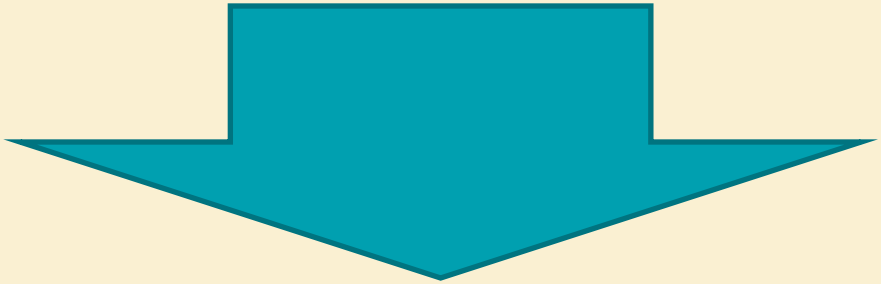
Cake

Enabling High-level SLOs on Shared Storage

Andrew Wang, Shivaram Venkataraman, Sara Alspaugh,
Randy Katz, Ion Stoica

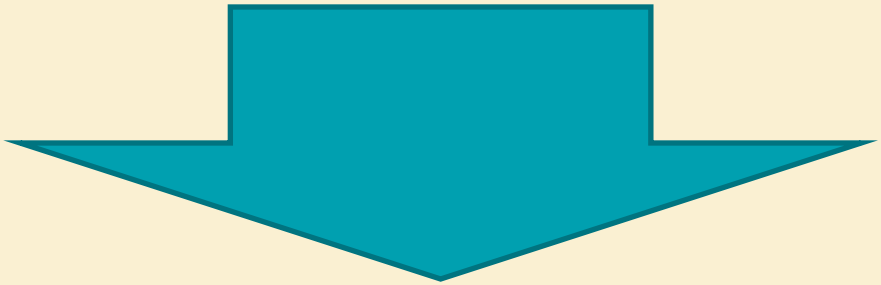
amplab


hadoop




hadoop


hadoop



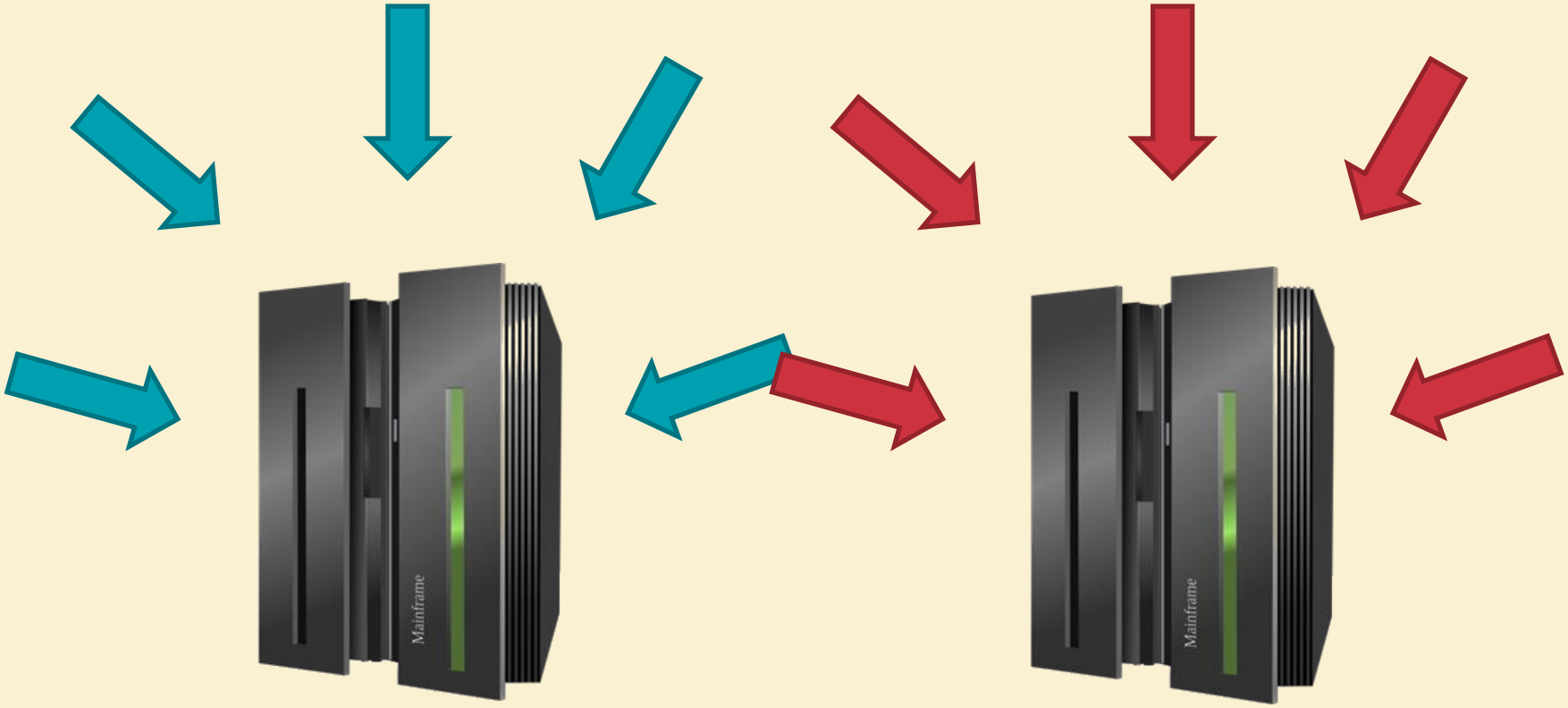

APACHE
HBASE


mongoDB

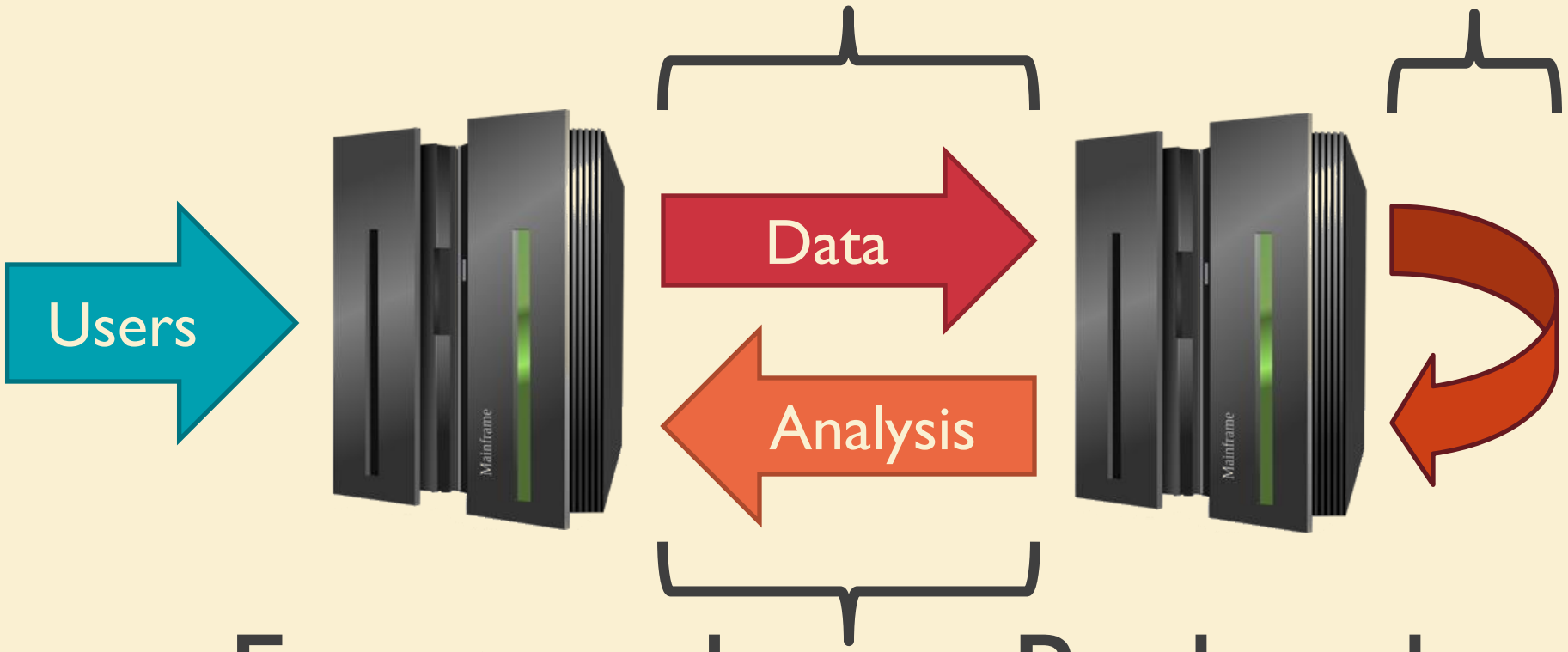

PostgreSQL


Cassandra

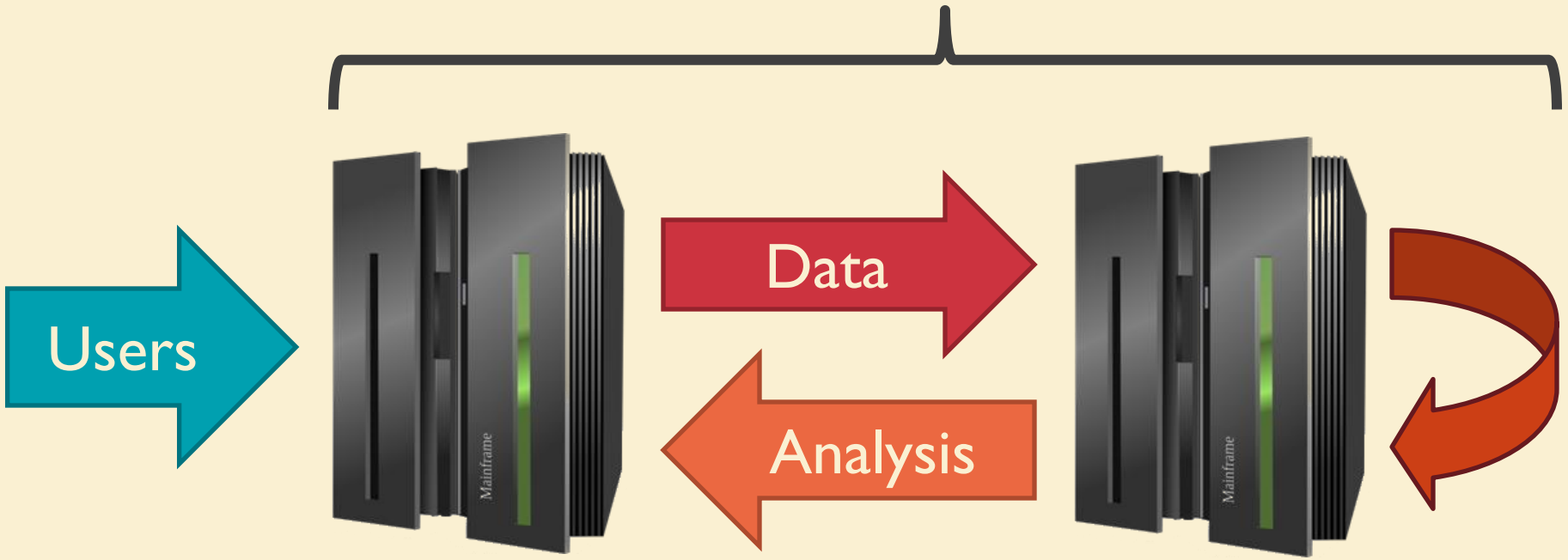
Trend toward consolidation of front-end and back-end storage systems



