IEEE MASS 2020
2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)

Reducing Makespans of DAG Scheduling through Interleaving Overlapping Resource Utilization

Yubin Duan[1], Ning Wang[2], and Jie Wu[1]

[1]Temple University

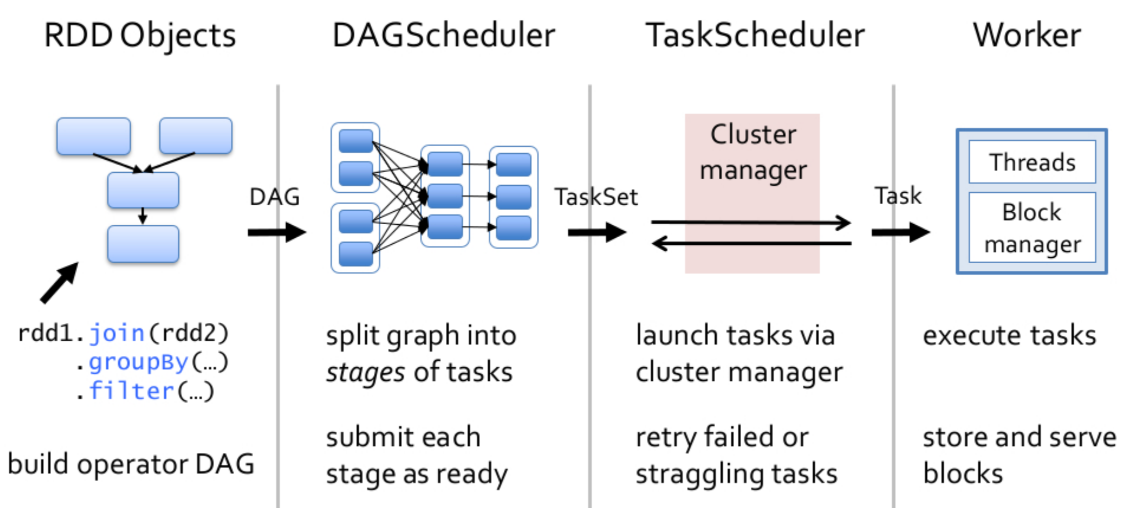
[2]Rowan University

Outline

- 1. Introduction
- 2. Problem Formulation
- 3. Scheduling for Perfectly Parallel Stages
- 4. Scheduling for General Stages
- 5. Experiment
- 6. Conclusion

1. Introduction

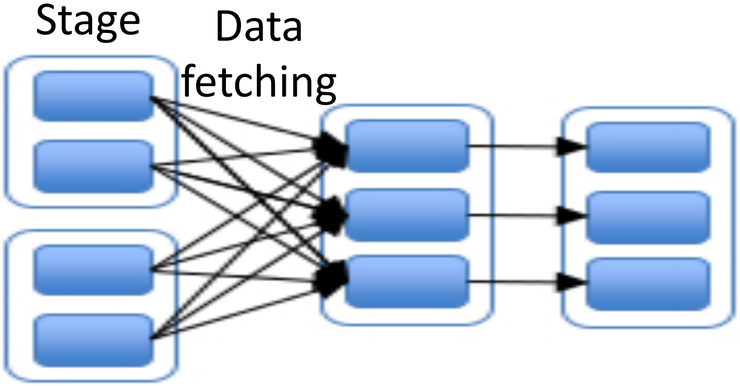
- Apache Spark
  - A general-purpose distributed computing engine for data processing
  - Large Data: stored as Resilient Distributed Dataset (RDD) objects
  - Data processing flow: RDD transformations with DAG structure



DAG Scheduler

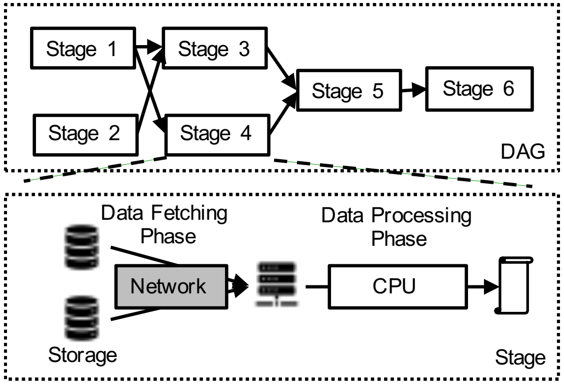
- Stages in Spark
  - Within each stage: computation tasks that can run in parallel
  - stage execution: data fetching phase and data processing phase
- DAG Scheduler in Spark
  - Parallelism level of each stage
  - Processing sequence of stages

