On the Duality of Data-intensive File System Design: Reconciling HDFS and PVFS

Wittawat Tantisiriroj

Swapnil Patil, Garth Gibson (CMU)

Seung Woo Son, Samuel J. Lang, Robert B. Ross (ANL)

Motivation

- Cloud workloads
  - Large input data set (e.g. the entire web)
  - Distributed, parallel application execution
- HPC: tightly coupled tasks
  - Low latency, high bisection bandwidth networking
  - Separation of compute and storage nodes
- Cloud: loosely coupled tasks
  - More loosely networking
  - Collocation of compute and storage nodes
  - Many new specialized distributed file systems

File systems for data-intensive workloads

Cloud

hadoop HDFS

Google FS

amazon web services™ S3

HPC

PVFS

panasas®

IBM GPFS

lustre®

Is there a key difference between these two types?

In this work …

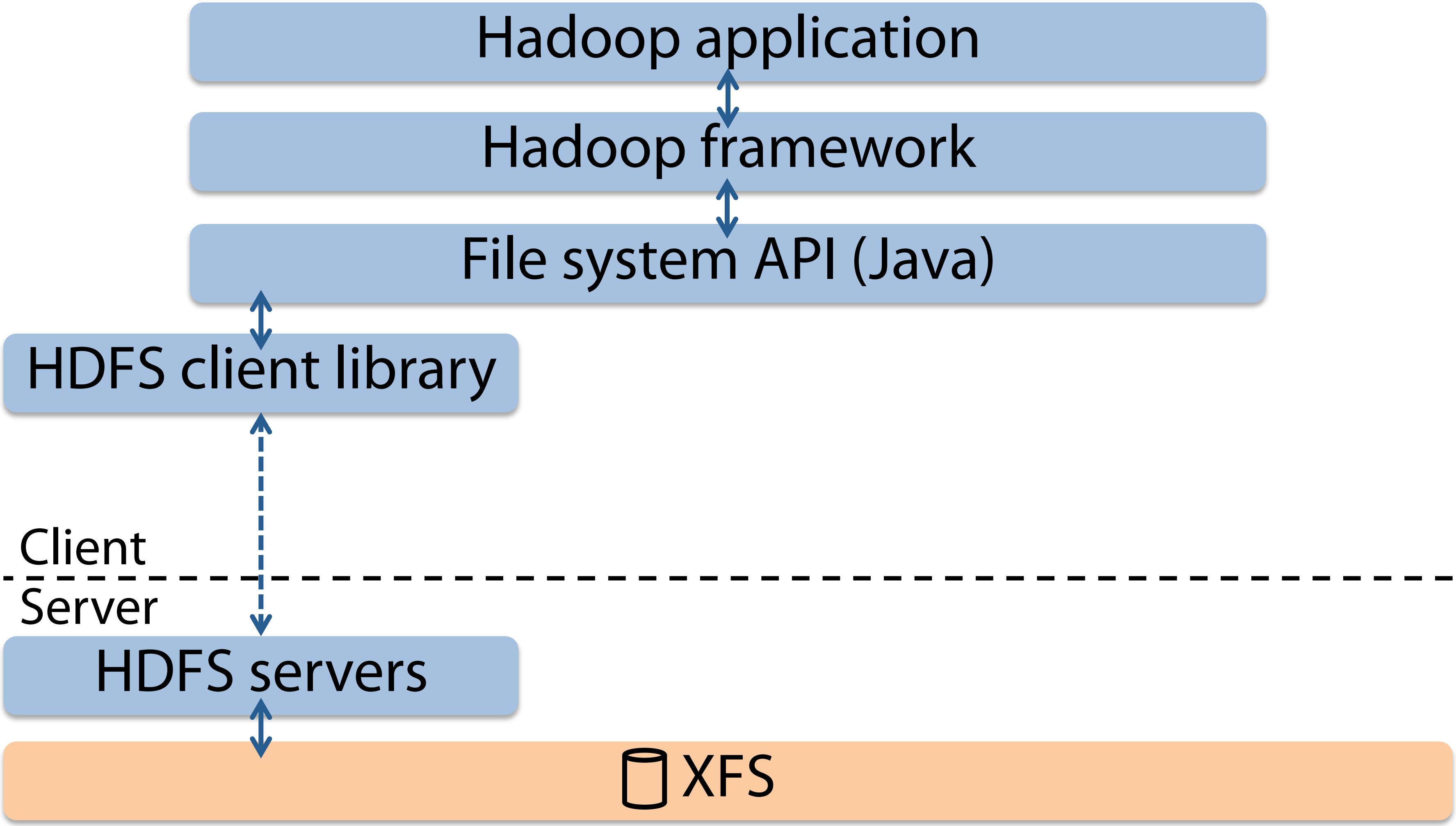
- Puts parallel FS into cloud framework and runs cloud workload (Hadoop)
  - **PVFS**: parallel file system
  - **HDFS**: file system designed for Hadoop
- Tells a story of making a parallel file system work well in a cloud workload

Outline

- ❖ Shim layer & vanilla PVFS
- Extra performance via shim layer
- Replication in shim layer
- Evaluation

Carnegie Mellon
Parallel Data Laboratory

Hadoop I/O path


Hadoop application

Hadoop framework

File system API (Java)

