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

Wittawat Tantisiroj

Swapnil Patil, Garth Gibson (CMU)

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

Motivation

- Applications become more data-intensive
 - Large input data set (e.g. the entire web)
 - Distributed, parallel application execution
- Many new distributed file systems
 - Commodity hardware and network
 - Purpose-built for anticipated workloads
 - New, diverse semantics (non-POSIX)

File systems for data-intensive workloads



Can existing parallel file systems support new workloads?

Why use parallel file systems?

- Handle a wide variety of workloads
 - High concurrent reads and writes
 - Small file support, scalable metadata
- Low capacity overhead reliability solutions
- Standard Unix FS interface
 - Support POSIX semantics

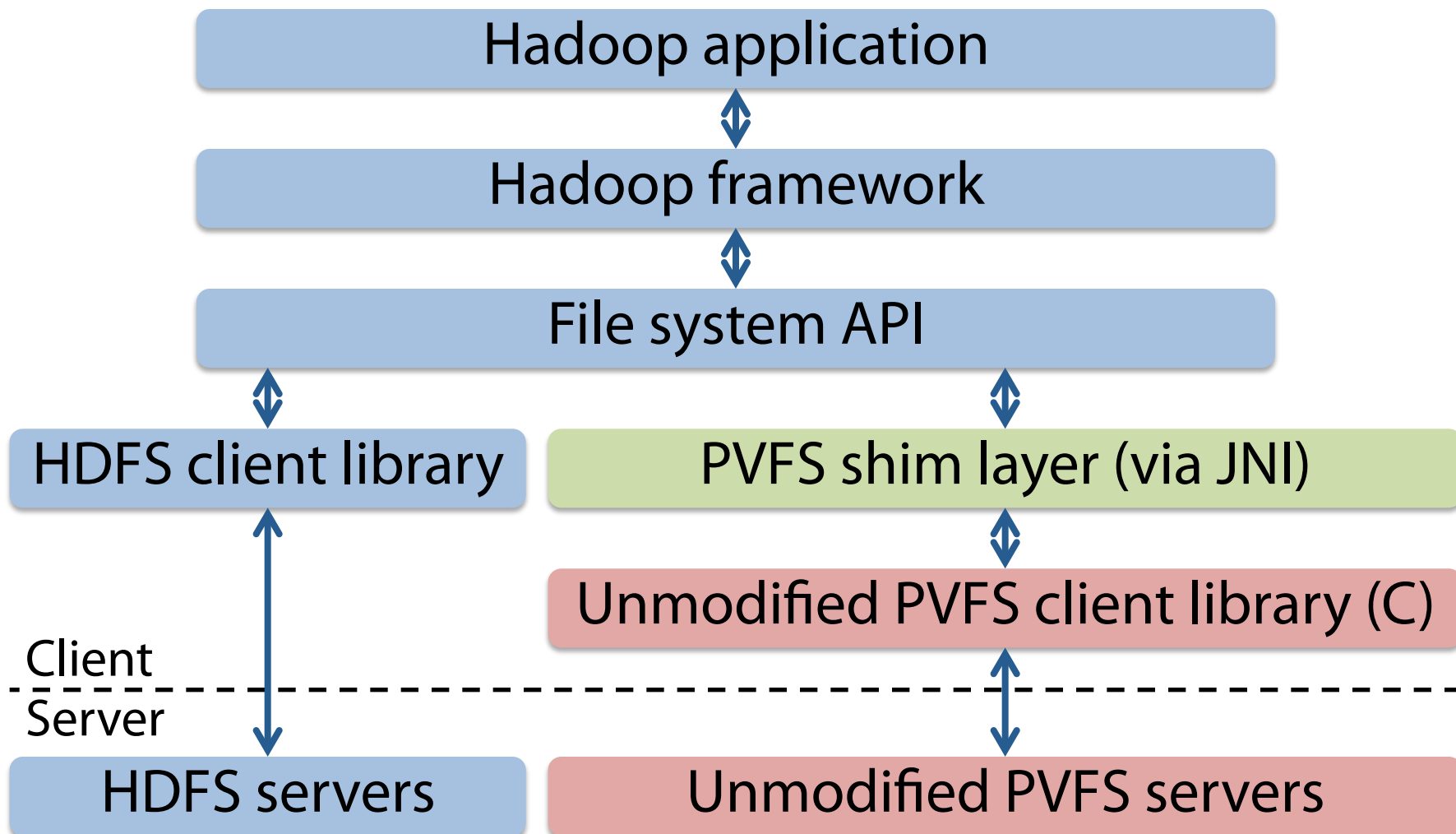
In this work ...

- Can we use an existing parallel file system for Hadoop workloads?
 - **PVFS**: parallel file system
 - **HDFS**: file system designed for Hadoop
- Goal:
 - No modification to Hadoop or PVFS

