Sparrow

Distributed Low-Latency Spark Scheduling

Kay Ousterhout, Patrick Wendell, Matei Zaharia, Ion Stoica

amplab

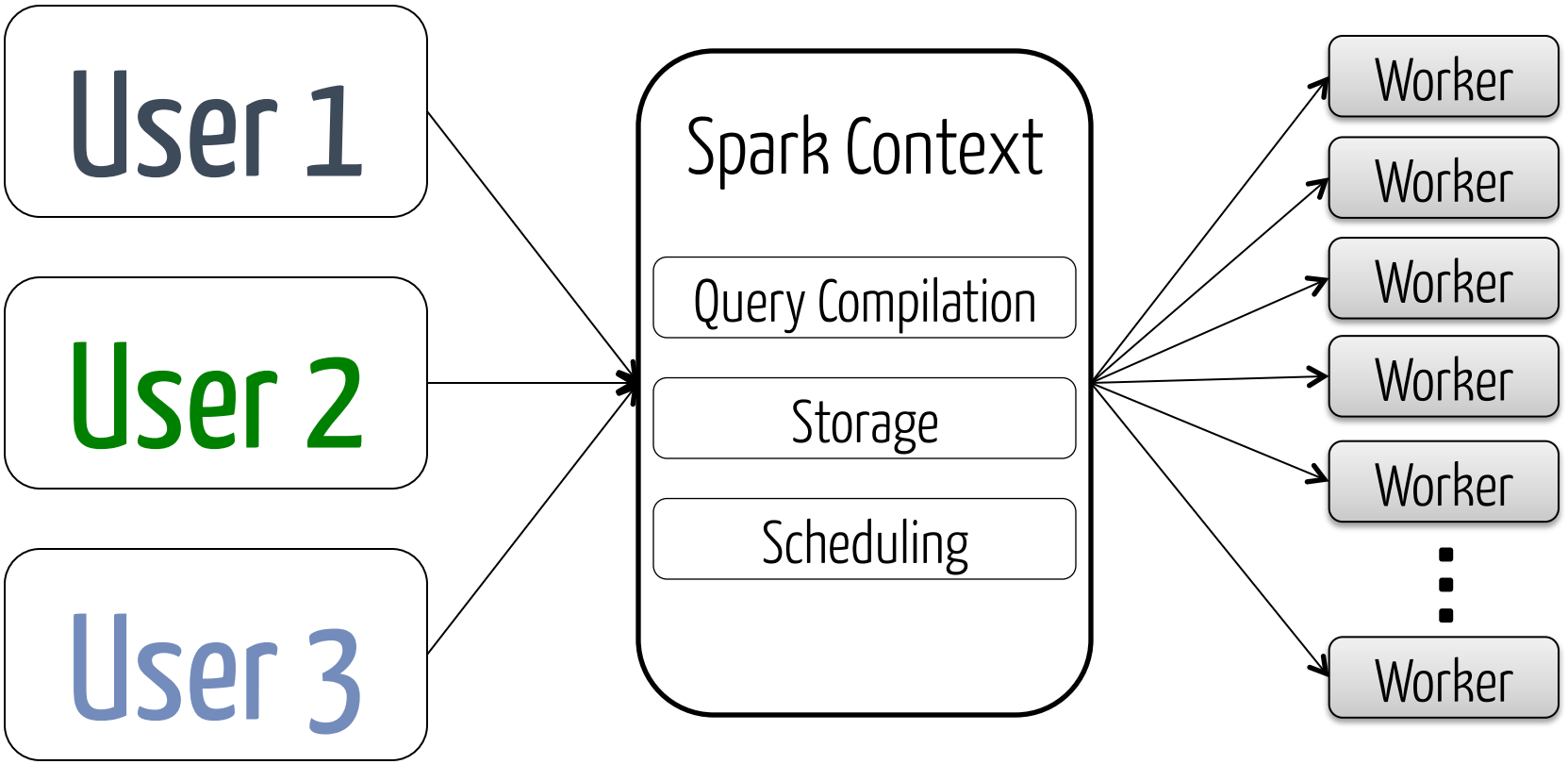
Outline

The Spark scheduling bottleneck

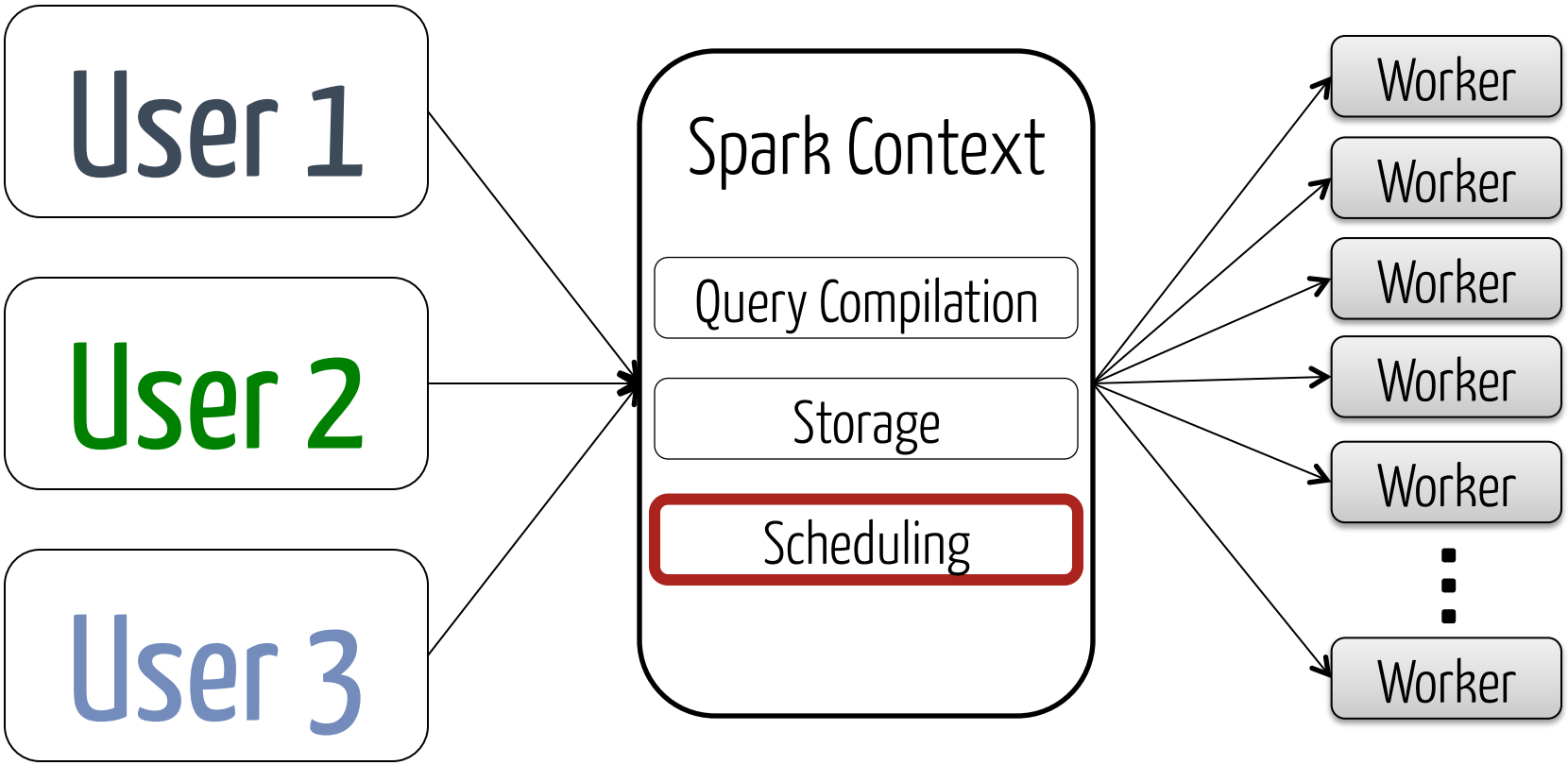
Sparrow's fully distributed, fault-tolerant technique

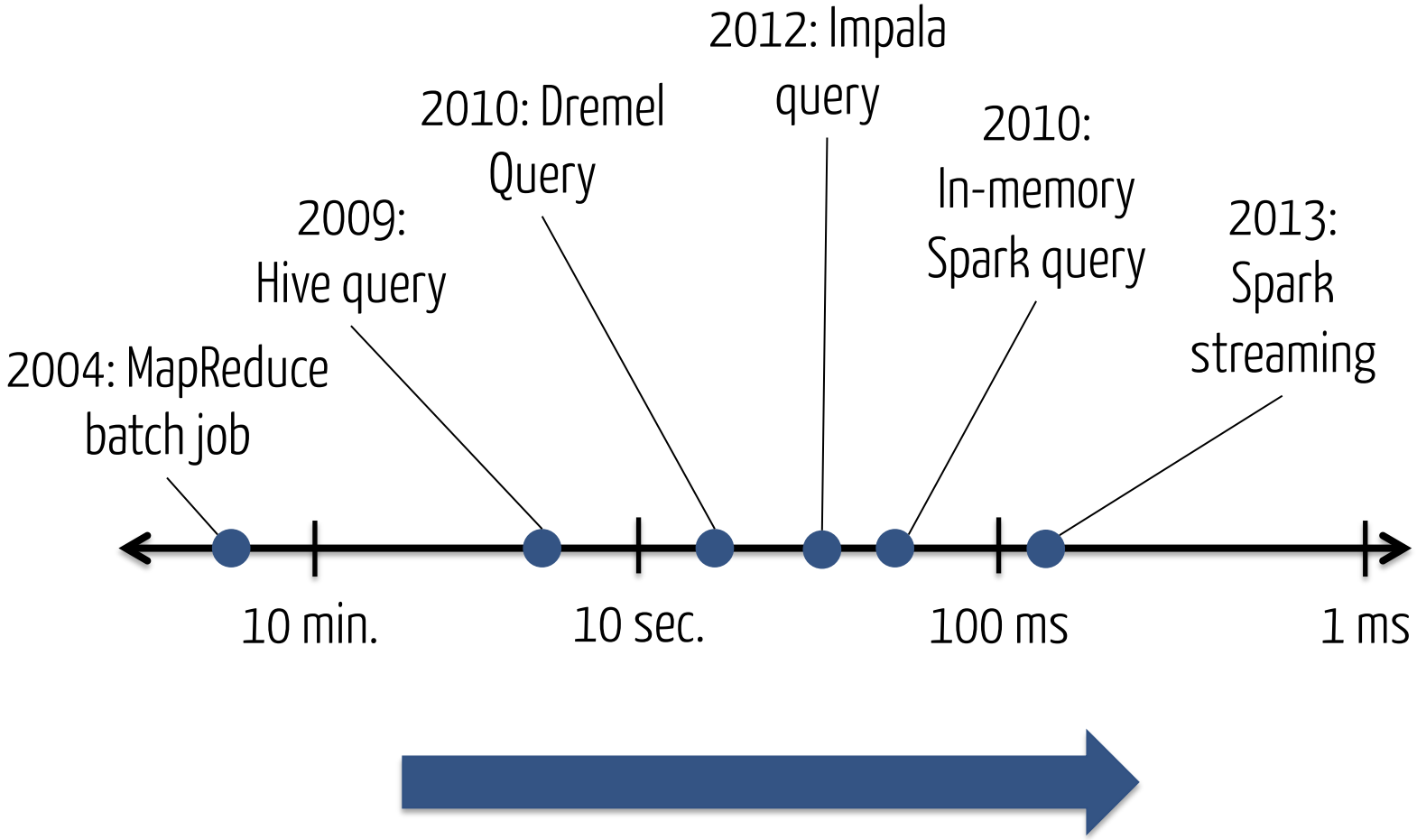
Sparrow's near-optimal performance

Spark Today


User 1
User 2
User 3
Spark Context
Query Compilation
Storage
Scheduling
Worker
Worker
Worker
Worker
Worker
Worker