Front-end Backend


Users
Data
Analysis
Front-end
Backend

forever ~~Minutes to hours~~



Front-end Backend

Consolidated

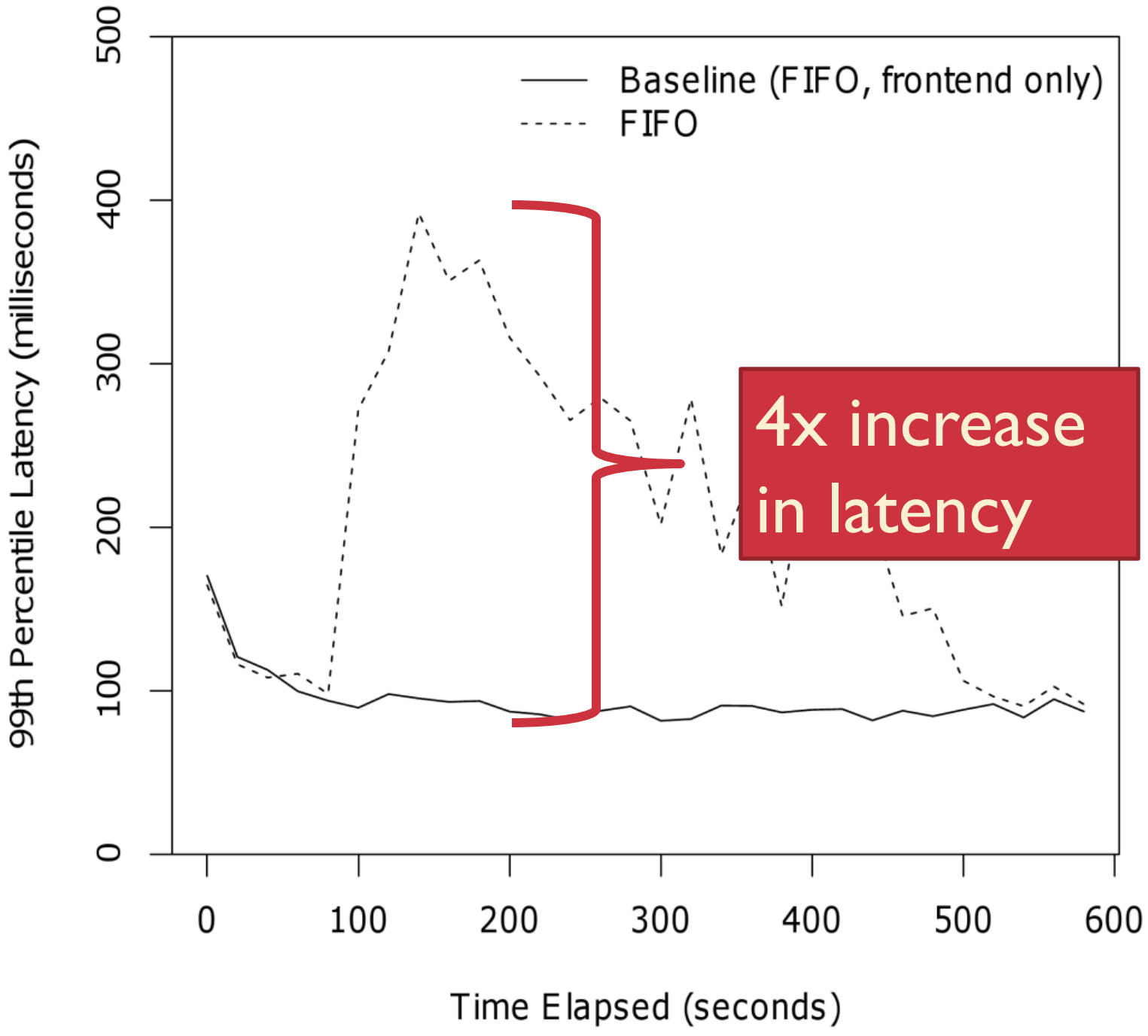
- Provisioning
- Utilization
- Analysis time

Consolidated

Nope.

Cluster scheduling
Privacy
Failure domains
Performance isolation

Performance isolation


99th Percentile Latency (milliseconds)
0
100
200
300
400
500
Baseline (FIFO, frontend only)
FIFO
4x increase in latency
0
100
200
300
400
500
600
Time Elapsed (seconds)

Objective

1. Combine two workloads
2. Meet front-end latency requirements
3. Then maximize batch throughput

Latency *vs.* Throughput

95th

High-percentile latency

99.9th

"…purely targeted at controlling performance at the 99.9th percentile."

High-level operations

service-level objective

performance metric

of **requirement**

for **type of request**

service-level objective

99th percentile latency

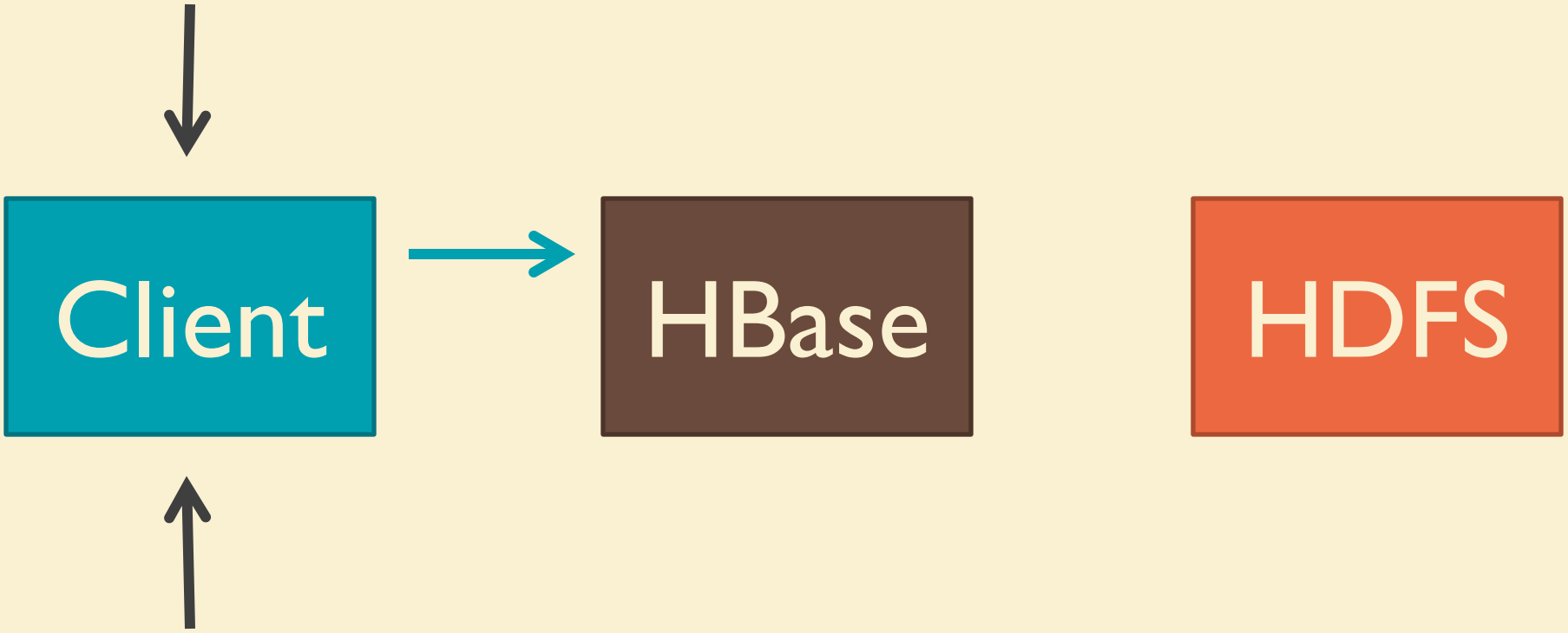
of 100 milliseconds

for get requests

Approach

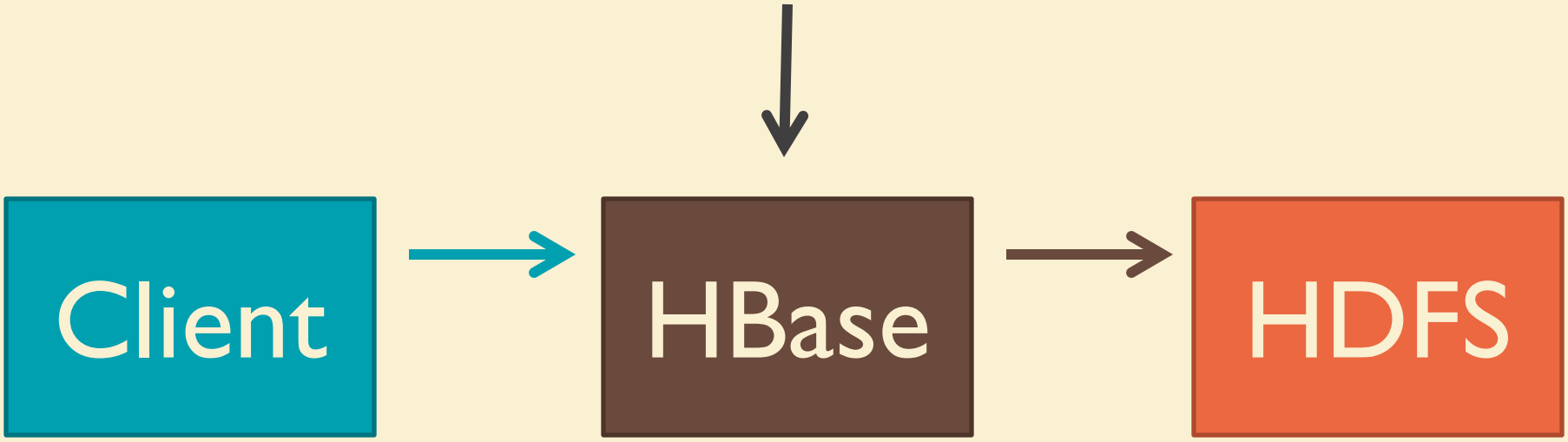
- Coordinated, multi-resource scheduler for consolidated storage workloads
- Add scheduling points within the system
- Dynamically adjust allocations to meet SLO of front-end workload
- Built on HBase/HDFS

