




# Learning Scheduling Algorithms for Data Processing Clusters

Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrisnan, Zili Meng\*, Mohammad Alizadeh



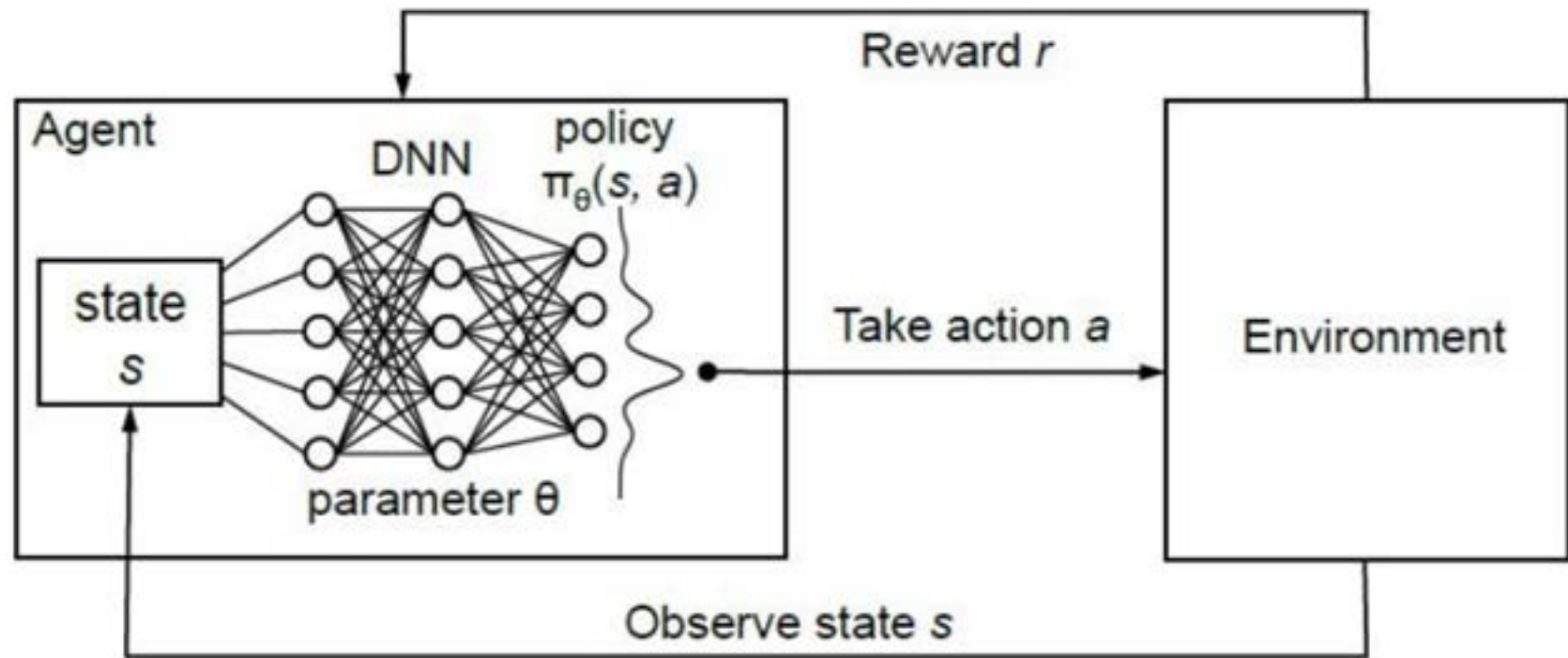
# Resource Management with Deep Reinforcement Learning

