Reducing Average Job Completion Time for DAG-style Jobs by Adding Idle Slots

Yubin Duan and Jie Wu
Dept. of Computer and Information Sciences
Temple University, USA

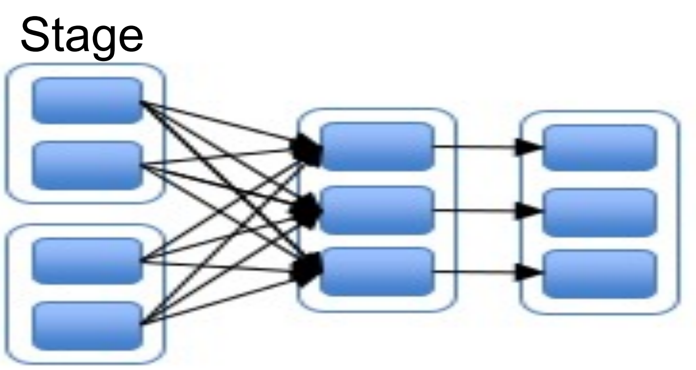
GLOBECOM 2022

Outline

1. Introduction
2. Model
3. Idle-Aware Scheduler
   - Optimal conditions
   - RL-based scheduler
4. Experiment
5. Conclusions

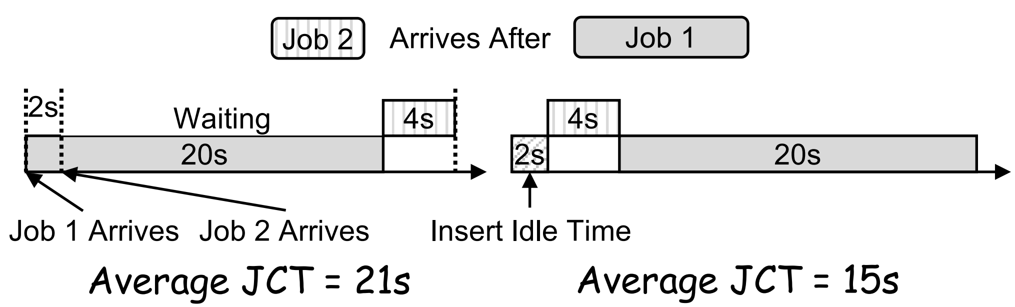
1. Introduction

- DAG-style job scheduling
  - Big data processing jobs usually have DAG-style comp. graphs
  - Scheduler:
    - Determine starting time of each stage
    - Decide number of executors allocated to each stage
- Objective
  - Minimize average job completion time (JCT) for online arrival jobs
    - JCT of each job: finish time – arrival time



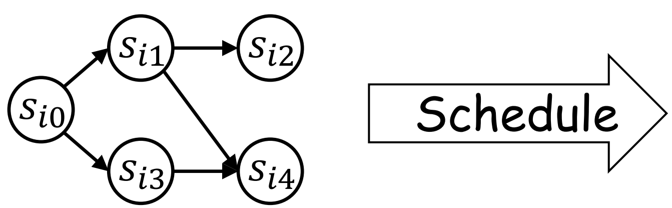
Motivation

- Challenges
  - DAG scheduling problem is NP-hard
    - Complex precedence constraints
  - Unknown online arrival pattern brings additional challenges
- Observation
  - Inserting deliberate idle time can reduce average JCT



2. Model

- List scheduling approach
  - Stage-level scheduling
    - Ordered list of processing sequence for job i: O_i
    - Parallelism level for stage j in job i: p_{ij}
    - Deliberate idle time for stage j in job i: d_{ij}



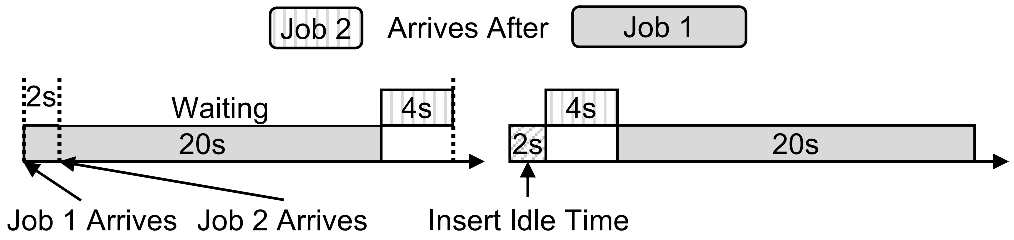
$$O_i = [s_{i0}, s_{i1}, s_{i3}, s_{i4}, s_{i2}]$$

3. Idle-Aware Job Scheduler

- Optimal conditions for one-stage jobs

> Theorem 1: For two adjacent jobs J_1 and J_2, there exists an idle slot with length d_1 such that inserting it before J_1 could reduce the average JCT of J_1 and J_2 when $0 < (a_2 - a_1) \leq (l_1 - l_2)/2$ and $l_1 > l_2$.

