

Hawk: Hybrid Datacenter Scheduling

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

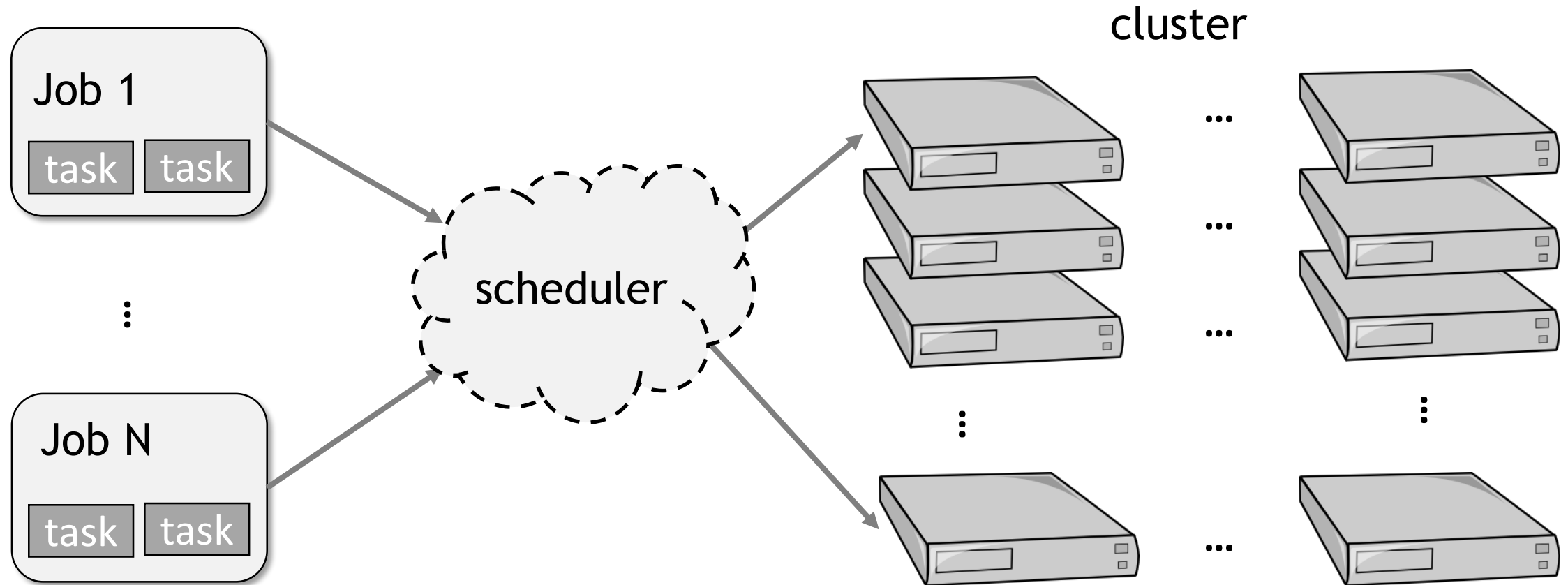
July 10th, 2015

USENIX ATC 2015

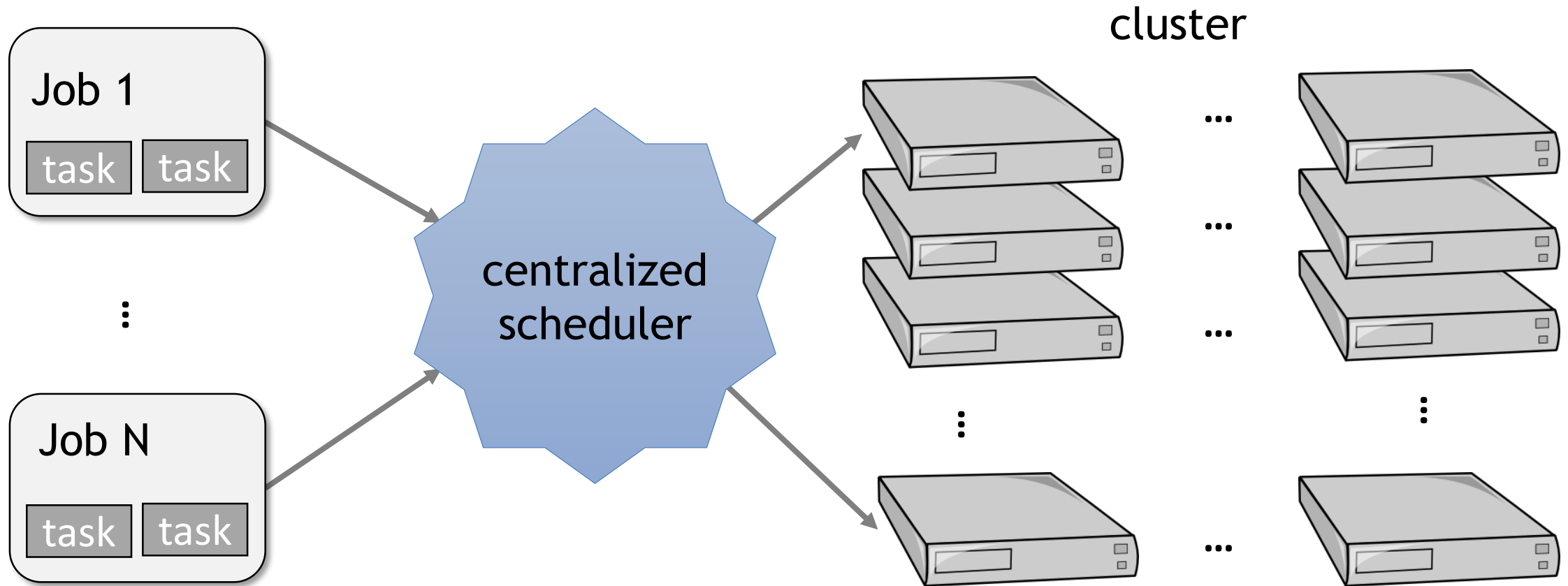


ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

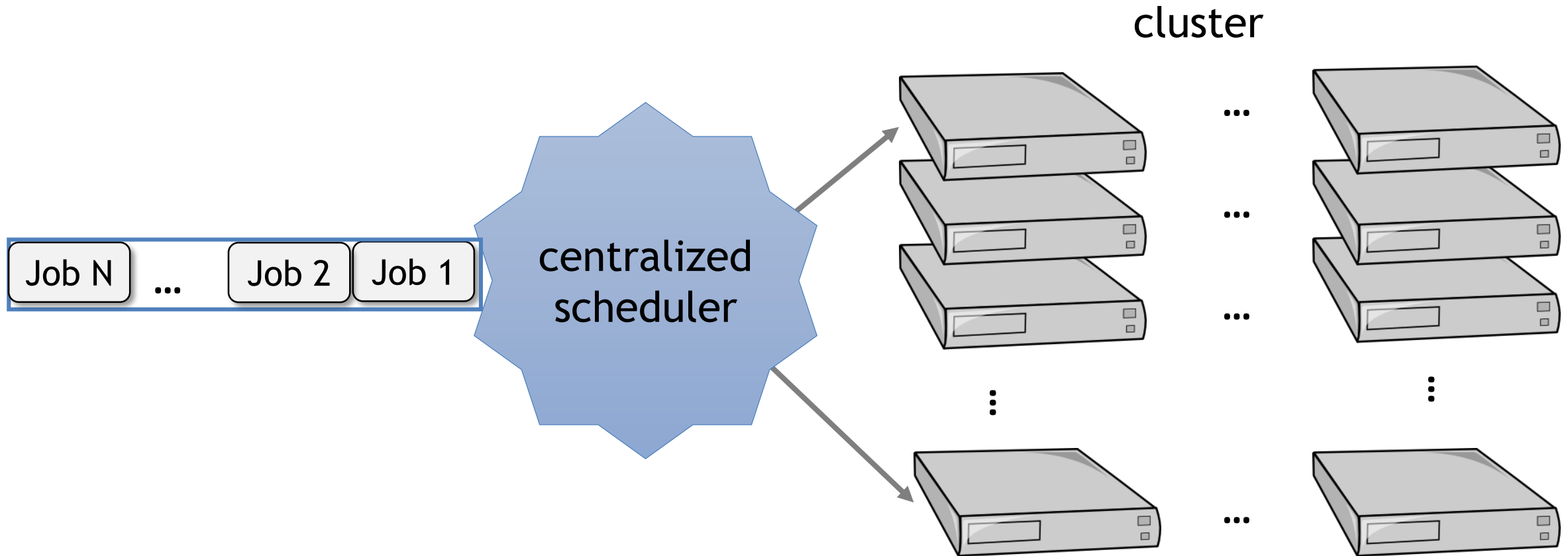
Introduction: datacenter scheduling



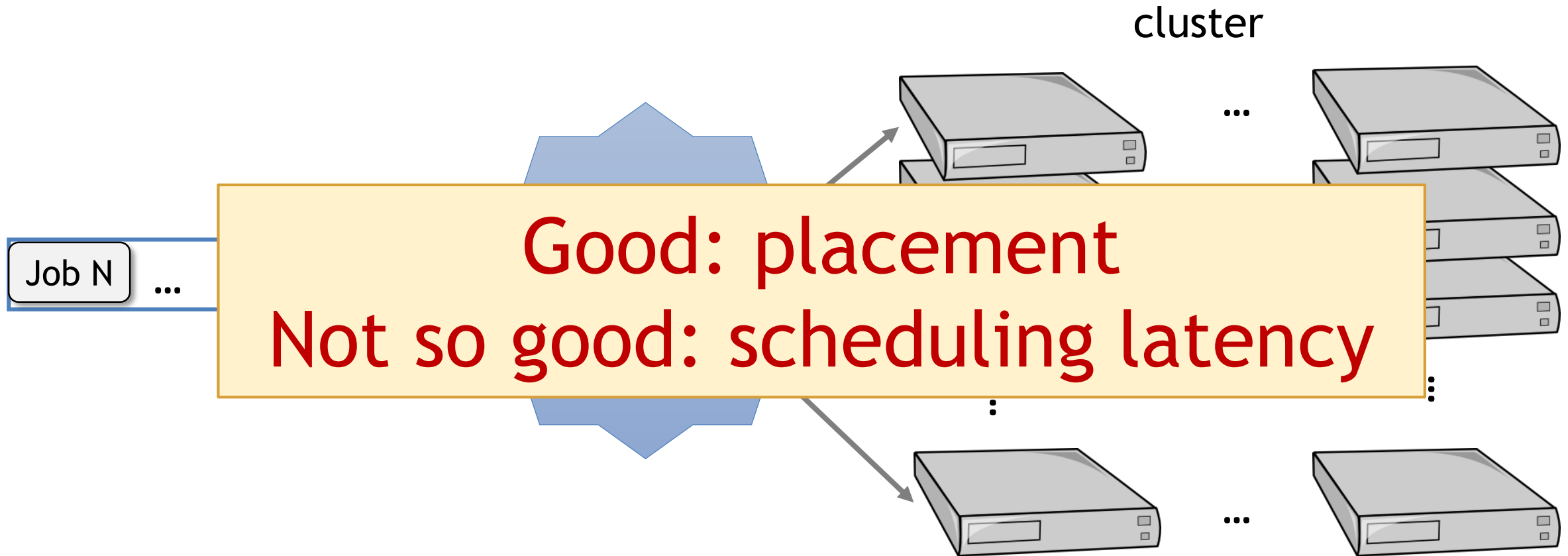
Introduction: centralized scheduling



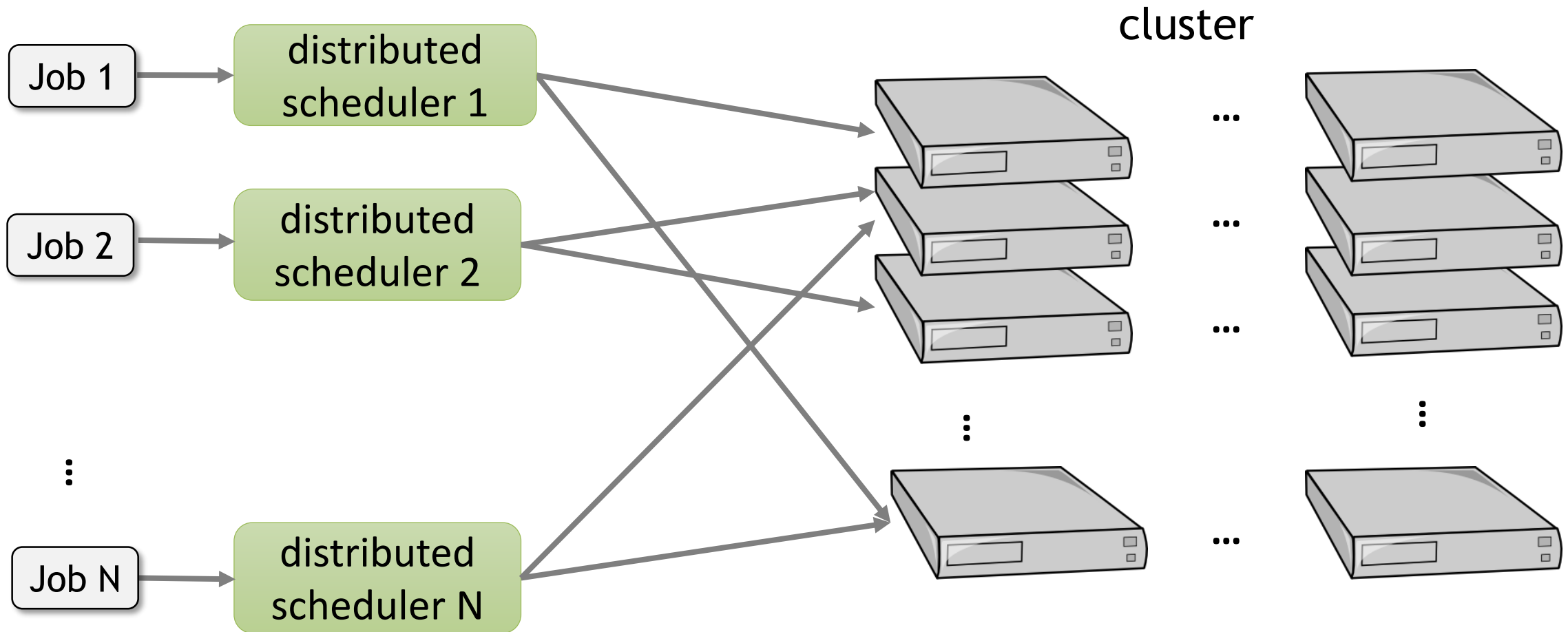
Introduction: centralized scheduling



