YCSB++ benchmarking tool

Performance debugging advanced features of scalable table stores

Swapnil Patil

M. Polte, W. Tantisiriroj, K. Ren, L.Xiao,

J. Lopez, G.Gibson, A. Fuchs *, B. Rinaldi *

Carnegie Mellon University

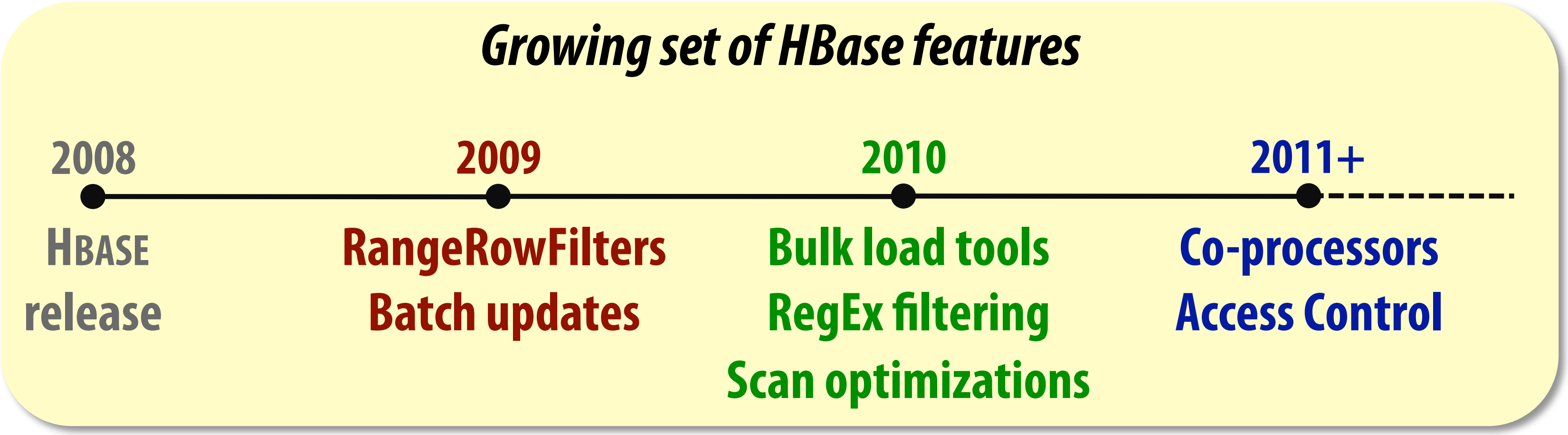
* National Security Agency

Carnegie Mellon
Parallel Data Laboratory

Importance of scalable table stores

- For data processing and analysis
- For systems services (e.g., metadata in Colossus)

Growing complexity of table stores



Growing set of HBase features

2008 HBASE release

2009 RangeRowFilters, Batch updates

2010 Bulk load tools, RegEx filtering, Scan optimizations

2011+ Co-processors, Access Control

Simple, lightweight → complex, feature-rich stores

- ▲ Supports a broader range of applications
- ▼ Hard to debug performance issue and complex component interactions

State of table store benchmarking

YCSB: Yahoo Cloud Serving Benchmark[Cooper2010]

- ▲ Modular design to test different table stores
- ▼ Great for CRUD (create-read-update-delete) benchmarking, but not for sophisticated features

Need richer tools for understanding advanced features in table stores . . .

This talk: YCSB++ tool

New Extensions in YCSB++

Distributed, coordinated and multi-phase testing

Fine-grained, correlated monitoring using Otus*[Ren2011]*

Table store features tested by YCSB++

Batch writing • Table pre-splitting • Bulk loading

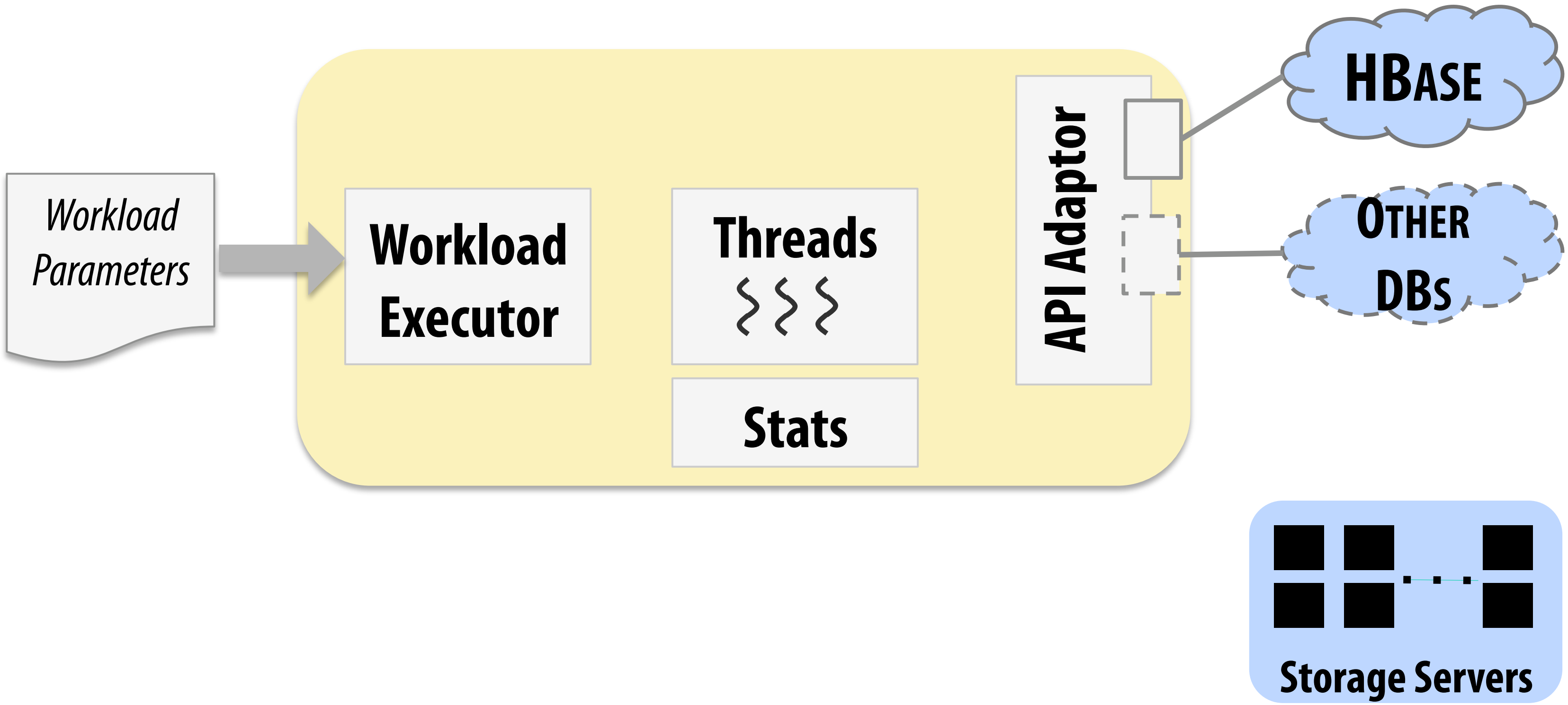
Weak consistency • Server-side filtering • Fine-grained security