HDFS client library

Client

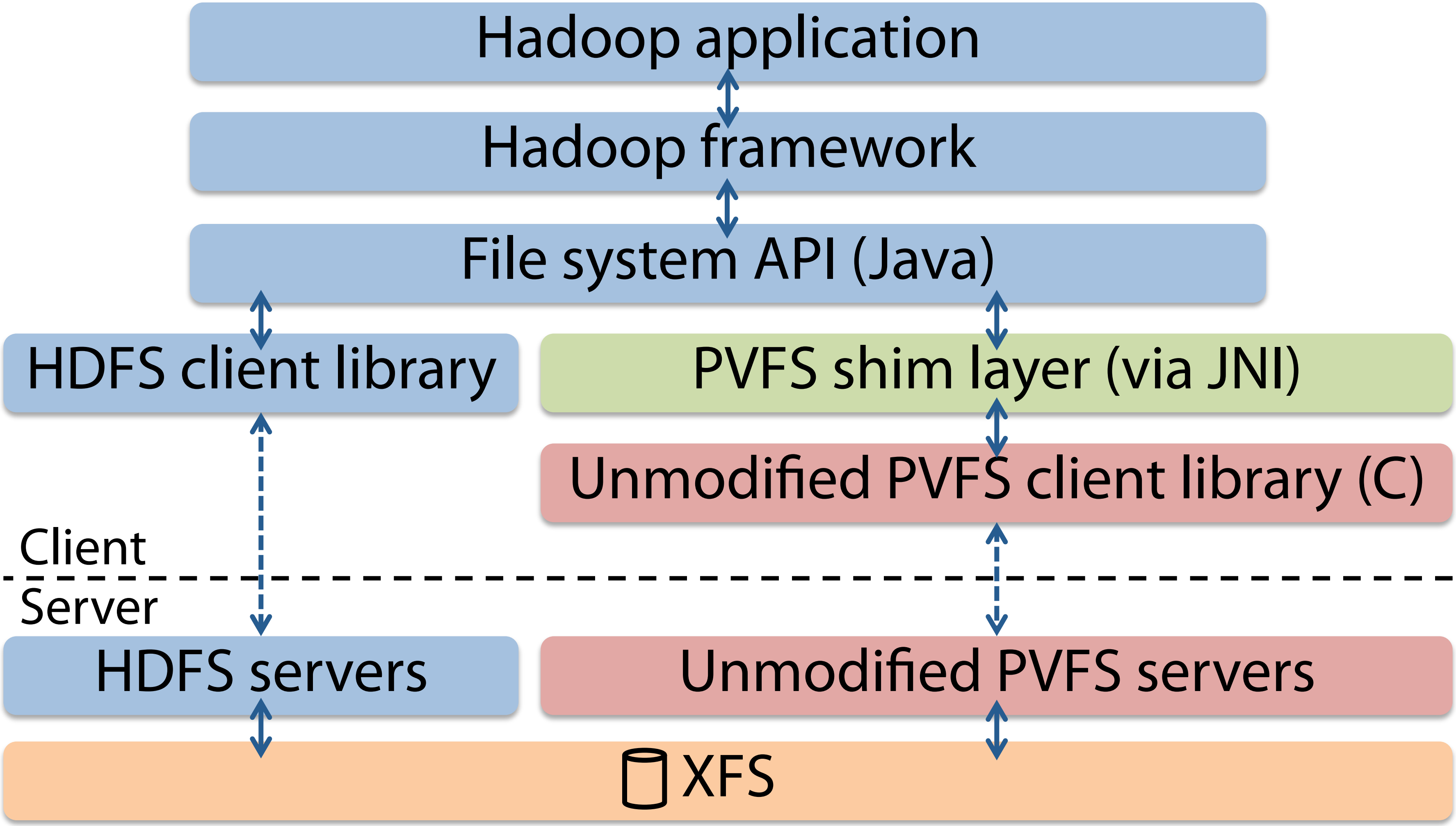
Server

HDFS servers

XFS

Carnegie Mellon
Parallel Data Laboratory

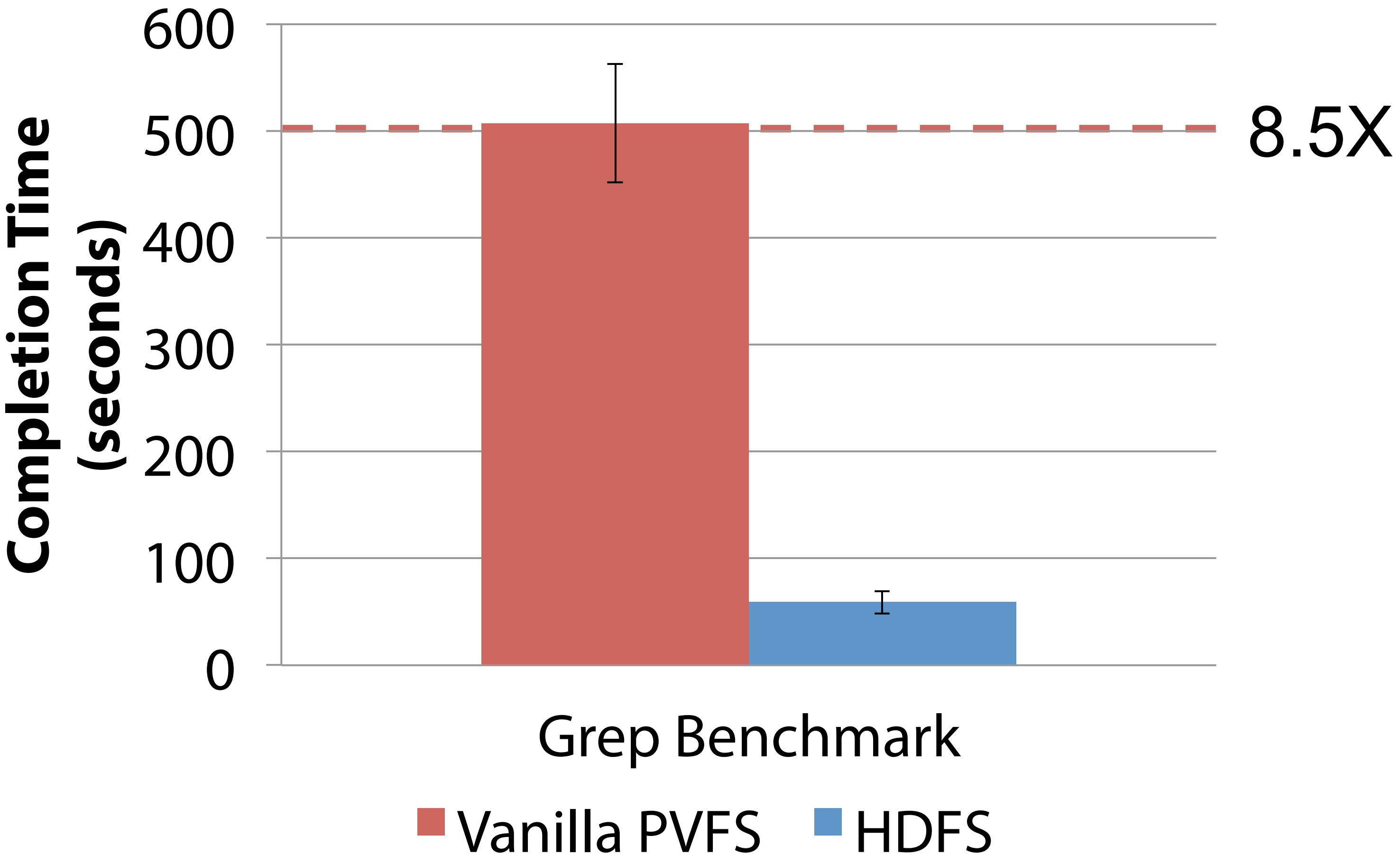
PVFS shim layer under Hadoop



Preliminary evaluation

- Text search (“grep”)
  - Common workload in Hadoop applications
- Searches for rare pattern in 100-byte records
  - 50GB dataset
  - 50 nodes
  - Each node serves as storage and compute nodes

Vanilla PVFS* is disappointing



* with 64MB chunk size to match HDFS'

Carnegie Mellon
Parallel Data Laboratory

Outline

- Shim layer & vanilla PVFS

❖ Extra performance via shim layer
  - Prefetching
  - File layout information

- Replication in shim layer

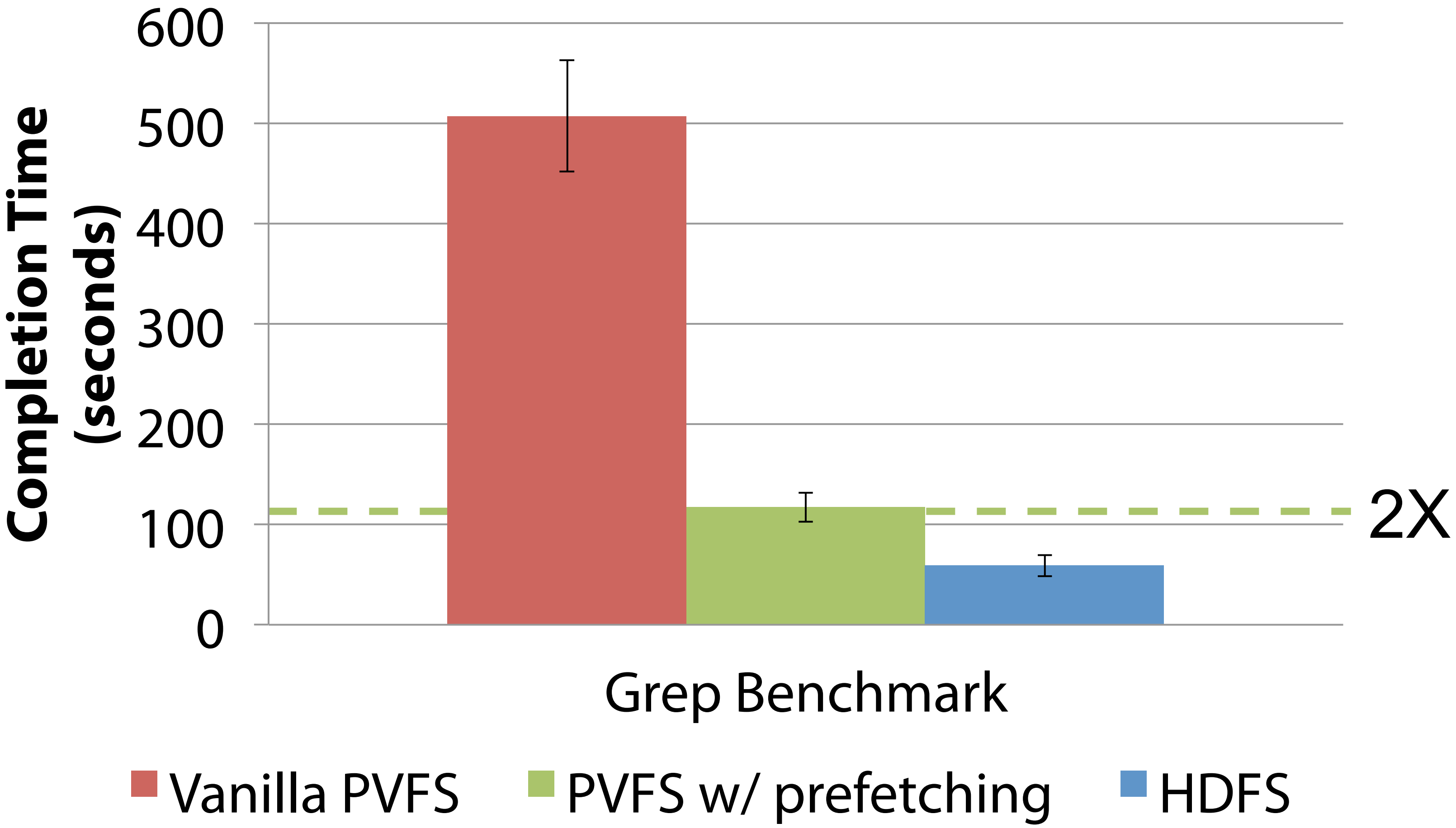
- Evaluation

Carnegie Mellon
Parallel Data Laboratory

Prefetching is useful for Hadoop

- Typical Hadoop workload:
  - Small read (less than 128KB)
  - Sequential through entire chunk (64MB)
- HDFS prefetches whole chunks
  - In Hadoop, file becomes immutable after closed
- PVFS: no need for prefetching
  - Apps do large reads
  - Simple: no cache coherence mechanism
- Prefetching added in shim layer