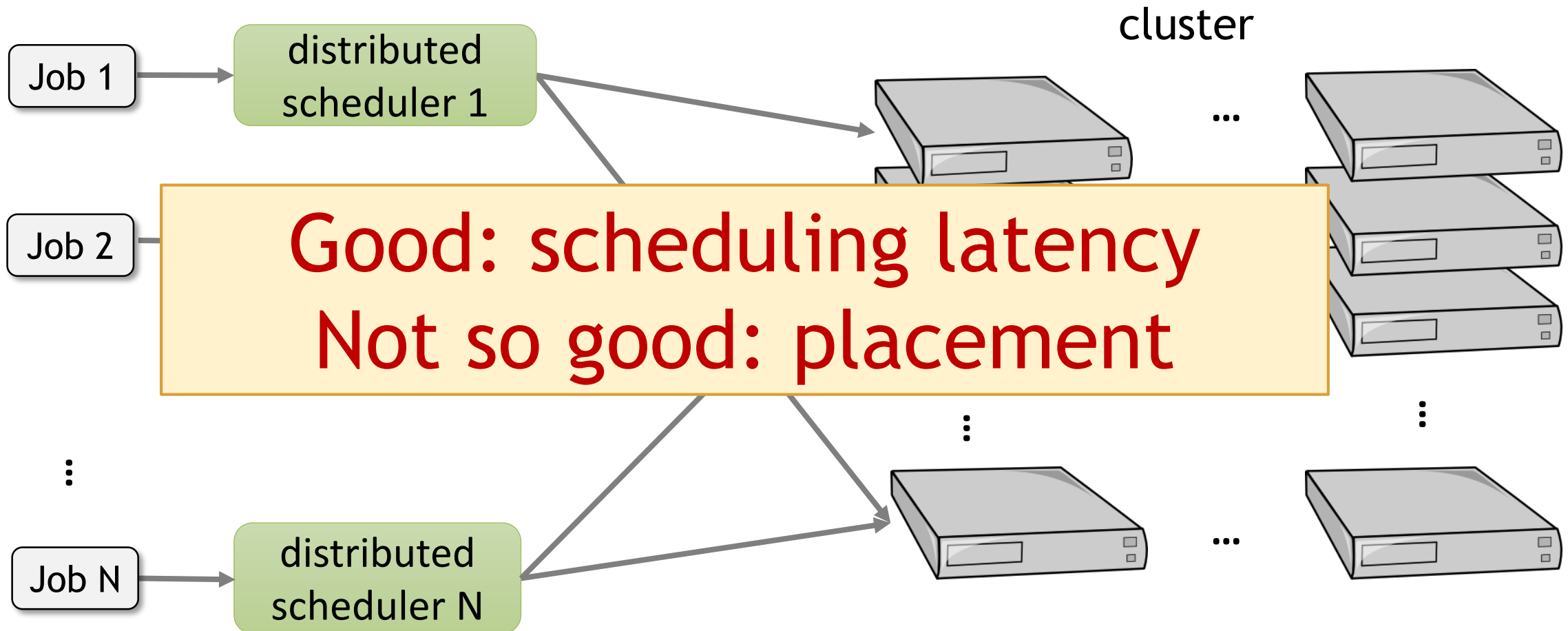
Introduction: centralized scheduling



Introduction: distributed scheduling



Introduction: distributed scheduling



Outline

1) Introduction

2) HAWK hybrid scheduling

- Rationale
- Design

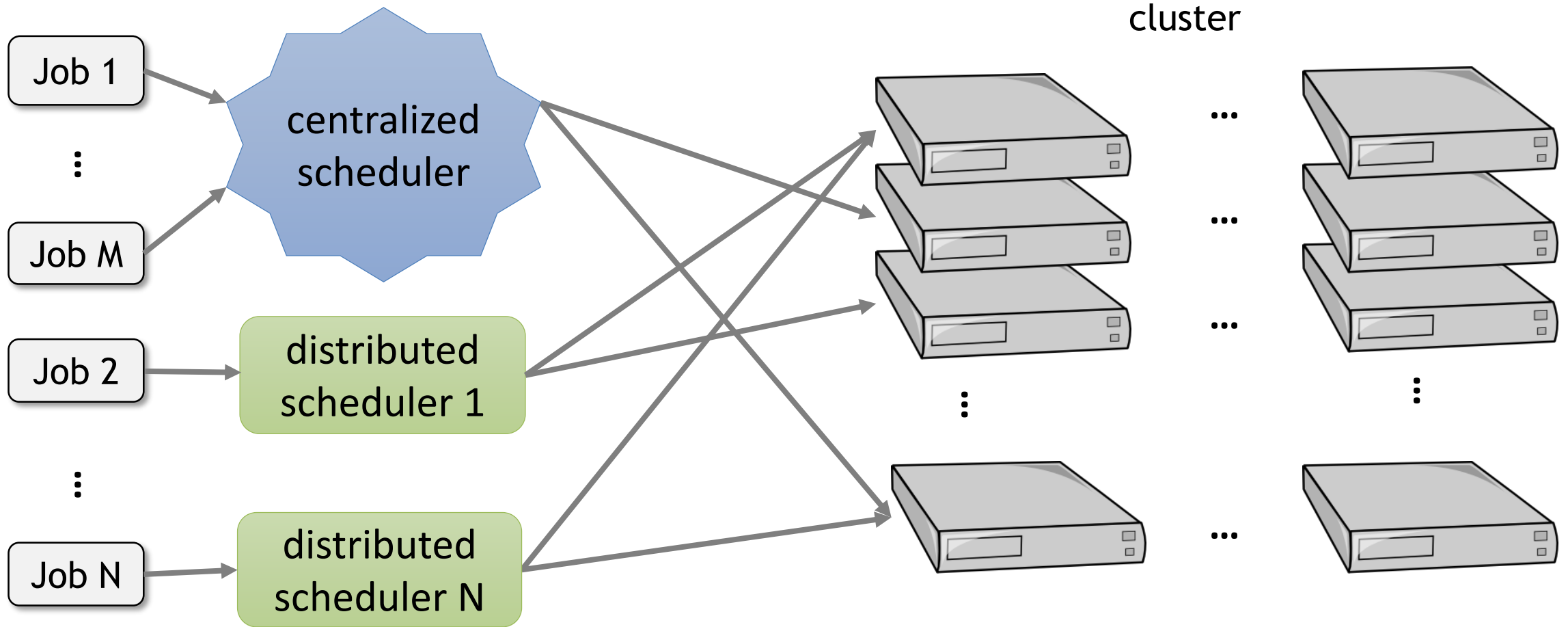


3) Evaluation

- Simulation
- Real cluster implementation

4) Conclusion

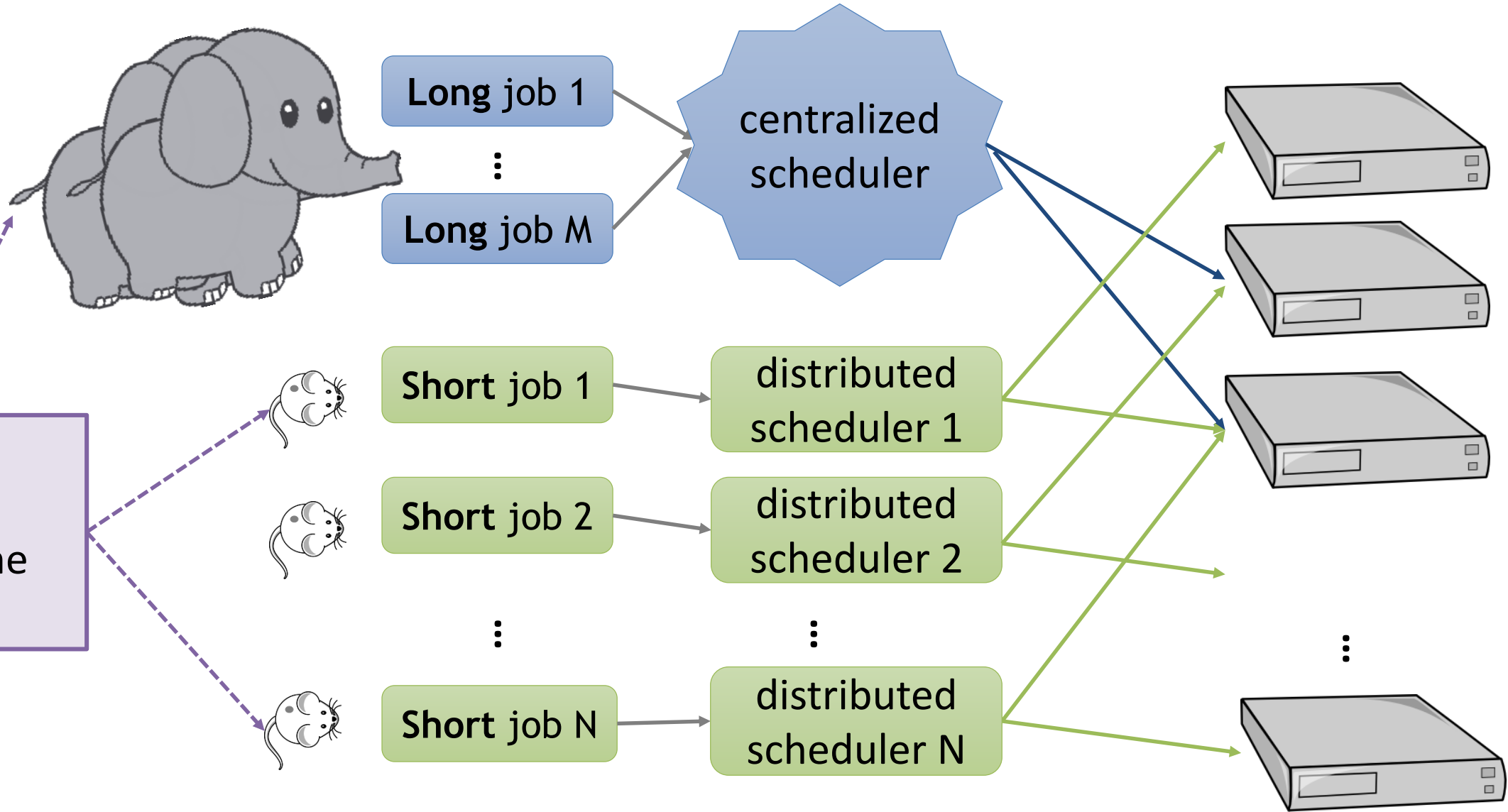
Hybrid scheduling



Hawk: hybrid scheduling

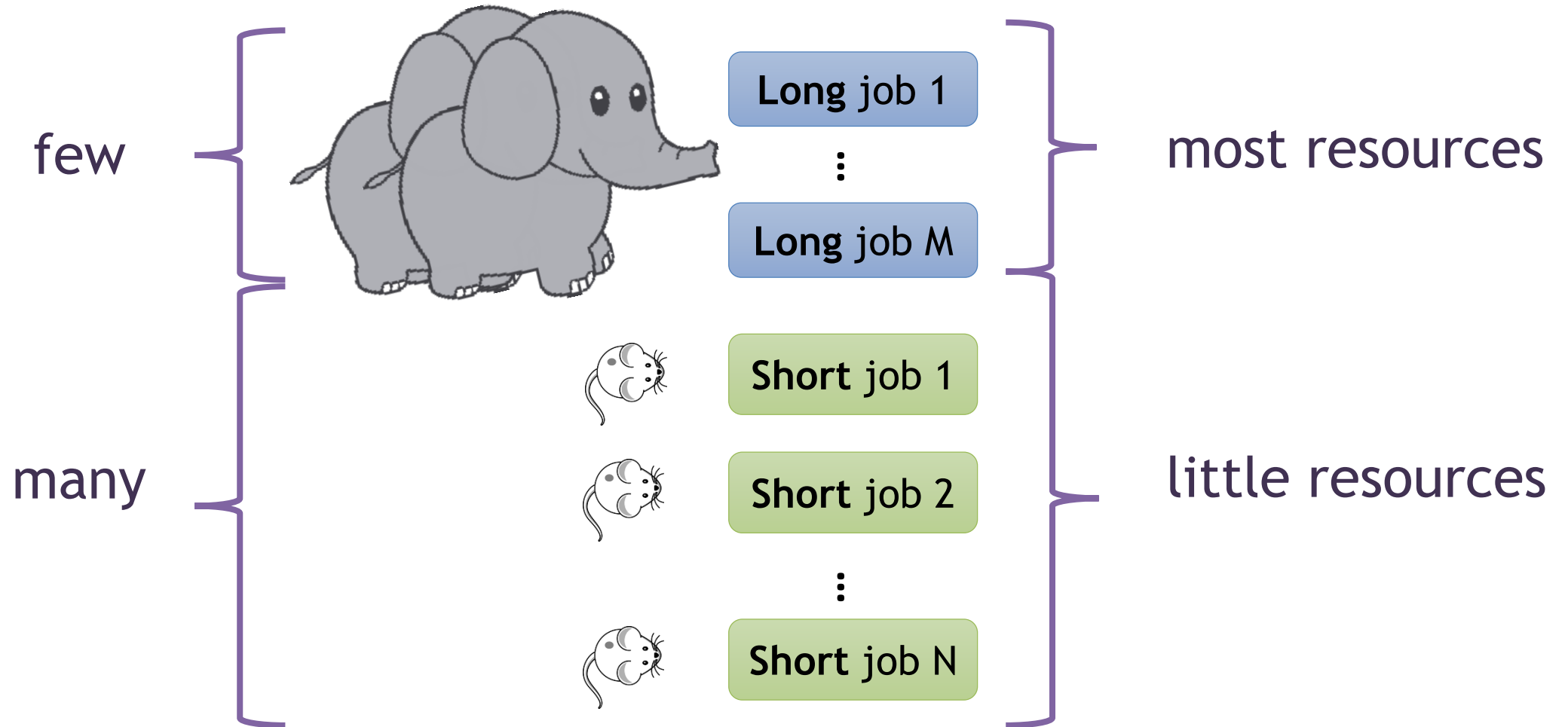
- ✓ Long jobs → centralized
- ✓ Short jobs → distributed

