

Building and Optimizing Learning-augmented Computer Systems

Hongzi Mao

October 24, 2019



- **Learning Scheduling Algorithms for Data Processing Clusters.** Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, Mohammad Alizadeh. *ACM SIGCOMM*, 2019.

Motivation

Scheduling is a fundamental task in computer systems

- Cluster management (e.g., Kubernetes, Mesos, Borg)
- Data analytics frameworks (e.g., Spark, Hadoop)
- Machine learning (e.g., Tensorflow)



kubernetes



MESOS

Efficient scheduler matters for large datacenters

- Small improvement can save millions of dollars at scale



TensorFlow

Designing Optimal Schedulers is Intractable

Must consider many factors for optimal performance:

- Job dependency structure *Graphene [OSDI '16], Carbyne [OSDI '16]*
- Modeling complexity *Tetris [SIGCOMM '14], Jockey [EuroSys '12]*
- Placement constraints *TetriSched [EuroSys '16], device placement [NIPS '17]*
- Data locality *Delayed Scheduling [EuroSys '10]*
-

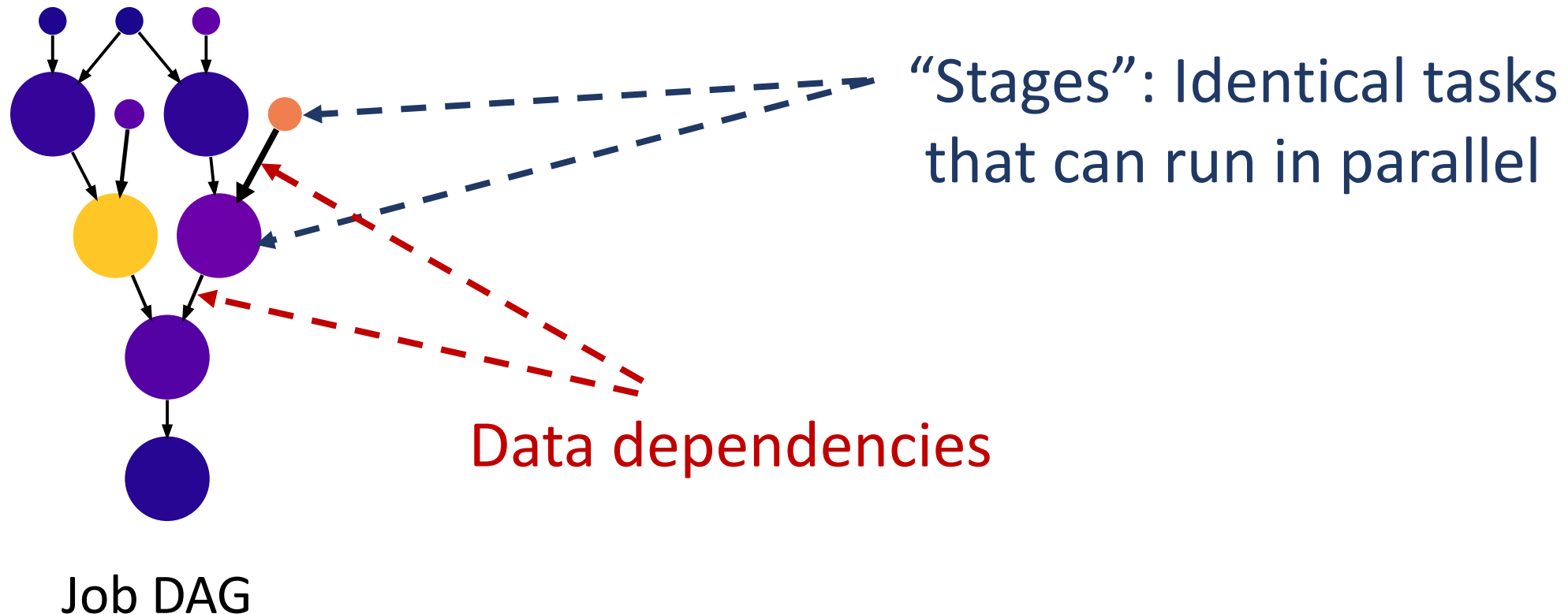
No “one-size-fits-all” solution:

Best algorithm depends on specific **workload** and **system**

**Can machine learning help tame the complexity of
efficient schedulers for data processing jobs?**

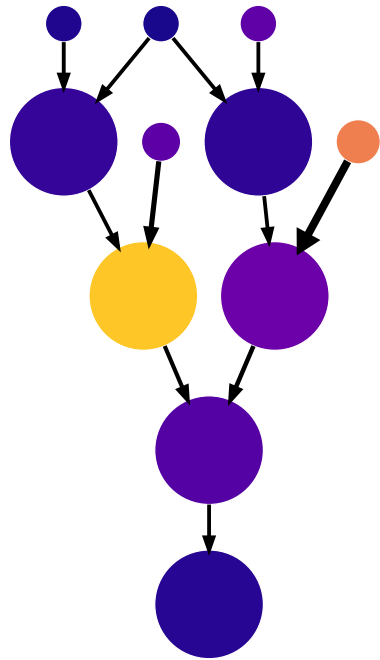
Decima: A Learned Cluster Scheduler

- Learns **workload-specific** scheduling algorithms for jobs with dependencies (represented as DAGs)

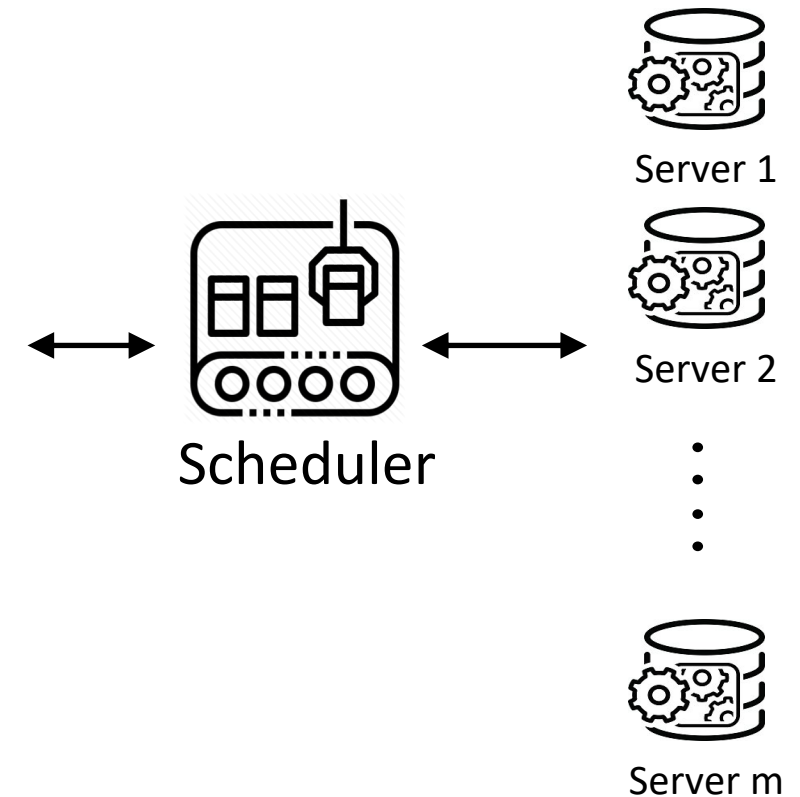


Decima: A Learned Cluster Scheduler

- Learns **workload-specific** scheduling algorithms for jobs with dependencies (represented as DAGs)

