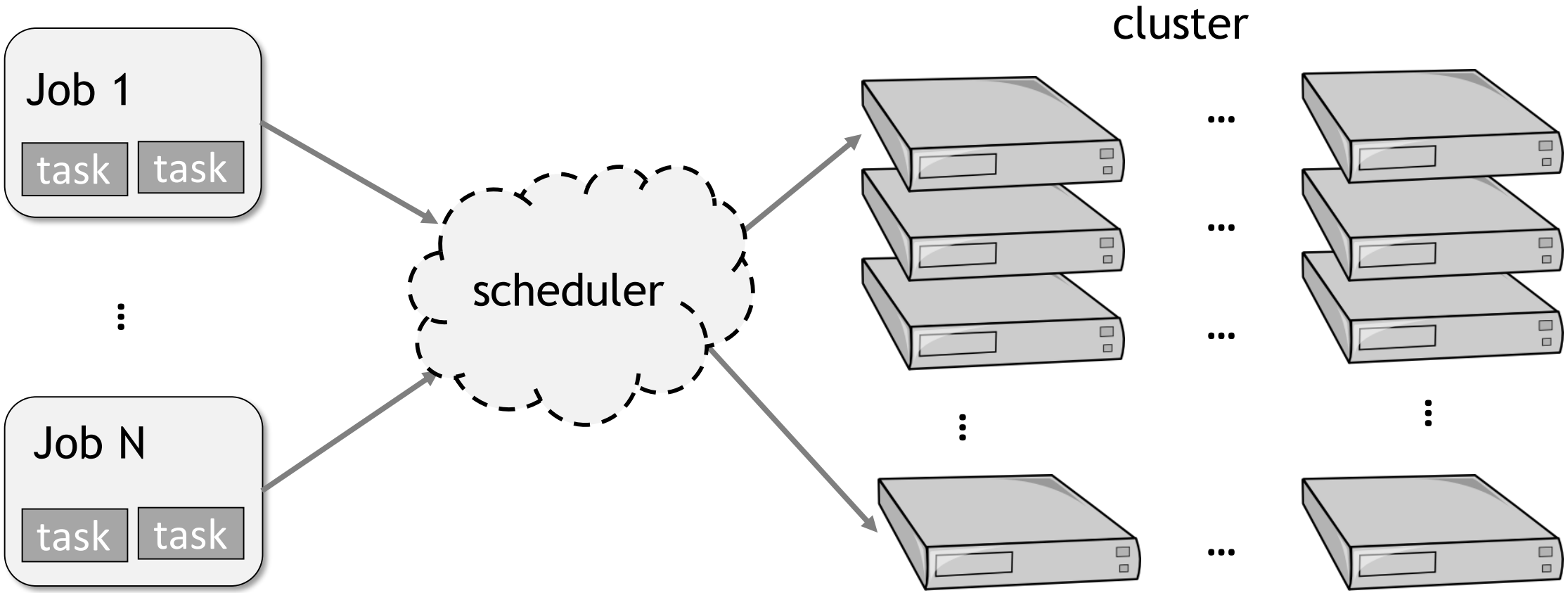
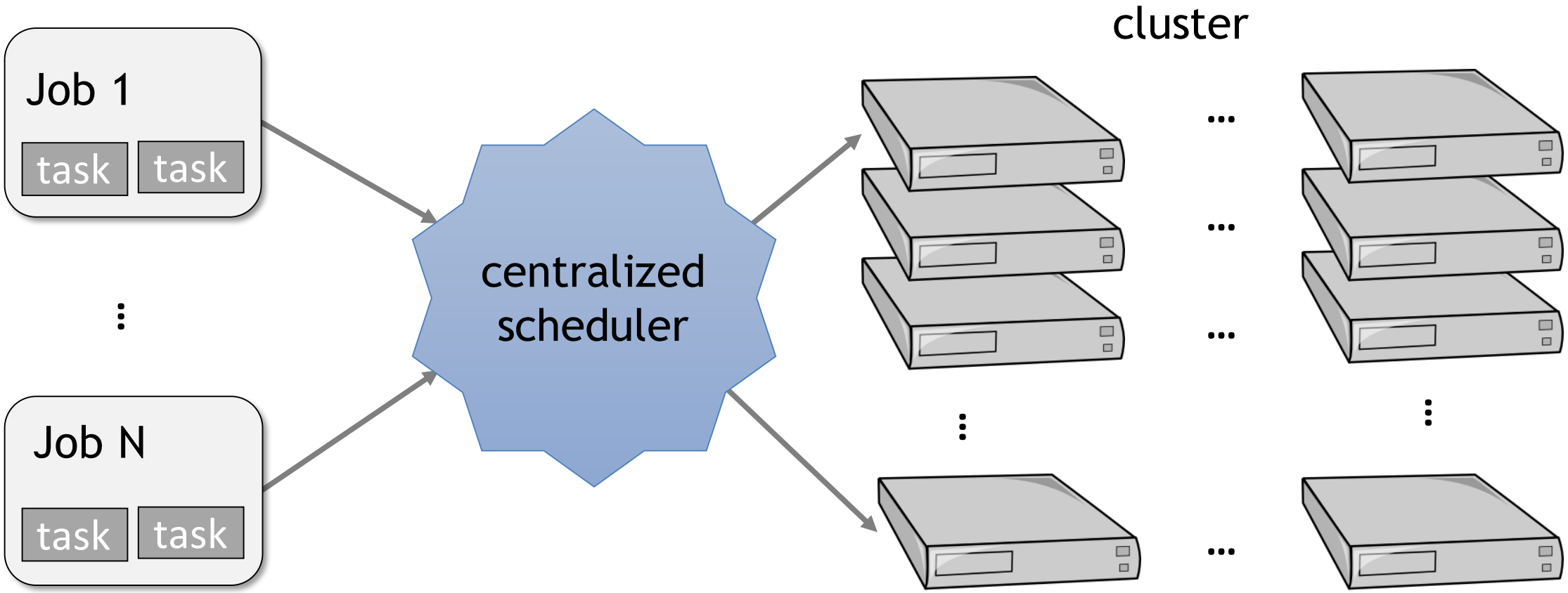
Hawk: Hybrid Datacenter Scheduling