Outline

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

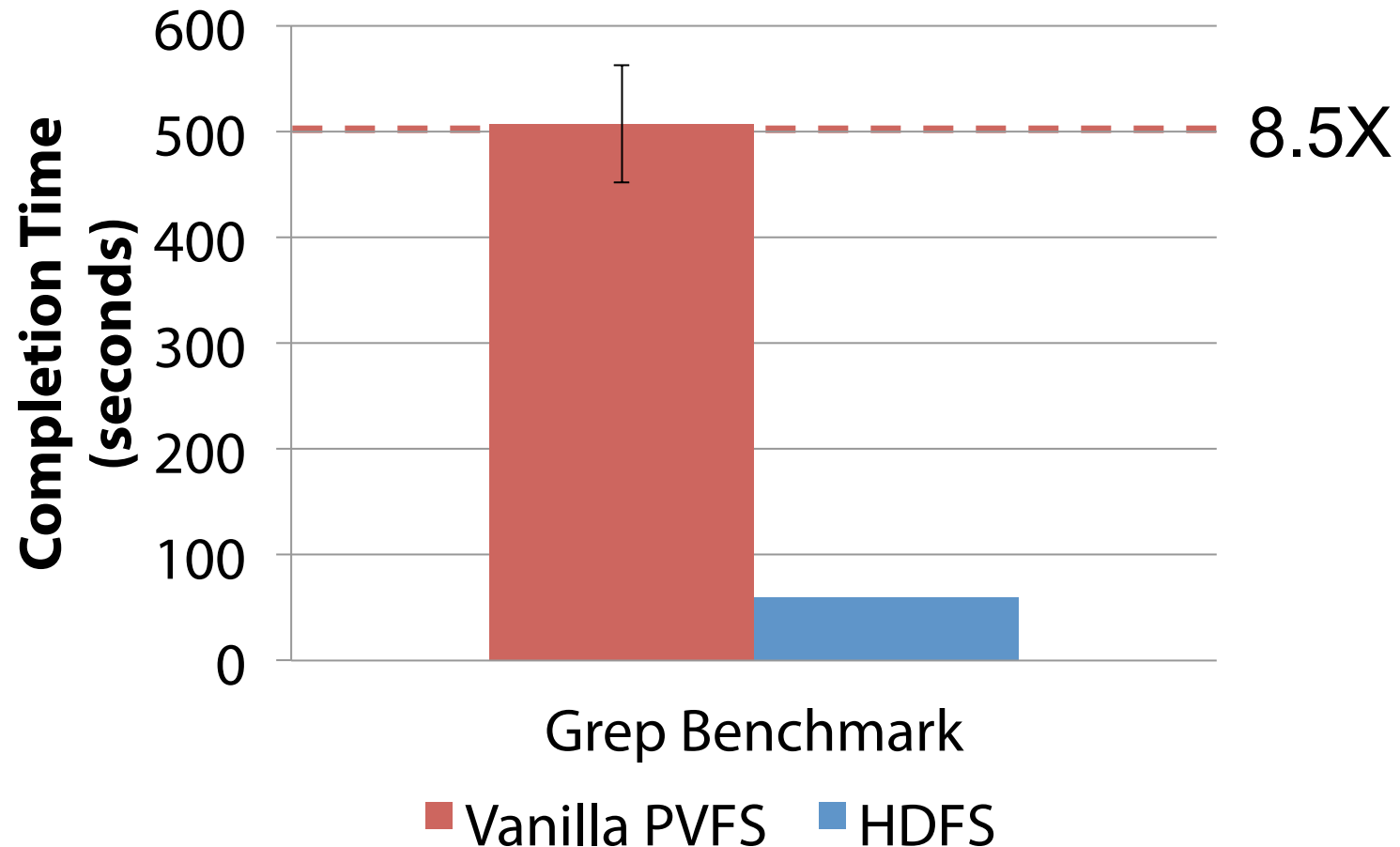
PVFS shim layer under Hadoop



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'

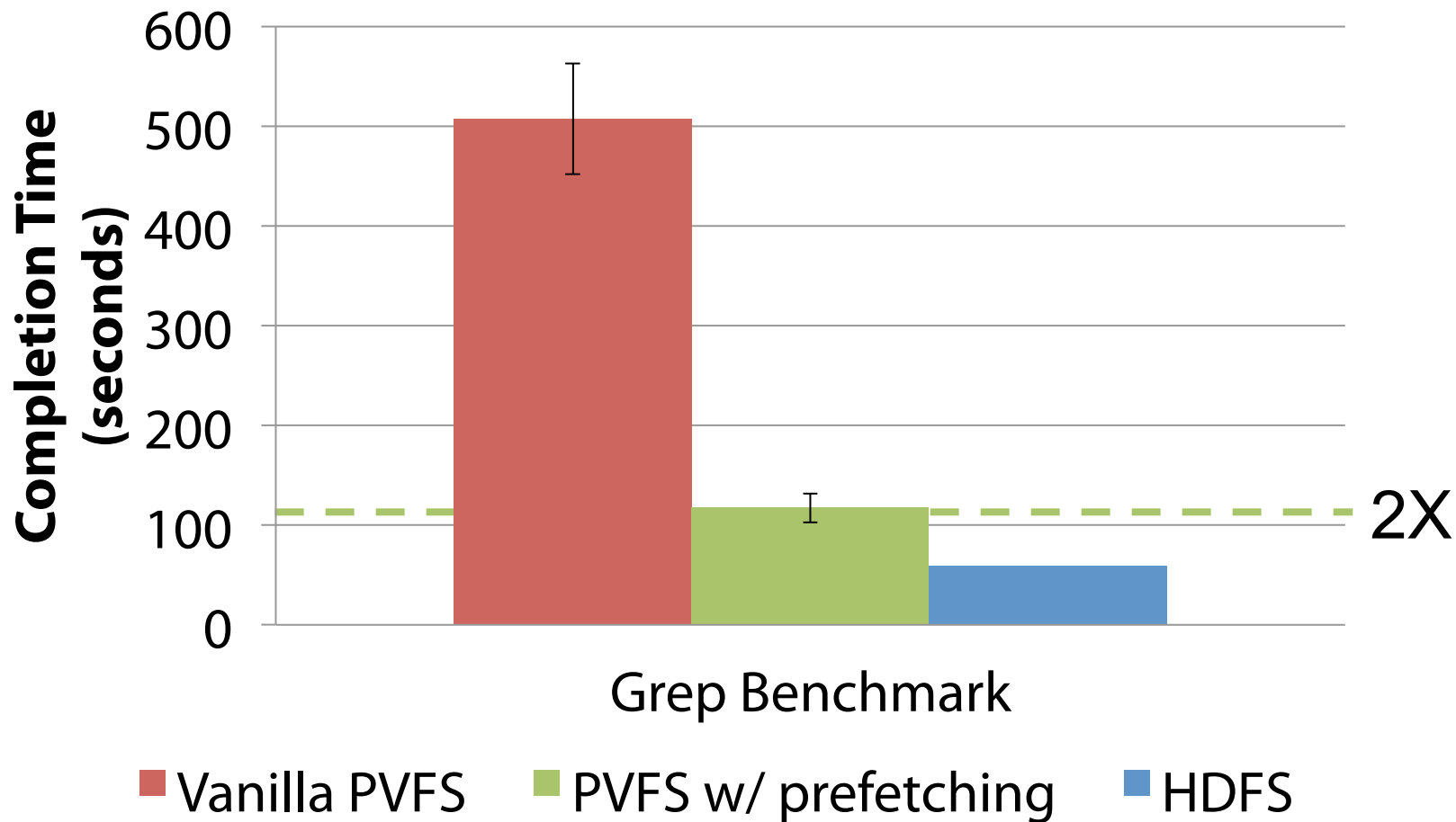
Outline

- Shim layer & vanilla PVFS
- ❖ Extra performance via shim layer
 - Prefetching
 - File layout information
- Replication
- Evaluation