Spark Today


User 1
User 2
User 3
Spark Context
Query Compilation
Storage
Scheduling
Worker
Worker
Worker
Worker
Worker
Worker



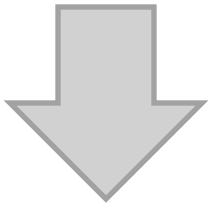
Job Latencies Rapidly Decreasing

Job latencies rapidly decreasing

Job latencies rapidly decreasing

+

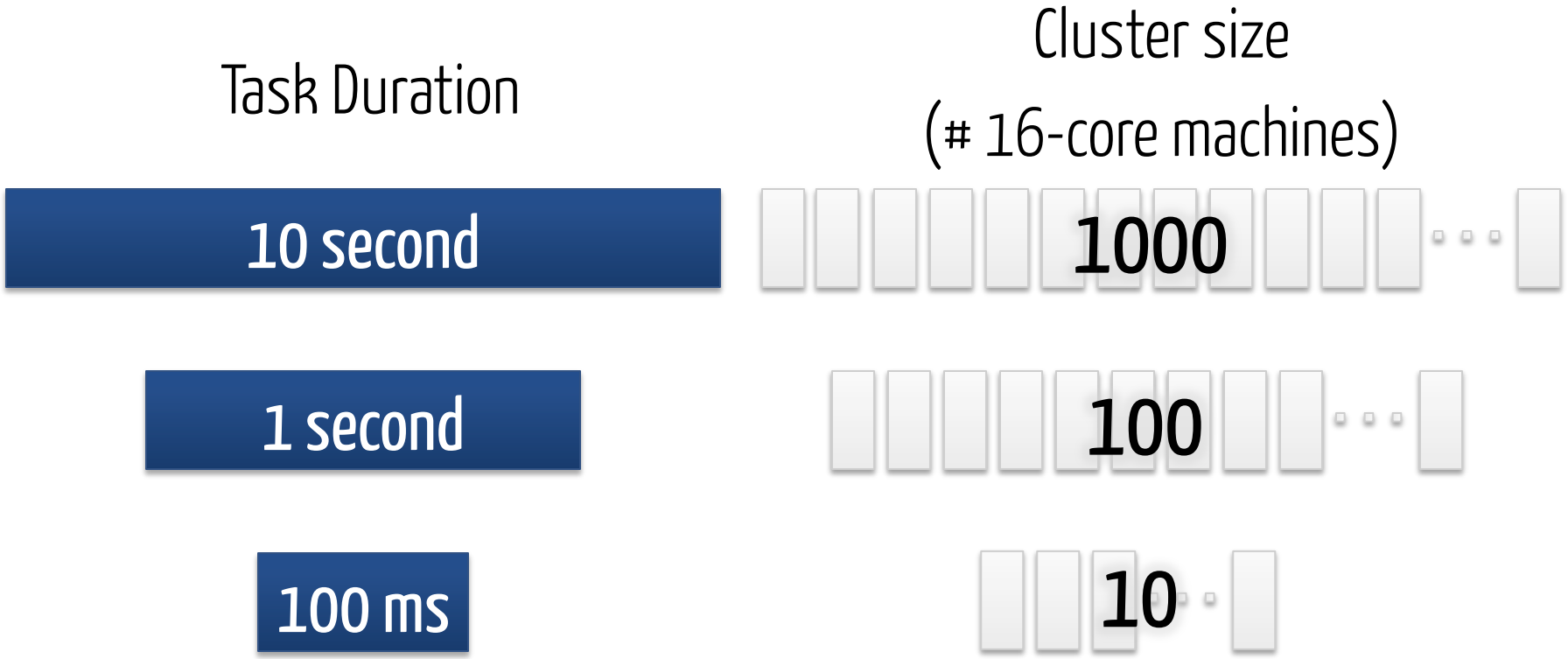
Spark deployments growing in size



Scheduling bottleneck!

Spark scheduler throughput:

1500 tasks / second



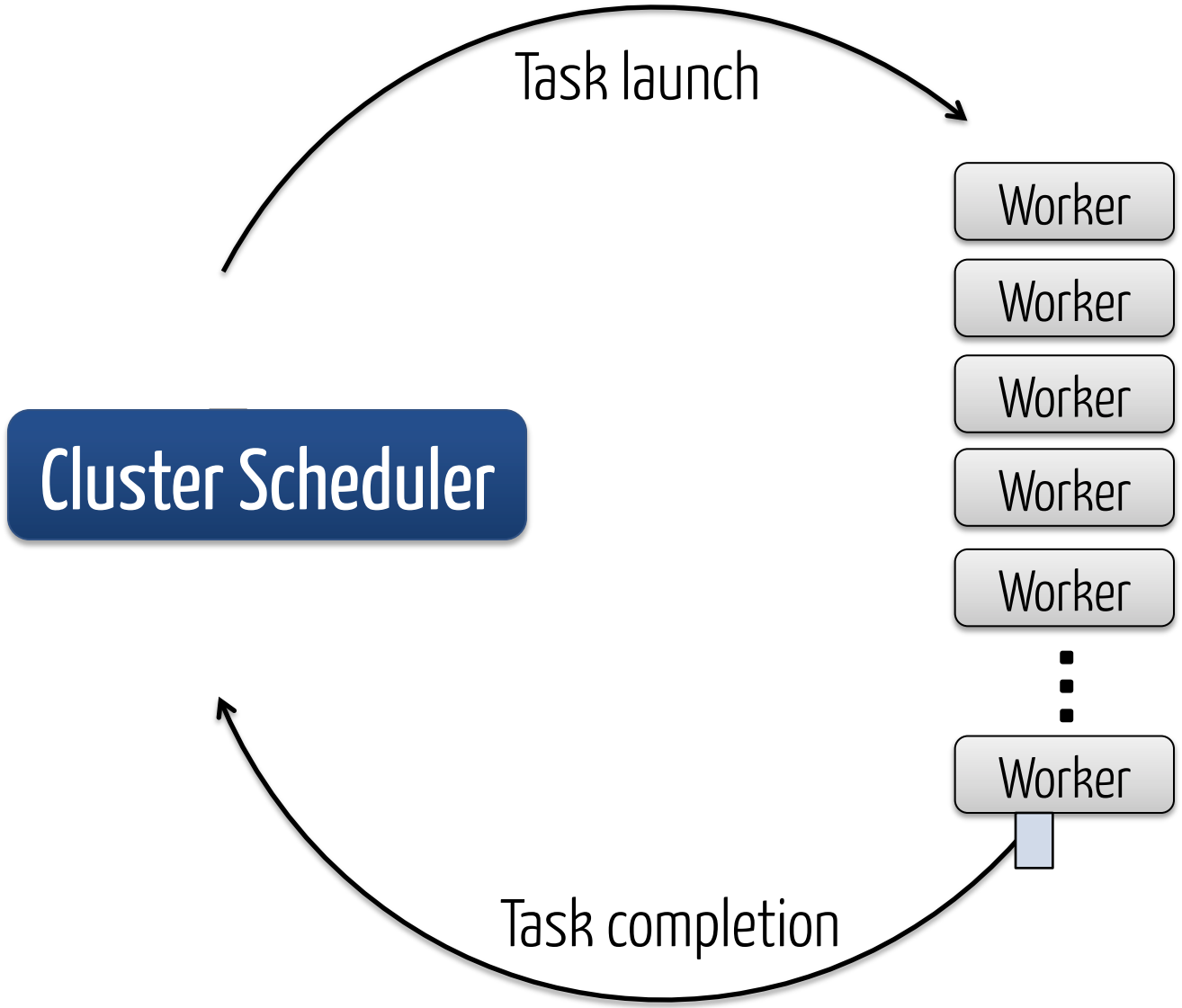
Optimizing the Spark Scheduler

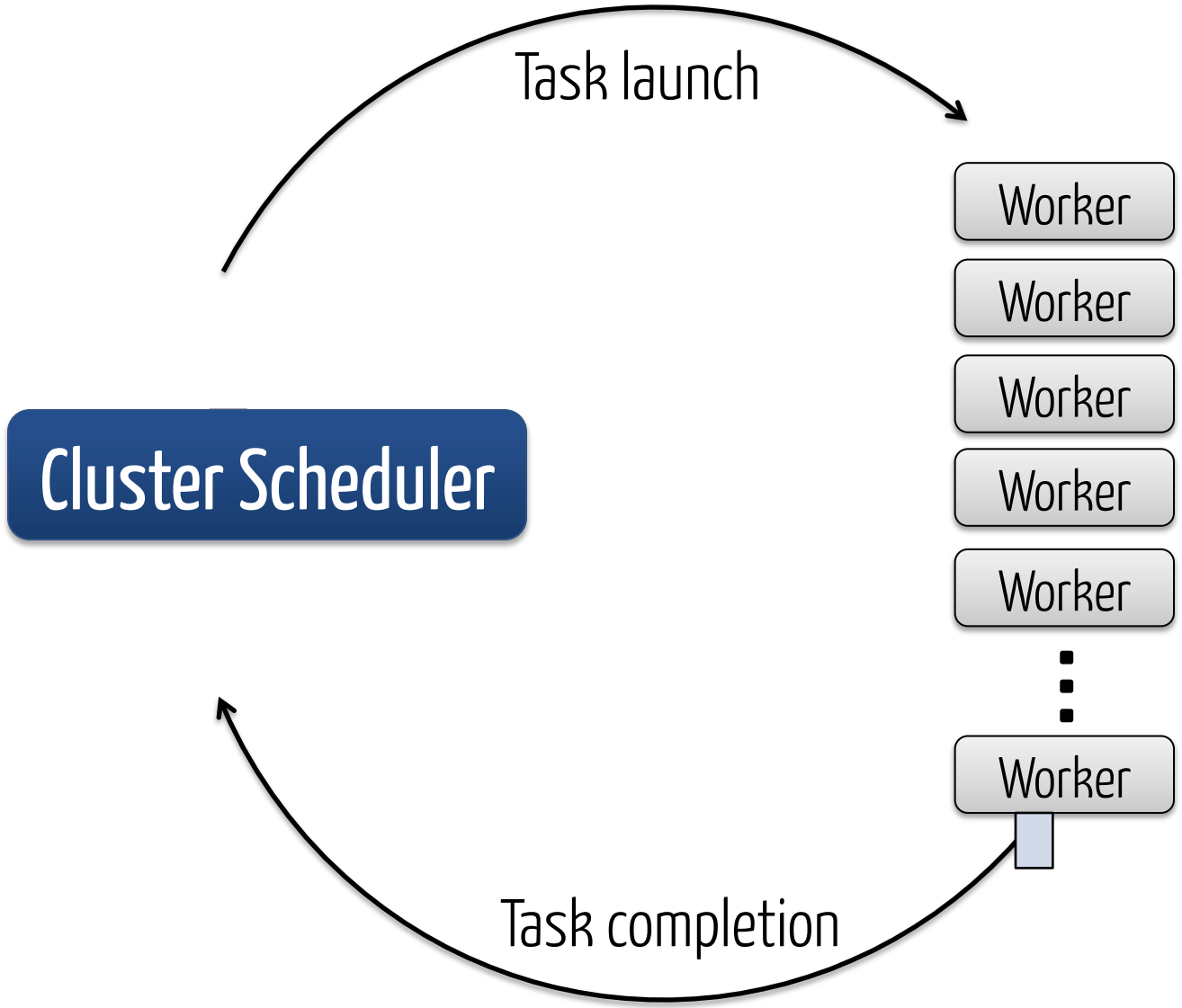
0.8: Monitoring code moved off critical path

