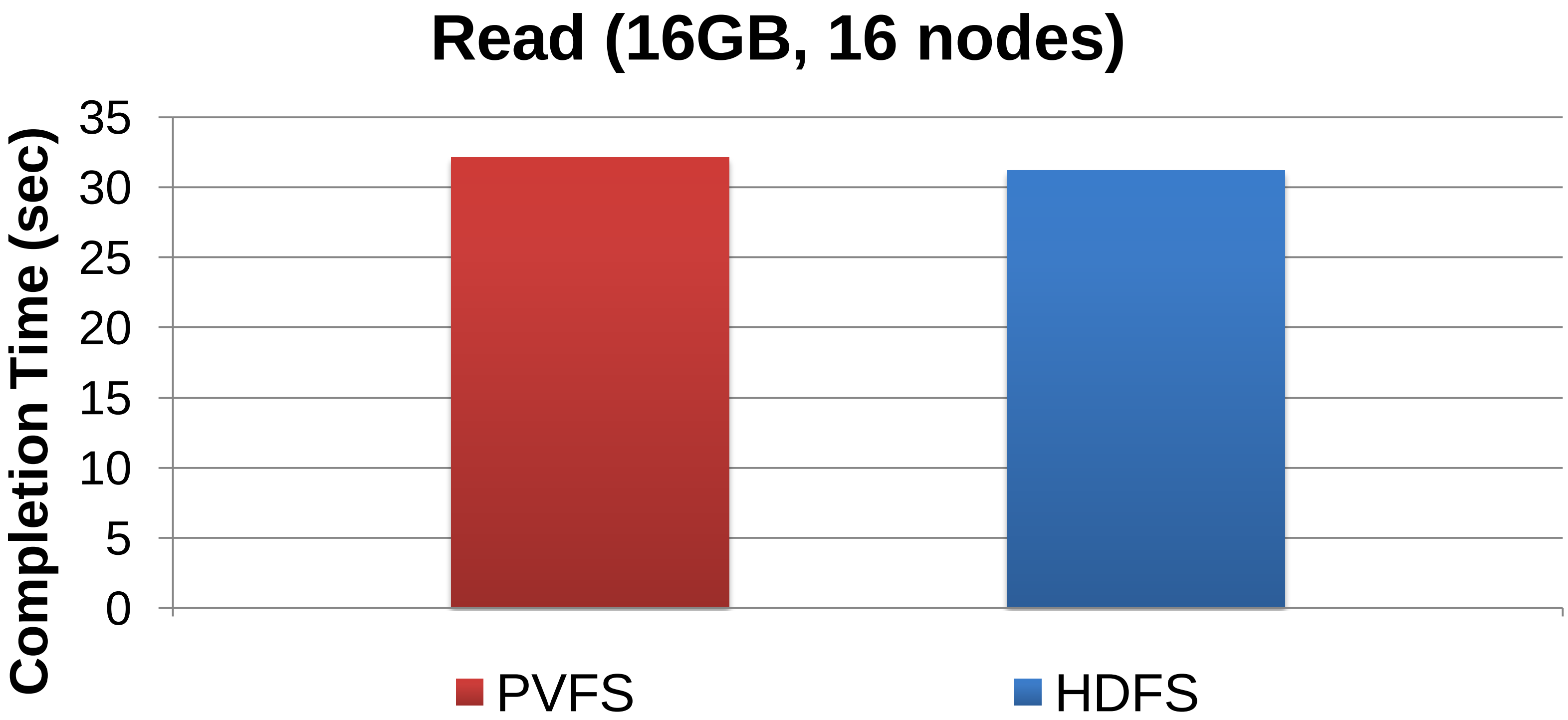
Carnegie Mellon
Parallel Data Laboratory

HDFS with Hadoop's scheduling

- Example 1:
  - Client1 reads A1 then A4
  - Client2 reads A2 then A5
  - Client3 reads A6 then A3