Prefetching is useful for Hadoop

- Hadoop typical workload:
 - Small read (less than 128KB)
 - Sequential through entire chunk
- Prefetching can be implemented in shim layer
 - In Hadoop, file becomes immutable after closed
 - No need for cache coherence mechanism

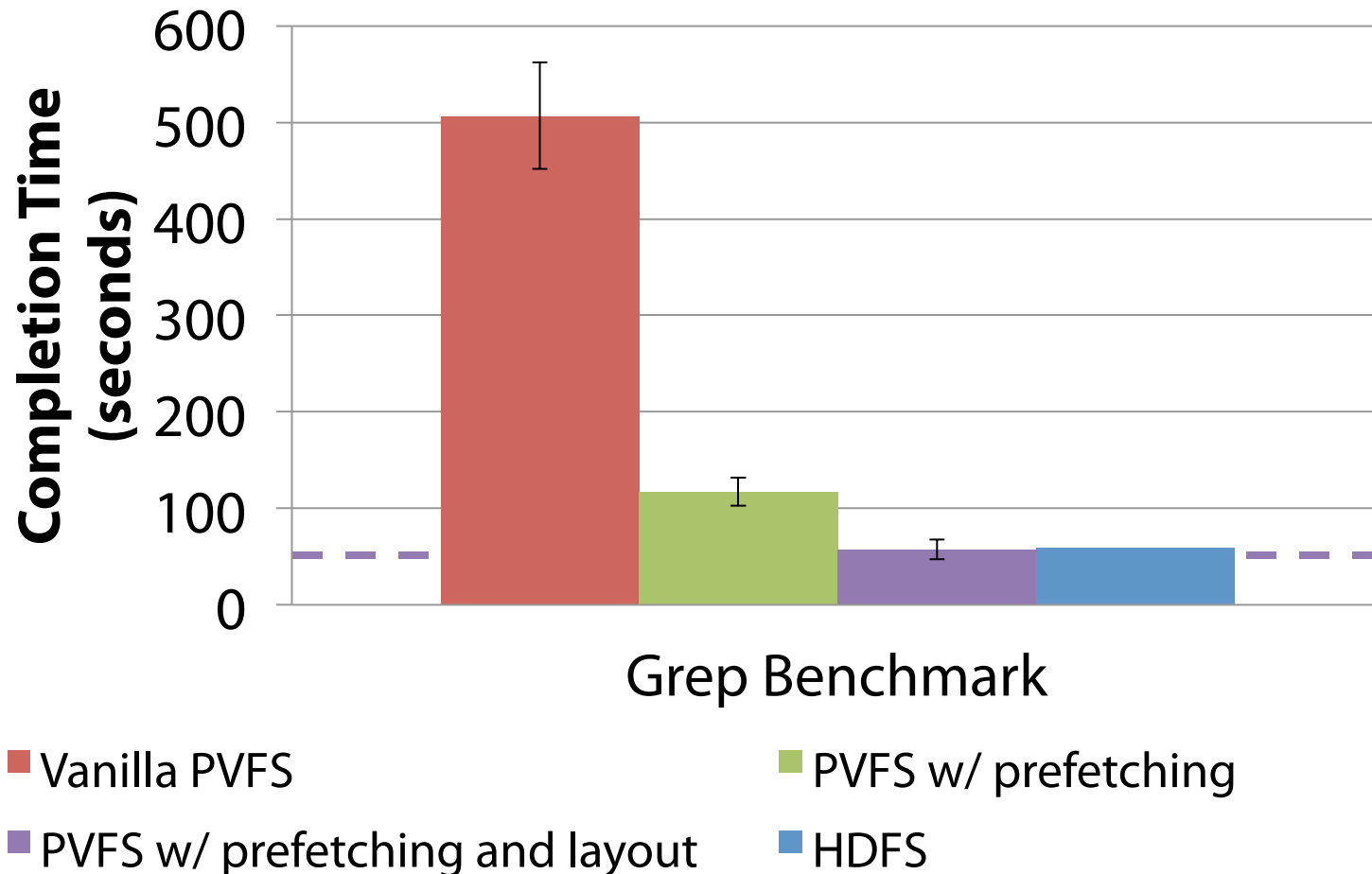
Improving but still 2X slower



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 fast I/O
 - Exposes this information to Hadoop in shim layer