Tool released at http://www.pdl.cmu.edu/ycsb++

Talk Outline

- Motivation
- YCSB++ architecture
- Illustrative examples of using YCSB++
- Summary and ongoing work

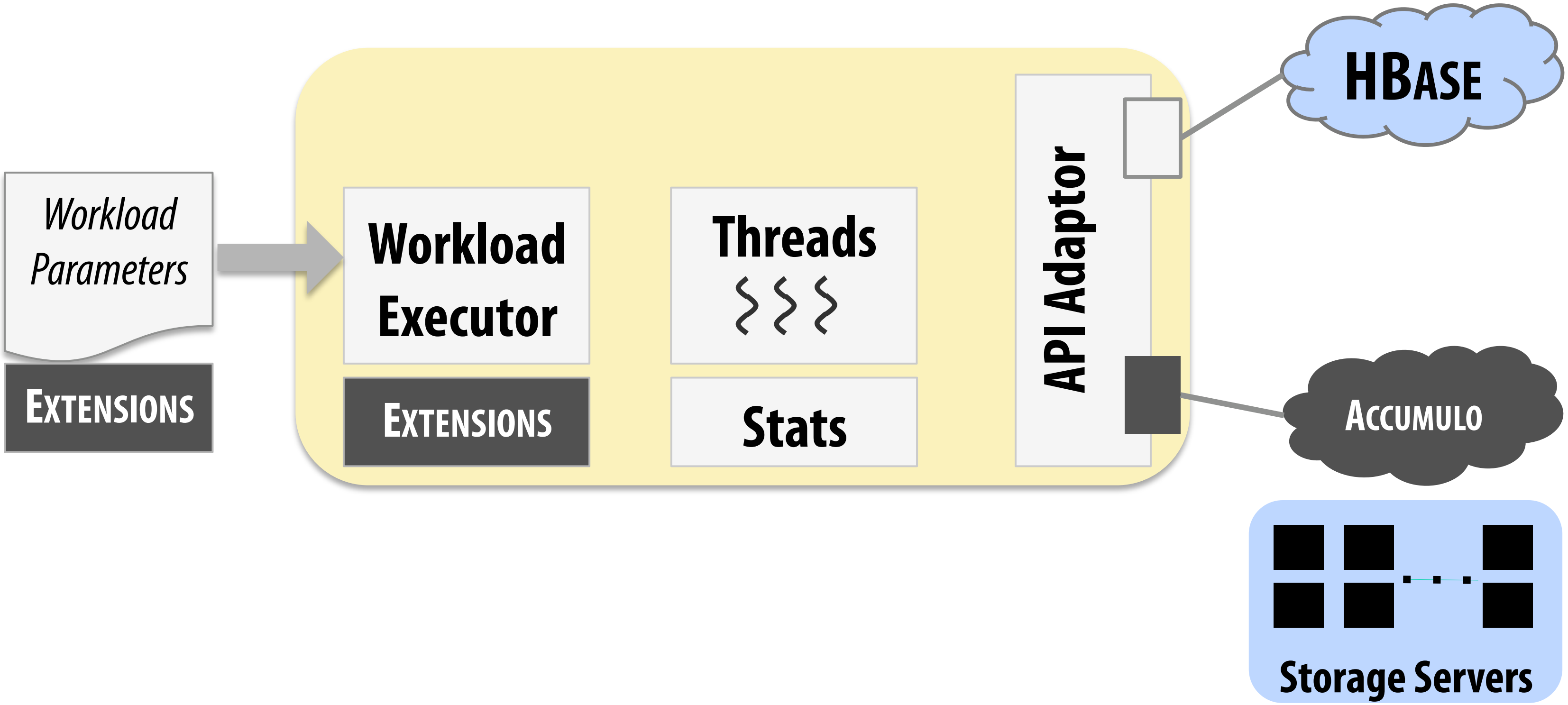
Original YCSB framework



Configurable workload generation to test stores

- API adaptor converts **`read(K)`** to **`hbase_get(K)`**

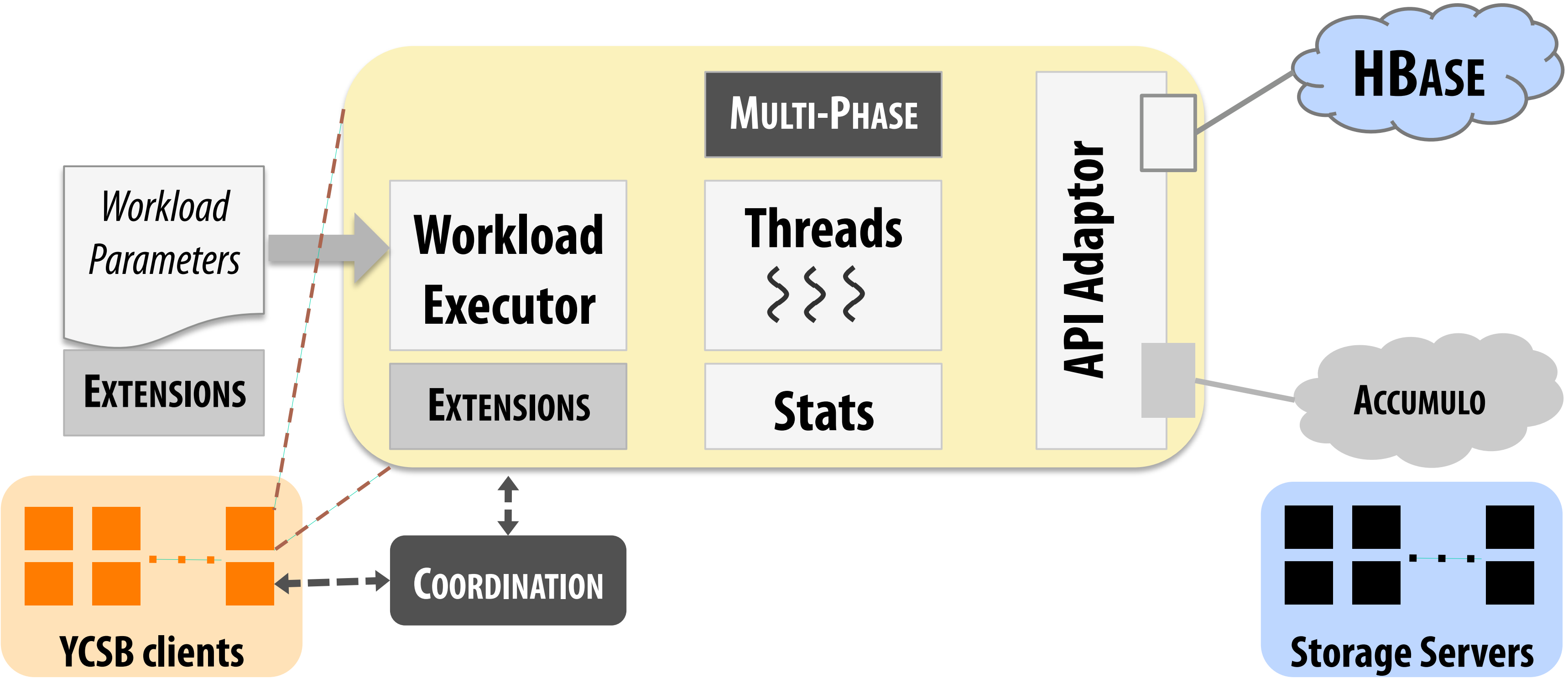
YCSB++ supports new table store



New DB adaptor for Apache Accumulo table store

- New parameters and workload executor extensions

Coordinated & multi-phase tests



ZooKeeper-based coordination & synchronization