Hongzi Mao?, Mohammad Alizadeh?, Ishai Menachey, Srikanth Kandulay

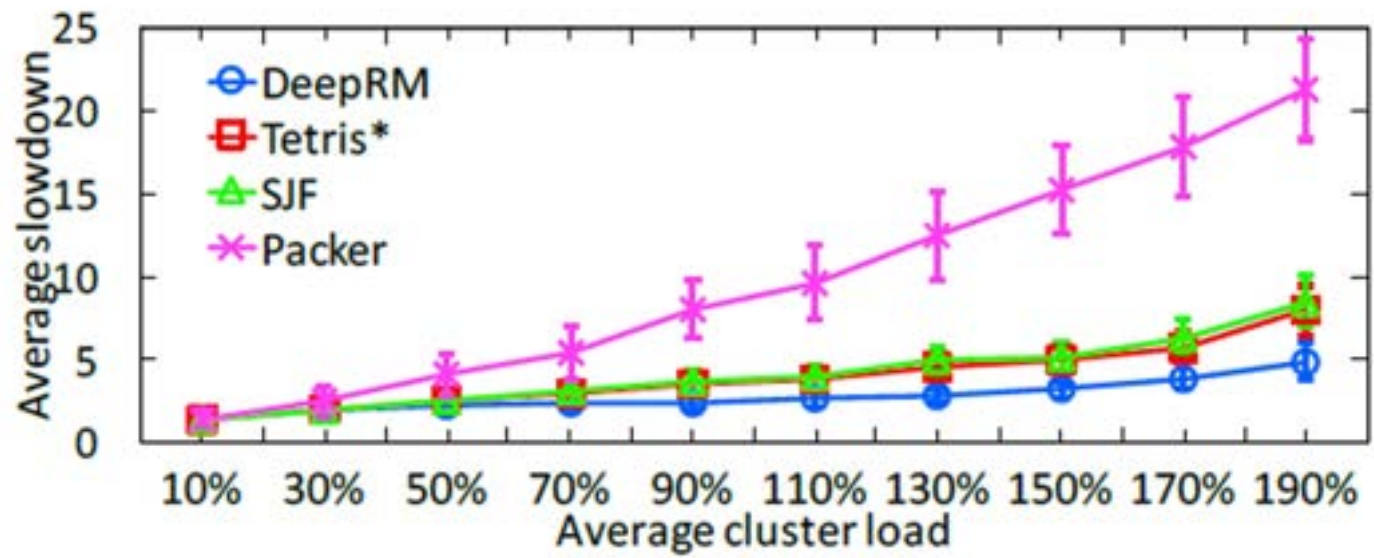


## Key Points

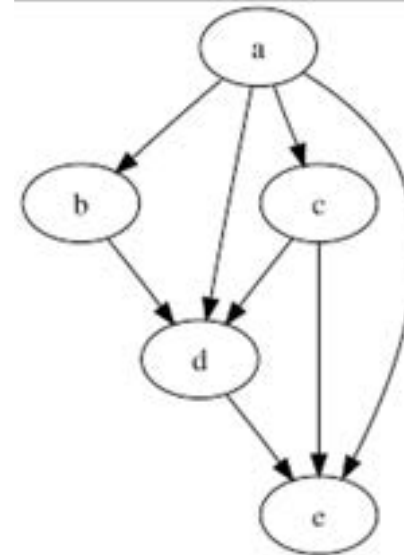
- Decima is a novel system for scheduling data processing operations on distributed computing **clusters**.
- Decima uses machine learning approaches (**reinforcement learning** and **neural networks**) for workload-specific scheduling algorithms.
- Decima uses **scalable RL models** and RL training approaches for handling continuous stochastic job arrivals.
- Decima can result in up to a 2x improvement in task completion time during peak cluster demand times.



# DRM

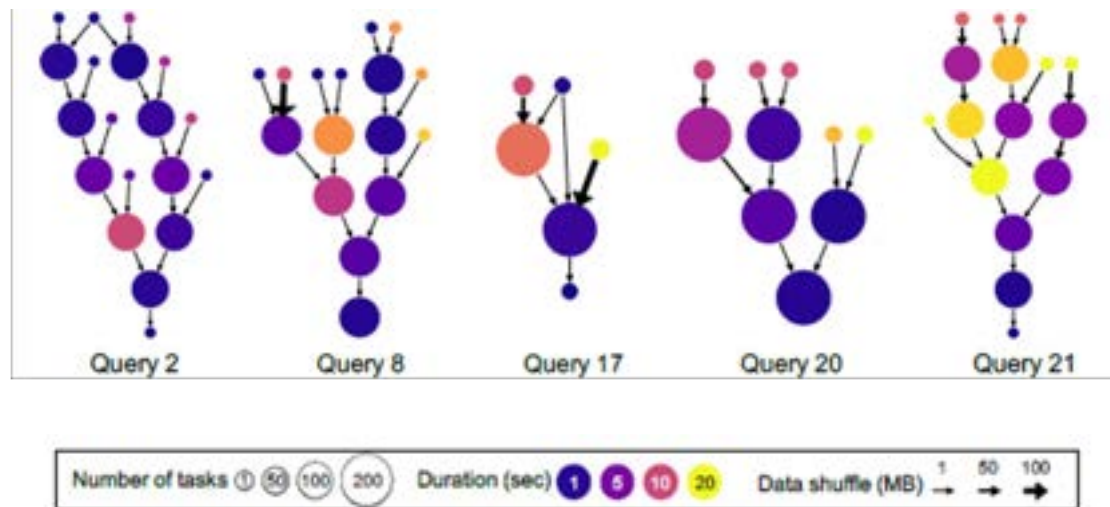


- Scheduling DAGs (directed acyclic graphs) is a complex algorithmic problem with intractable optimal solutions



Scheduling is a fundamental task in computer systems.

- HIVE
- PIG
- SPARK-SQL
- DryadLINQ





## **Must consider many factors for optimal performance:**

- Job dependency structure
- Modeling complexity
- Placement constraints
- Data locality





## Motivation

### **Why does it make more sense to use or build a new technology**

- (i) Dependency-aware scheduling.
- (ii) Automatically choosing the right number of parallel tasks.(Split)



## How does it build upon, utilize, or extend other work?

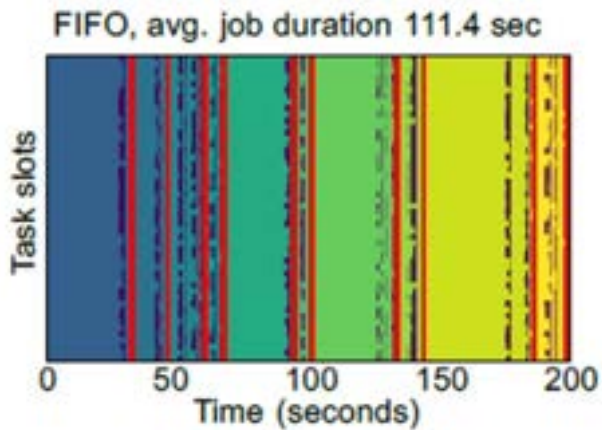
- Decima integration with Spark:
- DAG scheduler contacts Decima on startup and during scheduling events to receive next stage and parallelism limit
- Spark master contacts Decima to determine executor launch for new job and aids Decima in taking away executors once stage is complete.



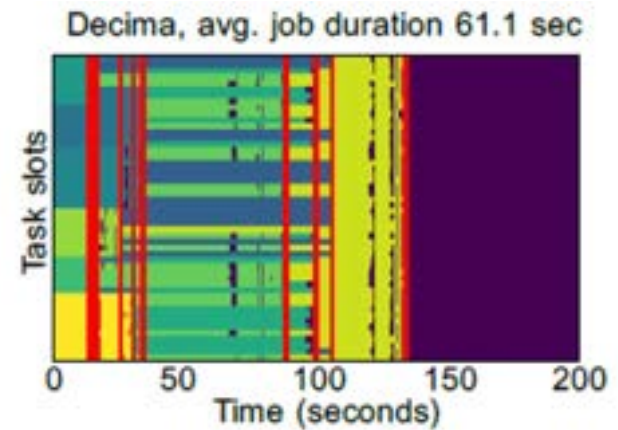
# Types of scheduling

1. FIFO (First in First out)
2. SJF (Shortest Job First )
3. Fair Scheduling

# Comparisons



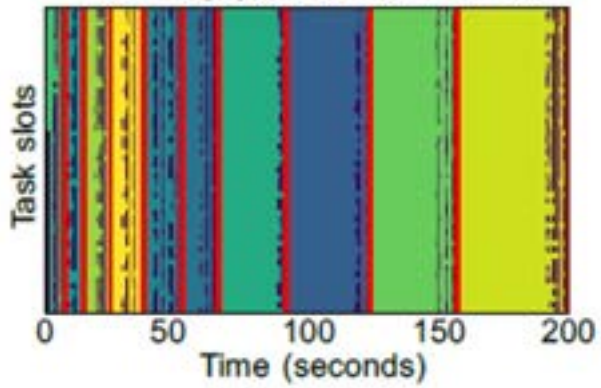
(a) FIFO scheduling.