Stage

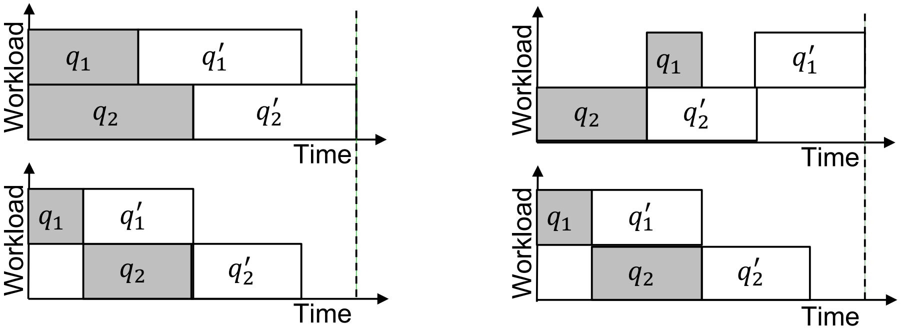


Motivation

- Observations
  - Data fetching and processing use different resources
  - They can run in pipeline

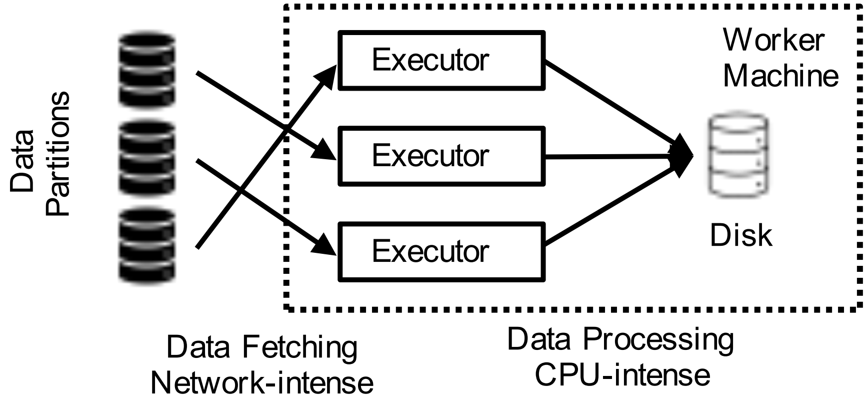


  - Contention of either resource would enlarge makespan



Objective

- Minimize job makespan by reducing resource contentions
  - Focus on optimizing the DAG scheduler of Spark
- Key assumptions:
  - Non-preemptive
  - Equally allocated resources



2. Problem Formulation

- DAG shop scheduling problem

$$\begin{aligned}\min\quad & \tau, \\ s.t.\quad & t_i' \leq t_j, \forall (s_i, s_j) \in E, \\ & \sum\nolimits_{s_i \in O(t)} p_i \leq P, \forall t > 0, \\ & \sum\nolimits_{s_i \in O(t)} b_i \leq B, \forall t > 0, \\ & t_i \geq 0, \forall s_i \in S.\end{aligned}$$

Precedence constraint

Computation resource constraint

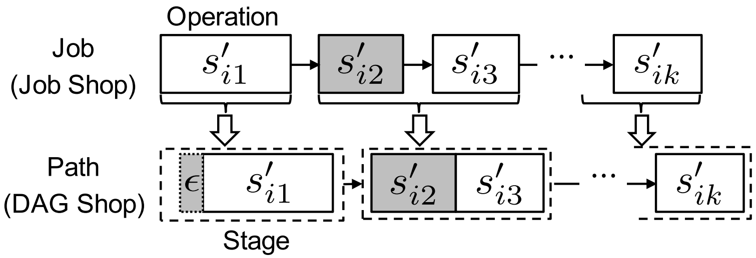
Bandwidth constraint

Schedule constraint

- Decision variables
  - Stage processing sequence or starting time t_i
  - Number of machines assigned to a stage: p_i
  - Bandwidth allocated to a stage: b_i