Hawk: hybrid scheduling



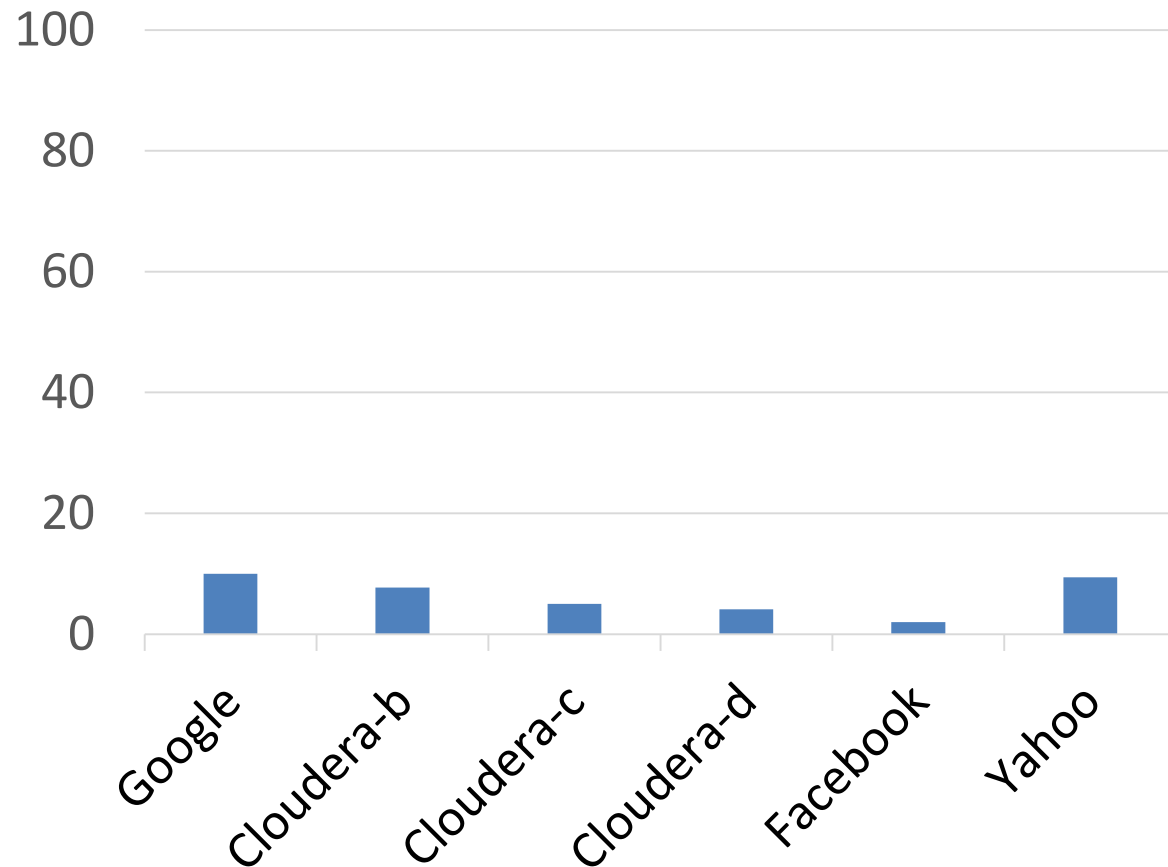
Rationale for Hawk

Typical production workloads

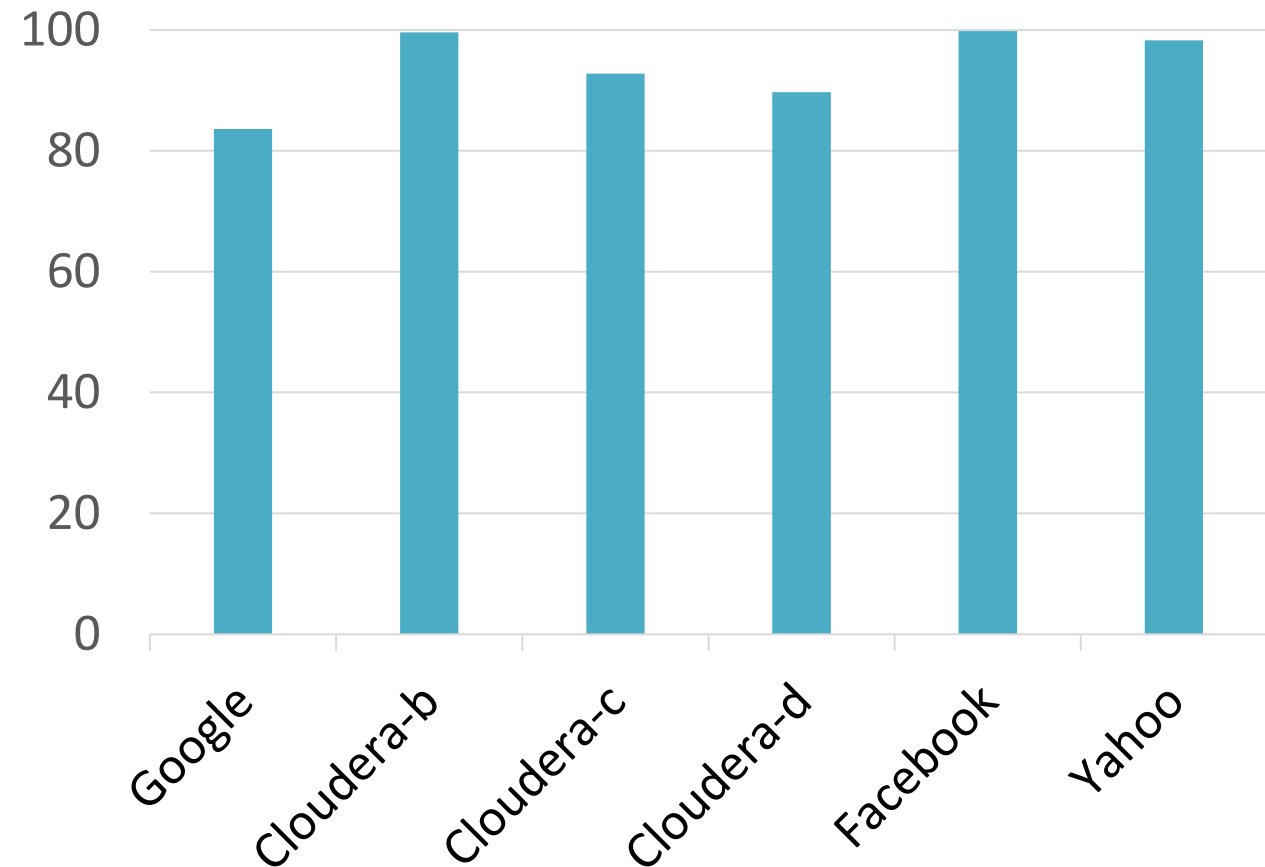


Rationale for Hawk (continued)

Percentage of long jobs

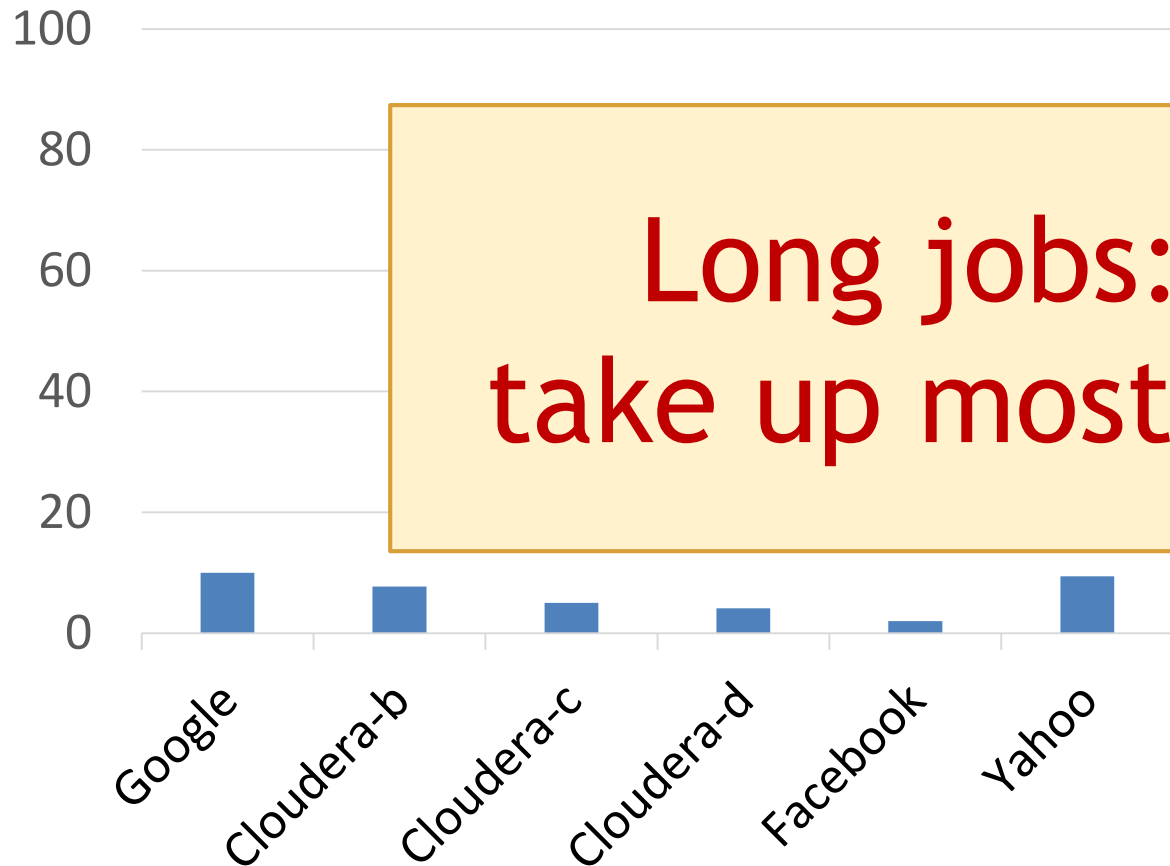


Percentage of task-seconds for long jobs



Rationale for Hawk (continued)

Percentage of long jobs



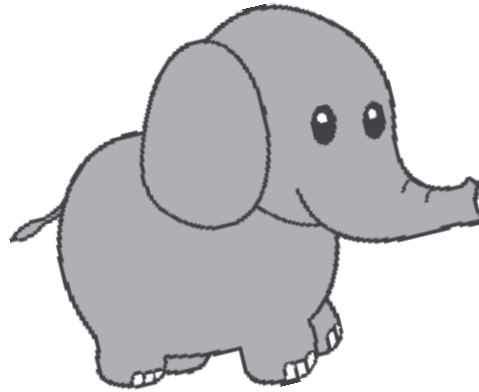
Percentage of task-seconds for long jobs



Long jobs: minority but take up most of the resources

Hawk: hybrid scheduling

Bulk of resources
→ good placement



Long job 1

⋮



Few jobs → reasonable scheduling latency

Latency sensitive
→ Fast scheduling



Short job 1

⋮



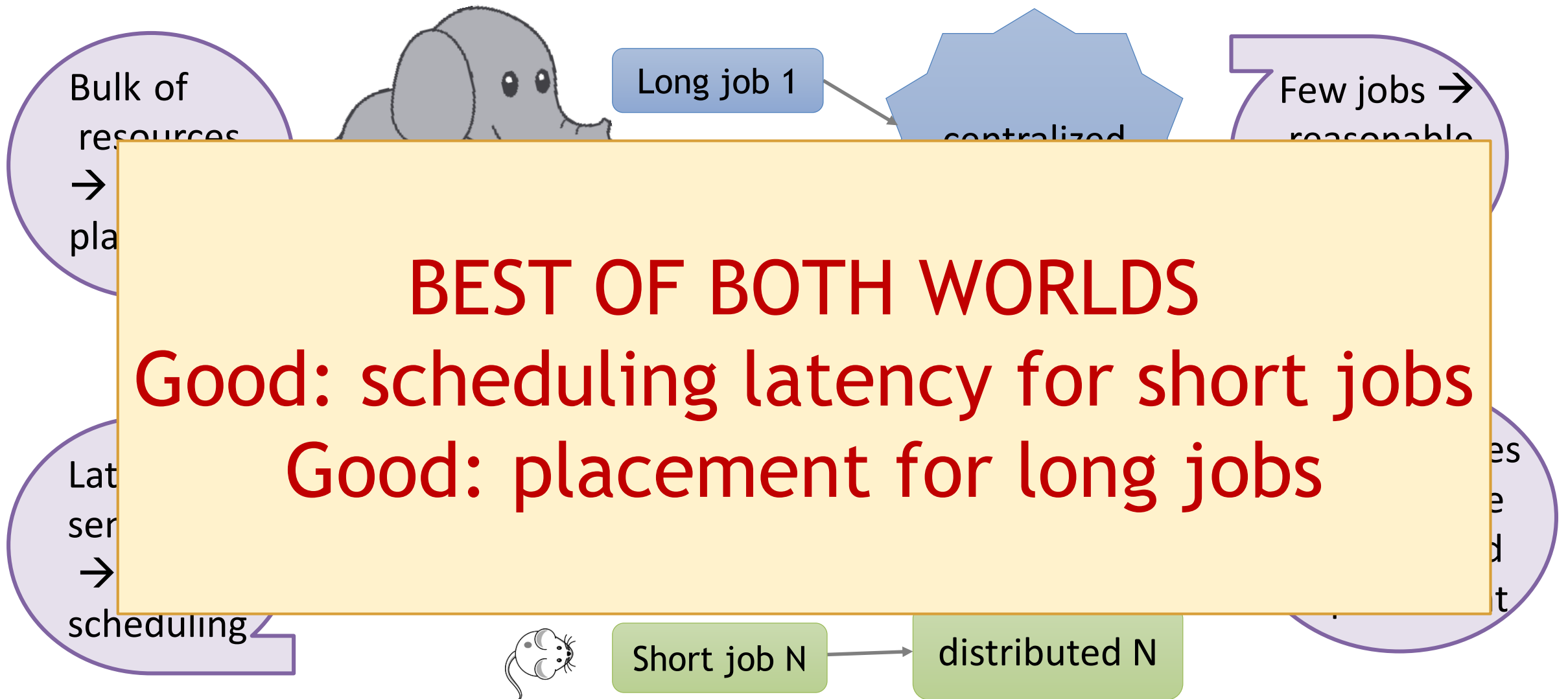
Few resources → can trade not-so-good placement



