Hopper

Decentralized Speculation-aware Cluster Scheduling at Scale

Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, Minlan Yu
Hopper is a decentralized job scheduler which jointly optimizes speculation and job scheduling and speeds up analytics jobs by >50%
Job A

Job B

Schedulers

Workers

straggler

B1

B2

B3

A1

A2

A3
Common fix: **Speculation**

Speculative tasks account for **21%** of resource usage in Facebook Hadoop cluster.
Speculation is a well-honed technique

Duplication (2004) Google

Mantri (2010) Cosmos

LATE (2008) Facebook

GRASS (2014)

Dolly (2013)

...
Speculation is a well-honed technique

How to jointly optimize job scheduling and speculation?

Mantri (2010)

LATE (2008)

Dolly (2013)
speculation
When should a scheduler

start an unscheduled task from a new job

vs. start an unscheduled task from a running job

vs. start a speculative task?
How do schedulers handle speculation today?

• Best effort speculation (practical)
  – budgets no slots. No guarantee when can speculate.

• Budgeted speculation
  – budgets a fixed number of slots. Slots are usually wasted.

No coordination between scheduling and speculation leads to huge performance degradation.
Example: Best effort

Slot 1: A1, A2
Slot 2: A2
Slot 3: A3 (straggler)
Slot 4: B1
Slot 5: B2
Slot 6: B3 (straggler)

Job A:
- A1
- A2
- A3

Job B:
- B1
- B2
- B3
- B4
- B5

Straggler: A3, B3
Example: Best effort

Slot 1: A1, A3+
Slot 2: A2, B3+
Slot 3: A3
Slot 4: B1
Slot 5: B2
Slot 6: B3

Time

Job A:
- A1
- A2
- A3

Job B:
- B1
- B2
- B3
- B4
- B5

Straggler:

- A3+
- B3+
Example: Best effort

Slot 1: A1, A3+
Slot 2: A2, B3+
Slot 3: A3
Slot 4: B1
Slot 5: B2
Slot 6: B3

Job A:
- A1
- A2
- A3

Job B:
- B1
- B2
- B3
- B4
- B5

straggler

A: 22
Example: Best effort

Slot 1: A1, A3+
Slot 2: A2, B3+
Slot 3: A3
Slot 4: B1, B3
Slot 5: B2, B5
Slot 6: B3

Time:

0 10 20 30 40

Job A: A1, A2, A3
Job B: B1, B2, B3, B4, B5

straggler
Example: Best effort

Job A
- A1
- A2
- A3

Job B
- B1
- B2
- B3
- B4
- B5

Straggler

A: 22
Example: Best effort

Inefficient because speculation comes too late.

Job A

Job B

straggler
Example: Budgeted

Inefficient because slots are wasted.

Job A
- A1
- A2
- A3

Job B
- B1
- B2
- B3
- B4
- B5

Stragglers:
- A3
- B4
- B3

Budgeted slots:
- Slot 1
  - A1
  - A3
- Slot 2
  - A2
  - B3
- Slot 3
  - B1
  - B4
- Slot 4
  - B2
  - B5
- Slot 5
  - A3+
  - B4+
- Slot 6
  - B3+
Example: Dynamic allocation

Slot 1
Slot 2
Slot 3
Slot 4
Slot 5
Slot 6

A1
A2
A3
A3+
B1
B2

Job A

B1
B2
B3
B4
B5

Job B

straggler

straggler

straggler
Example: Dynamic allocation

A: 12

Job A

Job B

straggler

straggler

straggler
Example: Dynamic allocation

Job A
- A1
- A2
- A3

Job B
- B1
- B2
- B3
- B4
- B5

Straggler: B3

A: 12
Example: Dynamic allocation

A: 12  B: 24

40% improvement

Job A

A1  A2  A3

Job B

B1  B2  B3  B4  B5

straggler
Example: Dynamic allocation

**Intuition for design**

*dynamically* allocate/budget slots for speculation among jobs

How valuable is one extra slot for a job?
How valuable is one extra slot for a job?

Two regimes:
How valuable is one extra slot for a job?

Two regimes:

- **A**: marginal return is large and constant
How valuable is one extra slot for a job?

Two regimes:

- **A**: marginal return is large and constant
- **B**: marginal return is small

![virtual size]

Job completion time versus the ratio of the number of assigned slots to the number of tasks.
How valuable is one extra slot for a job?

Pareto task duration distribution:

\[
\text{virtual size} = \frac{2}{\beta} \times \# \text{ of tasks}
\]

\( \beta \): straggler likelihood \( \in (1, 2) \)
How valuable is one extra slot for a job?