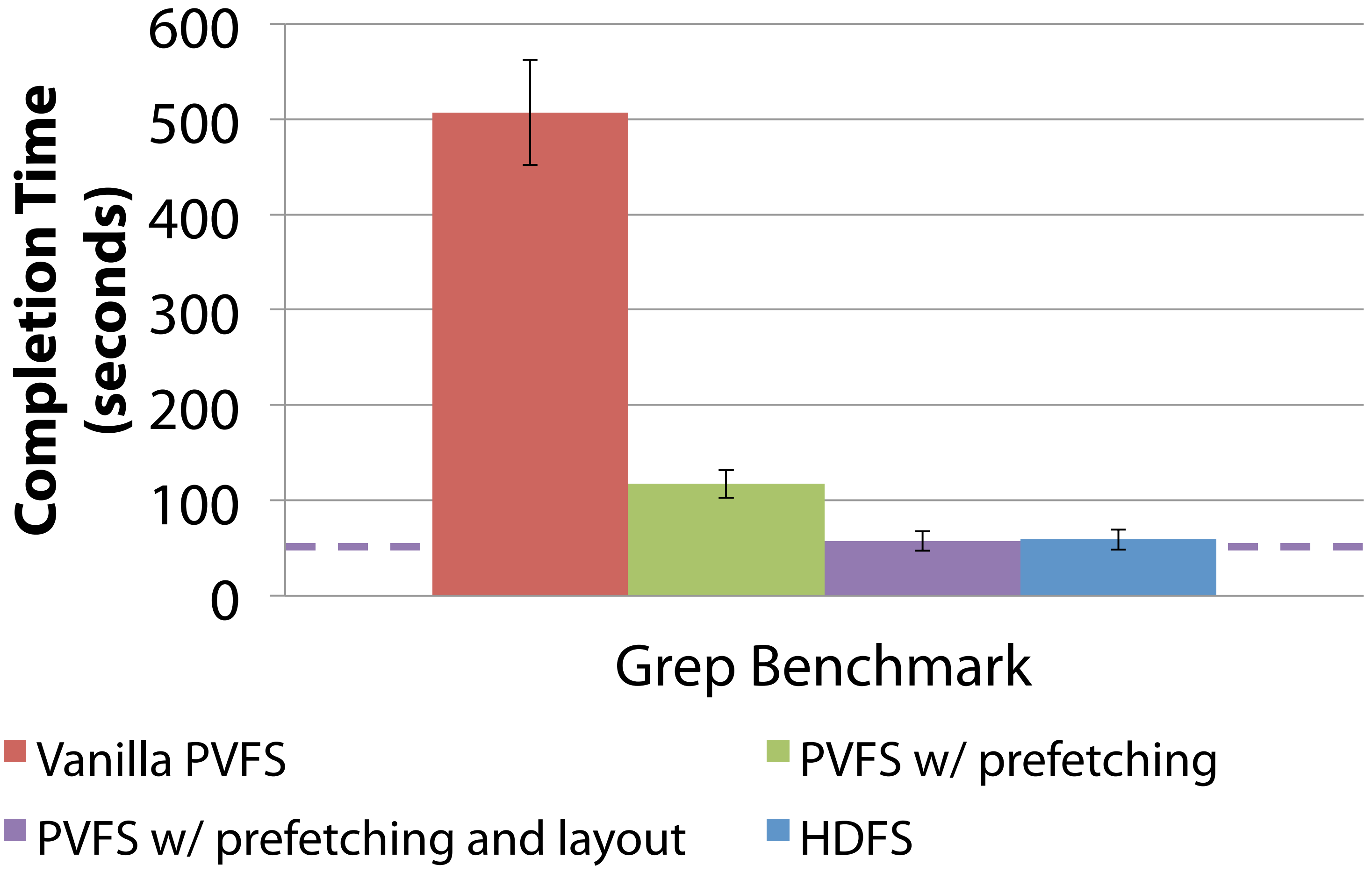
Improving but still 2X slower



Collocation in Hadoop

- Ships computation to where data is located
  - Helps reduce network traffic
- Requires file layout information
  - Describes where chunks are located
- Already available at PVFS client for direct I/O
  - No standard POSIX call to expose this info
  - Exposes this info to Hadoop in shim layer

Now, comparable performance



Carnegie Mellon
Parallel Data Laboratory

Outline

- Shim layer & vanilla PVFS
- Extra performance via shim layer
- ❖ Replication in shim layer
- Evaluation

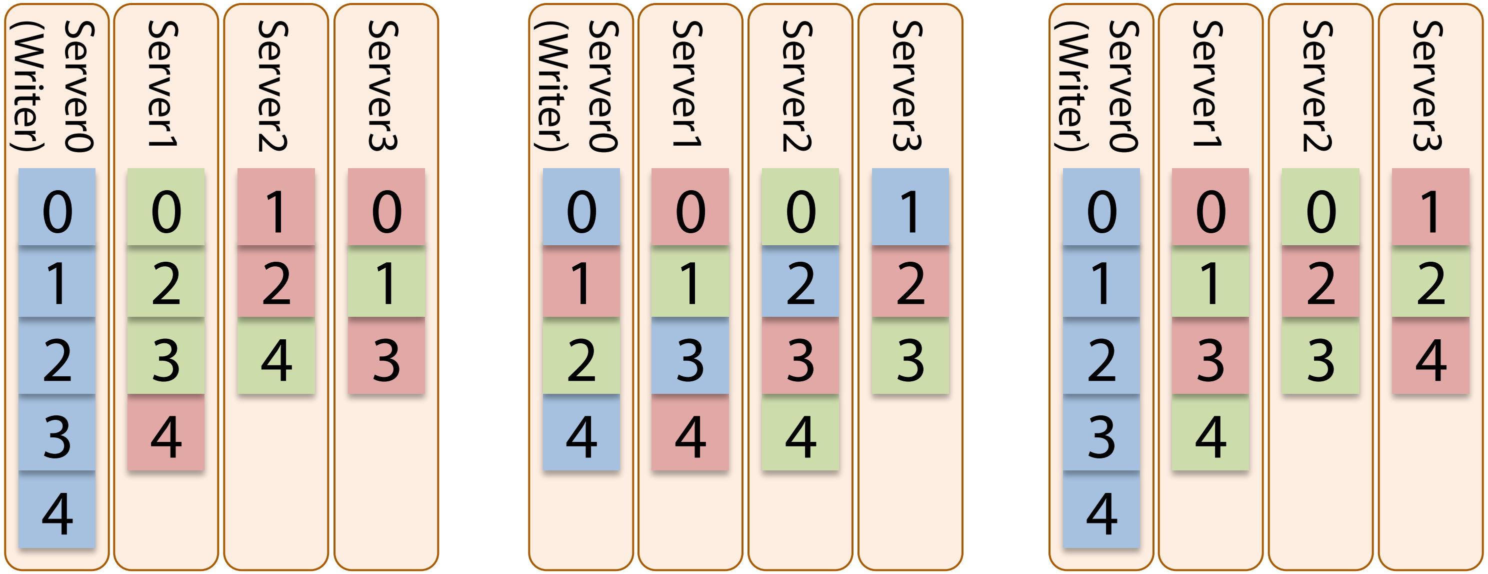
Carnegie Mellon
Parallel Data Laboratory

Replication vs RAID

- HDFS: software-based replication
  - 1 local copy on writer’s disks, 2 copies random
- PVFS: hardware-based RAID
  - Specialized hardware (RAID controller)

- PVFS with software-based replication
  - Implemented in shim layer
  - Demonstrate PVFS performance for comparison
  - Prototype has no error detection or repair process

Data layout schemes



Server0 (Writer)	Server1	Server2	Server3
🟦 0	🟩 0	🟥 1	🟥 0
🟦 1	🟩 2	🟥 2	🟩 1
🟦 2	🟩 3	🟩 4	🟥 3
🟦 3	🟥 4		
🟦 4			

HDFS

Server0 (Writer)	Server1	Server2	Server3
🟦 0	🟥 0	🟩 0	🟦 1
🟥 1	🟩 1	🟦 2	🟥 2
🟩 2	🟦 3	🟥 3	🟩 3
🟦 4	🟥 4	🟩 4	