Short job N



Hawk: hybrid scheduling



Hawk: distributed scheduling

- Sparrow
- Work-stealing

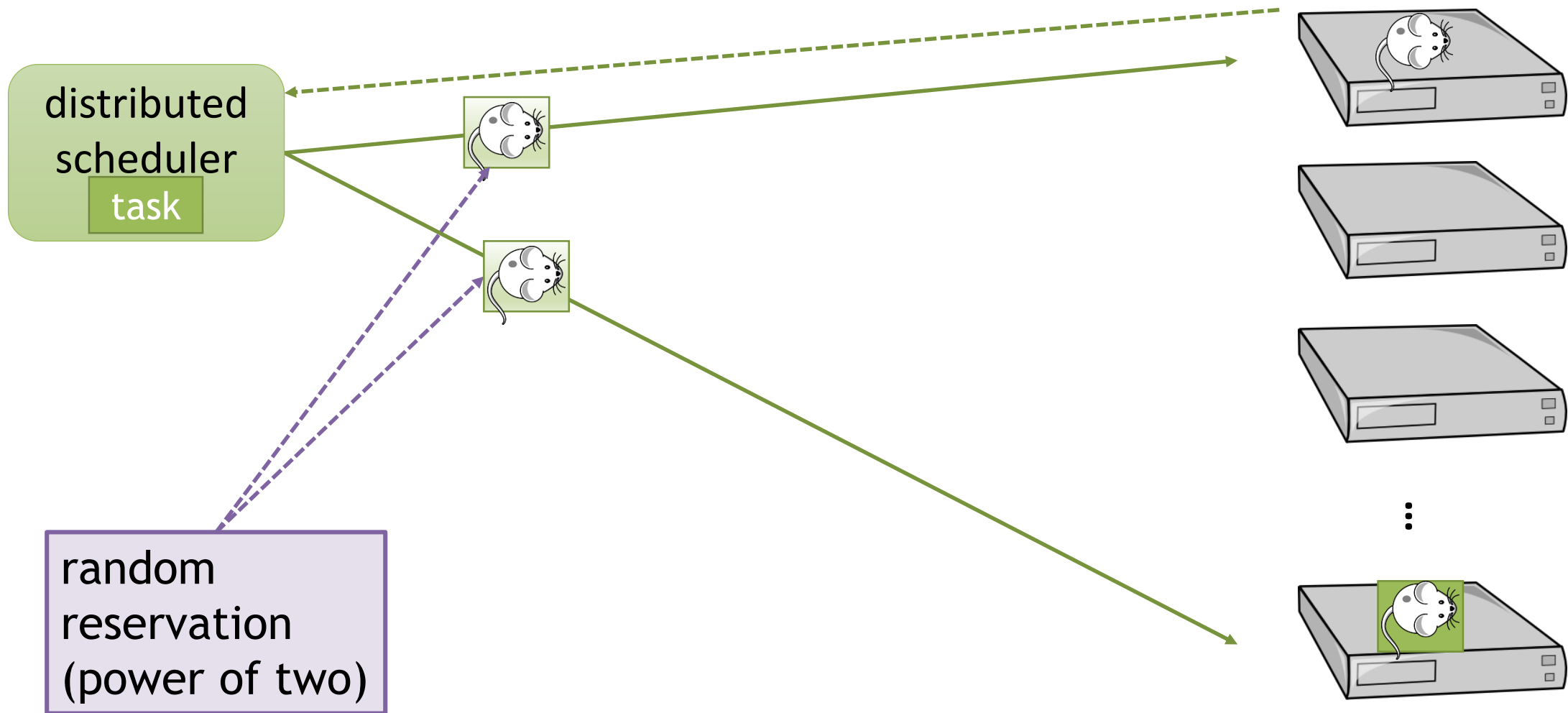
Hawk: distributed scheduling

- Sparrow



- Work-stealing

Sparrow



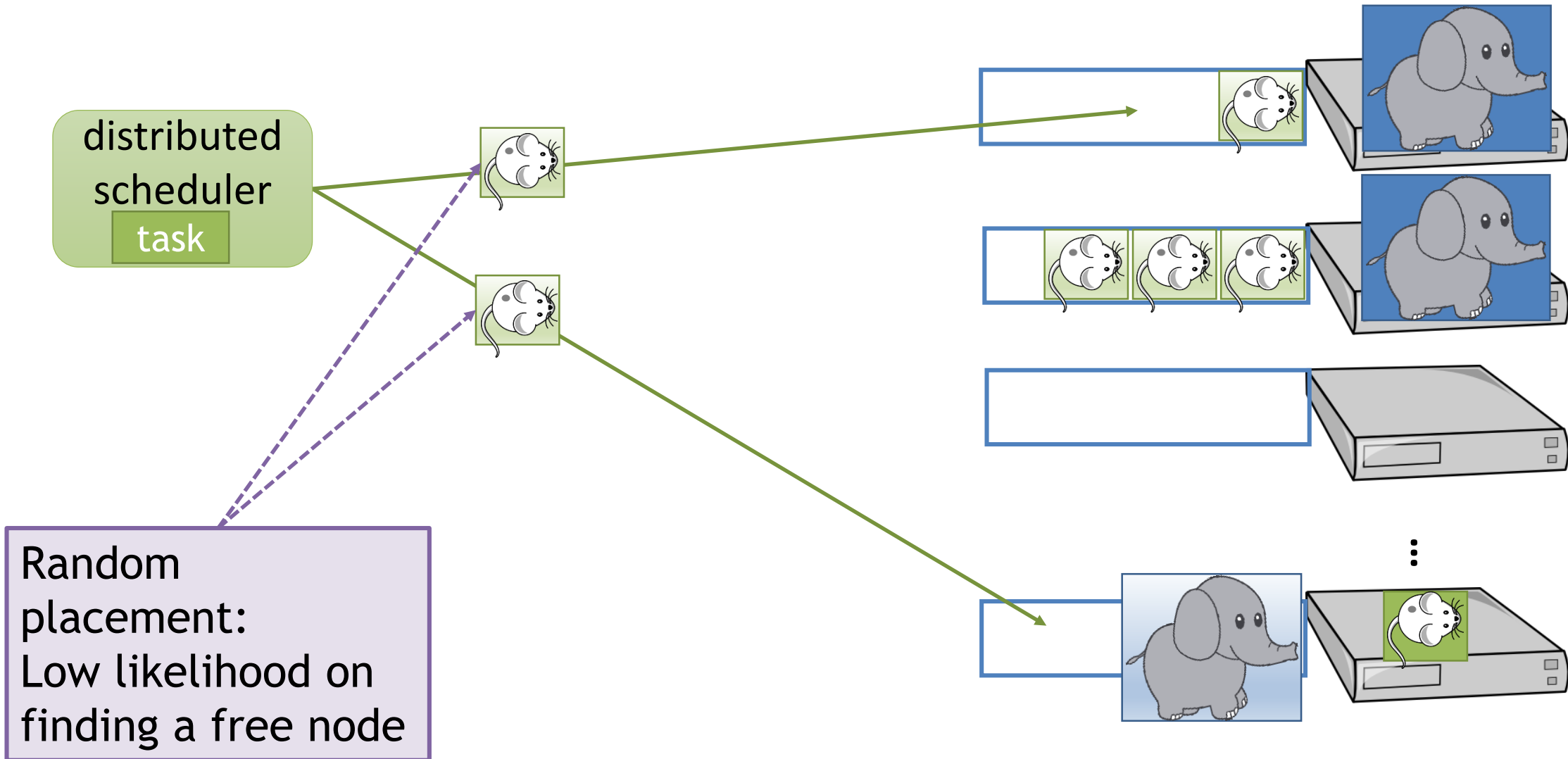
Hawk: distributed scheduling

- Sparrow

- Work-stealing



Sparrow and high load

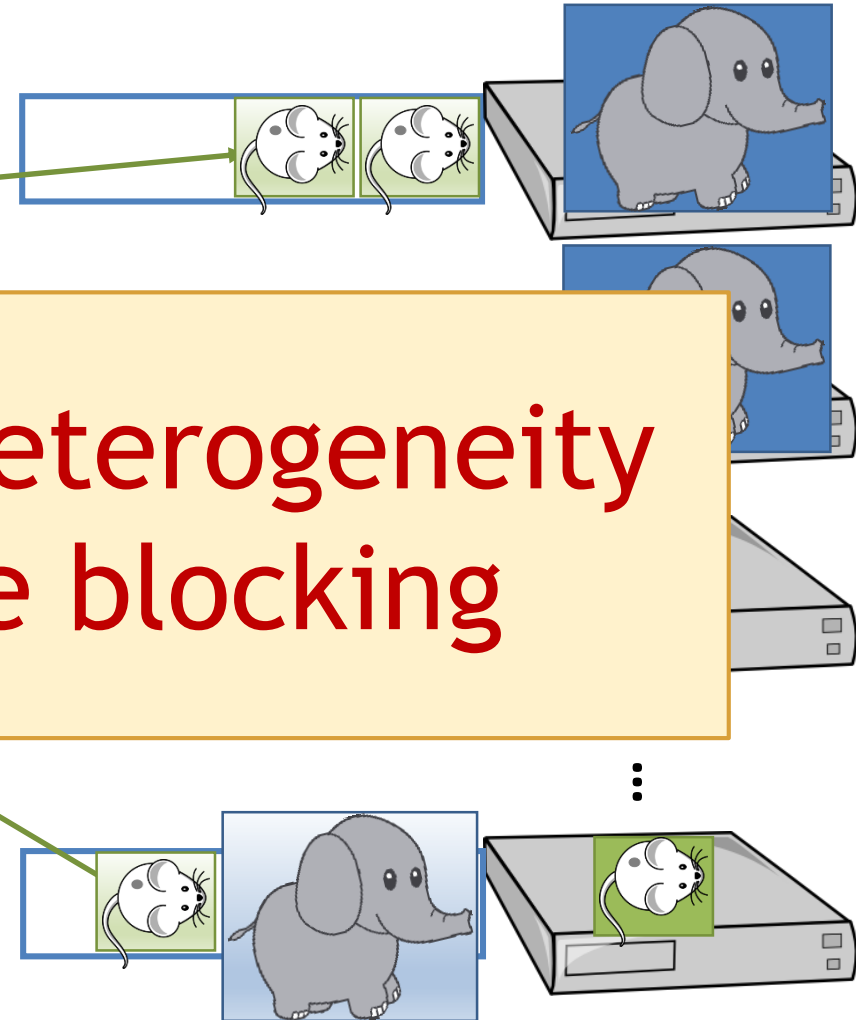


Sparrow and high load

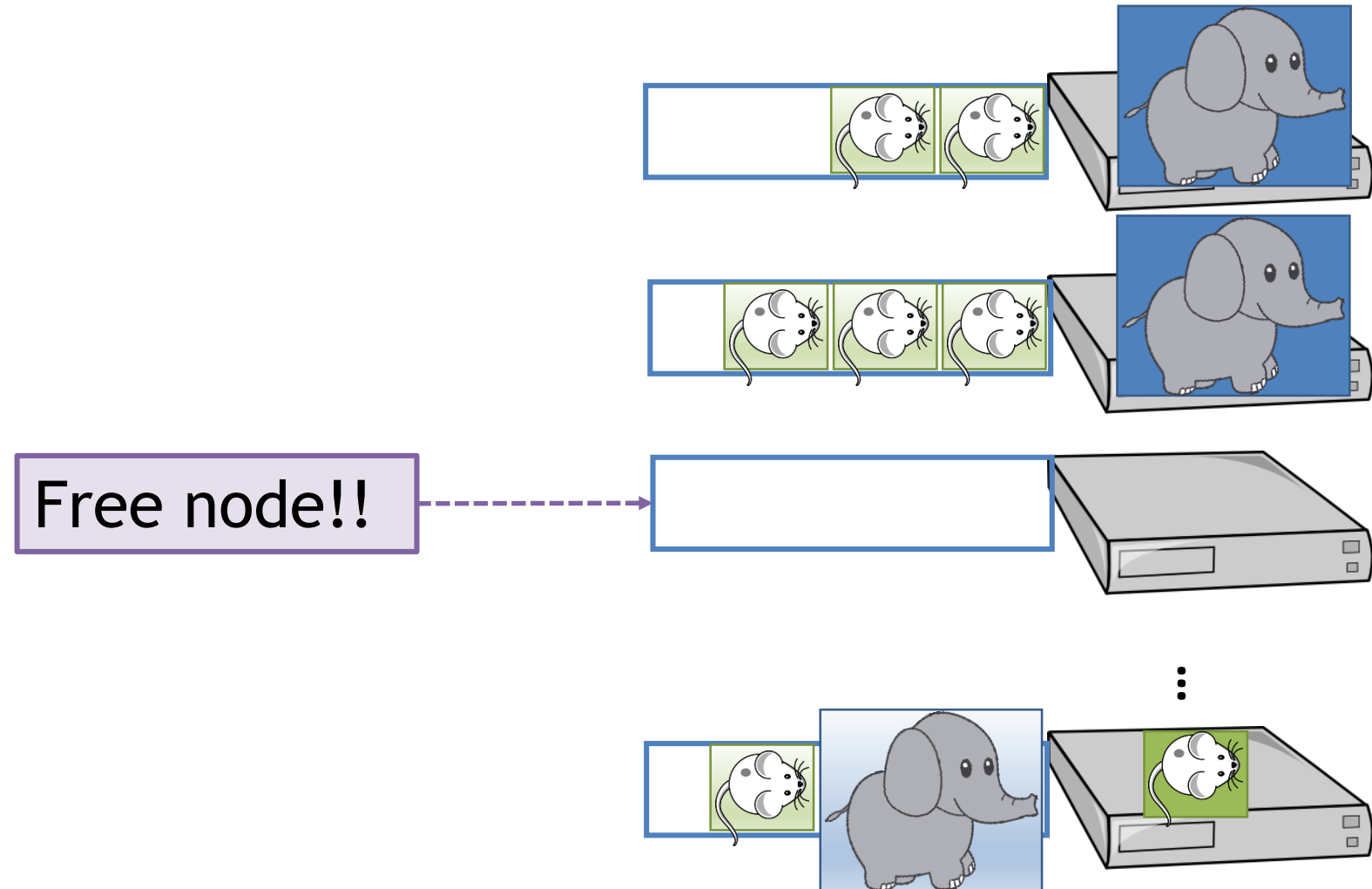
distributed scheduler
task

High load + job heterogeneity
→ head-of-line blocking

Random placement:
Low likelihood on finding a free node