Scheduler should make sure:

all jobs get their virtual sizes, before any job gets >its virtual size.
Two cases:

1) Jobs **can not all achieve** their virtual sizes

2) Jobs **can all achieve** their virtual sizes

Scheduler should make sure:

all jobs get their virtual sizes, before any job gets > its virtual size.
How should slots be shared?

1) Jobs **can not all achieve** their virtual sizes

   **Theorem**: Assigning the **smallest** jobs their **virtual sizes** is optimal.
   - **Shortest job first**

2) Jobs **can all achieve** their virtual sizes

---

Slots: 

Jobs:

- **Job A** (2)
- **Job B** (3)
- **Job C** (4)
How should slots be shared?

1) Jobs **cannot all achieve** their virtual sizes
   **Theorem:** Assigning the **smallest** jobs their **virtual sizes**
   is optimal. → **Shortest job first**

2) Jobs **can all achieve** their virtual sizes
   **Theorem:** Sharing slots **proportionally** to virtual sizes
   is optimal. → **Larger jobs have more stragglers**

**Jobs:**
- Job A (2)
- Job B (3)

**Slots:**
- S1
- S2
- S3
- S4
- S5
- S6
- S7
- S8
- S9
- S10
Dynamic allocation when decentralized

![Diagram showing dynamic allocation between Job, Scheduler1, Scheduler2, and Workers](image-url)
Dynamic allocation when decentralized

Workers “schedule” tasks in their waiting queues.
Hurdles to dynamic allocation

Any scheduler or worker only knows a subset of jobs

1) How to determine whether cluster is capacity constrained or not?

2) How to perform scheduling policies (assigning to smallest jobs/proportionally sharing)?
Hurdles to dynamic allocation

1) How to determine whether cluster is capacity constrained or not?
   – Assume capacity is constrained by default.

![Diagram of job allocation]

Job → Scheduler1 → Scheduler2 → Worker → ...
Hurdles to dynamic allocation

1) How to determine whether cluster is capacity constrained or not?
   - Assume capacity is constrained by default.
   - After several refusals, believe capacity is not constrained.
Hurdles to dynamic allocation

2) How to perform scheduling policies (assigning to smallest jobs/proportionally sharing)?
   – Power of two choices
Hurdles to dynamic allocation

2) How to perform scheduling policies (assigning to smallest jobs/proportionally sharing)?

– Power of two many choices.
  
  • Workers get enough information to perform scheduling policies.
  • Two choices is not enough for heavy-tailed workload [M. Bramson, et al., Sigmetrics, 2010]
Core features

- Dynamically budgeting slots across jobs
- Scalable: easy to decentralize

Other details

+ Independent of task-level speculation algorithms
+ Multiple phase jobs (DAG)
+ Fairness and policy constraints
+ Locality constraints

See our paper for more details…
How well does Hopper perform?

- Hadoop YARN 2.3 and Spark 0.7.3
- 200 node private cluster
- Real workload from Facebook and Microsoft
- Cosmos from Oct-Dec 2012
- Baseline: Best effort + decentralized scheduler
Decentralized Hopper reduces job completion time by **66%**

- Baseline: best effort speculation + decentralized scheduler

Results are consistent across **speculation algorithms, # of schedulers, and centralized Hopper.**
Decentralized Hopper performs efficient scheduling with little fairness loss

Fairness knob $\varepsilon$: guarantee $(1 - \varepsilon) \times$ perfect fairness

<table>
<thead>
<tr>
<th>Fairness knob $\varepsilon$ (%)</th>
<th>Job Completion Time Reduction (%) in Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair</td>
<td>Facebook workload</td>
</tr>
<tr>
<td>Unfair</td>
<td>Cosmos workload</td>
</tr>
</tbody>
</table>

Graph showing the relationship between the fairness knob $\varepsilon$ and job completion time reduction for Facebook and Cosmos workloads.
Hopper: Decentralized Speculation-aware Scheduling

- Predictable and scalable performance for analytics in large clusters
  - Predictability via speculative (extra) tasks
  - Scalability via decentralized scheduling

Spark & Hadoop prototype; production queries speed-up by 66%
Hopper
Decentralized Speculation-aware Cluster Scheduling at Scale

Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, Minlan Yu
Back up Slides
Centralized Hopper reduces job completion time by 50%

- Baseline: LATE (speculation) + SRPT (job scheduler)
Decentralized Hopper: CDF of gains

CDF

Reduction (%) in Average Job Duration

Facebook workload

Cosmos workload

0 20 40 60 80 100

0 20 40 60 80 100
Fairness

Fairness $\varepsilon$ (%)

(% Slowed Jobs)

Increase (%) in Job duration of Slowed Jobs

%Jobs Average Worst