- Enables heavy workloads and asymmetric testing

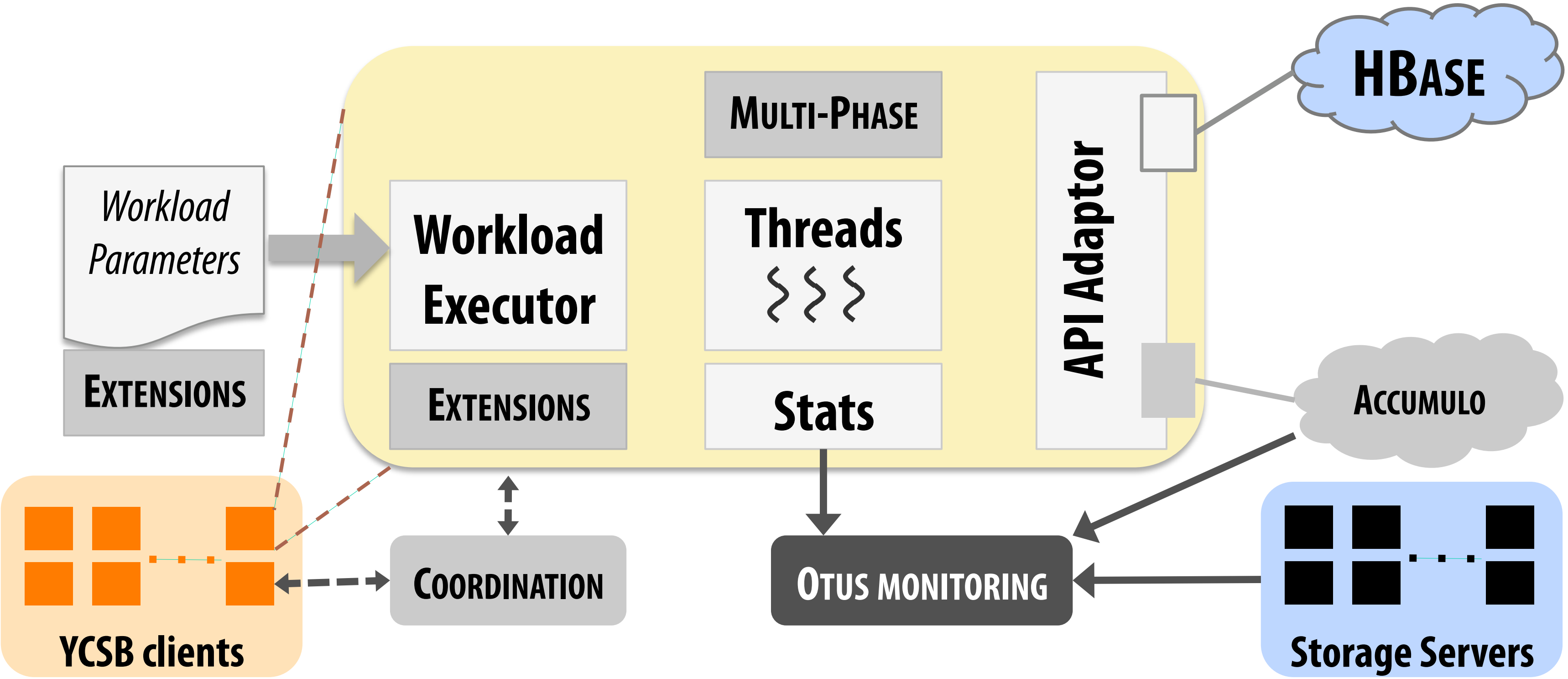
Carnegie Mellon
Parallel Data Laboratory

Coordinated & multi-phase tests

- Distributed, multi-client tests using YCSB++
  - Allows clients to co-ordinate their test actions
  - Rely on shared data structures in ZooKeeper
  - Useful for testing weak data consistency

- Multi-phase tests in YCSB++
  - Can construct tests comprising of different phases
  - Built on ZooKeeper-based barrier-synchronization
  - Used for understanding high-speed ingest features

Collective monitoring in YCSB++



Fine-grained resource monitoring using Otus[Ren2011]