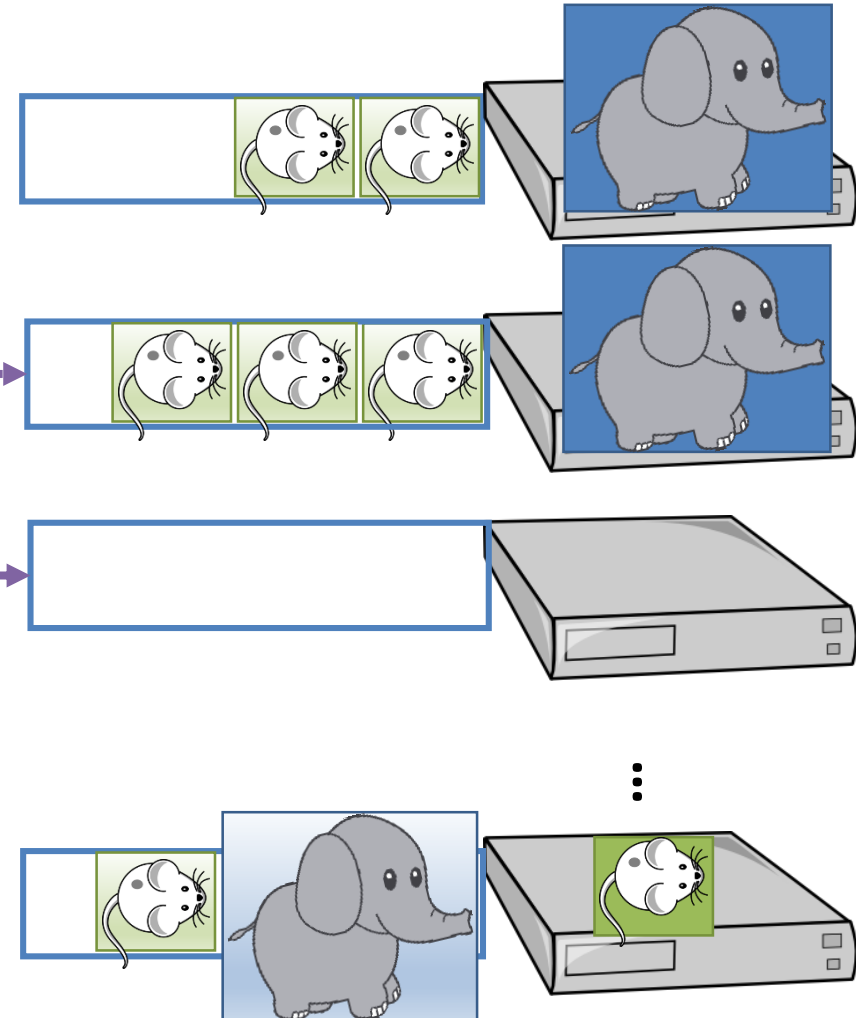
Hawk work-stealing



Hawk work-stealing

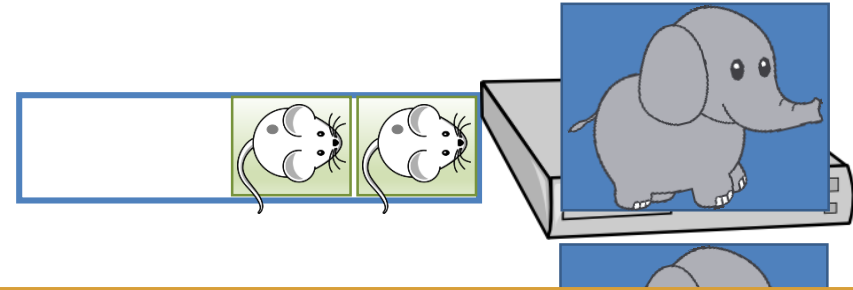
2. Random node:
send **short tasks**
reservation in queue

1. Free node:
contact random
node for probes!



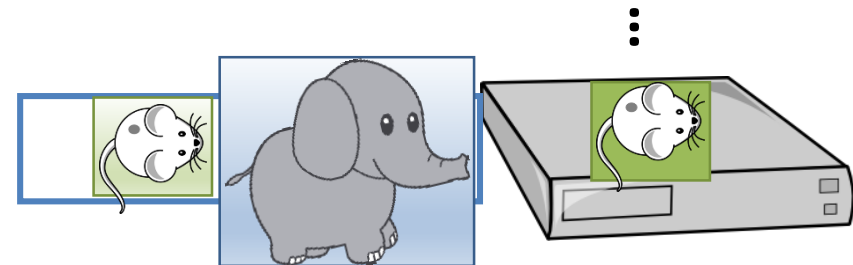
Hawk work-stealing

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

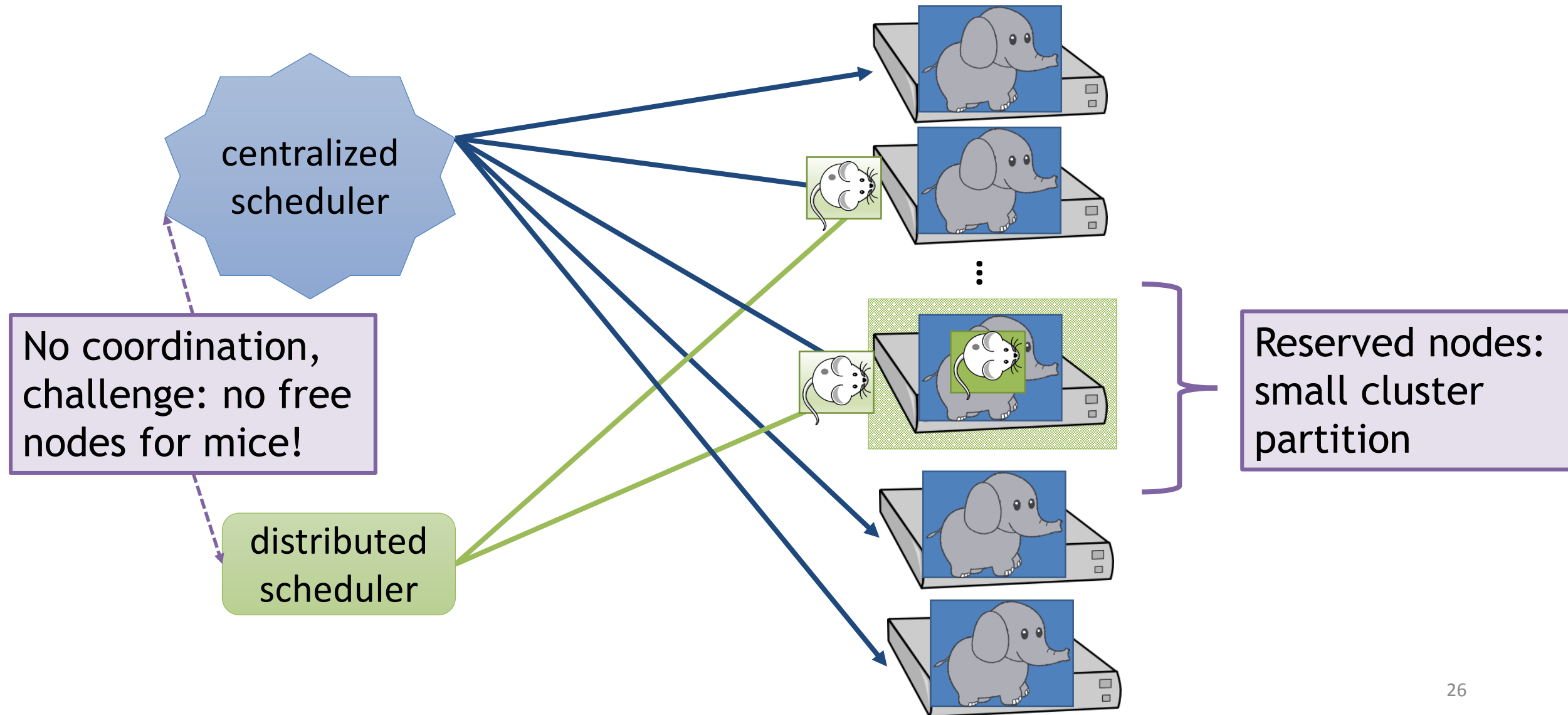


High load \rightarrow high probability
of contacting node with backlog

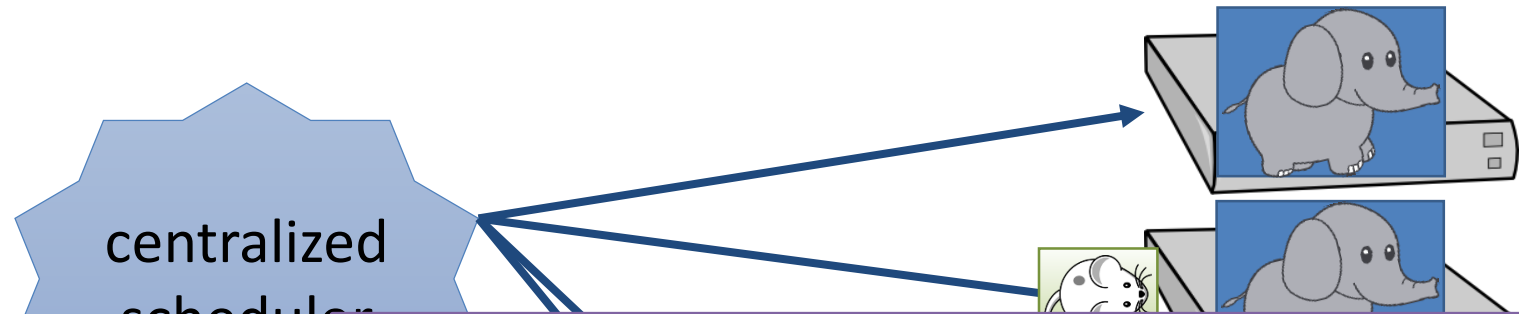
1. Free node:
contact random
node for probes!