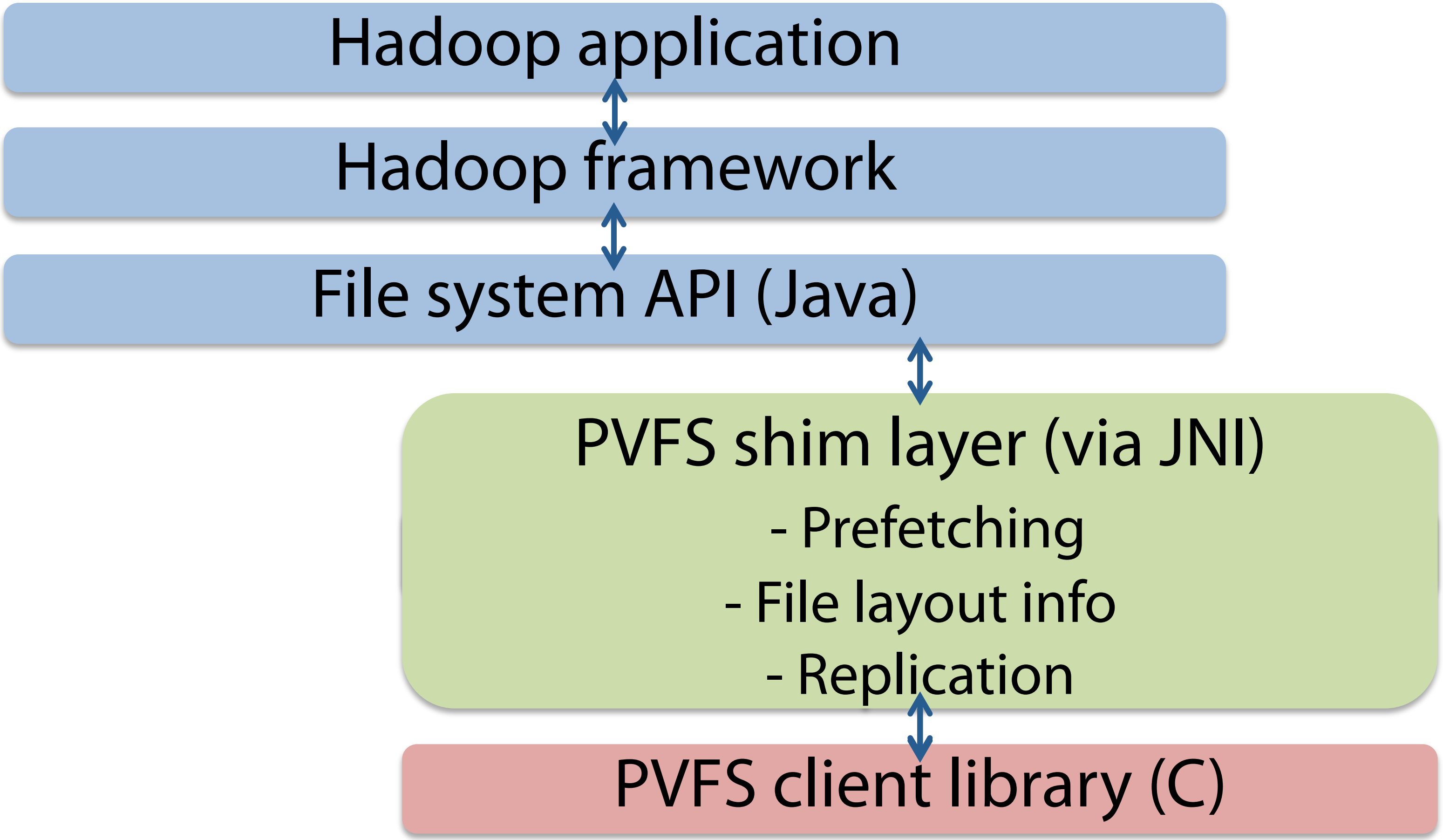
PVFS

Server0 (Writer)	Server1	Server2	Server3
🟦 0	🟥 0	🟩 0	🟥 1
🟦 1	🟩 1	🟥 2	🟩 2
🟦 2	🟥 3	🟩 3	🟥 4
🟦 3	🟩 4		
🟦 4			

PVFS Local

- HDFS: 1 copy on writer's disks, 2 copies random
- PVFS: 3 copies striped in file (straightforward way)
- PVFS Local: 1 copy on writer's disks, 2 striped
  - Small change and only change we made in PVFS server

Shim layer with 3 extensions



Hadoop application

Hadoop framework

File system API (Java)

PVFS shim layer (via JNI)

- Prefetching
- File layout info
- Replication

PVFS client library (C)

Outline

- Shim layer & vanilla PVFS

- Extra performance via shim layer

- Replication in shim layer

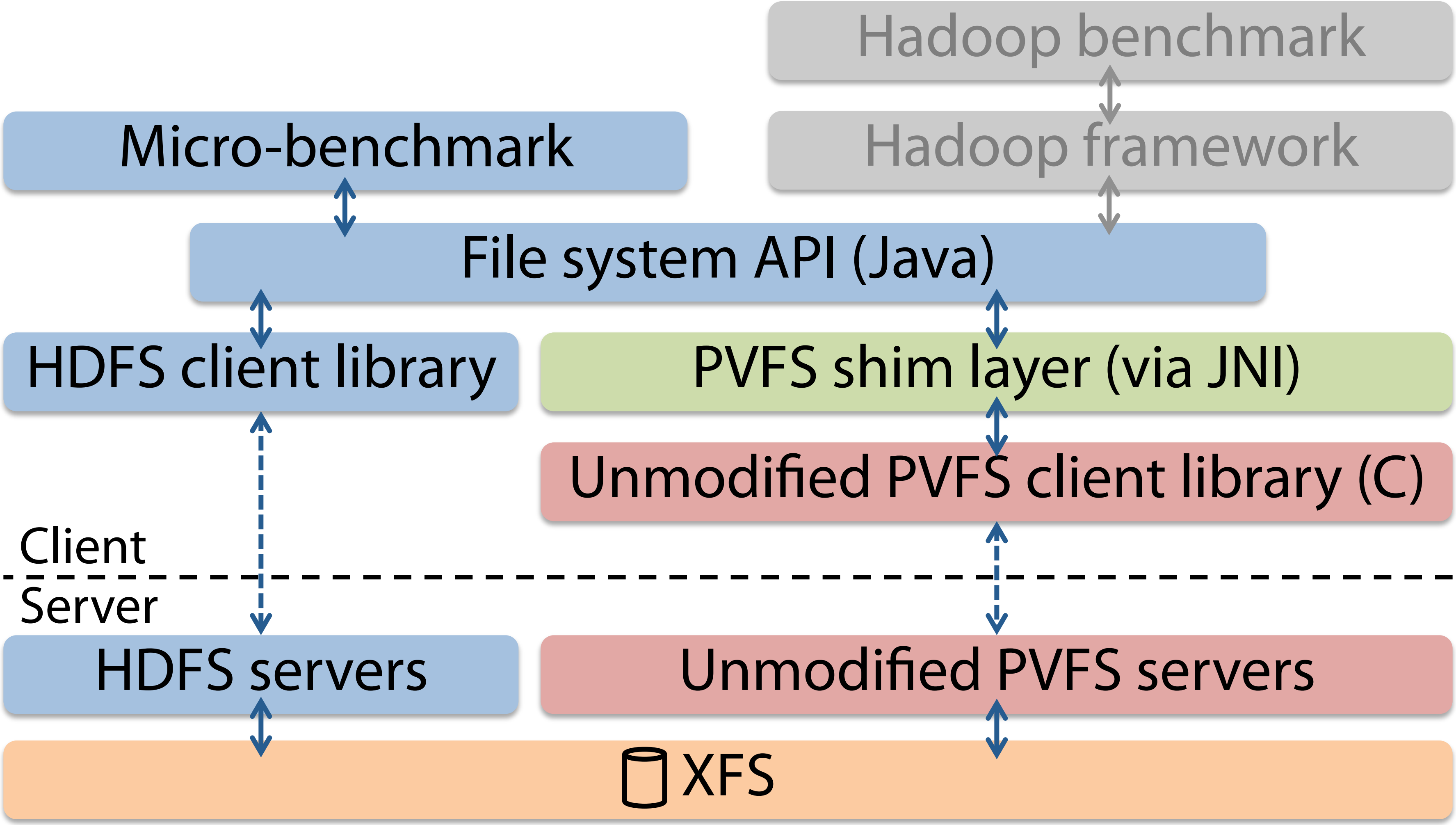
❖ Evaluation

  - Micro-benchmark
  - Hadoop benchmark

Micro-benchmark setting

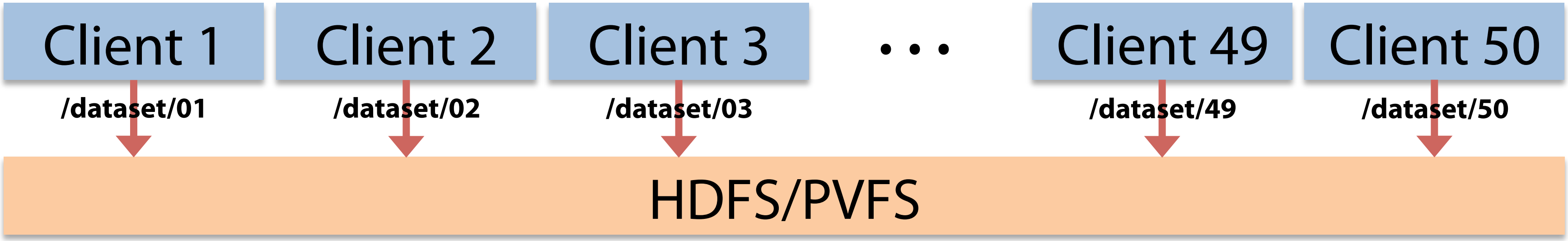
- Cluster configuration
  - 1 master nodes + 50 slave nodes
  - Pentium Xeon 2.8 GHz dual quad core
  - 16 GB Memory
  - One 7200 rpm SATA 160 GB (XFS)
  - 10 Gigabit Ethernet (60Gbps bisection bandwidth)