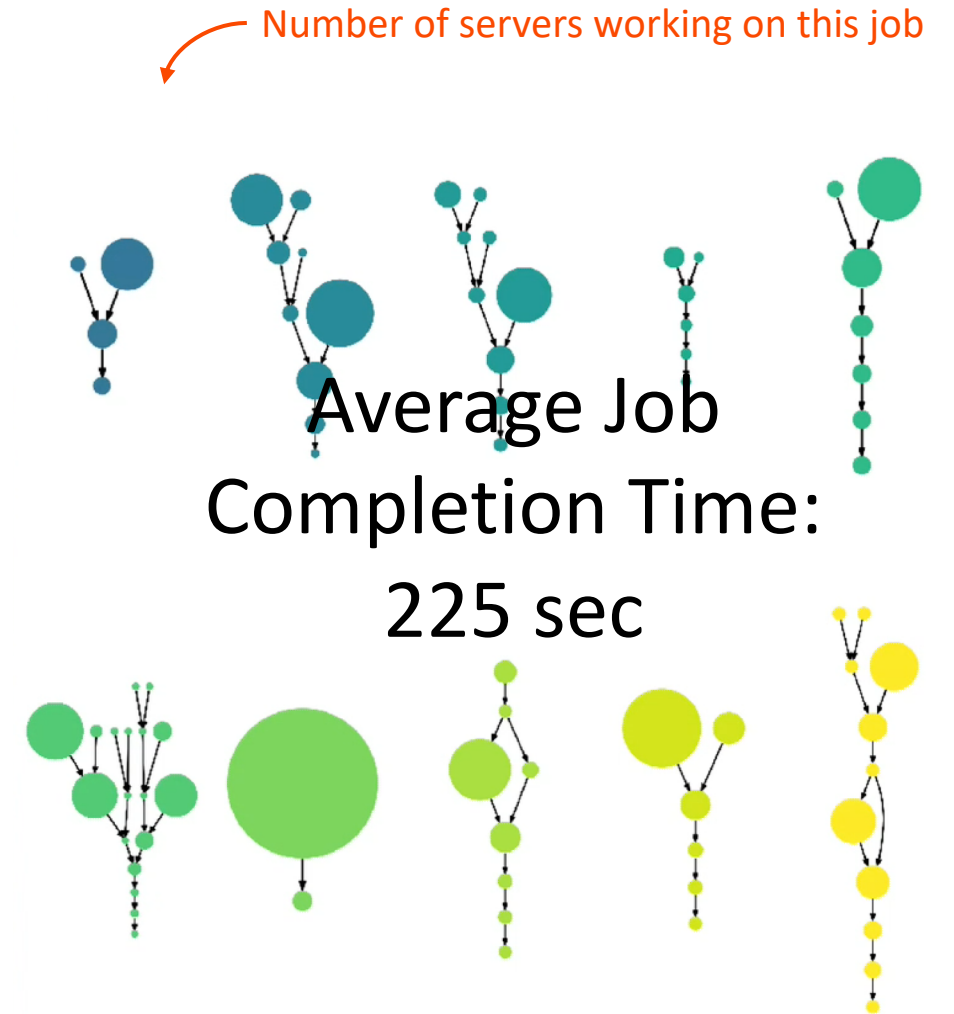


Job 1

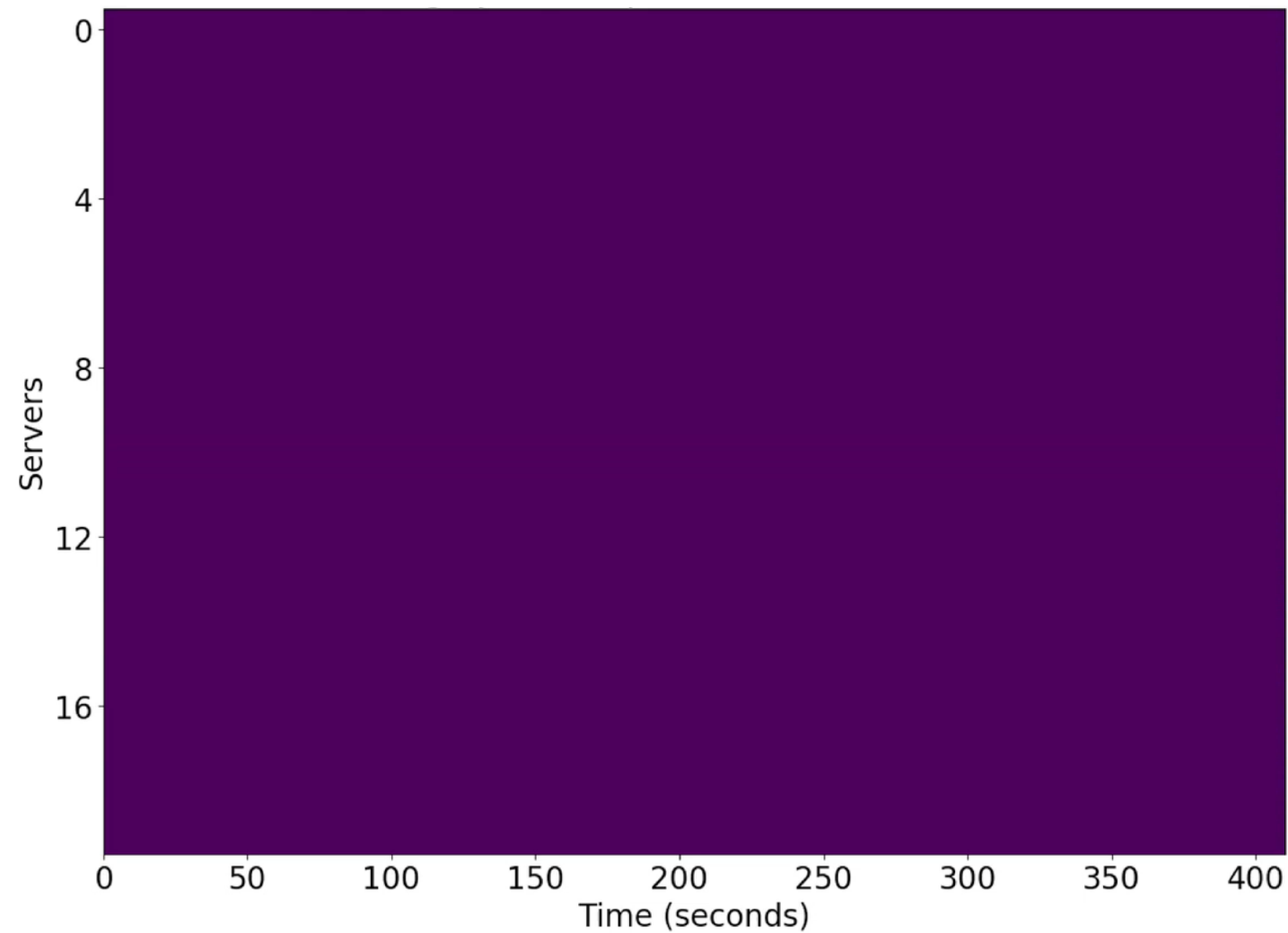


Scheduling policy: FIFO

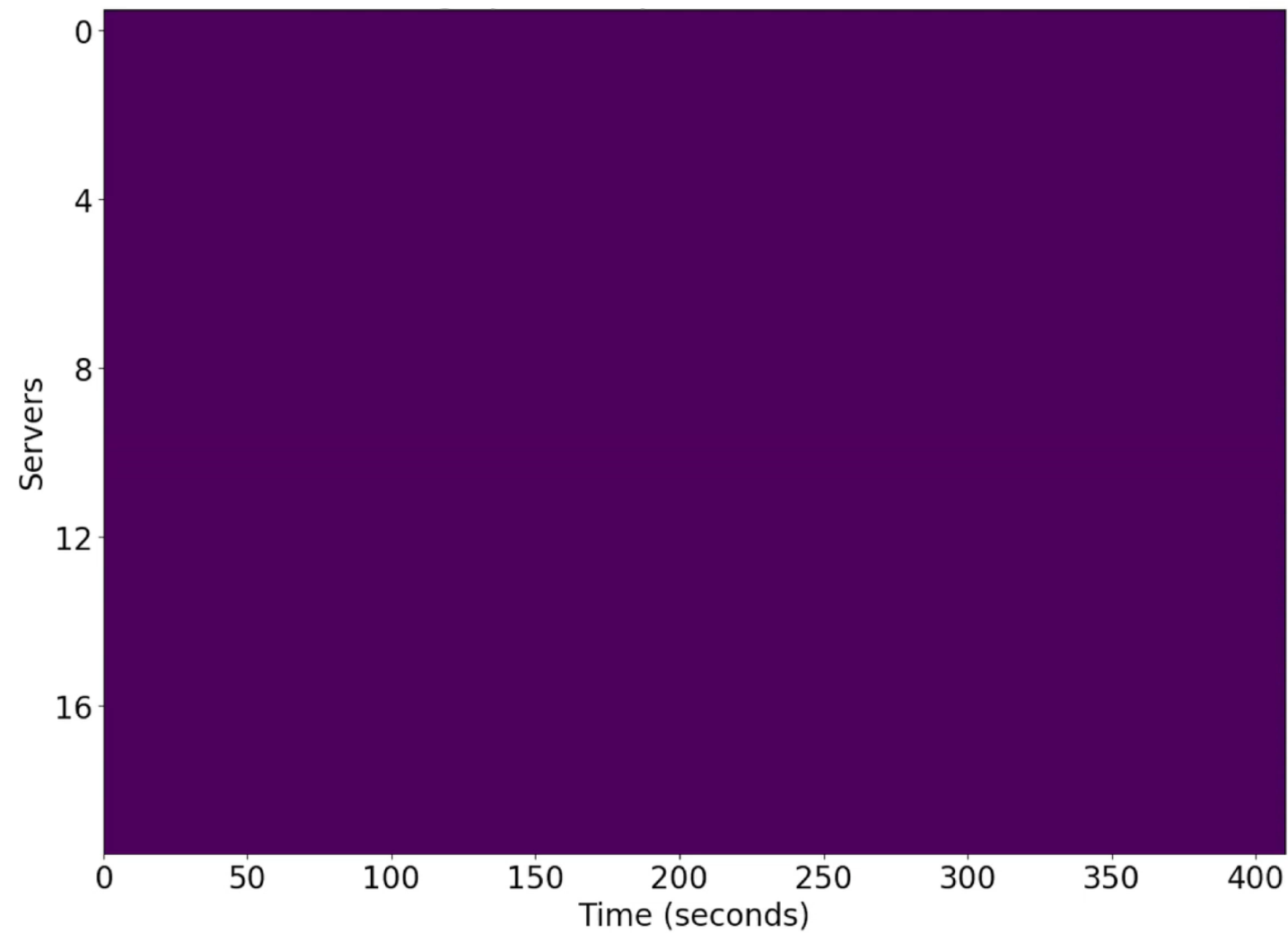
Demo



Scheduling policy: Shortest-Job-First

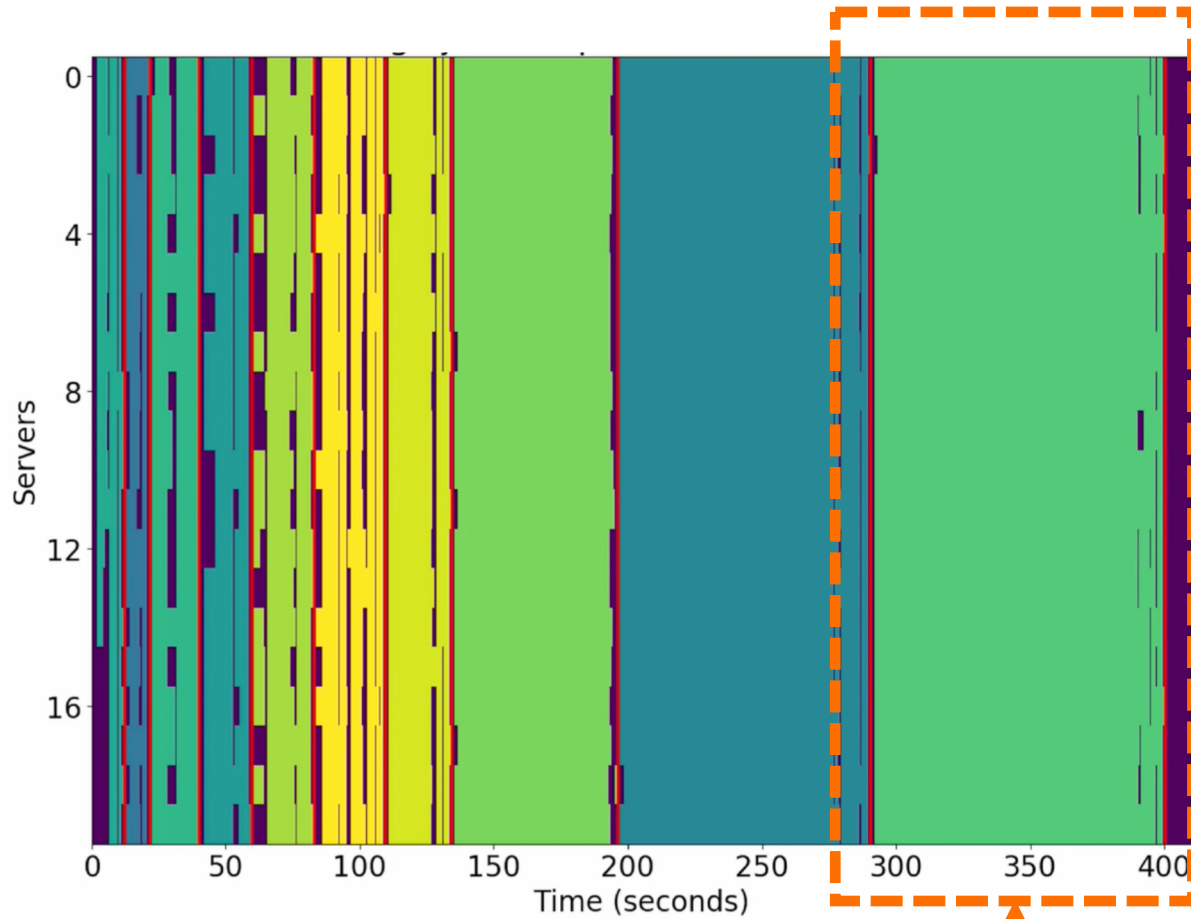


Scheduling policy: Fair



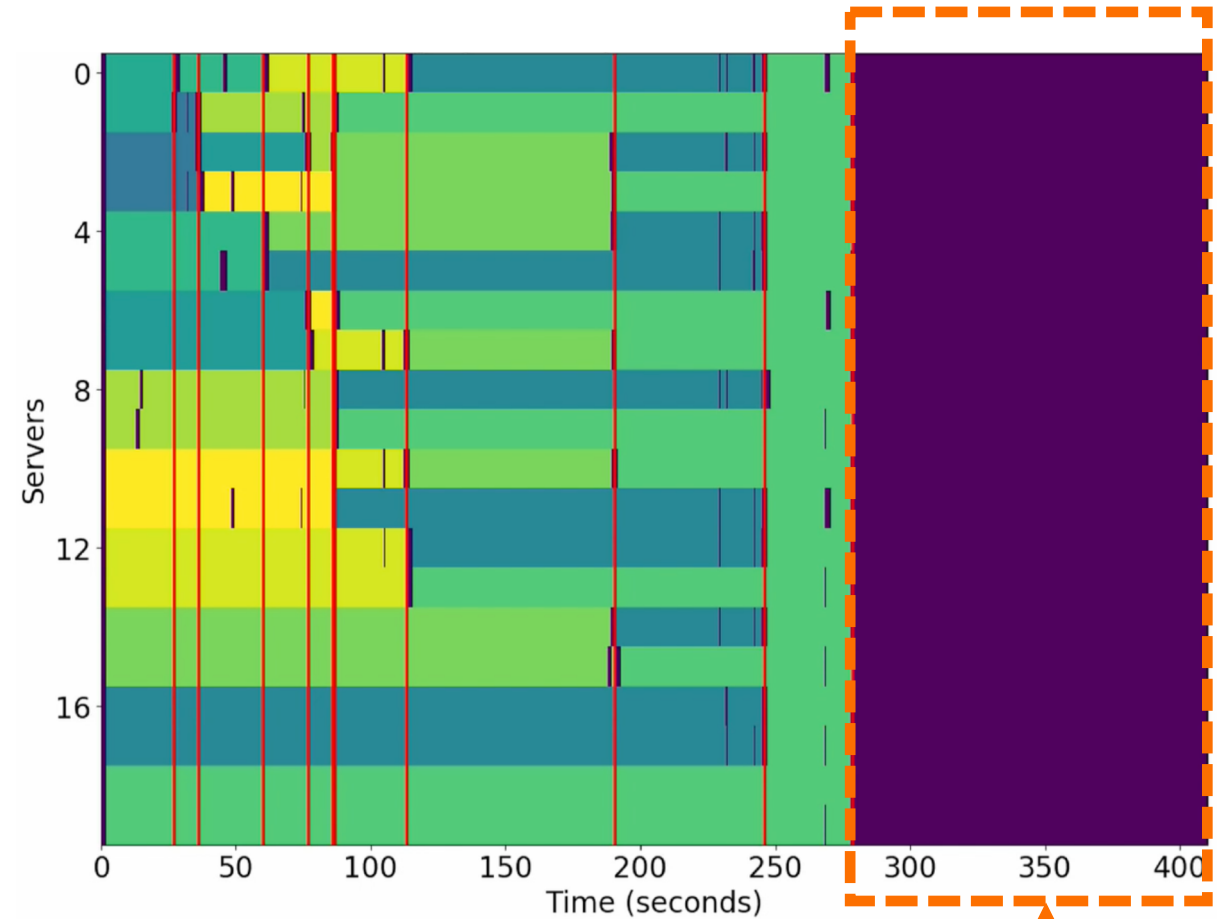
Average Job
Completion Time:
120 sec

Shortest-Job-First



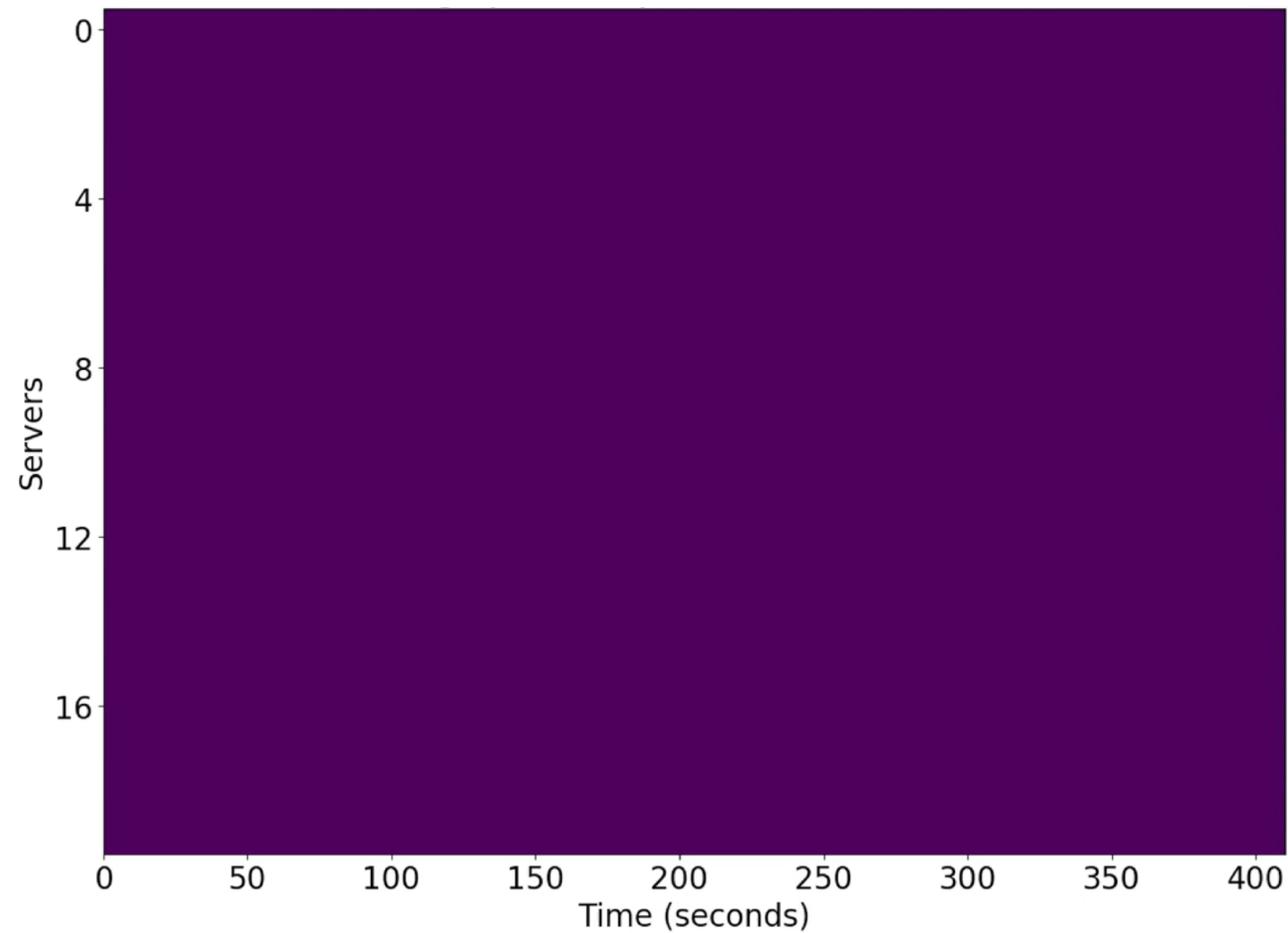
Average Job Completion Time:
135 sec

Fair

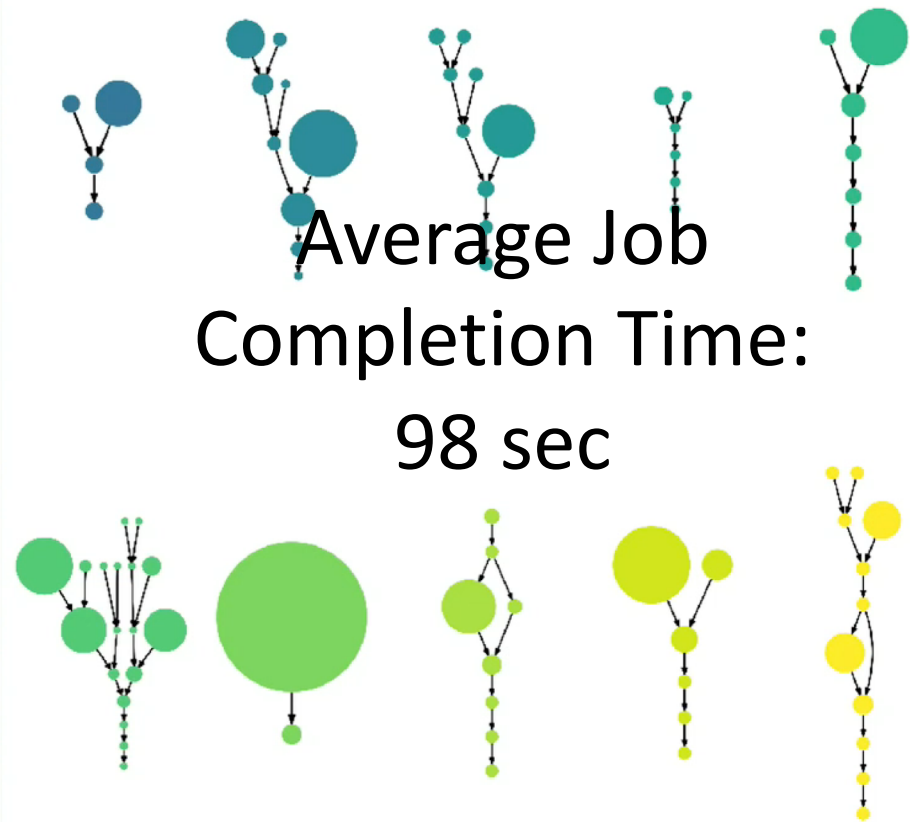


Average Job Completion Time:
120 sec

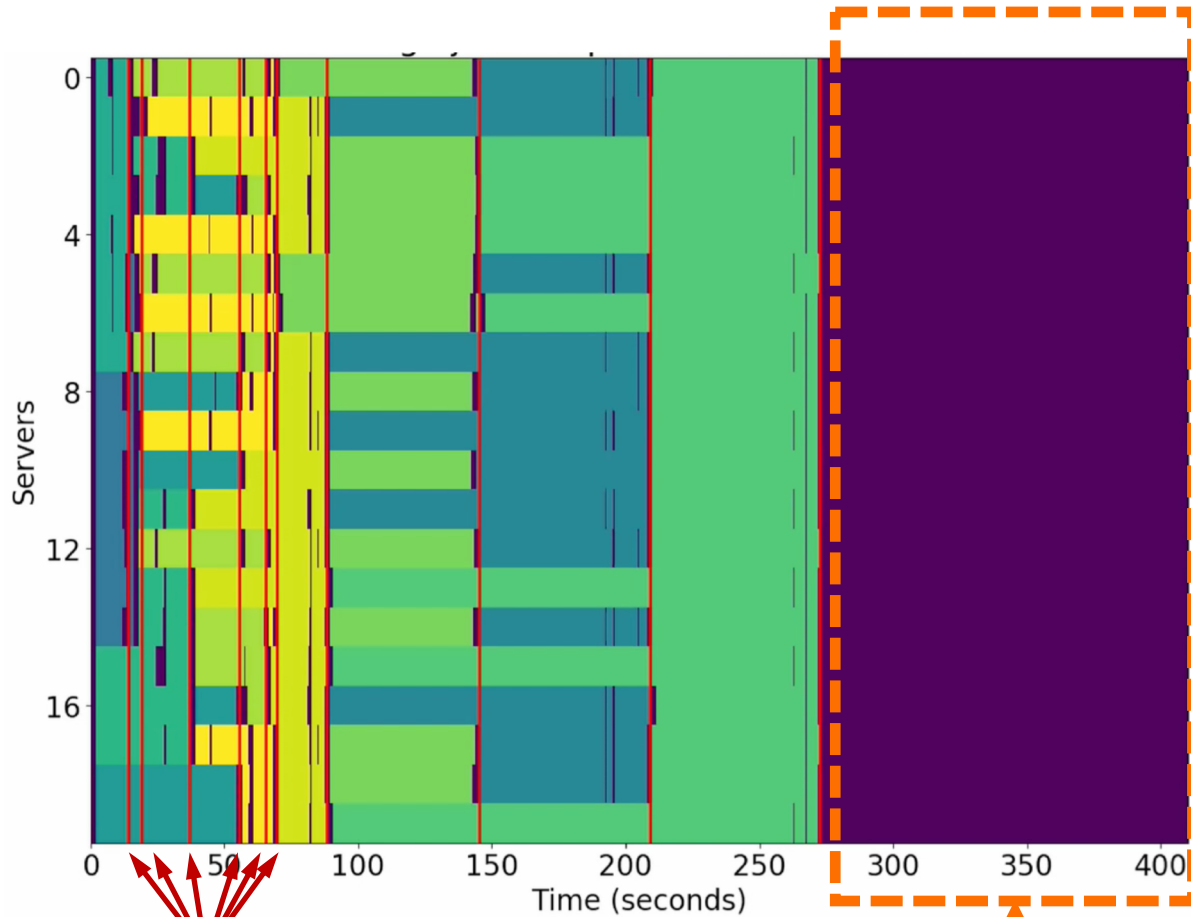
Scheduling policy: Decima



Average Job
Completion Time:
98 sec

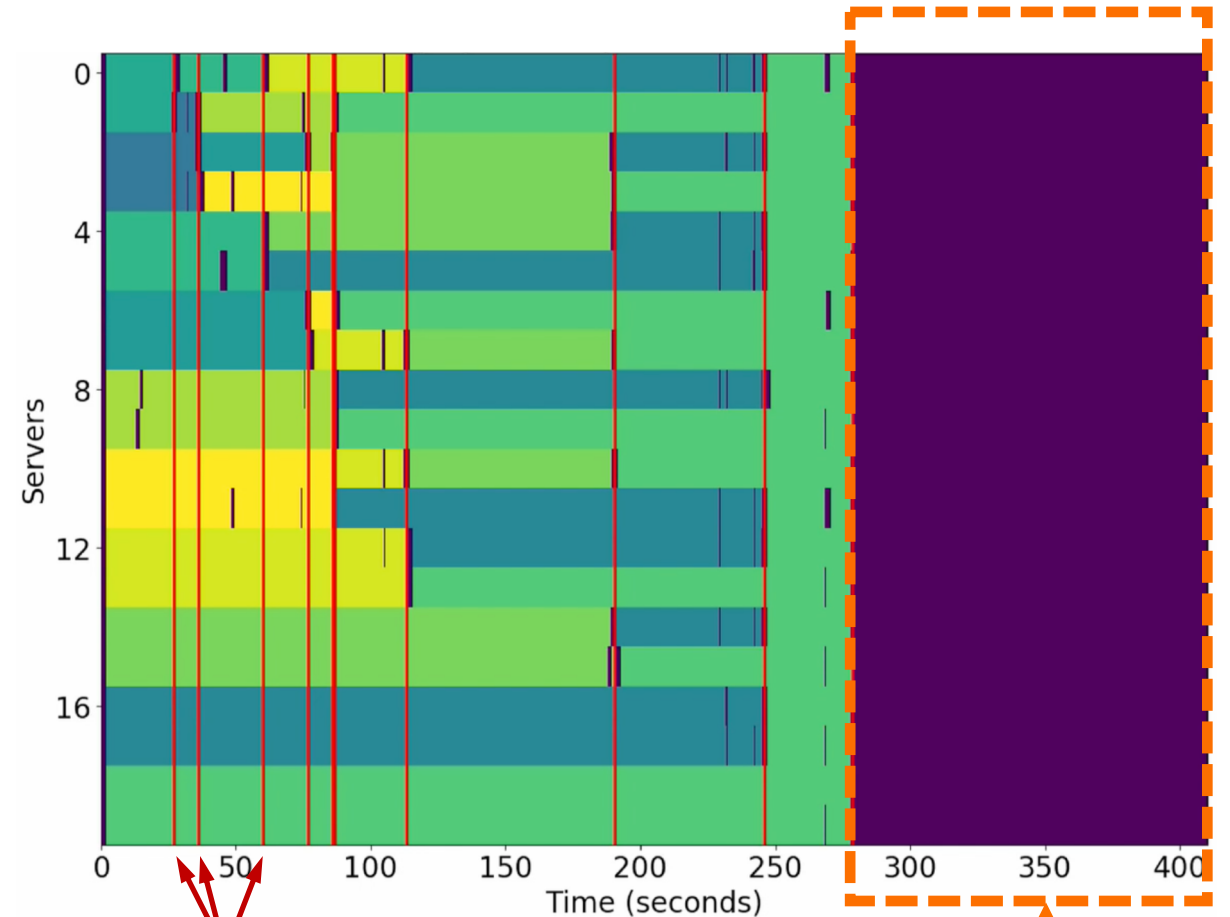


Decima



Average Job Completion Time:
