NURD : Negative-Unlabeled Learning for Online Datacenter Straggler Prediction

Yi Ding*, Avinash Rao†, Hyebin Song‡, Rebecca Willett†, Henry Hoffmann†

*MIT CSAIL, † University of Chicago, ‡ Penn State University

Stragglers in Datacenter Computation

Task parallelism:

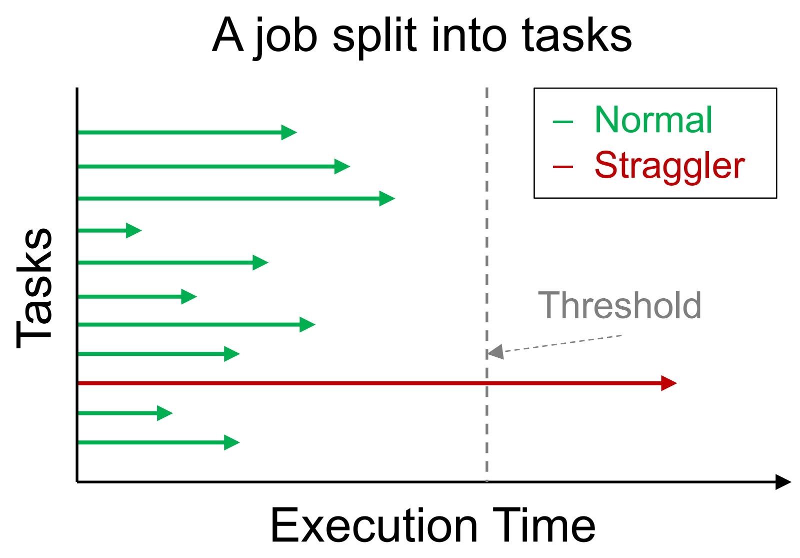
- A job is split into parallel tasks.
- Job completion time depends on the slowest task.

Stragglers:

- Rare, but extremely slow tasks within a job, e.g. > P95.
- Causes such as data skew, resource constraints, etc.

Straggler impact:

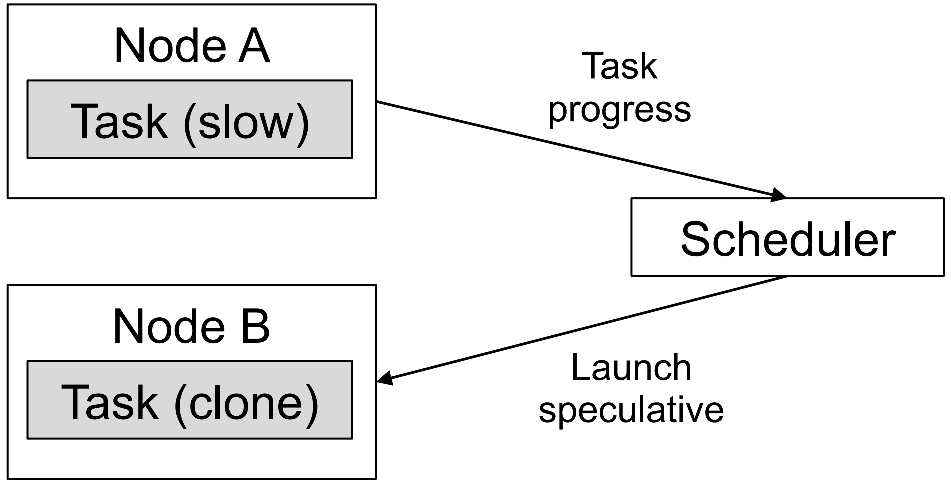
- Delay job completion time by 50% in 20% of Google jobs.



Straggler Mitigation

Speculative execution:

- Monitor job execution and launch duplicates of tasks that are slower.
- Need to *predict* stragglers early and accurately.



Straggler Prediction Problem

Predict task latency based on task and node features such as resource usage, scheduling behavior, machine capacity, μarch features.

Prior Work:

- Heuristic based: LATE (OSDI’08), Mantri (OSDI’10)
    - Using task progress metrics to estimate remaining time or slowdown analytically.
    - Not accurate enough or hard to generalize.
- Machine learning based: Wrangler (SOCC’14)
    - Using machine learning to predict latency.

Wrangler (SOCC'14)

Waiting for data collection to have both stragglers and nonstragglers in training.

- It can take a long time to wait for stragglers to appear.
- *Can we make accurate predictions without observing labeled stragglers in training?*

Offline training on jobs from past execution.