Hawk cluster partitioning



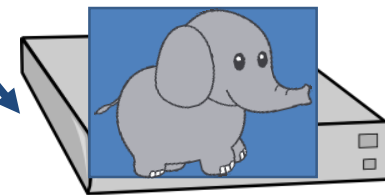
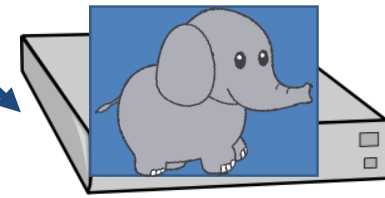
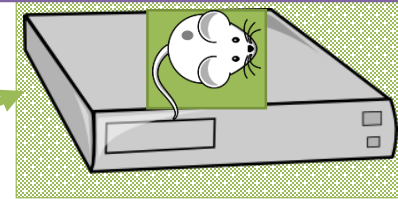
Hawk cluster partitioning



**Short jobs schedule anywhere.
Long jobs only in non-reserved nodes.**

No coordination,
challenge: no free
nodes for mice!

distributed
scheduler



Hawk design summary

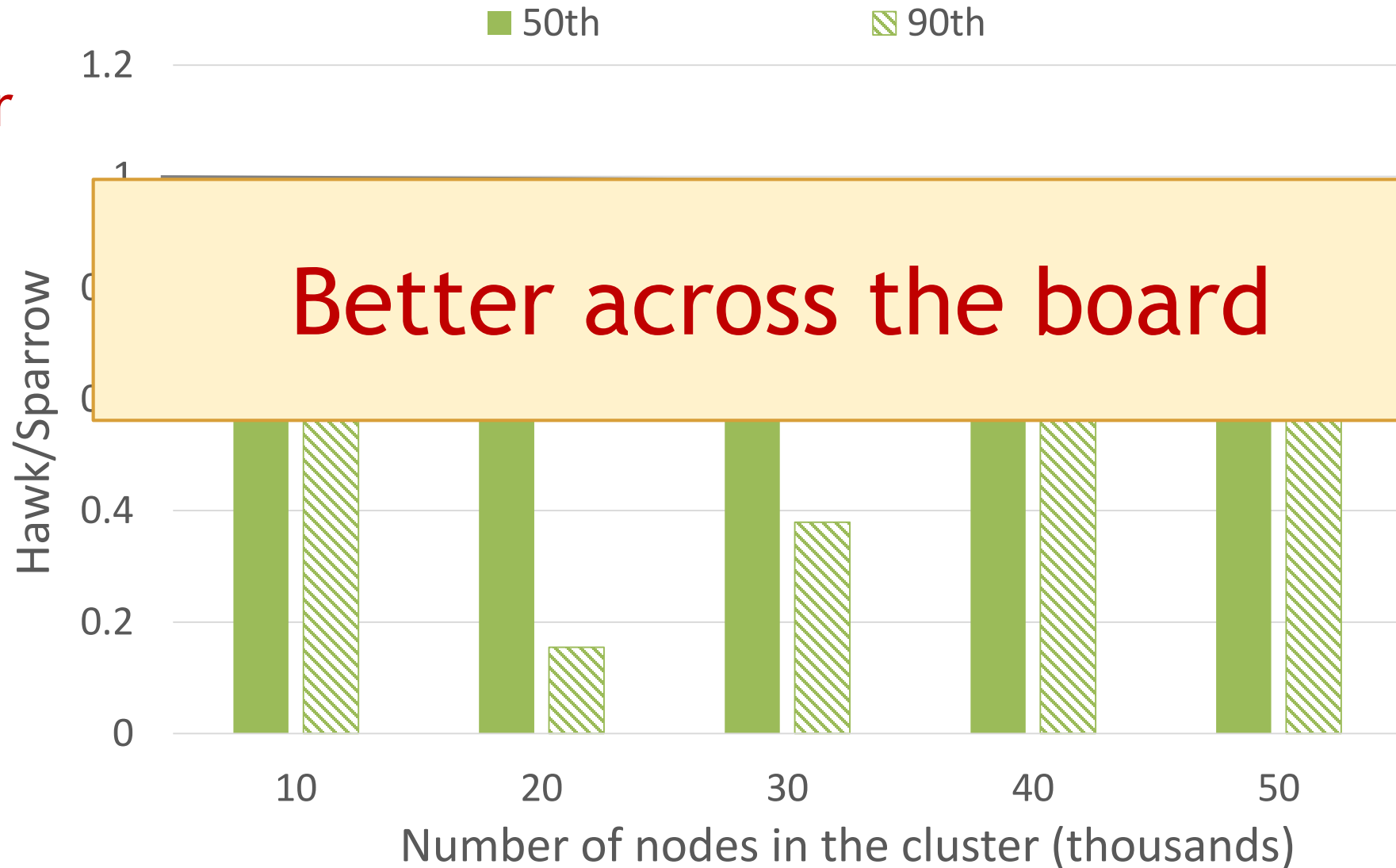
- ✓ Hybrid scheduler:
 long → centralized, short → distributed
- ✓ Work-stealing
- ✓ Cluster partitioning

Evaluation: 1. Simulation

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

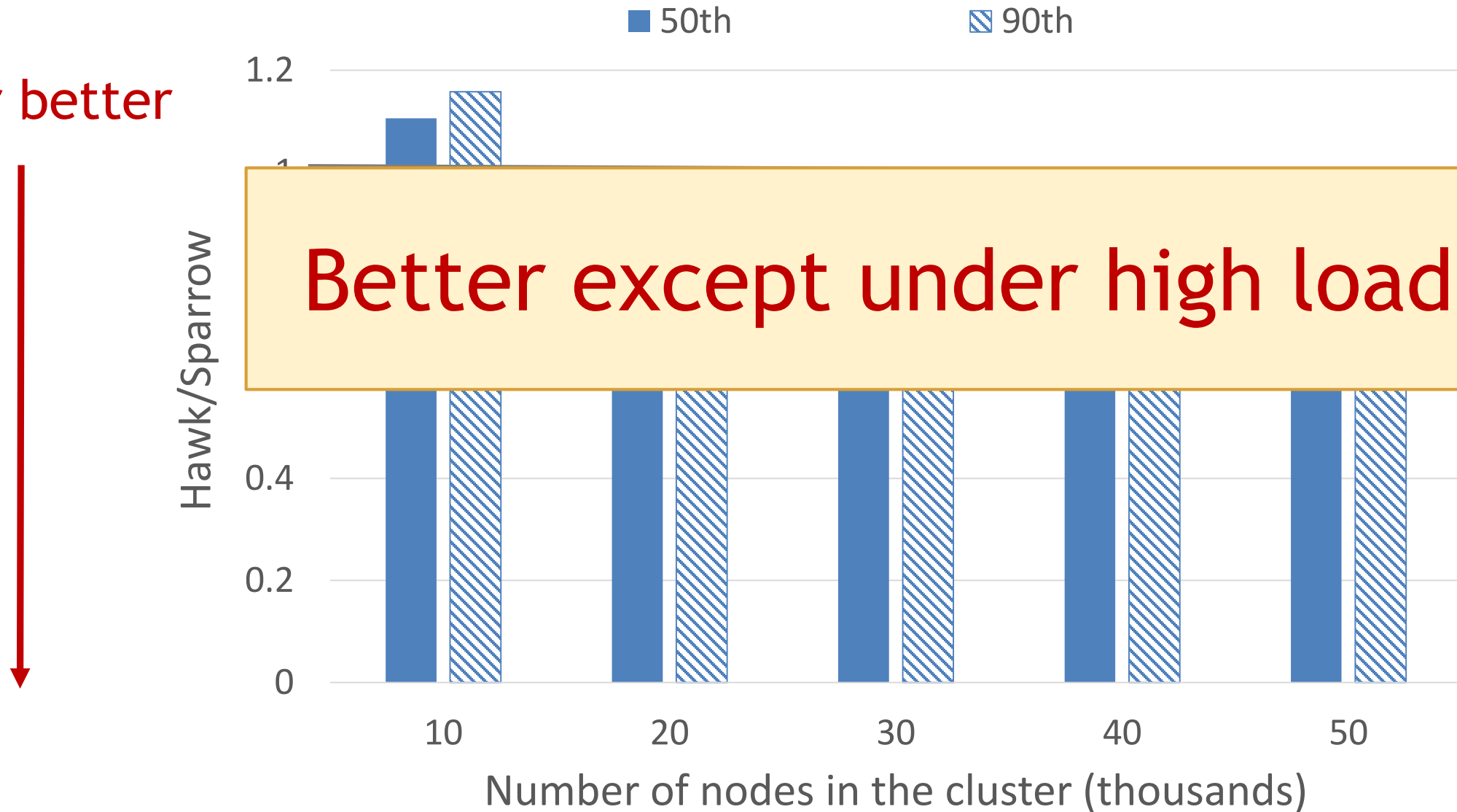
Simulated results: short jobs

lower better



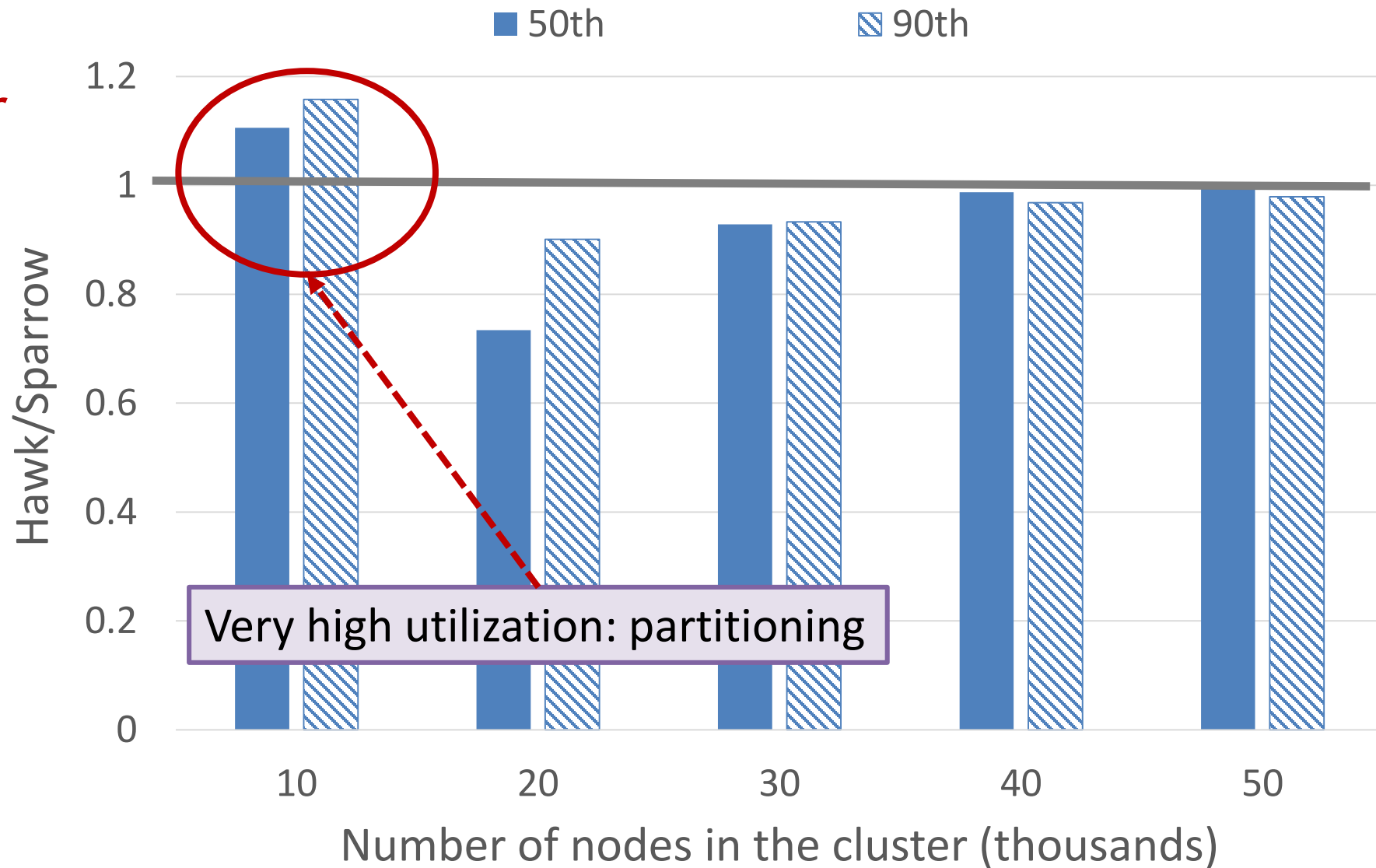
Simulated results: long jobs

lower better



Simulated results: long jobs

lower better

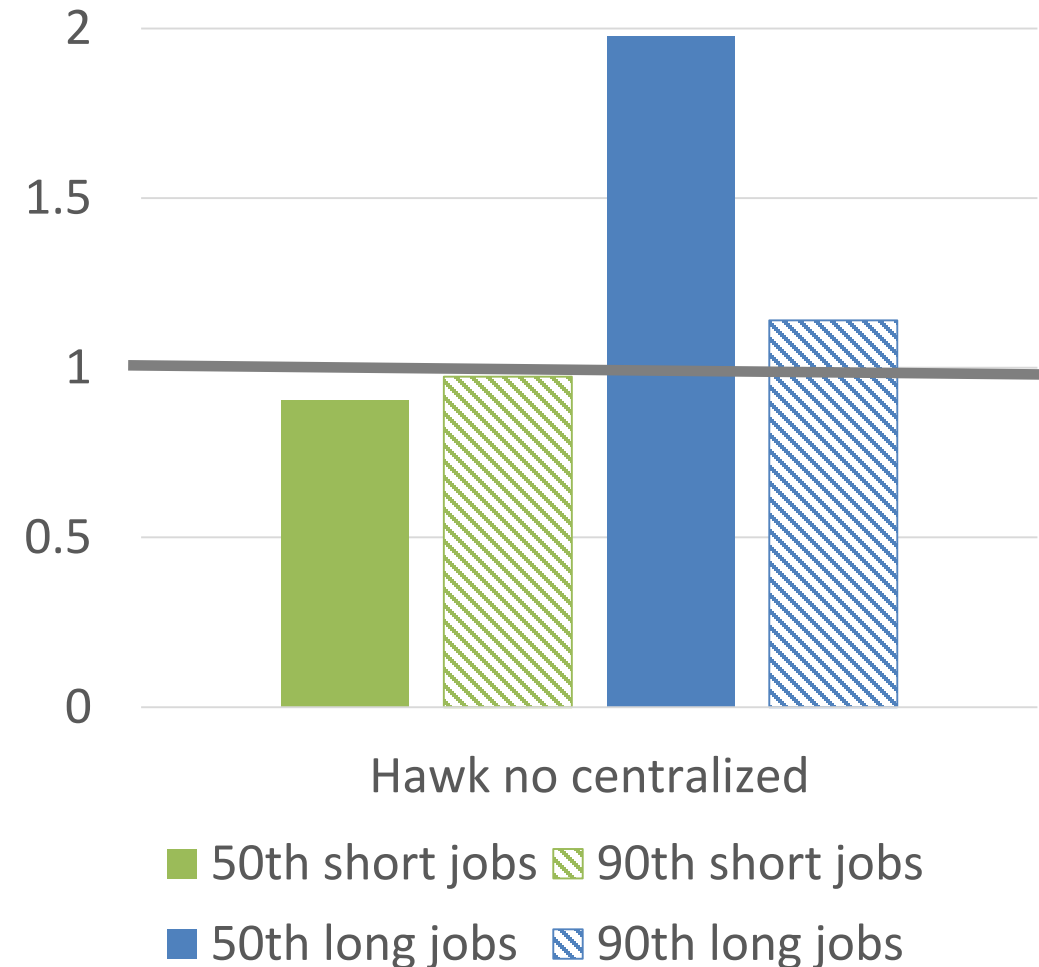


Decomposing Hawk

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning
(normalized to Hawk)