- Collects from YCSB, table stores, HDFS and `/proc`

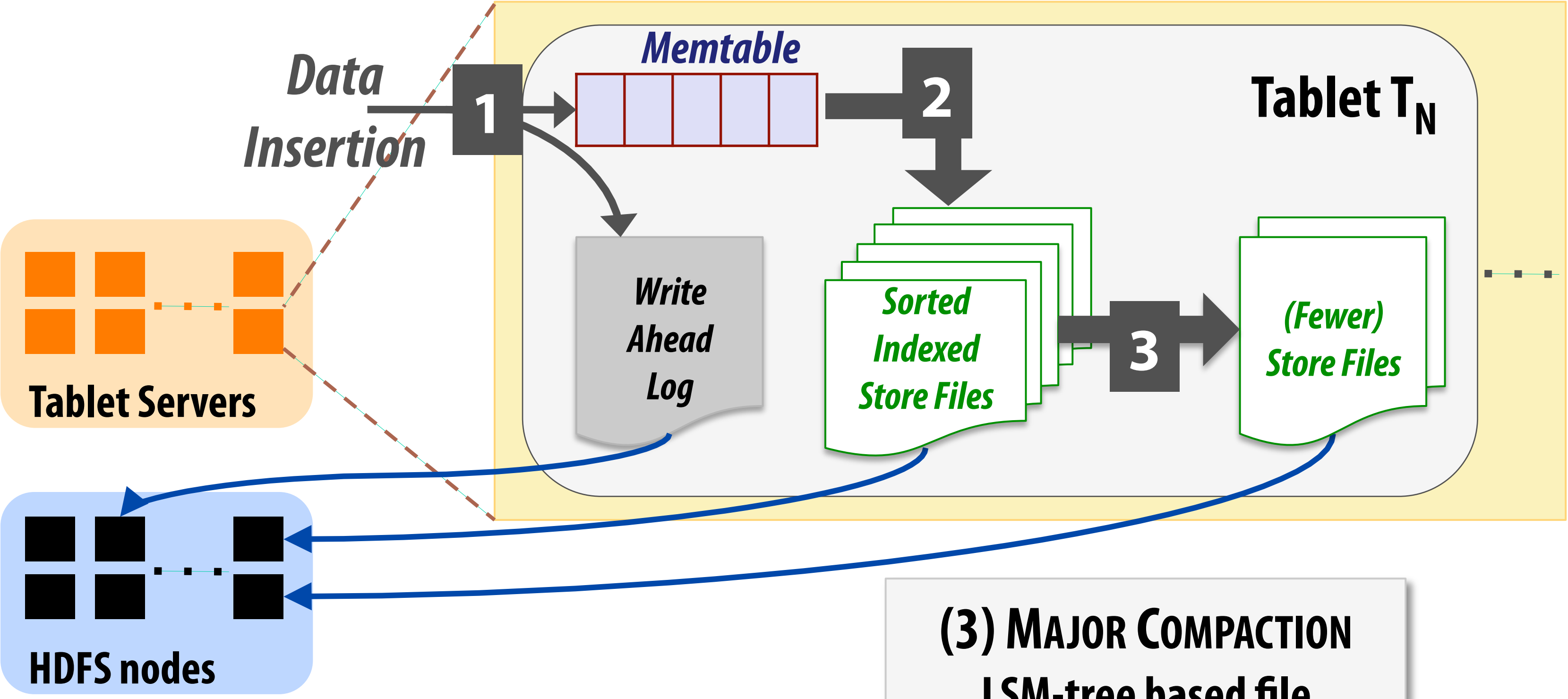
Talk Outline

- Motivation
- YCSB++ architecture
- Illustrative examples of using YCSB++
  - Case study: HBase and Accumulo
  - Both are Bigtable-like table stores
- Summary and ongoing work

Primer on Bigtable-like stores

(1) Incoming mutation logged in memory (unsorted order)

(2) MINOR COMPACTION
Memtables written to sorted, indexed store files in HDFS



(3) MAJOR COMPACTION
LSM-tree based file merging (in background)

Accumulo table store

- Started at NSA; now an Apache project
  - Built for high-speed ingest and scan workloads
  - http://incubator.apache.org/projects/accumulo.html
- New features in Accumulo
  - Iterator framework for user-specified programs placed in different stages of DB pipeline
    - E.g., Supports joins and stream processing
  - Also provides fine-grained cell-level access control

Before I talk about examples …

YCSB++ provides

- Abstractions to construct distributed, parallel tests
  - Has in-built tests that use these abstractions
- Monitoring that collects and correlates system (store/FS/OS) state with observed performance

YCSB++ does not provide

- Root cause diagnosis of performance problems
  - Merely points you to where you should look …

Features Tested by YCSB++

Table bulk loading

Batch writing

Weak consistency

Table pre-splitting

Server-side filtering

Access control

Illustrative Example

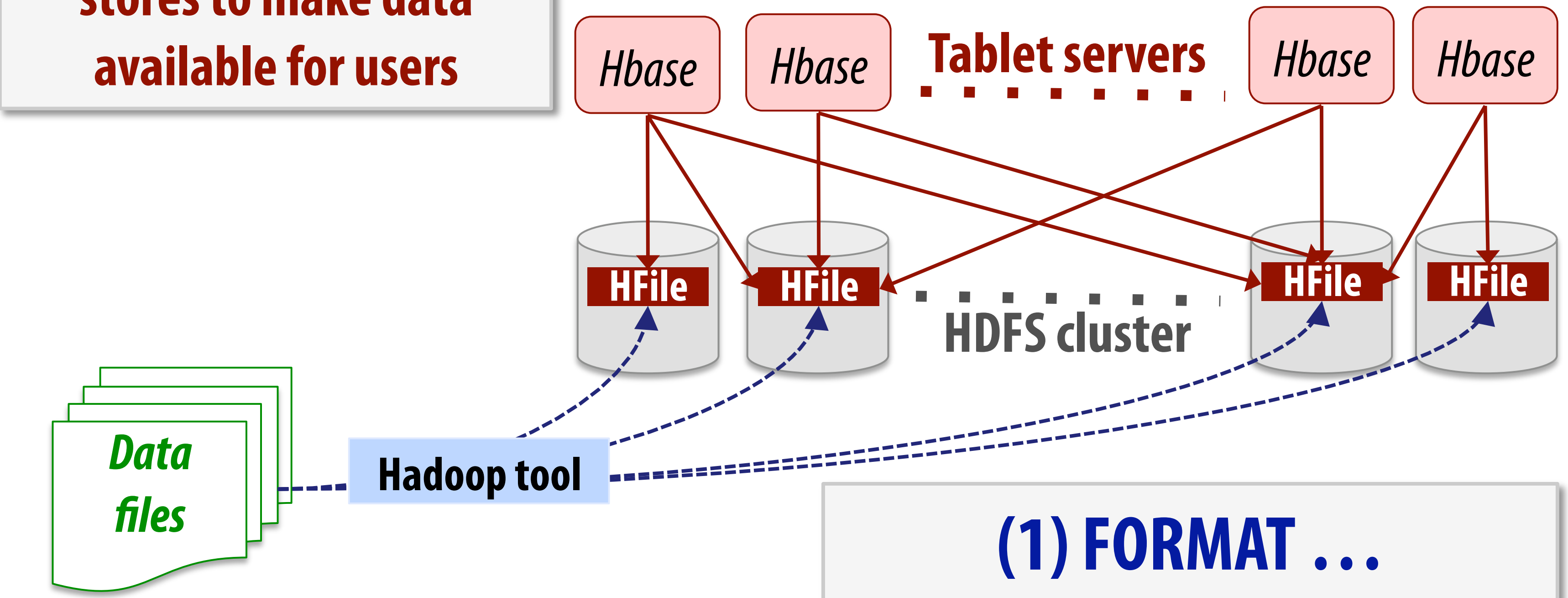
Table bulk loading

- ▲ High-speed ingestion through minimal data migration
- ▼ Need careful tuning and configuration [Sasha2002]