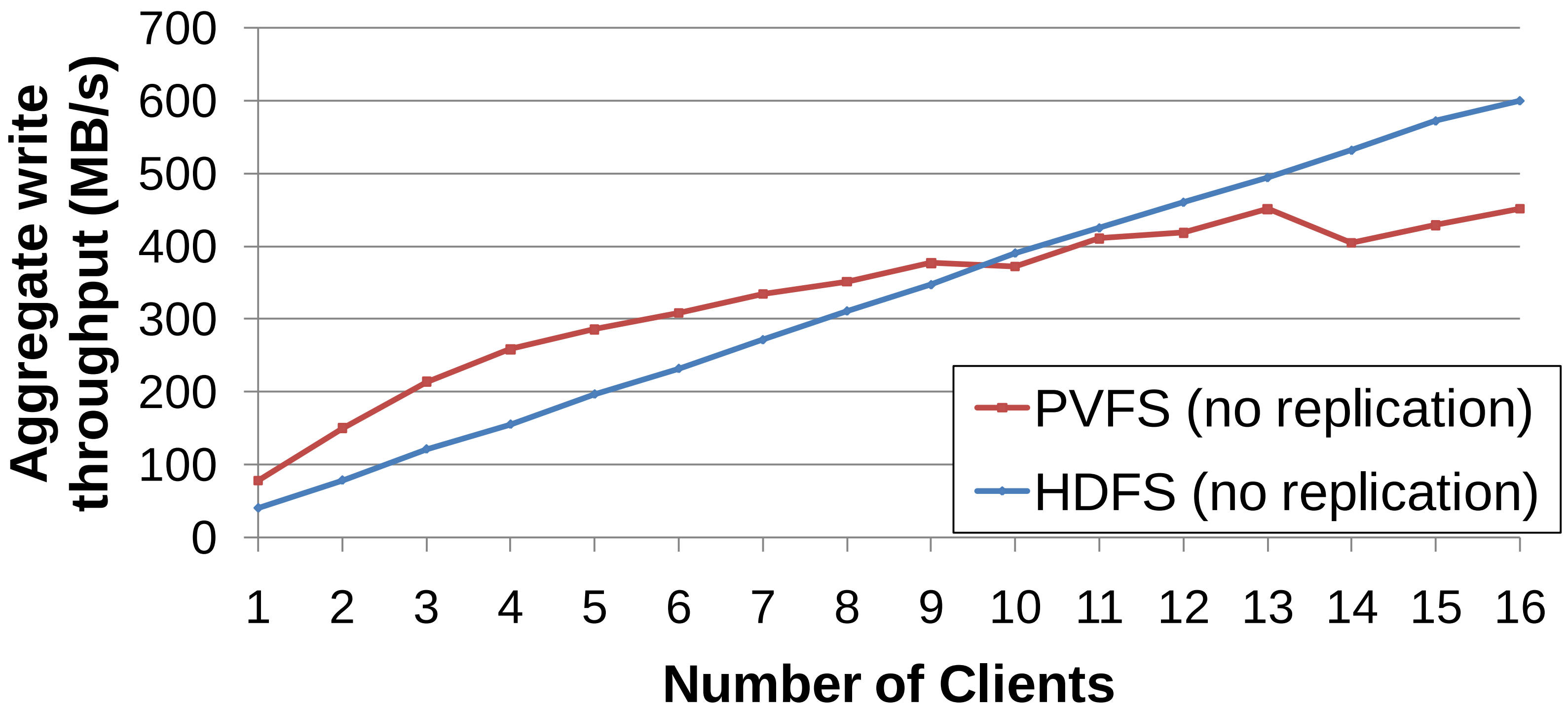
- Example 2:
  - Client1 reads A1 then A3
  - Client2 reads A2 then A5
  - Client3 reads A4 then A6