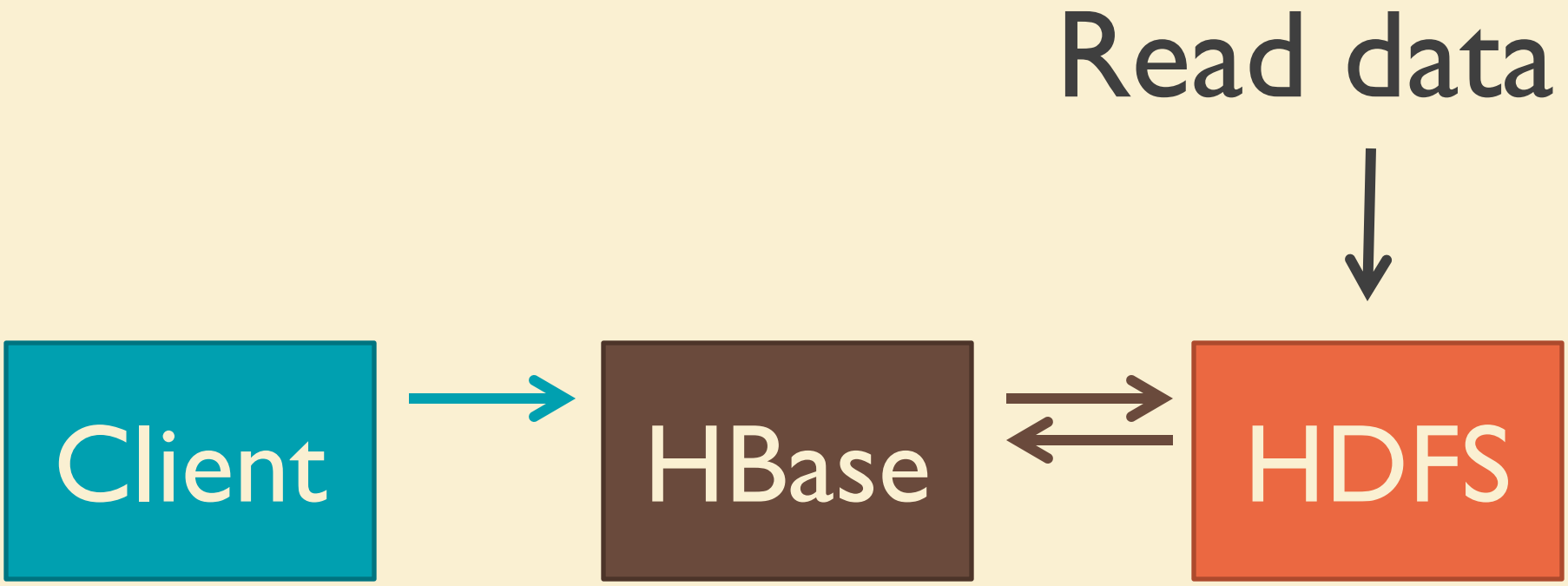
Front-end web server

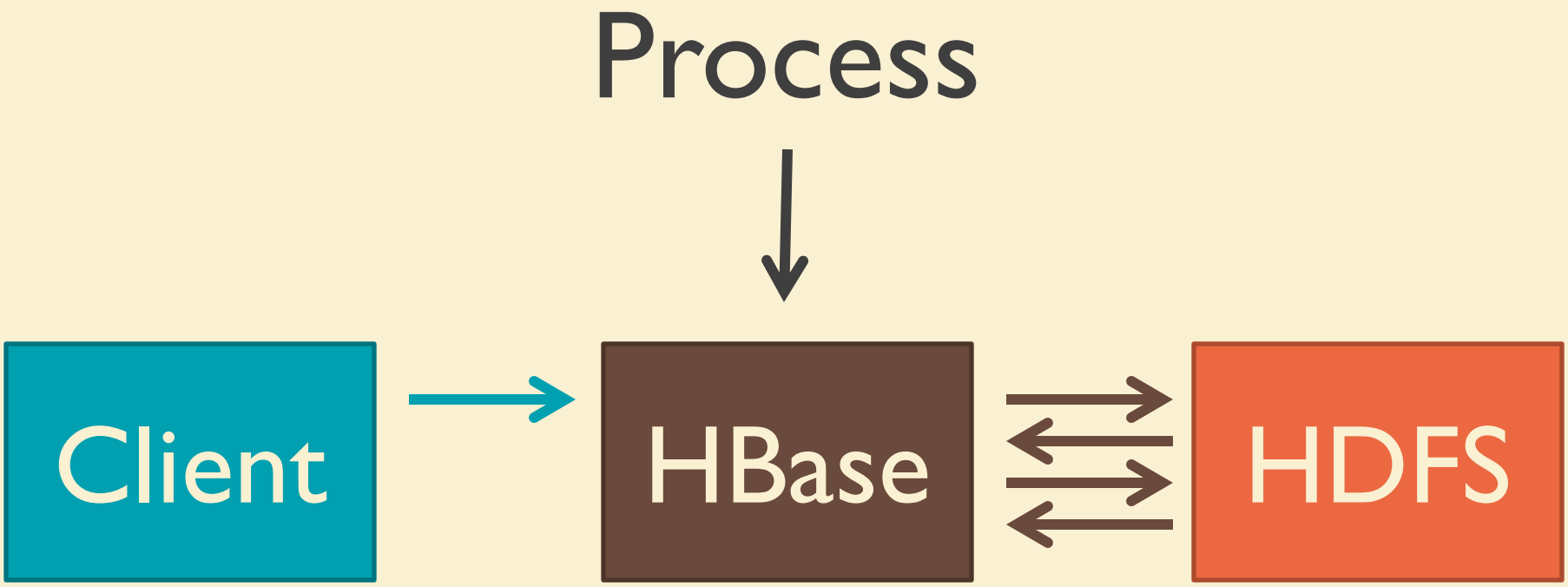


Batch MapReduce task

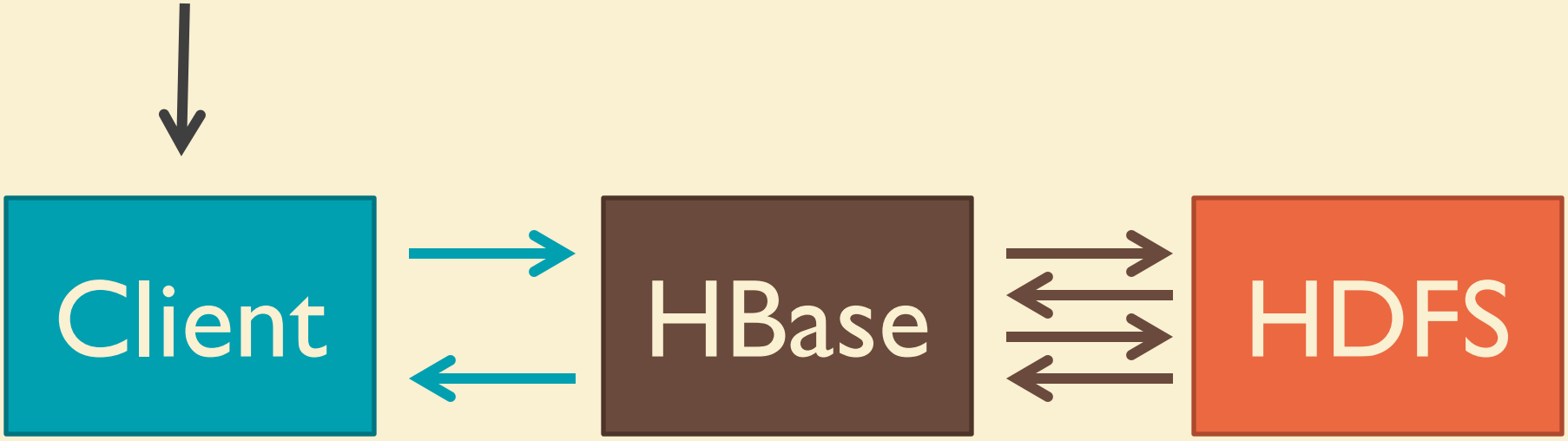
Index lookup



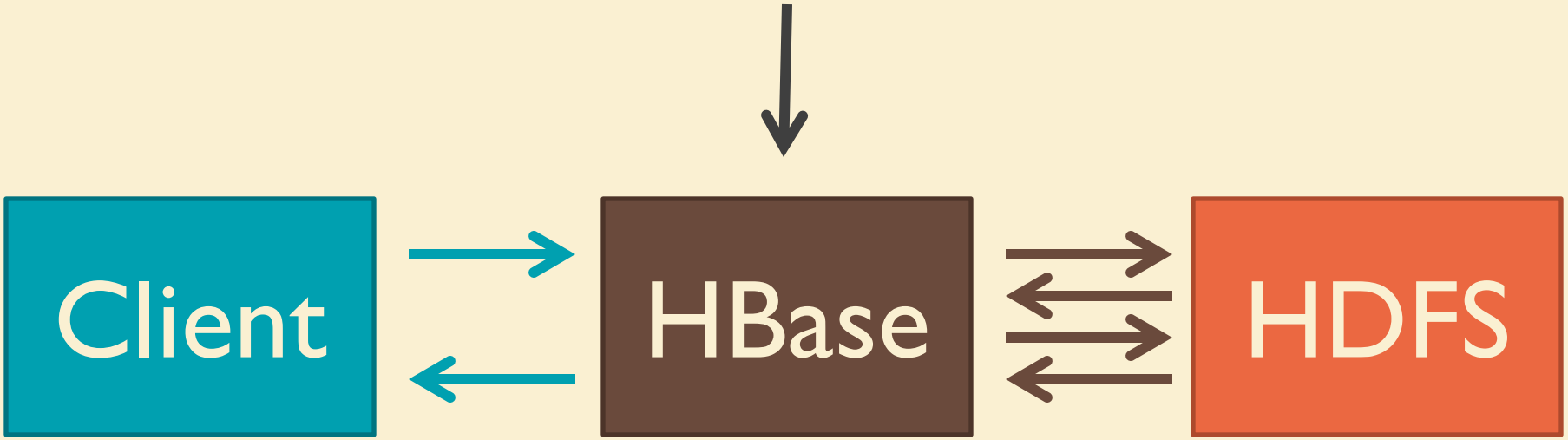

Read data
Client
HBase
HDFS

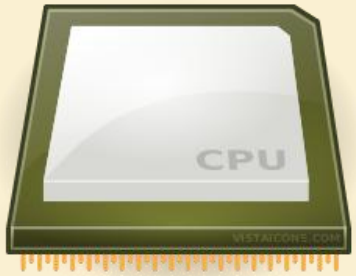

Process
Client
HBase
HDFS

Response

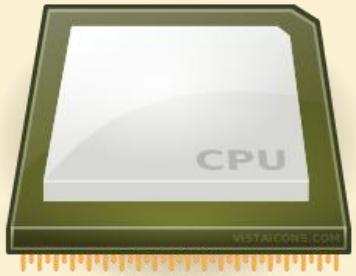


CPU usage

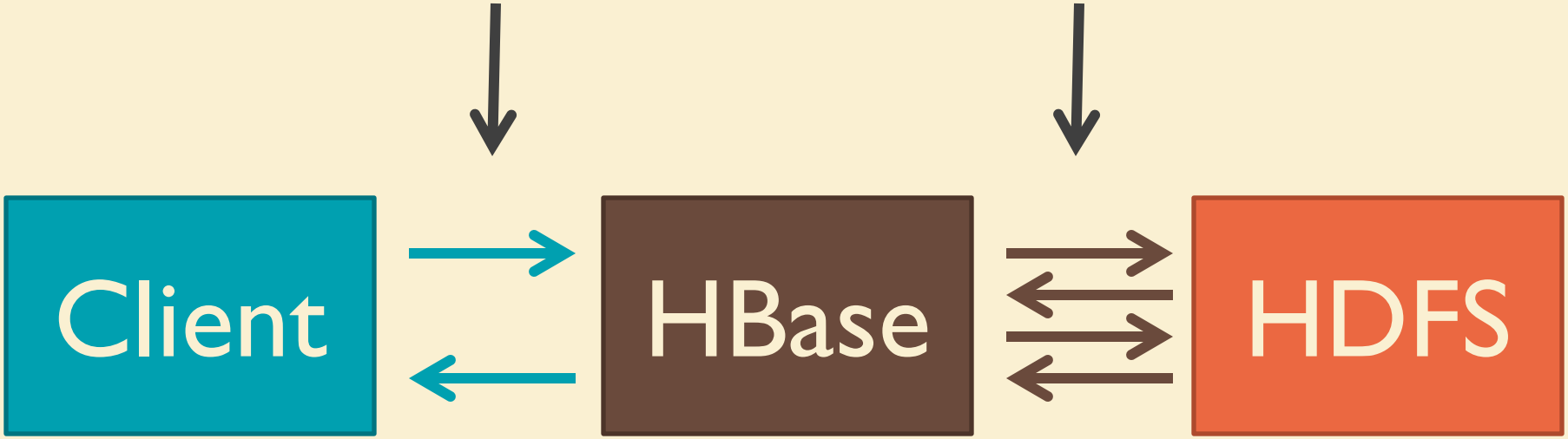