Using file system API directly

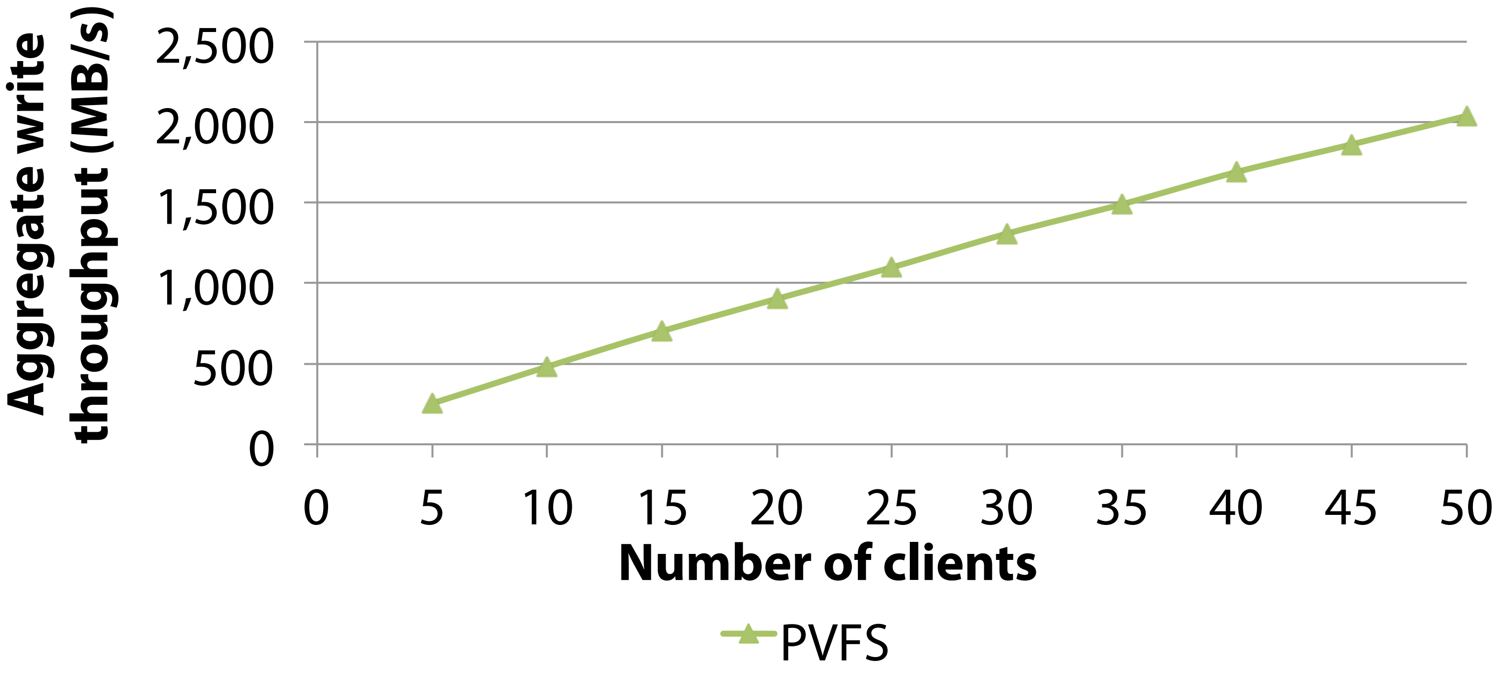


Write benchmark

1. 5,10,…, or 50 clients write to 50 data servers

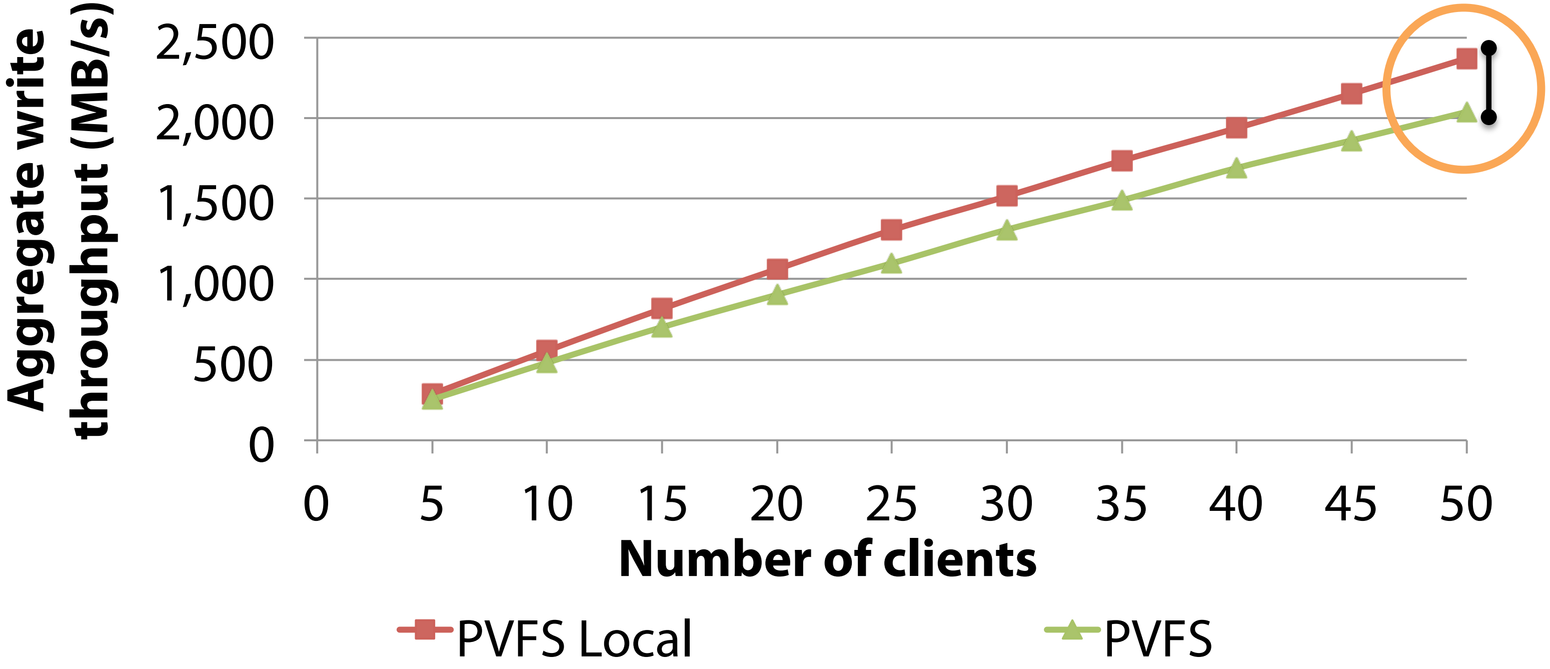


PVFS: linear speed up



Carnegie Mellon
Parallel Data Laboratory

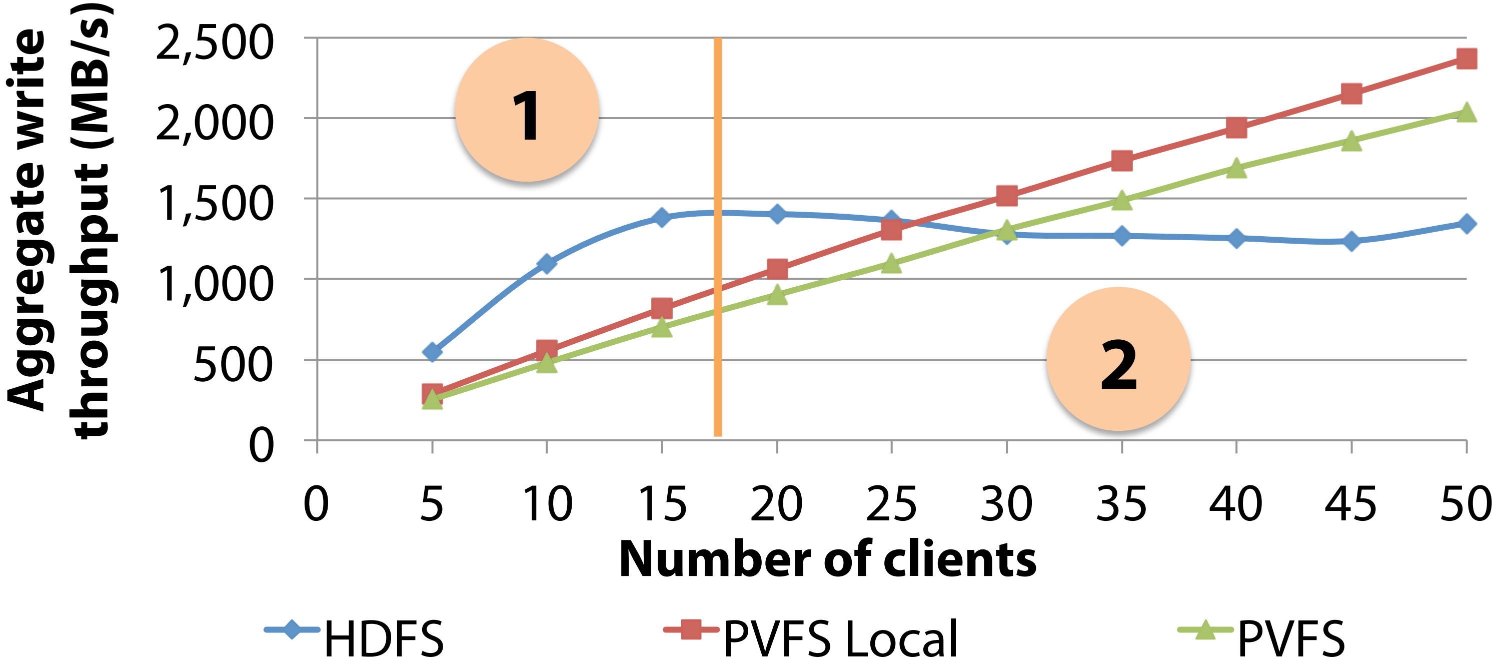
PVFS Local: 20% faster writes



- Reduces network traffic by 33%