98 sec

Fair

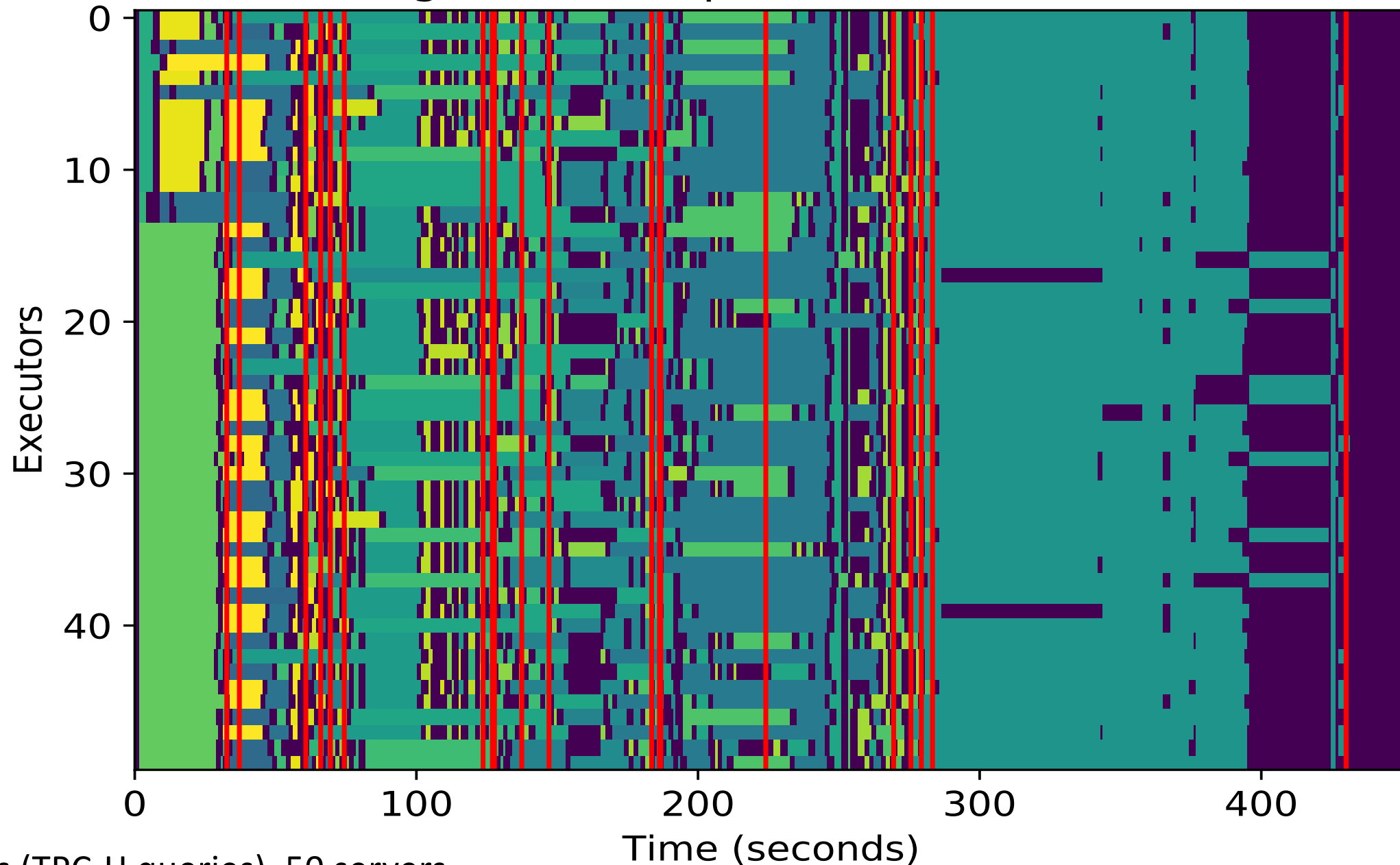


Average Job Completion Time:
120 sec

Decima

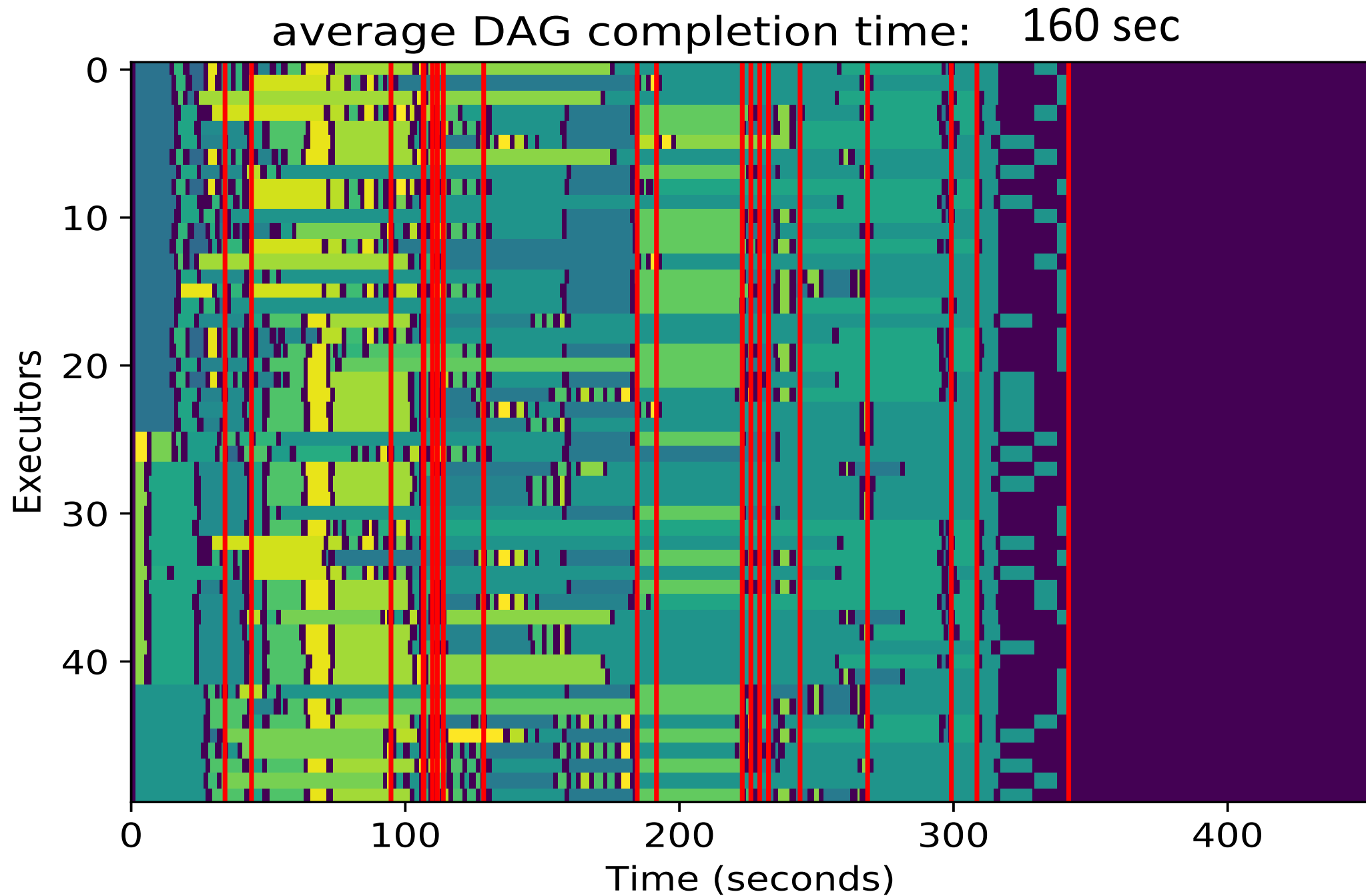
it=0

average DAG completion time: 166 sec

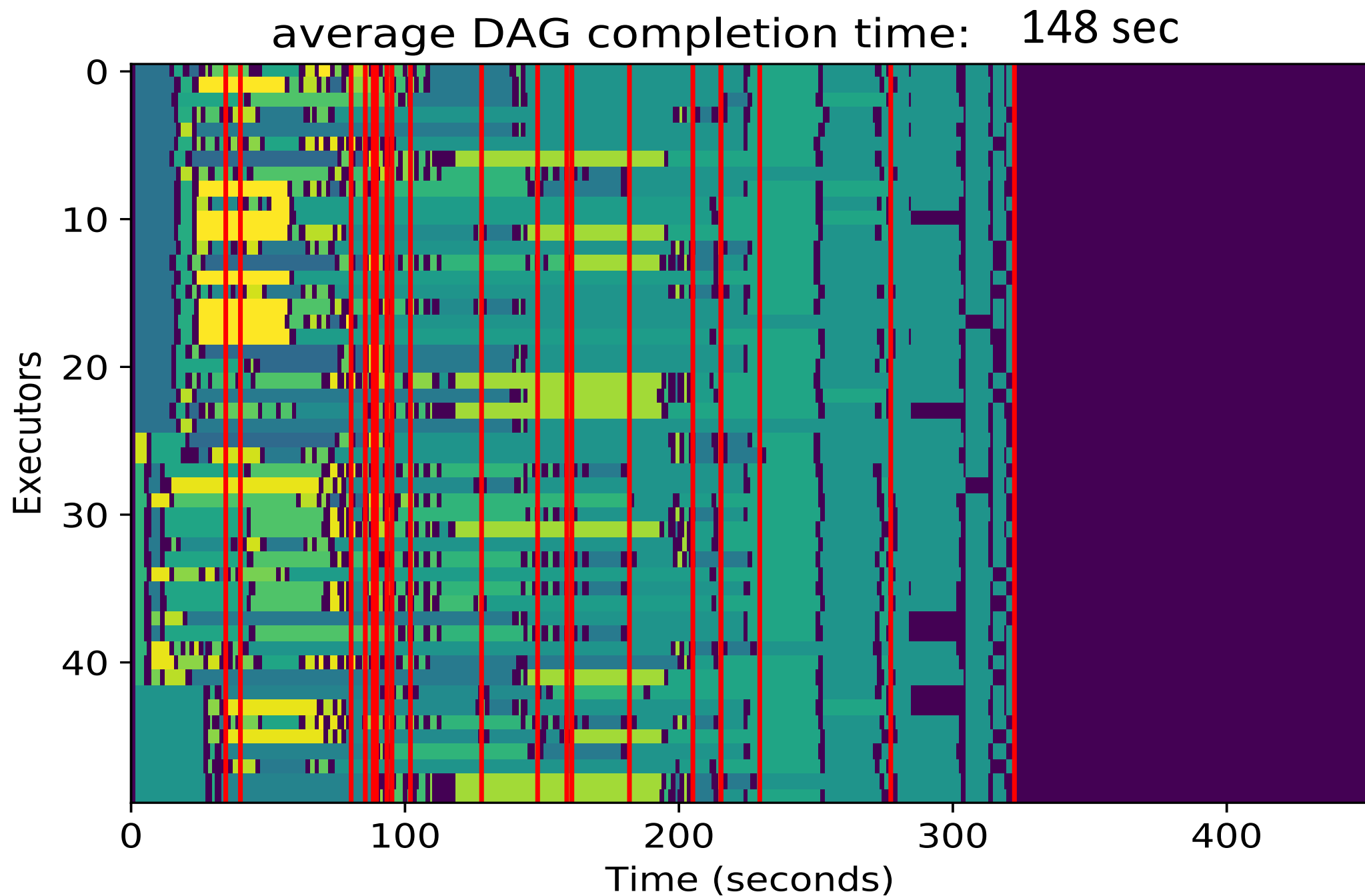


✧ 20 Spark jobs (TPC-H queries), 50 servers

Decima
it=3000

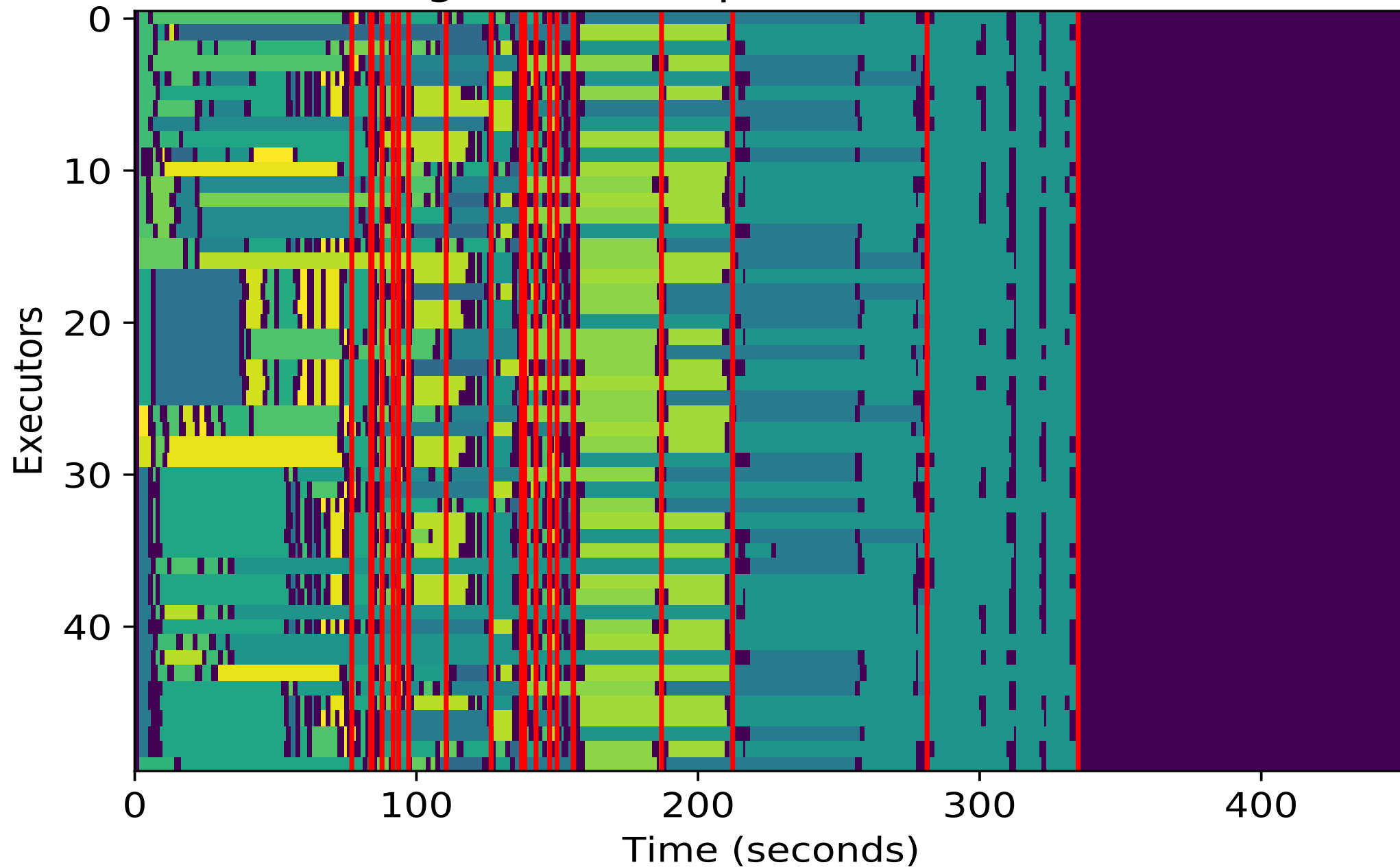


Decima
it=6000



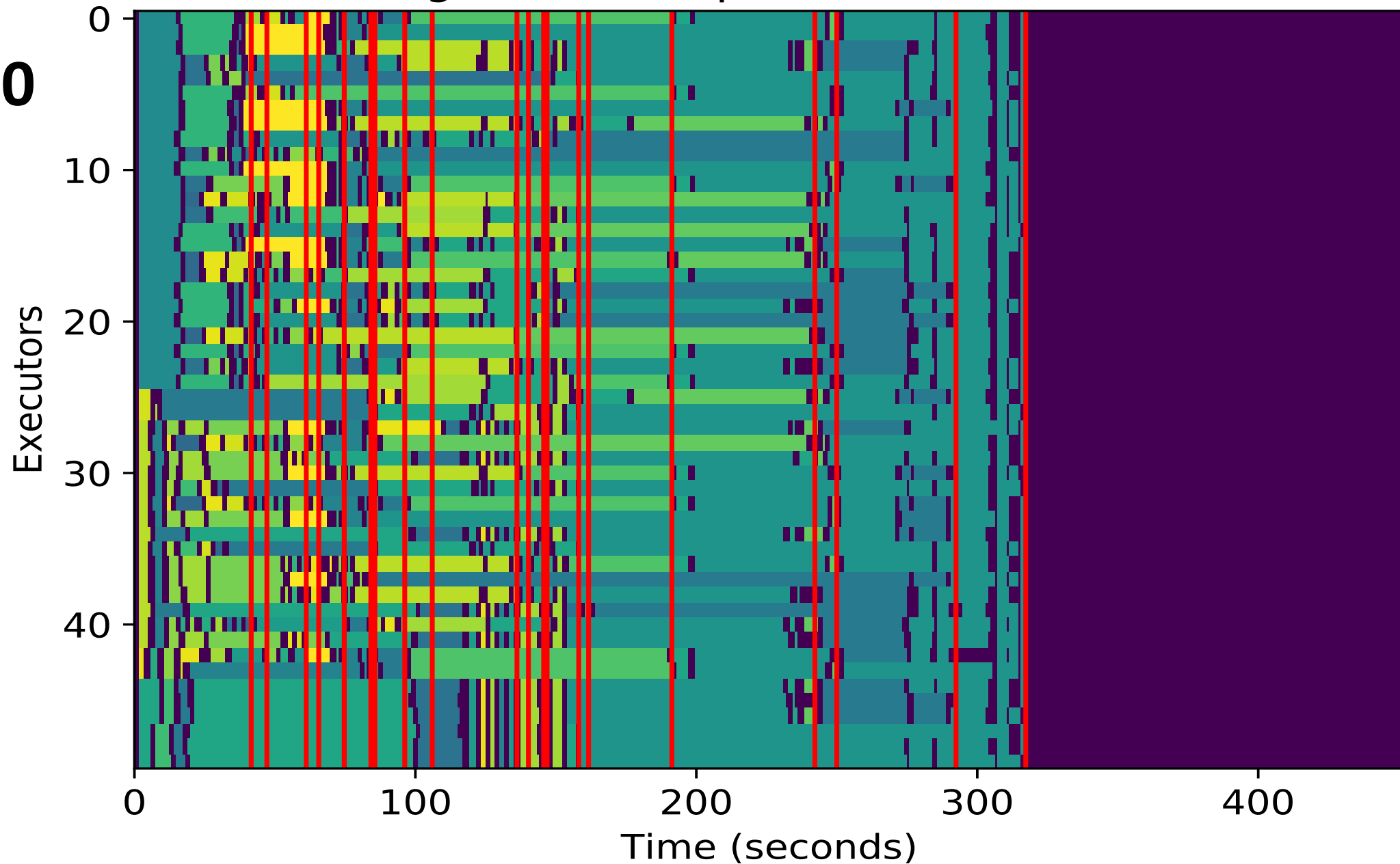
Decima
it=9000

average DAG completion time: 145 sec



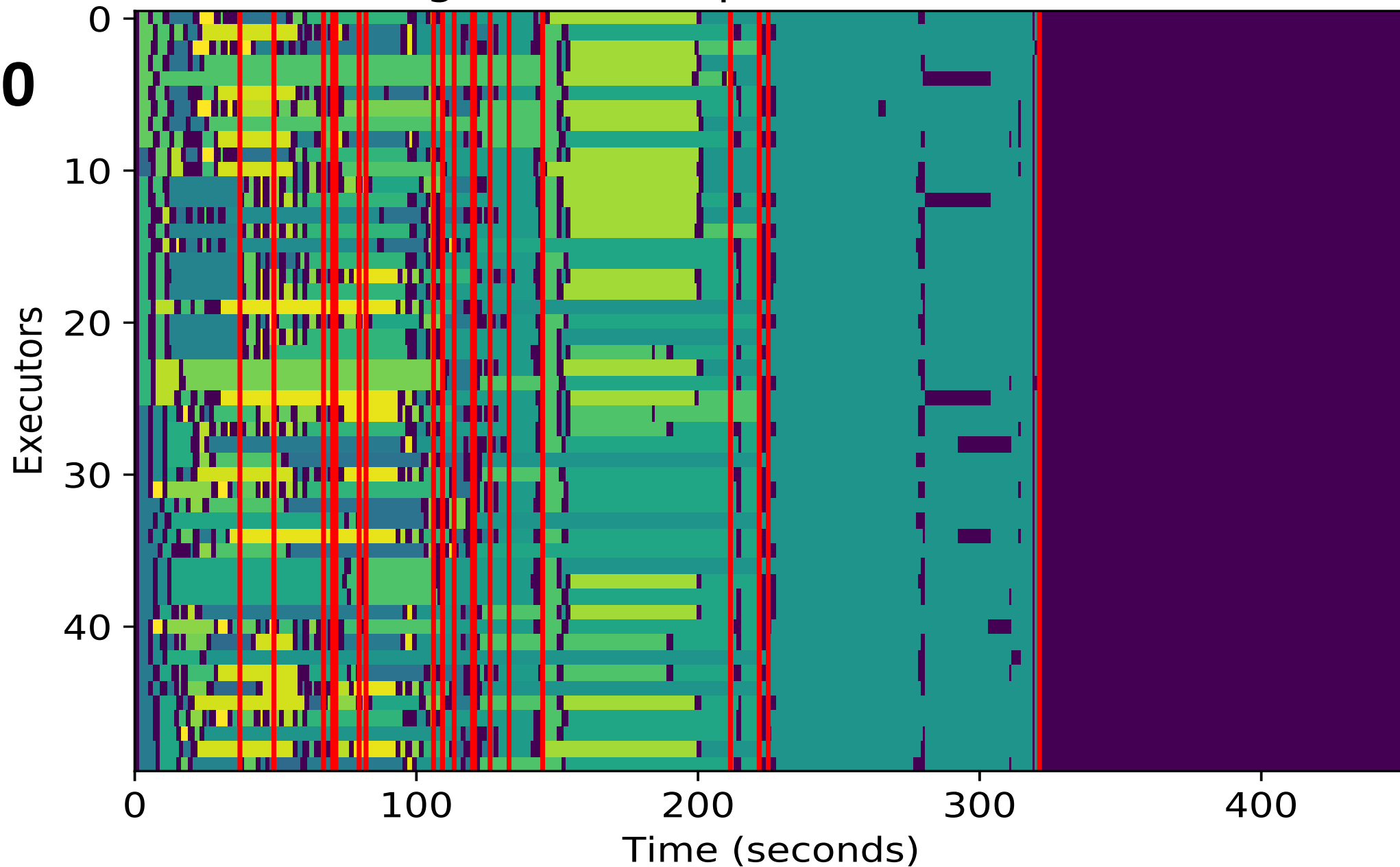
Decima
it=12000

average DAG completion time: 142 sec



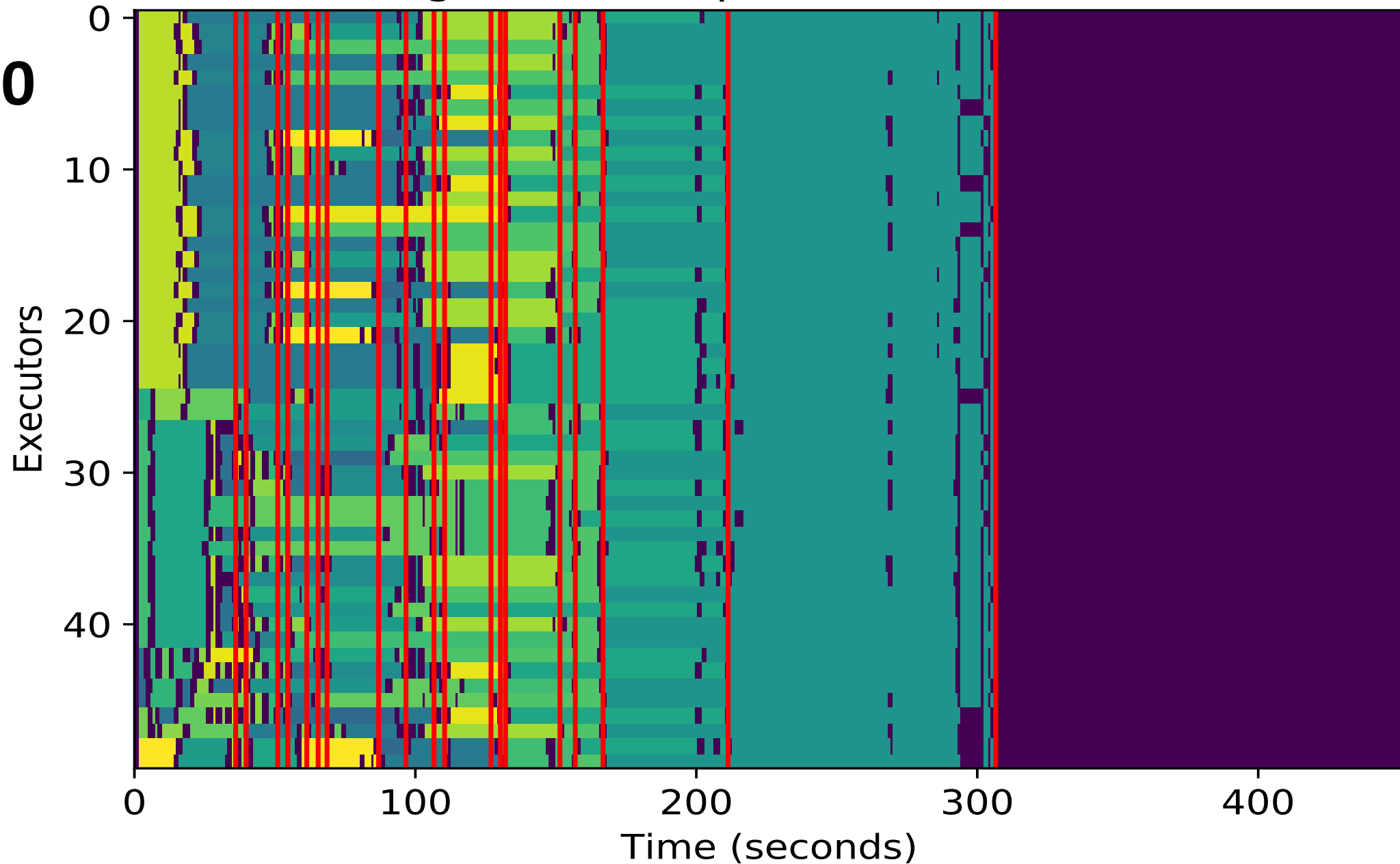
Decima
it=15000

average DAG completion time: 126 sec



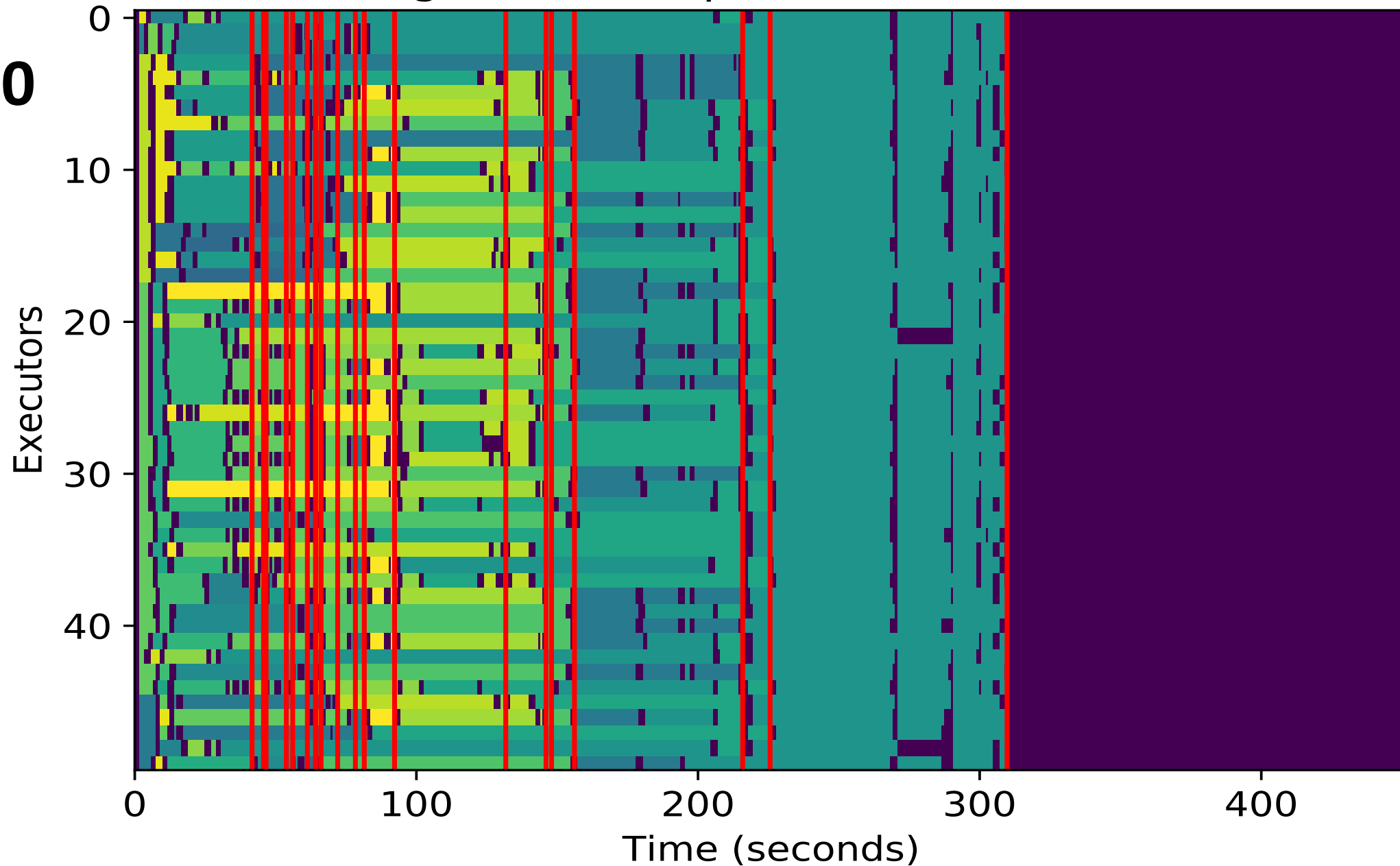
Decima
it=18000

average DAG completion time: 111 sec