Pamela Delgado, Florin Dinu,
Anne-Marie Kermarrec, Willy Zwaenepoel

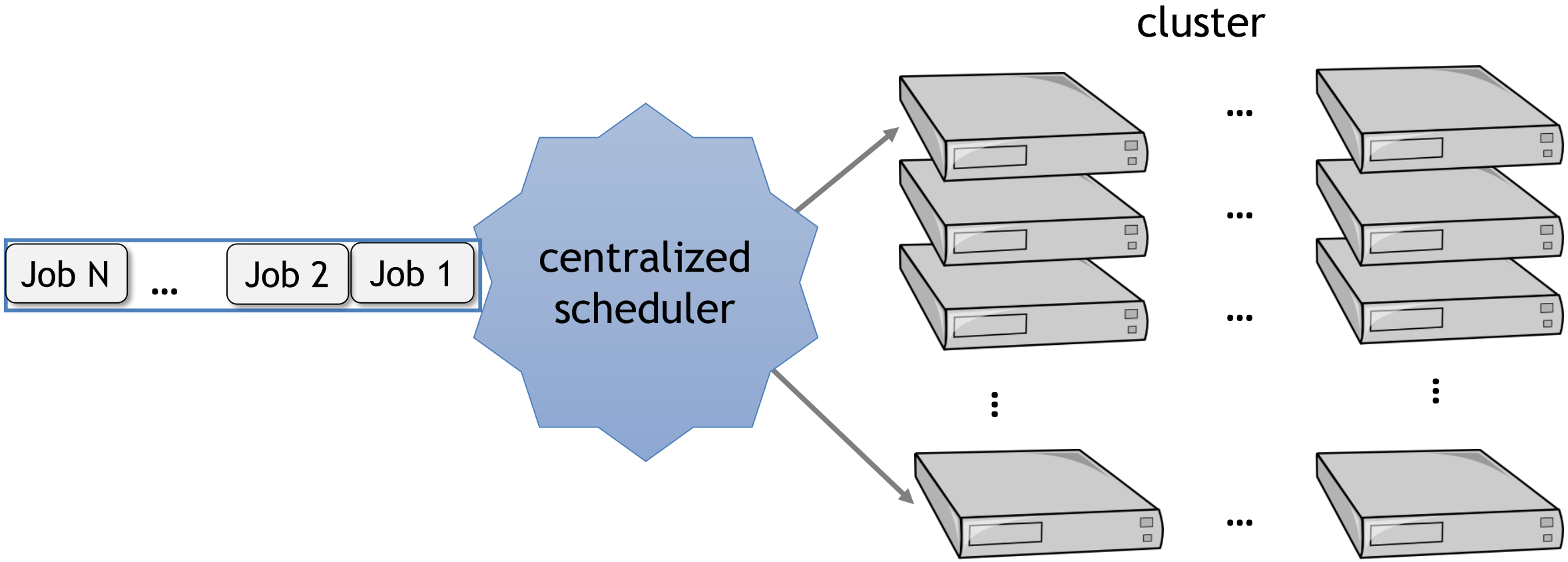
Introduction: datacenter scheduling



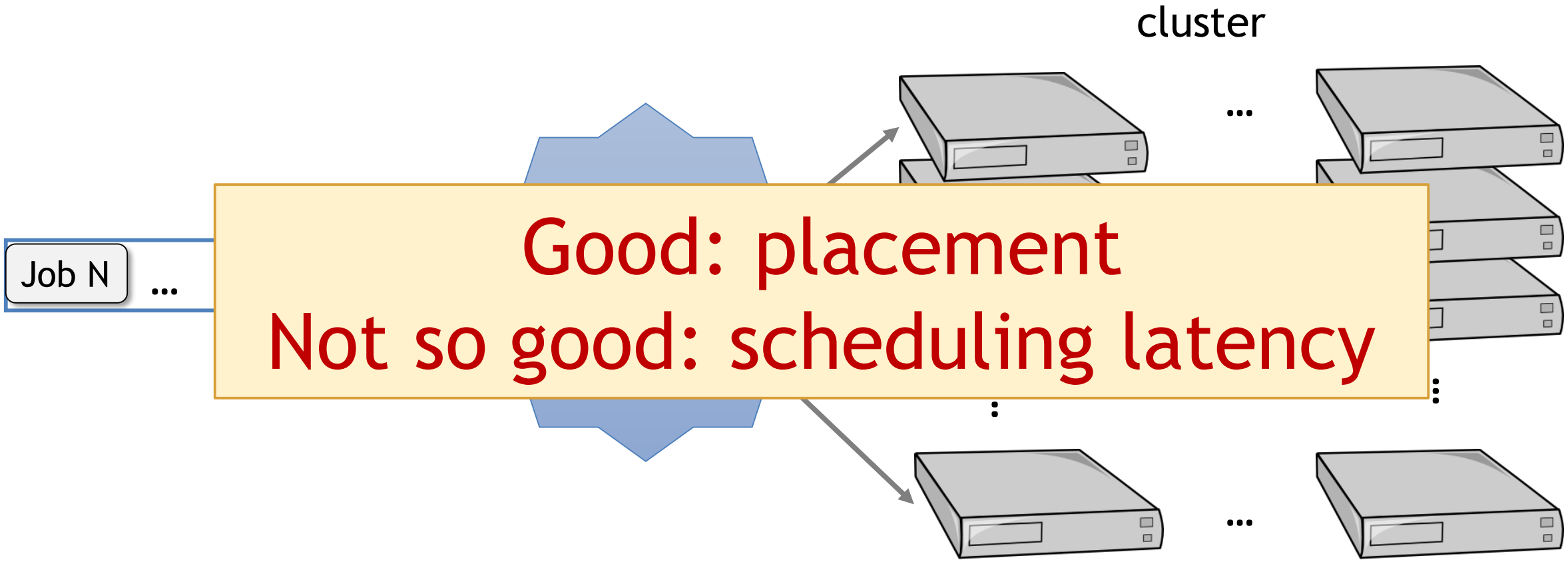
Introduction: centralized scheduling



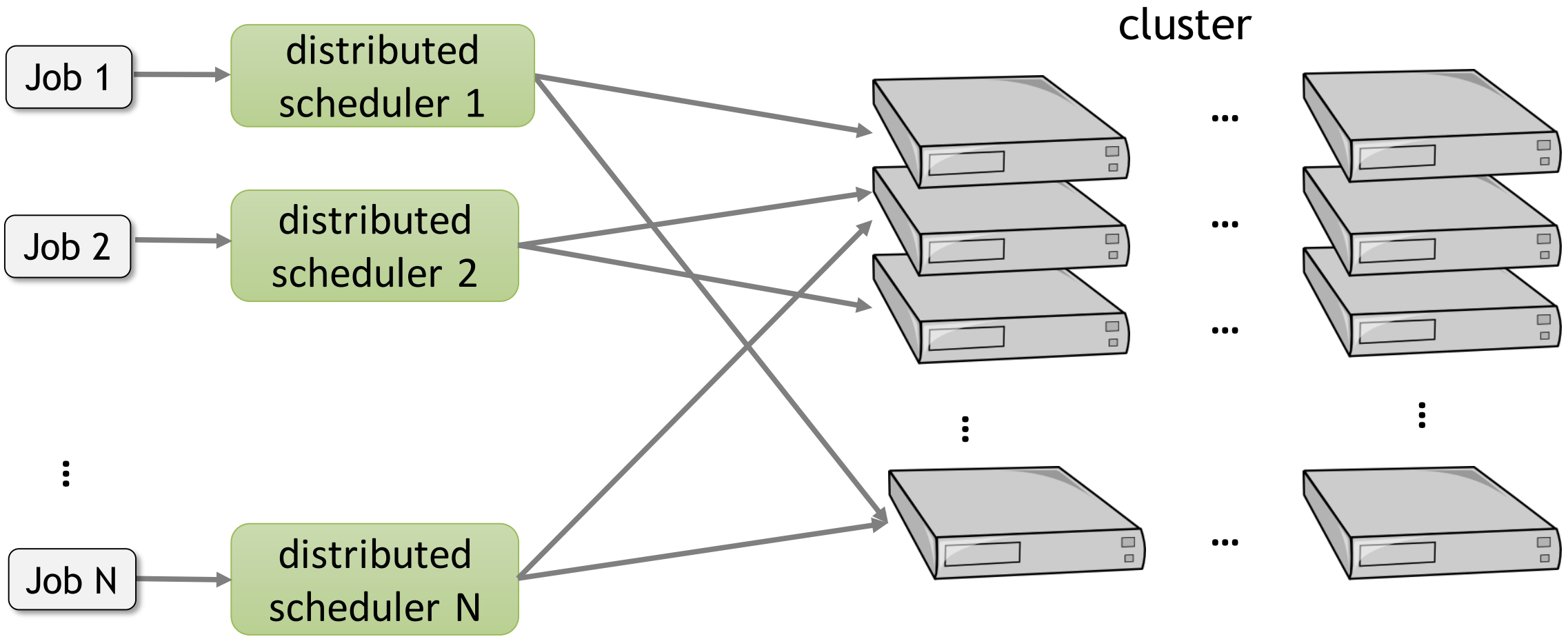
Introduction: centralized scheduling



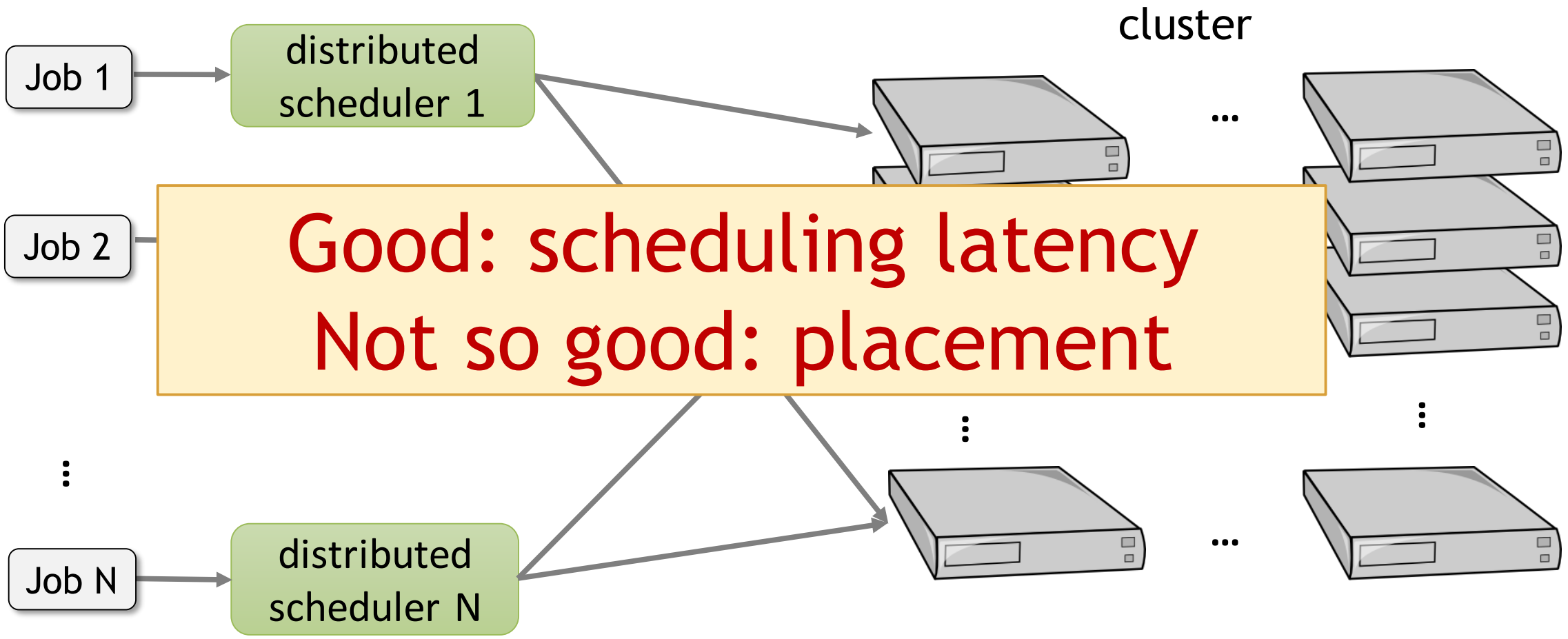
Introduction: centralized scheduling



Introduction: distributed scheduling



Introduction: distributed scheduling



Outline

1) Introduction

2) HAWK hybrid scheduling

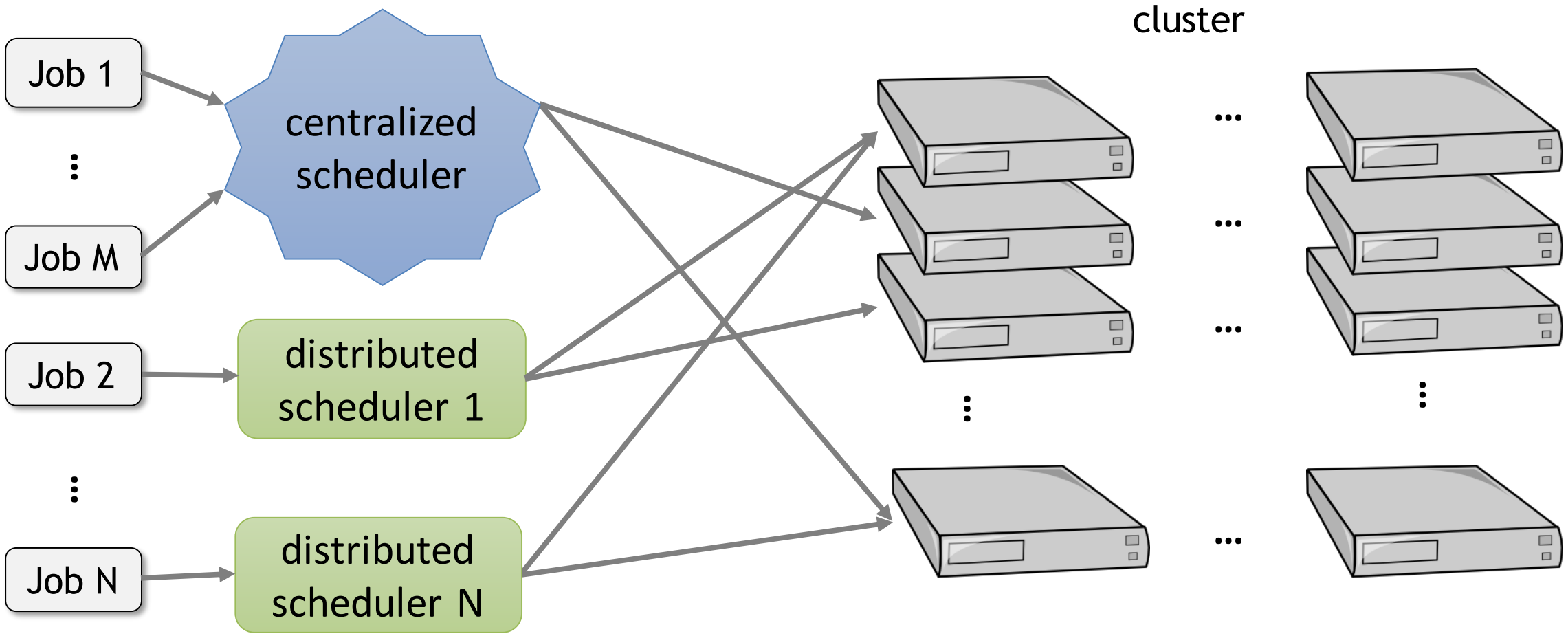
- Rationale
- Design

3) Evaluation

- Simulation
- Real cluster implementation

4) Conclusion

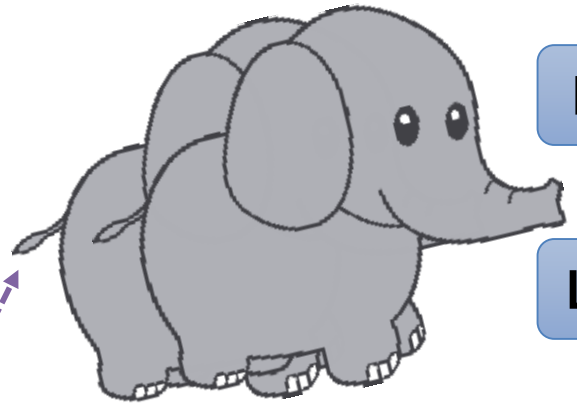
Hybrid scheduling



Hawk: hybrid scheduling

- ✓ Long jobs → centralized
- ✓ Short jobs → distributed

Hawk: hybrid scheduling

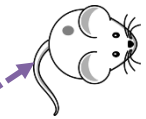


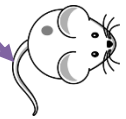
Long/short:
estimated execution time vs cut-off

