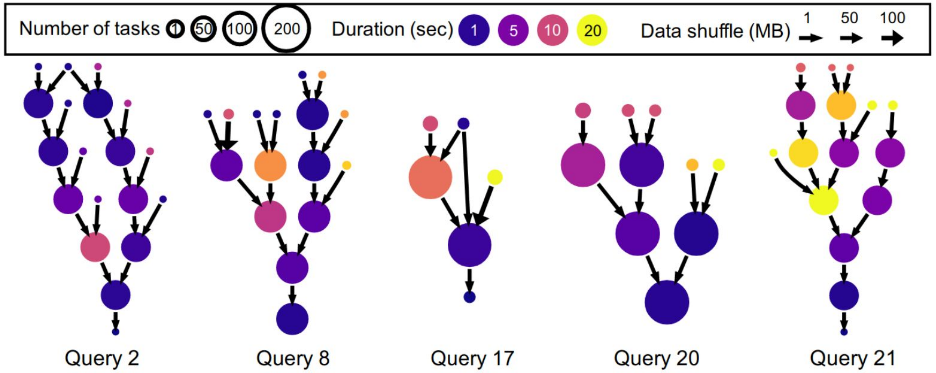
Learning Scheduling Algorithms for Data Processing Clusters

Authors: Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, Mohammad Alizadeh

Presenter: Aruth Kandage

Background

- Data processing in distributed compute clusters using systems such as Hive, Spark-SQL and DryadLINQ
- DAG-structured compute jobs
- Graph nodes are called *stages* each with a number of *tasks*
- Graph edges indicate data dependencies
- Cluster scheduler assigns tasks to *executors* in the cluster



Motivation

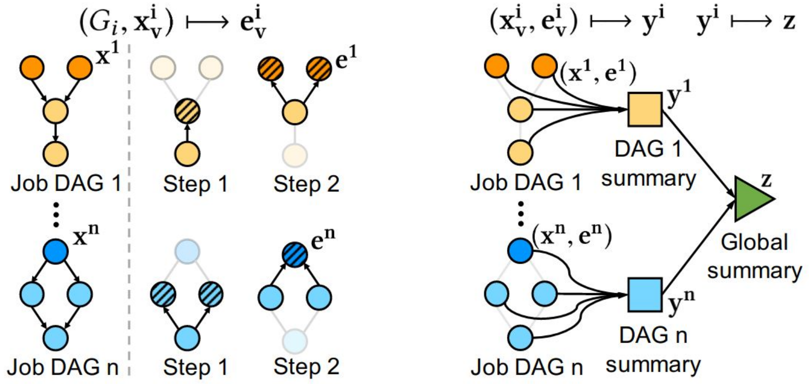
- Efficient scheduling can save millions of dollars at scale!
- Challenge: Large amount of input information
- Challenge: Large space of possible schedules
- Challenge: Online arrival of jobs to the cluster
- Common to run same job multiple times in commercial clusters

Decima Cluster Scheduler

- RL agent learns scheduling policy
- Graph neural network learns to compute embeddings from input job graph
- Policy network makes scheduling decisions
- Offline training in simulated environment