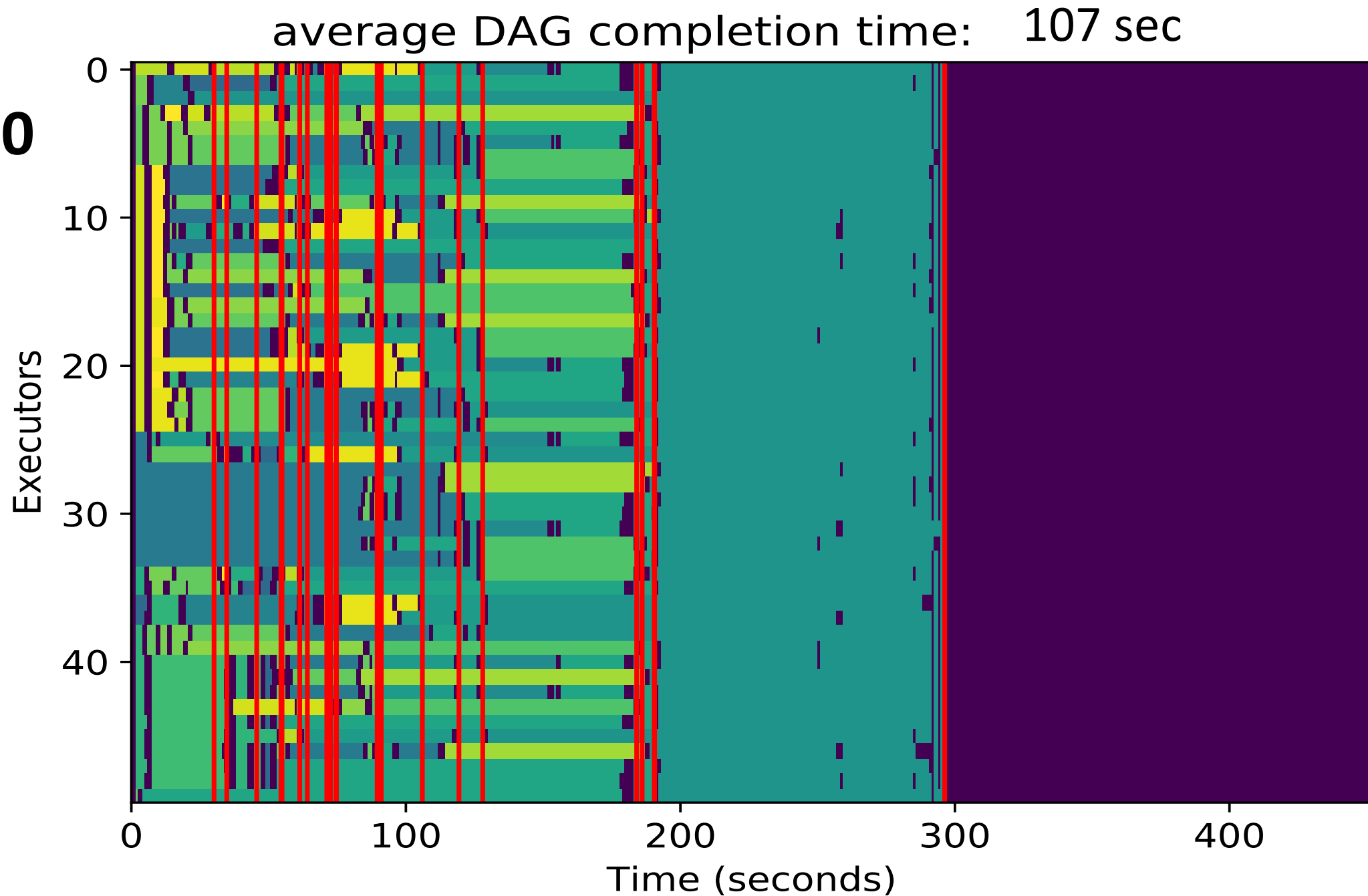


Decima
it=21000

average DAG completion time: 108 sec

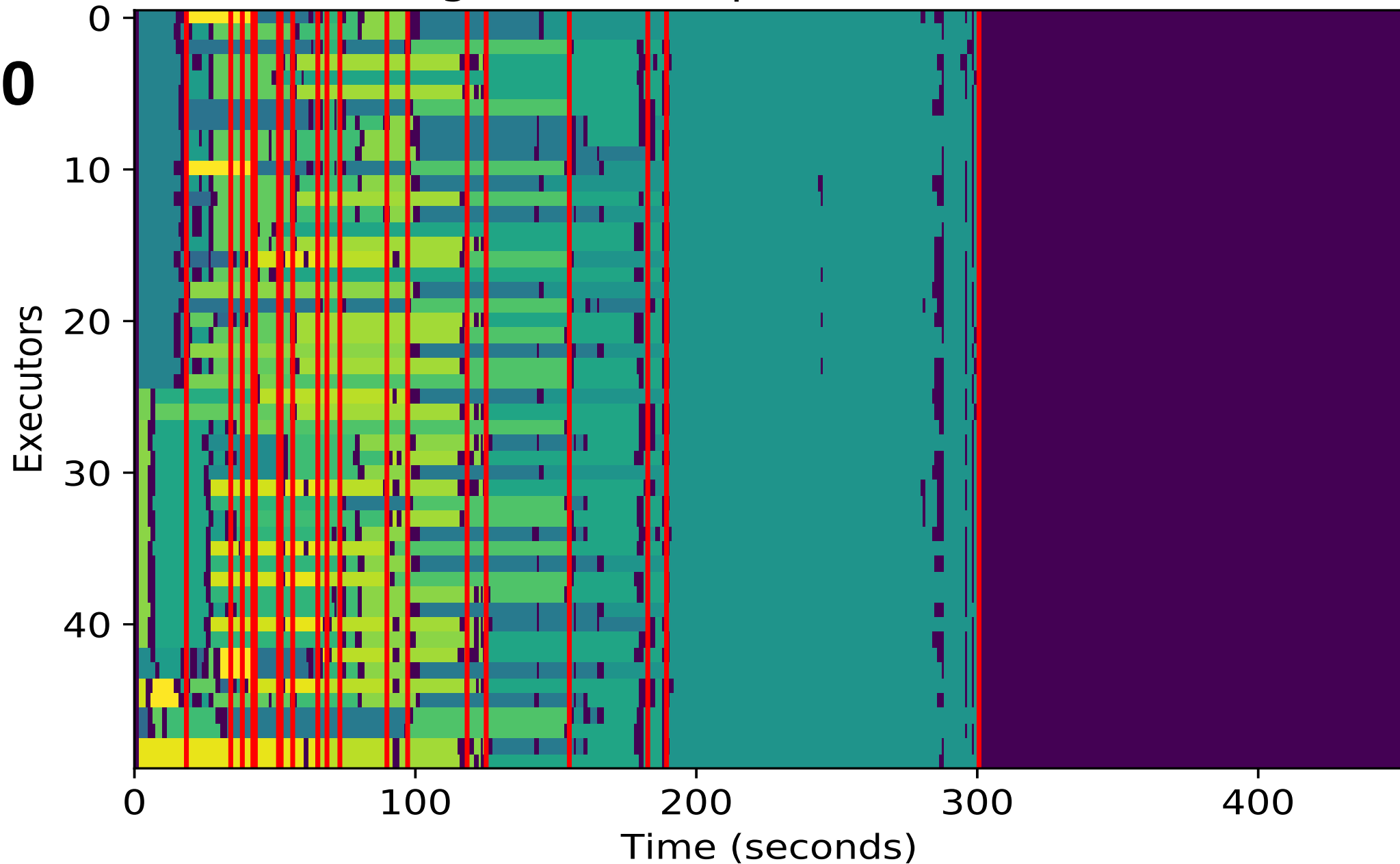


Decima
it=24000



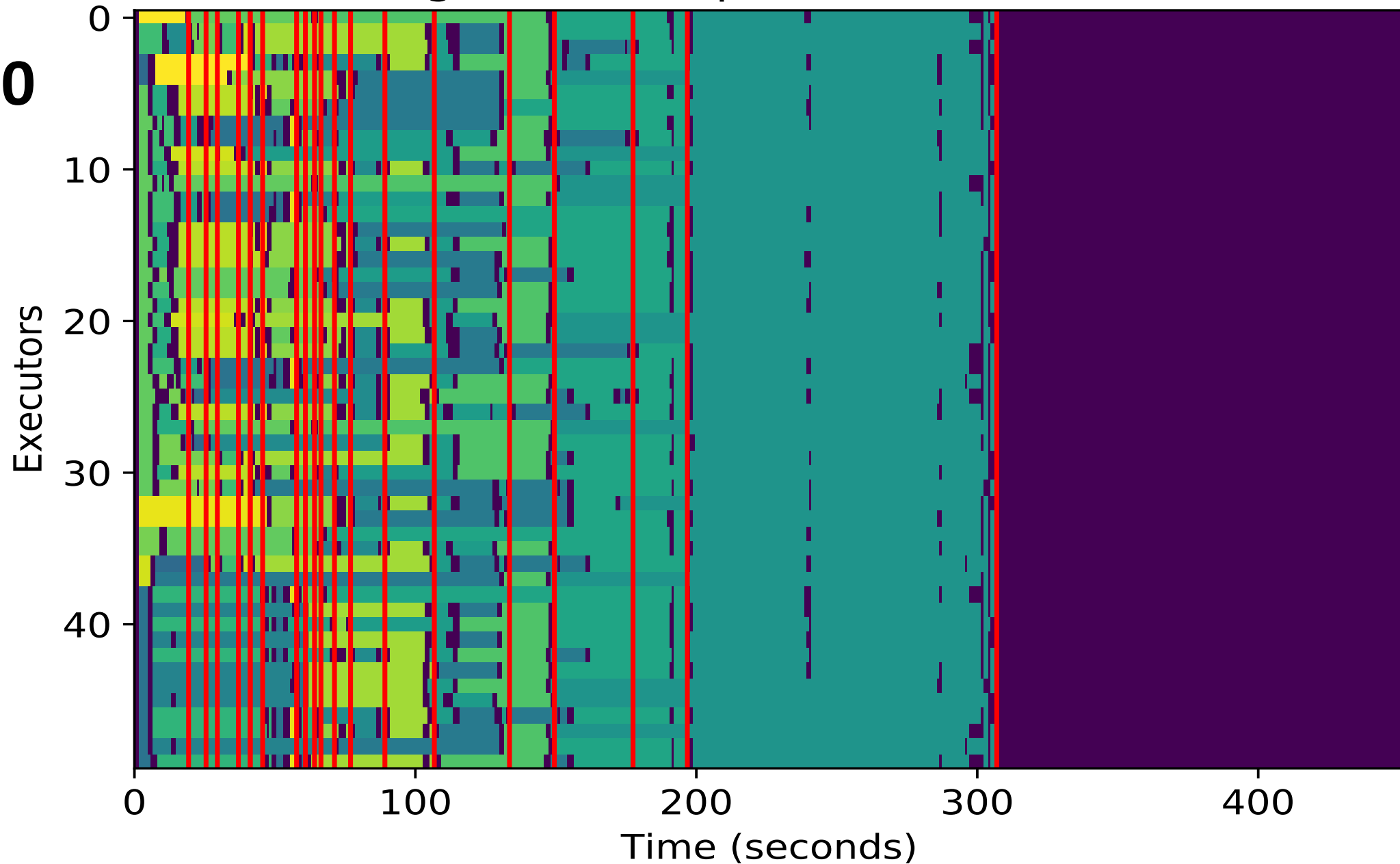
Decima
it=27000

average DAG completion time: 93 sec



Decima
it=30000

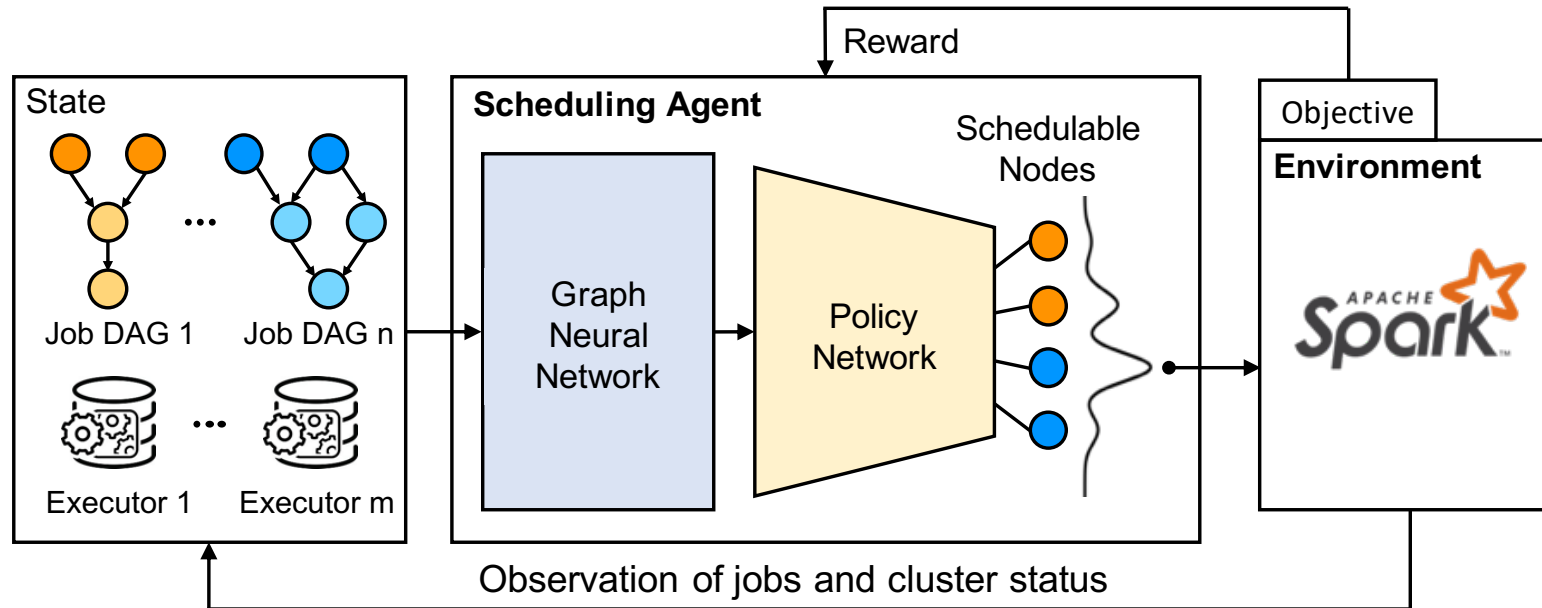
average DAG completion time: 89 sec



Design

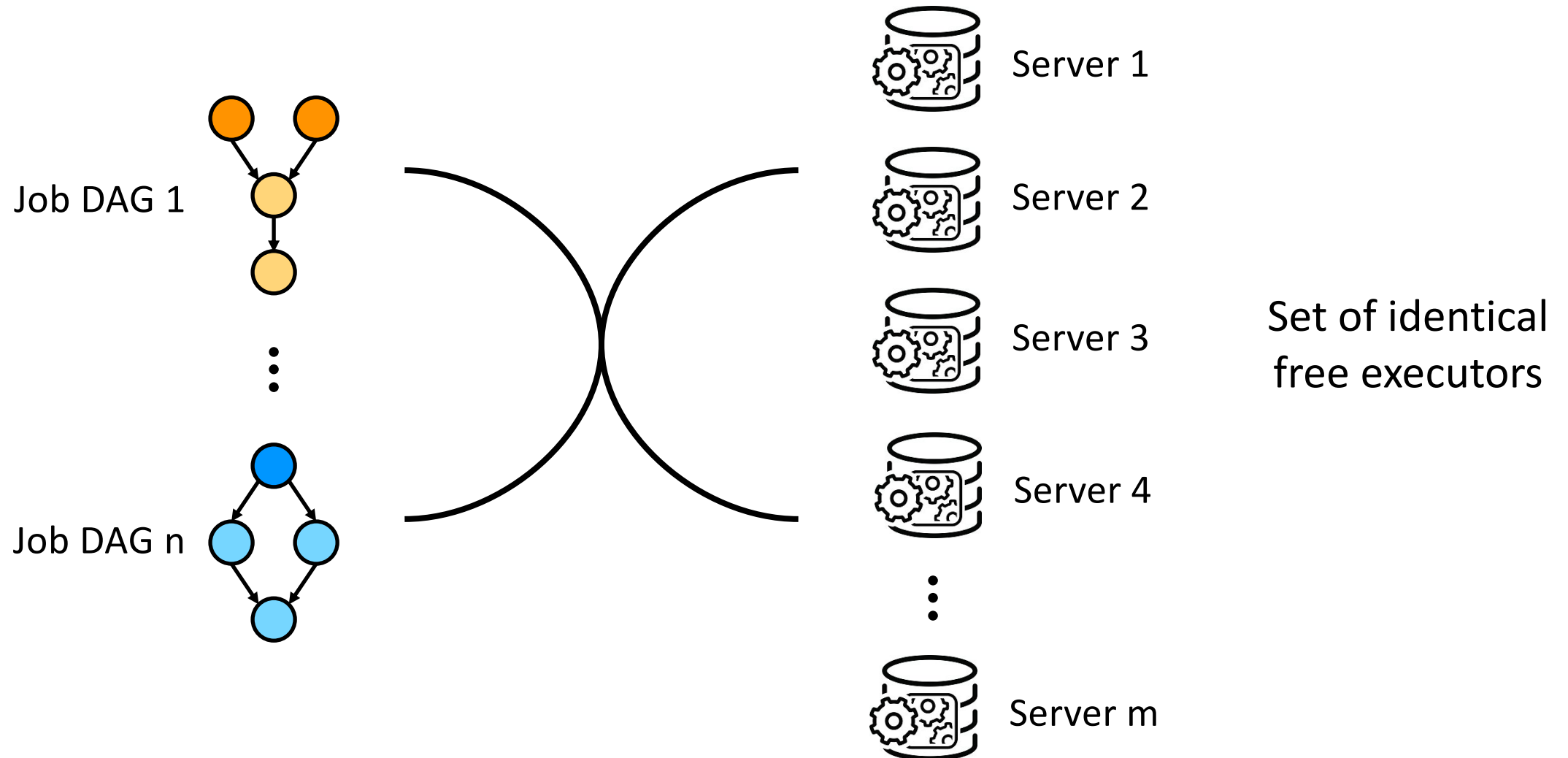
Design overview

Contributions

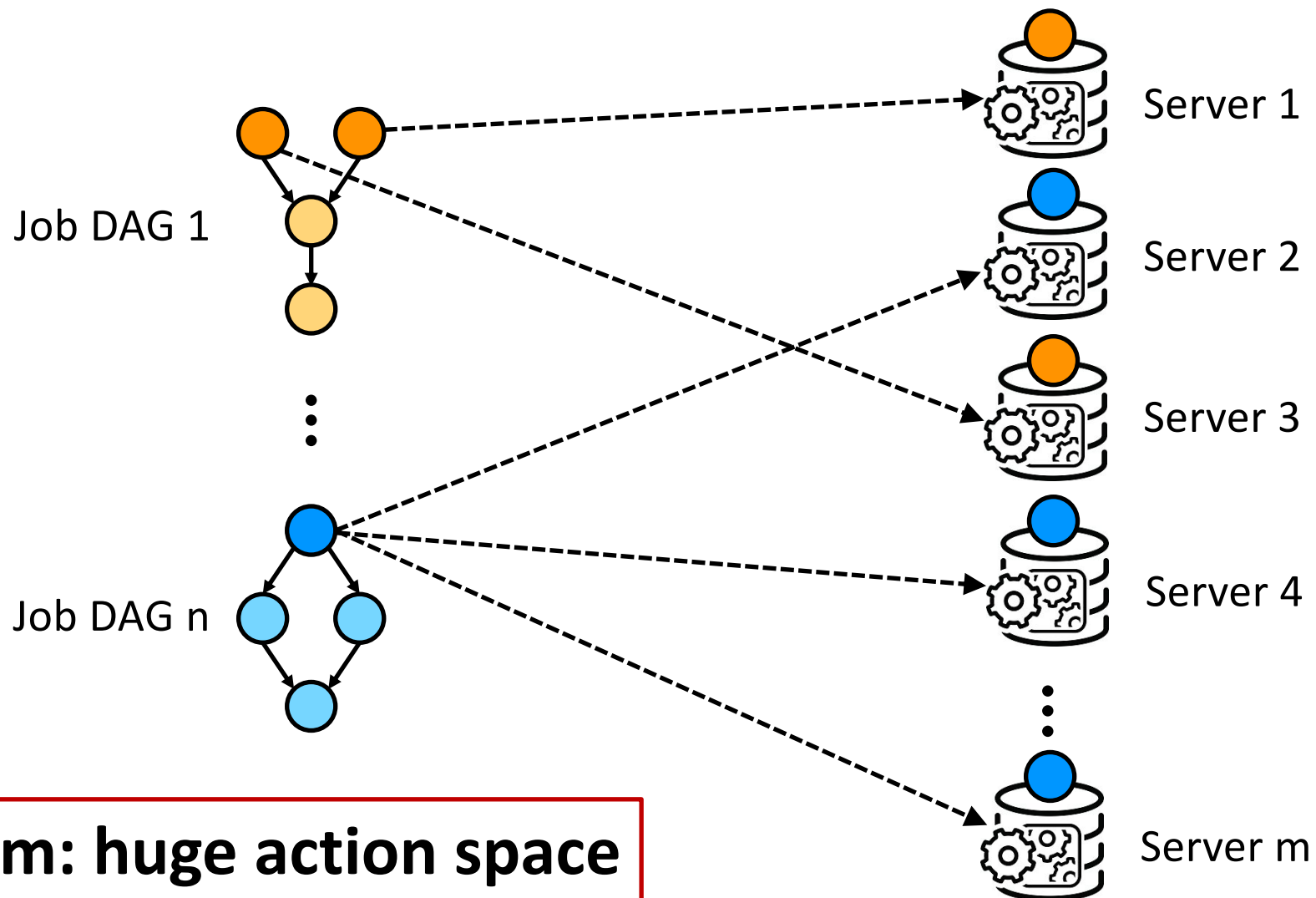


1. First RL-based scheduler for complex data processing jobs
2. Scalable graph neural network to express scheduling policies
3. New learning methods that enables training with online job arrivals

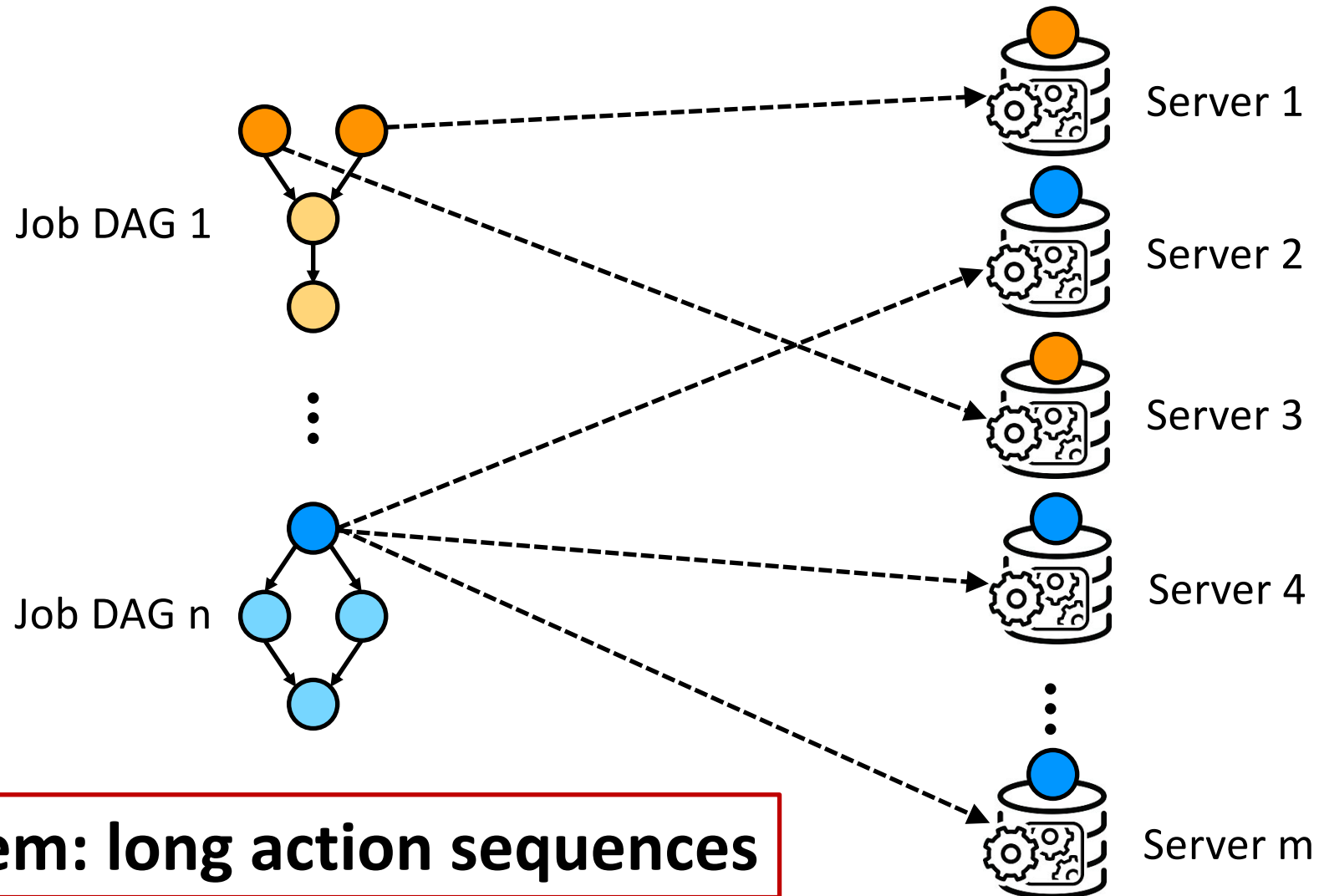