Read with Hadoop's scheduling



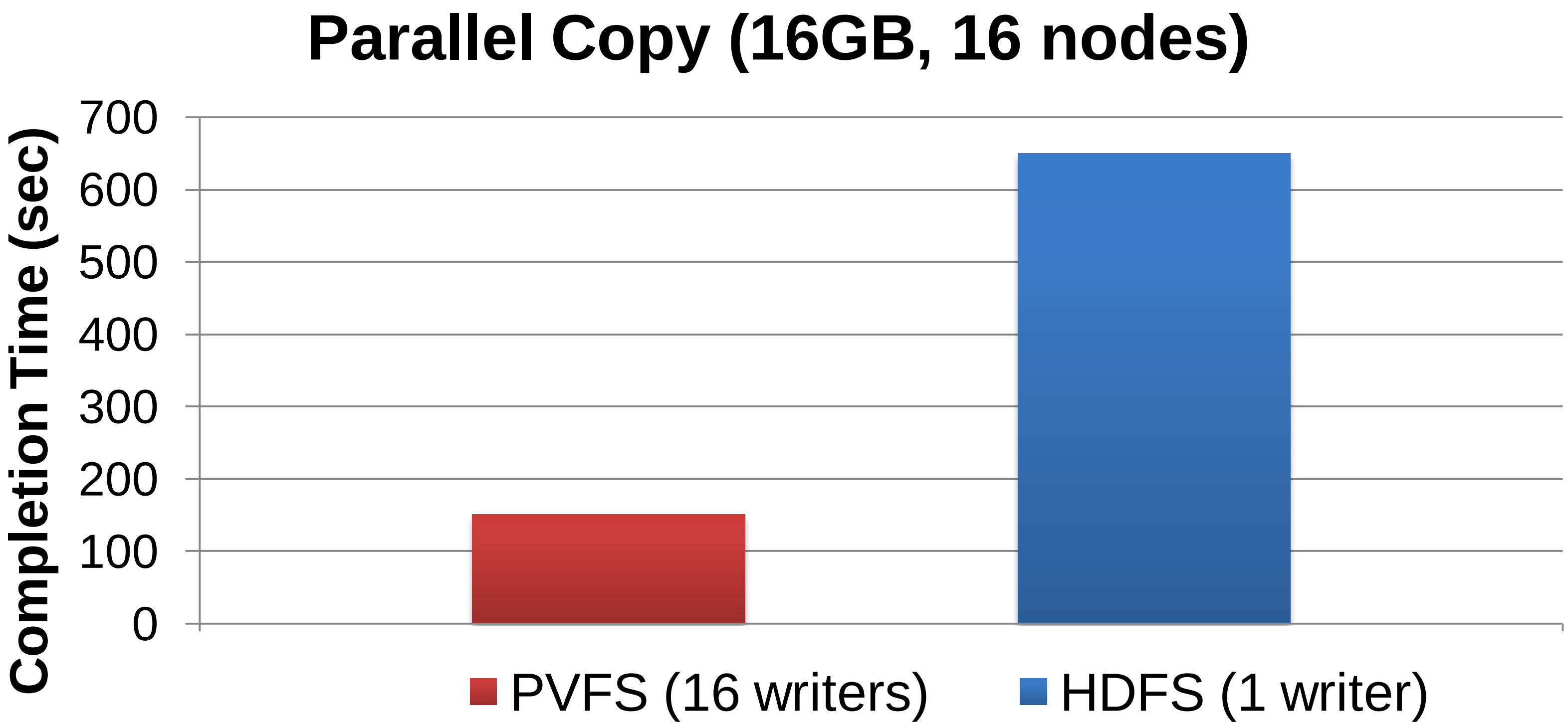
- Hadoop's scheduling can mask a problem with a non-uniform file layout in HDFS

N clients write to n distinct files



- By writing one of three copies locally, HDFS write throughput grows linearly

Concurrent writes to a single file



- By allowing concurrent writes in PVFS, "copy" completes faster by using multiple writers

Outline

- A basic shim layer & preliminary evaluation
- Three add-on features in a shim layer
- ➢ Evaluation
  - Micro-benchmark (non MapReduce)
  - ✓ MapReduce benchmark