Long job 1

Long job M

centralized scheduler





Short job 1

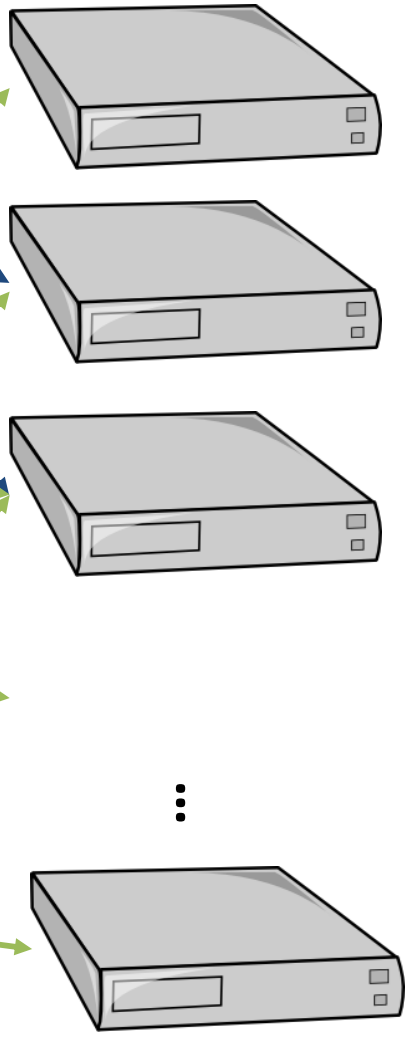
Short job 2

Short job N

distributed scheduler 1

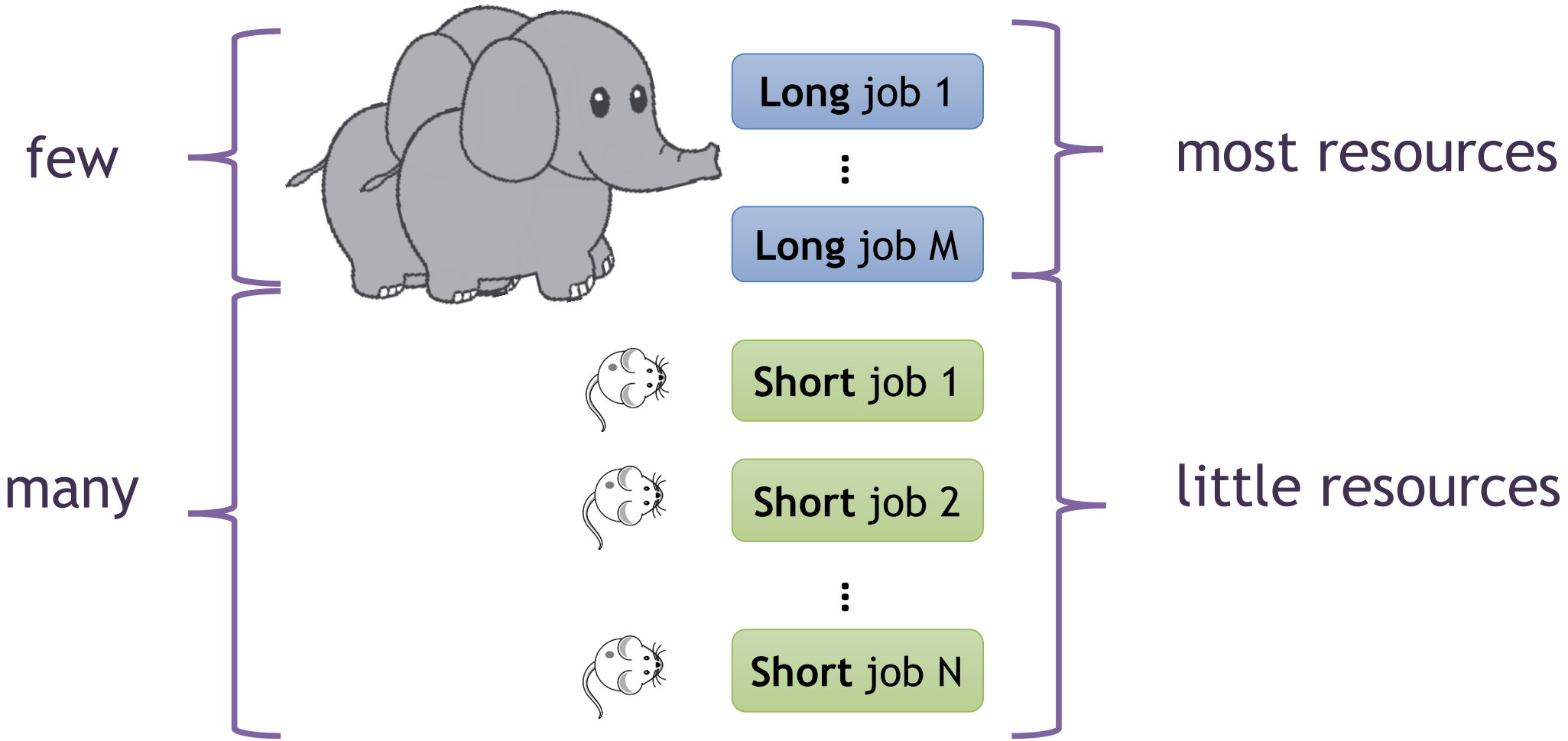
distributed scheduler 2

distributed scheduler N

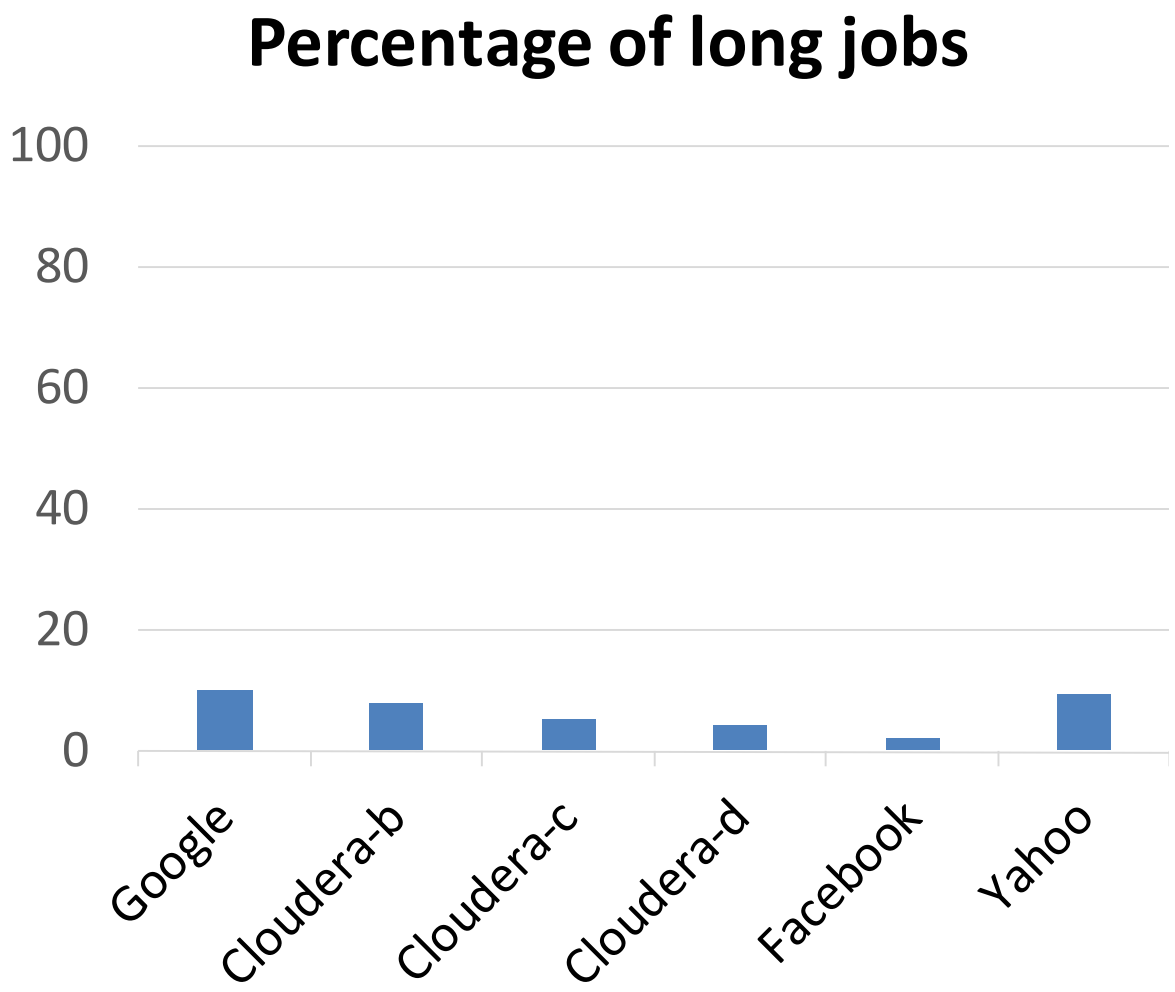


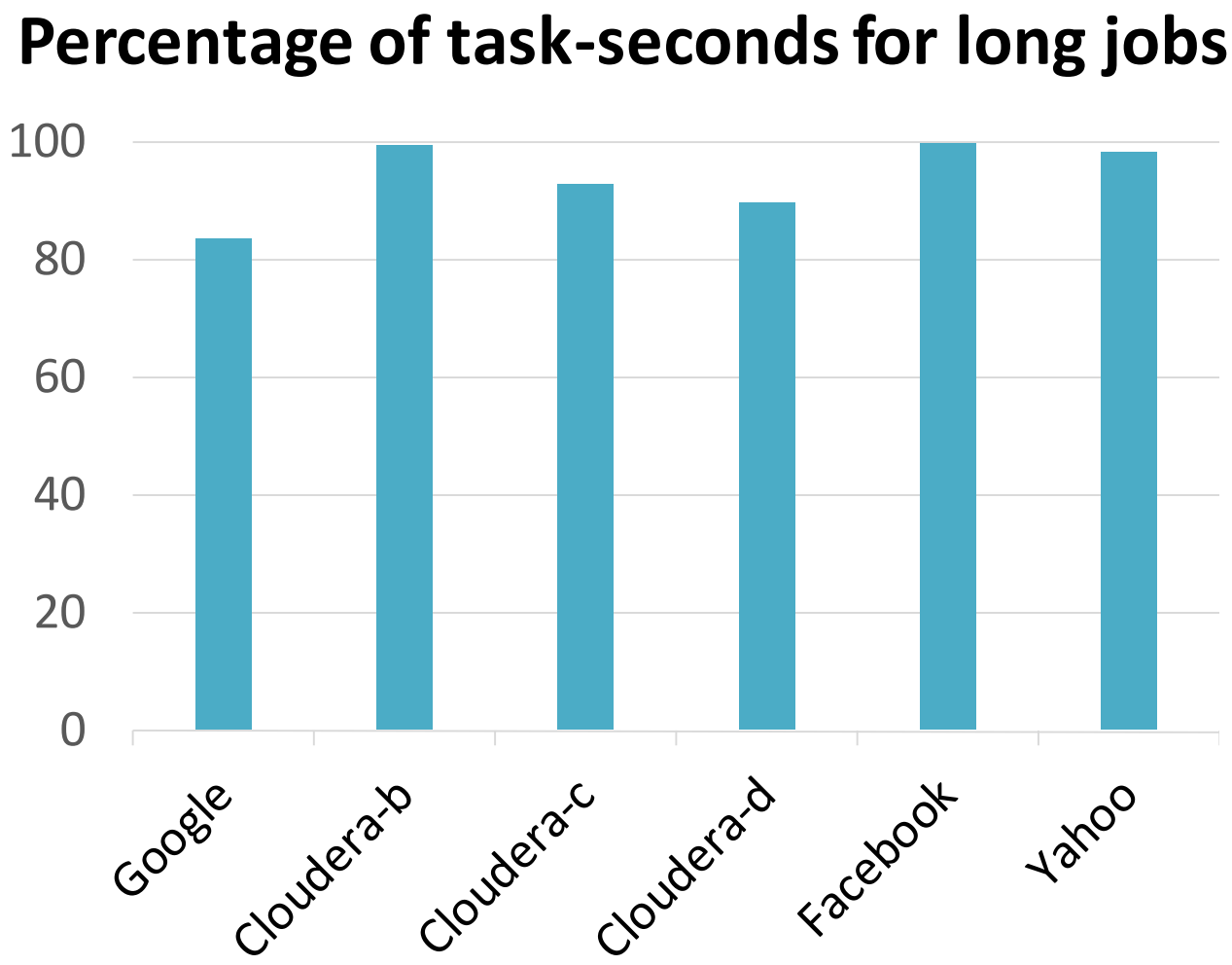
Rationale for Hawk

Typical production workloads



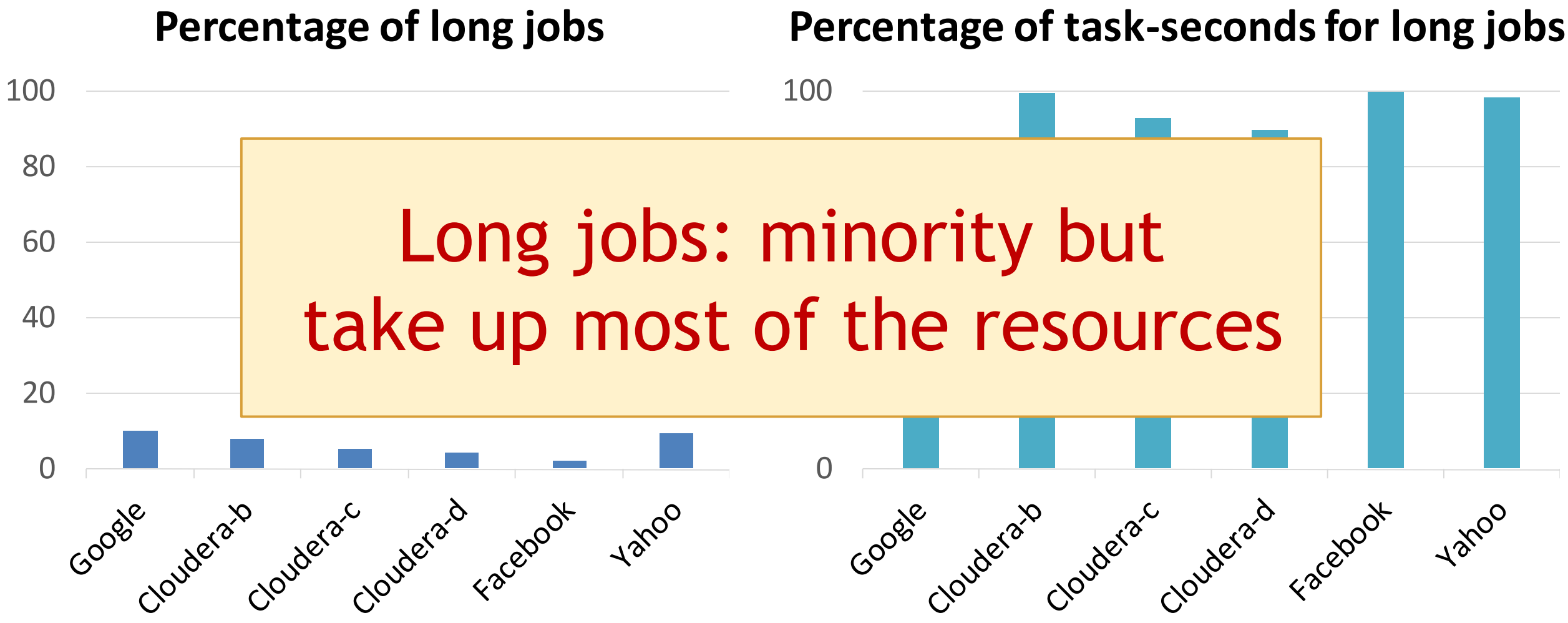
Rationale for Hawk (continued)





Source: Design Insights for MapReduce from Diverse Production Workloads, Chen et al 2012

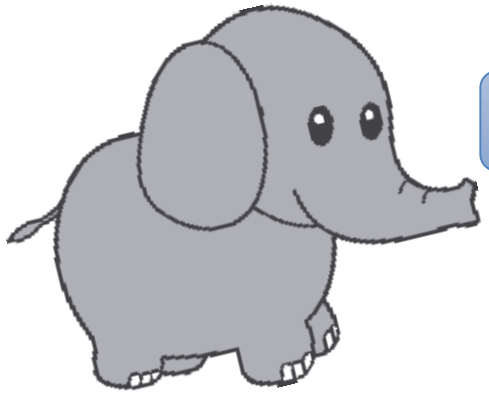
Rationale for Hawk (continued)



Source: Design Insights for MapReduce from Diverse Production Workloads, Chen et al 2012