(d) Decima.

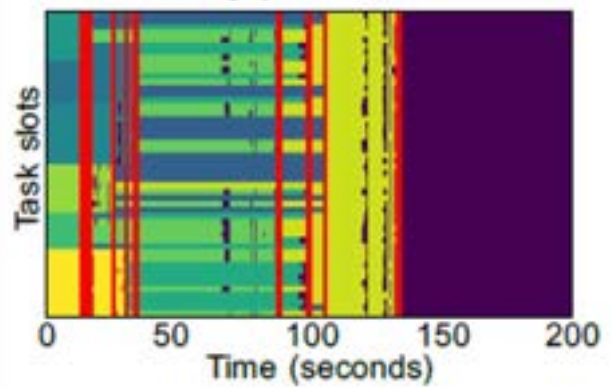


SJF, avg. job duration 81.7 sec

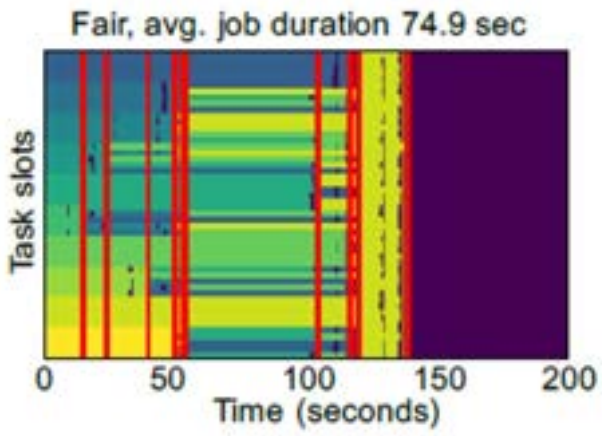


(b) SJF scheduling.

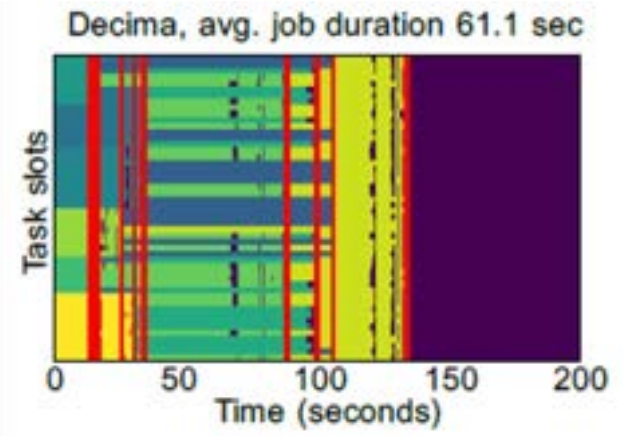
Decima, avg. job duration 61.1 sec



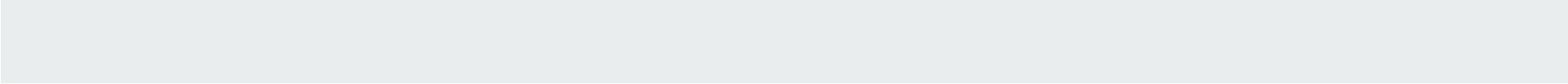

(d) Decima.



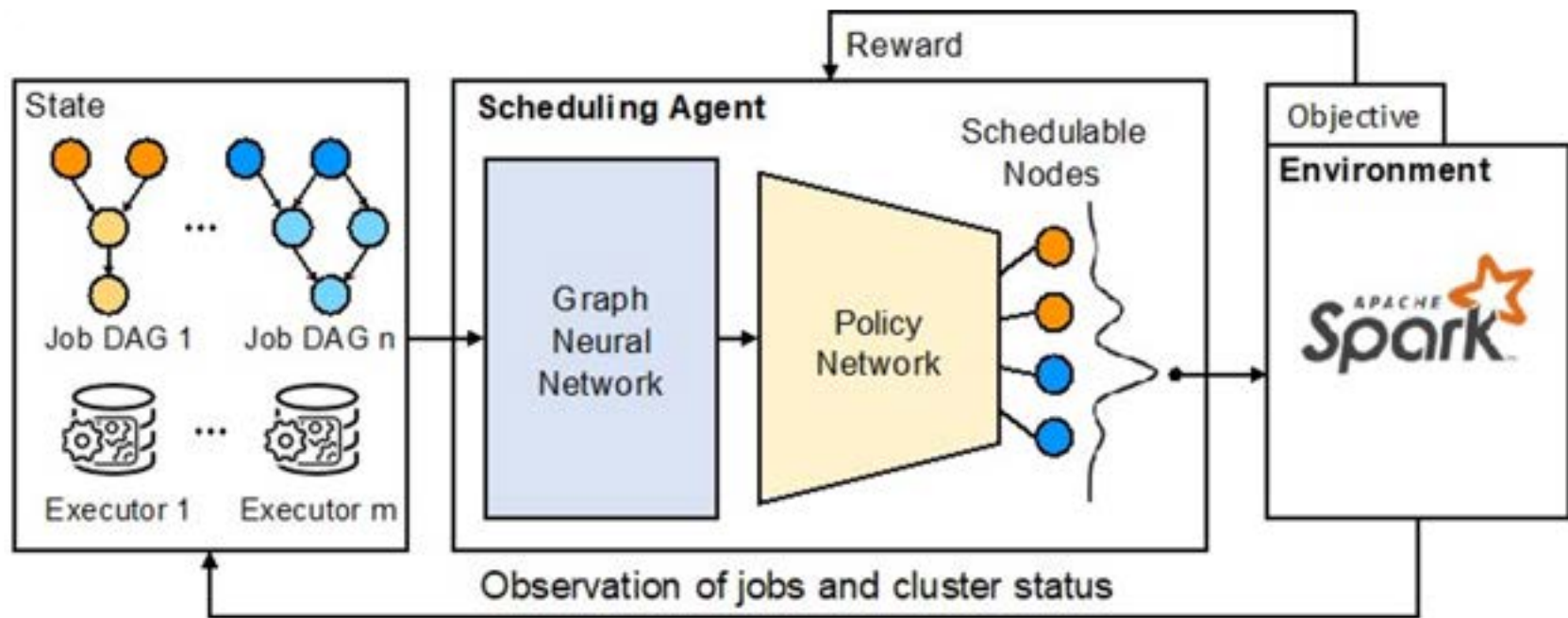
(c) Fair scheduling.