Now, comparable performance



Outline

- Shim layer & vanilla PVFS
- Extra performance via shim layer

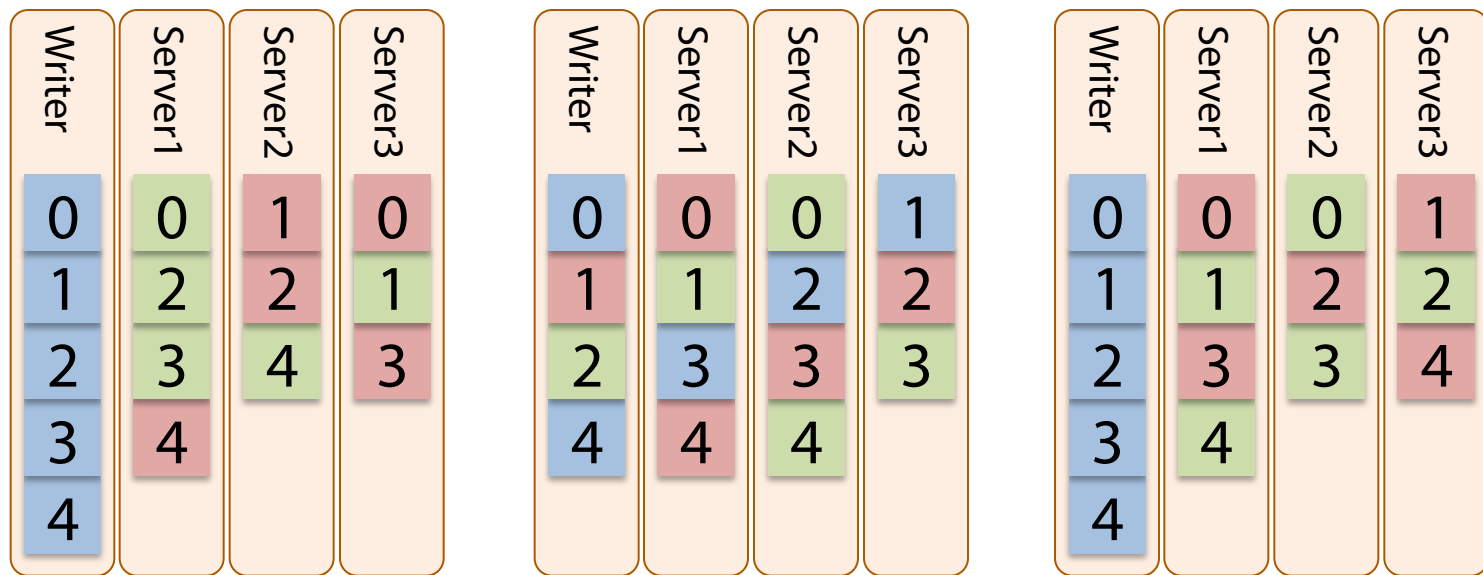
❖ Replication

- Evaluation

Replication vs RAID

- HDFS: software-based replication
 - 1 local copy on writer's disks, 2 copies random
- PVFS: hardware-based RAID
 - Expensive RAID controller
- Can we use PVFS with replication instead?
 - Get rid of RAID controller
 - Implemented in shim layer

Data layout schemes



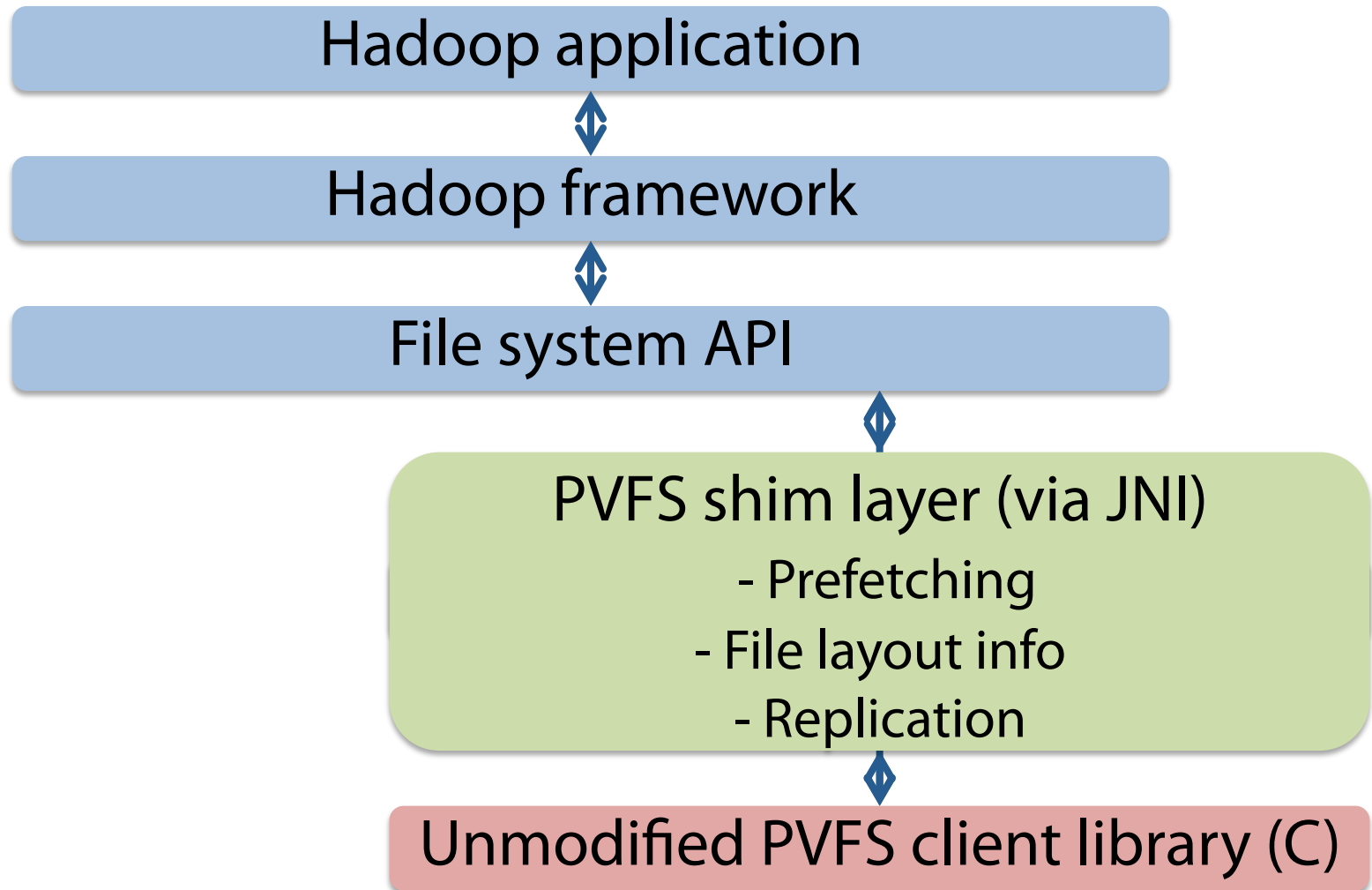
HDFS

PVFS

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
 - Need support from PVFS developers