  - 3 remote copies vs 2 remote copies

Carnegie Mellon
Parallel Data Laboratory

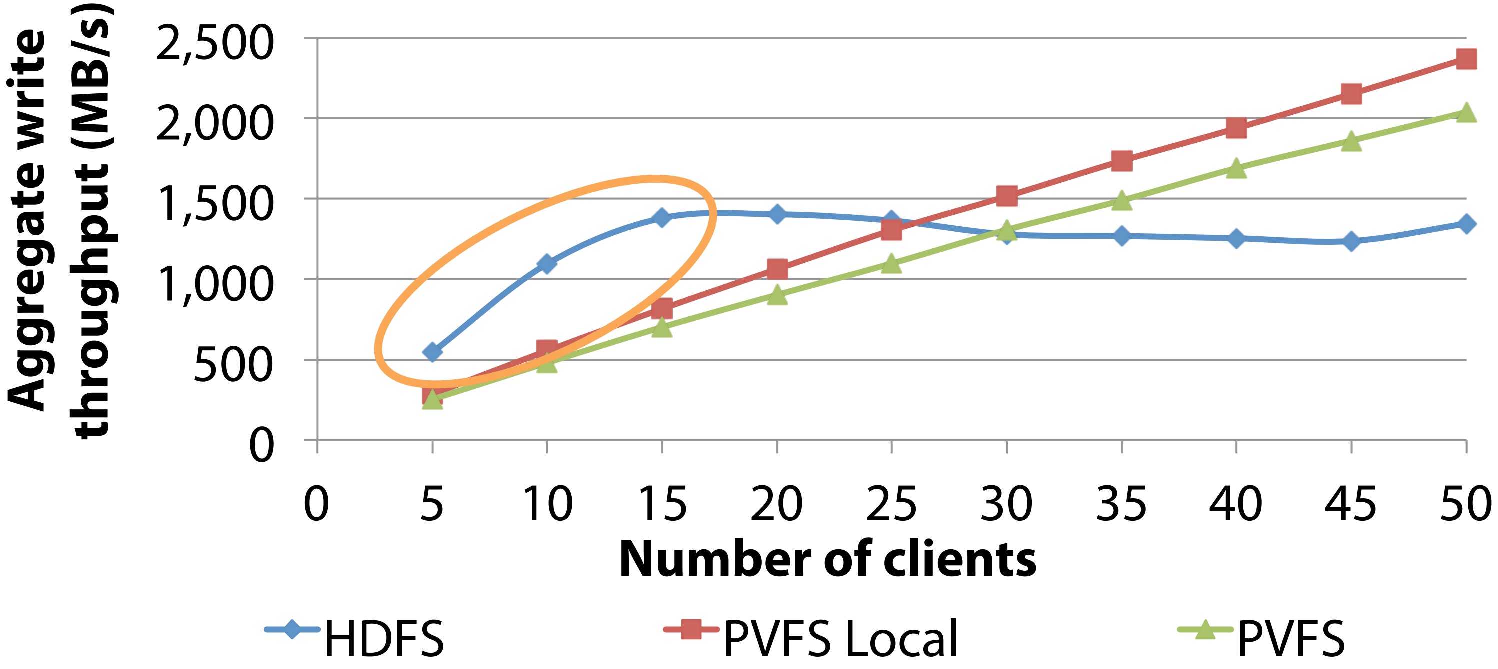
HDFS: two phases



① Faster writes

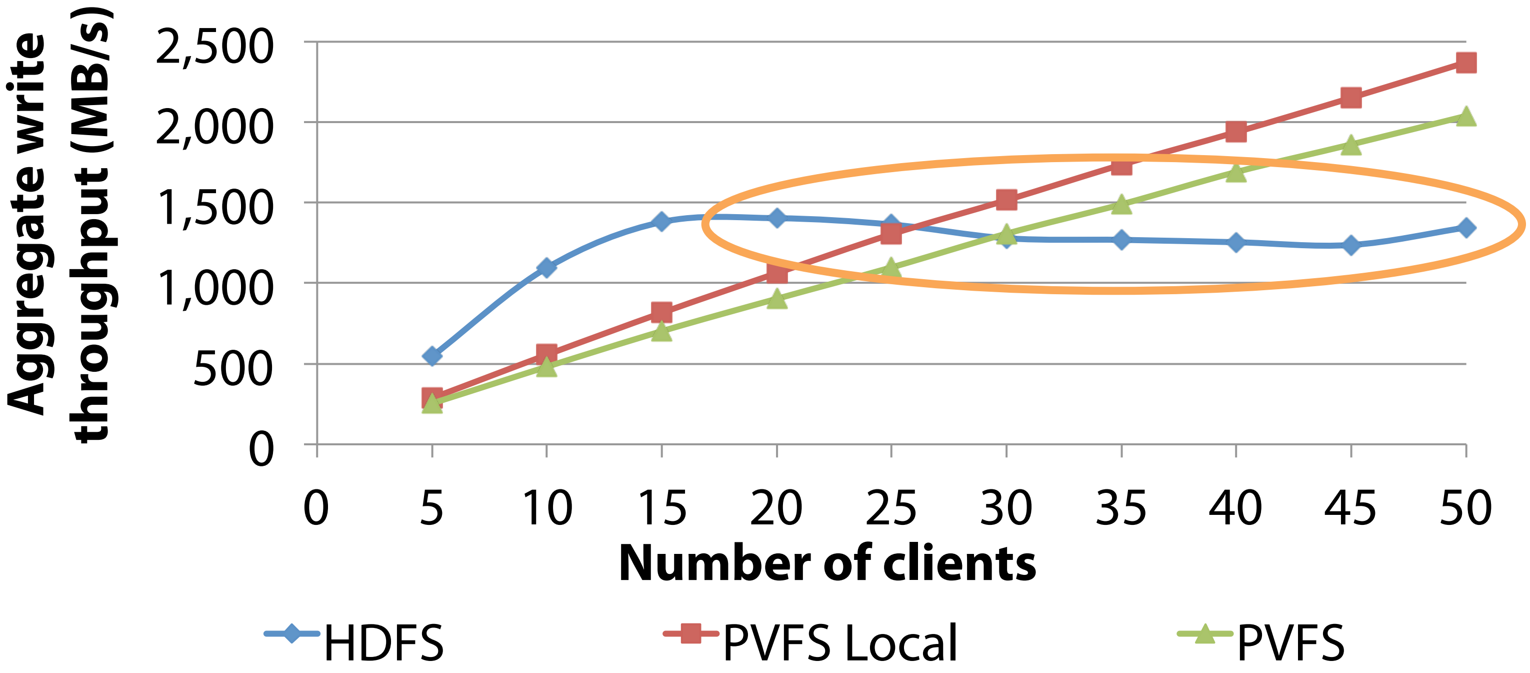
② Saturation

HDFS utilizes idle resource better



- HDFS pipelined replication improves parallelism
  - Better than client driven replication in PVFS