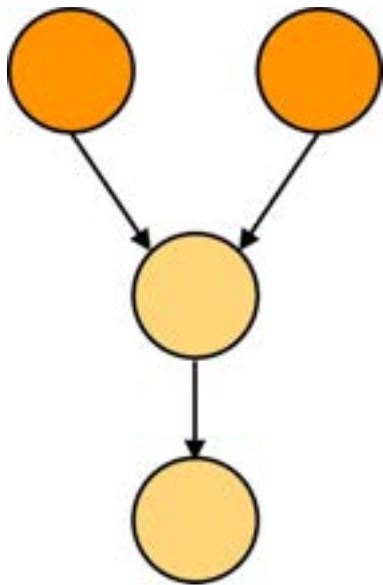
(d) Decima.

- 
- 
- Uses monitoring information and past workload logs to automatically learn scheduling policies
  - Learns to allocate resources optimally for each job to improve overall performance
  - Learns job-specific parallelism levels to avoid wasting resources on jobs with low inherent parallelism

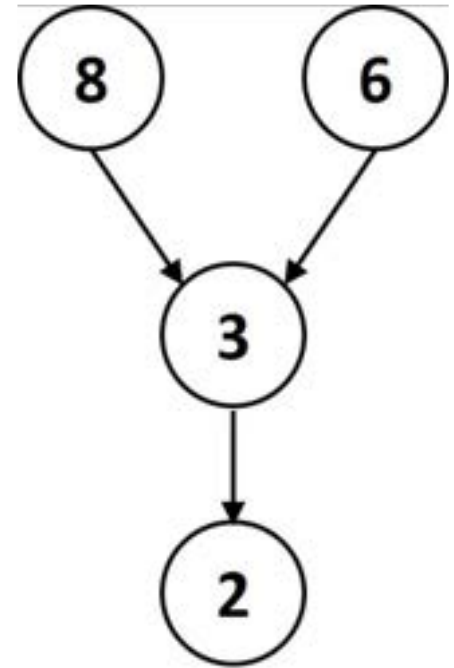
# Design







$f$






# Challenges

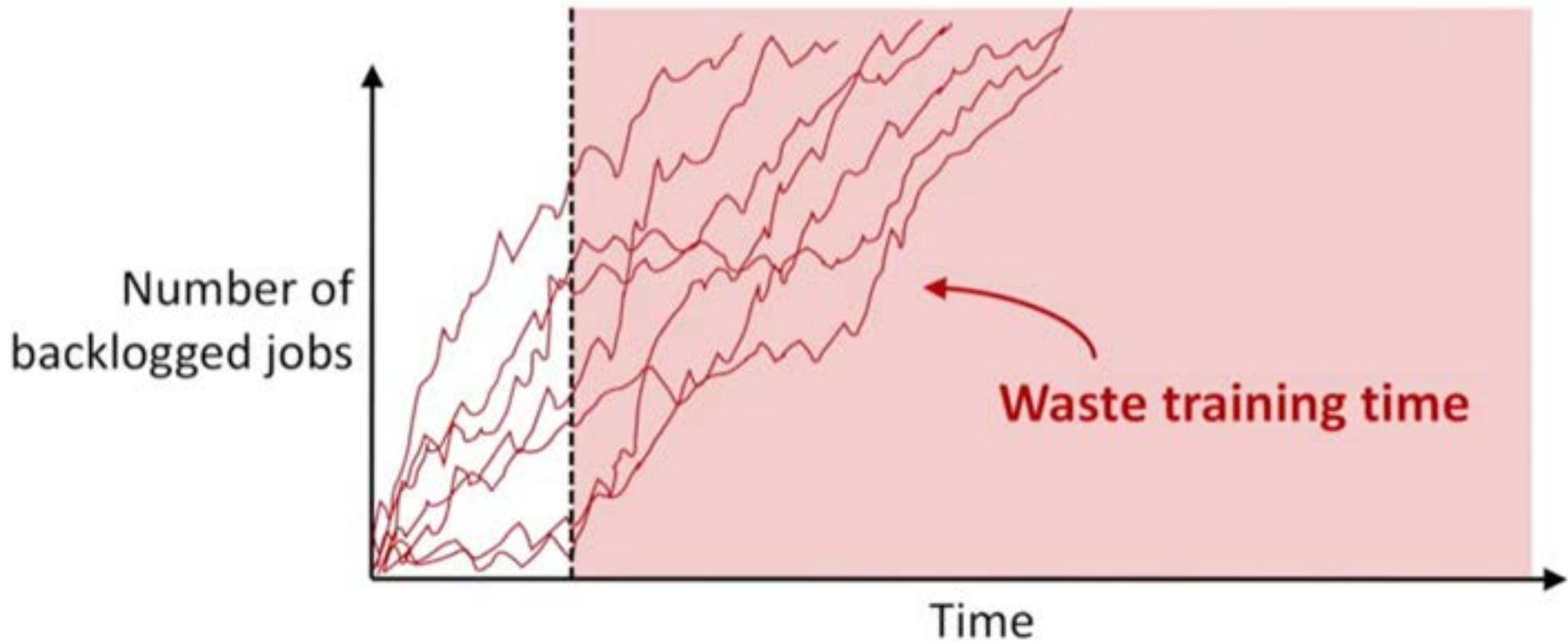
## 1) Training with continuous job arrivals