Shim layer with 3 extensions



Outline

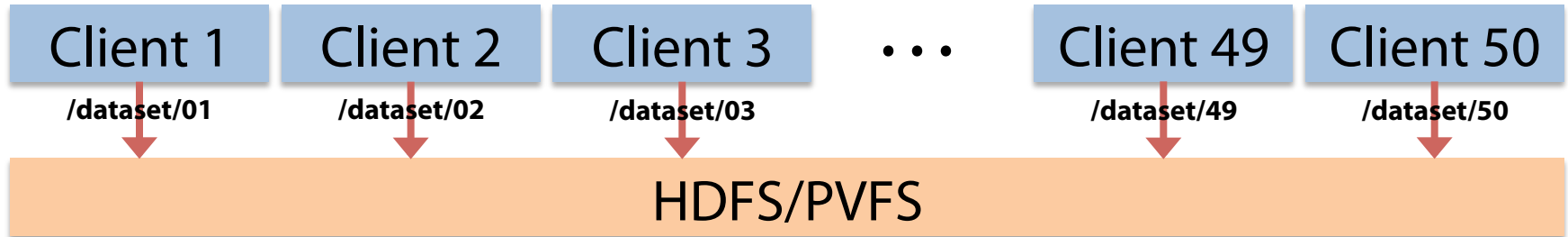
- Shim layer & vanilla PVFS
- Extra performance via shim layer
- Replication
- ❖ Evaluation
 - Micro-benchmark
 - Hadoop benchmark

Micro-benchmark

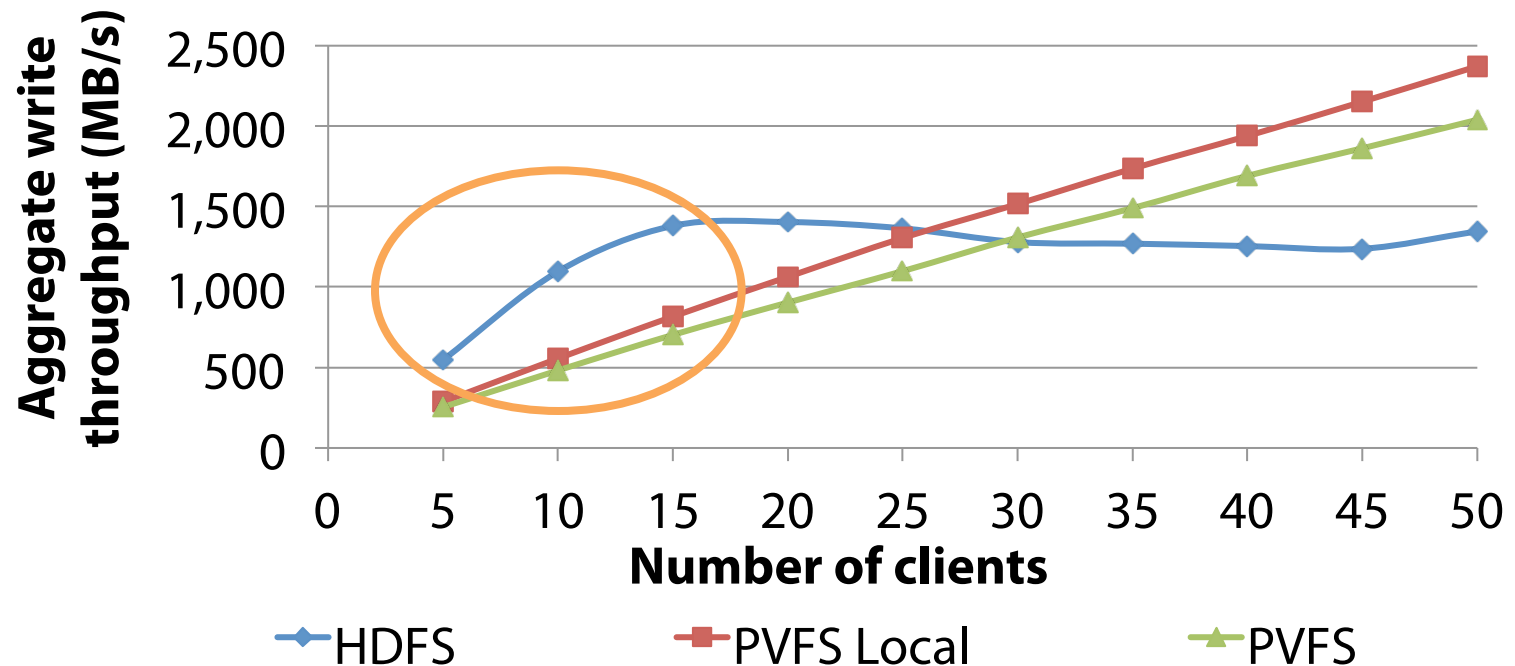
- Cluster configuration
 - 1 master nodes + 50 slave nodes
 - Pentium Xeon 2.8 GHz dual quad core
 - 16 GB Memory
 - One 7200 rpm SATA 160 GB
 - 10 Gigabit Ethernet
- Uses file system API directly without Hadoop