Hawk: hybrid scheduling

Bulk of resources → good placement



Long job 1

⋮

centralized

Few jobs → reasonable scheduling latency

Short job 1 → distributed 1

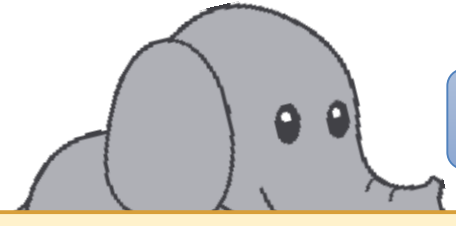
Latency sensitive → Fast scheduling

⋮ ⋮

Short job N → distributed N

Few resources → can trade not-so-good placement

Hawk: hybrid scheduling



Long job 1

centralized

Few jobs →

Bulk of

→

pla

Lat

ser

→

scheduling

es

e

d

t

Short job N

distributed N

BEST OF BOTH WORLDS

Good: scheduling latency for short jobs

Good: placement for long jobs

Hawk: distributed scheduling

- Sparrow

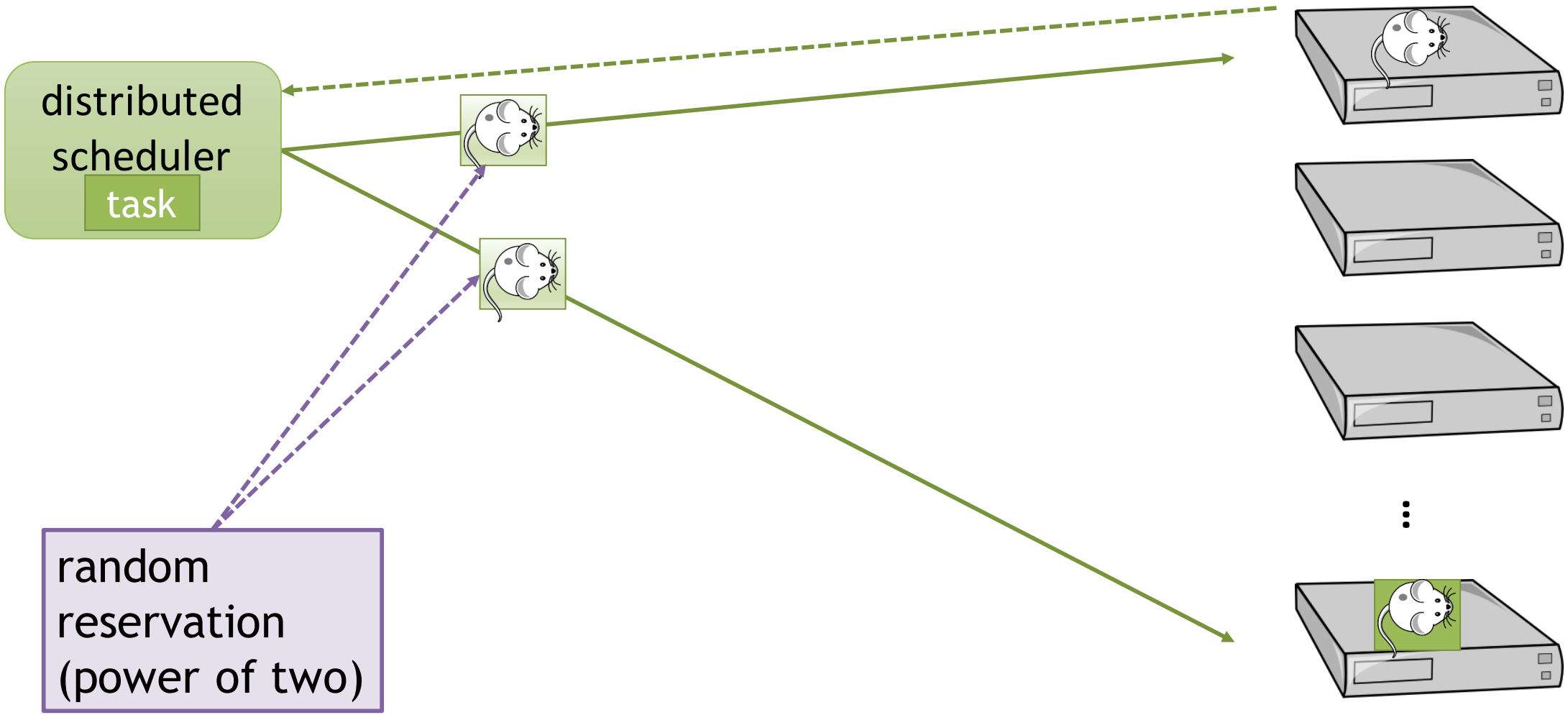
- Work-stealing

Hawk: distributed scheduling

- Sparrow

- Work-stealing

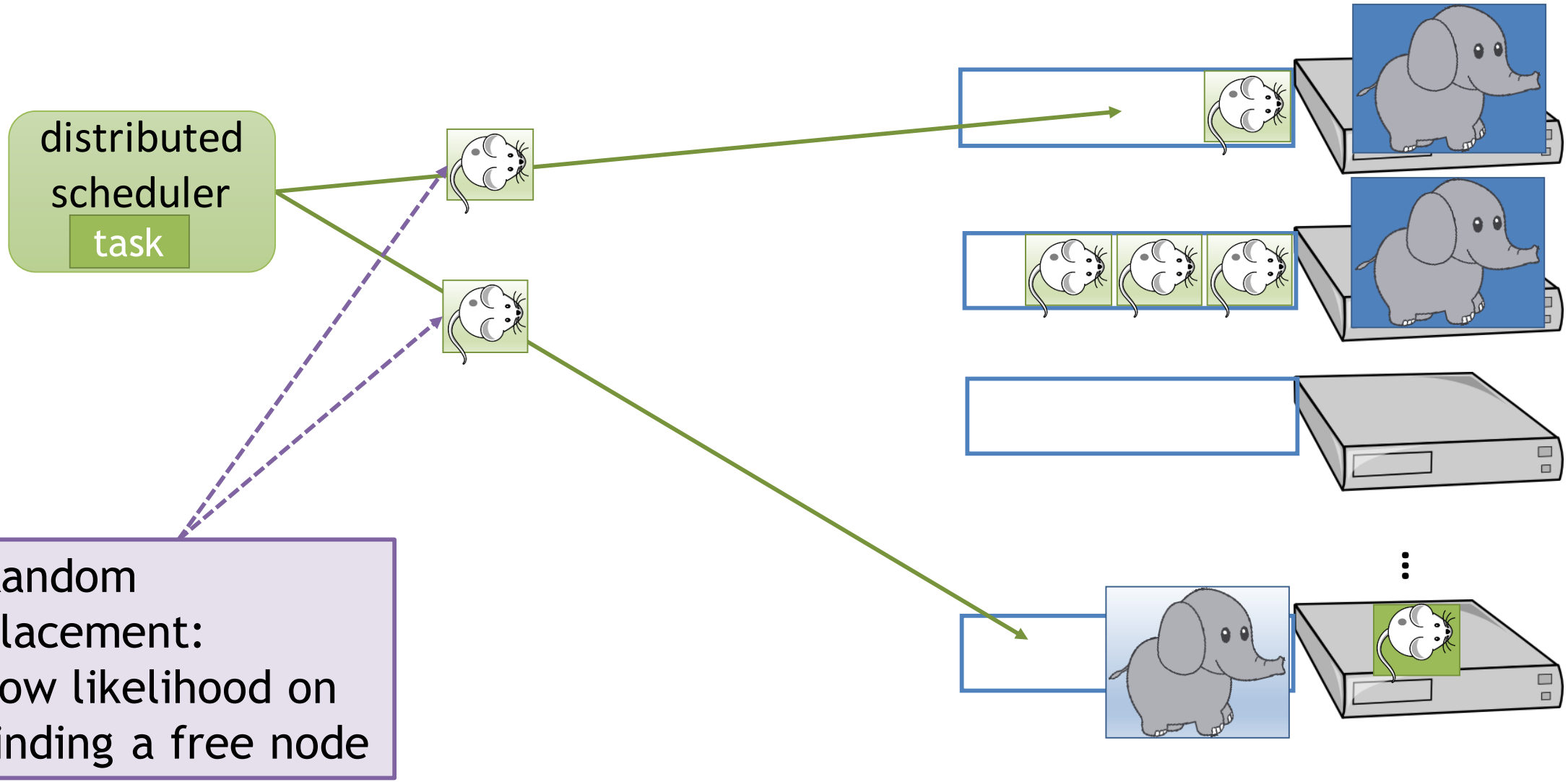
Sparrow


distributed scheduler
task
random reservation (power of two)

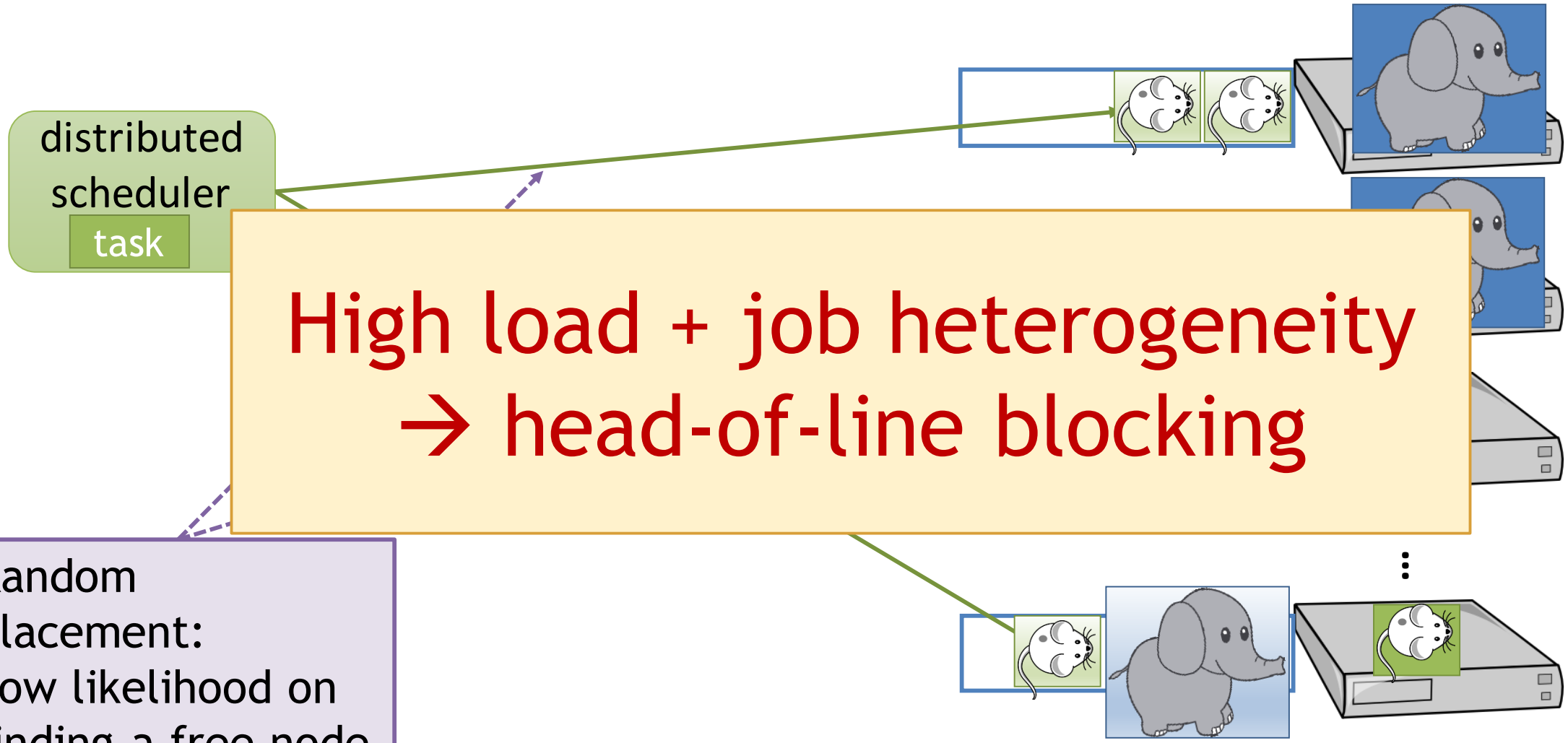
Hawk: distributed scheduling

- Sparrow
- Work-stealing

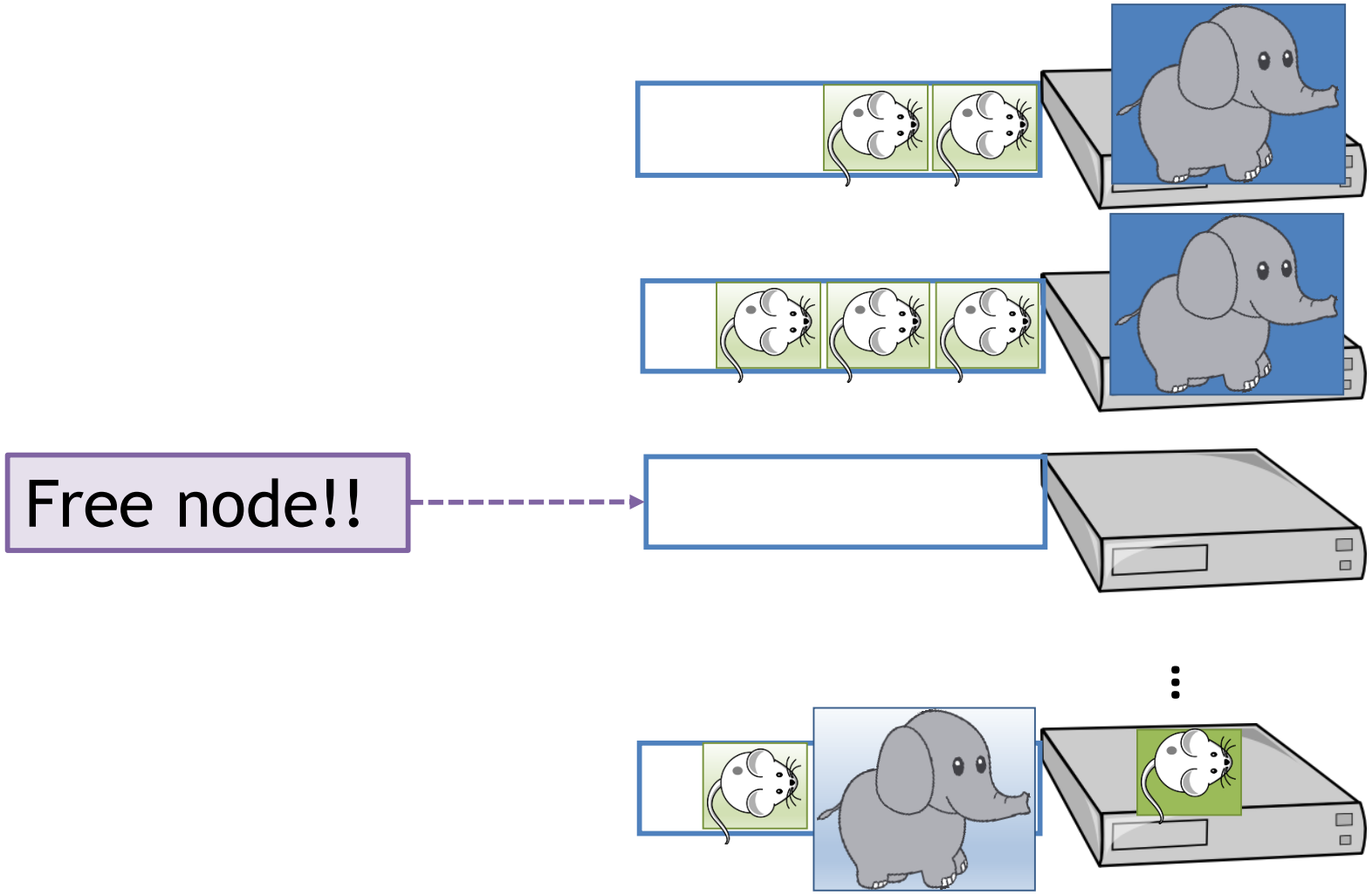
Sparrow and high load



Sparrow and high load



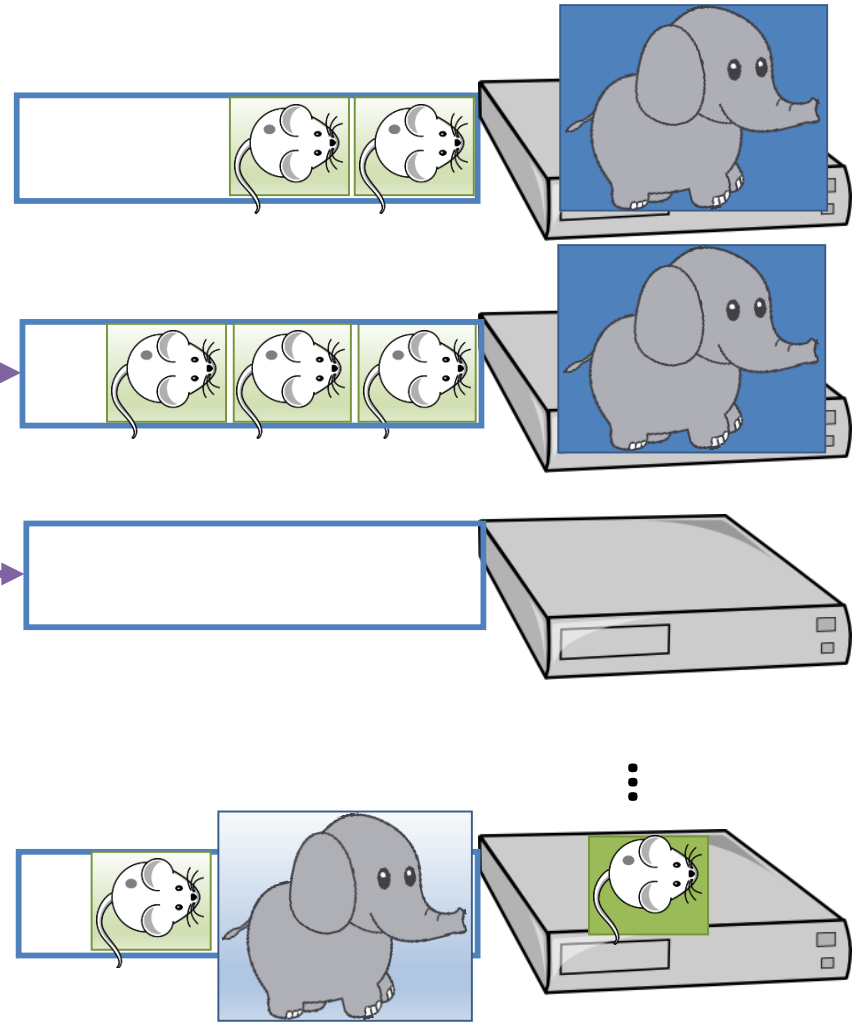
Hawk work-stealing



Hawk work-stealing

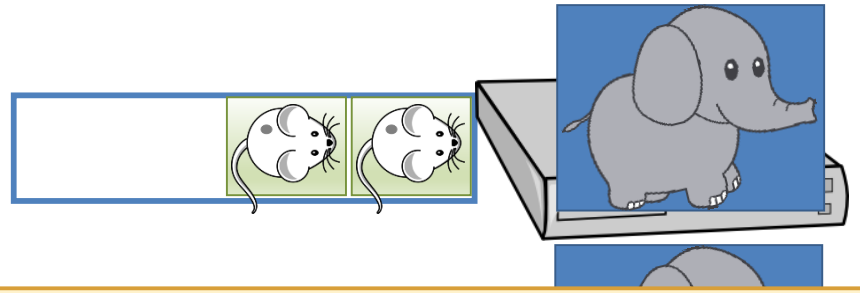
2. Random node: send **short tasks** reservation in queue