Table bulk loading in action

(2) IMPORT …

… store files into table stores to make data available for users



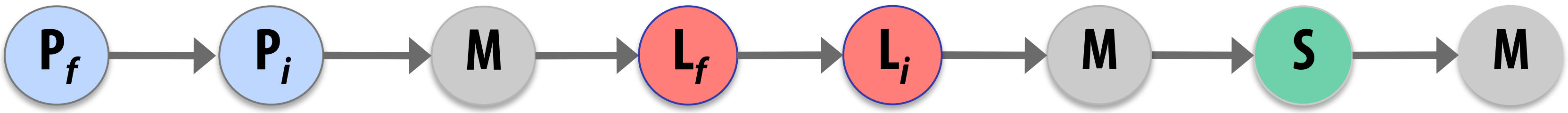
(1) FORMAT …

… existing data files to store-file specific format using Hadoop

8-phase bulk load test in YCSB++

Measurement phase

- Light mix of Read/Update operations
- Interleaved to study performance over time



Pre-load data

- Insert 6M rows in empty table

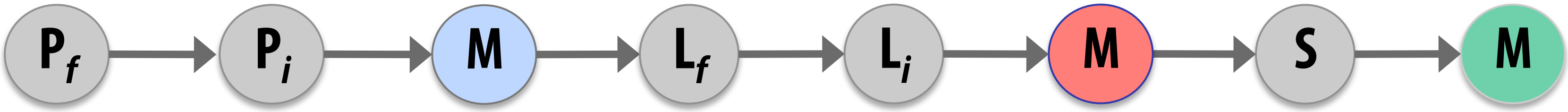
Load data

- Load 48M rows in existing table

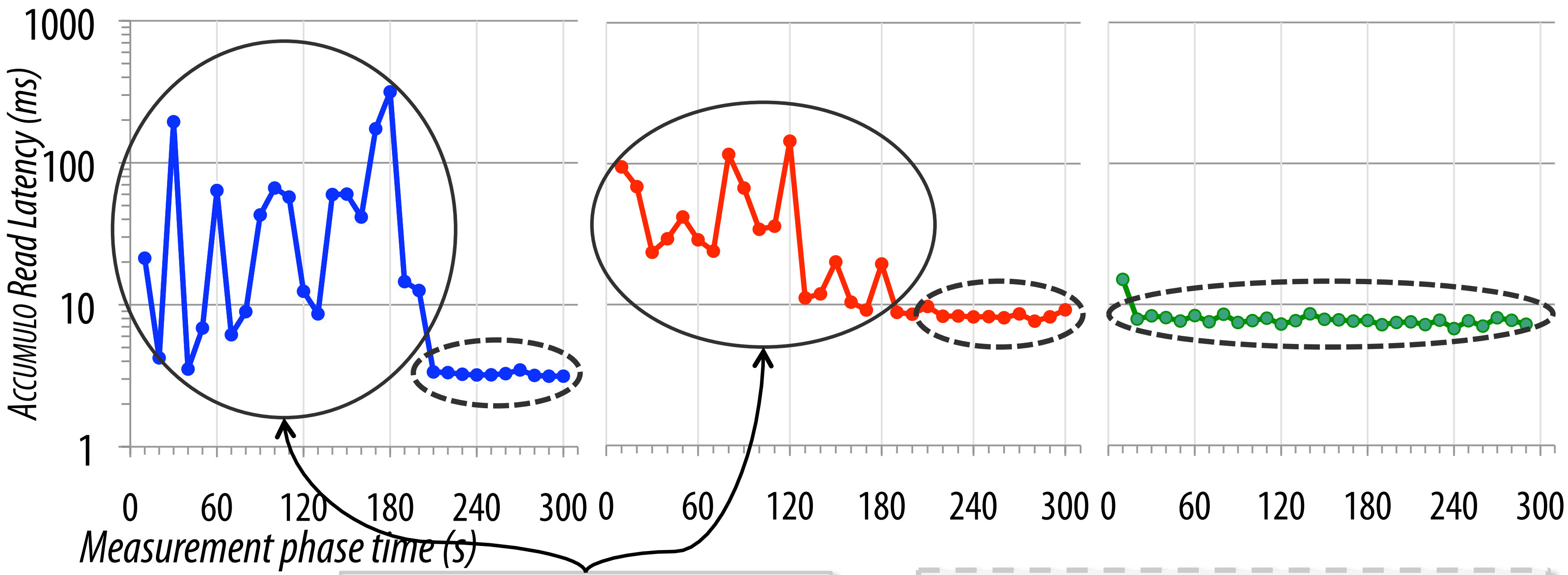
Sleep

- Let servers finish balancing work

Multi-phase tests show variation



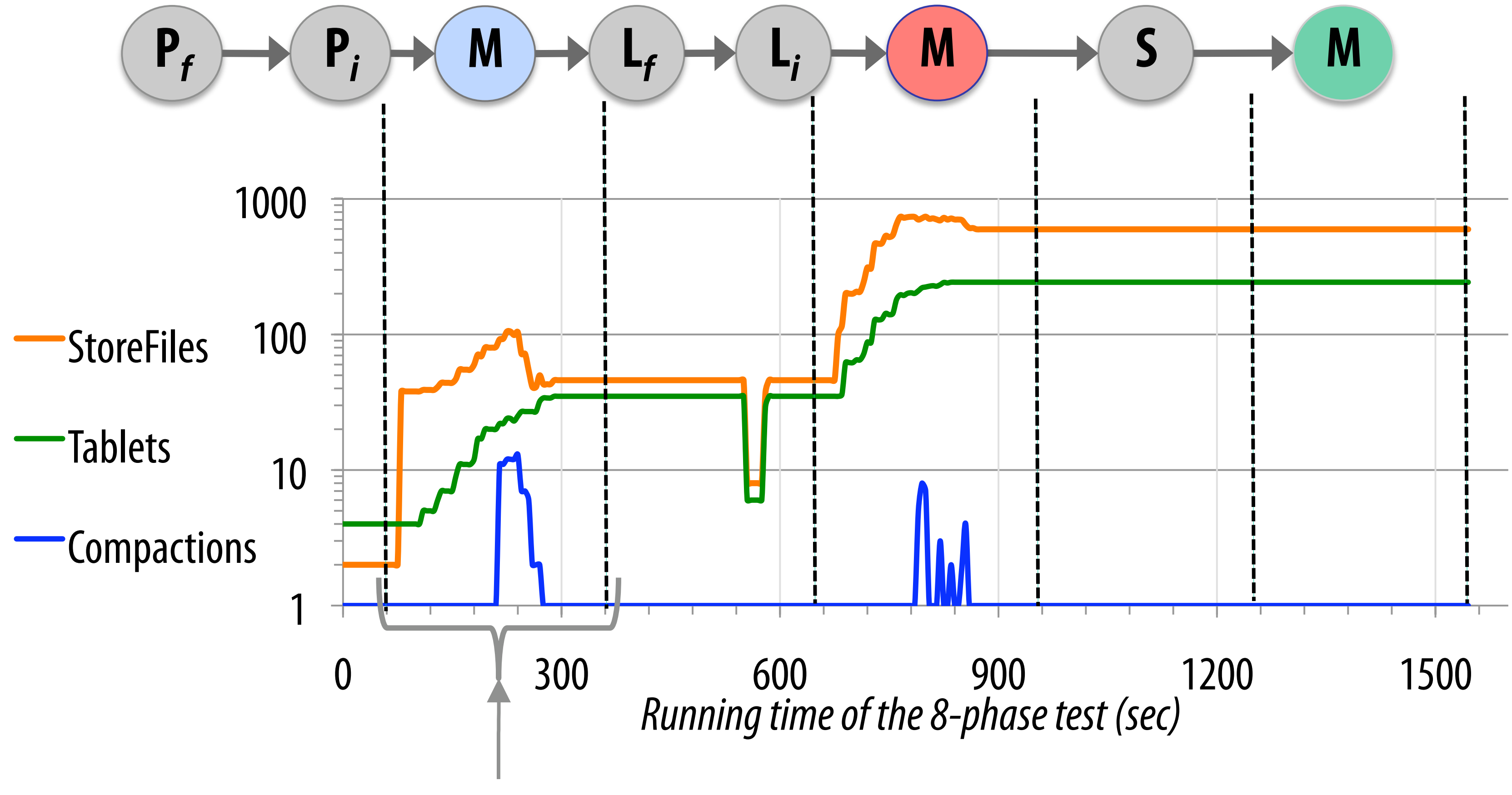
$P_f \rightarrow P_i \rightarrow M \rightarrow L_f \rightarrow L_i \rightarrow M \rightarrow S \rightarrow M$