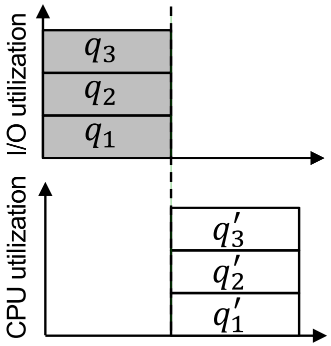
NP-hardness

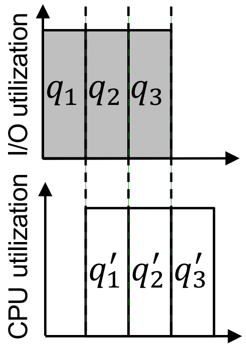
- NP-hard, even assuming the speedup is linear (ideal case)
  – Ideal speedup: 2 workers brings 2x speedup
- Proof:
  – Job shop problem (JSP) is NP-hard
  – Instances of JSP can be converted to our problem in polynomial time



3. Scheduling for Perfectly Parallel Stages

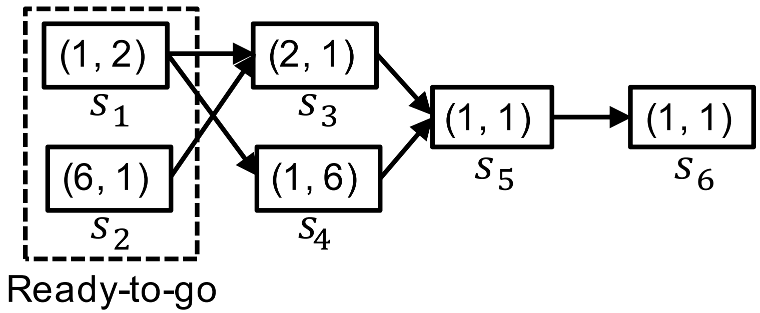
- Additional assumption
  - Speedup of any stage s_i in DAG is linear to p_i
- Contention-free scheduler
  - Contention brings no benefits
  - Should assign all resources to a stage
  - Problem reduced: only need to determine a processing sequence





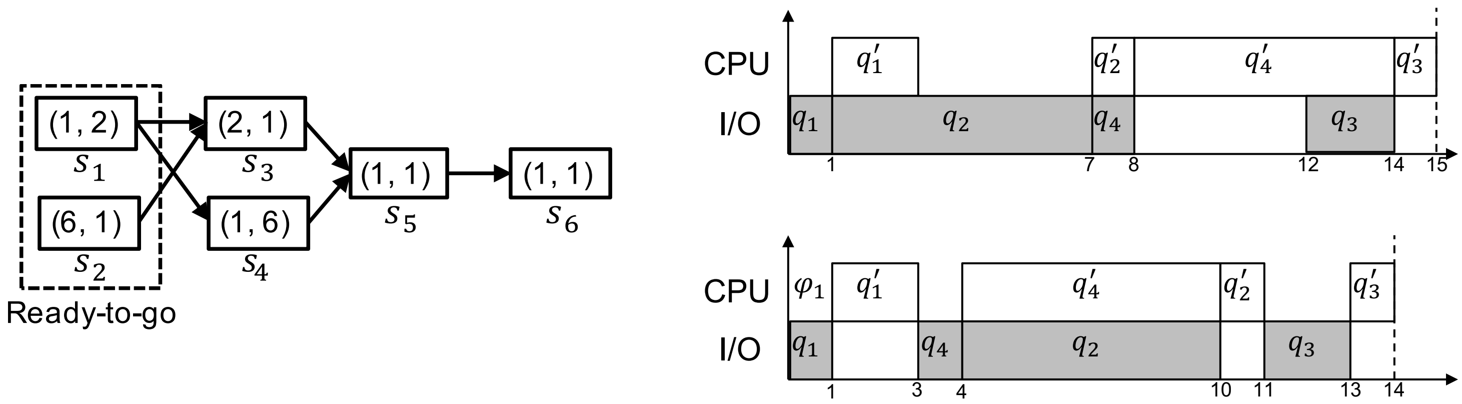
Contention-free Scheduler

- Apply Johnson's rule on ready-to-go stages
  - Divide stages into comm.-heavy and comp.-heavy groups
  - Sort comp.-heavy group by comm. time in ascending order
  - Sort comm.-heavy group by comp. time in descending order
- Example
  - (a, b): tuple represents the length of comm. and comp., respectively