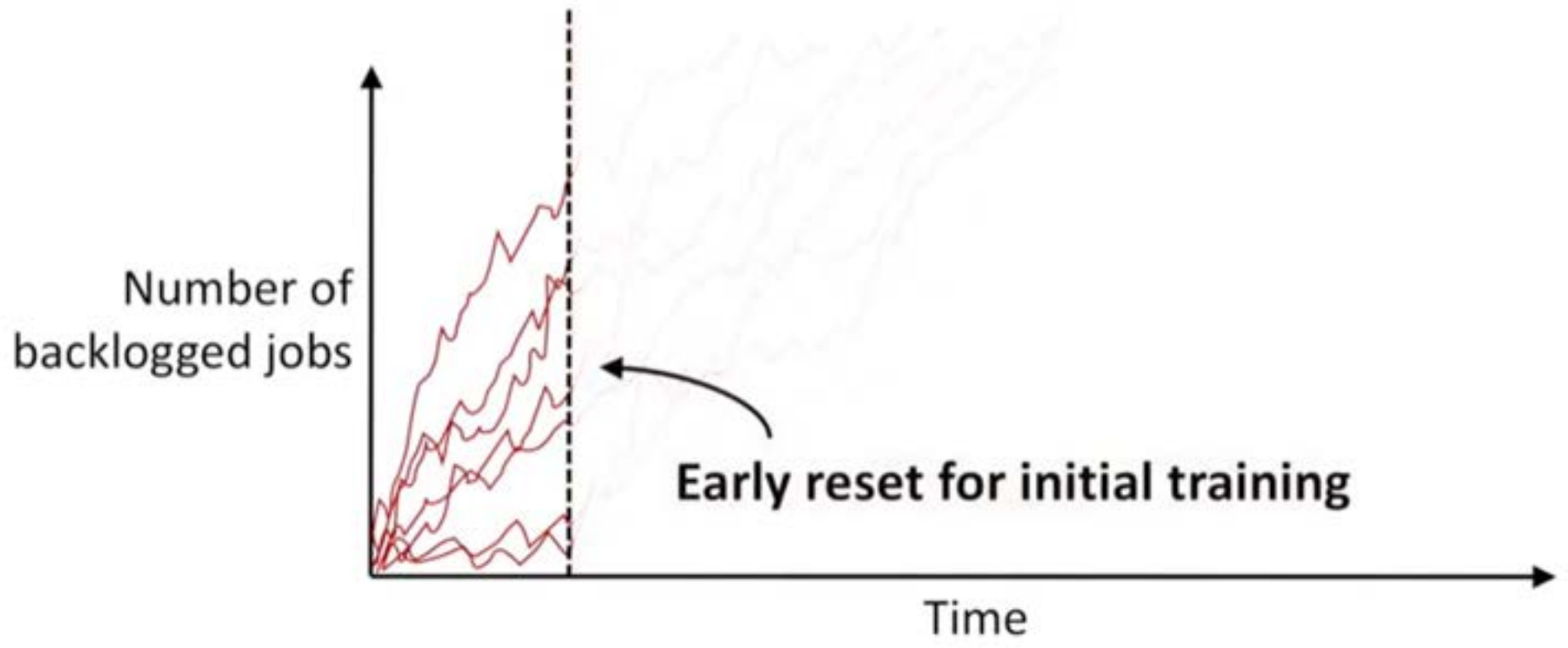
- Training with continuous stream of job arrivals is non-trivial
- Initial policy is poor, leading to a build-up of job queue in early training episodes
- To avoid waste, early episodes are terminated early to allow the agent to reset and quickly try again
- As scheduling policy improves, episode length increases, making the problem more challenging.