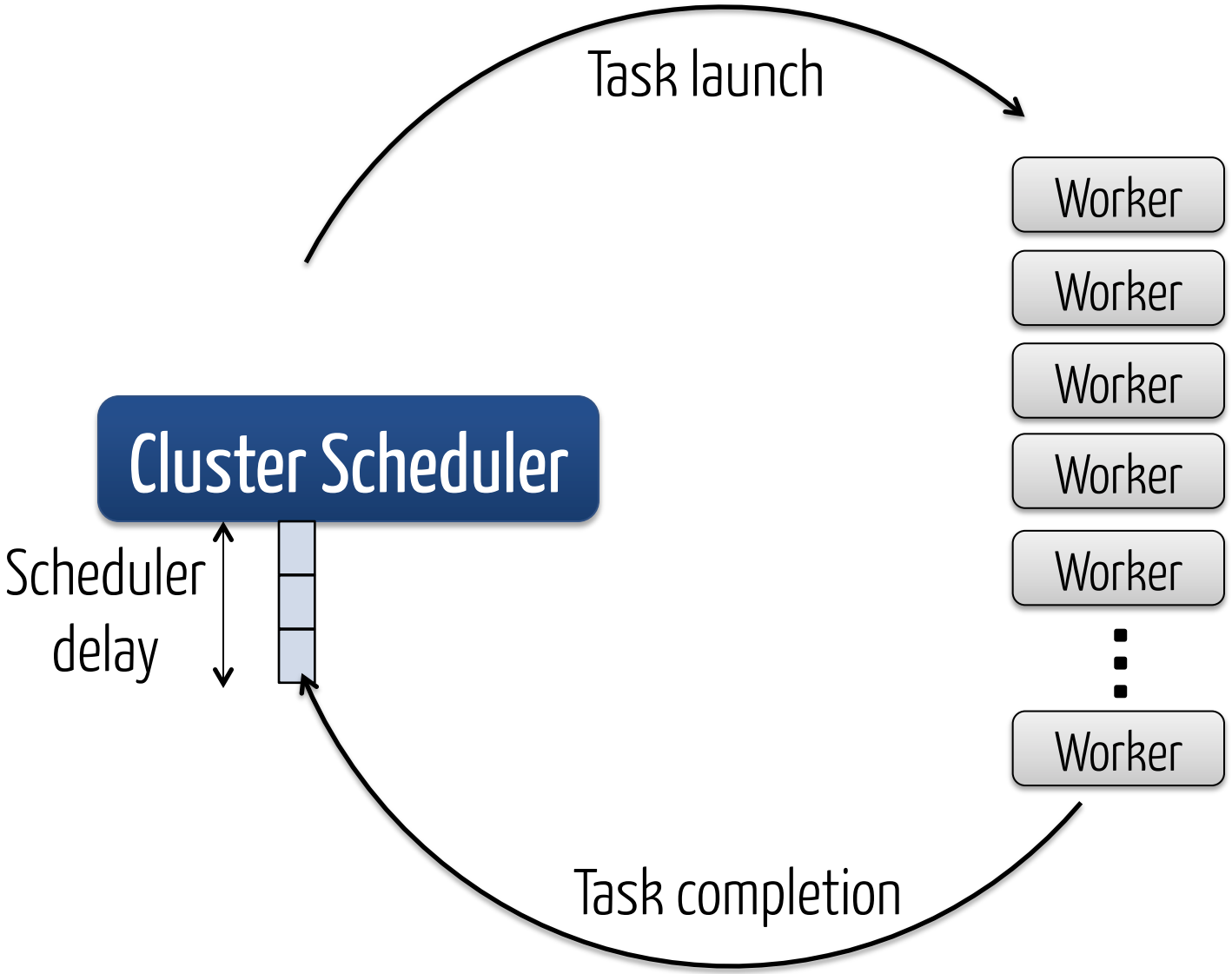
0.8.1: Result deserialization moved off critical path

Future improvements may yield 2-3x higher throughput

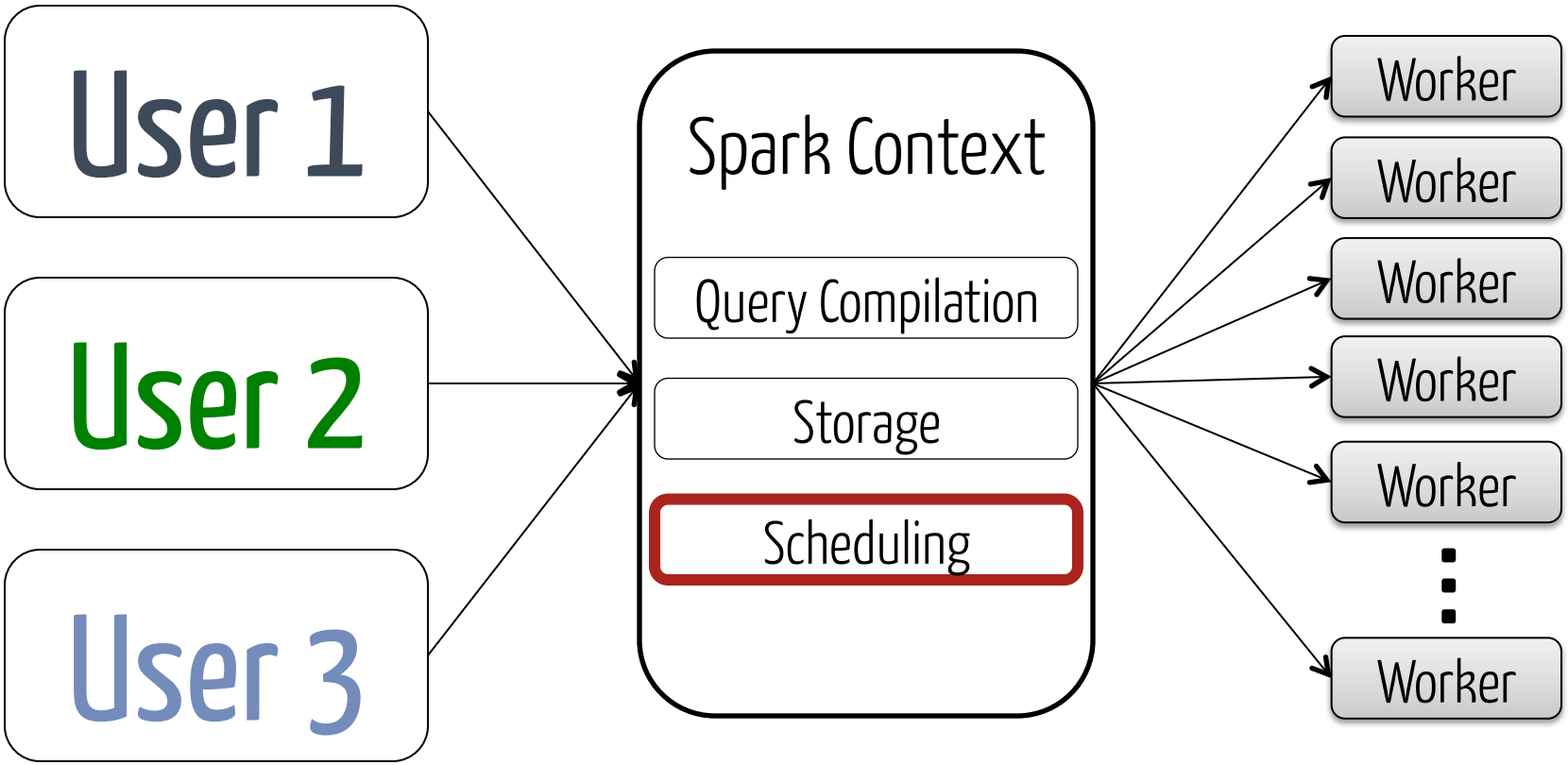
Is the scheduler the bottleneck in my cluster?


Task launch
Cluster Scheduler
Worker
Worker
Worker
Worker
Worker
Worker
Task completion

