GRASS: Trimming Stragglers in Approximation Analytics

Ganesh Ananthanarayanan, Michael Hung, Xiaoqi Ren, Ion Stoica, Adam Wierman, Minlan Yu
Next Generation of Analytics

- **Timely** results, even if **approximate**
  - Data deluge makes this necessary
Approximation Dimensions

- **Deadline**: Maximize accuracy within deadline
  “Pick the best ad to display within 2s”

- **Error**: Minimize time to get desired accuracy
  “#cars sold to the nearest thousand”

Optimal Scheduler

- Improve accuracy by **48%**
- Speedup by **40%**

* w.r.t. *state-of-the-art schedulers* (production workloads from Facebook and Bing)
Scheduling Challenge

• **Prioritize** tasks
  – Subset of *tasks* to complete
  – #tasks » #slots (*multi-waved jobs*)
  (*NP-Hard but many known heuristics...*)

• **Straggler** tasks
  – Slowest task can be 8x slower than median task
  – **Speculation**: Spawn a duplicate, earliest wins
    • Google[OSDI’04], FB[OSDI’08], Microsoft[OSDI’10]
**Challenge:** dynamically prioritize between speculative & unscheduled tasks to meet deadline/error bound
Opportunity Cost

Speculative copies consume *extra* resources.

Is speculation worth the payoff?
Roadmap

1. Two natural scheduling designs

2. **GRASS**: Combining the two designs

3. Evaluation of **GRASS**
**Greedy Scheduling (GS)**

*Greedily* improve accuracy, i.e., earliest finishing task

---

<table>
<thead>
<tr>
<th>Task ID</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time remaining</strong></td>
<td>5</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>New copy</strong></td>
<td>2</td>
<td>---</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Accuracy = 7/9

Deadline = 6

(at time = 1)
Resource Aware Scheduling (RAS)

Speculate only if it saves time and resources.

(straggler)

One copy for 5s (vs.) Two copies for 2s

<table>
<thead>
<tr>
<th>Task ID</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time remaining</td>
<td>5</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>New copy</td>
<td>2</td>
<td>---</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Accuracy = 8/9

Deadline = 6
Neither **GS** nor **RAS** is uniformly better
Intuition:

Use **RAS** early in the job (be “conservative”), switch to **GS** towards the end (be “aggressive”)

Theoretical Scheduling Model

- Multi-waved scheduling of tasks
  - Constant wave-width
  - Agnostic to fairness policies
  - Heavy-tailed (Pareto) distribution of task durations
- **Speculation**: GS, RAS, Switching, Optimal

**Theorem:**
Using **RAS** when \( >2 \) waves of tasks remain, and **GS** when \( \leq 2 \) waves of tasks remain is “near-optimal”
How to estimate two remaining waves?

• Wave boundaries are not strict
  – Non-uniform task durations
• Wave-width is not constant

Start with **RAS** and switch to **GS** close to the deadline/error-bound
Learning the switching point

- **GS**-only and **RAS**-only job samples
  - “Exploration vs. Exploitation”
  - Multi-armed bandit solution, \( \varepsilon = 0.1 \)
GRASS ($= GS + RAS$) Scheduler

- **Opportunity Cost** in speculation for stragglers
  - $GS \rightarrow$ Greedy Scheduling
  - $RAS \rightarrow$ Resource Aware Scheduling

- Switch $RAS \rightarrow GS$ close to deadline/error-bound
  - Learn switching point empirically from job samples

- Provably **near-optimal** in theoretical model
Implementation

• Hadoop 0.20.2 and Spark 0.7.3
  – Modified Fair Scheduler
  – Job bins with GS-only and RAS-only samples

• Task Estimators
  – Remaining time is extrapolated from data-to-process
    • progress reports at 5% intervals
  – New copy’s time is sampled from completed tasks
How well does GRASS perform?

• Workload from Facebook and Bing traces
  – Hadoop and Dryad production jobs
  – Added deadlines and error bounds

• Baselines: **LATE** & **Mantri**
  
  ![facebook](facebook.png) ![bing](bing.png)

• 200 node EC2 deployment (m2.2xlarge instances)
Accuracy of **deadline-bound** jobs improve by **47%**

Gains hold across deadlines (lenient and stringent)
GRASS is 22% better than statically picking GS or RAS

... and is near-optimal
Error-bound Jobs

• Overall speedup of 38% (optimal is 40%)
  – Gains hold across all error bounds

• **Exact jobs** (0% error-bound) speed up by 34%

Unified Straggler Mitigation
Conclusion

• Next gen. of analytics: *Approximate* but timely results
• Challenge: Dynamic and unpredictable *stragglers*

• **GRASS** – Conservative *speculation* early in the job; aggressive towards its end

• Evaluation with Hadoop & Spark
  – Accuracy of deadline-bound jobs improve by *47%*
  – Error-bound jobs speed up by *38%*