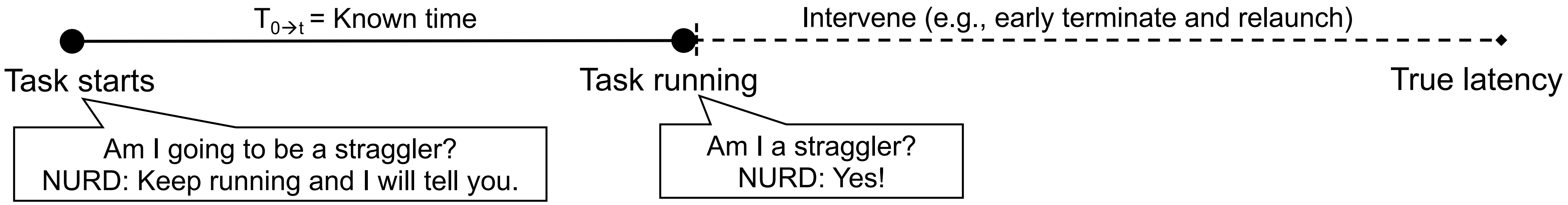
- Characteristics of each job are unique. No guarantee that past executions can generalize.
- *Can we do online training and prediction?*

This Work

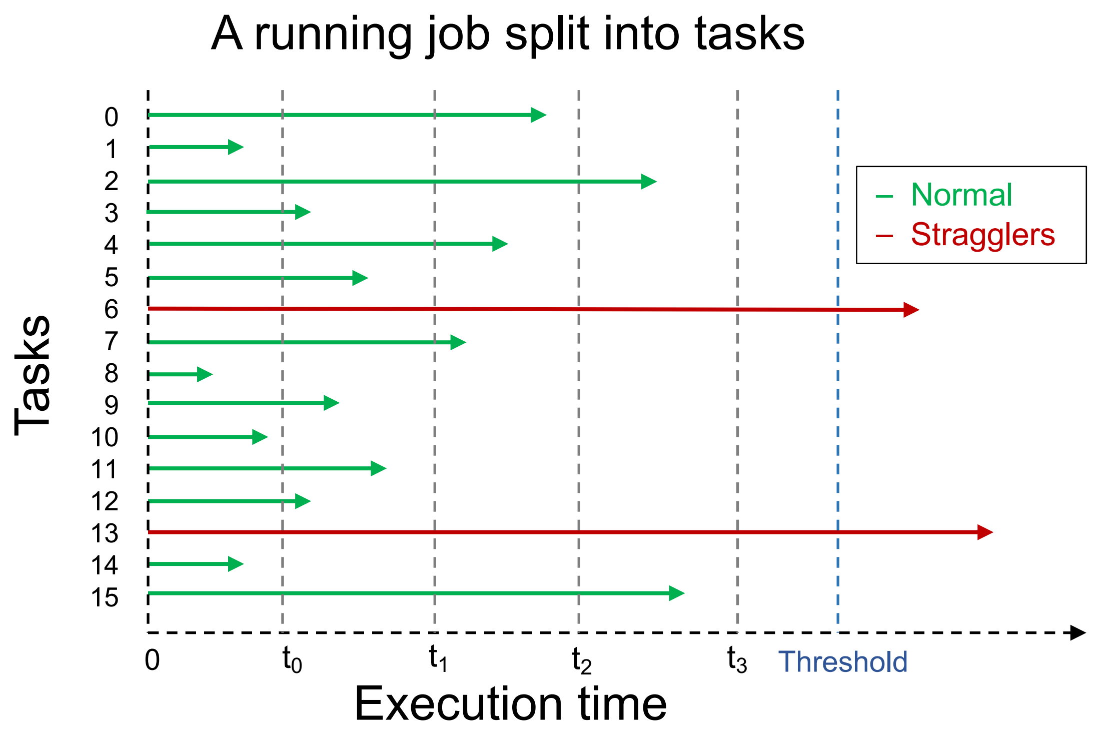
NURD: ***N***egative-***U***nlabeled Learning with ***R***eweighting and ***D***istribution-compensation

- Positive: stragglers, true latency unrevealed
- Negative: nonstragglers, finished tasks, true latency revealed
- Unlabeled: running tasks, true latency unrevealed, including both nonstragglers and stragglers

Goal: constructs a unique predictive model for each job online while it executes.


$T_{0 \to t}$ = Known time
Intervene (e.g., early terminate and relaunch)
Task starts
Task running
True latency
Am I going to be a straggler? NURD: Keep running and I will tell you.
Am I a straggler? NURD: Yes!

Illustration



Timepoint	Finished tasks	Test (running) tasks
t_0	1,8,10,14	0,2,3,4,5,6,7,9,11,12,13,15
t_1	1,3,5,8,9,10,11,12,14	0,2,4,6,7,13,15
t_2	0,1,3,4,5,7,8,9,10,11,12,14	2,6,13,15
t_3	0,1,2,3,4,5,7,8,9,10,11,12,14,15	6,13

When any task is predicted to straggle at any timepoint, it will be terminated and relaunched immediately.