Encode scheduling decisions as actions



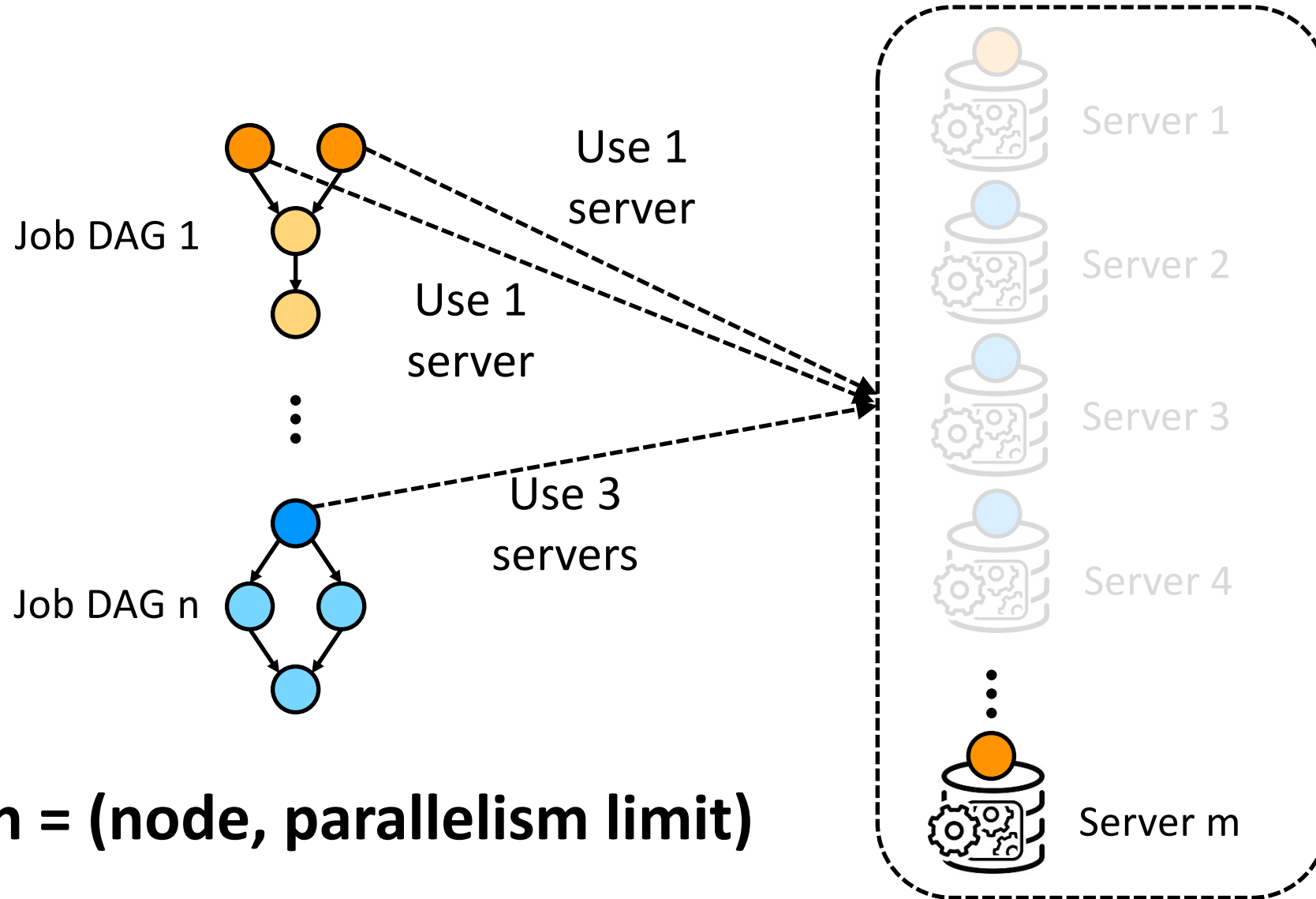
Option 1: Assign all Executors in 1 Action



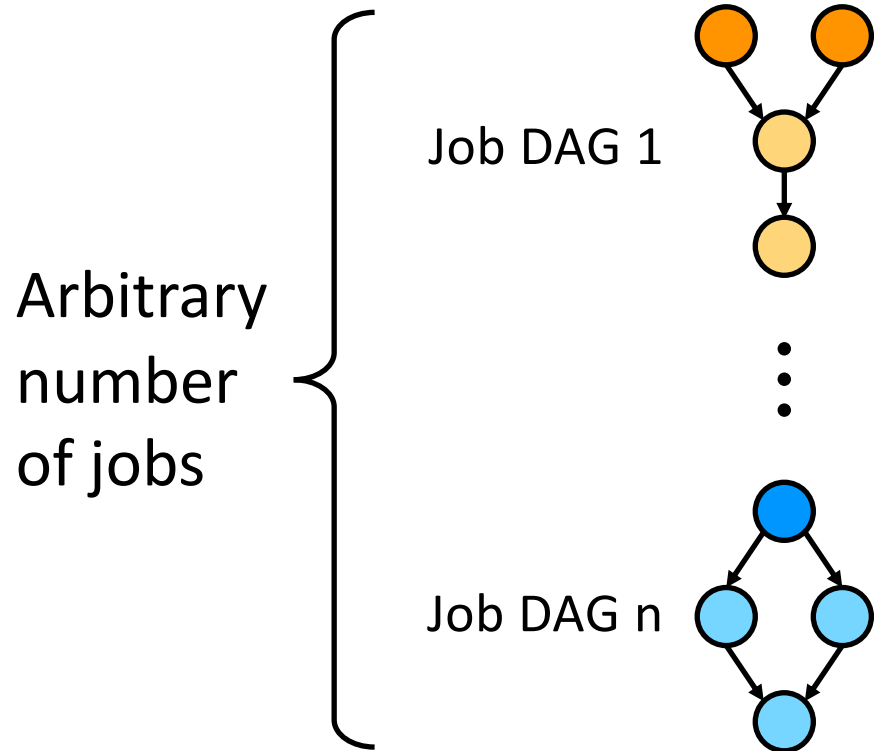
Option 2: Assign One Executor Per Action



Decima: Assign Groups of Executors per Action



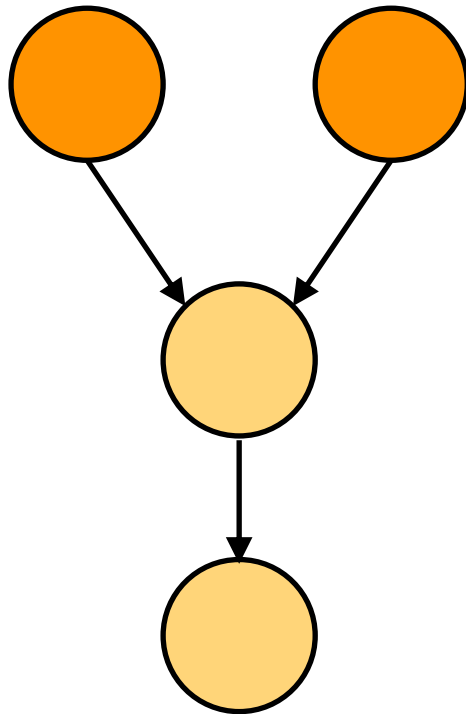
Process Job Information



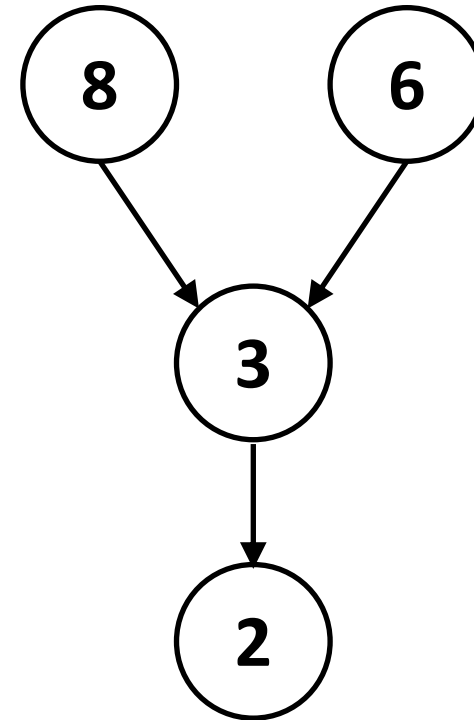
Node features:

- # of tasks
- avg. task duration
- # of servers currently assigned to the node
- are free servers local to this job?

Graph Neural Network

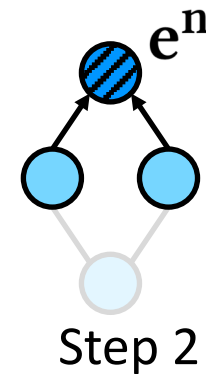
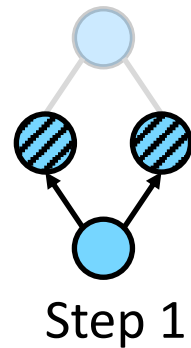
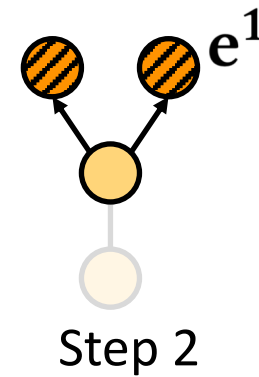
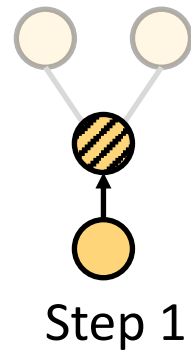
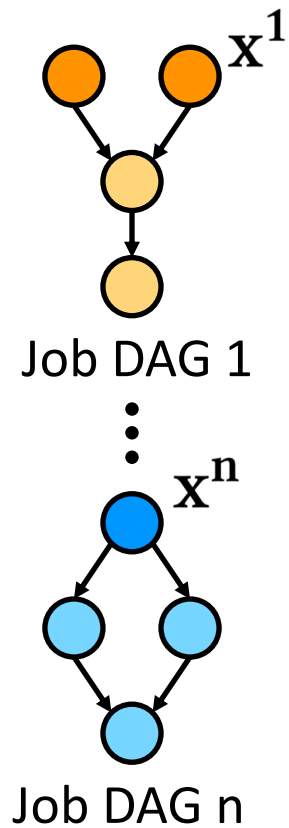