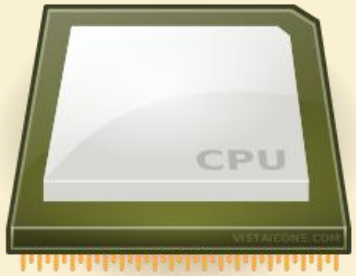


Disk usage

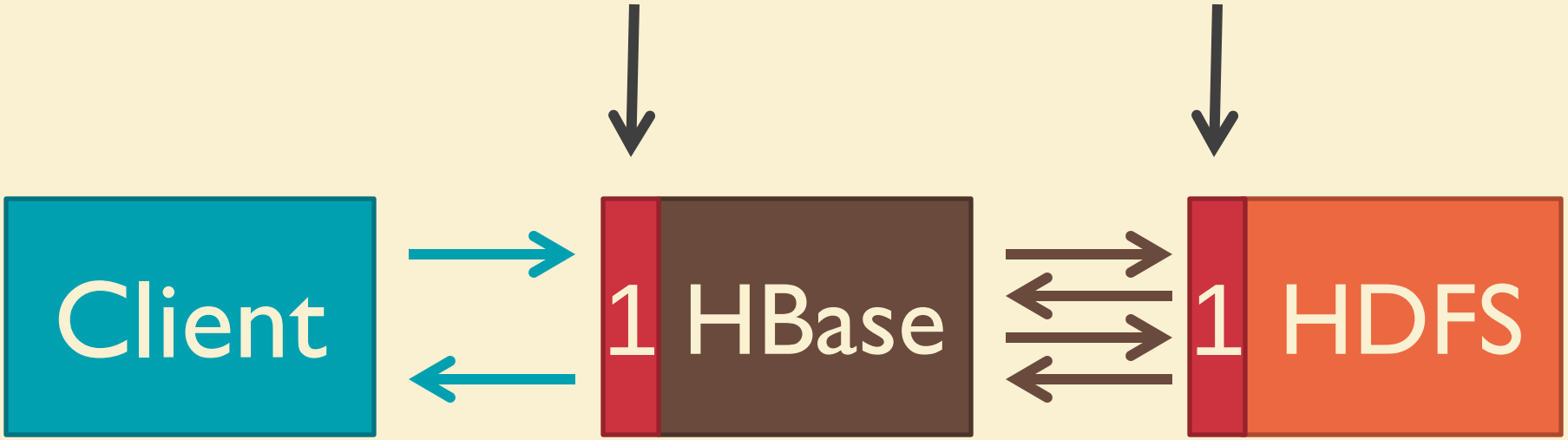


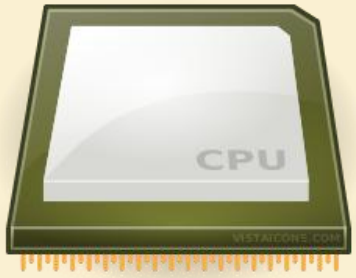
Clean RPC interfaces



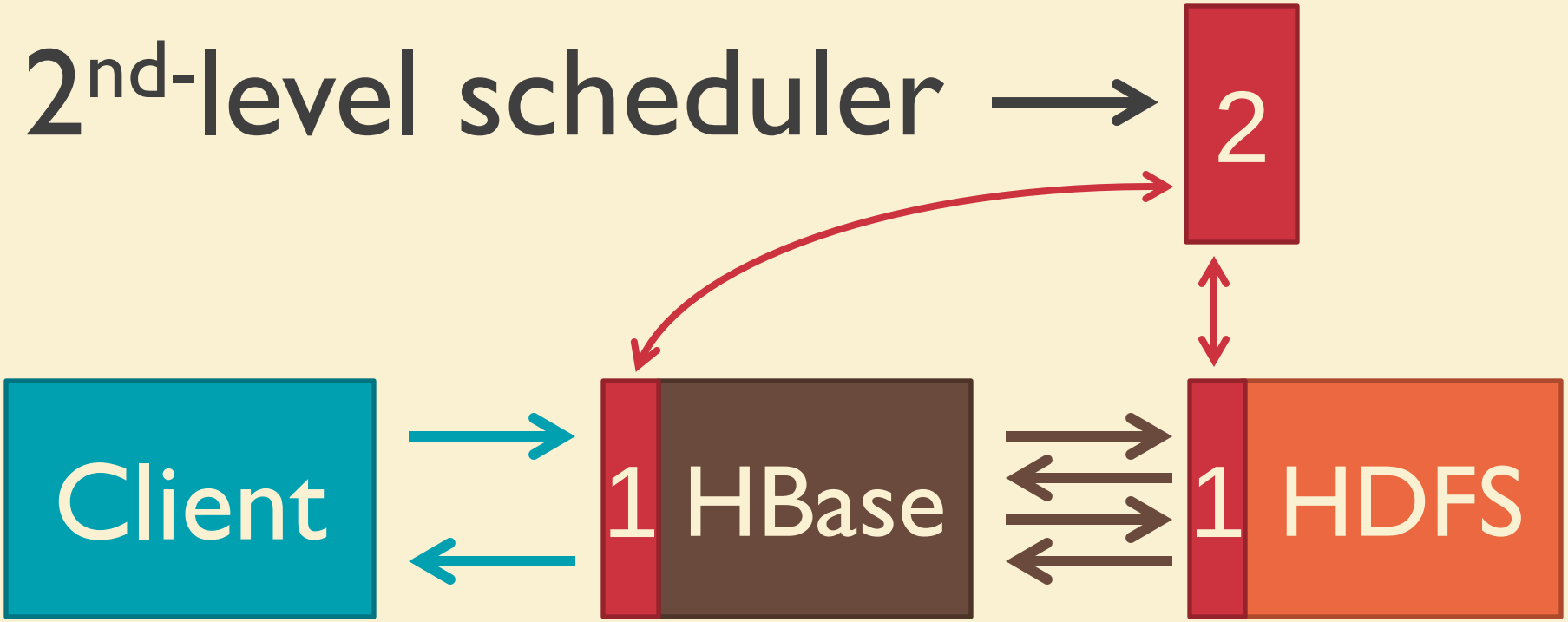


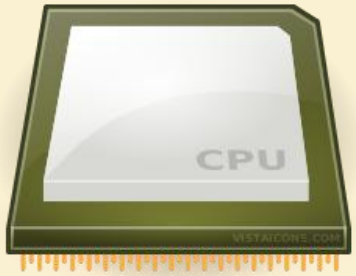
1st-level schedulers



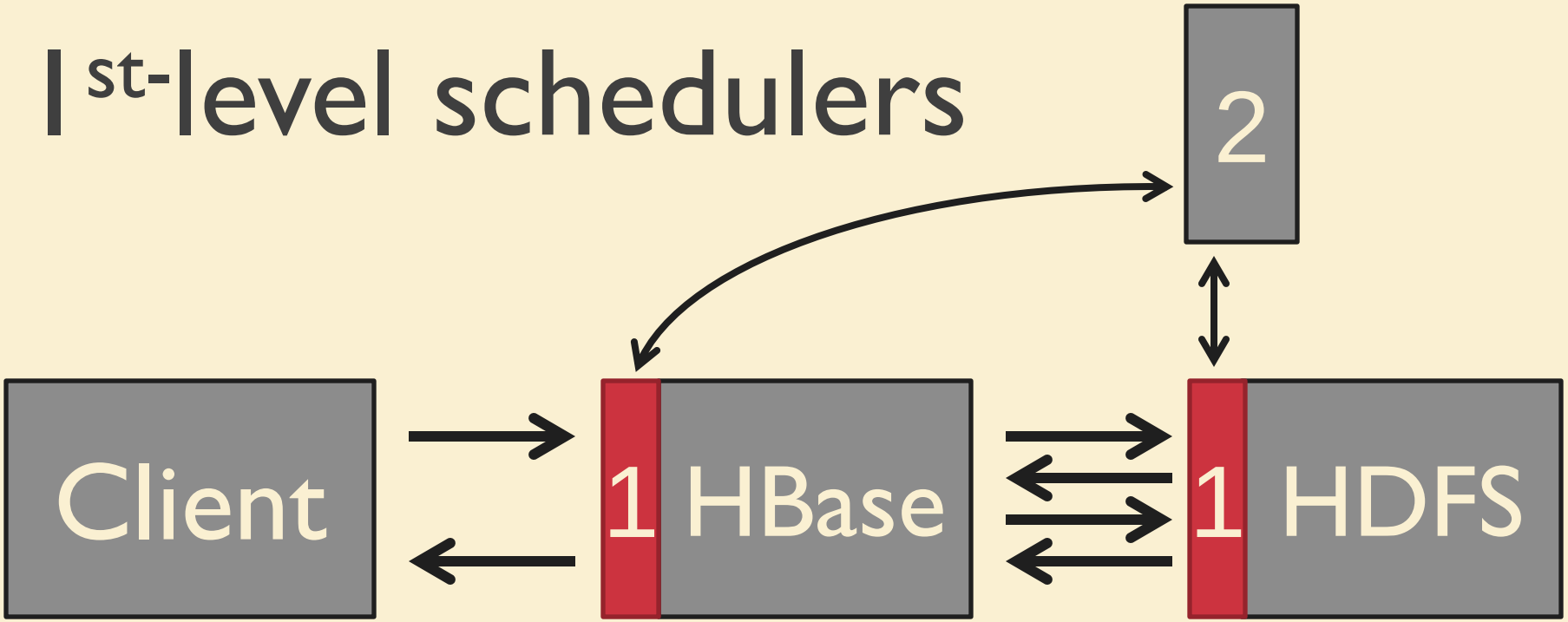


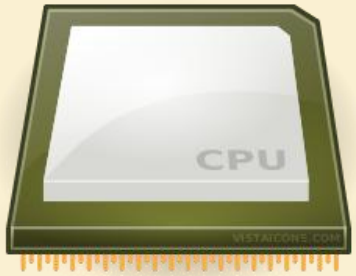
2nd-level scheduler





1st-level schedulers





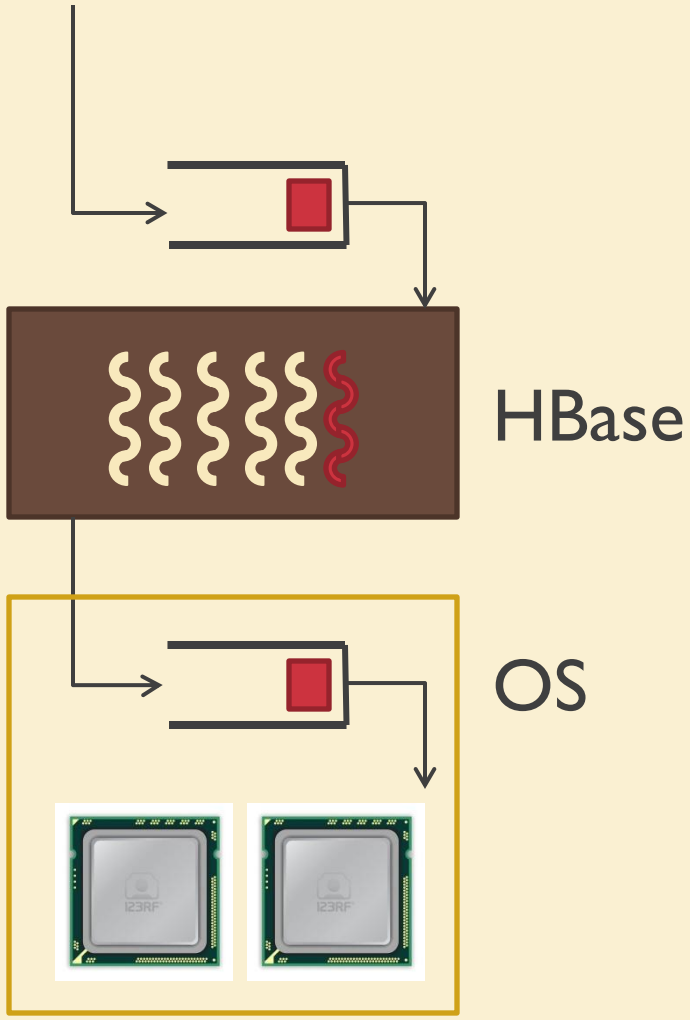
1st-Level Schedulers