Contributions

NURD benefits:

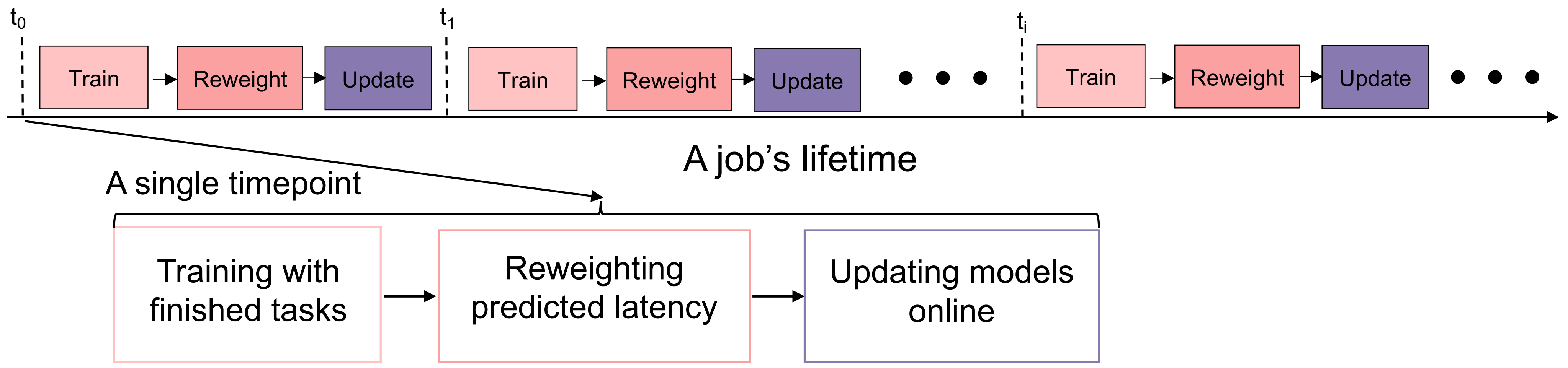
- No reliance on previous jobs --- train and predict online on the current running job.
- No reliance on complete labels --- only have nonstragglers and running tasks in training.

Evaluation on Google and Alibaba production traces. Compared to Wrangler, NURD has

- 35 and 21 percentage point increases in F1.
- Up to 9.3 and 8.9 percentage point improvements in job completion time.

NURD Framework

Key idea: train a latency predictor using finished tasks (i.e., non-stragglers), and reweight latency predictions based on a function of dissimilarity between finished and running tasks.



Step 1: Training with Finished Tasks

At t-th timepoint, train a regression model h_t on finished tasks.

At t-th timepoint for the i-th running task, the predicted latency will be

$$\hat{y}_{ti} = ht(x_{ti})$$

But, these predictions will be biased towards finished tasks (i.e., non-stragglers).

Step 2: Reweighting

Predictions $\hat{y}_{ti}$ trained on finished tasks will be biased towards nonstragglers.

To reduce bias, NURD reweights $\hat{y}_{ti}$ using a weighting function $\mathrm{w}_{\mathrm{ti}} \in [0, 1)$ such that

$$\hat{y}_{ti}^{adj} = \frac{\hat{y}_{ti}}{\mathrm{w}_{\mathrm{ti}}}$$

Intuitively, we want

- When i-th task's features are similar to finished tasks', $\mathrm{w}_{\mathrm{ti}} \rightarrow 1$, $\hat{y}_{ti}^{adj} \approx \hat{y}_{ti}$.
- When i-th task's features are different finished tasks', $\mathrm{w}_{\mathrm{ti}} \rightarrow 0$, $\hat{y}_{ti}^{adj} \gg \hat{y}_{ti}$.

Propensity score matches our intuition

- Definition: conditional probability that a task belongs to the class of finished tasks given its features at t-th timepoint. More details in the paper.

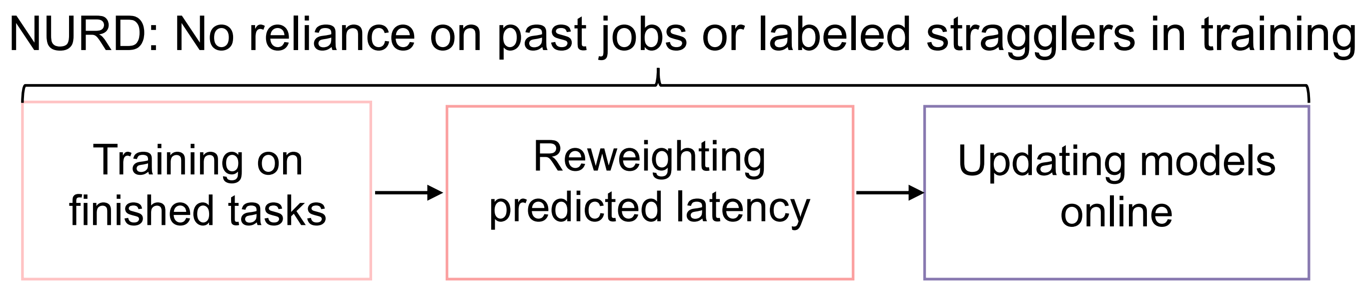
Step 3: Updating Models Online

As the job is running, NURD accumulates finished tasks.

NURD uses new finished tasks to update both latency predictor and weighting function.

NURD improves prediction results as it collects more finished tasks.

Conclusion


NURD: No reliance on past jobs or labeled stragglers in training
Training on finished tasks
Reweighting predicted latency
Updating models online

Thank you!

Yi Ding, Avinash Rao, Hyebin Song, Rebecca Willet, Henry Hoffmann. NURD: Negative-Unlabeled Learning for Online Datacenter Straggler Prediction. Conference on Machine Learning and Systems (MLSys), 2022.