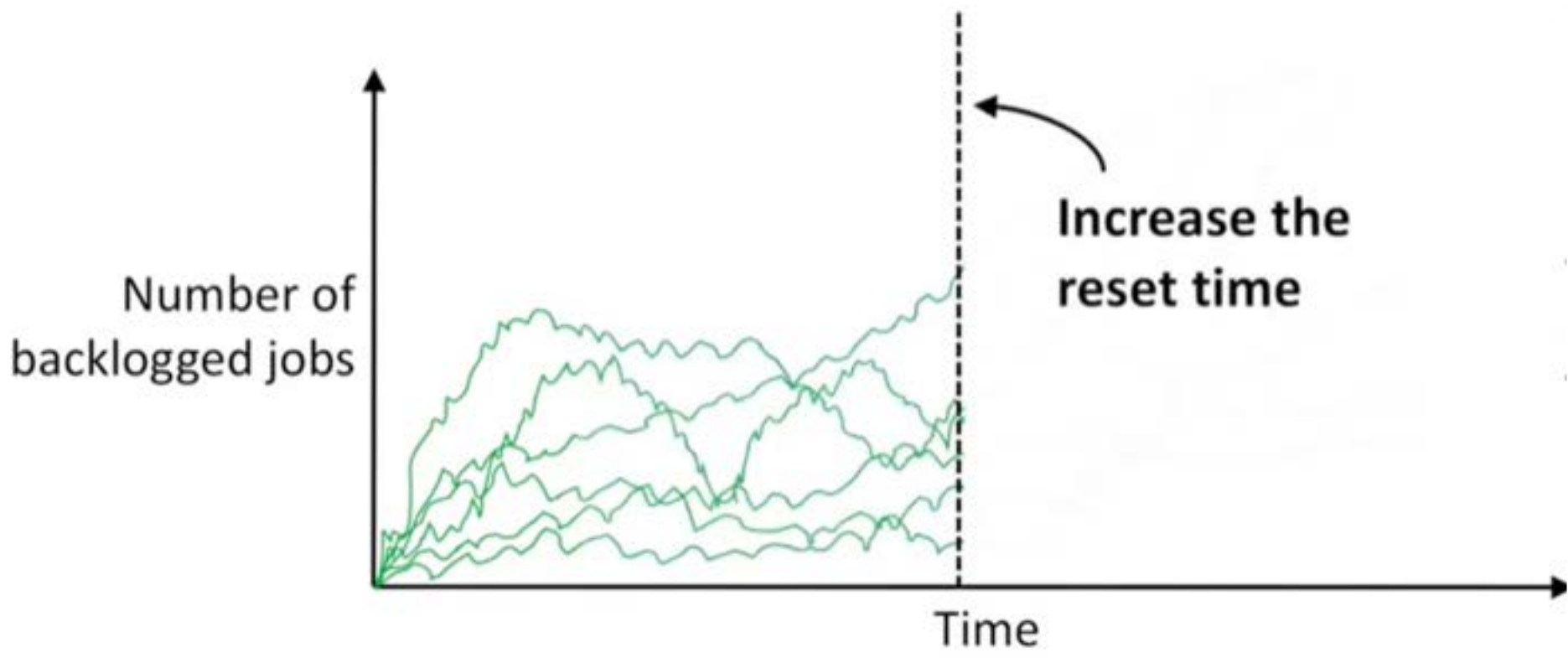


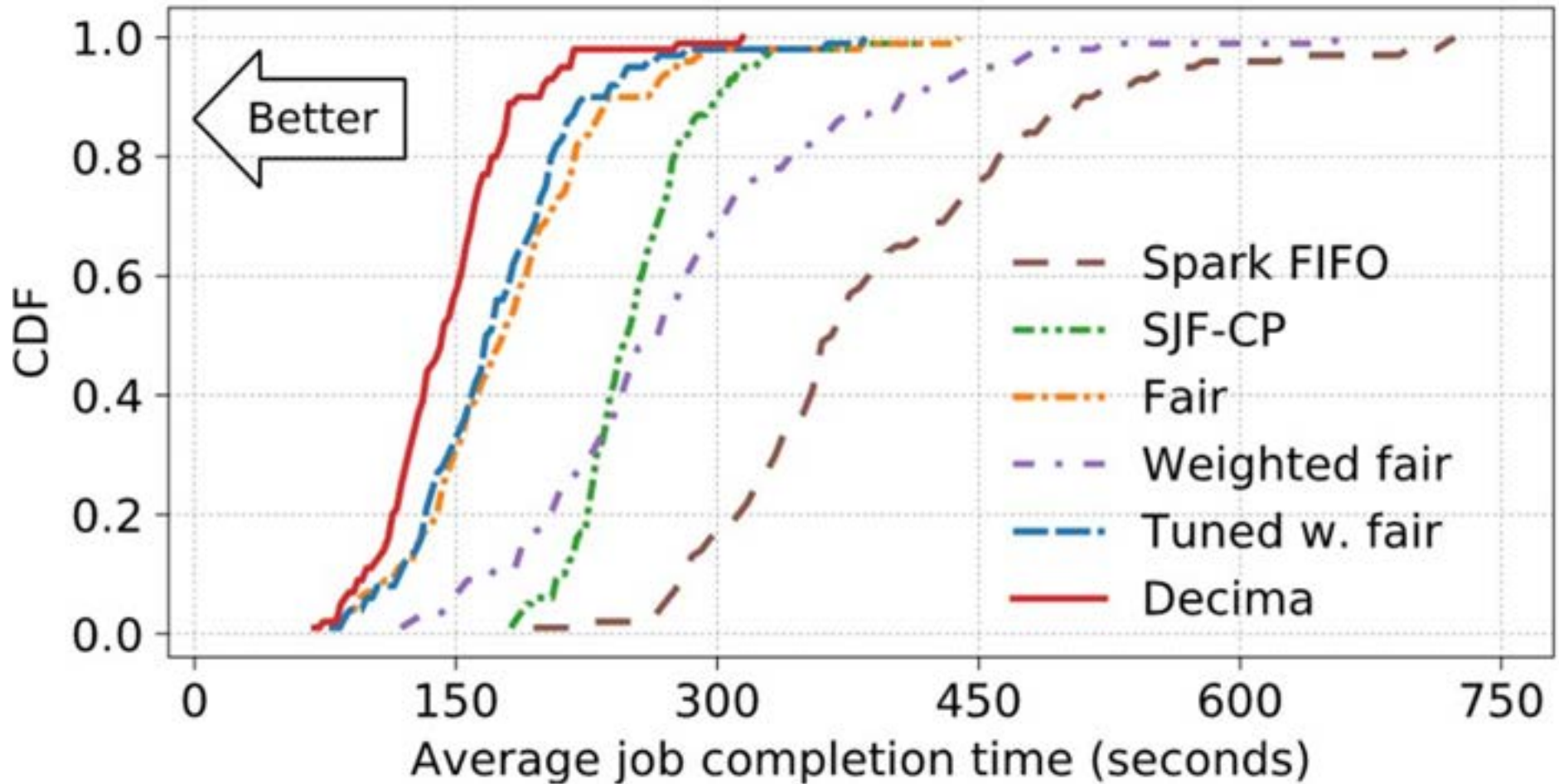
## 2) Variance caused by stochastic job arrivals

- Policy must generalize well in a streaming setting, requiring many different job arrival patterns in training episodes
- Different job arrival patterns lead to vastly different rewards, making it difficult to assess the goodness of different actions
- To address this problem, variance reduction technique for "input-driven" environments is used
- Technique involves fixing the same job arrival sequence in multiple training episodes and computing separate baselines for each sequence