Graph Neural Network

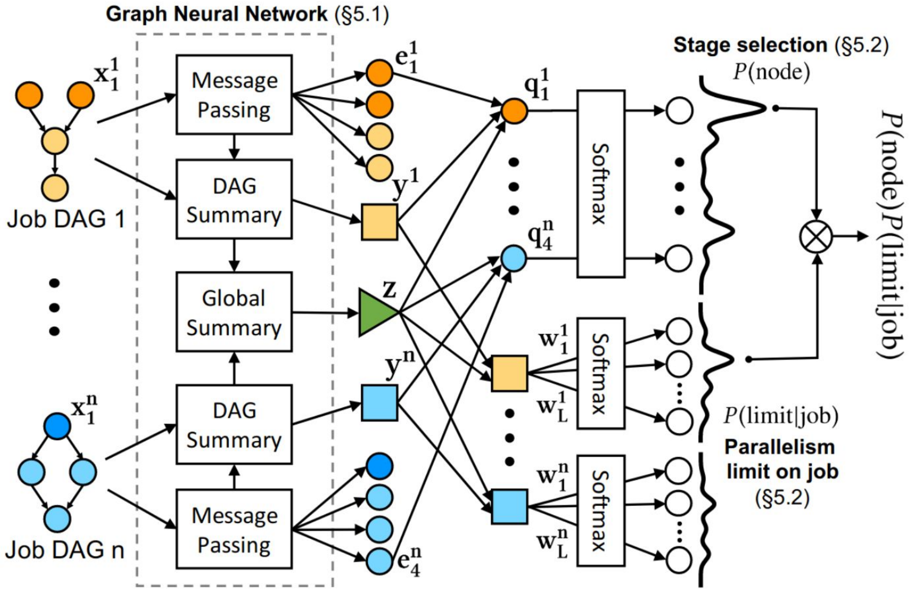
- Job represented as a DAG
- Stages are nodes in the graph each with a vector x_v^i of stage attributes
- GNN computes an embedding e_v^i for each stage
- GNN computes summary embedding y^i for each job DAG
- GNN computes summary embedding z for all jobs



$$\mathbf{e}_v^i = g\left[\sum_{u \in \xi(v)} f(\mathbf{e}_u^i)\right] + \mathbf{x}_v^i,$$

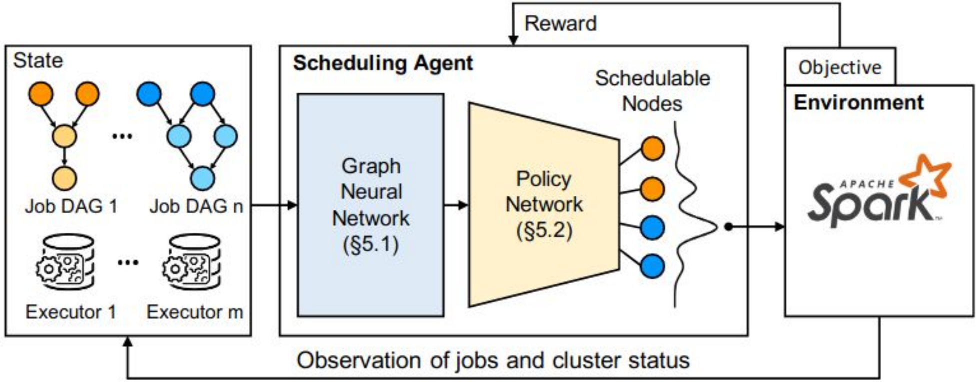
Policy Network

- Policy network scores each embedding
- One score q_v^i is used to select node/stage to schedule
- Another score w_l^i is used to select parallelism limit for job



Scheduling

- Overall running the agent once will produce a tuple (v, l) indicating the stage to schedule and its job parallelism limit
- Agent is asked to make a scheduling decision whenever there are free executors in the cluster



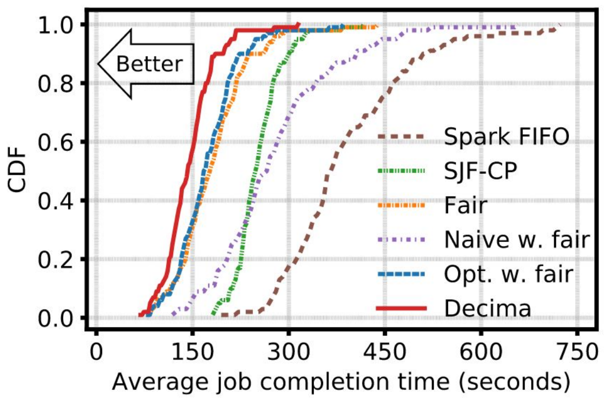
Offline Training

- RL agent trained in a simulator of Spark cluster using workload traces
- Reward function applies a penalty based on number of active jobs
- Train using policy gradient (REINFORCE)
- Train multiple episodes using same job arrival sequence
- Reduce policy gradient variance by subtracting baseline value b_k
- Increase the episode length as training progresses