Write benchmark

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

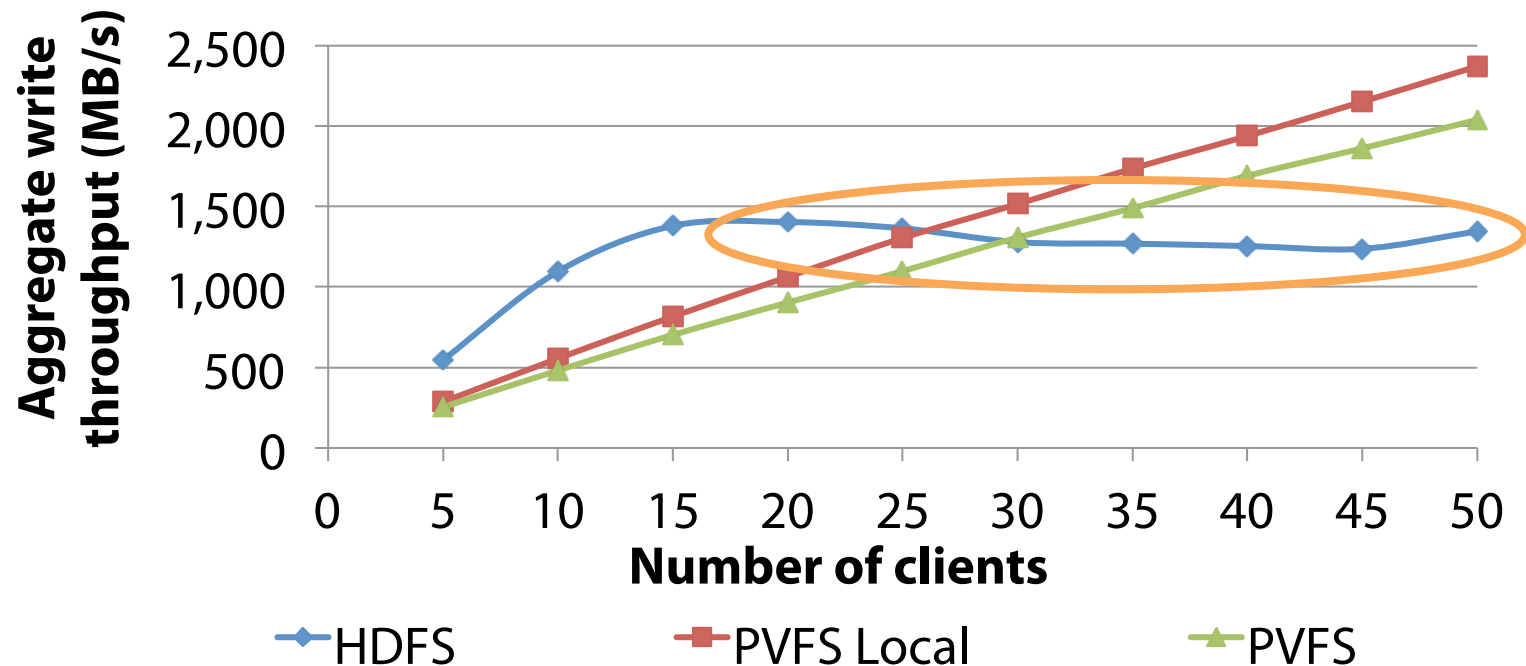


HDFS utilizes idle resource better



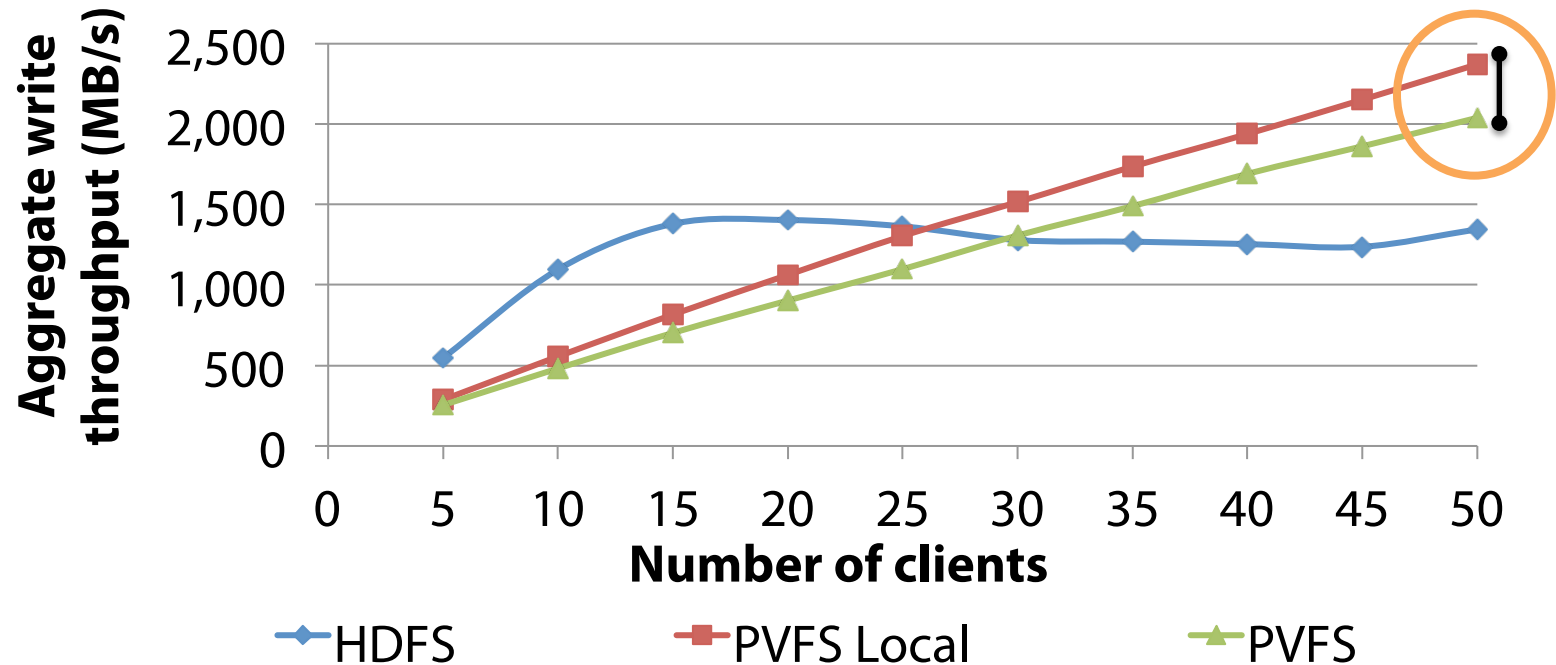
- HDFS pipelined replication improves parallelism
 - Better than client driven replication in PVFS

HDFS is surprisingly tied to disk performance



- Limited by synchronous file creation
 - New file for each chunk in HDFS
 - Head-of-line blocking while flushing write-back buffer