- Provide effective control over access to the underlying hardware resource
  - Low latency
- Expose scheduling knobs to the 2nd-level scheduler

Three Requirements

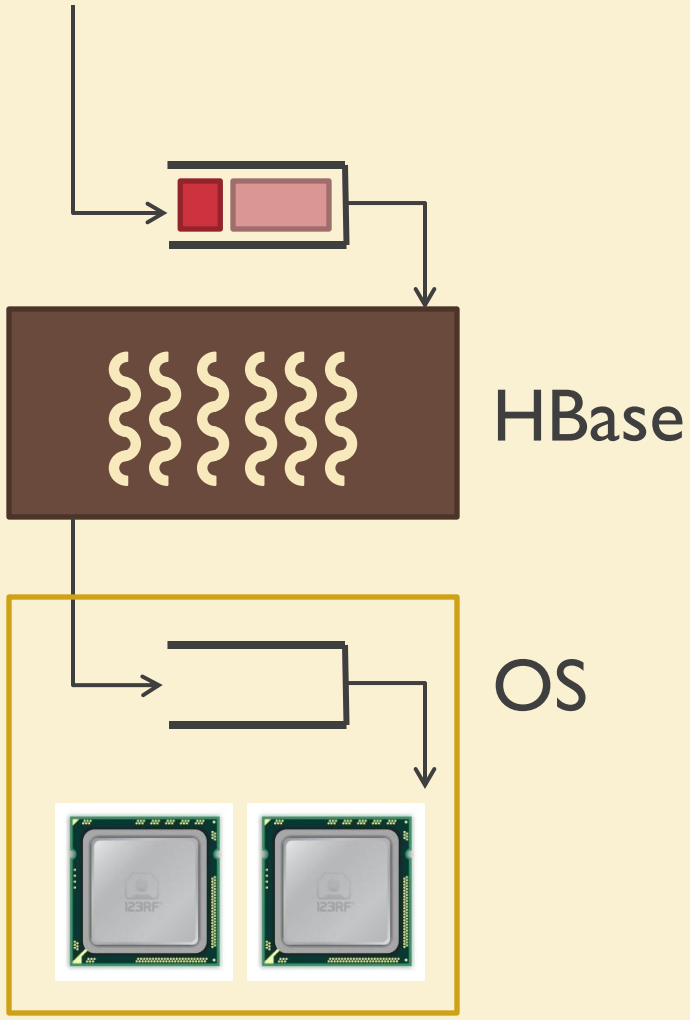
1. Differentiated Scheduling
2. Chunk Large Requests
3. Limit # Outstanding Requests

Differentiated Scheduling



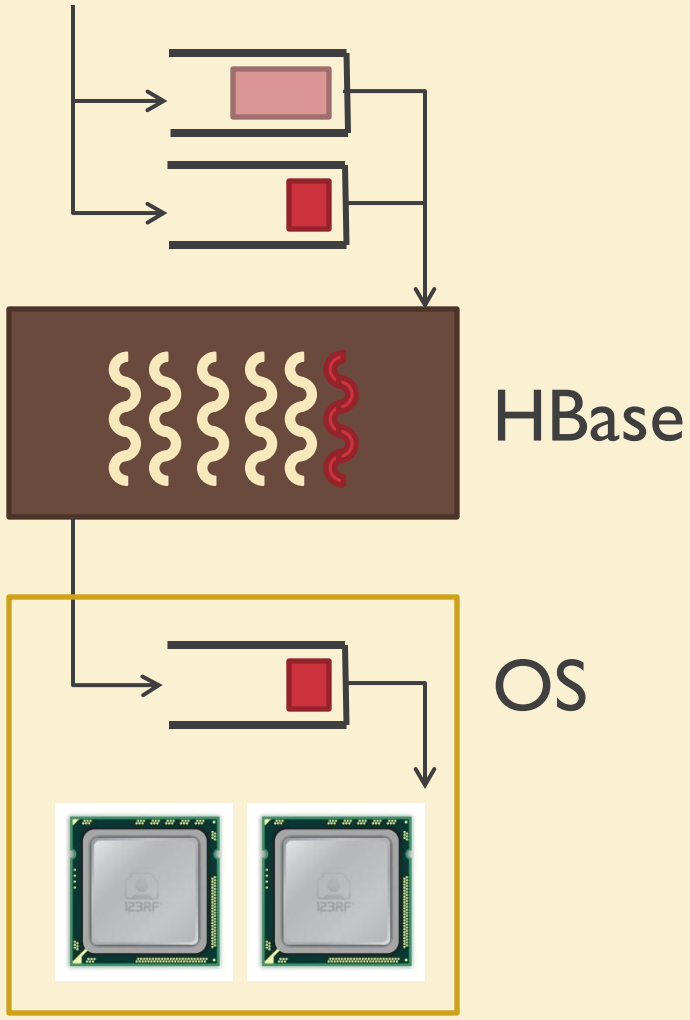
- Requests are handled by HBase threads
- Threads wait in OS to be scheduled on a CPU