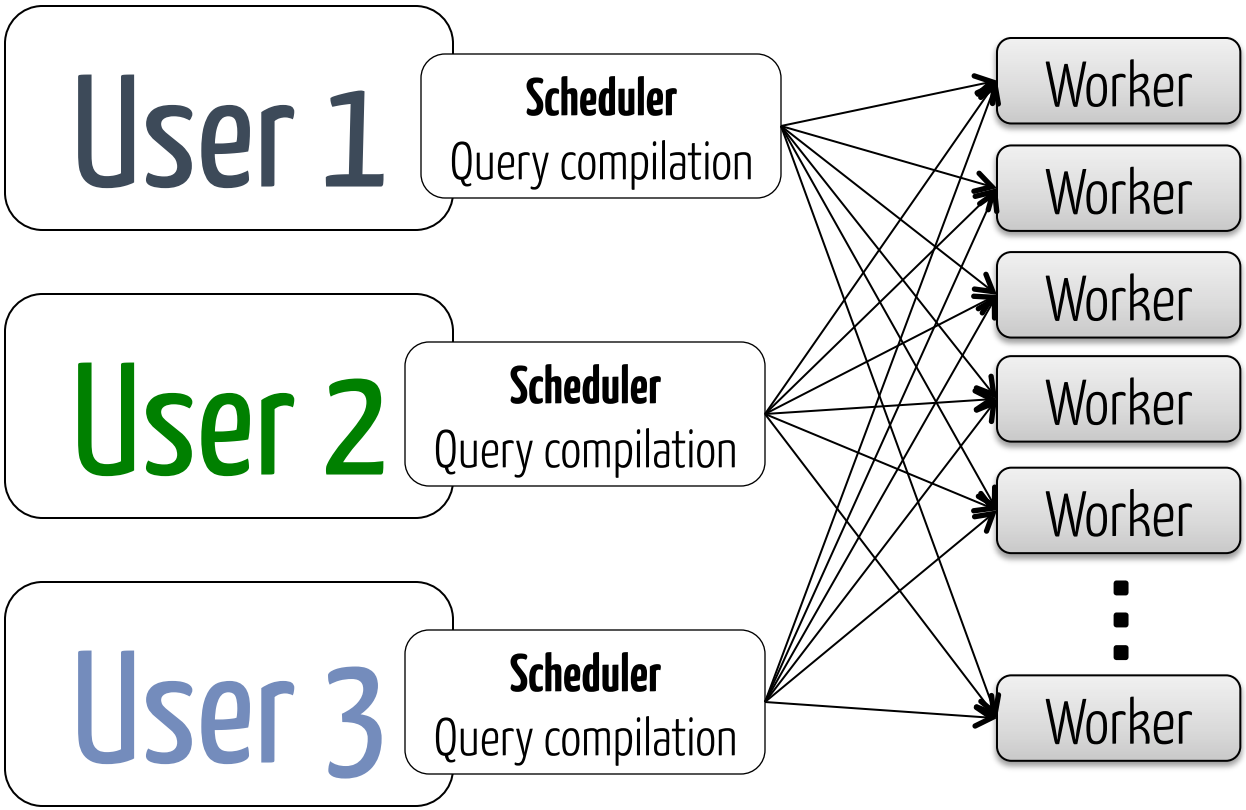

Task launch
Cluster Scheduler
Worker
Worker
Worker
Worker
Worker
Worker
Task completion


Task launch
Worker
Worker
Worker
Worker
Worker
Worker
Cluster Scheduler
Scheduler delay
Task completion

Spark Today


User 1
User 2
User 3
Spark Context
Query Compilation
Storage
Scheduling
Worker
Worker
Worker
Worker
Worker
Worker

Future Spark

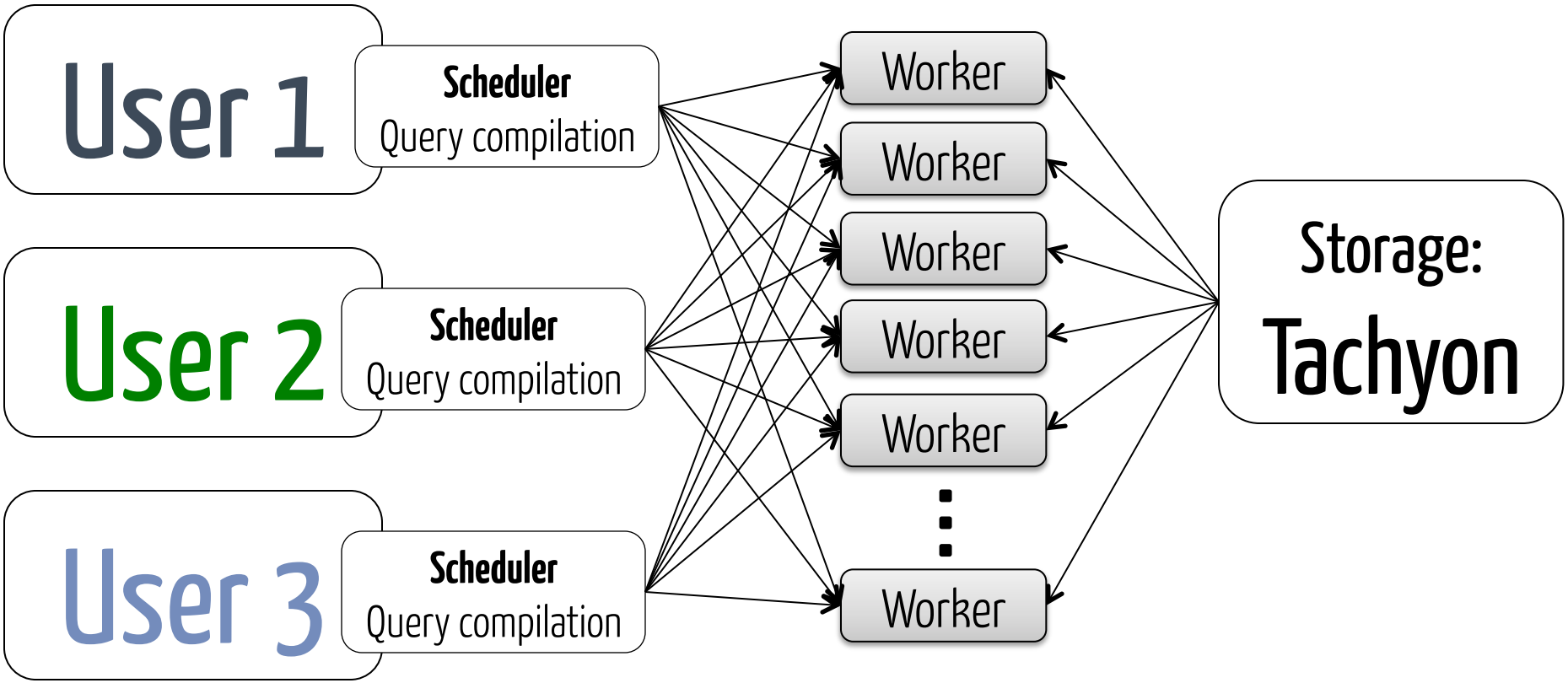


Benefits:

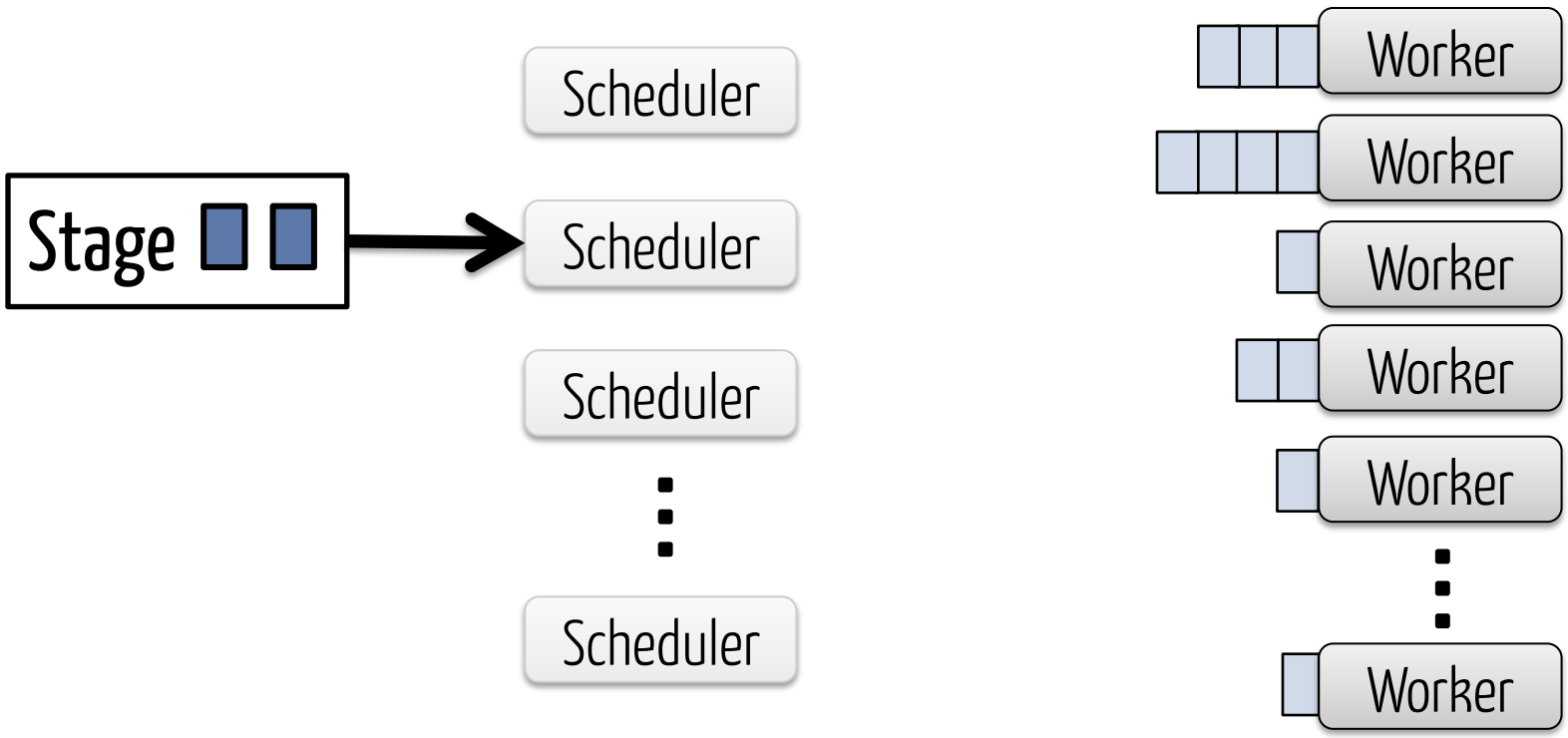
High throughput

Fault tolerance

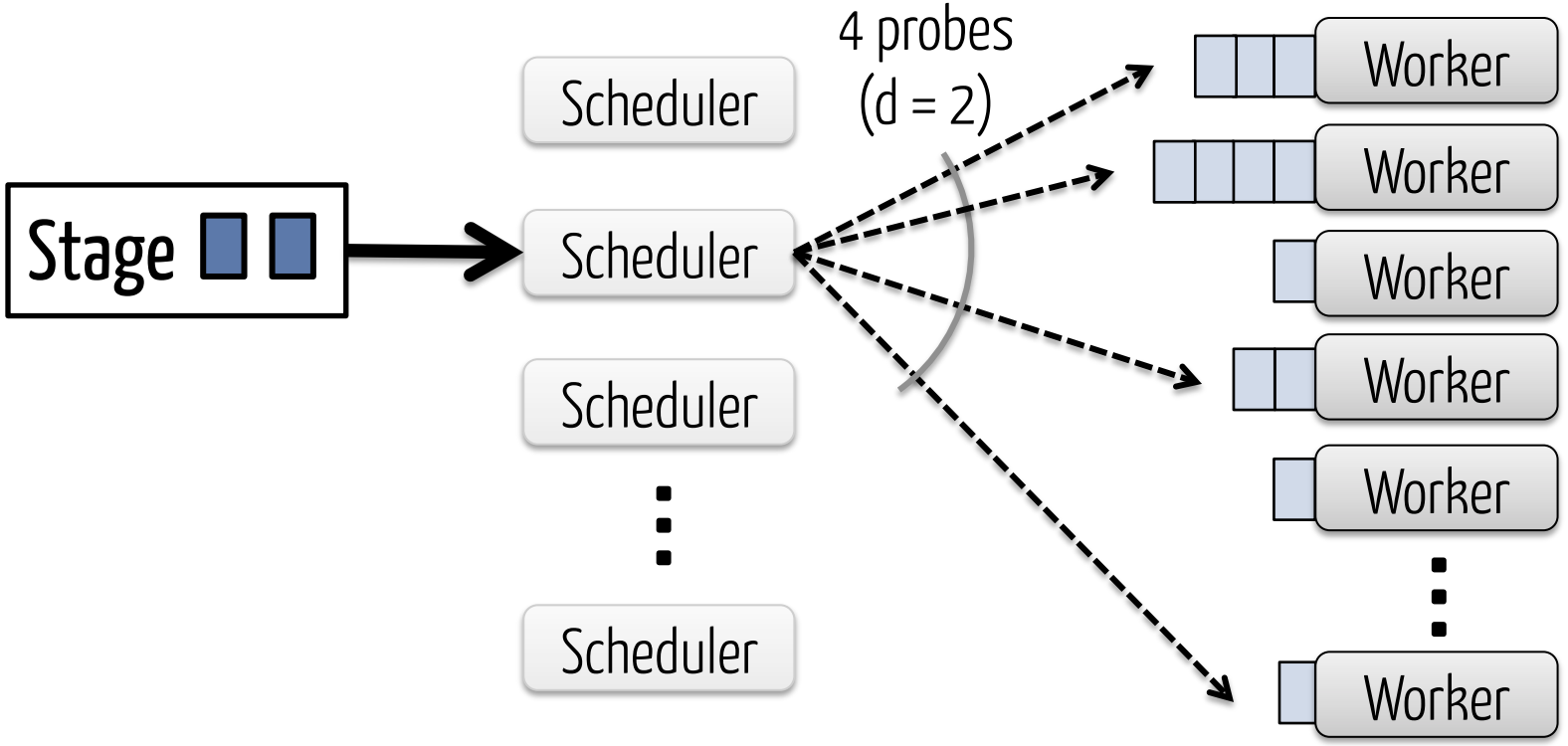
Future Spark



Scheduling with Sparrow

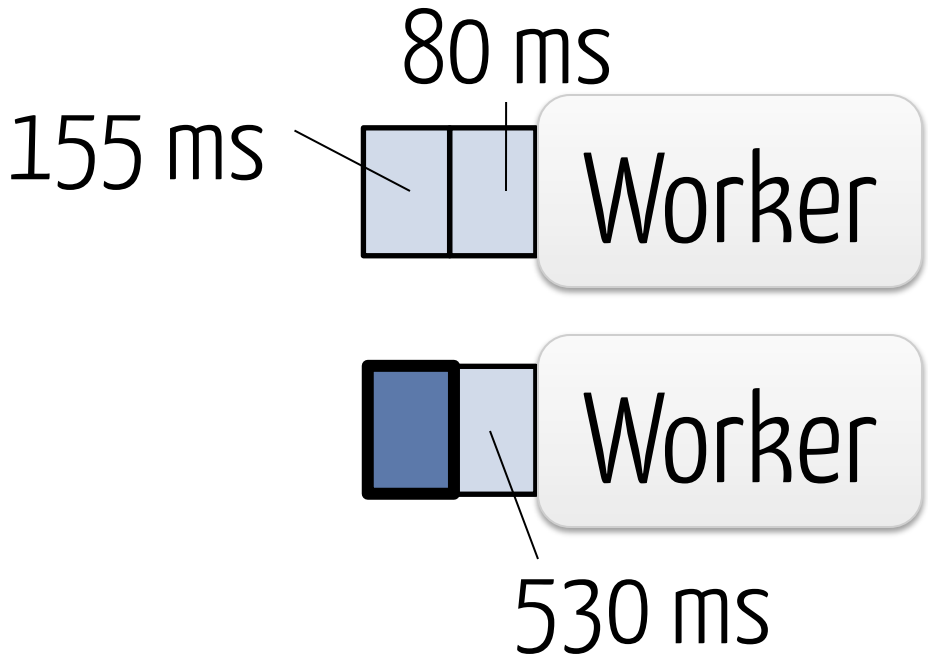


Batch Sampling



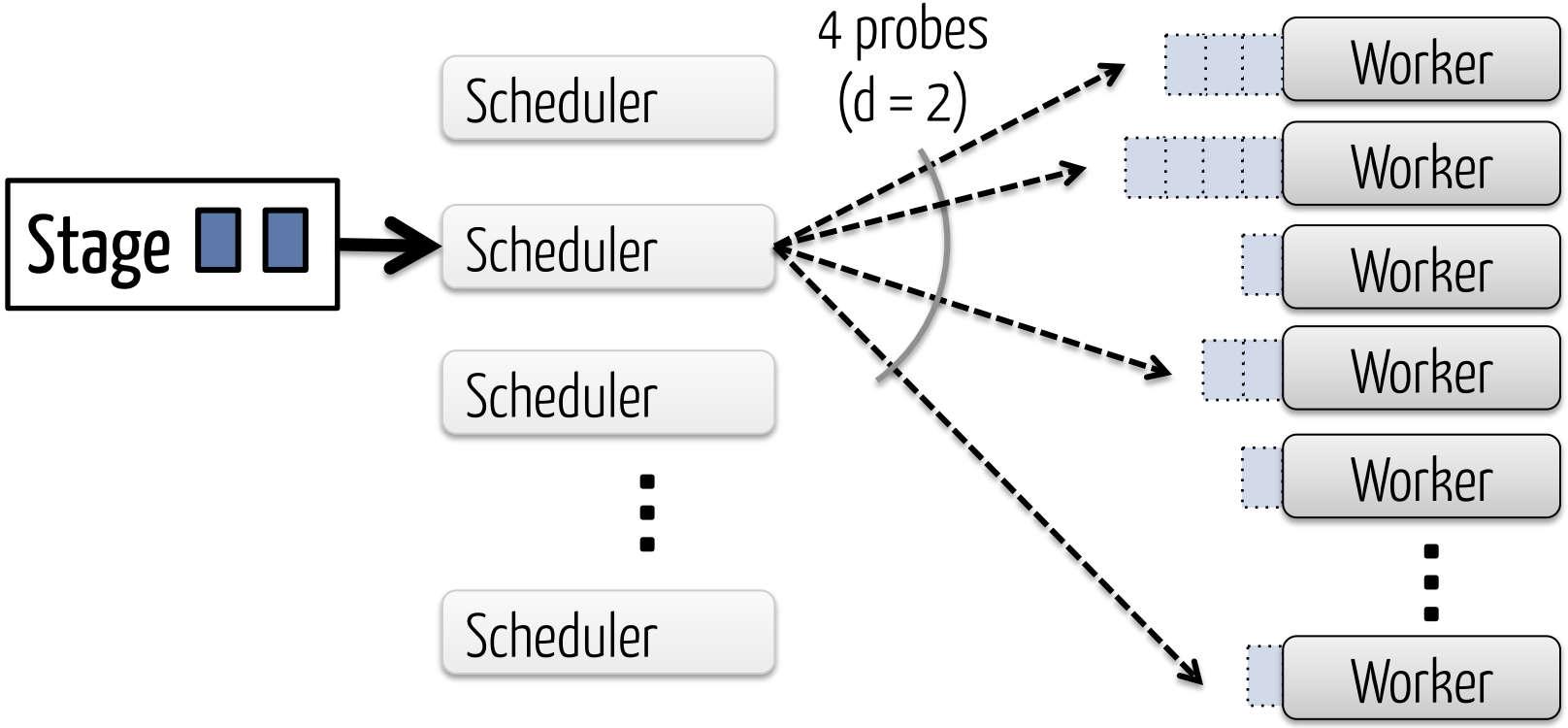
Place m tasks on the least loaded of $d \bullet m$ workers

Queue length poor predictor of wait time



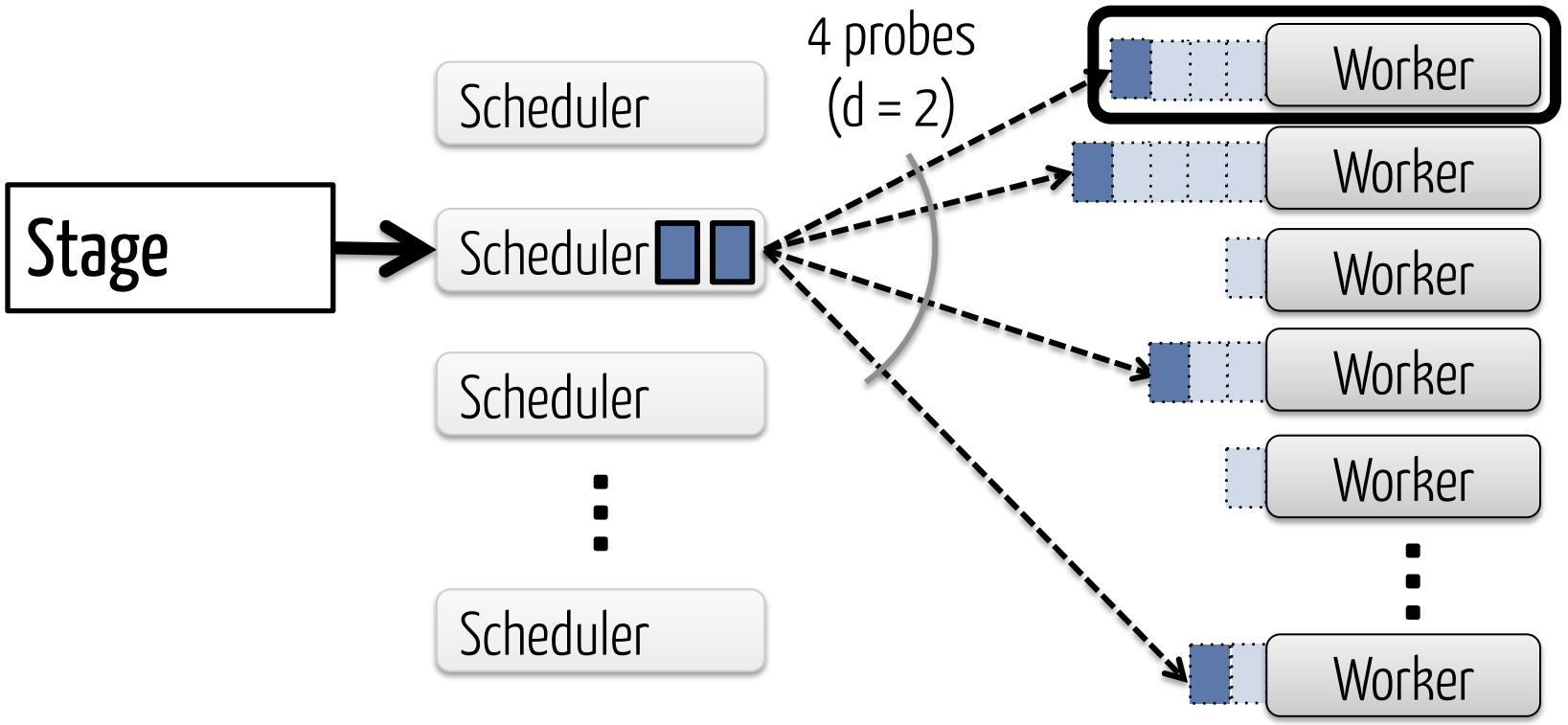
Poor performance on heterogeneous workloads