Job DAG



Score on
each node

Graph Neural Network

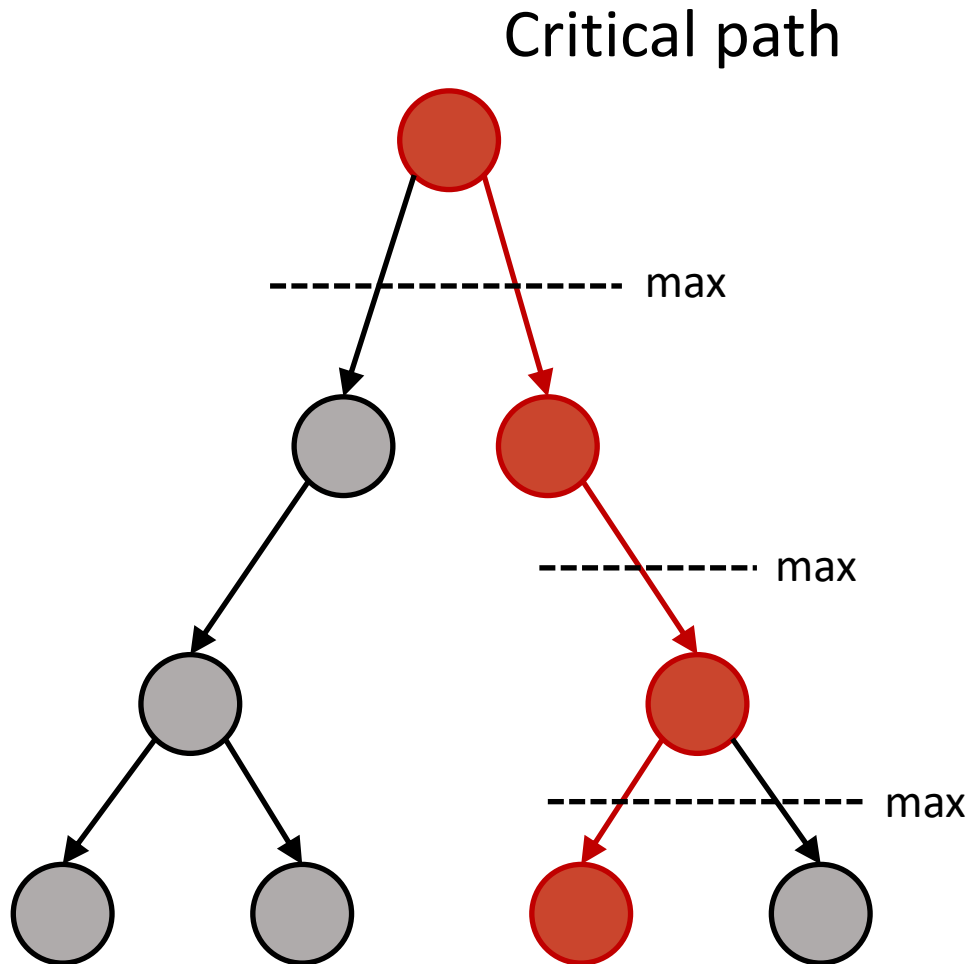


$$e_v = g \left(x_v, \sum_{w \in \xi(v)} e_w \right); \theta_v$$

Children of v

Same aggregation applied to all nodes for each DAG

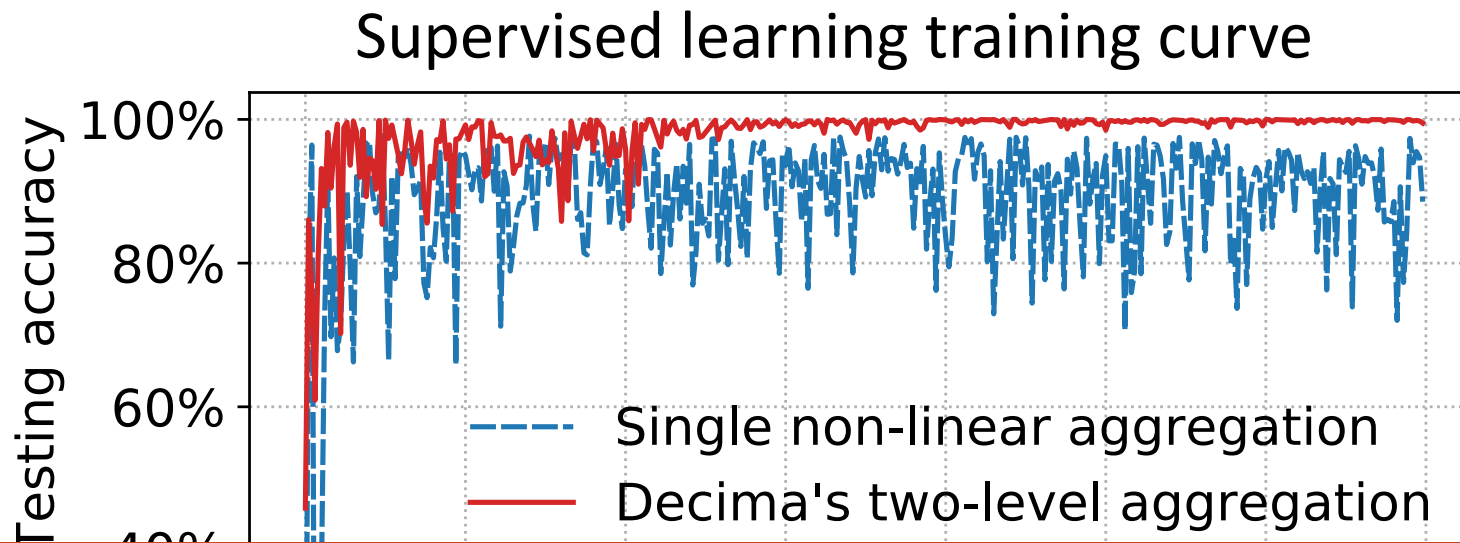
Graph Neural Network



$$\mathbf{e}_v = g \left[\sum_{w \in \xi(v)} f(\mathbf{e}_w) \right] + \mathbf{x}_v$$

Same aggregation applied everywhere in the DAGs

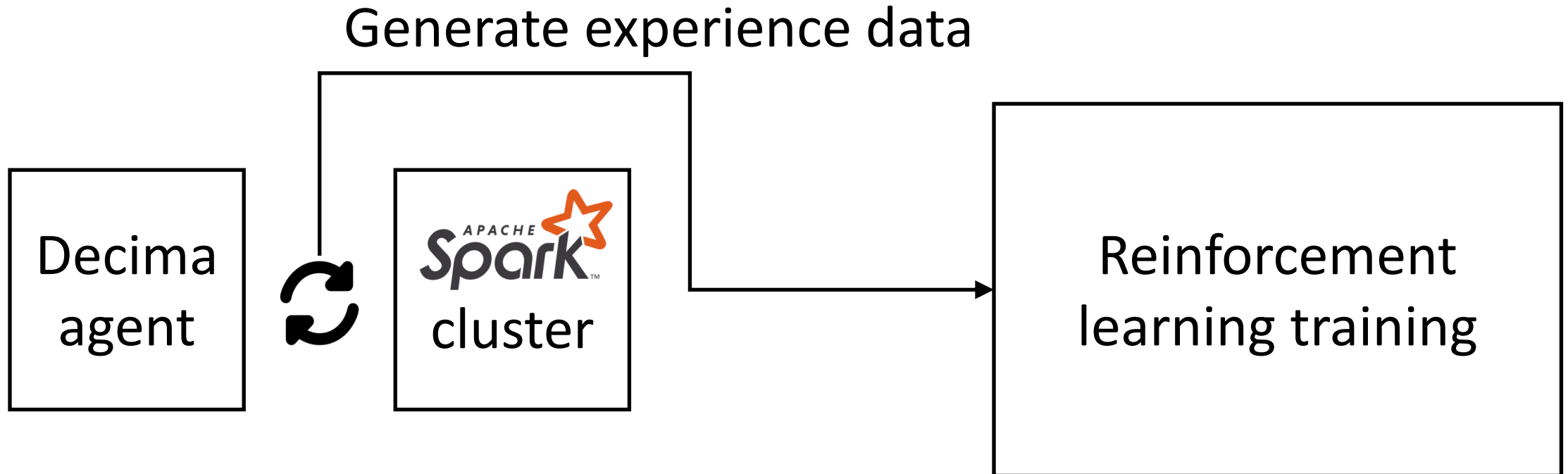
Graph Neural Network



$$\mathbf{e}_v = g \left[\sum_{w \in \xi(v)} f(\mathbf{e}_w) \right] + \mathbf{x}_v$$

Use supervised learning to verify a representation

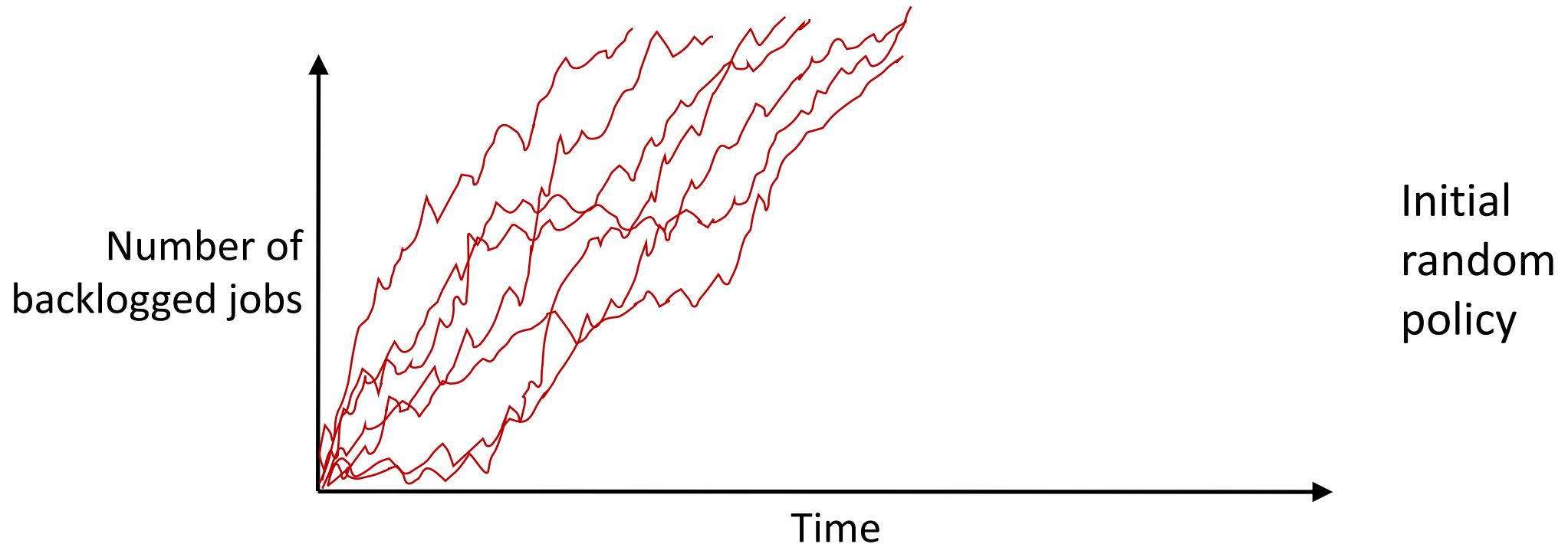
Training



Handle Online Job Arrival

The RL agent has to experience **continuous job arrival** during **training**.

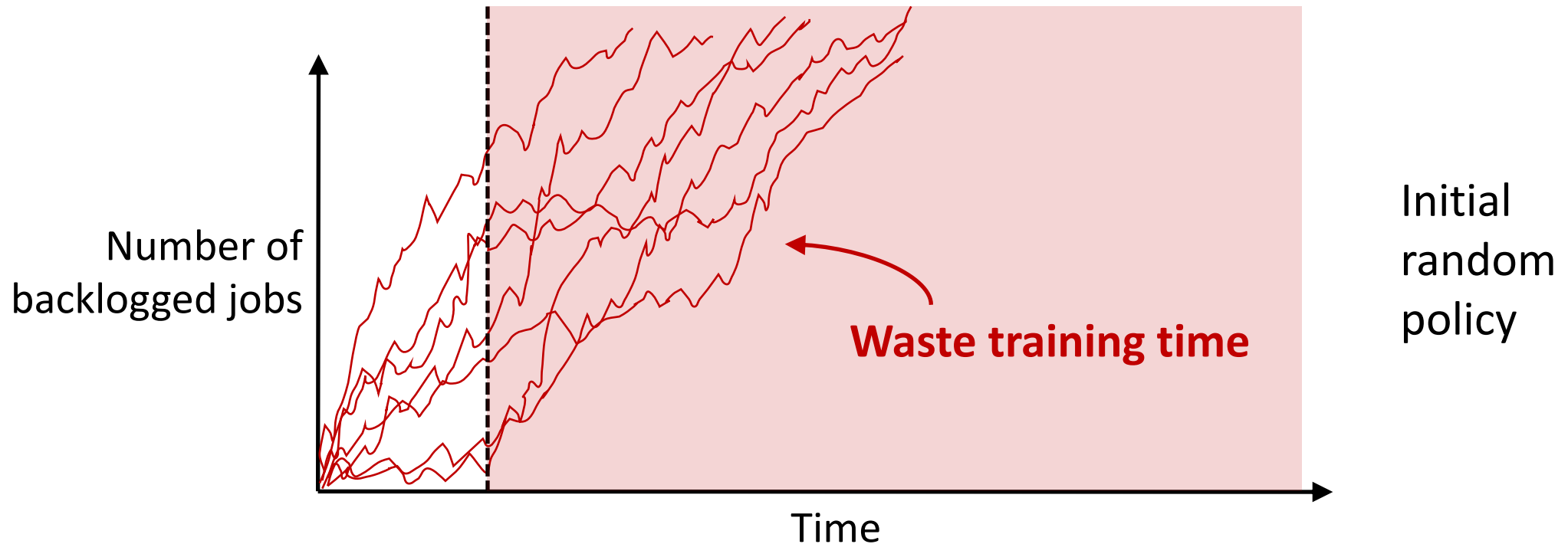
→ inefficient if simply feeding long sequences of jobs



Handle Online Job Arrival

The RL agent has to experience **continuous job arrival** during **training**.

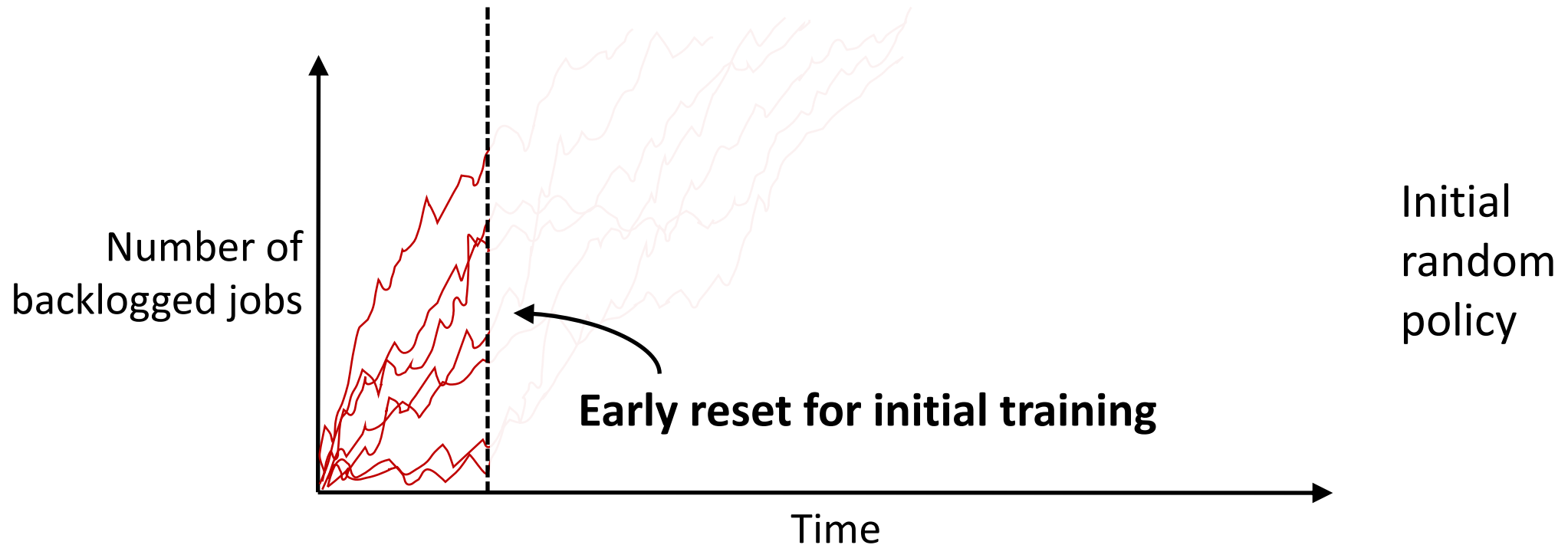
→ inefficient if simply feeding long sequences of jobs



Handle Online Job Arrival

The RL agent has to experience **continuous job arrival** during **training**.

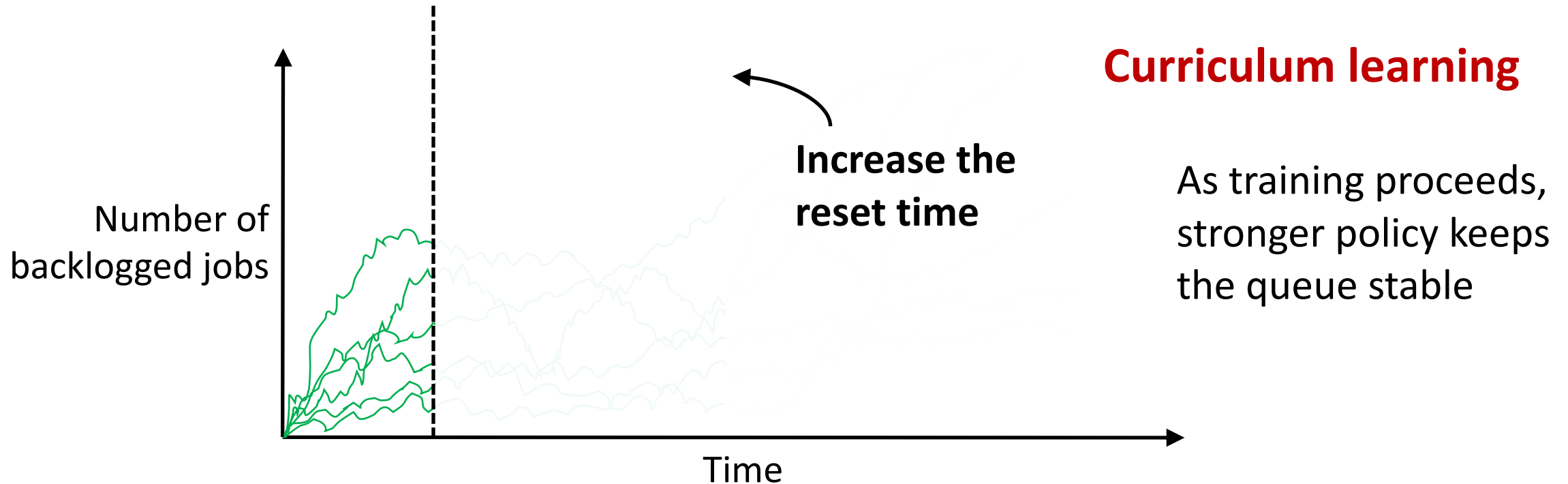
→ inefficient if simply feeding long sequences of jobs



Handle Online Job Arrival

The RL agent has to experience **continuous job arrival** during **training**.

→ inefficient if simply feeding long sequences of jobs

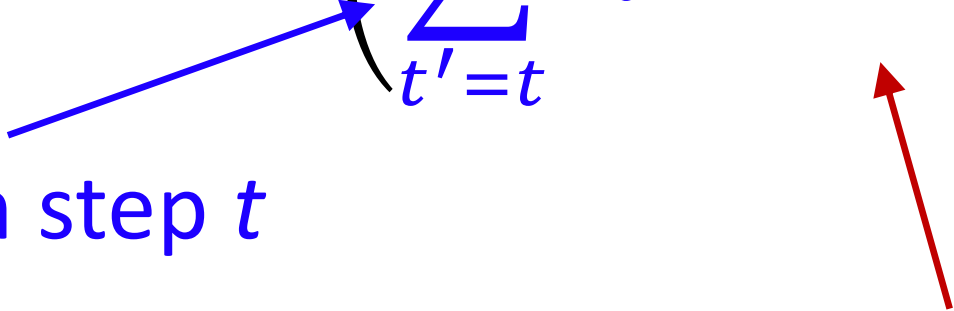


Variance from Job Sequences

RL agent needs to be **robust** to the **variation** in job arrival patterns.