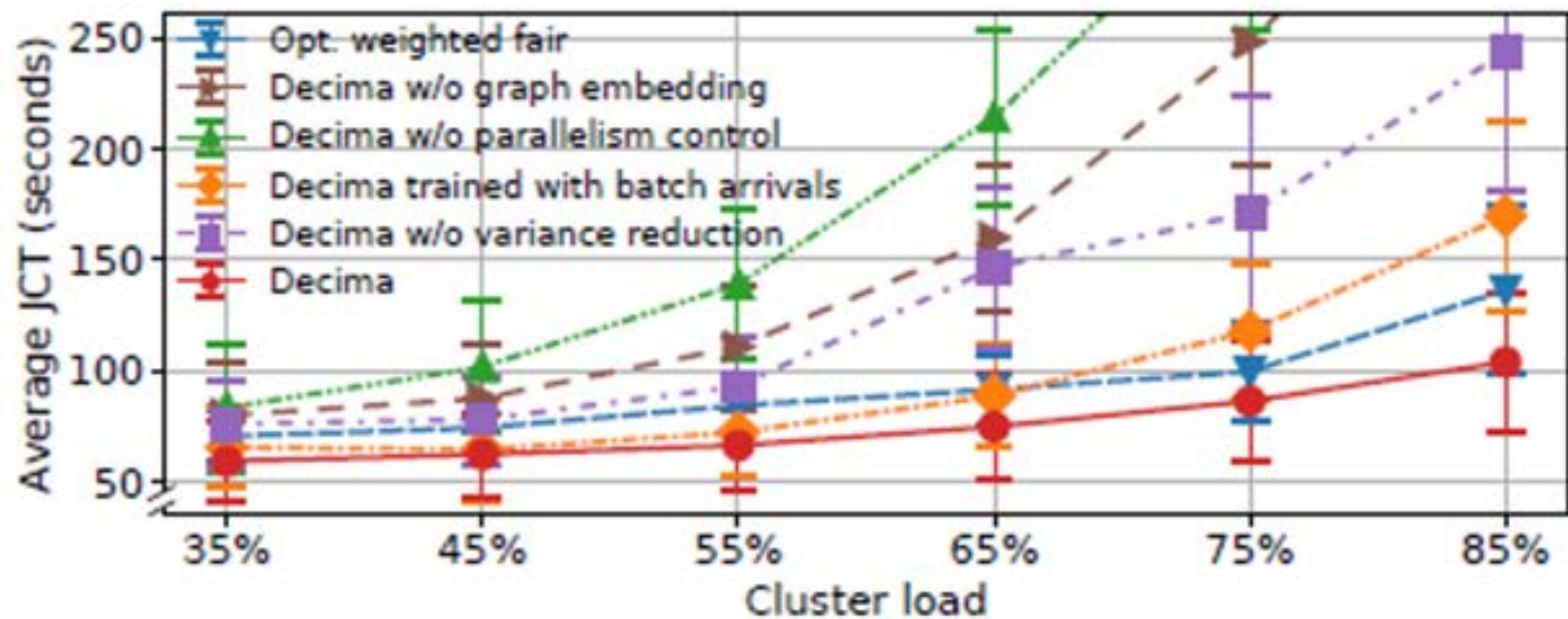






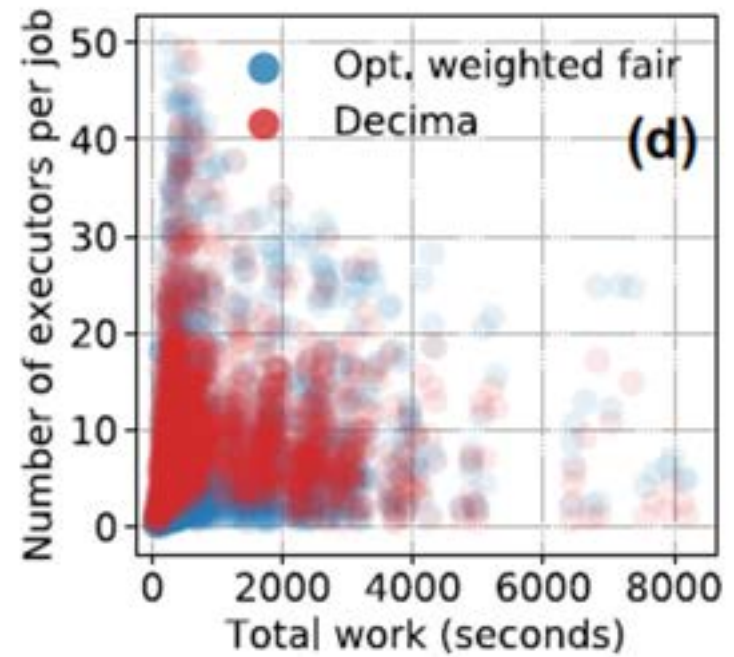
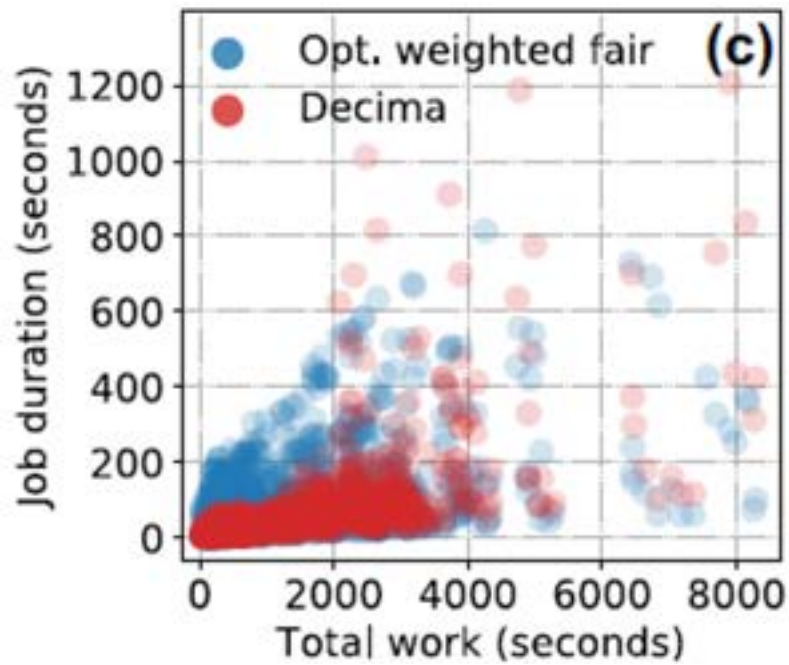