10x latency variation; lasts for a long time!

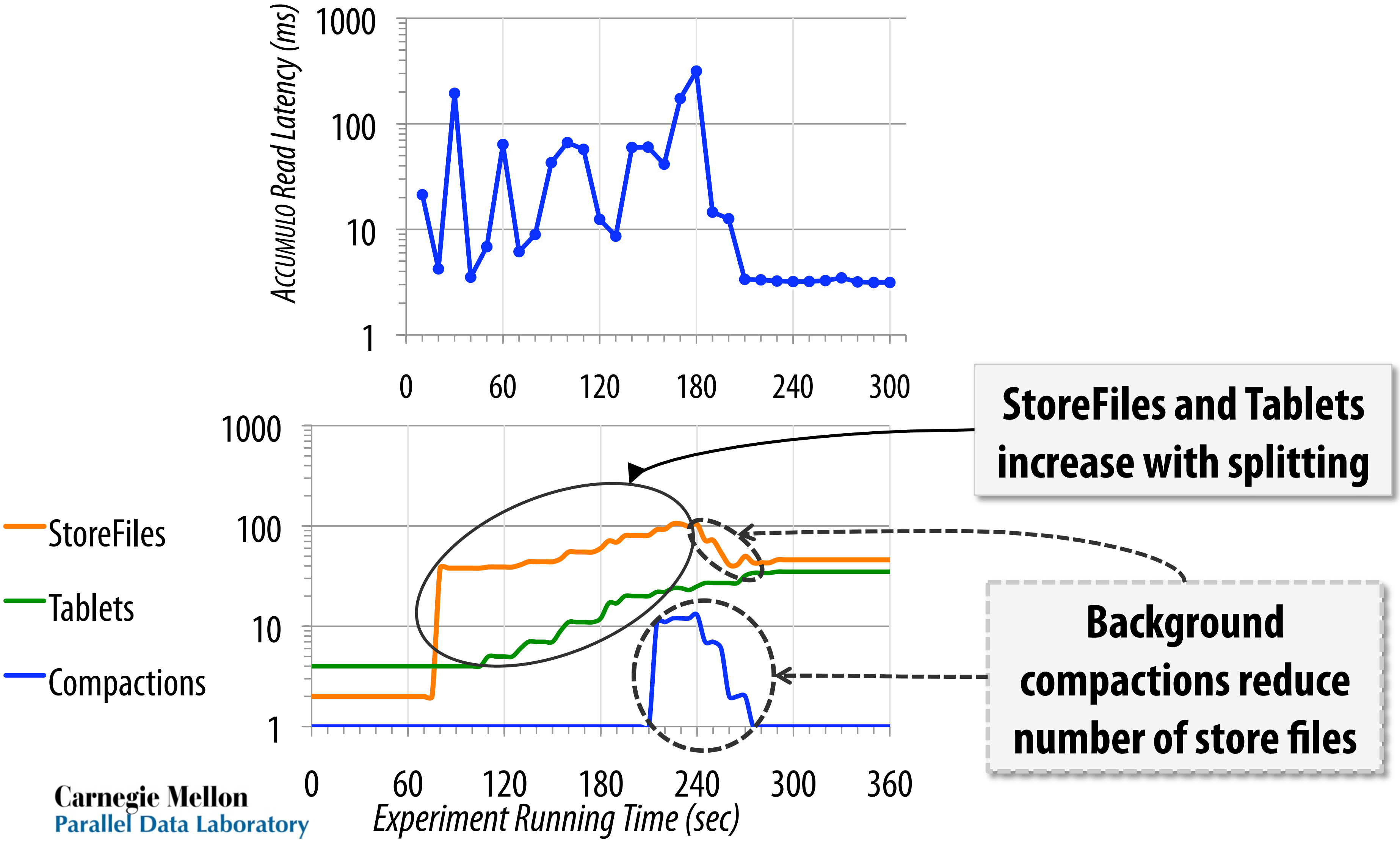
Uniformly low latency after store is steady (no inserts)