→ huge variance can throw off the training process

Review: Policy Gradient RL Methods

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) \left(\sum_{t'=t}^T r_{t'} - b(s_t) \right)$$


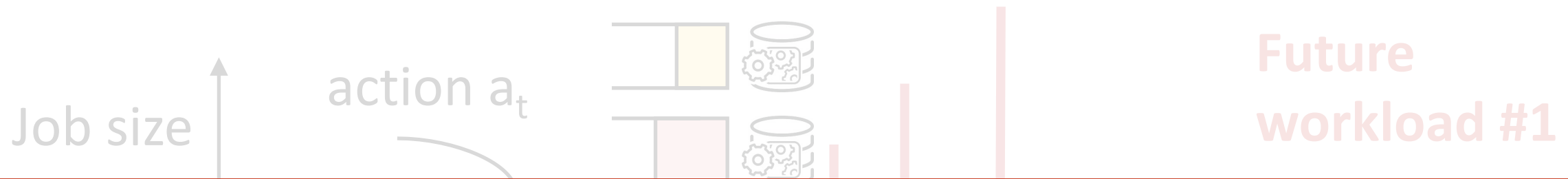
“return” from step t

“baseline”

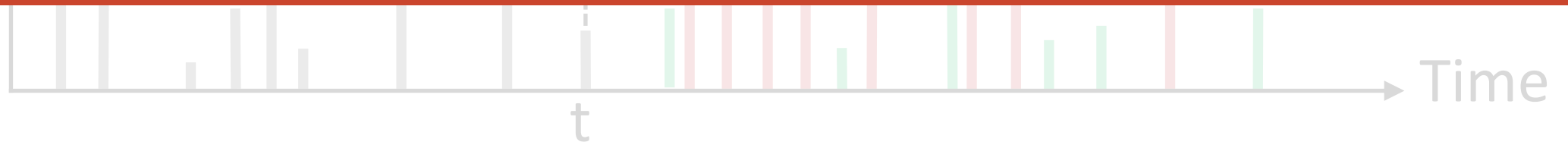
Increase probability of actions with
better-than-expected returns

Expected return
from state s_t

Variance from Job Sequences



Must consider the entire job sequence to score actions



$$\begin{aligned}\text{Score for action } a_t &= (\text{return after } a_t) - (\text{average return}) \\ &= \sum_{t'=t}^T r_{t'} - b(s_t)\end{aligned}$$

Input-Dependent Baseline

$$\text{Score for action } a_t = \sum_{t'=t}^T r_{t'} - b(s_t, z_t, z_{t+1}, \dots)$$



Average return for trajectories from state s_t
with job sequence z_t, z_{t+1}, \dots

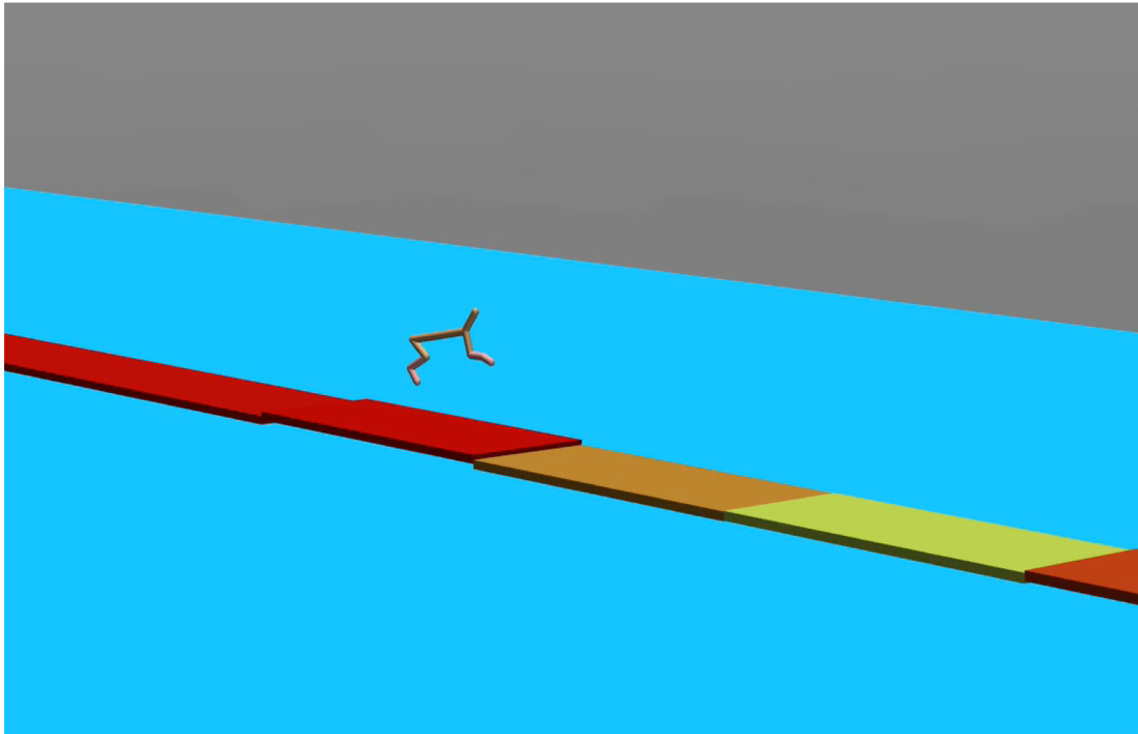
Theorem: Input-dependent baselines reduce variance without adding bias

$$\mathbb{E}[\nabla \log \pi_{\theta}(s_t, a_t) b(s_t, z_t, z_{t+1}, \dots)] = 0$$

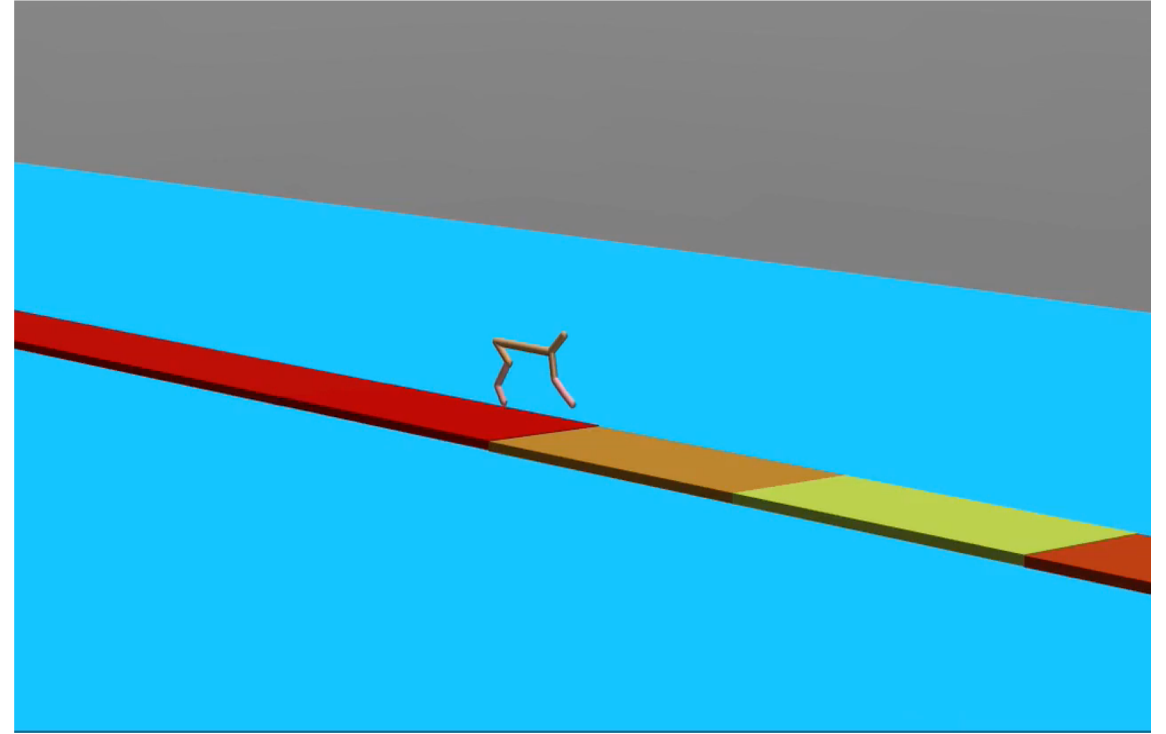
Input-Dependent Baseline

Broadly applicable to other systems with external input process:

Adaptive video streaming, load balancing, caching, robotics with disturbance...



Train with standard baseline



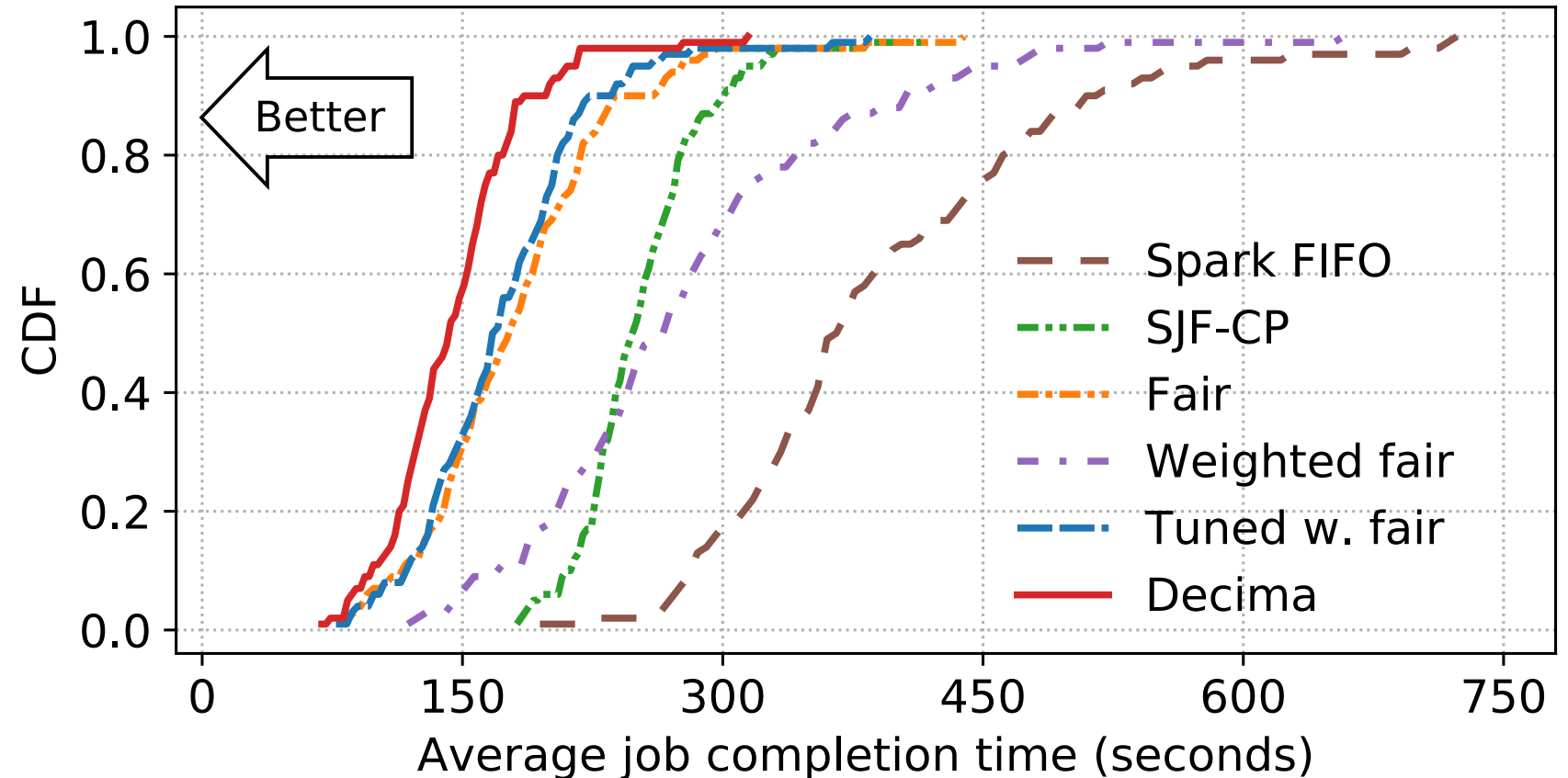
Train with input-dependent baseline

- **Variance reduction for reinforcement learning in input-driven environments.** Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, Mohammad Alizadeh. *International Conference on Learning Representations (ICLR)*, 2019.

Evaluation

Decima vs. Baselines: Batched Arrivals

- 20 TPC-H queries sampled at random; input sizes: 2, 5, 10, 20, 50, 100 GB
- Decima trained on simulator; tested on real Spark cluster

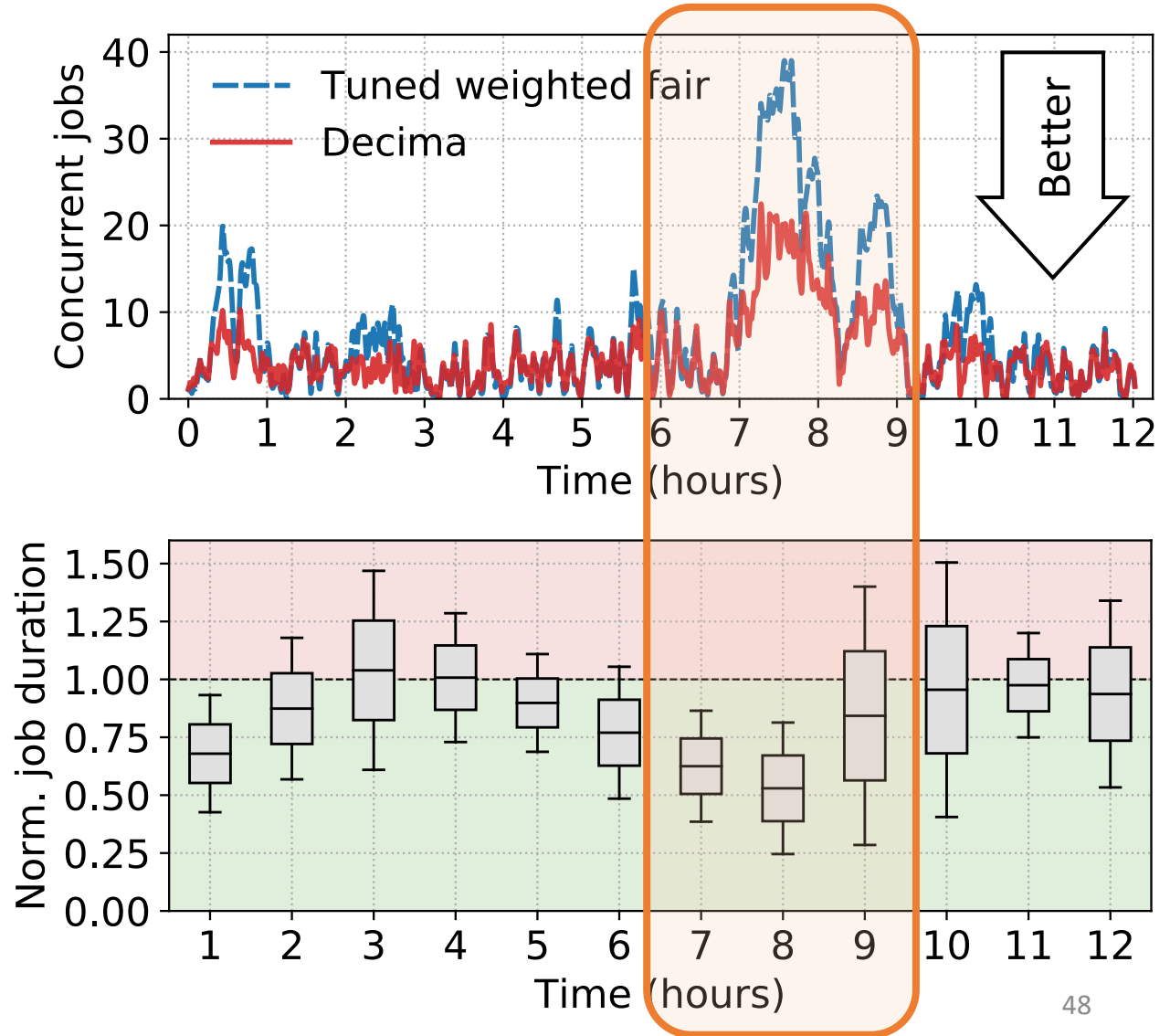


Decima improves average job completion time by 21%-3.1x over baseline schemes

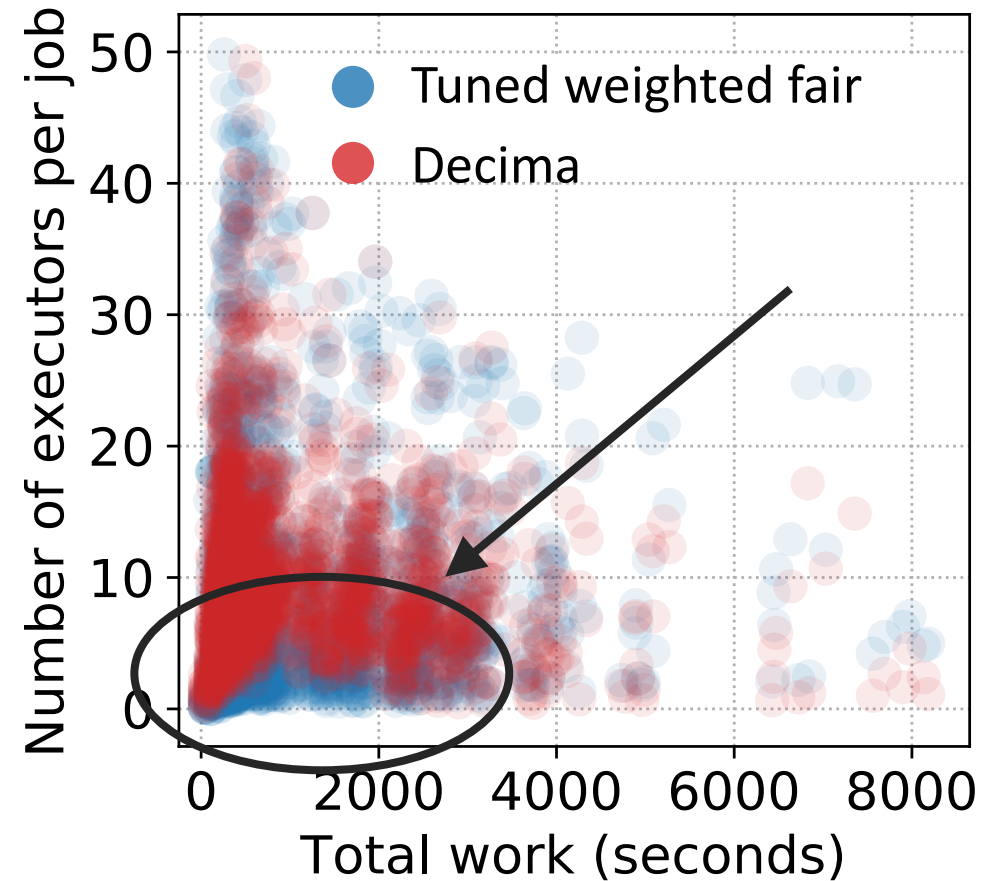
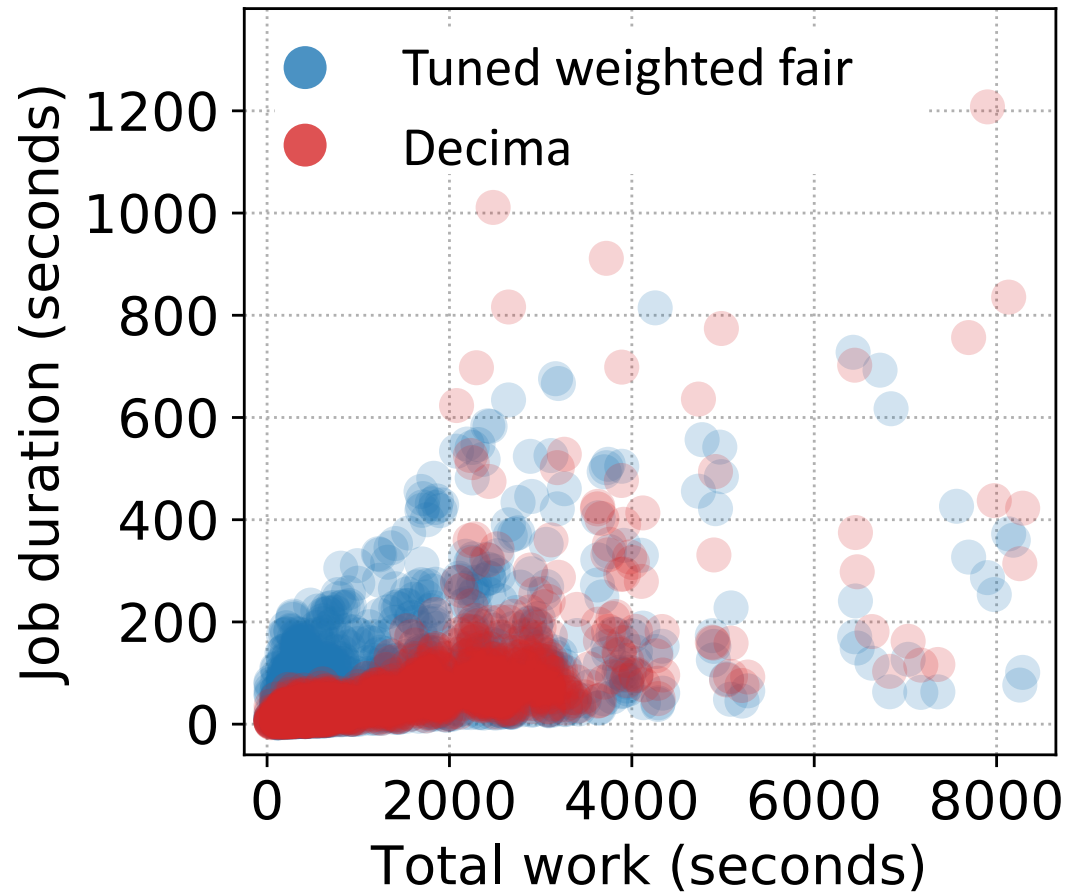
Decima with Continuous Job Arrivals

1000 jobs arrives as a Poisson process with avg. inter-arrival time = 25 sec

Decima achieves 28% lower average JCT than best heuristic, and 2X better JCT in overload