HDFS is surprisingly tied to disk performance

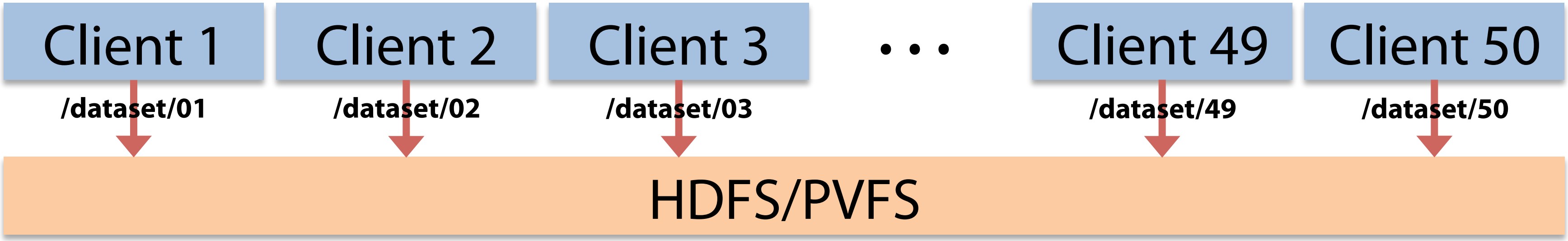


- Limited by synchronous chunk file creation
  - More metadata operation in HDFS
    - – New chunk file for each 64MB
  - XFS: file creation queues behind large writes

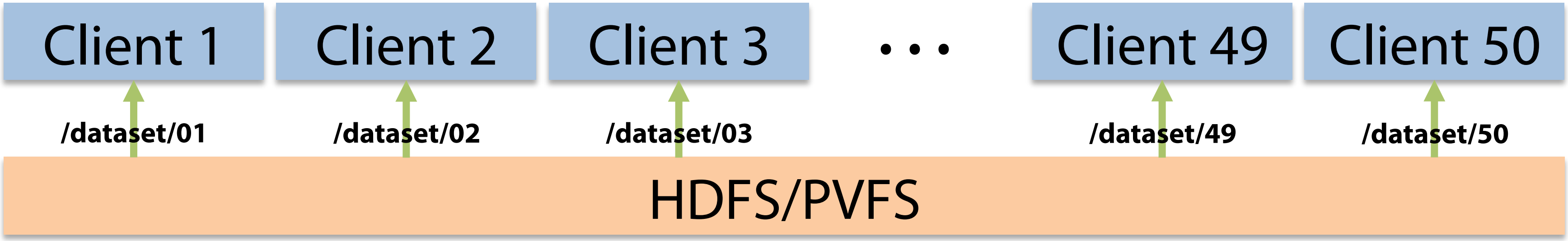
Carnegie Mellon
Parallel Data Laboratory

Read benchmark

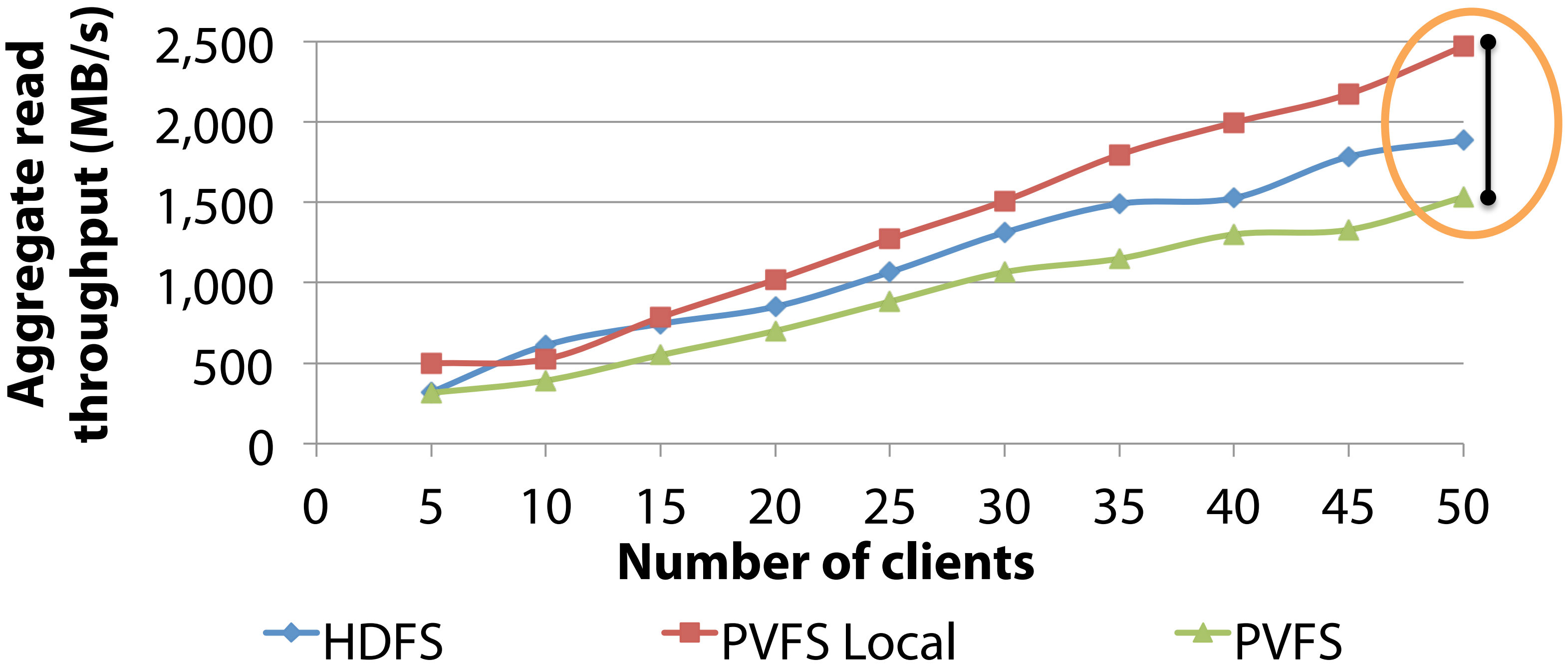
1. 50 clients write to 50 data servers



2. 5,10,…, or 50 clients read back
   - The same file each client creates



PVFS Local: 60% faster reads



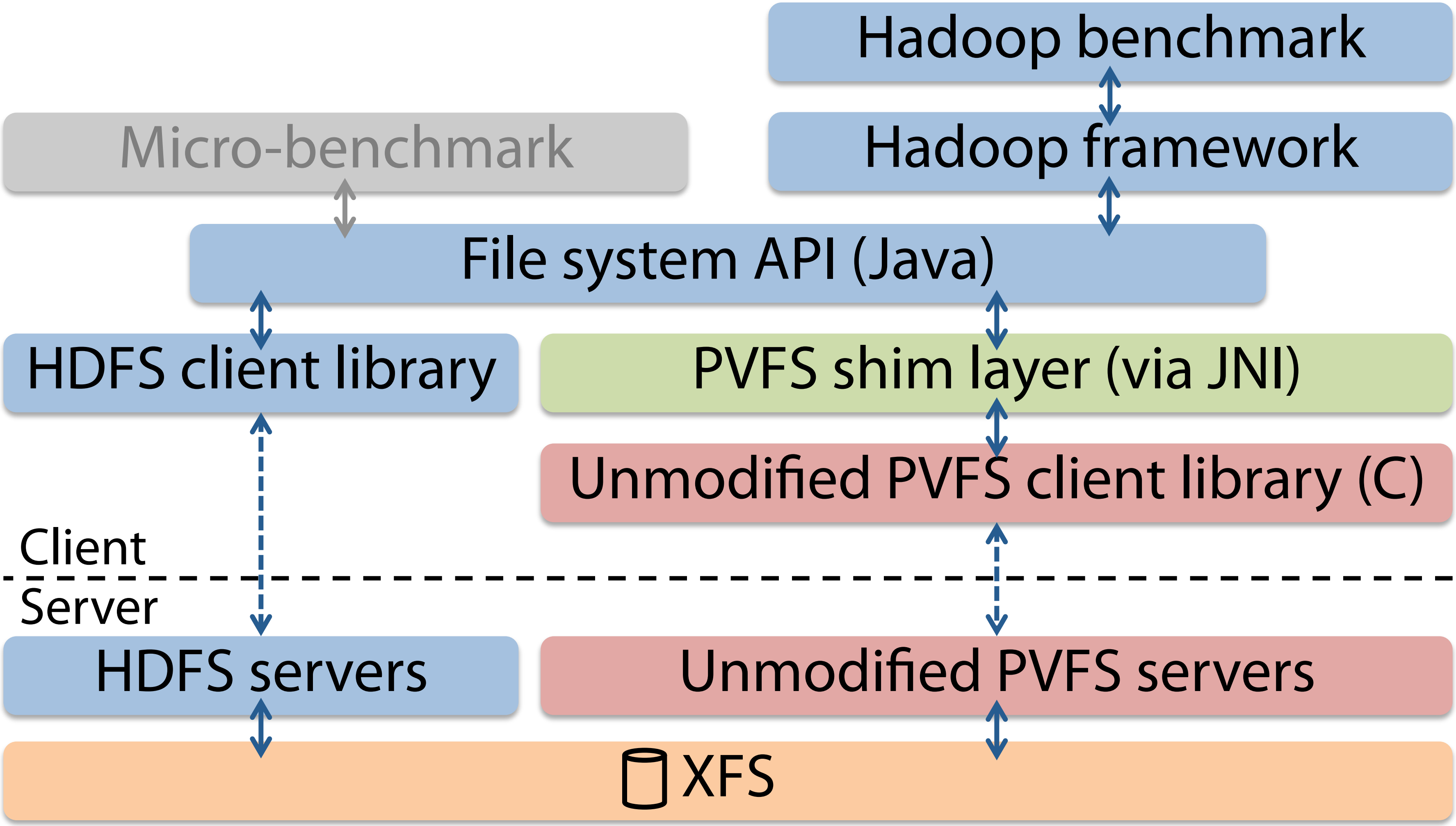
- Larger improvement than write
    - No network traffic at all