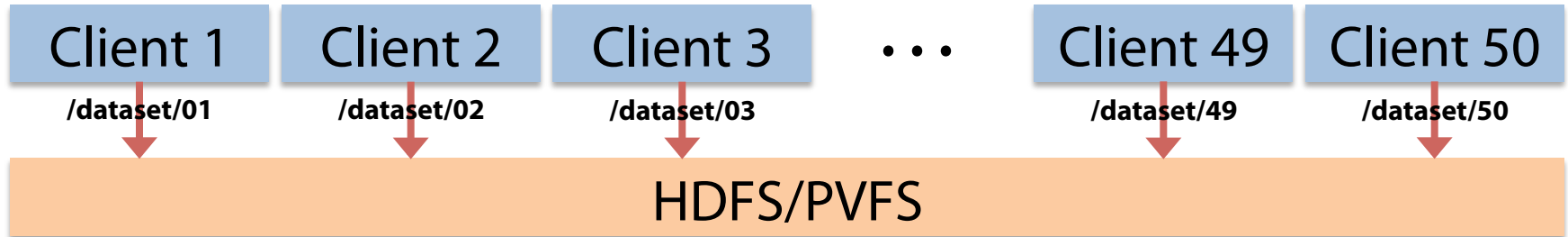
PVFS local improves write tput by 20%



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

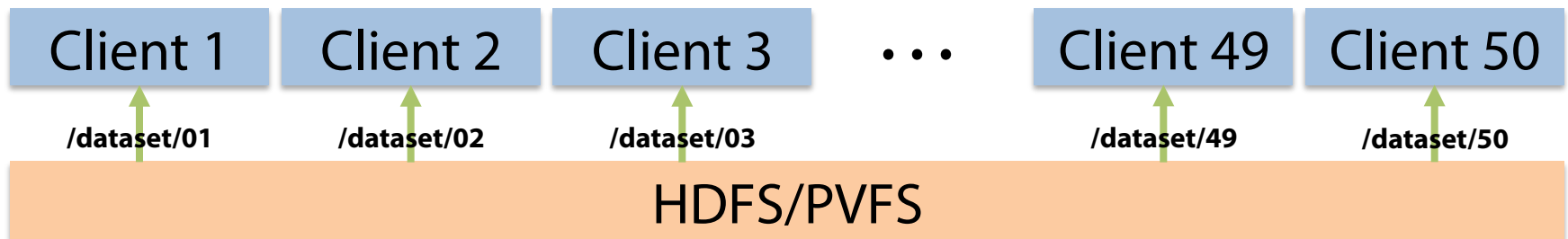
Read benchmark

1. 50 clients write to 50 data servers

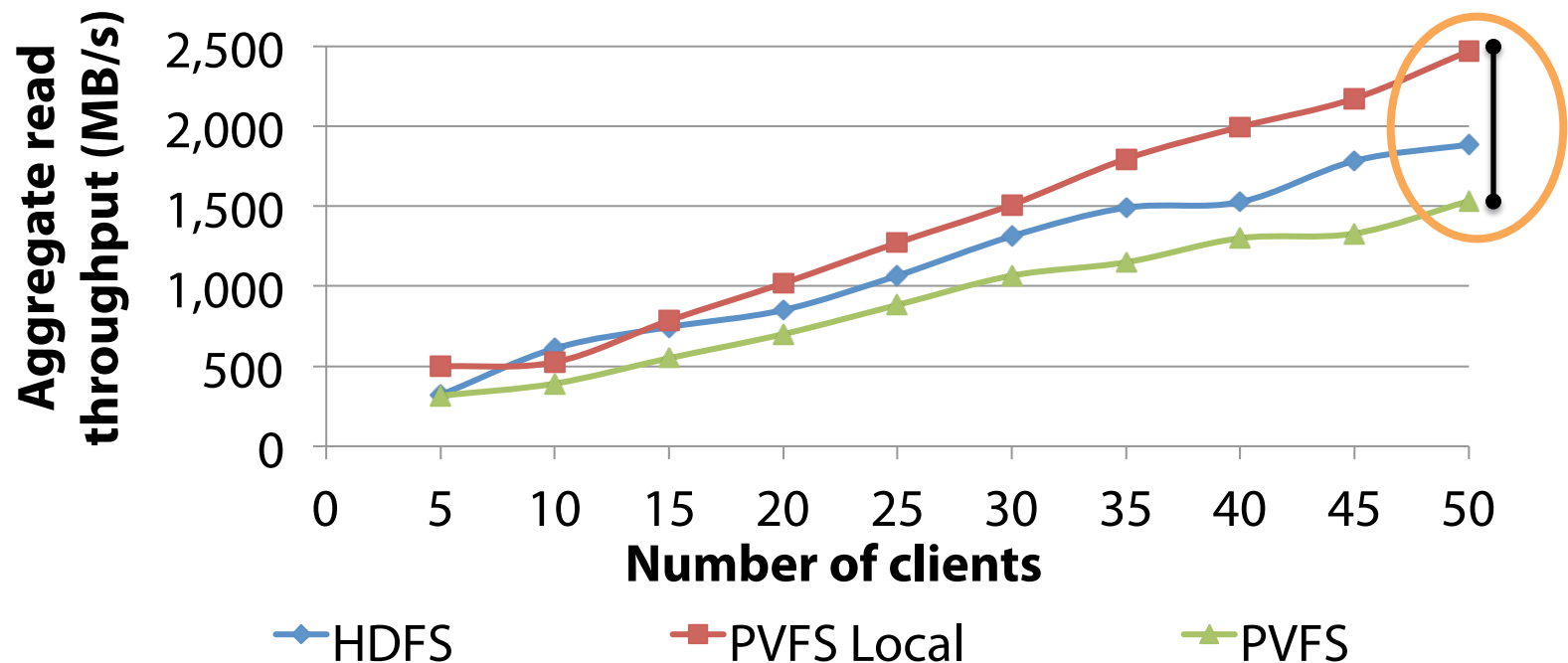


2. 5, 10, ..., or 50 clients read back

- The same file each client creates



Local copy improves read tput by 60%



- Larger improvement than write
 - No network traffic at all

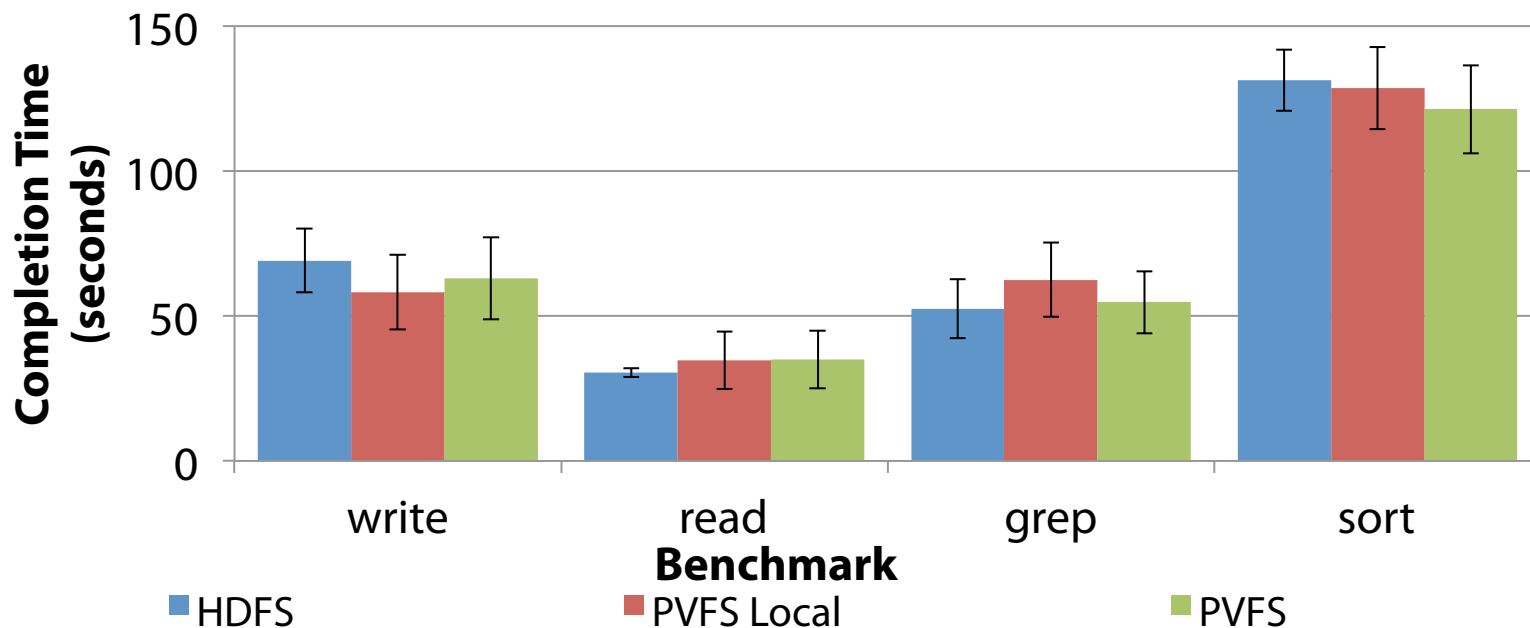
Hadoop benchmark setting

- 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
 - 10 Gigabit Ethernet
- Use Hadoop for map-reduce processing