Properties

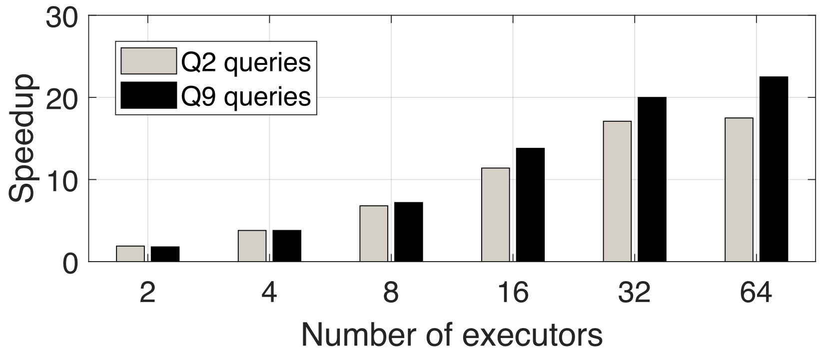
- Our scheduler brings additional precedence constraints



- Our contention-free scheduler is 3/2-approximate if comp. and comm. of each stage have unit lengths.
  - The lost is bounded: our scheduler will not leave both resources idle

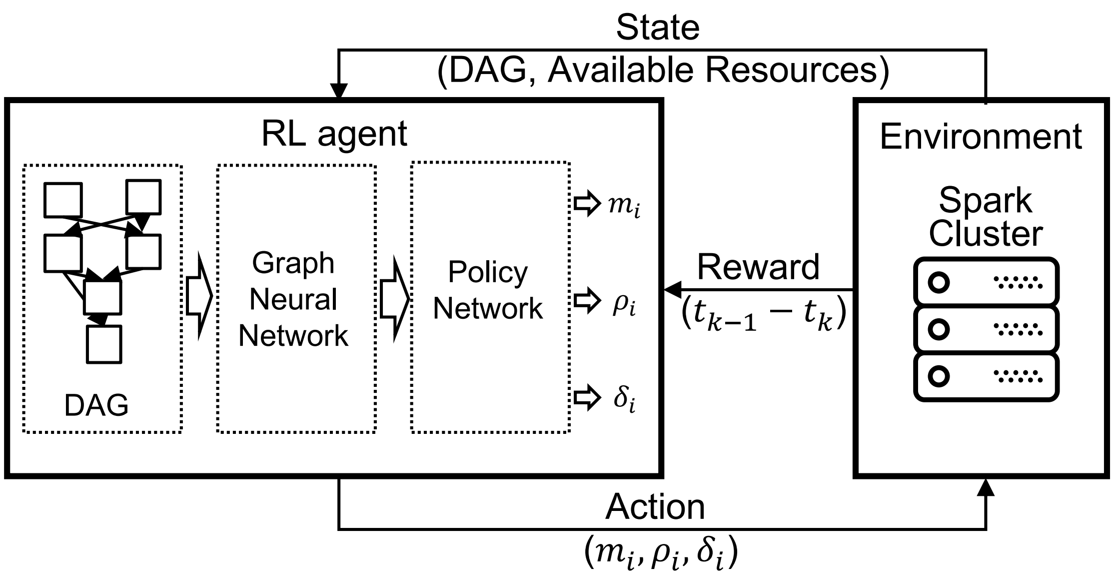
4. Scheduling for General Stages

- General cases: speedup of s_i is not linear to p_i



- Should have limitations on number of workers
  - Assigning too many workers to a stage is a waste
  - Set parallelism level limitations for comp. resources p_i