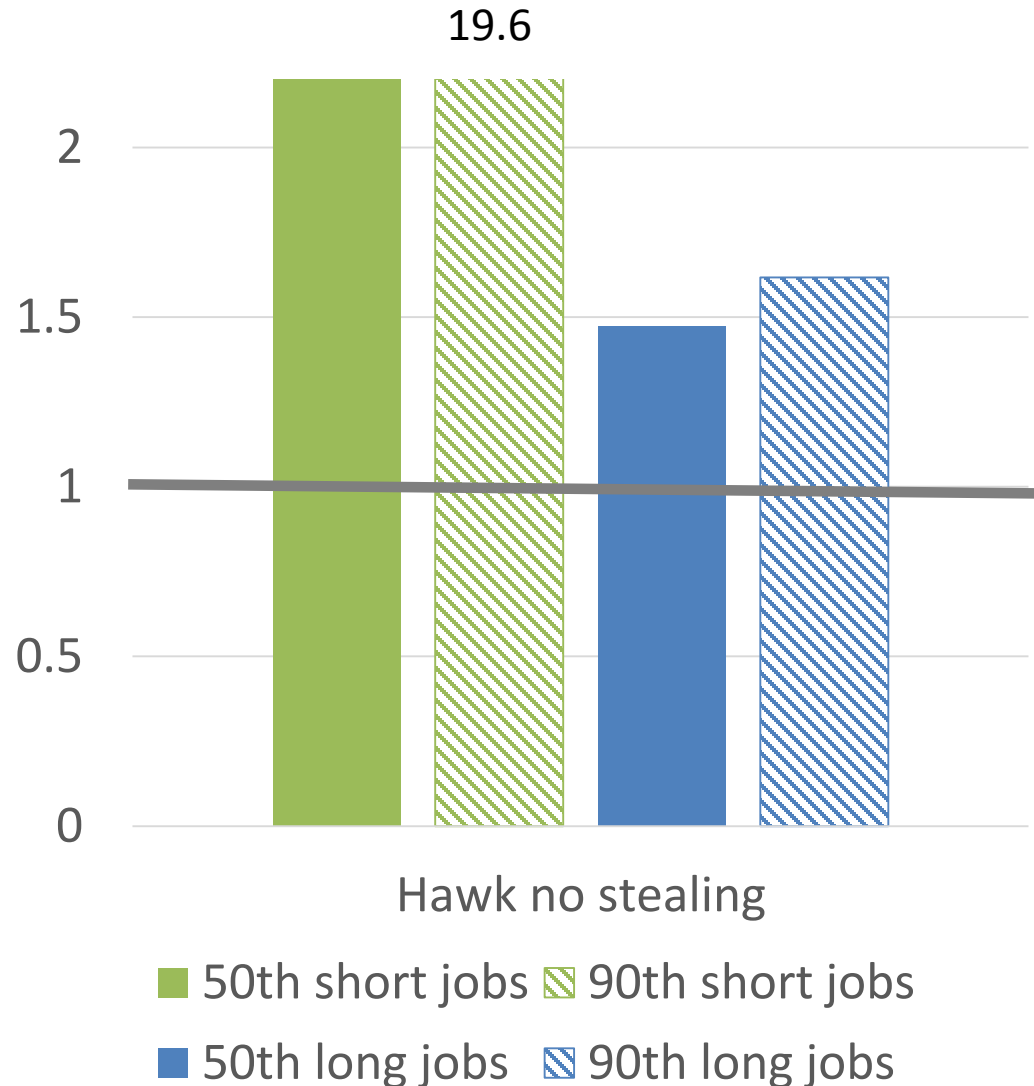
Decomposing Hawk: no centralized

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning
(normalized to Hawk)



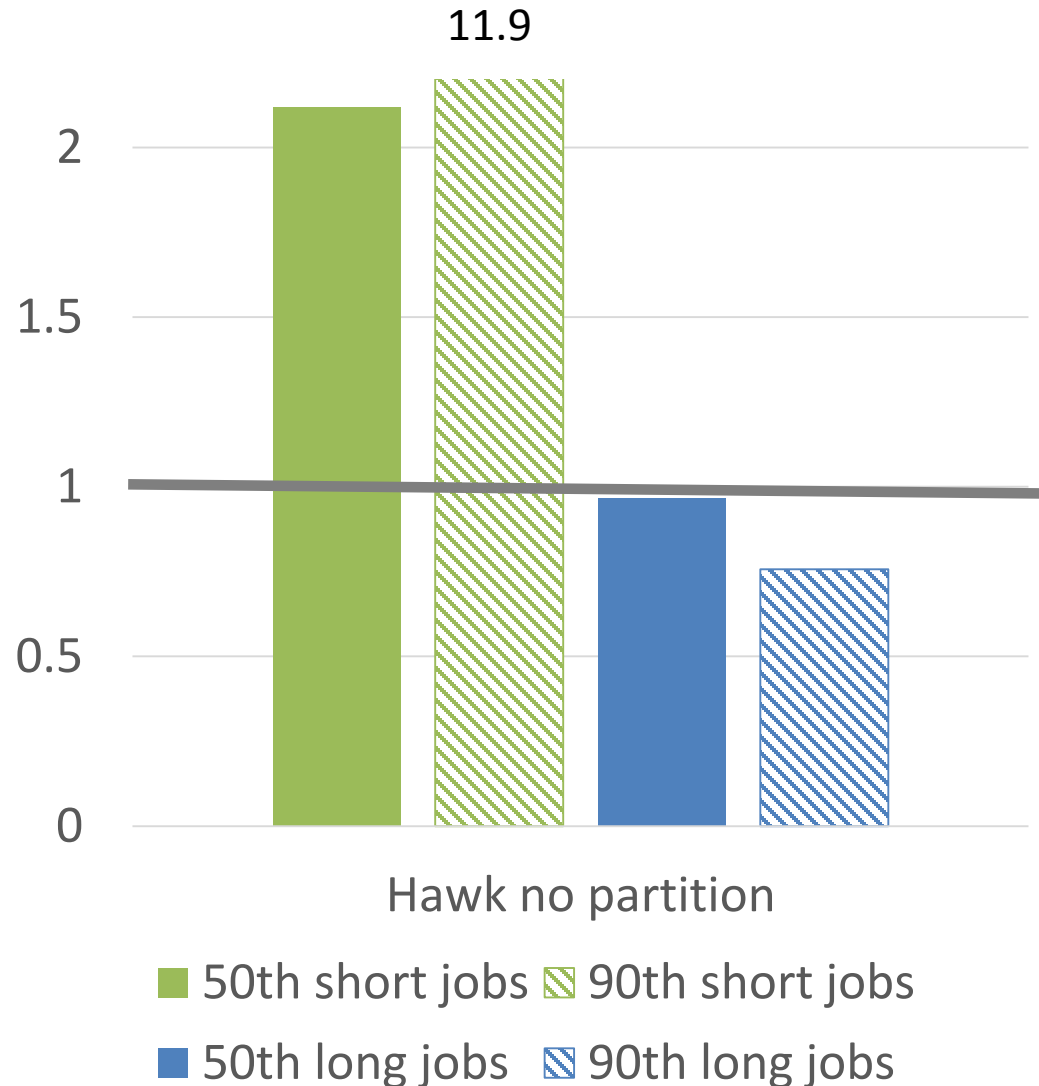
Decomposing Hawk: no stealing

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning
(normalized to Hawk)