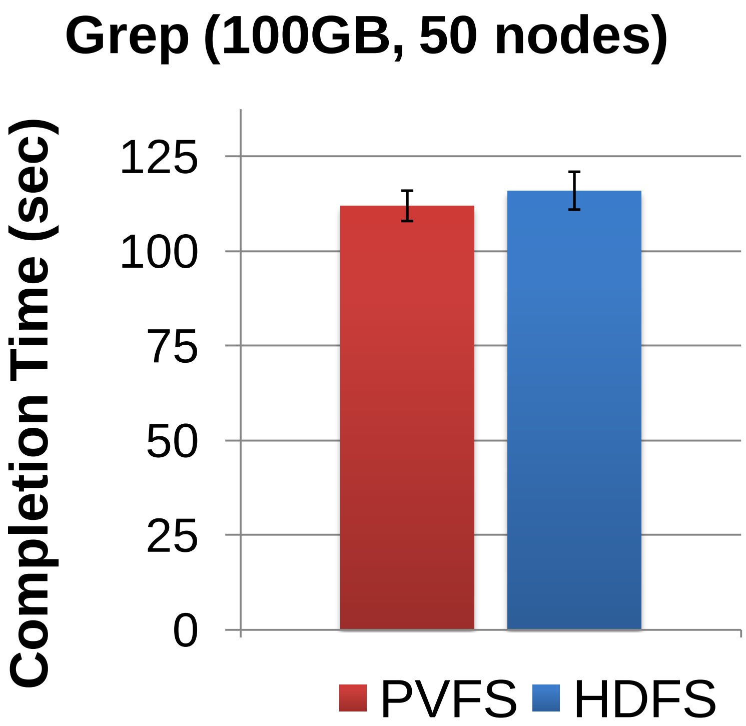
MapReduce benchmark setting

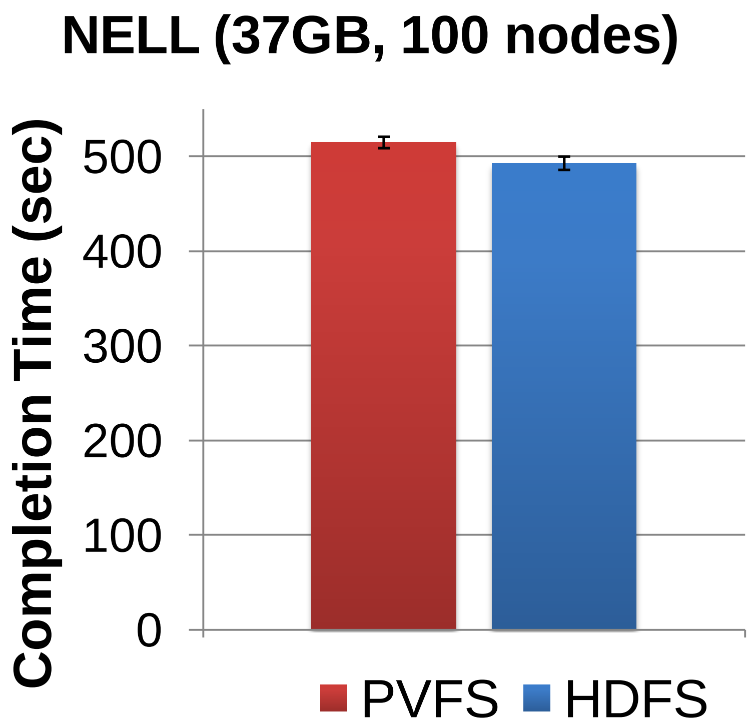
- Yahoo! M45 cluster
  - Use 50-100 nodes
  - Xeon quad-core 1.86 GHz with 6GB Memory
  - One 7200 rpm SATA 750 GB (8 MB buffer)
  - Gigabit Ethernet
- Use Hadoop framework for MapReduce processing

MapReduce benchmark

- **Grep**: Search for a rare pattern in hundred million 100-byte records (100GB)
- **Sort**: Sort hundred million 100-byte records (100GB)
- **Never-Ending Language Learning (NELL)**: (J. Betteridge, CMU) Count the numbers of selected phrases in 37GB data-set