1. Free node: contact random node for probes!



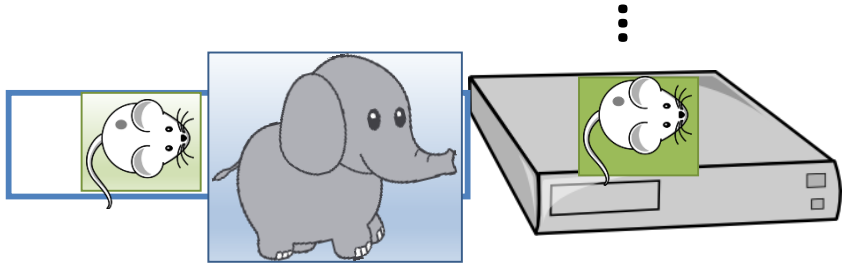
Hawk work-stealing

2. Random node: send **short tasks** reservation in queue

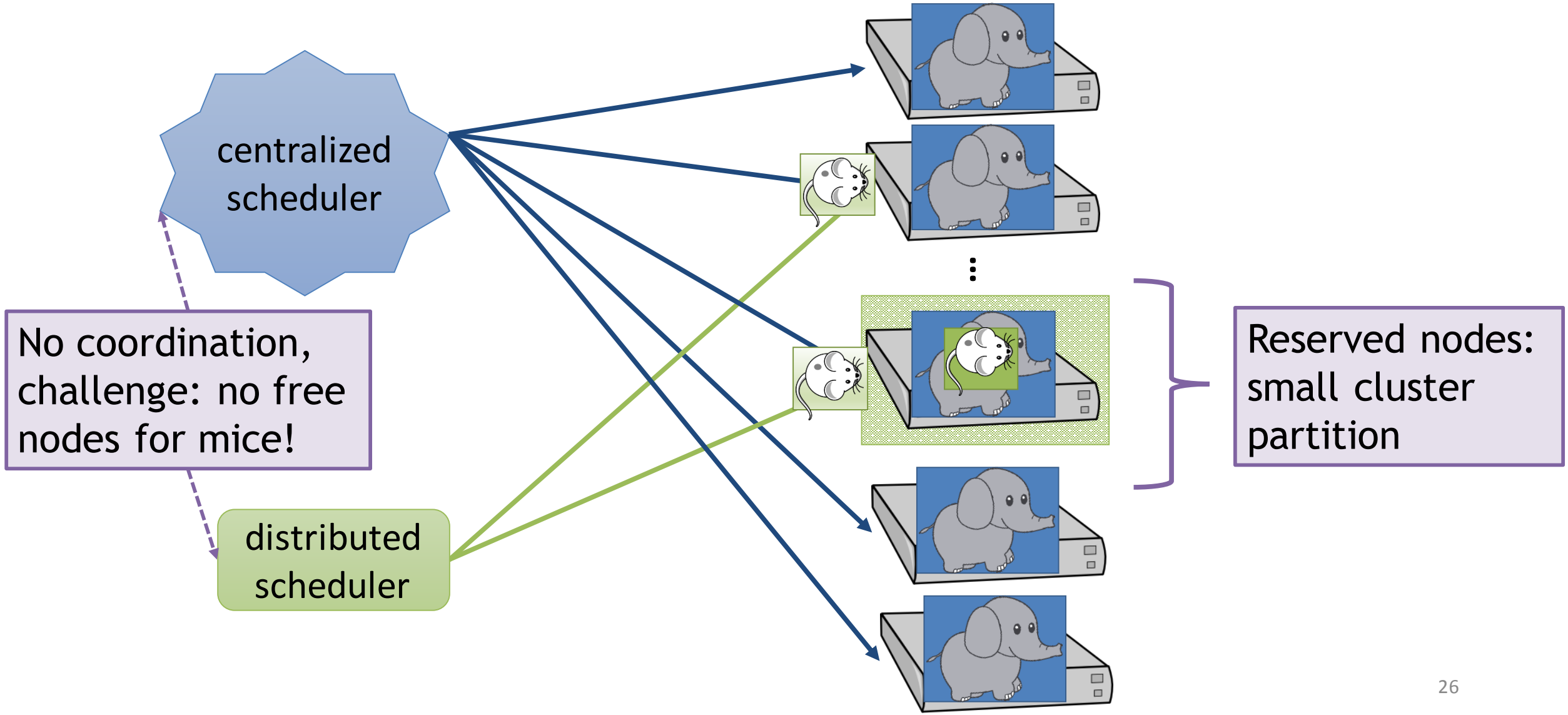


High load → high probability of contacting node with backlog

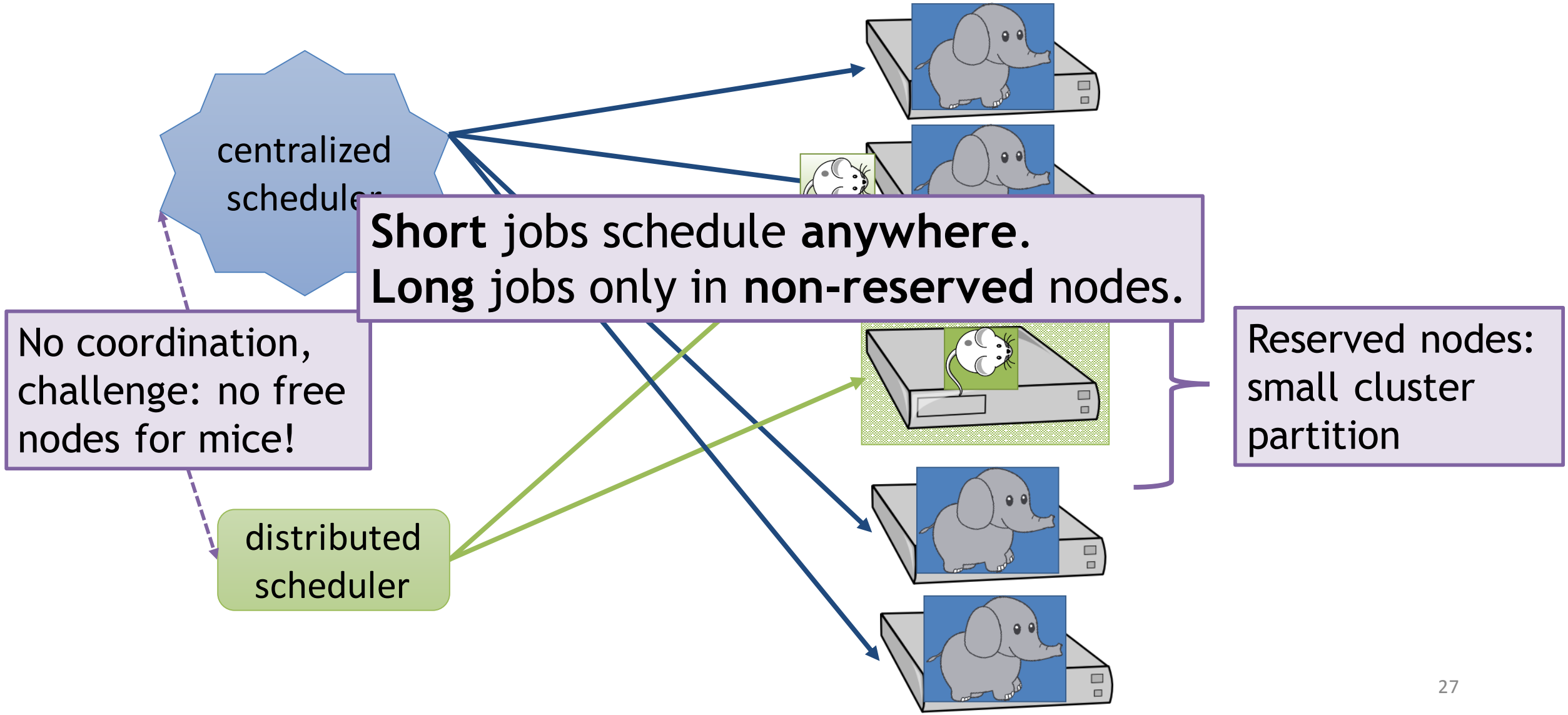
1. Free node: contact random node for probes!



Hawk cluster partitioning



Hawk cluster partitioning



Hawk design summary

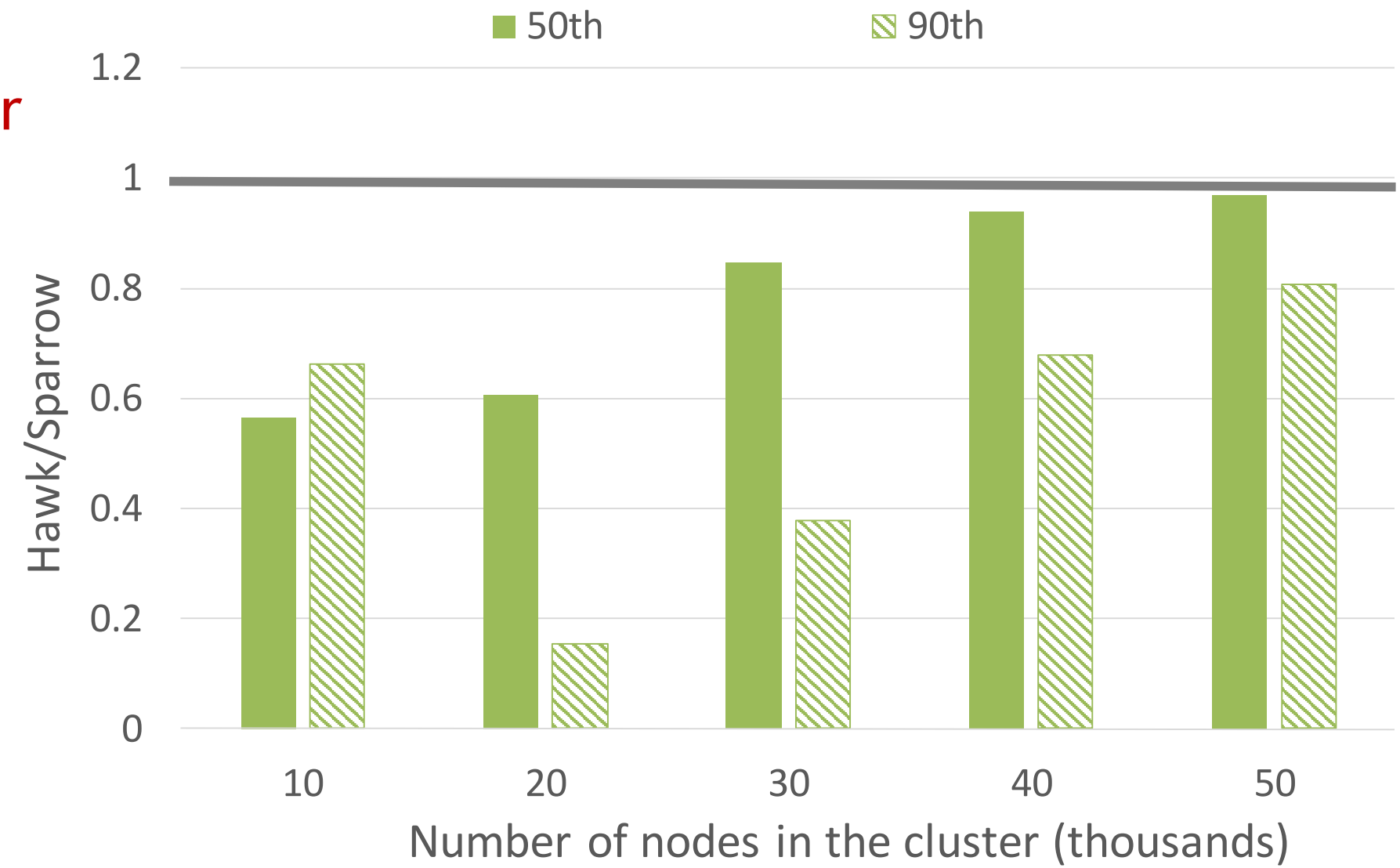
- ✓ Hybrid scheduler:
  long → centralized, short → distributed
- ✓ Work-stealing
- ✓ Cluster partitioning

Evaluation: 1. Simulation

- Sparrow simulator
- Google trace
- Vary number of nodes to vary cluster utilization
- Measure: Job running time
- Report 50^{th} and 90^{th} percentiles for short and long jobs
- Normalized to Sparrow