Carnegie Mellon
Parallel Data Laboratory

Hadoop benchmark setting

- Cluster configuration
  - 2 master nodes + 50 slave nodes
  - Pentium Xeon 2.8 GHz dual quad core
  - 16 GB Memory
  - One 7200 rpm SATA 160 GB (XFS)
  - 10 Gigabit Ethernet (60Gbps bisection bandwidth)

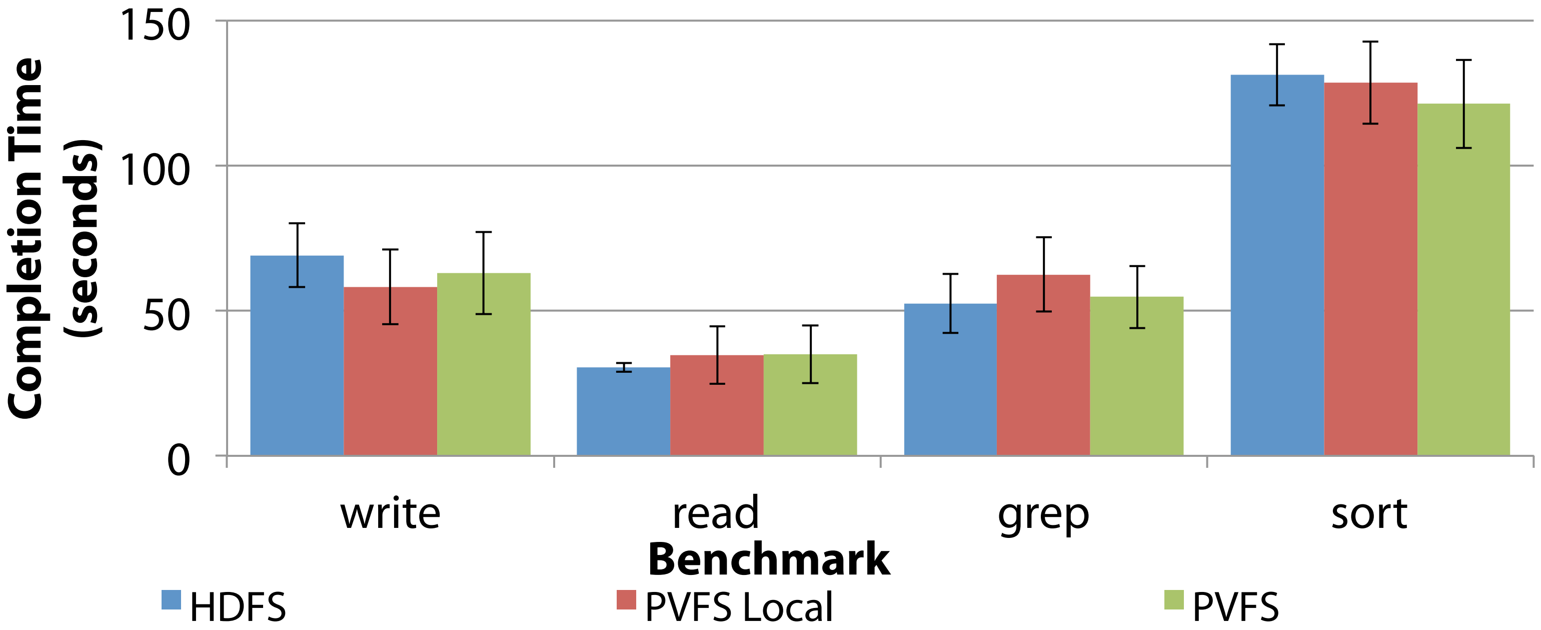
Using Hadoop scheduling



Hadoop benchmark

- **Dataset:** 50GB of 100-byte records
- **Standard Hadoop benchmarks:**
  - **Write**: generate dataset
  - **Read**: read with no computation
  - **Grep**: search for rare pattern
  - **Sort**: sort all records

Effect of Hadoop job scheduler

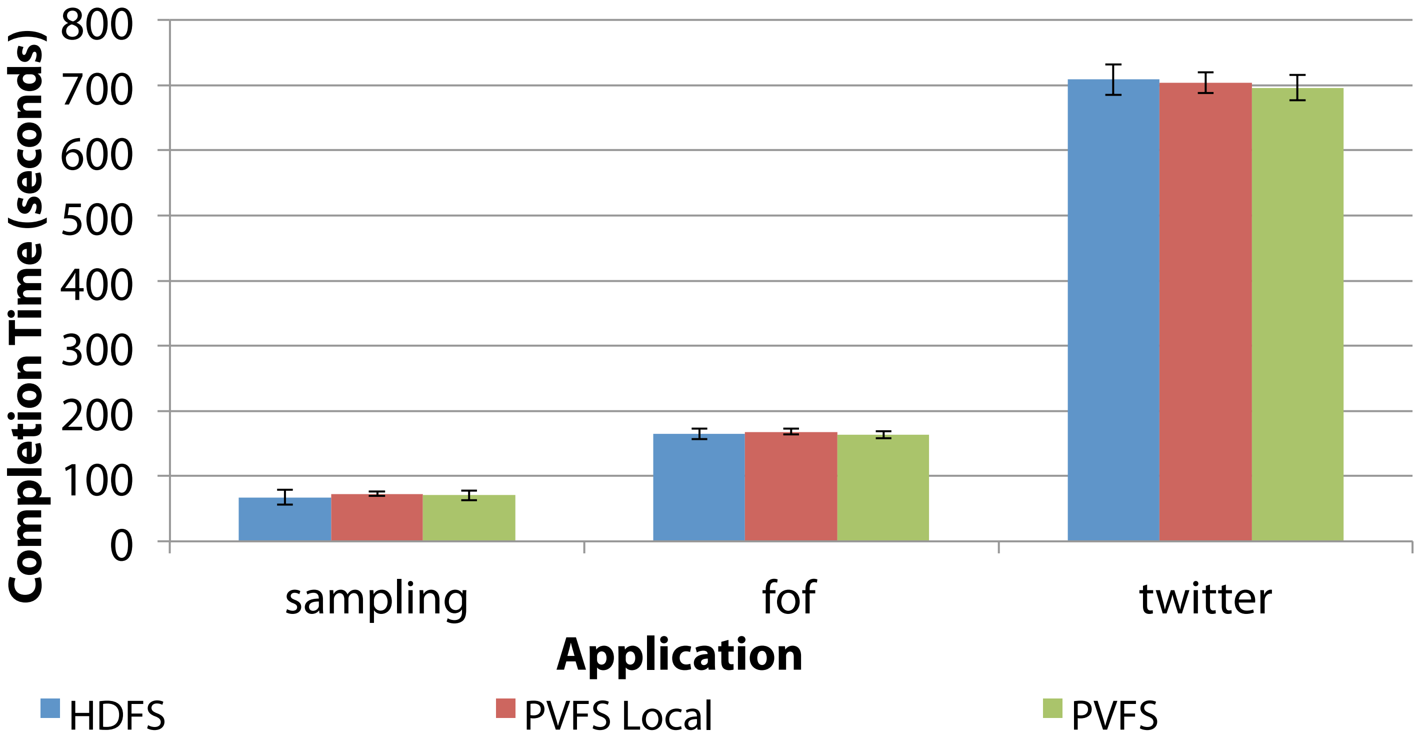


- Masks inefficiencies of PVFS w/o local copy
  - With collocation, most reads become local reads
  - Blocks often processed out of order

Scientific Hadoop applications

- **Sampling**: read 71GB astronomy dataset to determine partitions
- **FoF**: cluster & join astronomical objects of the same 71GB dataset
- **Twitter**: process raw 24GB Twitter dataset into different format

Results



- PVFS performance is comparable to HDFS

Carnegie Mellon
Parallel Data Laboratory

Summary

- Out-of-the box solution does not perform well
- Comparable performance after tuned
  - Prefetching + Collocation
- Key difference is in fault tolerance
  - Software replication vs hardware RAID
- Difference in performance
  - Significant in some use-cases (micro-benchmarks)
  - Little when using Hadoop scheduling
- Prototype available @ goo.gl/ooysb