Differentiated Scheduling



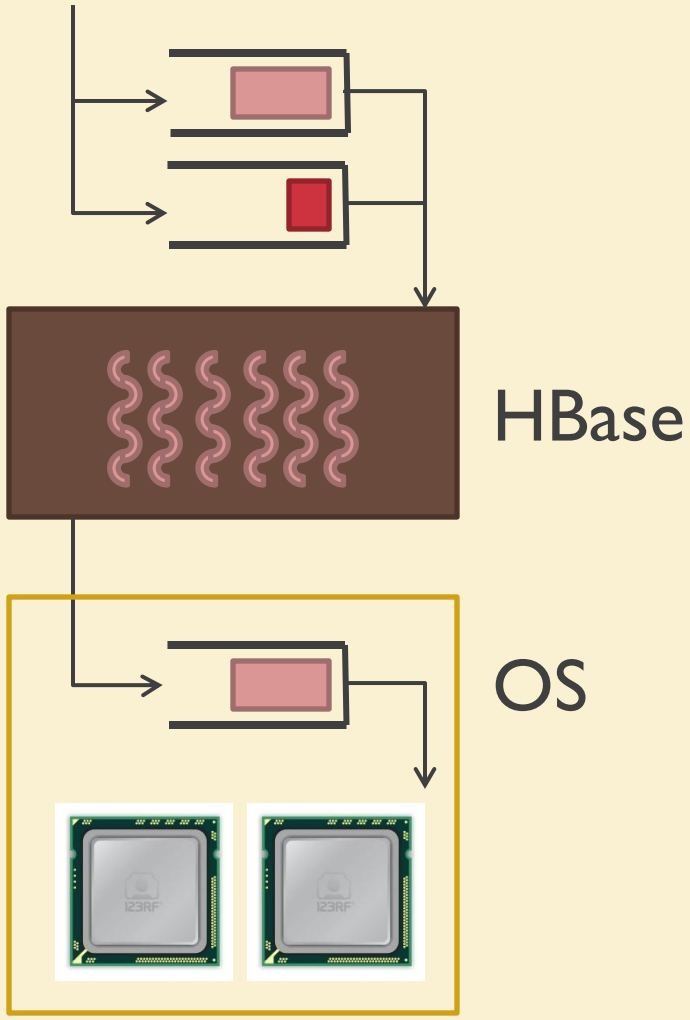
- Unfairness with a single FIFO queue
- Front-end requests get stuck behind batch

Differentiated Scheduling



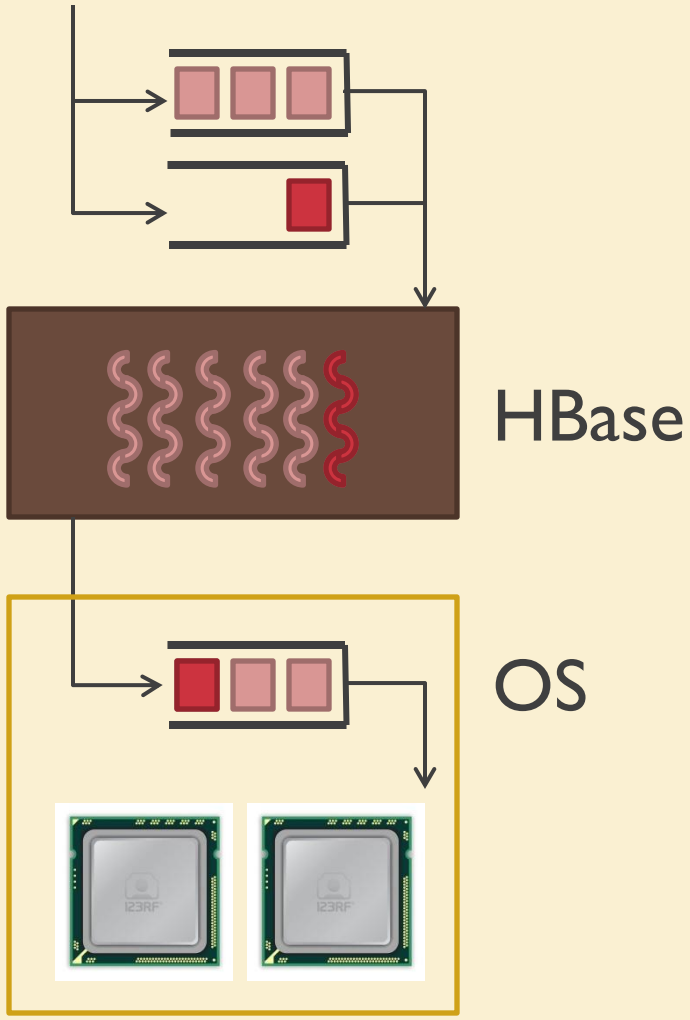
- Separate queues for each client
- Schedule based on allocations set by the 2nd-level scheduler

Chunk Large Requests



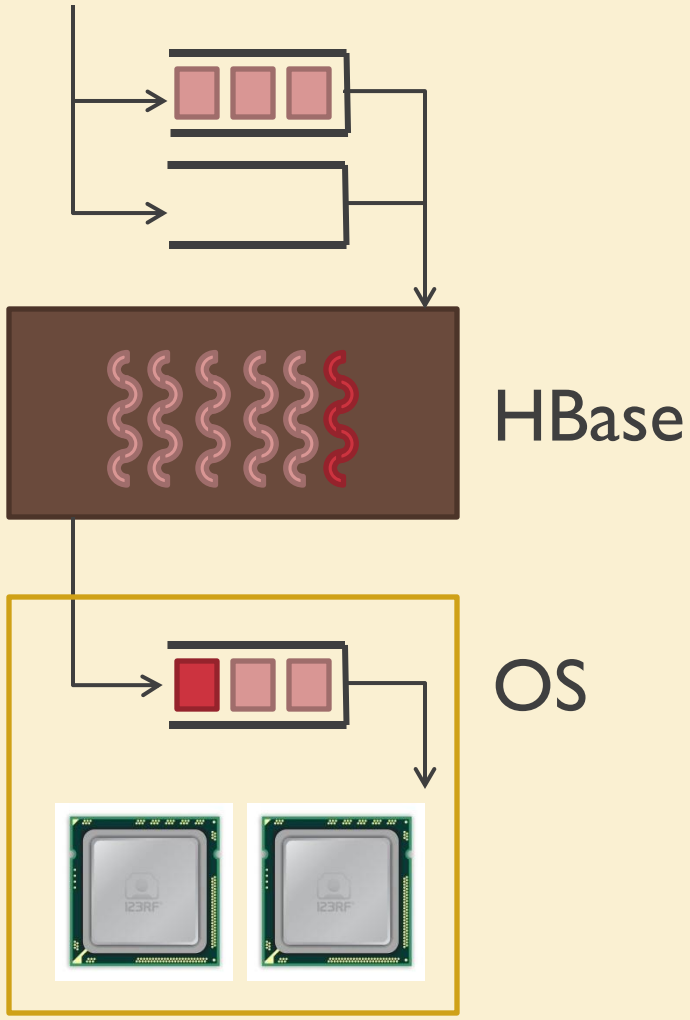
- Large requests tie up resources while being processed
- Lack of preemption

Chunk Large Requests



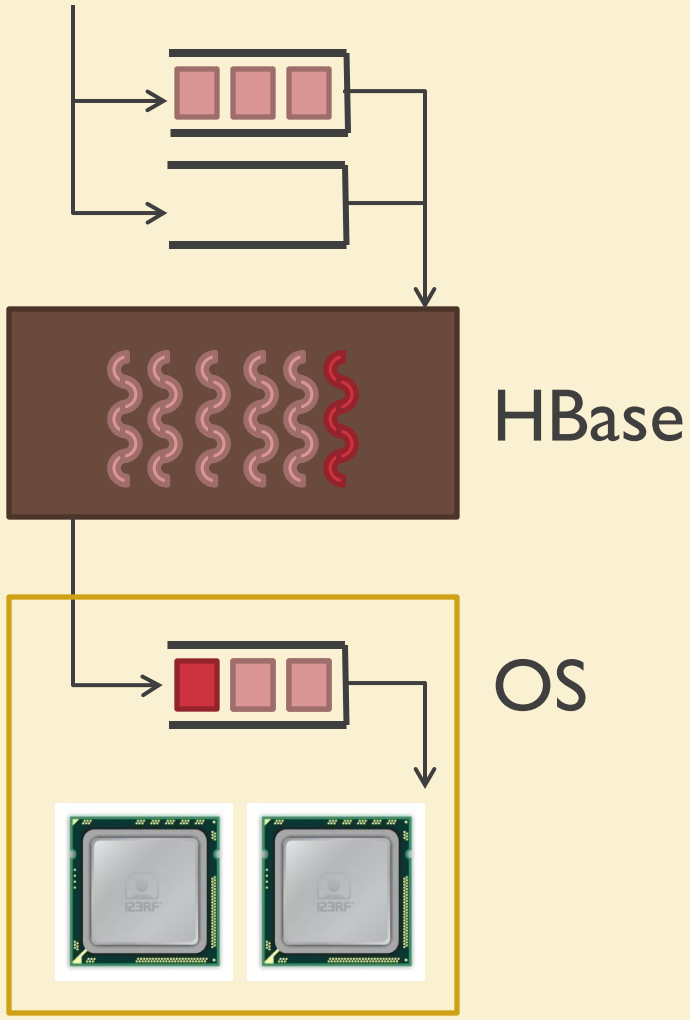
- Split large requests into multiple smaller chunks
- Only wait for a chunk, rather than an entire large request