Late Binding



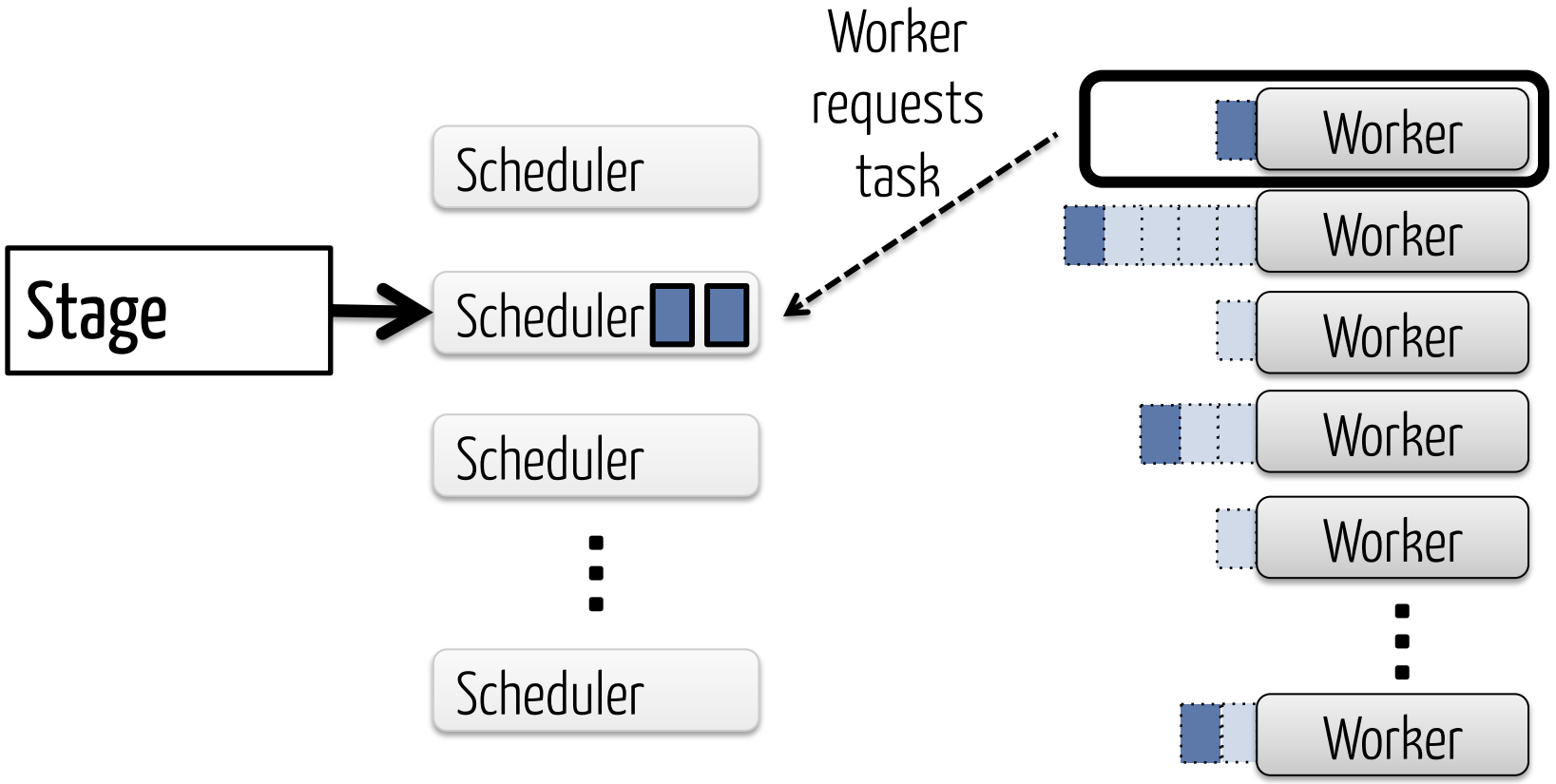
Place *m* tasks on the least loaded of *d*•*m* workers

Late Binding



Place m tasks on the least loaded of $d \bullet m$ workers

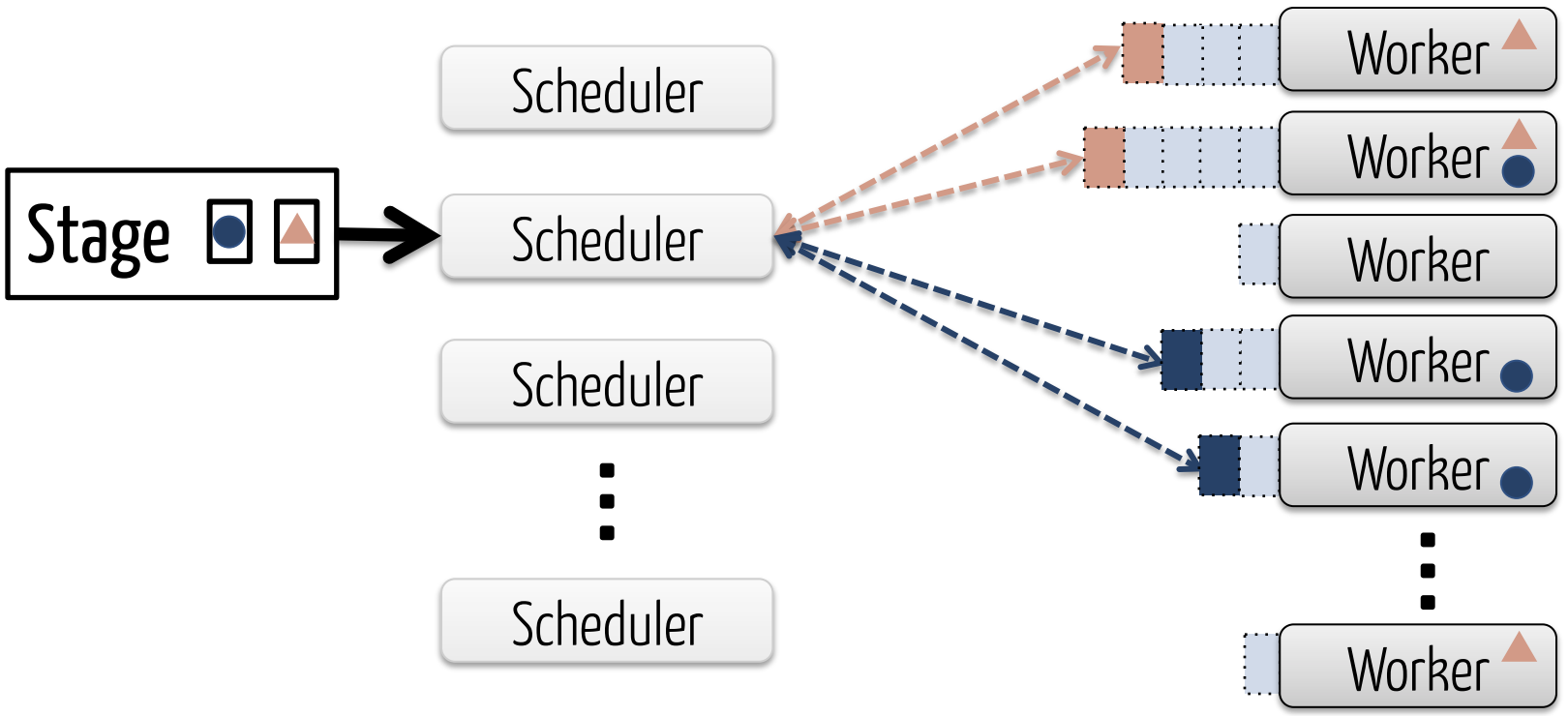
Late Binding



Place m tasks on the least loaded of $d \bullet m$ workers

What about constraints?