## DECIMA ON CONTINUOUS JOB ARRIVALS





# Understanding DECIMA





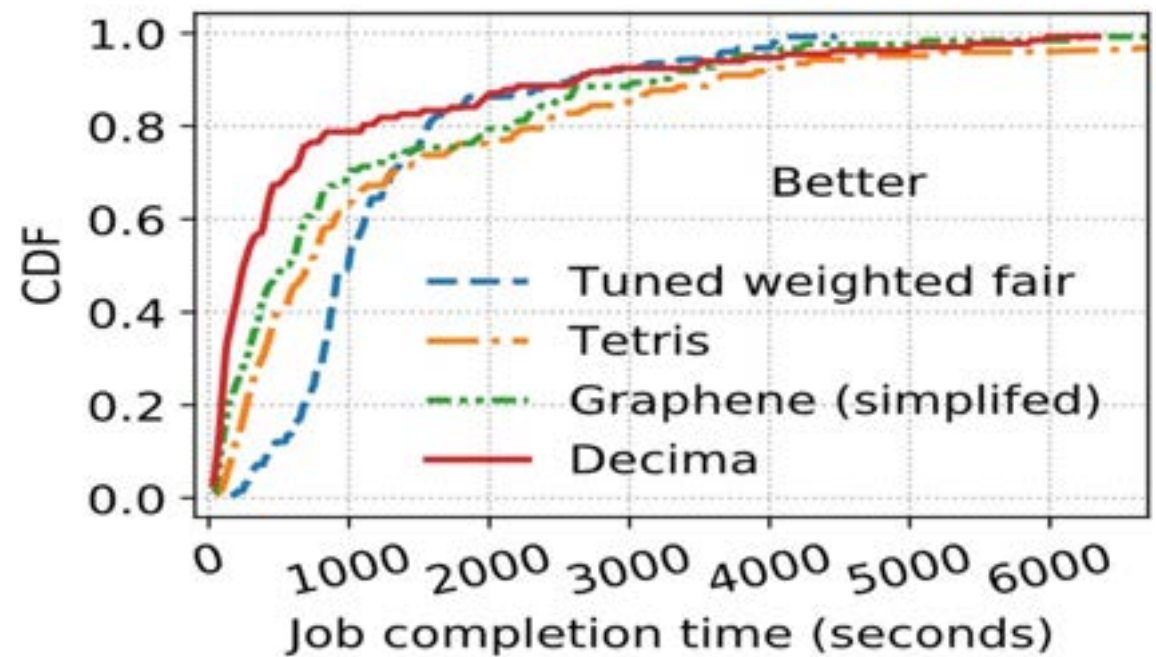
## Key Points

- Improved resource utilization
- Better performance
- Increased scalability

## Real Life DECIMA

**Industrial trace  
(Alibaba):**

20,000 jobs from  
production  
cluster.



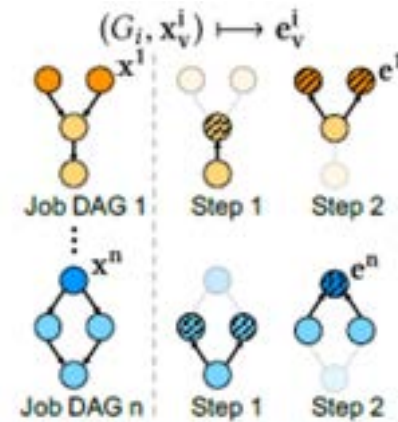


## Similar to DECIMA

- 1) PARAGON (Collaborative Filtering)
- 2) QUASAR
- 3) FIRMAMENT (Constraint solver)
- 4) GRAPHENE (Heuristics)

# GNN

$$e_v^i = g \left[ \sum_{u \in \xi(v)} f(e_u^i) \right] + x_v^i,$$



Here  $f(\cdot)$  and  $g(\cdot)$  are non-linear transformations done over nodes to produce vector outputs.

The reason is that without  $g(\cdot)$ , the graph neural network cannot compute certain useful features for scheduling. For example, it cannot compute the critical path




# QUESTIONS

Why does parallelism slow down Spark Clusters?

When we schedule the tasks in parallel it only works better until a certain limit i.e until enough executors are available to assign so that is the sweet-spot so after that as number of executors become less then even if we add more task in parallel it won't affect much.

Can Decima be used in clusters with different types of workloads, or is it specific to certain types of data processing tasks?

Yes we have tested Decima on different workloads and have plotted the graph for Batched Job arrivals. We have tested on different workload like 2,5,10,20,50,100 GB

What reinforcement learning algorithm was used to train the Decima system.

It Uses Policy Gradient Algorithm to train itself. Main idea is to find the local minima by gradient descent method. It is basically an optimization technique its done by differentiating.

Can the approach used by Decima be applied to other types of cluster scheduling problems beyond data processing?

Yes to database query management.


Can you briefly explain scheduling events in section 5.2?

$\langle v, l_i \rangle$

It passes embedding vectors to policy parameter then it produces 2 dimensional output  $\langle v, l_i \rangle$ . Here  $V$  is stage and  $l_i$  = parallelism limit &  $i$  is job. If job has less executors than its limit i.e  $l_i$  Decima assign executors.