Limit # of Outstanding Requests



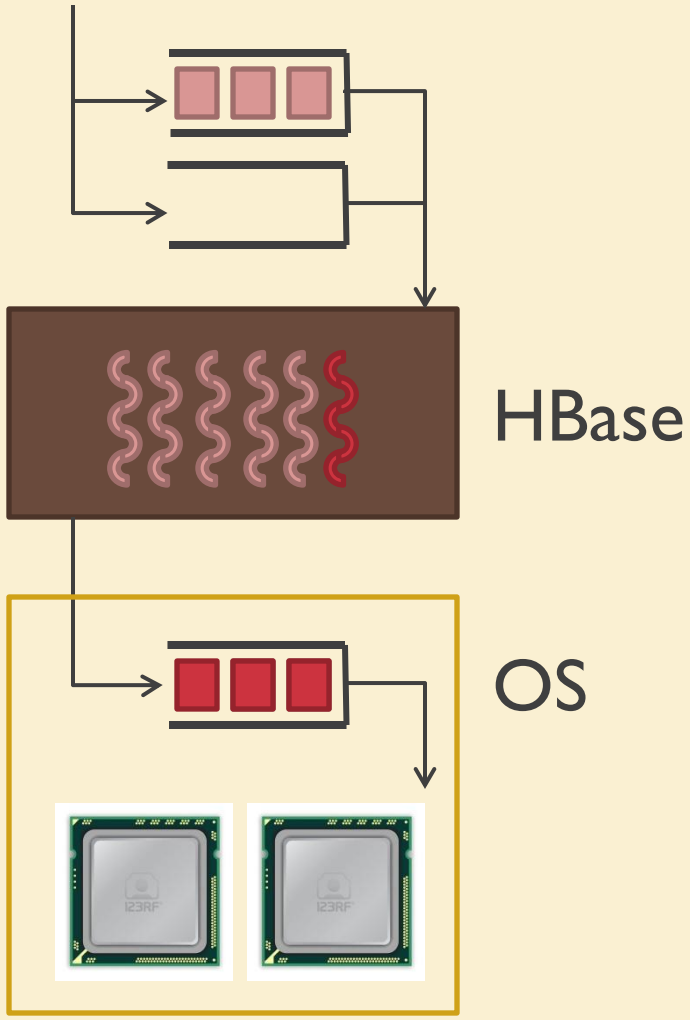
- Long queues outside of Cake impact latency
- Caused by too many outstanding requests
- Need to determine level of overcommit

Limit # of Outstanding Requests



- TCP Vegas-like scheme
- Latency as estimate of load on a resource
- Dynamically adjust # of threads

Limit # of Outstanding Requests



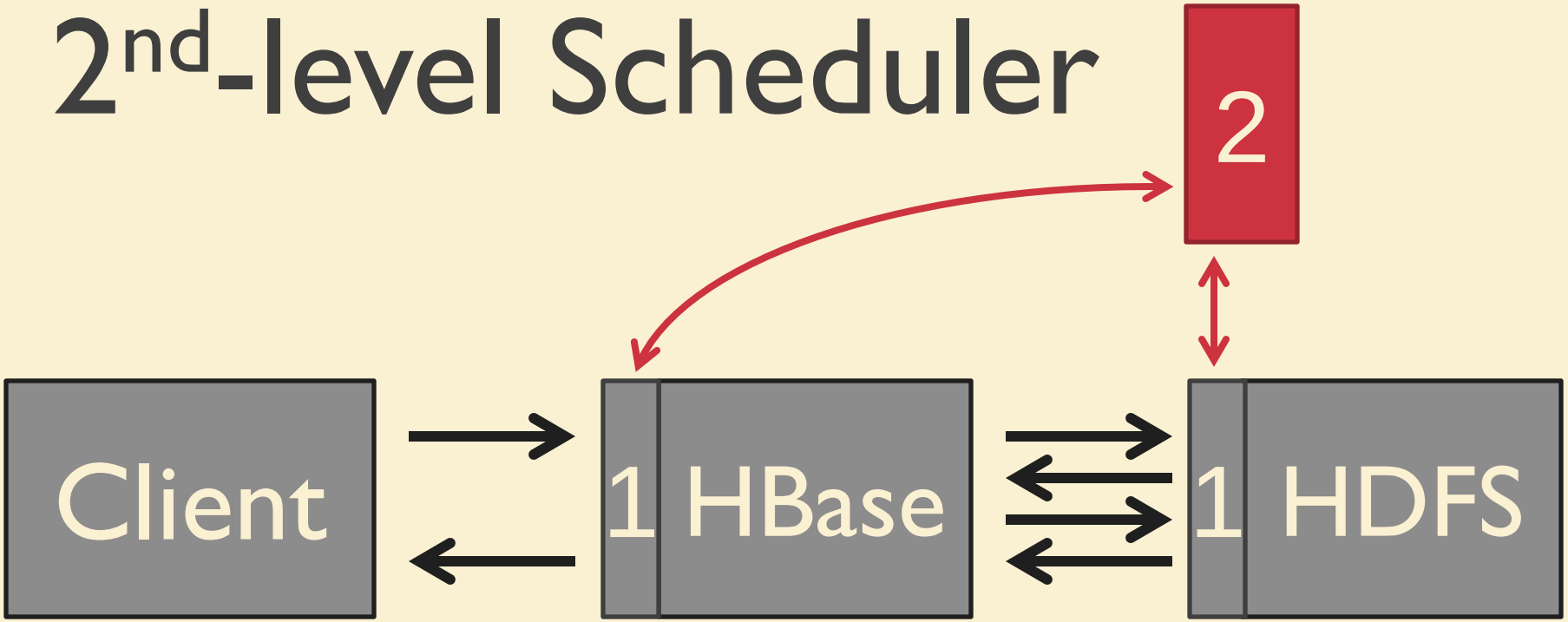
- Upper and lower latency bounds
- Additively increase # threads if underloaded
- Multiplicatively decrease # threads if overloaded
- Converges in the general case

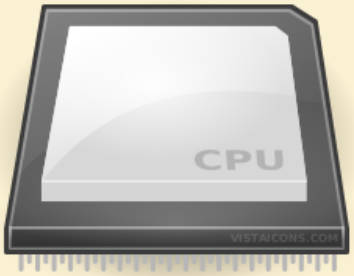
1st-level Schedulers

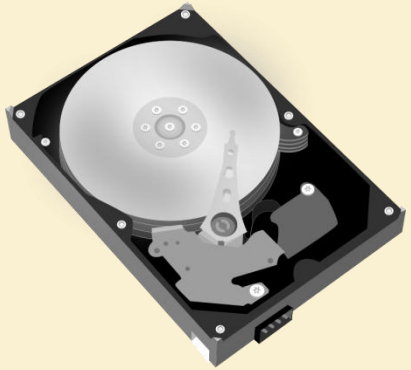
- Separate queues enable scheduling at each resource
- Chunk large requests for better preemption
- Dynamically size thread pools to prevent long queues outside of Cake

- More details in the paper

2nd-level Scheduler







2nd-level Scheduler

- Coordinates allocation at 1st-level schedulers
- Monitors and enforces front-end SLO
- Two allocation phases
  - SLO compliance-based
  - Queue occupancy-based

- Just a sketch, full details are in the paper

SLO compliance-based

- Disk is the bottleneck in storage workloads
- Improving performance requires increasing scheduling allocation at disk (HDFS)

- Increase allocation when not meeting SLO
- Decrease allocation when easily meeting SLO

Queue Occupancy-based

- Want HBase/HDFS allocation to be balanced
- HBase can incorrectly throttle HDFS
- Look at queue occupancy
  - % of time a client's request is waiting in the queue at a resource
  - Measure of queuing at a resource
- Increase when more queuing at HBase
- Decrease when more queuing at HDFS

Details

- Using both proportional share and reservations at 1st-level schedulers
- Linear performance model
- Evaluation of convergence properties

Recap

- 1st-level schedulers
  - Effective control over resource consumption
  - HBase and HDFS
- 2nd-level scheduler
  - Decides allocation at 1st-level schedulers
  - Enforces front-end client's high-level SLO

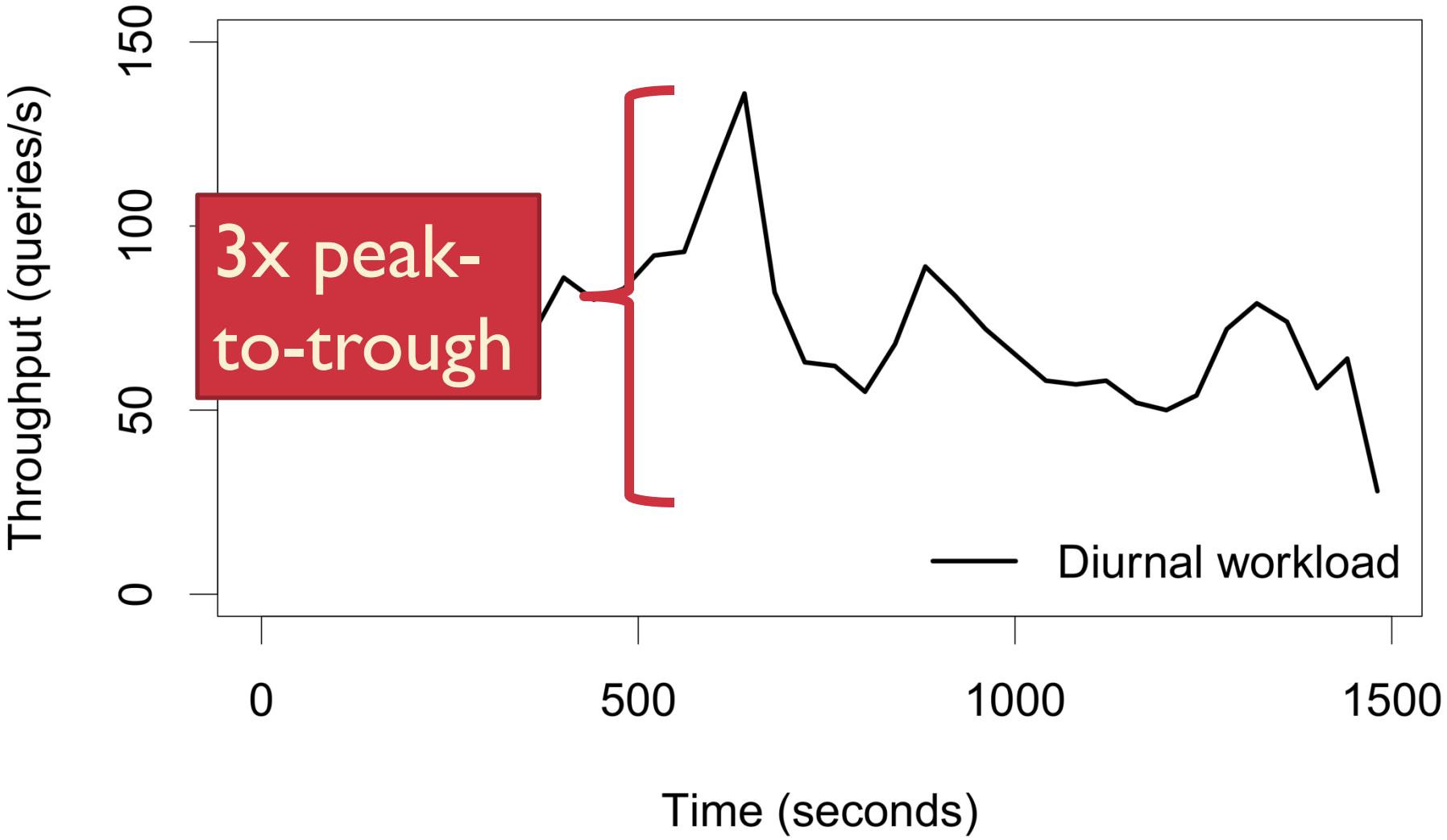
Evaluation

- Consolidated workloads
- Front-end YCSB client doing 1-row gets
- Batch YCSB/MR client doing 500-row scans
- c1.xlarge EC2 instances