Monitoring rebalancing at servers



Let's take a closer look at correlating performance with server-side state

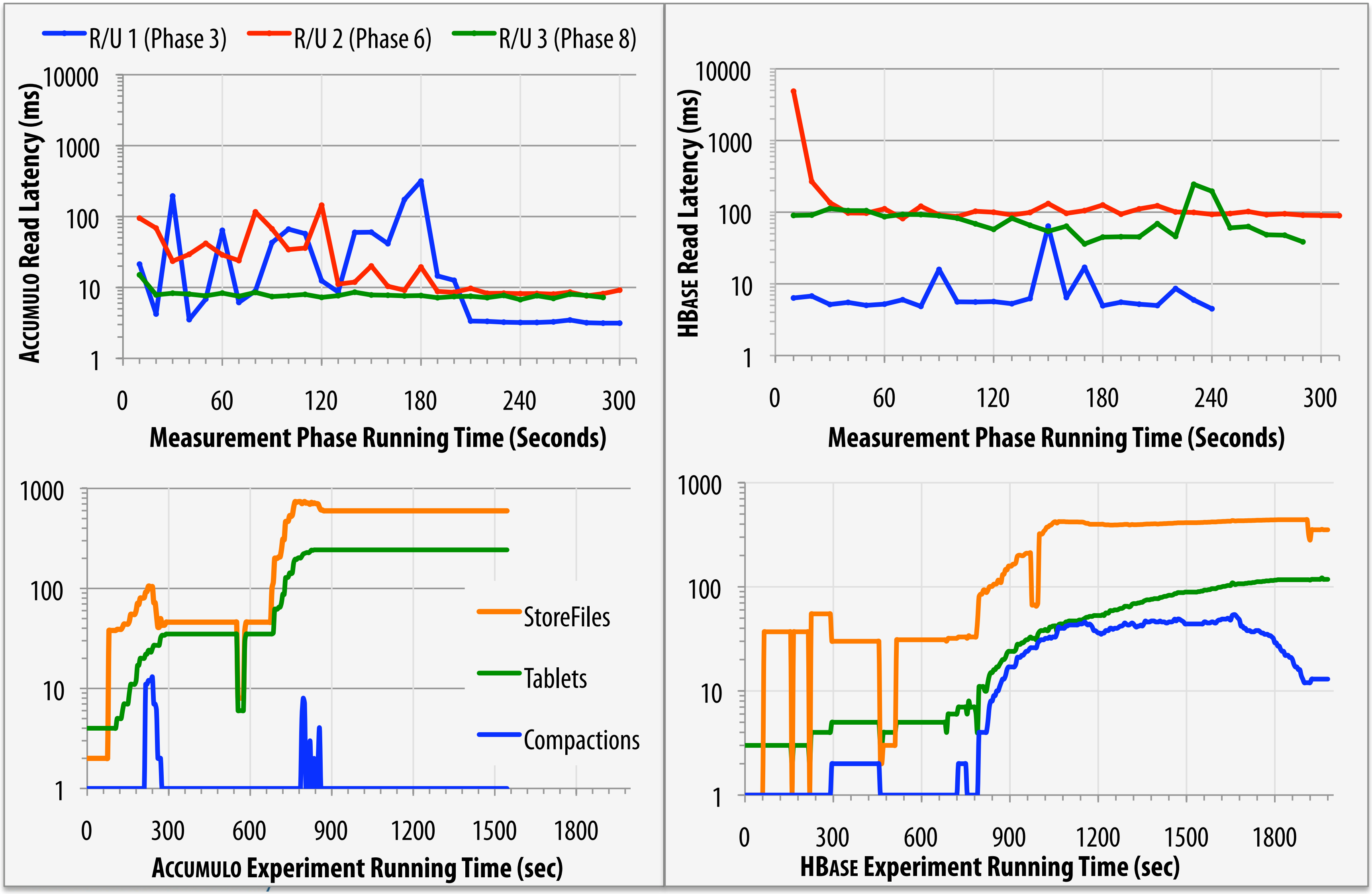
Effect of server-side work on latency



StoreFiles and Tablets increase with splitting

Background compactions reduce number of store files

YCSB++ helps study different policies



Features Tested by YCSB++

Table bulk loading

Batch writing

Weak consistency

Table pre-splitting

Server-side filtering

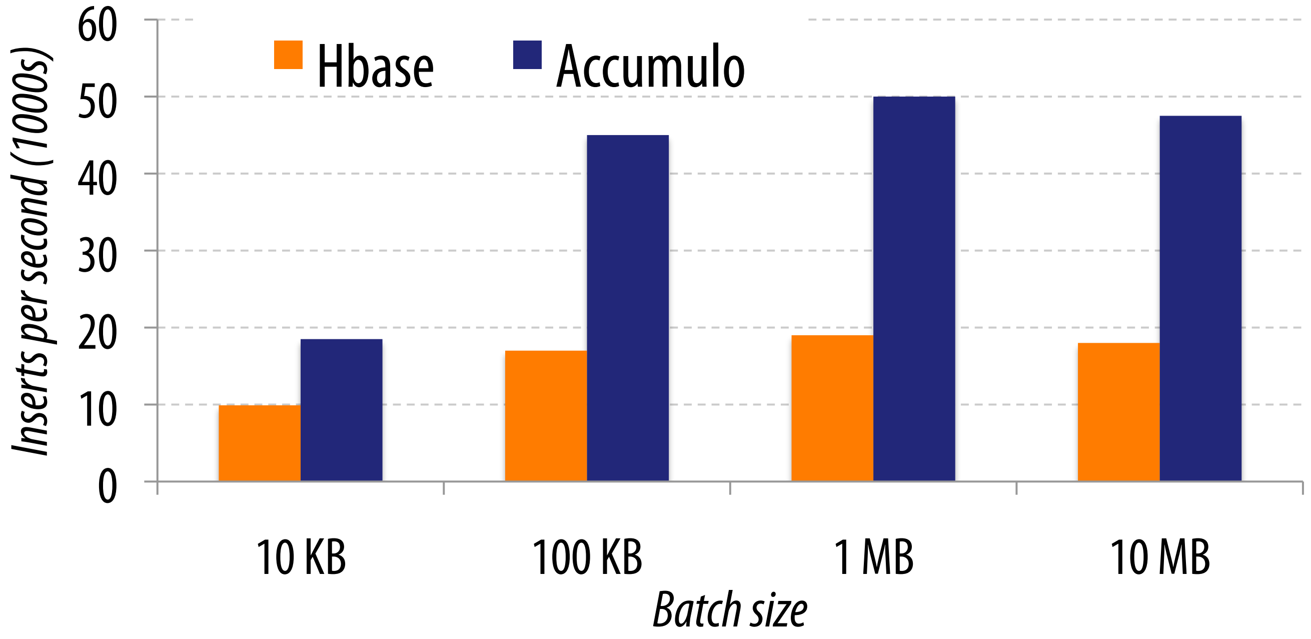
Access control

Illustrative Example

Batching writes at clients

- ▲ Improves insert throughput and latency
- ▼ Newly inserted data may not be immediately visible to others