Per-Task Constraints



Probe separately for each task

Technique Recap

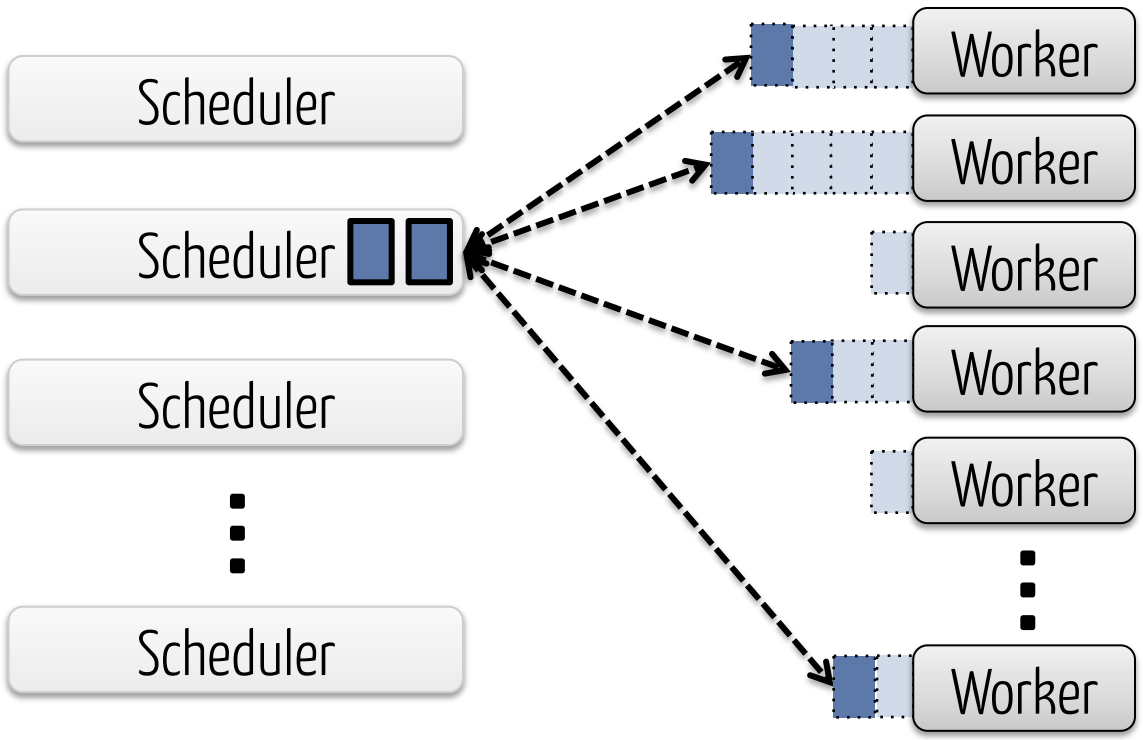
Batch sampling

+

Late binding

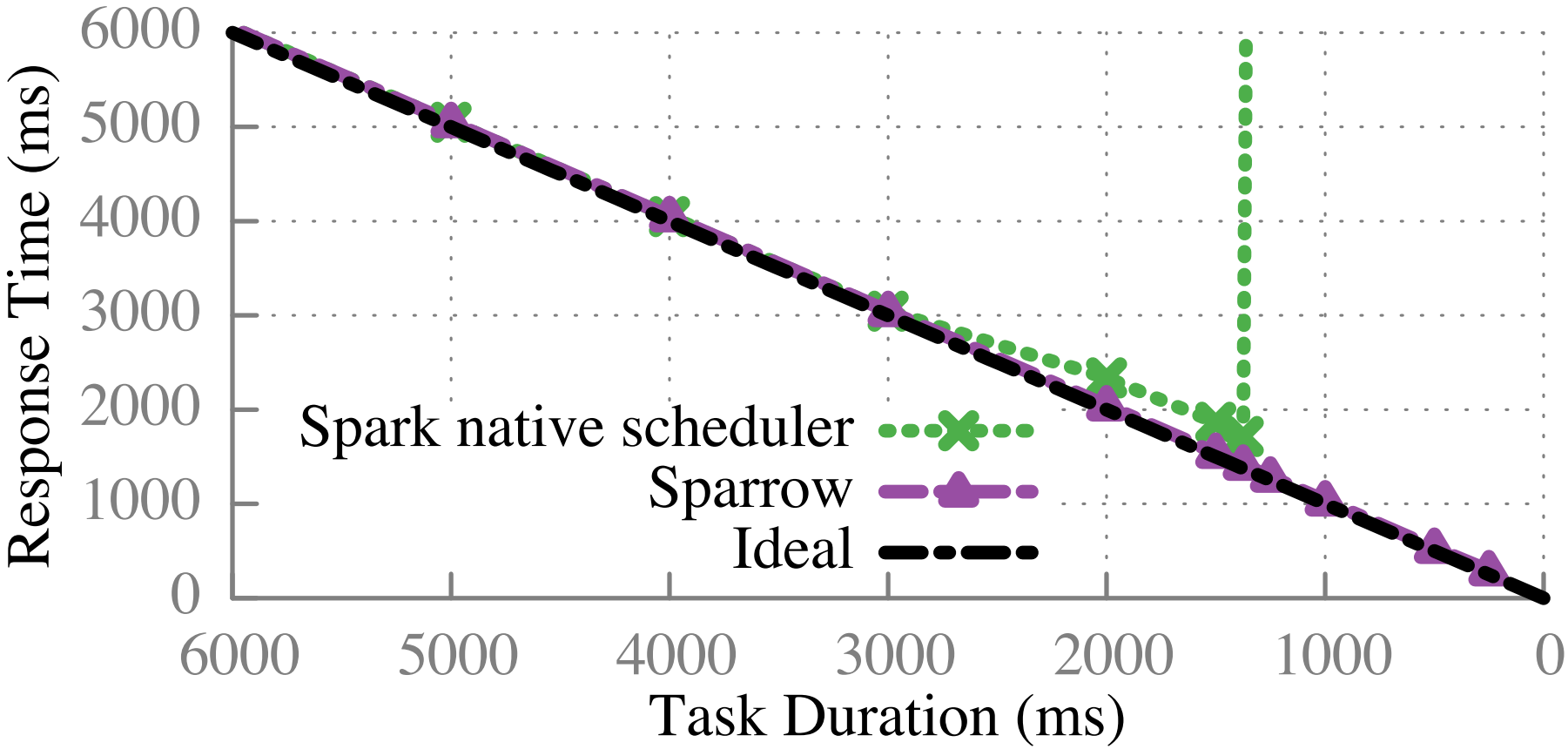
+

Constraints



How well does Sparrow perform?

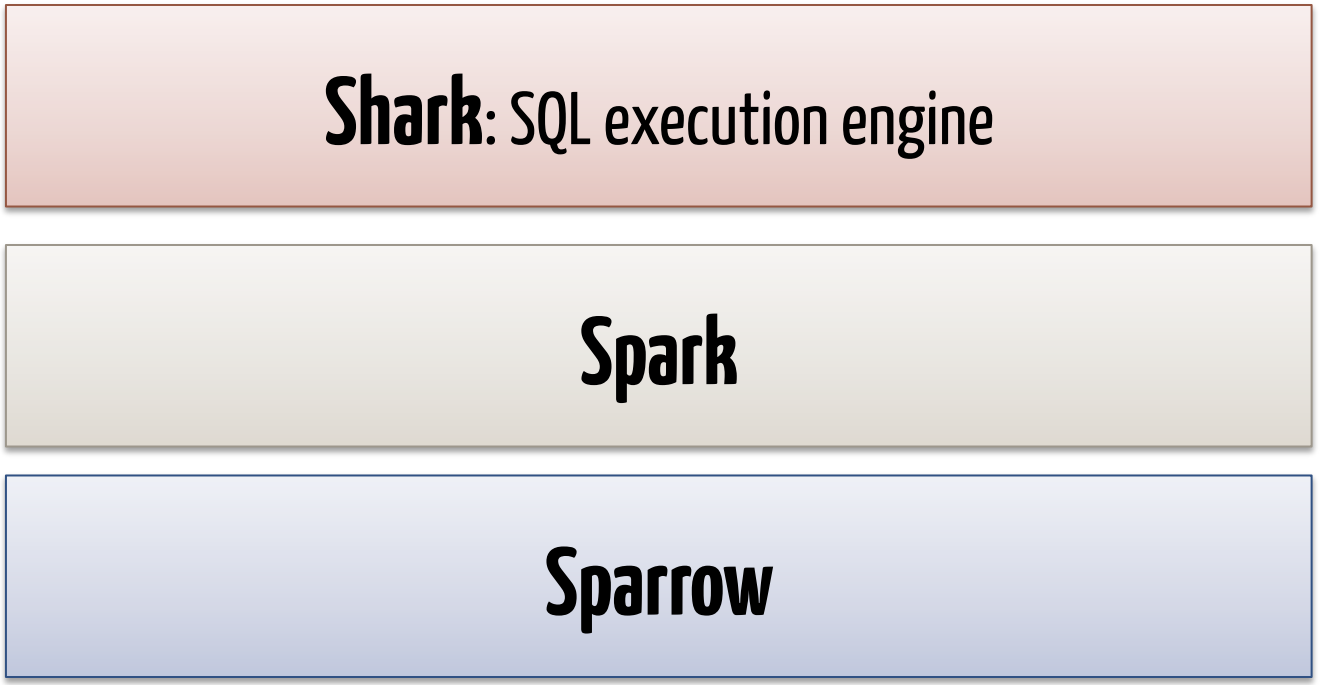
How does Sparrow compare to Spark's native scheduler?



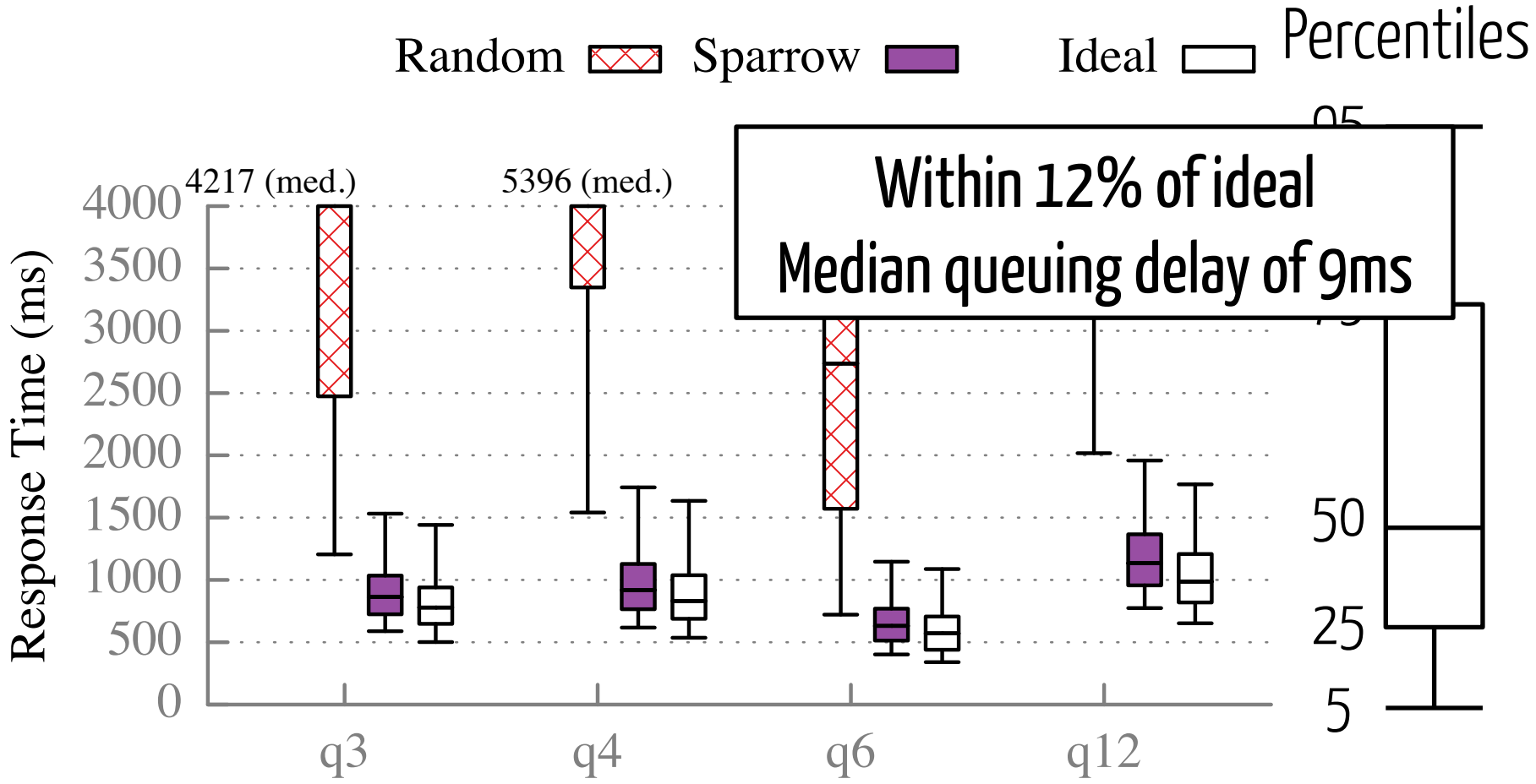
100 16-core EC2 nodes, 10 tasks/job, 10 schedulers, 80% load