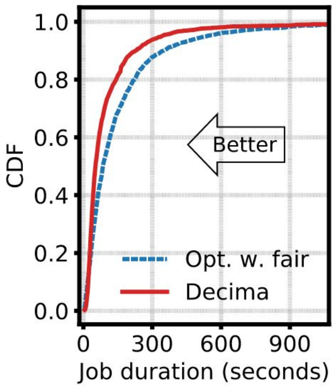
Algorithm 1 Policy gradient method used to train Decima.

for each iteration **do**
  $\Delta\theta \leftarrow 0$
  Sample episode length $\tau \sim \text{exponential}(\tau_{\text{mean}})$
  Sample a job arrival sequence
  Run episodes $i = 1, \ldots, N$:
    $\{s_1^i, a_1^i, r_1^i, \ldots, s_\tau^i, a_\tau^i, r_\tau^i\} \sim \pi_\theta$
  Compute total reward: $R_k^i = \sum_{k'=k}^{\tau} r_{k'}^i$
  for $k = 1$ to τ **do**
    compute baseline: $b_k = \frac{1}{N} \sum_{i=1}^{N} R_k^i$
    for $i = 1$ to N **do**
      $\Delta\theta \leftarrow \Delta\theta + \nabla_\theta \log \pi_\theta(s_k^i, a_k^i)(R_k^i - b_k)$
    end for
  end for
  $\tau_{\text{mean}} \leftarrow \tau_{\text{mean}} + \epsilon$
  $\theta \leftarrow \theta + \alpha \Delta\theta$
end for

Evaluation (TPC-H)



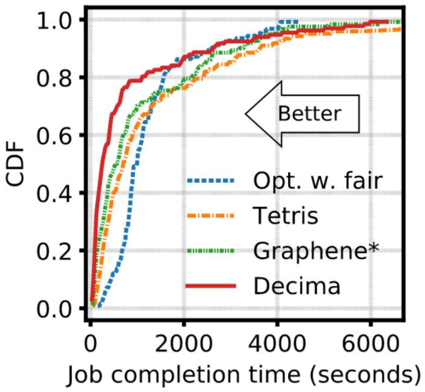
TPC-H Batch Job Arrival



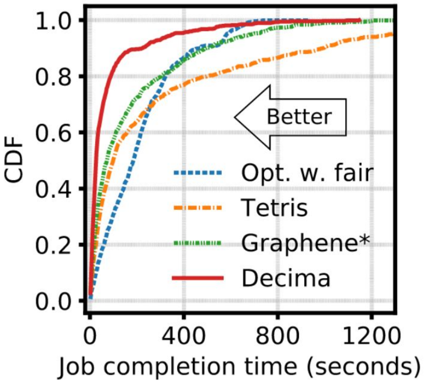
TPC-H Continuous Job Arrival

- Batch jobs - Decima has 21% lower average JCT (Job Completion Time)
- Continuous jobs - Decima has 29% lower average JCT (Job Completion Time)
- Up 2x lower average JCT vs. nearest heuristic when cluster under heavy load
- Decima completes small jobs faster with correct assignment of executors to task

Evaluation (Multi-Resource Scheduling)