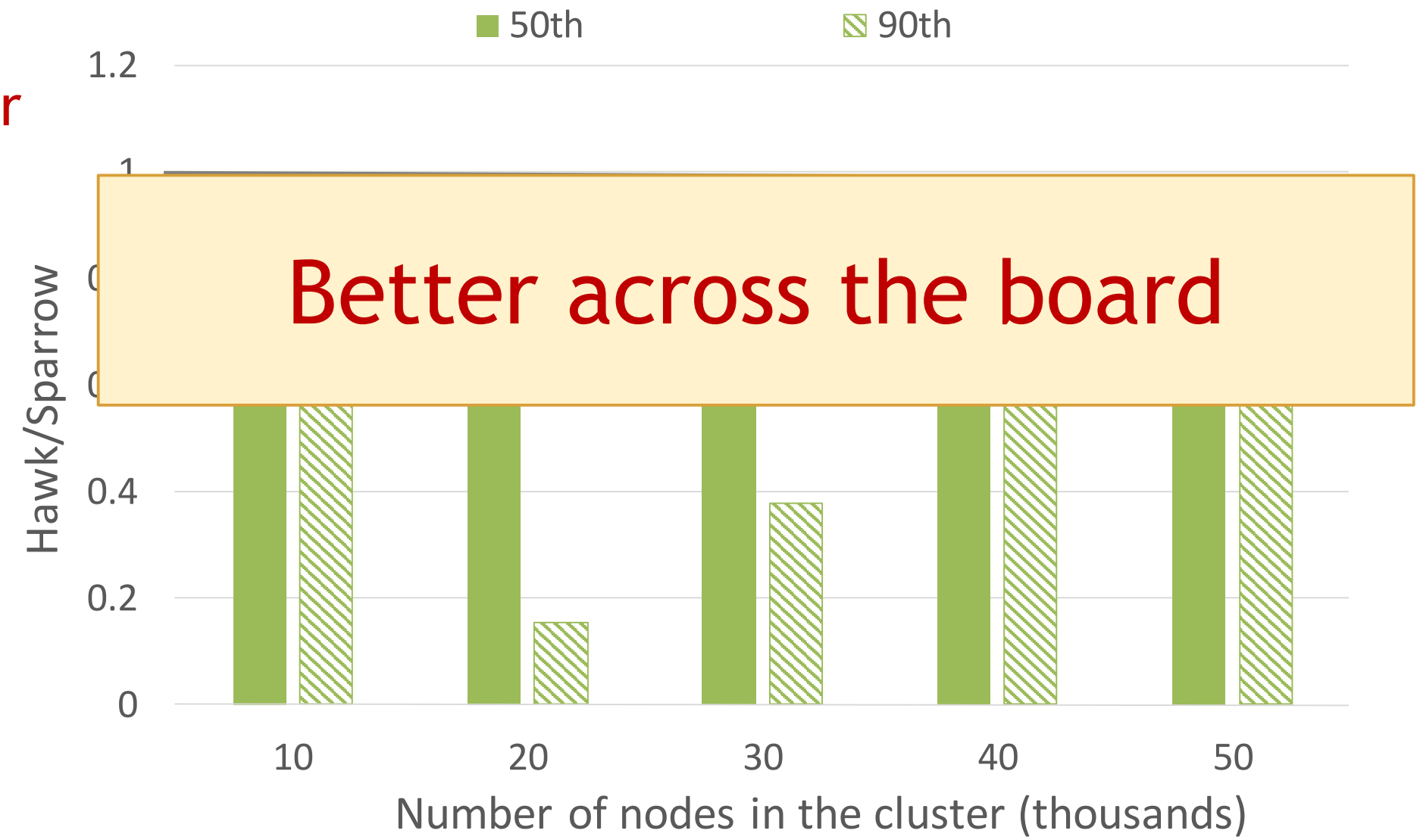
Simulated results: short jobs

lower better



Simulated results: short jobs

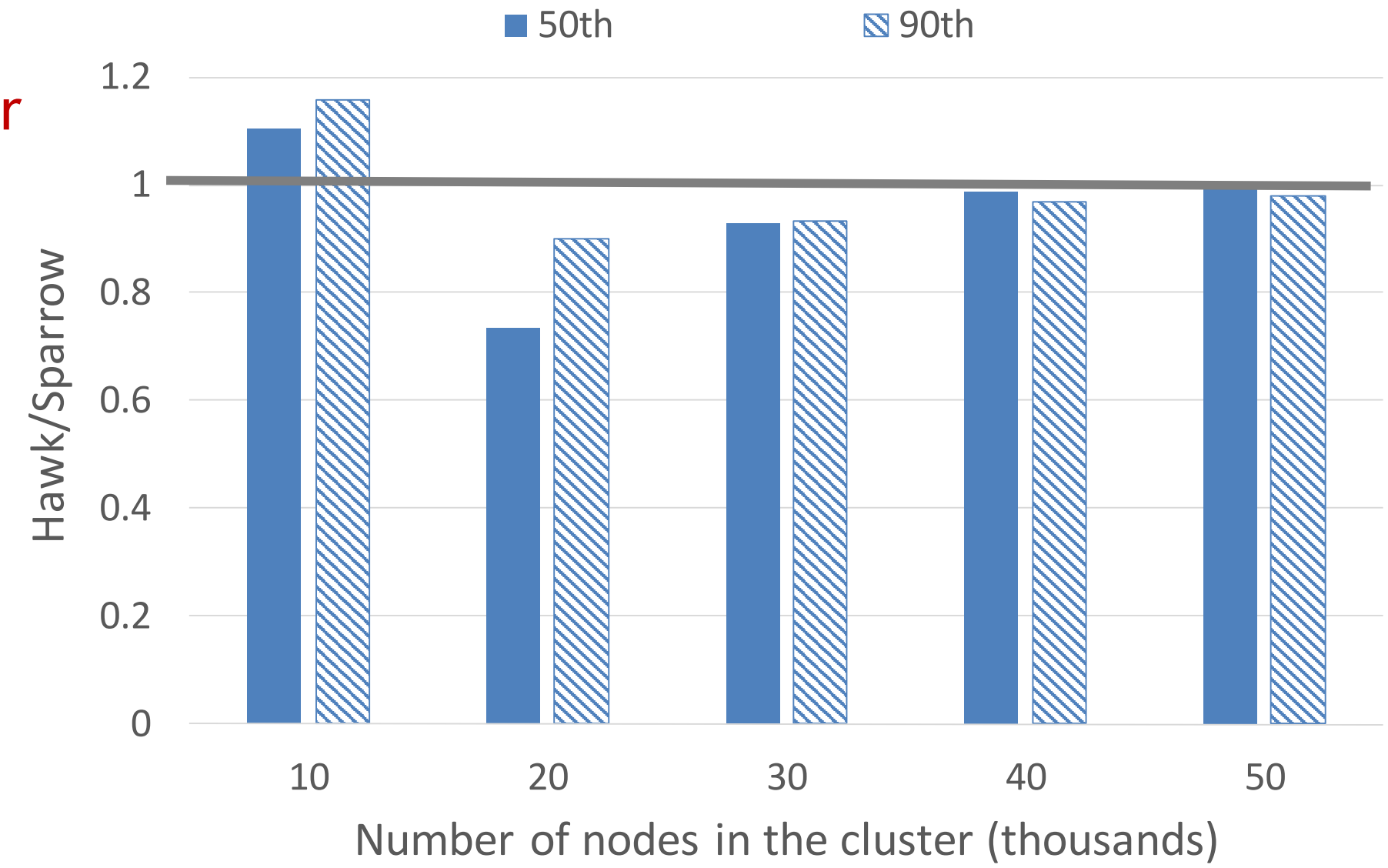
lower better



Better across the board

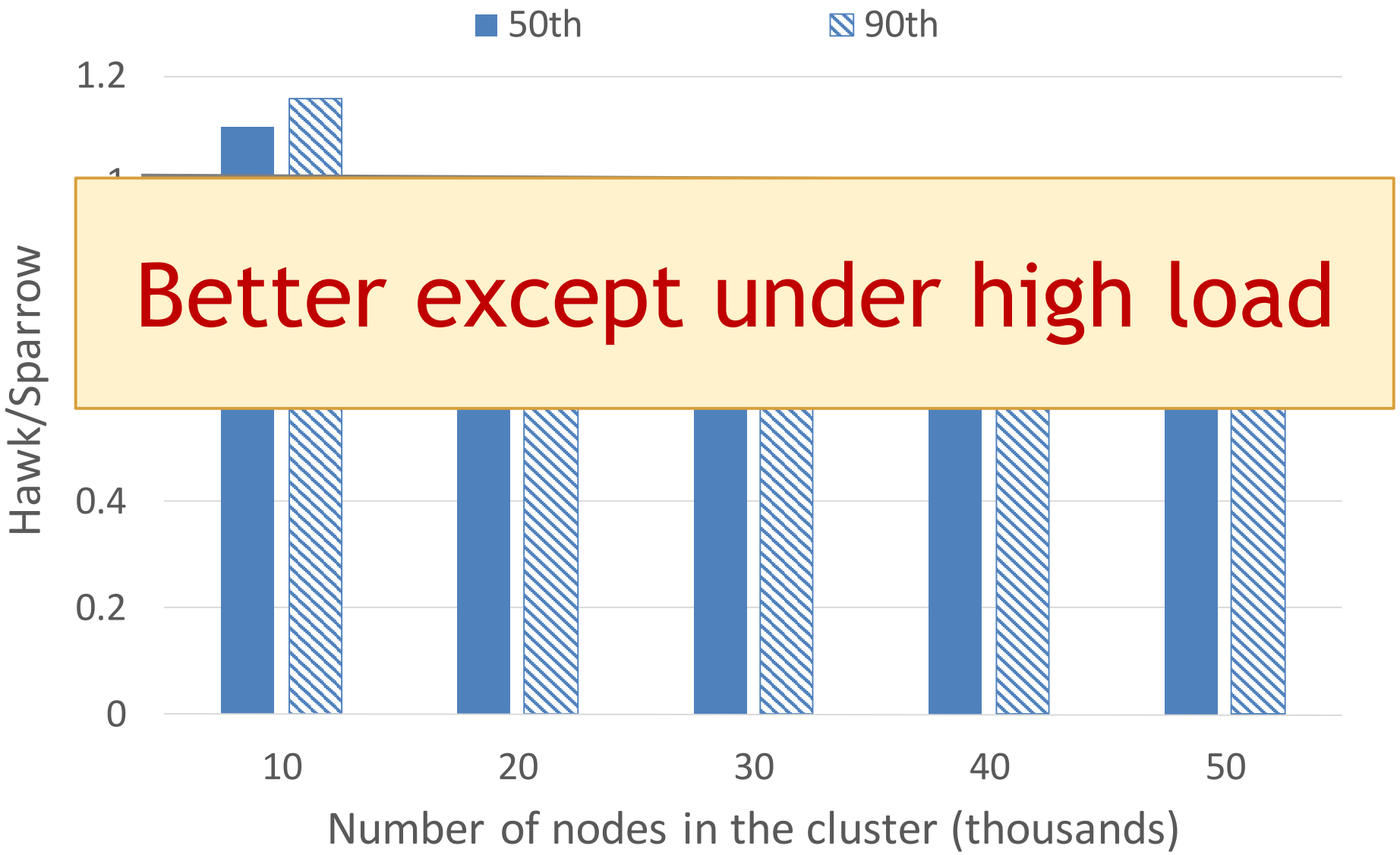
Simulated results: long jobs

lower better



Simulated results: long jobs

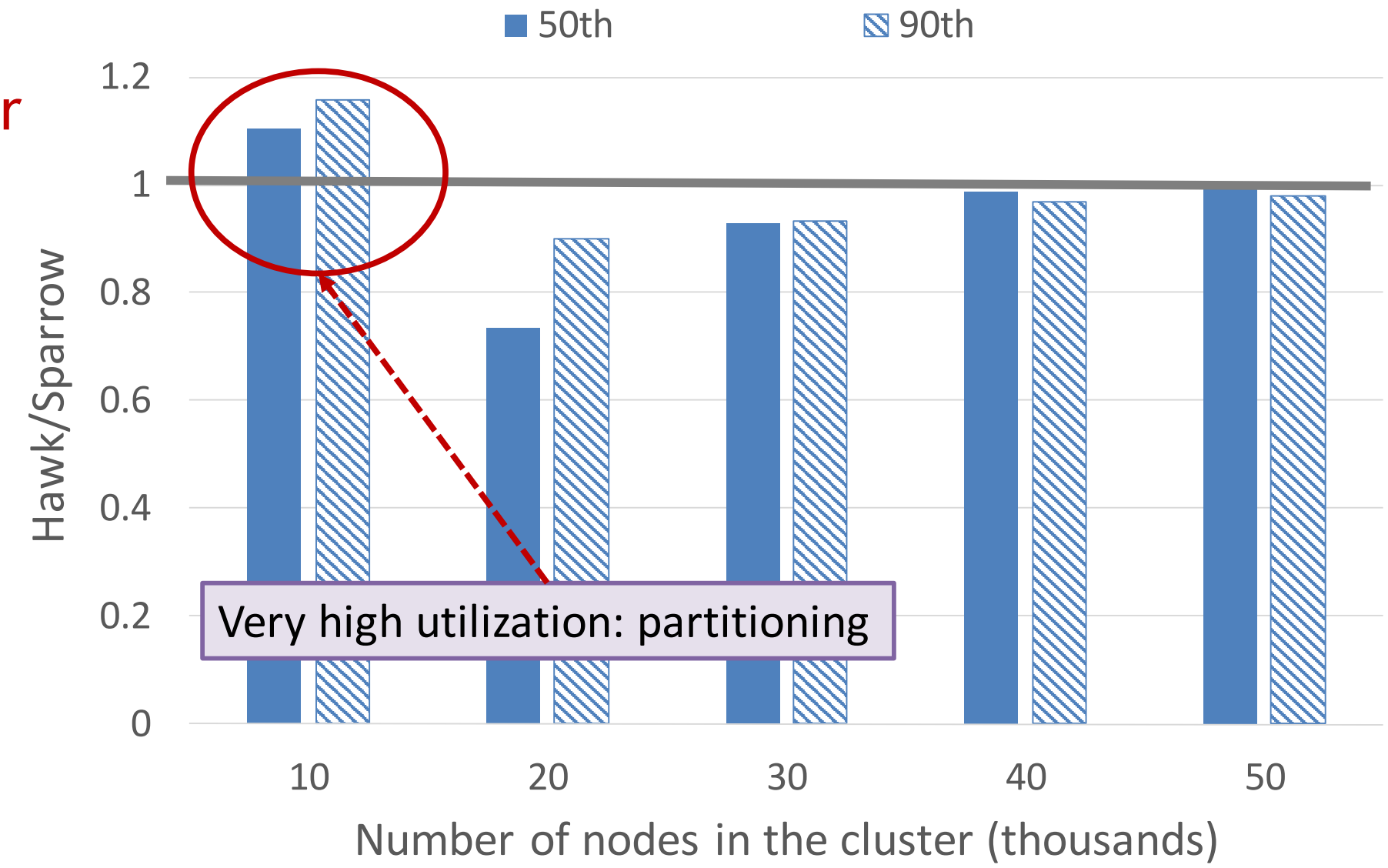
lower better



Better except under high load

Simulated results: long jobs

lower better



Decomposing Hawk

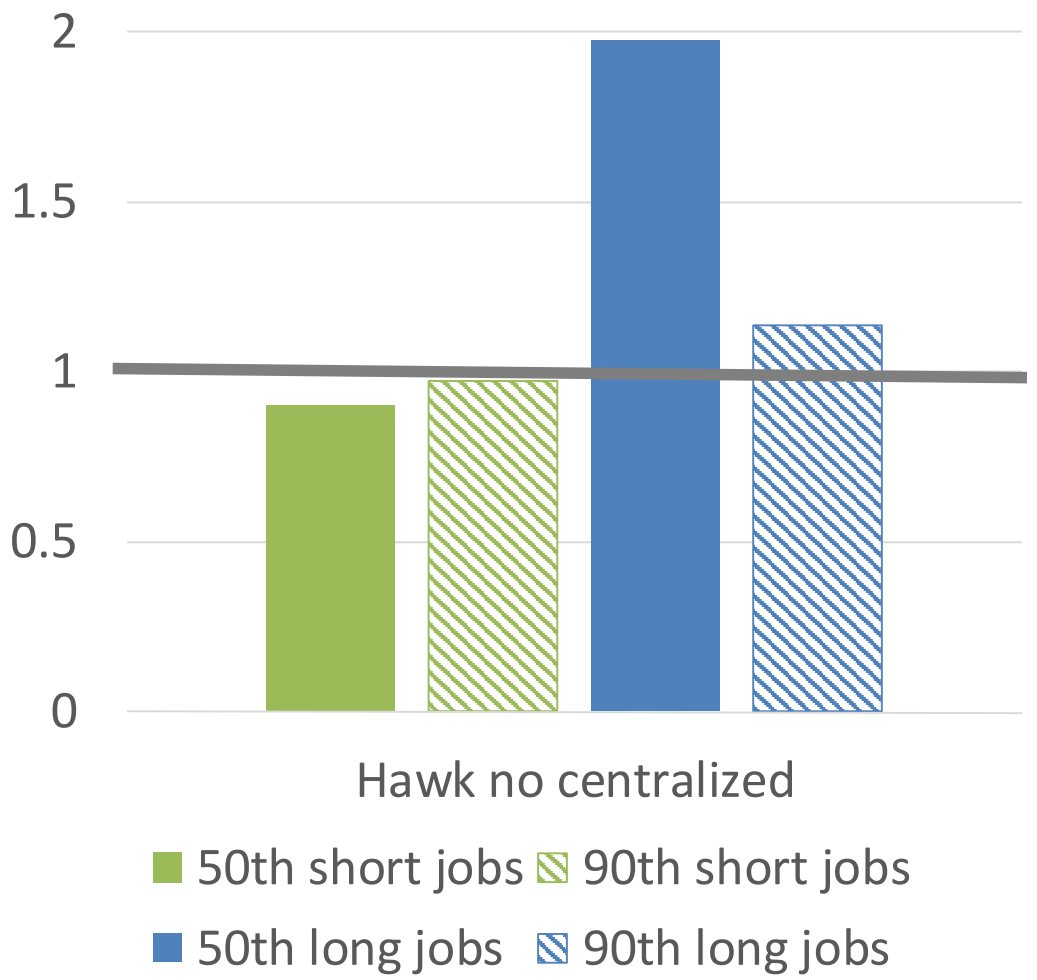
1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning

(normalized to Hawk)