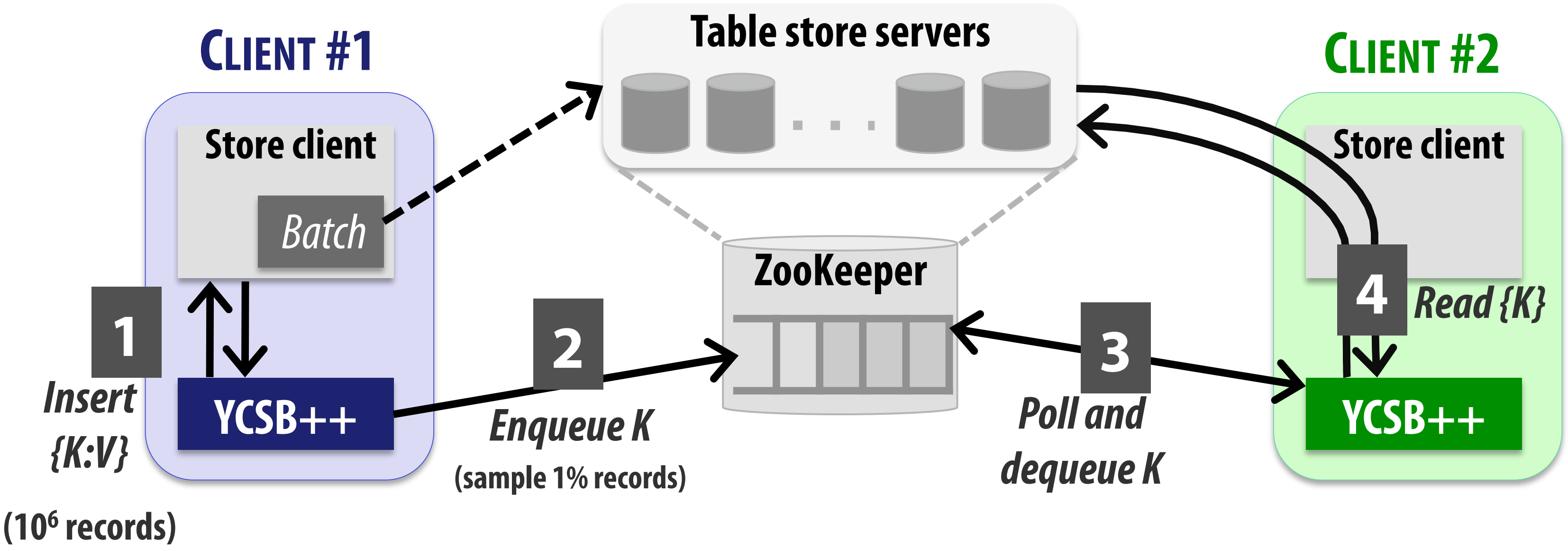
Batching improves throughput



6 clients create 9 million 1-KB records on 6 servers

- Small batches: high client CPU utilization limits work
- Large batches: servers are saturated, limits benefit

Weak consistency test in YCSB++



ZooKeeper-based multi-client coordination

- Clients use a shared producer-consumer queue to communicate keys to be tested

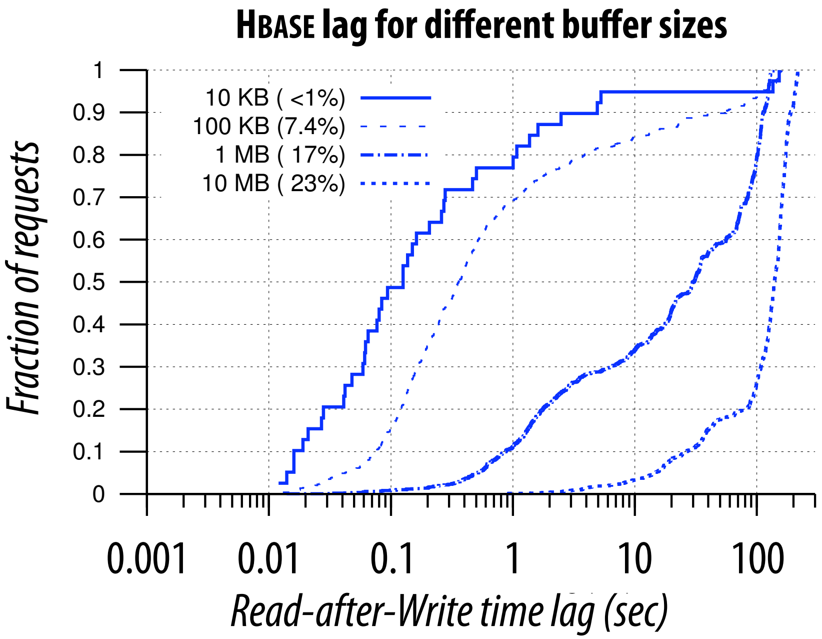
Carnegie Mellon
Parallel Data Laboratory

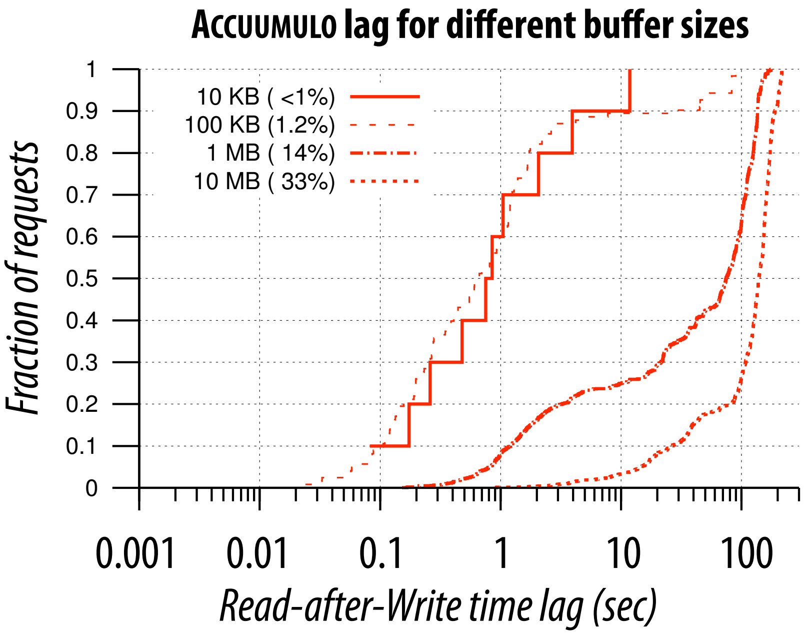
Test setup details

- YCSB++ tests on 1% of keys inserted by C1
  - C1 inserts 1 million keys, C2 reads 10K keys
  - Sampling avoids overloading ZooKeeper

- R-W lag for key K = time required by C2 to read K successfully
  - If C2 can't read K in the first attempt, tries again
  - Report the time lag for fraction of keys that need multiple read()s

Batch writing causes time lag





Delayed writes may not be seen for ~100 seconds

- Batching implementations affect latency; YCSB++ helps understand differences

Features Tested by YCSB++

Batch writing

Weak consistency

Table bulk loading

Table pre-splitting

Server-side filtering

Access control

Other Details

ACM SOCC 2011 paper available

Poster session(s) ☺

Future work

- Evolving YCSB++
  - Study additional table stores (Cassandra, MongoDB and CouchDB)
  - Test more features: Iterators and co-processors

- Understanding table store features
  - Cost-benefit tradeoff of different heuristics for compactions on tablet servers
  - Dynamo-style eventual consistency

Summary: YCSB++ tool

- For performance debugging & benchmarking **advanced features** using **extensions to YCSB**

Weak consistency semantics	Distributed clients using ZooKeeper
Fast insertion (pre-splits, bulk loads)	Multi-phase testing (with Hadoop)
Server-side filtering	New workload generators and database client API extensions
Fine-grained access control	

- Two case-studies: HBase & Accumulo
- Download at http://www.pdl.cmu.edu/ycsb++