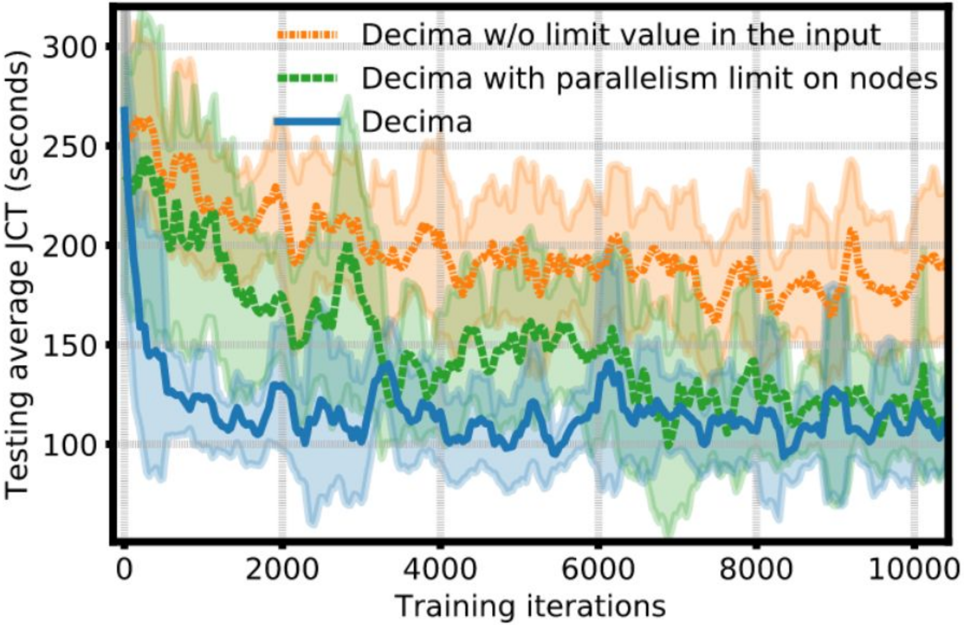
Multi-Resource Trace Replay



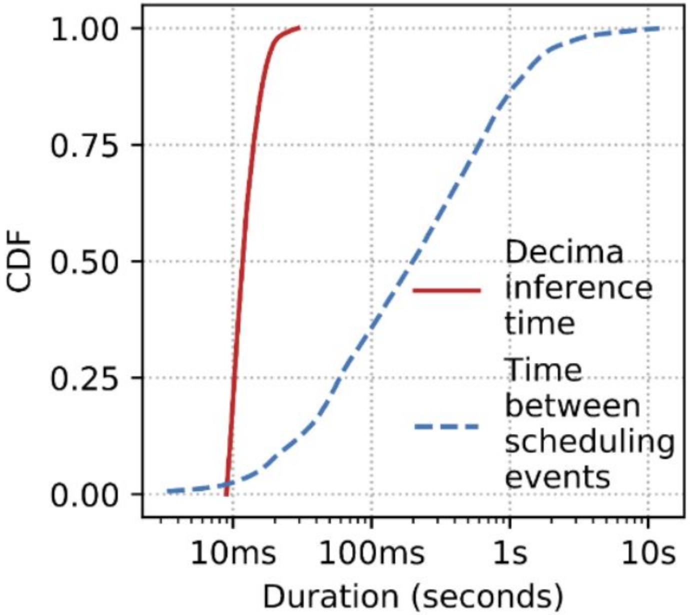
Multi-Resource TPC-H

- Multi-resource setting: jobs have both CPU and memory demands
- Multiple classes of executors with different CPU and memory resources
- Decima has 32% lower average JCT vs. nearest algorithm (Graphene)
- Decima learns to trade-off cluster resource fragmentation for lower JCT

Evaluation



Decima w/o limit value in the input
Decima with parallelism limit on nodes
Decima
Testing average JCT (seconds)
300
250
200
150
100
0
2000
4000
6000
8000
10000
Training iterations



1.00
0.75
0.50
0.25
0.00
CDF
Decima inference time
Time between scheduling events
10ms
100ms
1s
10s
Duration (seconds)

Evaluation

Setup (IAT: interarrival time)	Average JCT [sec]
Opt. weighted fair (best heuristic)	91.2±23.5
Decima, trained on test workload (IAT: 45 sec)	65.4±28.7
Decima, trained on anti-skewed workload (IAT: 75 sec)	104.8±37.6
Decima, trained on mixed workloads	82.3±31.2
Decima, trained on mixed workloads with interarrival time hints	76.6±33.4

Future Work

- Robustness
- Online Training
- Different Learning Objectives (e.g. SLA for job completion time)
- Pre-emption of Jobs
- Other applications such as database query optimization

Summary

- Decima demonstrates RL agent which outperforms heuristics in distributed job scheduling
- Novel use of graph neural network and policy network for scheduling decisions
- Offline training in simulated environment but can generalize well to real workloads

Thank you!