Decomposing Hawk: no centralized

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning

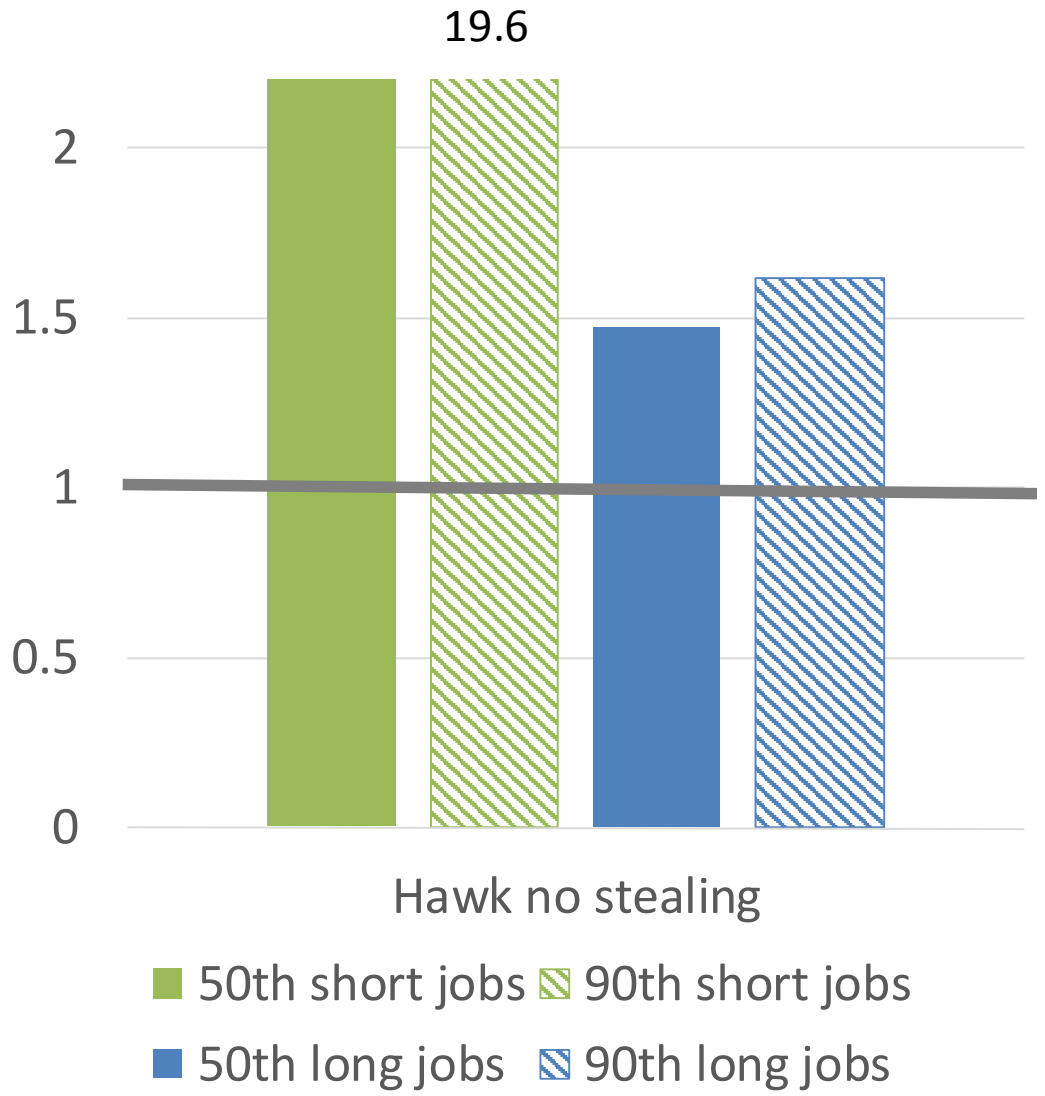
(normalized to Hawk)



Decomposing Hawk: no stealing

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning

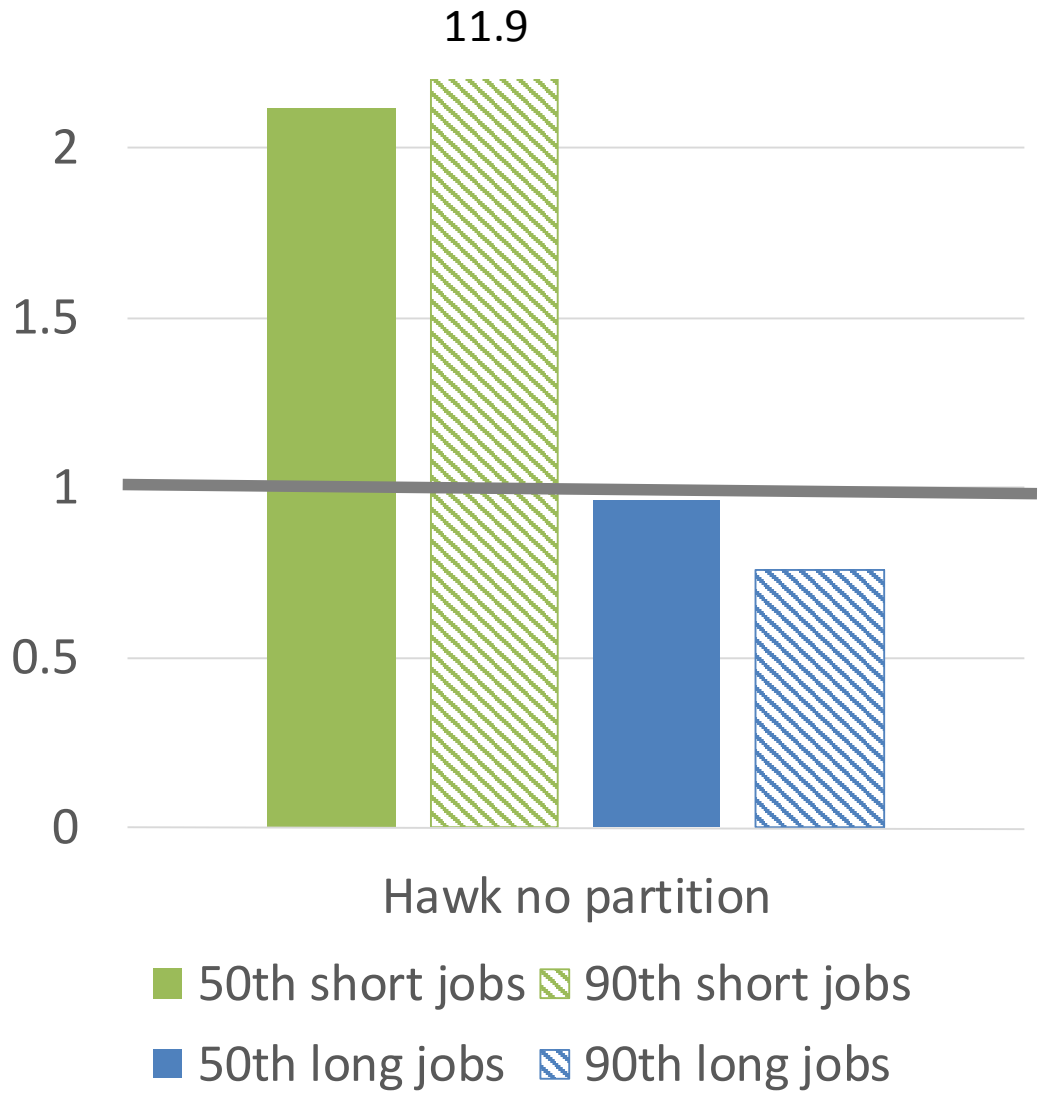
(normalized to Hawk)



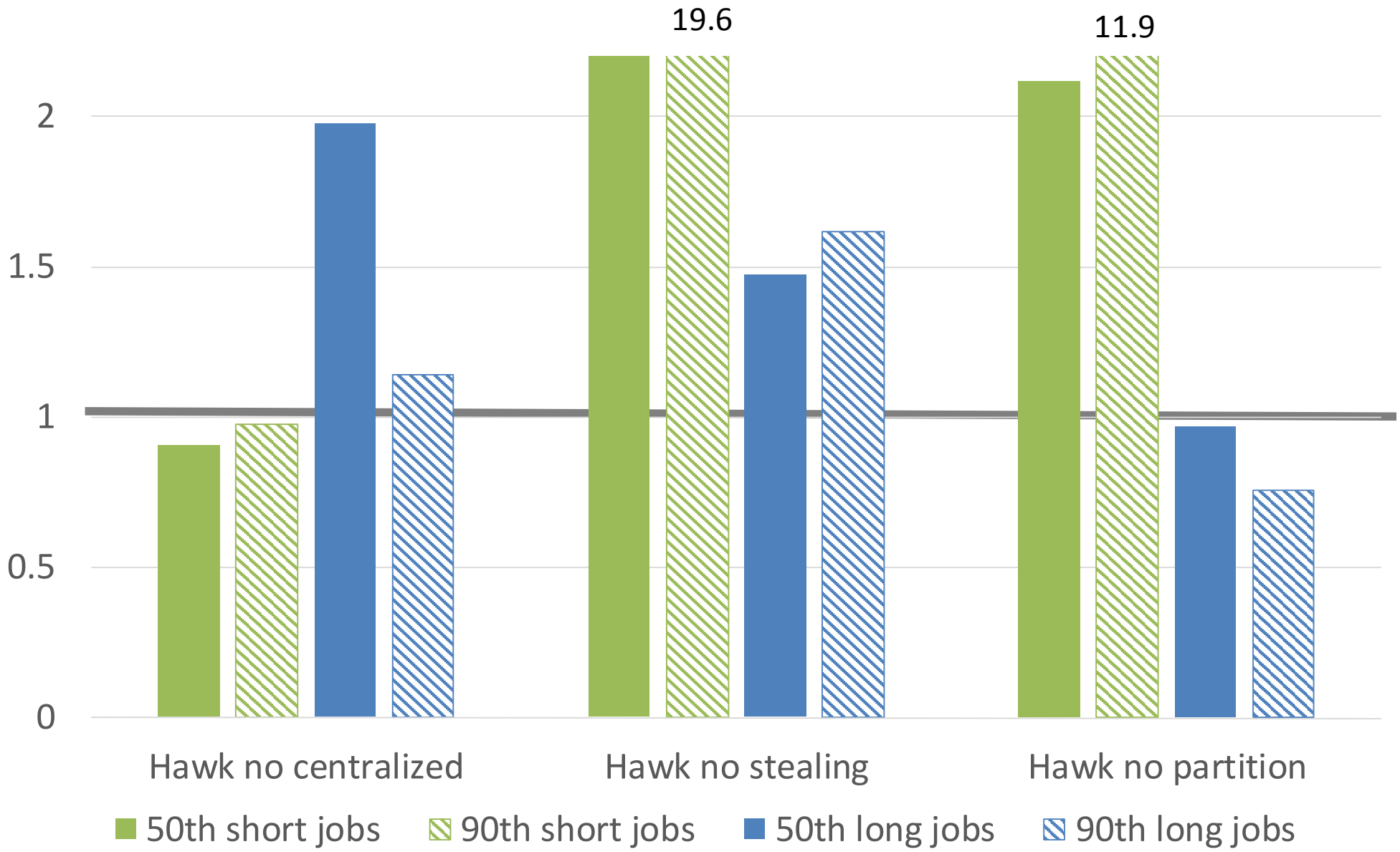
Decomposing Hawk: no partitioning

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning

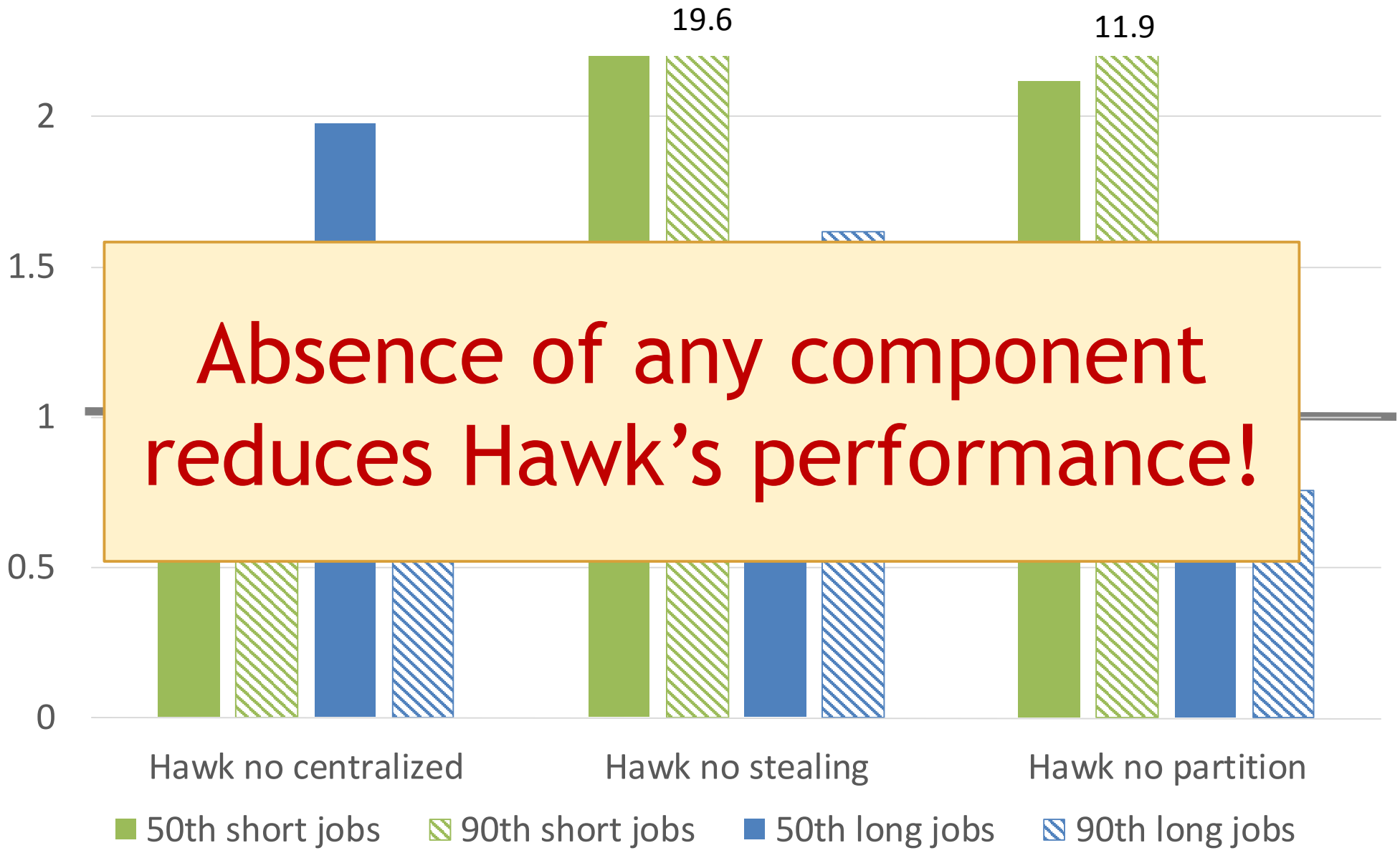
(normalized to Hawk)



Decomposing Hawk summary



Decomposing Hawk summary



Absence of any component reduces Hawk's performance!