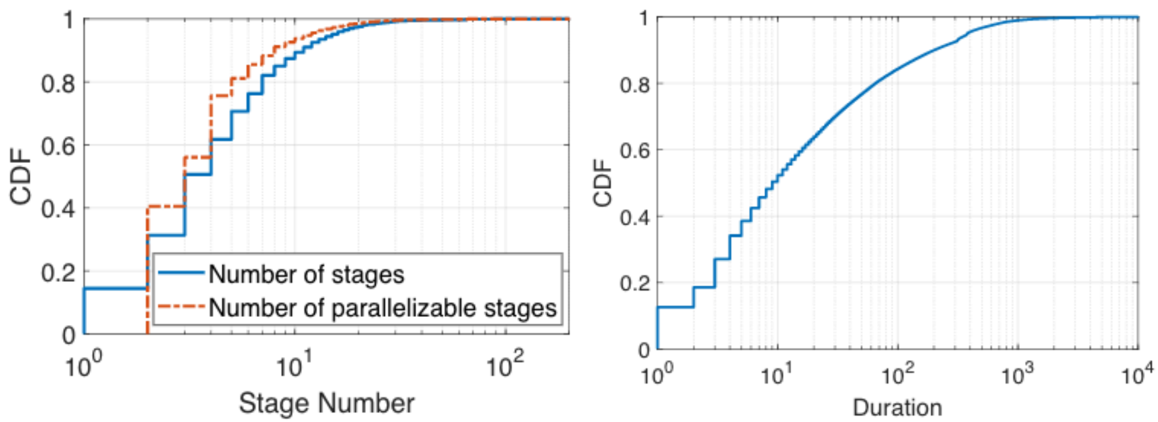
Reinforcement Learning based Scheduler



- RL framework:
  - State: use graph neural network to encode a DAG
  - Action: parallelism limitations, priority level and delay of each stage
  - Reward: expected time consumption for executing remaining stages

5. Experiment

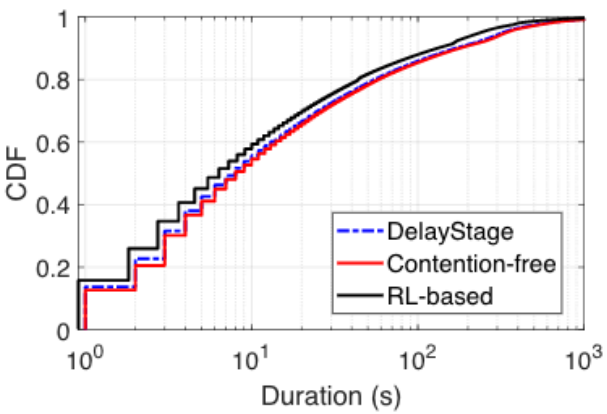
- Experiment setting:
  - Alibaba trace data v2018: contains 2,775,025 jobs
  - Use m4.xlarge instance of AWS EC2 to build spark clusters

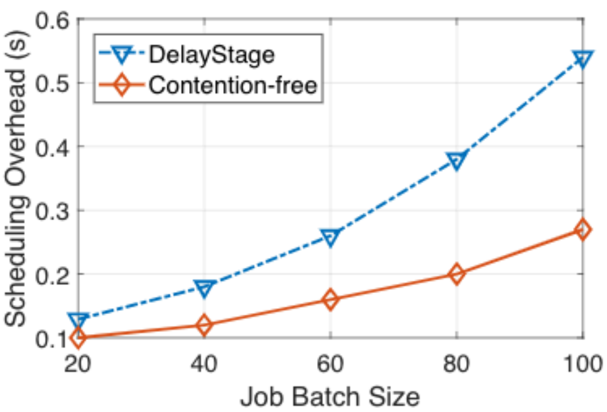


(a) The stage number distribution (b) The stage duration distribution

Experiment Result

- Makespan comparison





- Utilization improvement

THE AVERAGE RESOURCE UTILIZATION

	Default	DelayStage	RL-based
Average CPU utilization	37.9%	46.1%	50.4%
Average Network utilization	43.5%	54.5%	56.4%

6. Conclusion

- Resource contention in DAG scheduling increase makespan
- Contention-free scheduler for ideal stages
  - DAG shop scheduling problem
  - NP-hard
  - A 3/2 approximation algorithm based on Johnson's rule
- RL-based scheduler for general stages
  - Apply graph neural network to encode stages
  - Adaptively adjust the contention level
- Resource utilization improve by about 30%

Thanks

For any questions, don’t hesitate to email me at
yubin.duan@temple.edu