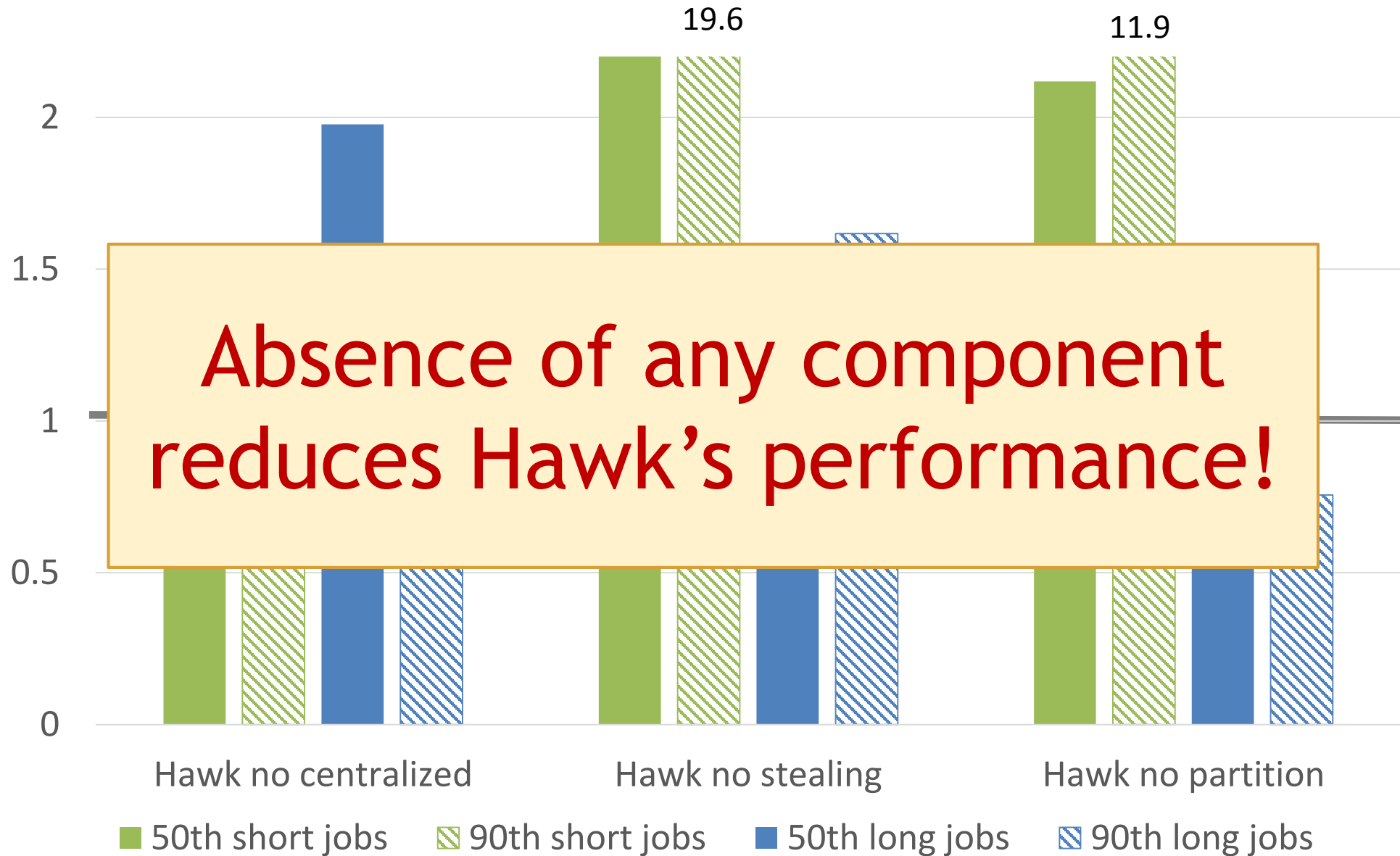


Decomposing Hawk: no partitioning

1. Hawk minus centralized
2. Hawk minus stealing
3. Hawk minus partitioning
(normalized to Hawk)



Decomposing Hawk summary



Sensitivity analysis

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

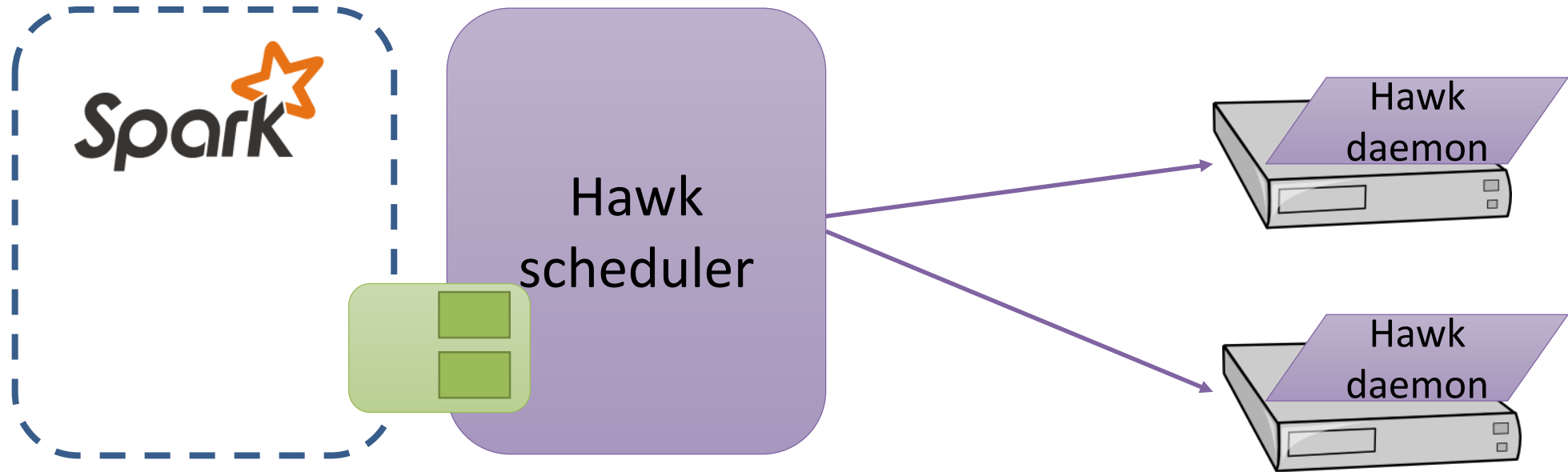
Sensitivity analysis

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

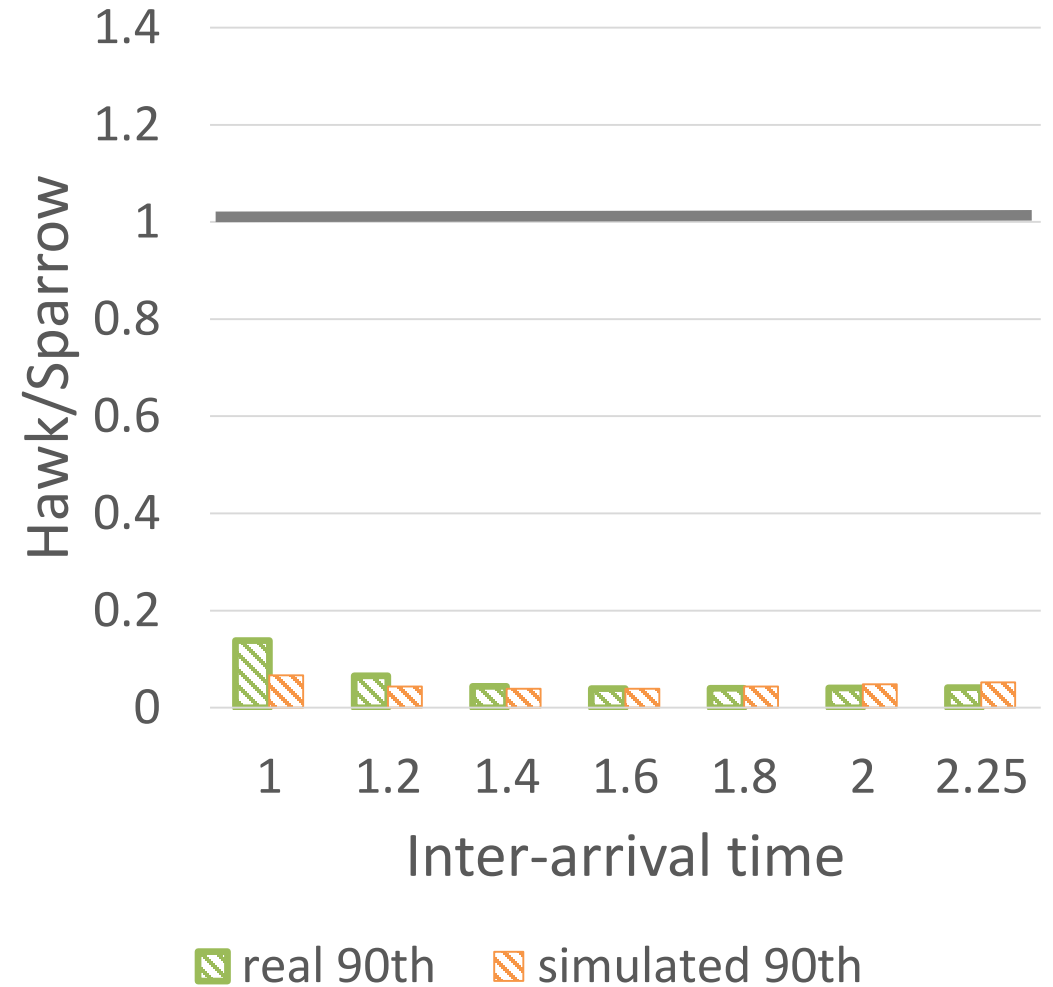
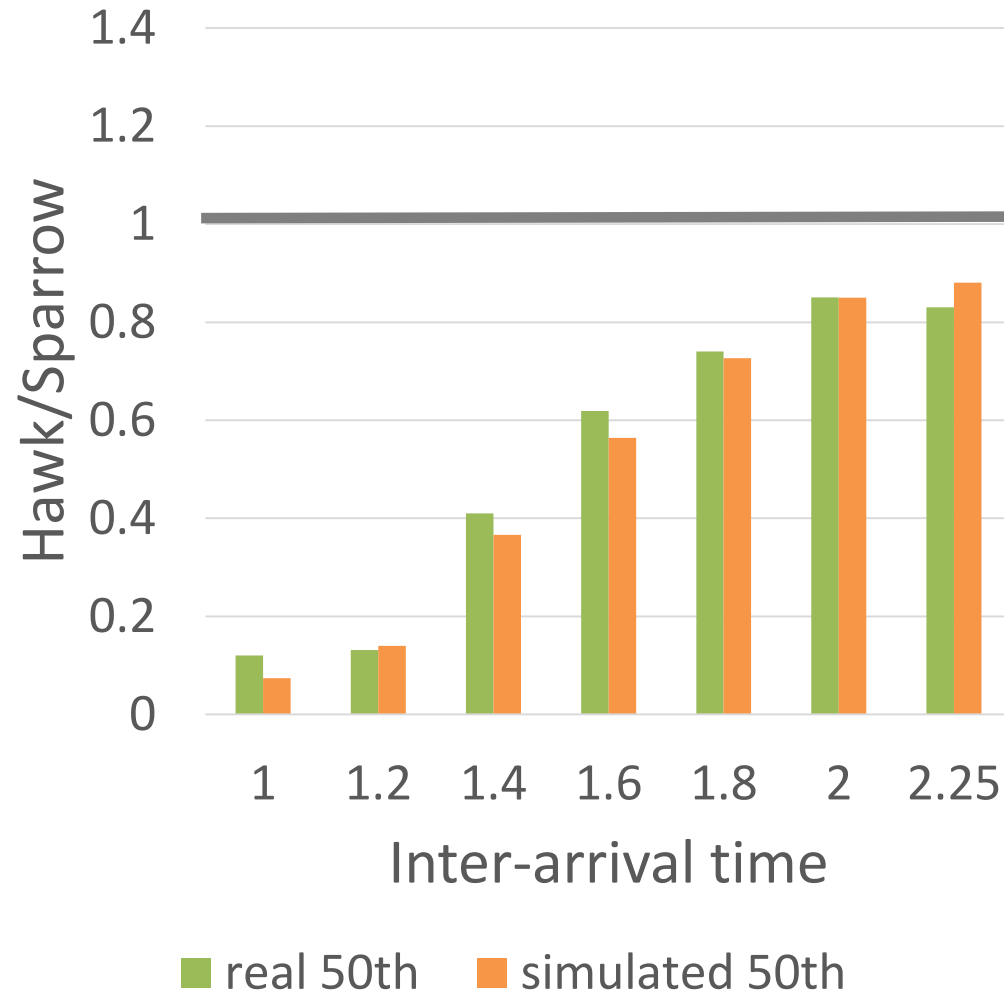


Experiment

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

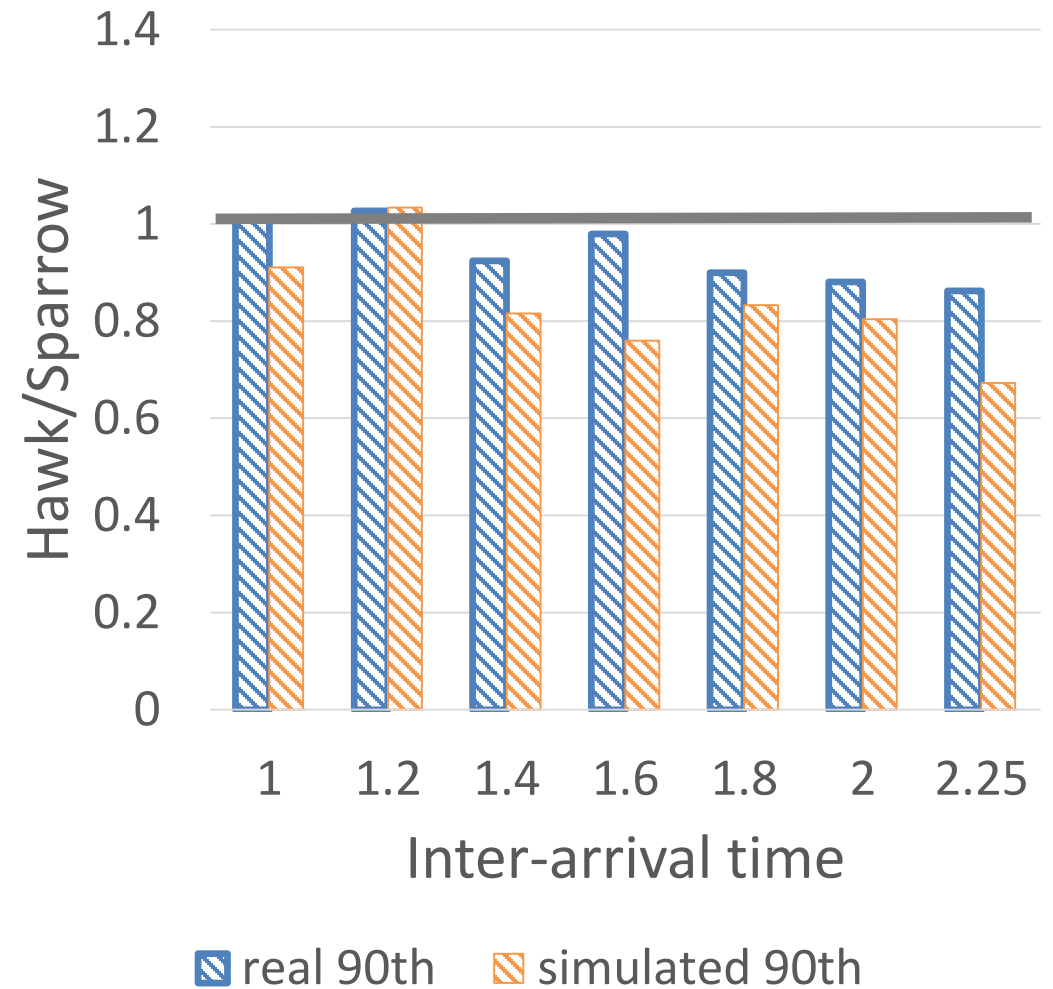
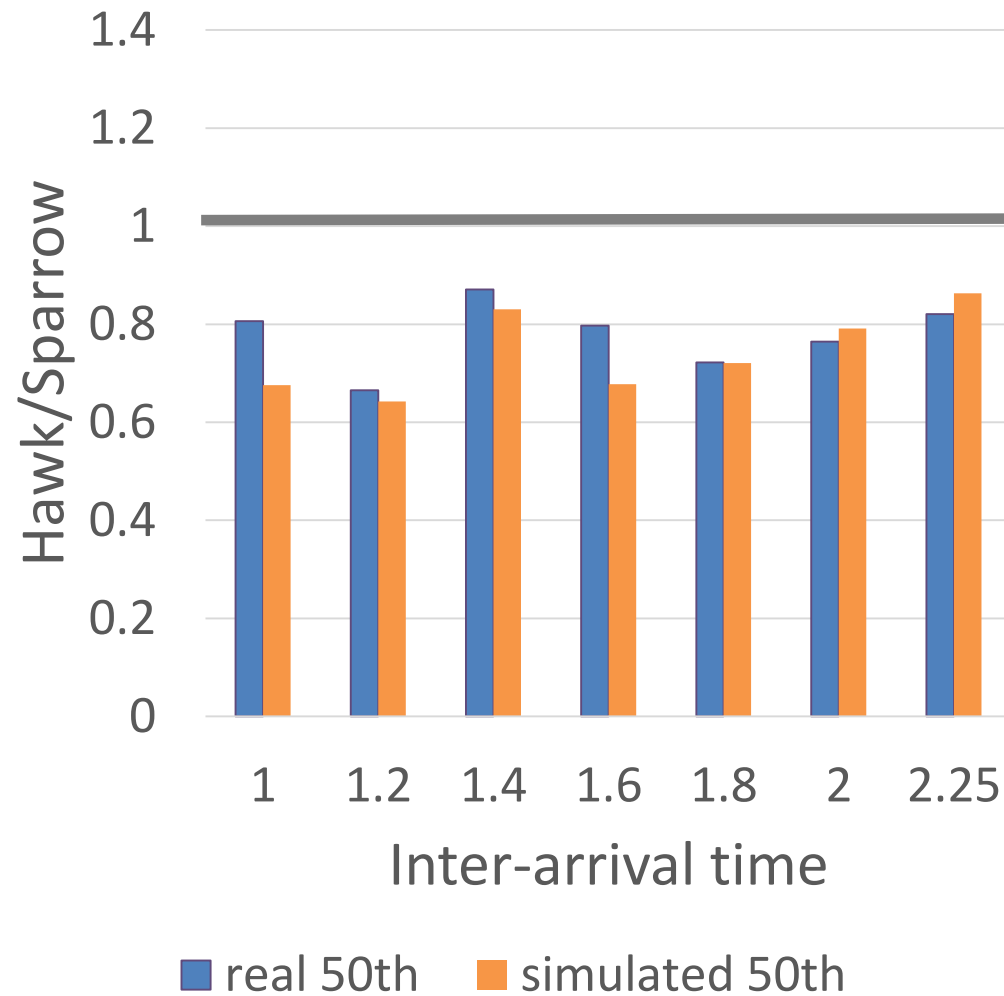
Short jobs

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, SOSPP'09

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

Distributed schedulers: Omega, Sparrow
Eurosys'12, SOSPP'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