Sensitivity analysis

1. Incorrect estimates of runtime
2. Cut off long/short
3. Details of stealing

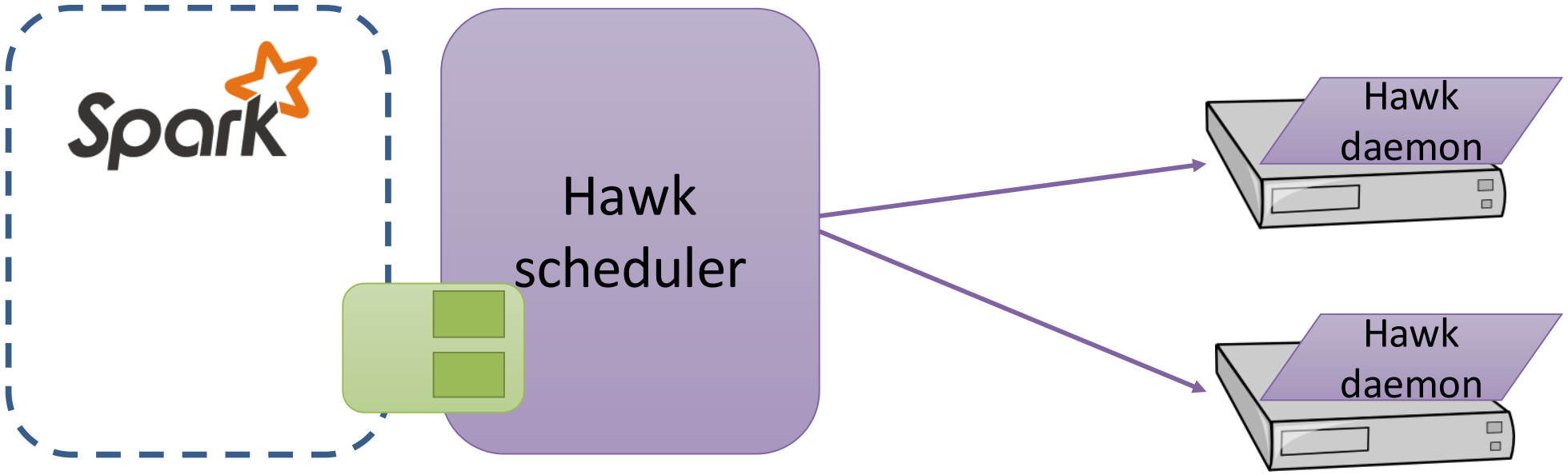
Sensitivity analysis

1. Incorrect estimates of runtime
2. Cut off long/short
3. Details of stealing

Bottom line: benefits of Hawk remain despite variation

See paper for details

Evaluation: 2. Implementation

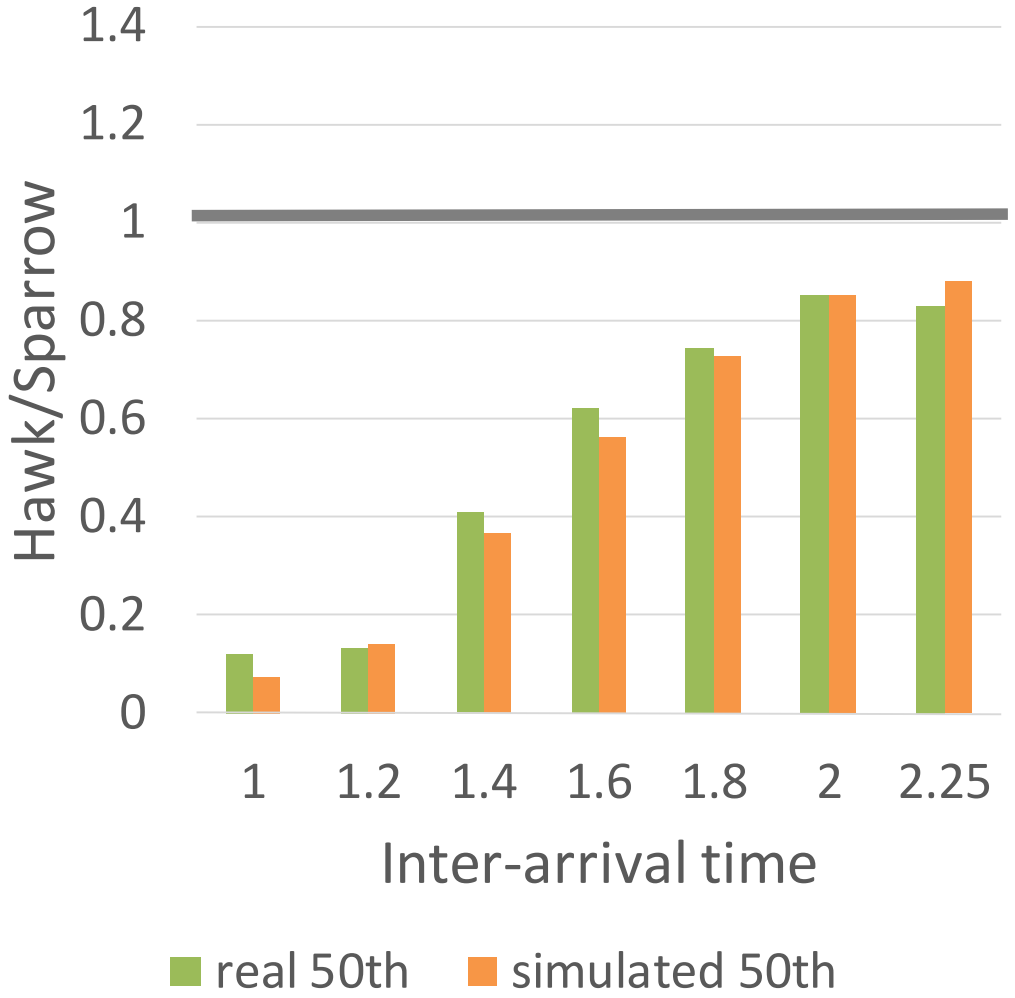


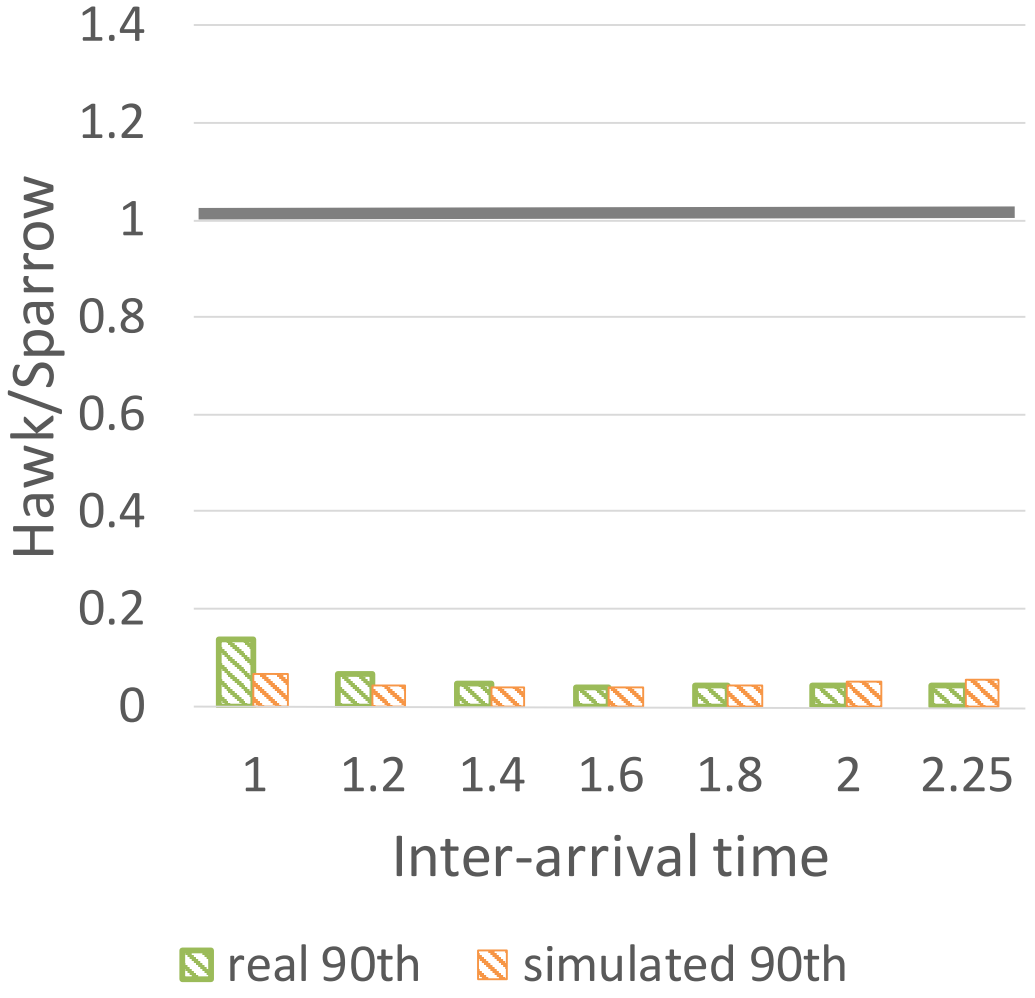
Experiment

- 100-node cluster
- Subset of Google trace
- Vary inter-arrival time to vary cluster utilization
- Measure: Job running time
- Report 50^{th} and 90^{th} percentile for short and long jobs
- Normalized to Sparrow

Short jobs

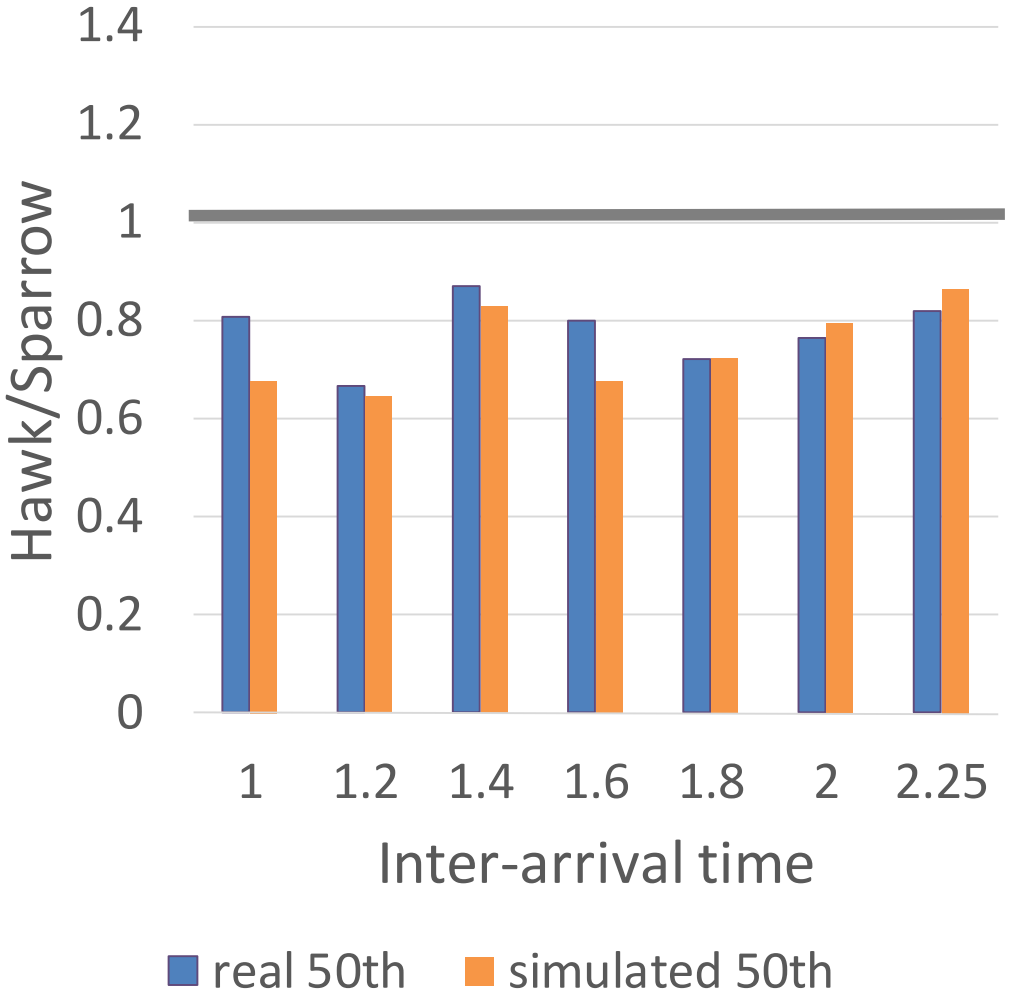
lower better

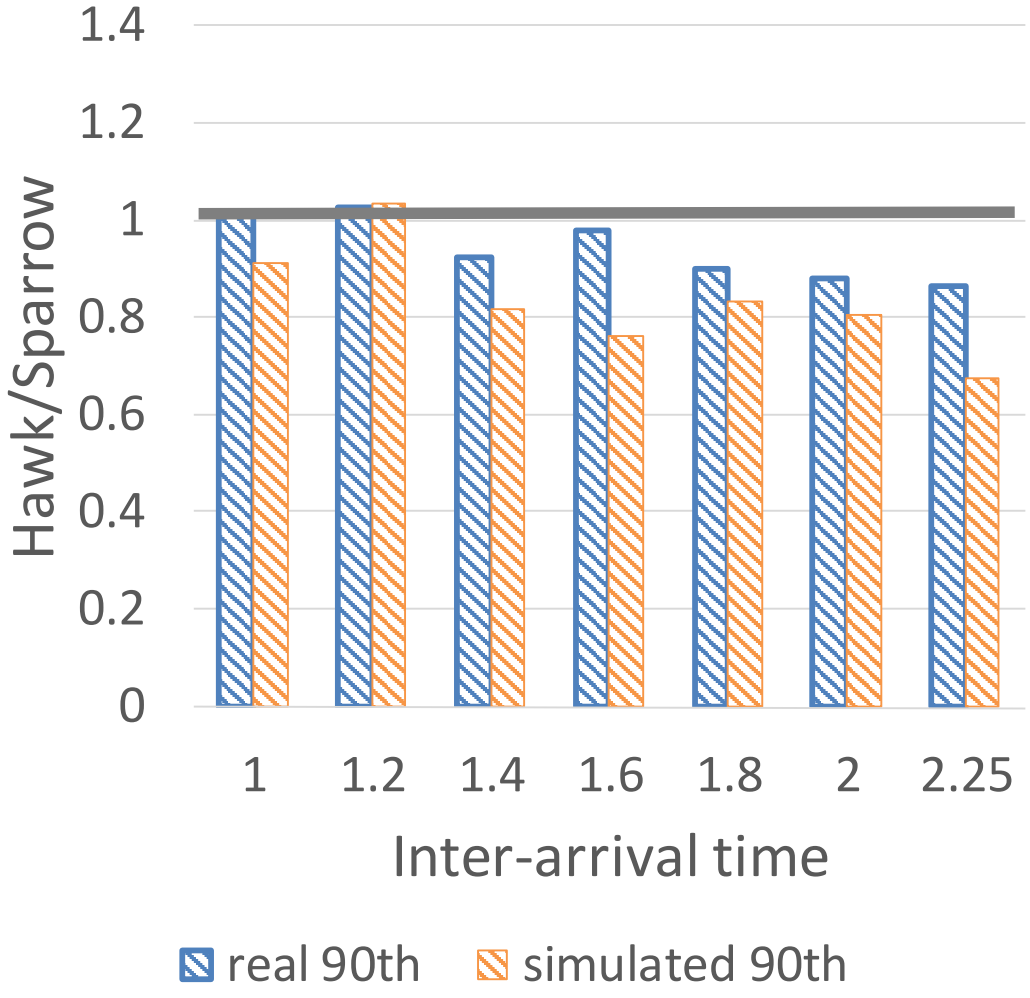




Long jobs

lower better





Implementation

1. Hawk works well in real cluster
2. Good correspondence implementation/simulation

Related work

Centralized: Hadoop Fair Scheduler, Quincy
Eurosys'10, SOSP'09

Two level: Yarn, Mesos
SoCC'12, NSDI'11

Distributed schedulers: Omega, Sparrow
Eurosys'12,SOSP'13

Hybrid schedulers: Mercury

Conclusion

- Hawk: hybrid scheduler

  ✓ long: centralized, short: distributed
  ✓ work-stealing
  ✓ cluster partitioning

- Hawk provides good results for short and long jobs

- Even under high cluster utilization