Read-Intensive Benchmark



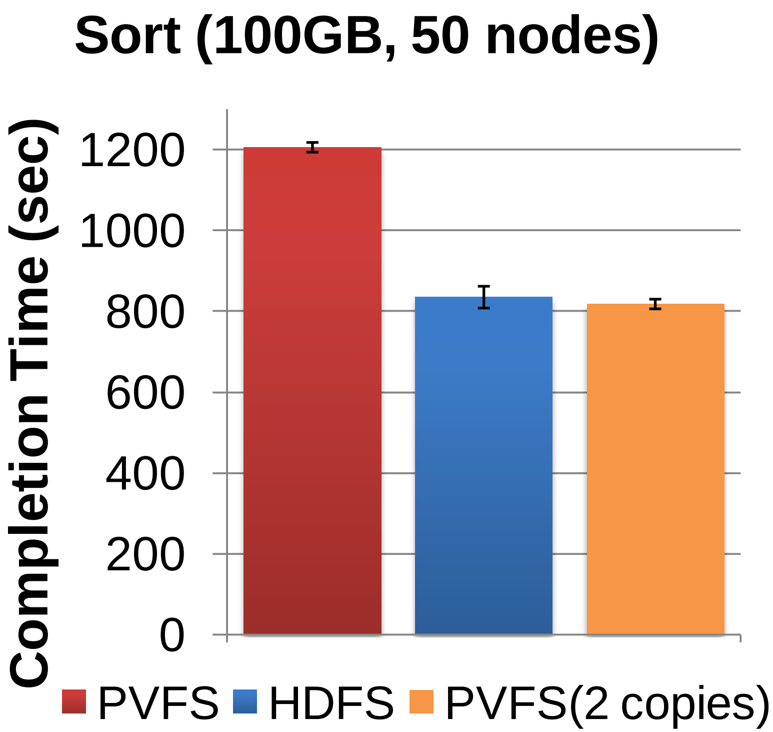


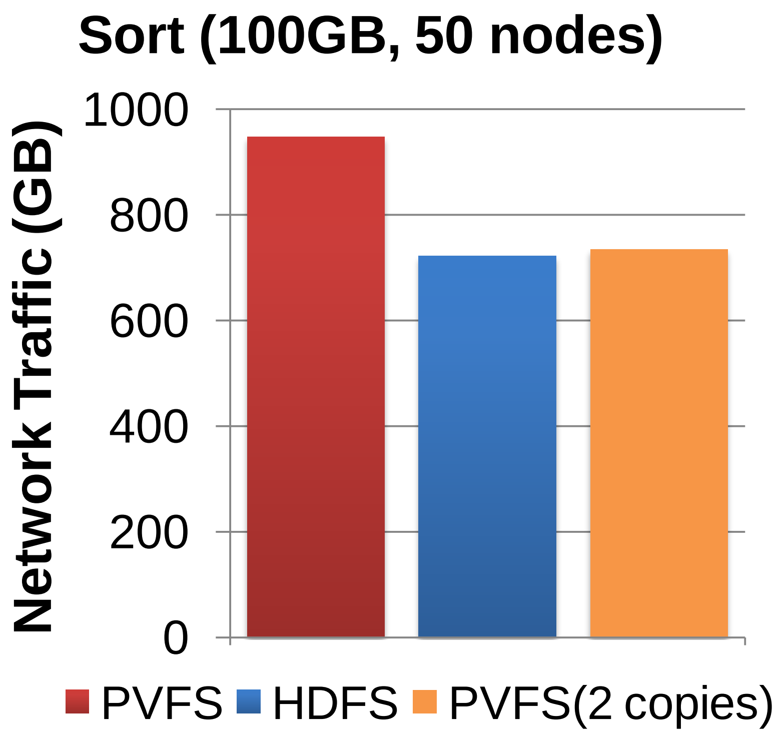
- PVFS's performance is similar to HDFS

Carnegie Mellon
Parallel Data Laboratory

 © November 08

Write-Intensive Benchmark





- By writing one of three copies locally, HDFS does better than PVFS

Summary

- PVFS can be tuned to deliver promising performance for Hadoop applications
  - Simple shim layer in Hadoop
  - No modification to PVFS
- PVFS can expose file layout information
  - Enable Hadoop to collocate computation and data
- Hadoop application can benefit from concurrent writing supported by parallel file systems

Acknowledgements

- Sam Lang and Rob Ross for help with PVFS internals
- Yahoo! for the M45 cluster
- Julio Lopez for help with M45 and Hadoop
- Justin Betteridge, Le Zhao, Jamie Callan, Shay Cohen, Noah Smith, U Kang and Christos Faloutsos for their scientific applications