- Insights
  - Small jobs waiting for large jobs would enlarge average JCT
  - Inserting idle slots before small jobs can prevent this case



Optimal Idle Time

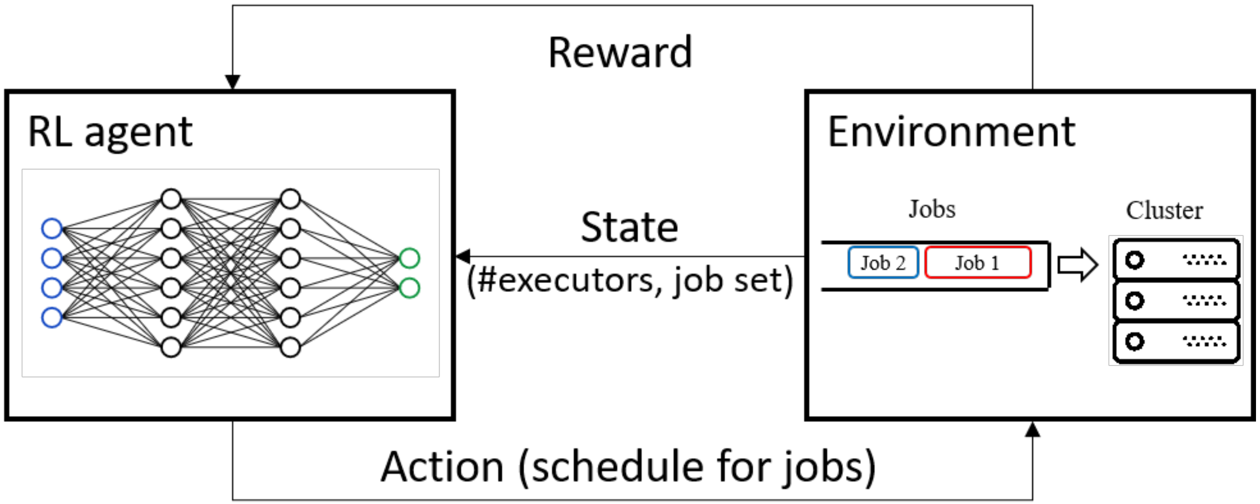
- Need online arrival patterns to calculate
  - Optimal idle time: $d^* = \operatorname*{argmin}_d \mathbf{E}[\eta|d]$
    - η: average JCT. For the two-job case:

$$\eta' = \begin{cases} (\max\{x, l_1+d_1\} + l_1+d_1+l_2-x)/2, & 0 \le d_1 < x; \\ (\max\{x+l_2, d_1\}+l_1+l_2)/2, & d_1 \ge x. \end{cases}$$

  - Hard to find closed-form solutions
- Learn the unknown online arrival pattern
  - Assumption: job arrival pattern is stable

RL-based Scheduler

- Reinforcement learning framework



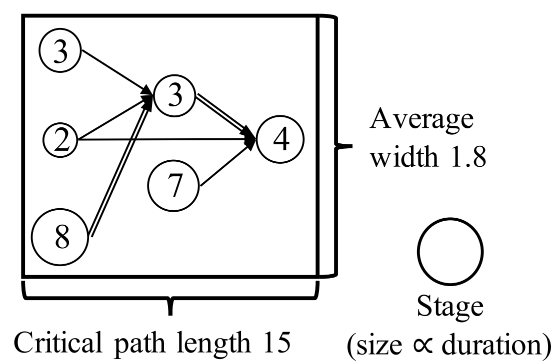
  - Scheduling events:
    - New job arrival
    - An executor becomes available