Diurnal Workload

- Front-end running web-serving workload
- Batch YCSB client running at max throughput

- Evaluate adaptation to dynamic workload
- Evaluate latency vs. throughput tradeoff


Throughput (queries/s)
0
50
100
150
0
500
1000
1500
Time (seconds)
3x peak-to-trough
Diurnal workload

Front-end 99th SLO	% Meeting SLO	Batch Throughput
100ms	98.77%	23 req/s
150ms	99.72%	38 req/s
200ms	99.88%	45 req/s

Analysis Cycle

- 20-node cluster
- Front-end YCSB client running diurnal pattern
- Batch MapReduce scanning over 386GB data

- Evaluate analytics time and provisioning cost
- Evaluate SLO compliance

Scenario	Time	Speedup	Nodes
Unconsolidated	1722s	1.0x	40
Consolidated	1030s	1.7x	20

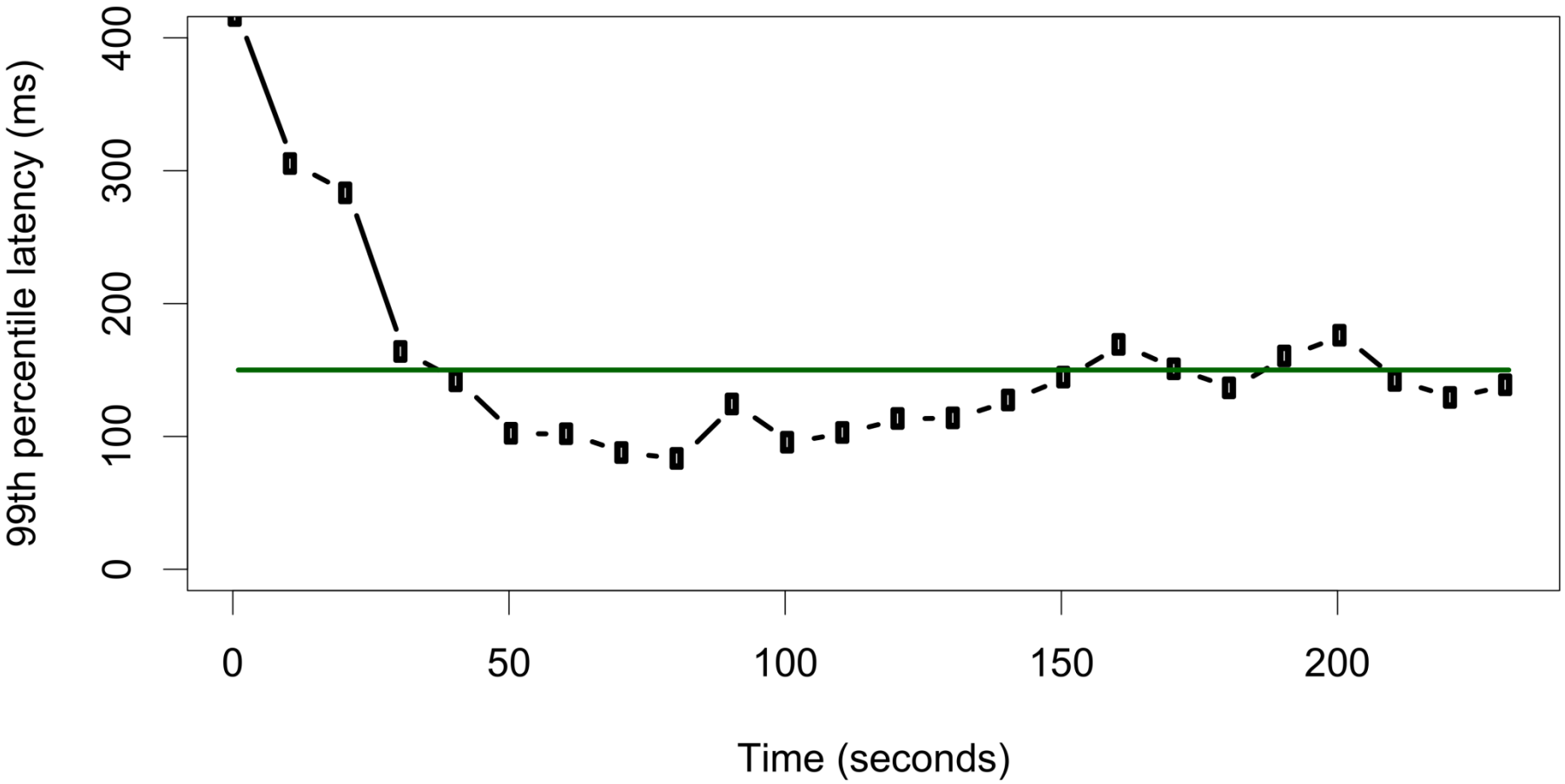
Front-end 99th SLO	% Requests Meeting SLO
100ms	99.95%

Evaluation

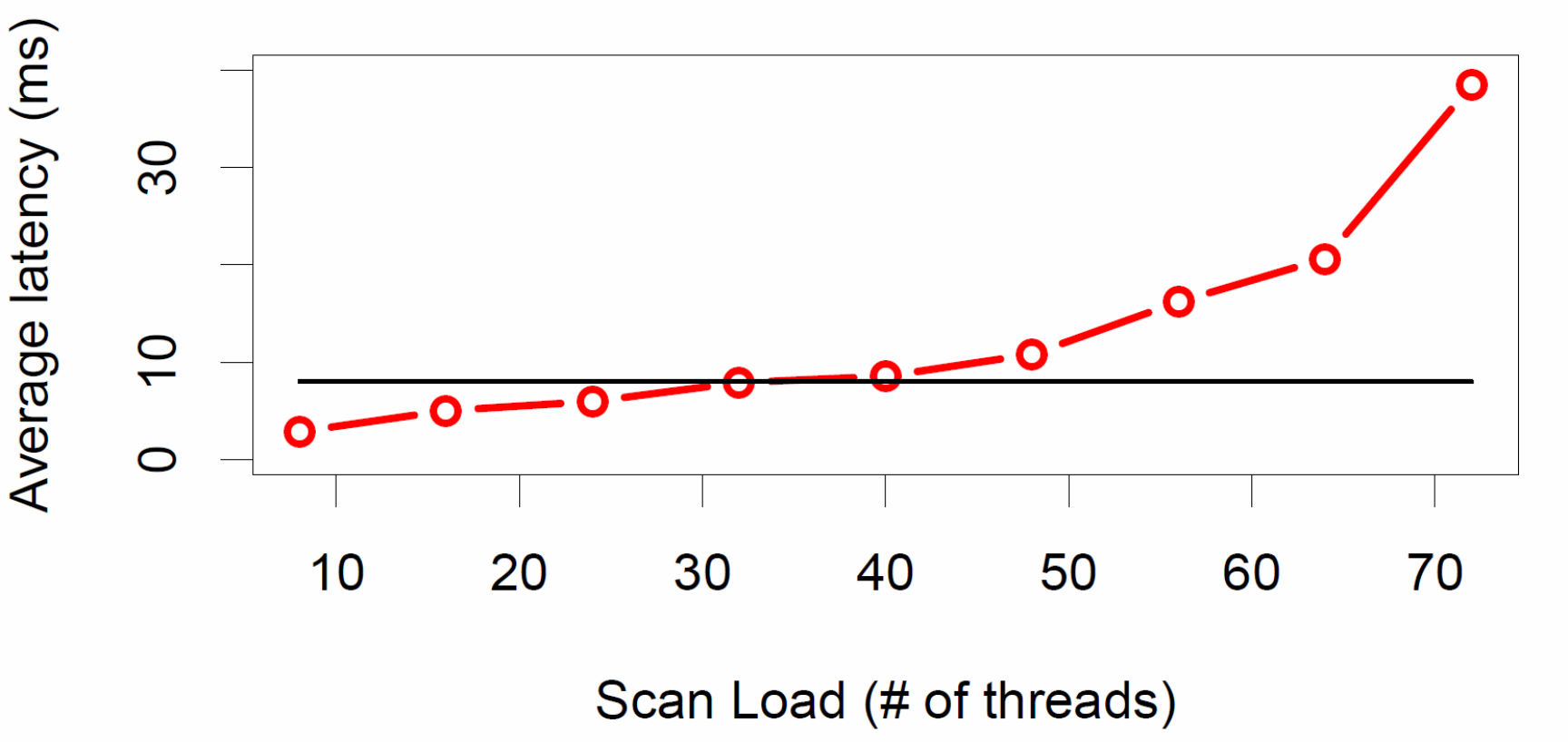
- Adapts to dynamic front-end workloads
- Can set different SLOs to flexibly move within latency vs. throughput tradeoff
- Significant gains from consolidation
  - 1.7x speedup
  - 50% provisioning cost

Conclusion

- Coordinated, multi-resource scheduler for storage workloads
- Directly enforce high-level SLOs
  - High-percentile latency
  - High-level storage system operations
- Significant benefits from consolidation
  - Provisioning costs
  - Analytics time


99th percentile latency (ms)
0
100
200
300
400
0
50
100
150
200
Time (seconds)

Limit Outstanding Requests



Limit Outstanding Requests