Hawk: Hybrid Datacenter Scheduling

Pamela Delgado, Florin Dinu,
Anne-Marie Kermarrec, Willy Zwaenepoel
Introduction: datacenter scheduling
Introduction: centralized scheduling

Job 1

centralized scheduler

Job N

cluster...
Introduction: centralized scheduling
Introduction: centralized scheduling

Good: placement
Not so good: scheduling latency
Introduction: distributed scheduling

Job 1 → distributed scheduler 1 → cluster
Job 2 → distributed scheduler 2 → cluster
...
Introduction: distributed scheduling

Good: scheduling latency
Not so good: placement

Job 1
- distributed scheduler 1

Job 2
- distributed scheduler 2

Job N
- distributed scheduler N

cluster
1) Introduction

2) HAWK hybrid scheduling
   • Rationale
   • Design

3) Evaluation
   • Simulation
   • Real cluster implementation

4) Conclusion
Hybrid scheduling

- Centralized scheduler
- Distributed scheduler 1
- Distributed scheduler N
- Jobs 1 to N

Cluster
Hawk: hybrid scheduling

✓ Long jobs $\rightarrow$ centralized

✓ Short jobs $\rightarrow$ 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

few

many

most resources

little resources

Long job 1

\[ \vdots \]

Long job M

Short job 1

Short job 2

\[ \vdots \]

Short job N
Rationale for Hawk (continued)

Source: Design Insights for MapReduce from Diverse Production Workloads, Chen et al 2012
Long jobs: minority but take up most of the resources.

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

- Bulk of resources $\rightarrow$ good placement
- Latency sensitive $\rightarrow$ Fast scheduling
- Few jobs $\rightarrow$ reasonable scheduling latency
- Few resources $\rightarrow$ can trade not-so-good placement

Centralized

- Long job 1
- Short job 1
- Short job N

Distributed

- distributed 1
- distributed N

Latency sensitive $\rightarrow$ Fast scheduling

Bulk of resources $\rightarrow$ good placement

Few jobs $\rightarrow$ reasonable scheduling latency

Few resources $\rightarrow$ can trade not-so-good placement
Hawk: hybrid scheduling

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

Random placement: Low likelihood on finding a free node
High load + job heterogeneity \( \rightarrow \) head-of-line blocking

Random placement:
Low likelihood on finding a free node
Hawk work-stealing

Free node!!
1. Free node: contact random node for probes!

2. Random node: send short tasks reservation in queue
Hawk work-stealing

1. Free node: contact random node for probes!

2. Random node: send short tasks reservation in queue

High load → high probability of contacting node with backlog
Hawk cluster partitioning

centralized scheduler

No coordination, challenge: no free nodes for mice!

distributed scheduler

Reserved nodes: small cluster partition
Hawk cluster partitioning

Centralized scheduler

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

No coordination, challenge: no free nodes for mice!

Reserved nodes: small cluster partition

Distributed scheduler
Hawk design summary

✓ Hybrid scheduler:
  long $\rightarrow$ centralized, short $\rightarrow$ 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\textsuperscript{th} and 90\textsuperscript{th} percentiles for short and long jobs

• Normalized to Sparrow
Simulated results: short jobs

lower better

Better across the board
Simulated results: long jobs

Better except under high load
Simulated results: long jobs

lower better

Very high utilization: partitioning
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)
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

Hawk scheduler

Hawk daemon

Hawk daemon
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

![Graph showing inter-arrival time for Hawk/Sparrow with real and simulated 50th and 90th percentiles.](image)
Long jobs

lower better

![Graph showing inter-arrival time and Hawk/Sparrow ratio for real and simulated 50th and 90th percentiles.](image-url)
1. Hawk works well in real cluster
2. Good correspondence implementation/simulation
Related work

Centralized: Hadoop Fair Scheduler, Quincy
    Eurosyst’10, SOSP’09

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

Distributed schedulers: Omega, Sparrow
    Eurosyst’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