Action Space Design

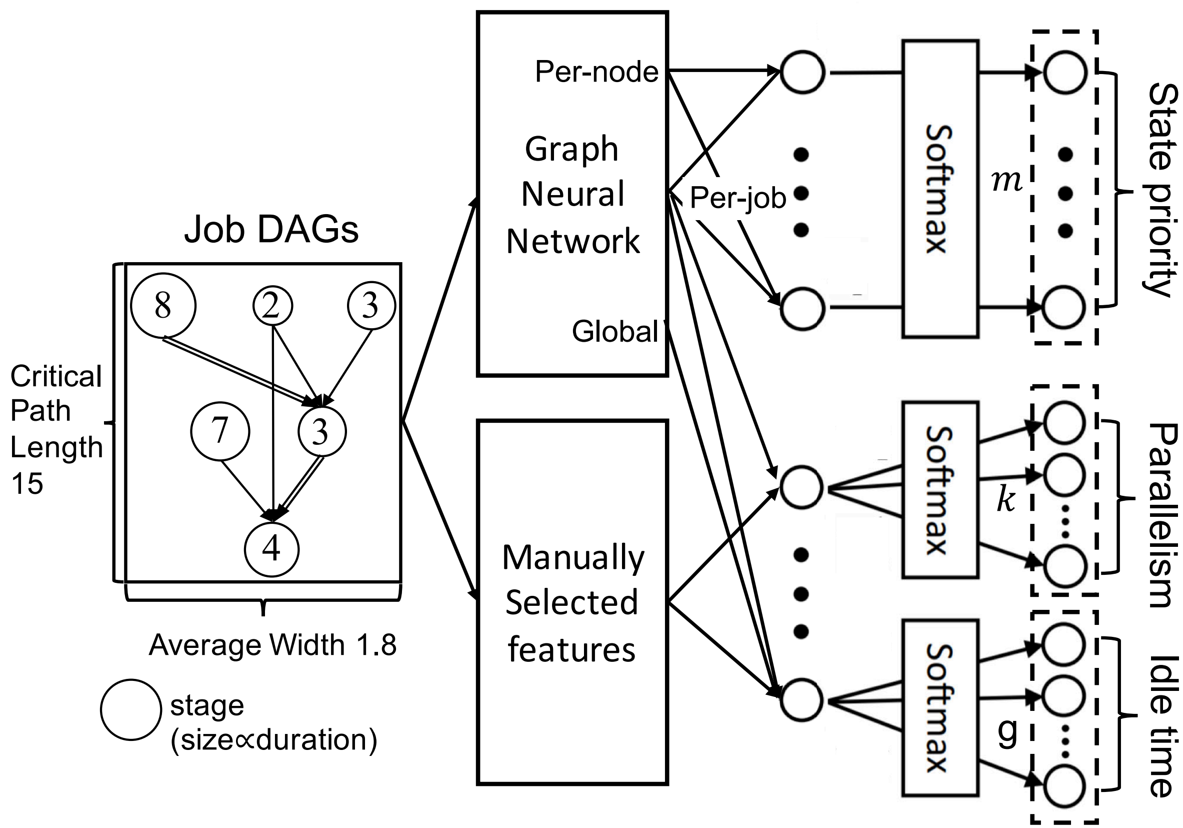
- Priority score
  - Determine the processing sequence
- Parallelism level
  - Determine the number of executors allocated to each stage
- Discretized idle time
  - Discretize idle time based on the stage size
    - Idle block: $1/G$ of the stage
    - Scheduler choose the number of idle blocks to insert

Policy Network Design

- Use graph neural network to capture DAG structure
- Use job abstraction to estimate job processing time
  - Job abstraction
    - Job length: critical path length
    - Job width: total job size / critical path length
  - Insights
    - Optimal idle time length is closely related to job length



Policy Network Overview



4. Experiment

- Experiment Setup
  - Synthetic dataset
    - Short/long jobs randomly arrives
  - Real-world dataset
    - TPC-H queries
  - Mixed dataset
    - Randomly sample from synthetic and real-world datasets with a given ratio
  - Training procedure
    - Gradually increase the workload
  - Training platform
    - Ubuntu 20.04
    - 64 GB RAM
    - GTX 1080

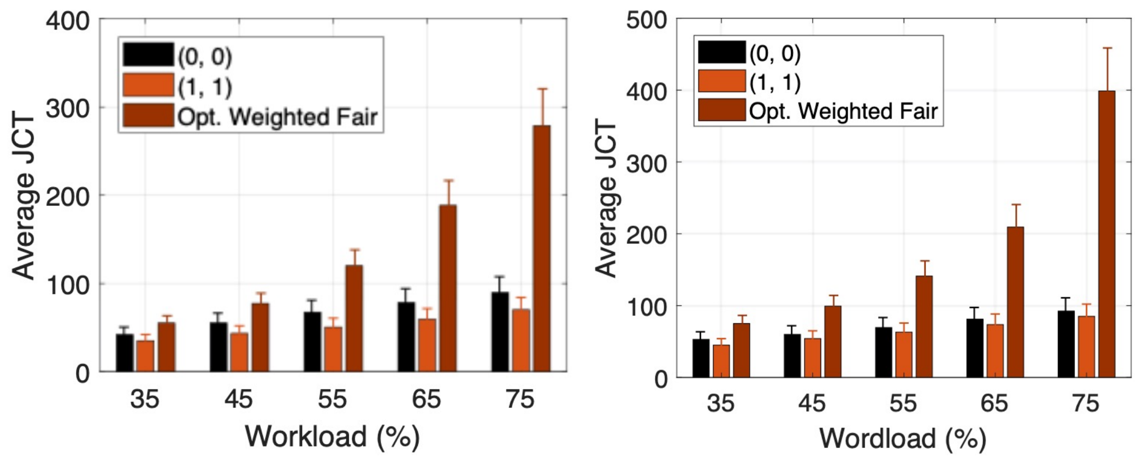
Experiment Results

- Compare RL agents
  - Label: (whether inserting idle slots, whether using job abstraction)

AVERAGE JCT EVALUATED IN DIFFERENT DATASETS

	(1,1)	(0,1)	(1,0)	(0,0)
Synthetic	46.3	52.7	53.5	55.0
Mixed	69.4	75.2	74.5	77.6

- Performance under different cluster workloads



(a) synthetic dataset (b) mixed dataset

5. Conclusion

- Investigated online DAG-style job scheduling
  - NP-hard problem
- Proposed to insert idle slots to reduce average JCT
  - Prevent short jobs waiting for long jobs
- Theoretically proved the benefits of idle slots
  - Optimal conditions
- Enhanced the RL-based scheduler
  - Job abstractions

Thank you!
Q & A

yubin.duan@temple.edu