TPC-H Queries: Background

TPC-H: Common benchmark for analytics workloads



TPC-H Queries

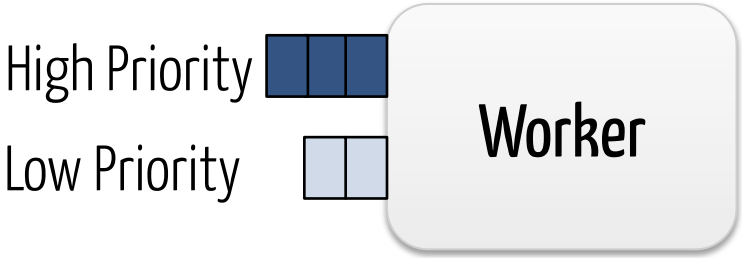


100 16-core EC2 nodes, 10 schedulers, 80% load

Policy Enforcement

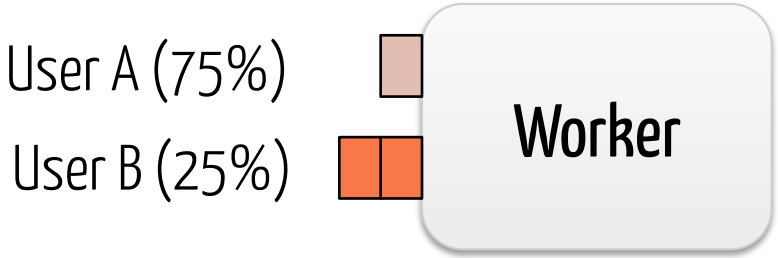
Priorities

Serve queues based on strict priorities

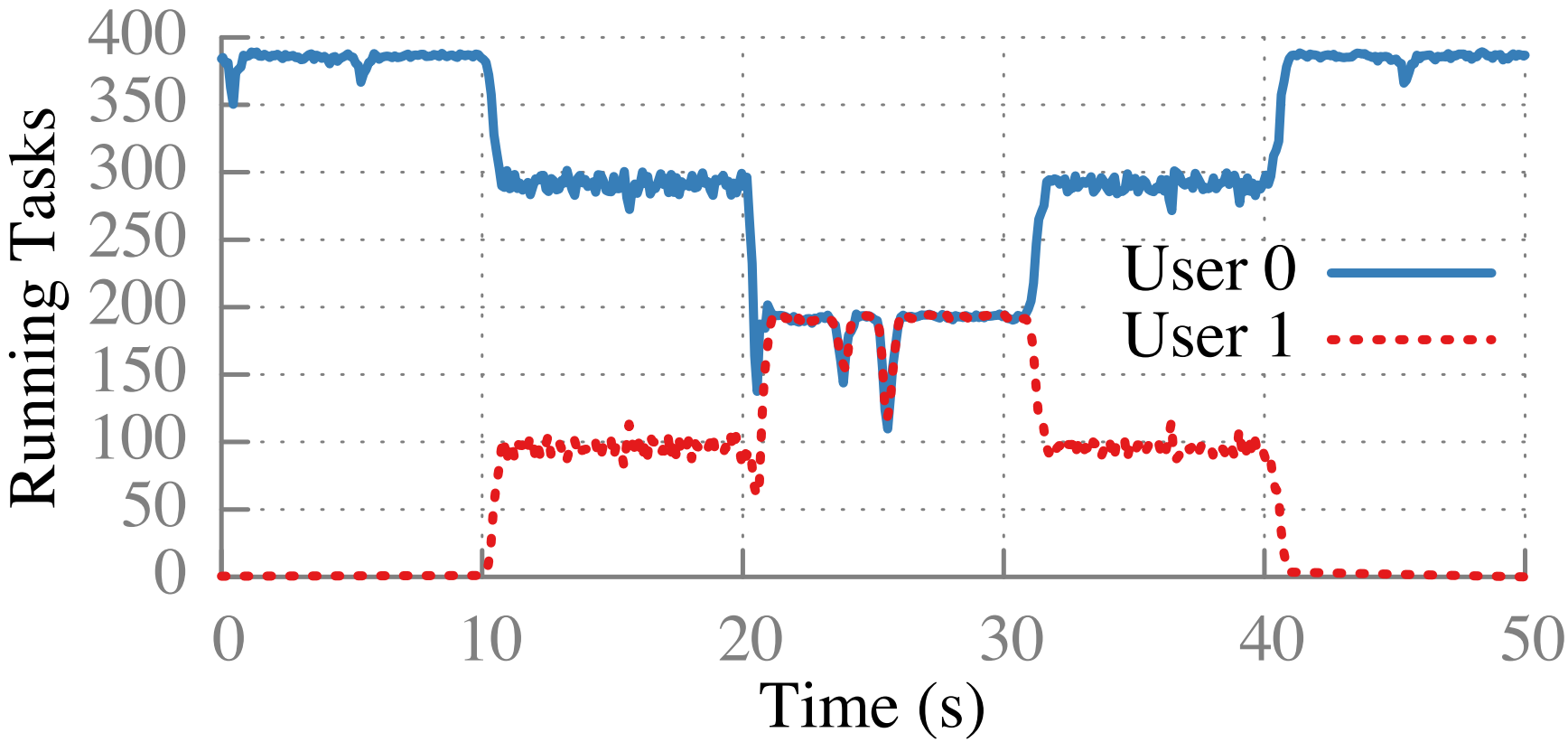


Fair Shares

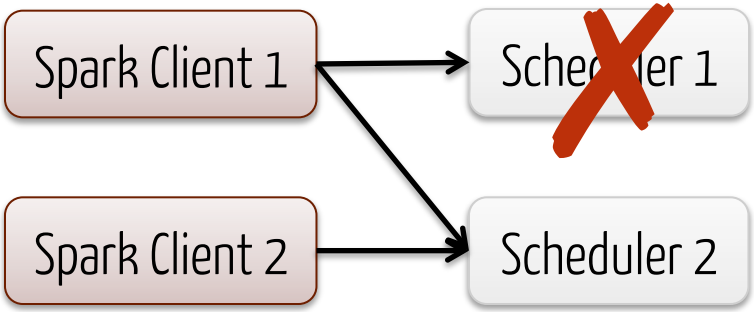
Serve queues using weighted fair queuing

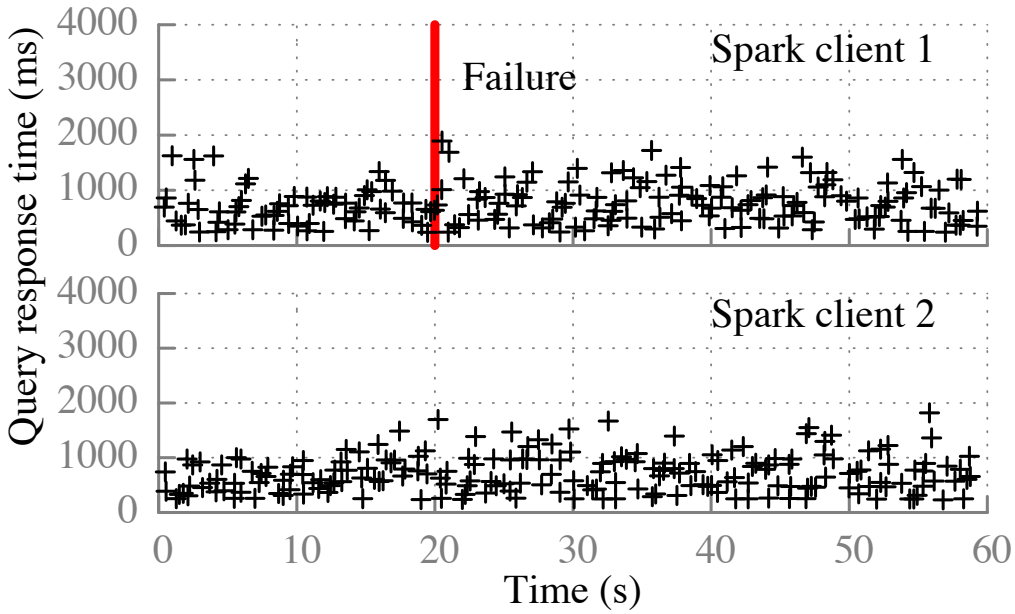


Weighted Fair Sharing



Fault Tolerance





Timeout: 100ms

Failover: 5ms

Re-launch queries: 15ms

Making Sparrow feature-complete

Interfacing with UI

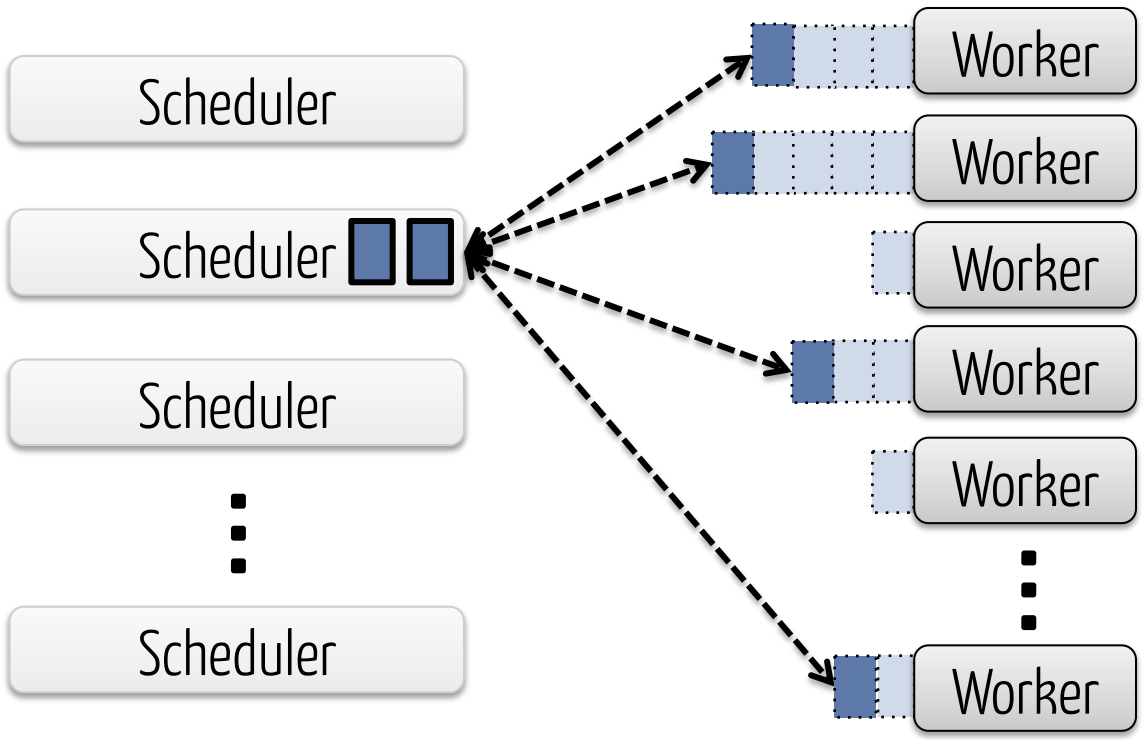
Delay scheduling

Speculation

(1) Diagnosing a Spark scheduling bottleneck

Metric	Min	25th percentile	Median	75th percen
Duration	1 ms	1 ms	1 ms	1 ms
Time spent fetching task results	0 ms	0 ms	0 ms	0 ms
Scheduler delay	30 ms	81 ms	91 ms	95 ms

(2) Distributed, fault-tolerant scheduling with Sparrow



www.github.com/radlab/sparrow