Hadoop benchmark

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

Effect of Hadoop job scheduler

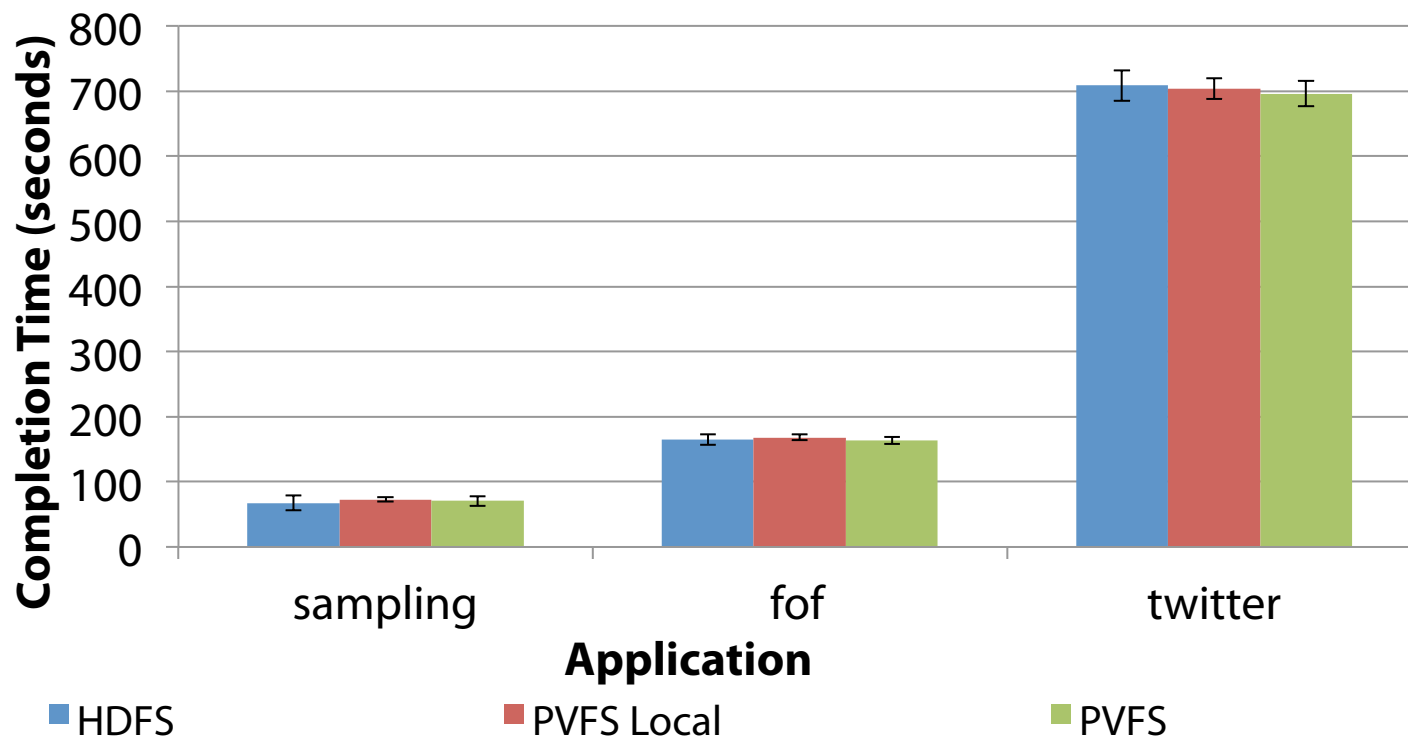


- Masks inefficiencies of PVFS w/o local copy
 - With collocation, most reads become local reads
 - Blocks can be processed out of order

Scientific Hadoop applications

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

Results



- PVFS performance is comparable to HDFS

Summary

- PVFS can be tuned to deliver comparable performance for Hadoop applications
 - Simple shim layer in Hadoop
 - Local copy optimization
- Little modification necessary
 - Only for local copy optimization