Understanding Decima



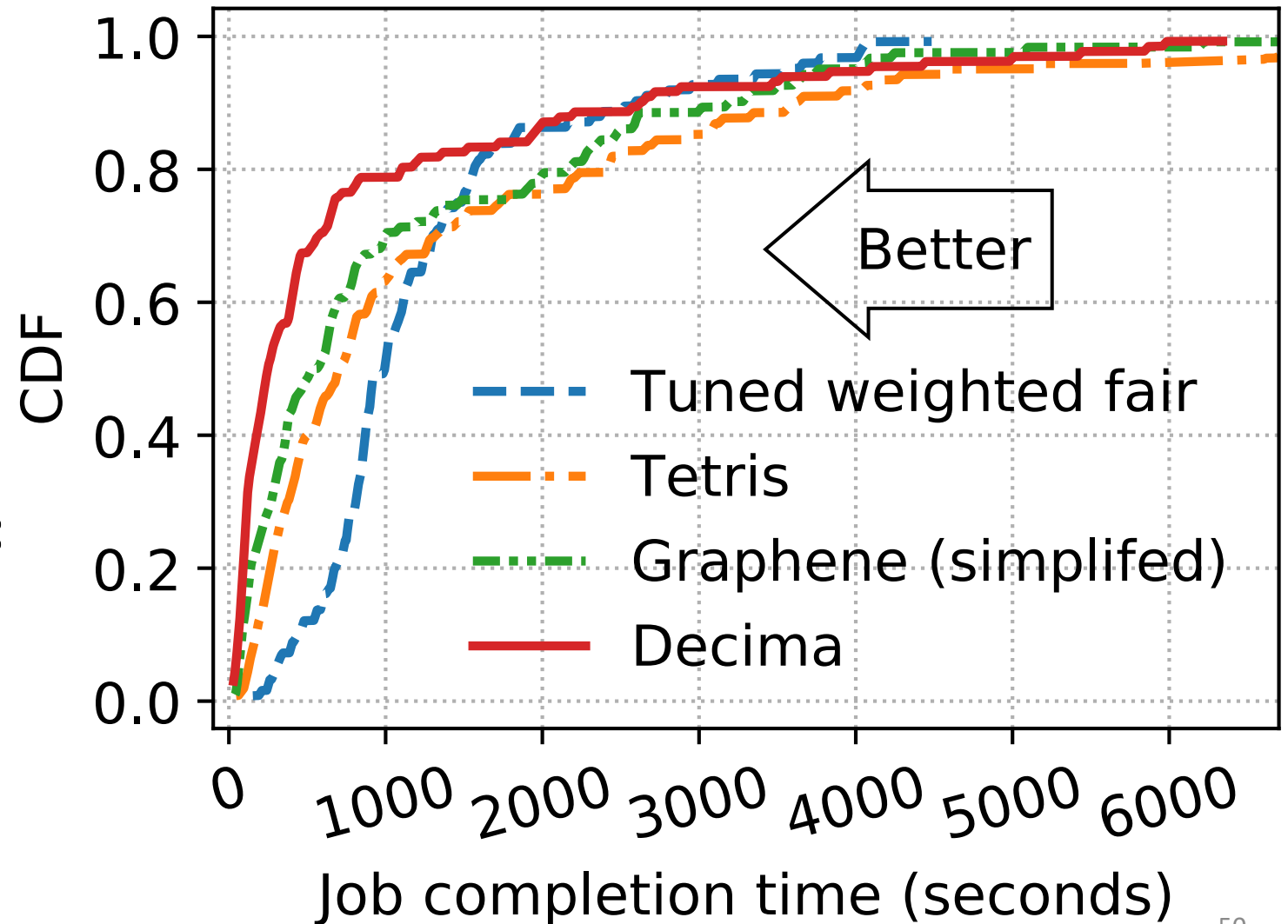
Flexibility: Multi-Resource Scheduling

Industrial trace (Alibaba):

20,000 jobs from
production cluster

Multi-resource requirement:

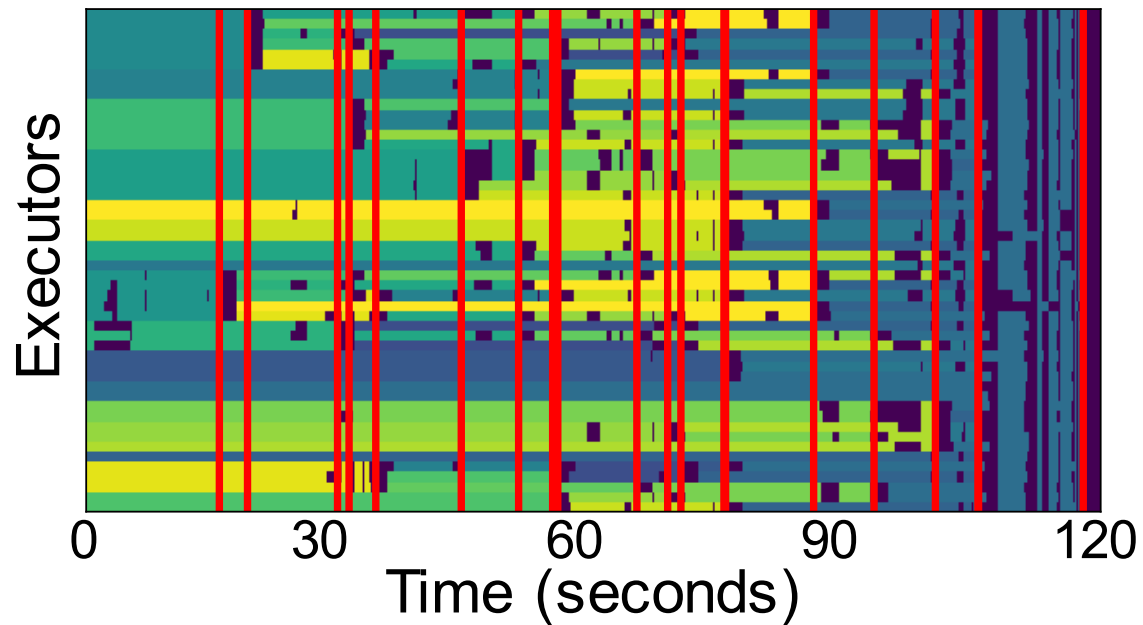
CPU cores + memory units



Flexibility: Different Objectives & Systems

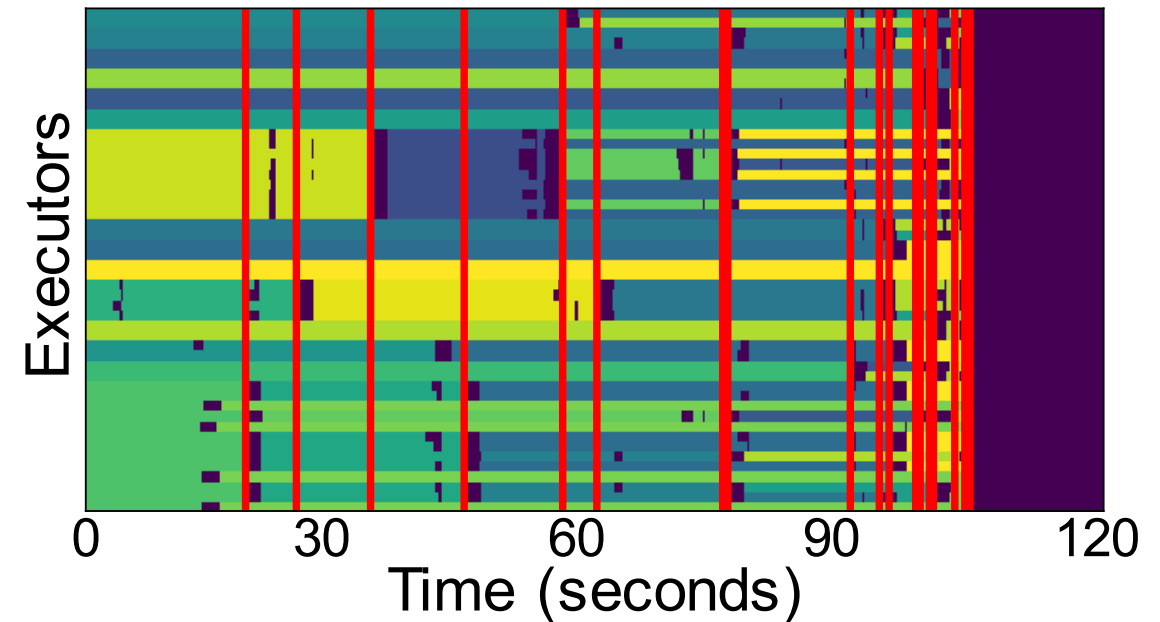
Objective: Avg JCT

Avg. JCT 67.3 sec, makespan 119.6 sec



Objective: Makespan

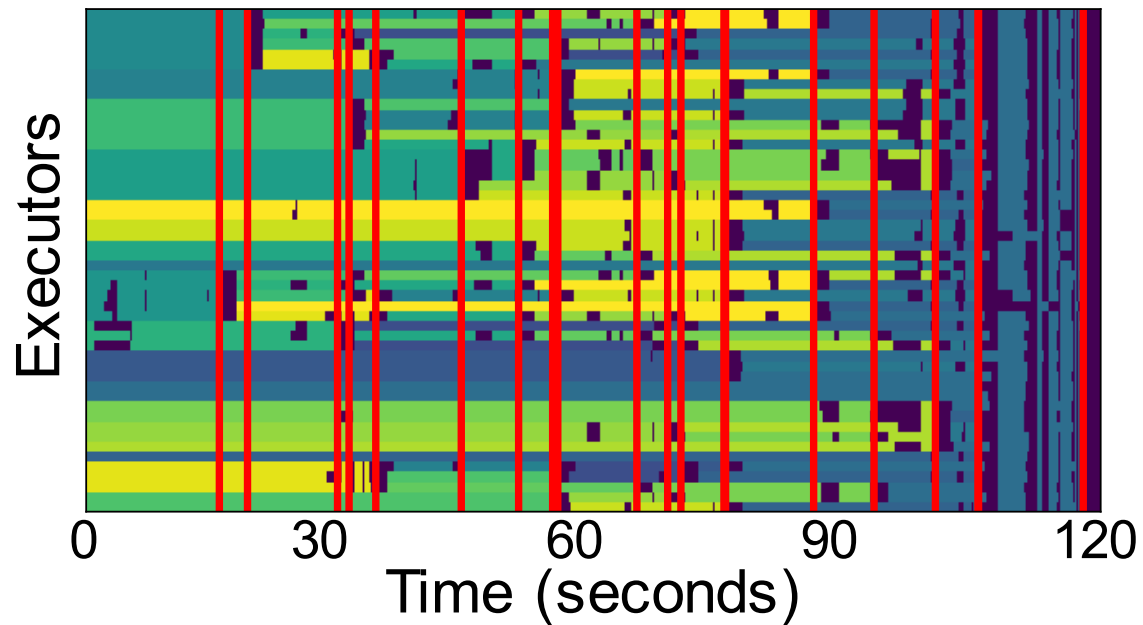
Avg. JCT 74.5 sec, makespan 102.1 sec



Flexibility: Different Objectives & Systems

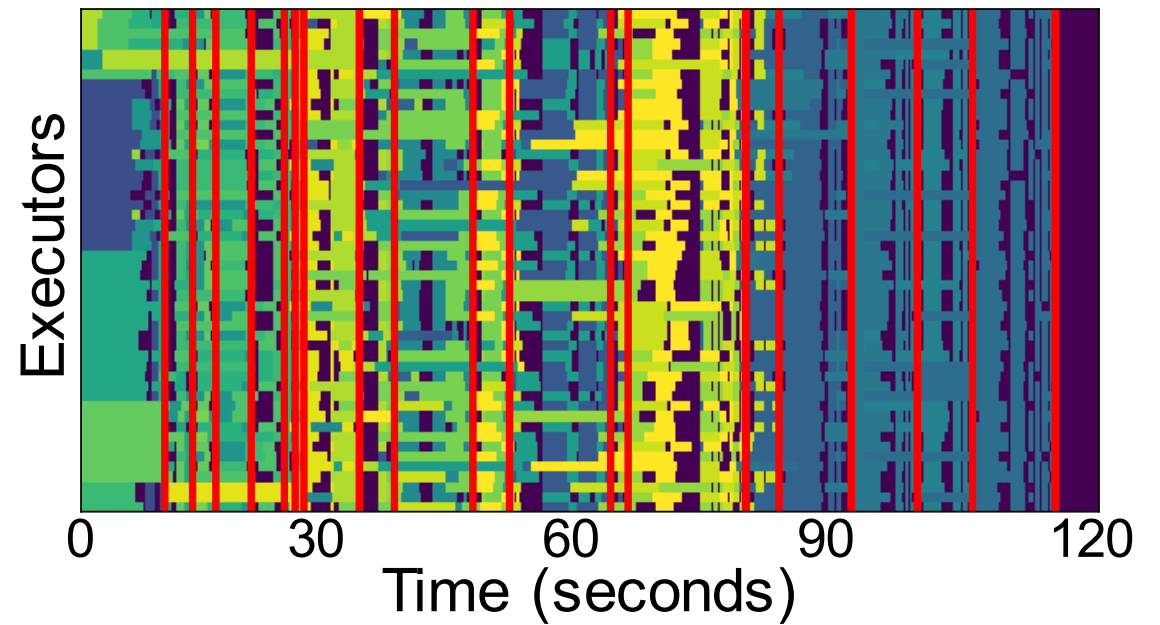
Objective: Avg JCT

Avg. JCT 67.3 sec, makespan 119.6 sec



Objective: Avg JCT zero-cost migration

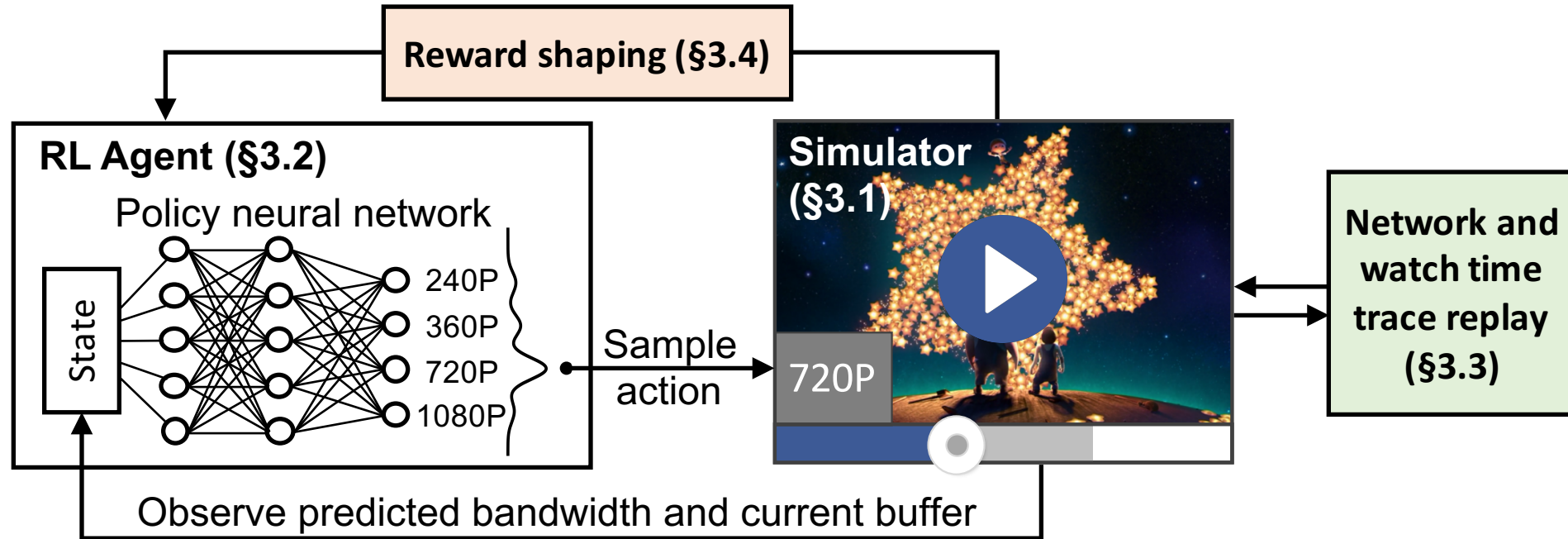
Avg. JCT 61.4 sec, makespan 114.3 sec



Other Evaluations

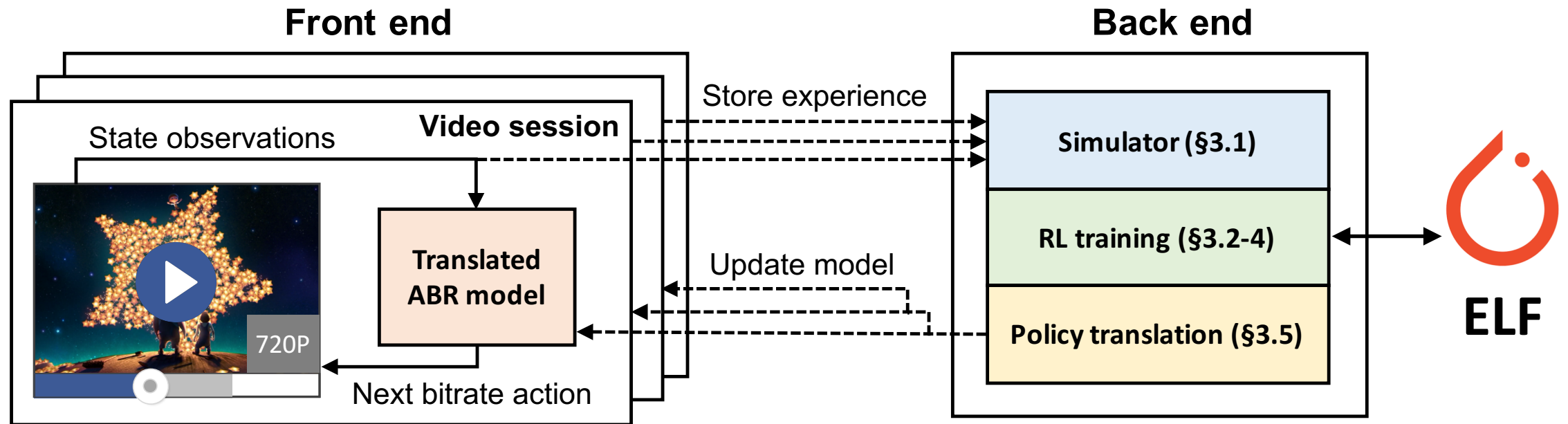
- Impact of each component in the learning algorithm
- Generalization to different workloads
- Training and inference speed
- Handling missing features
- Optimality gap

Real-world Video Bitrate Adaptation with RL



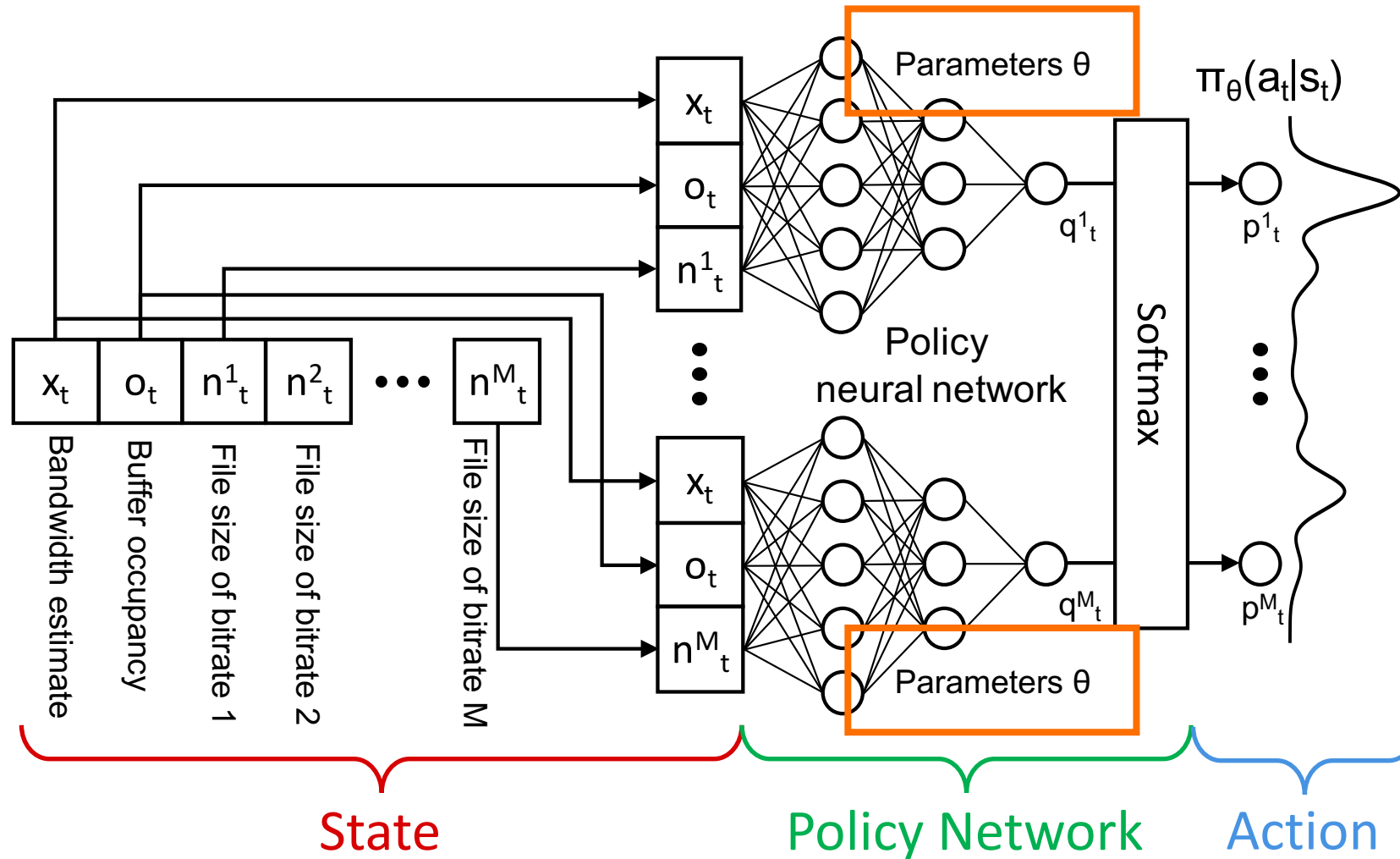
- **Real-world Video Adaptation with Reinforcement Learning.** Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, Eytan Bakshy. *ICML Workshop*, 2019.

Real-world Video Bitrate Adaptation with RL



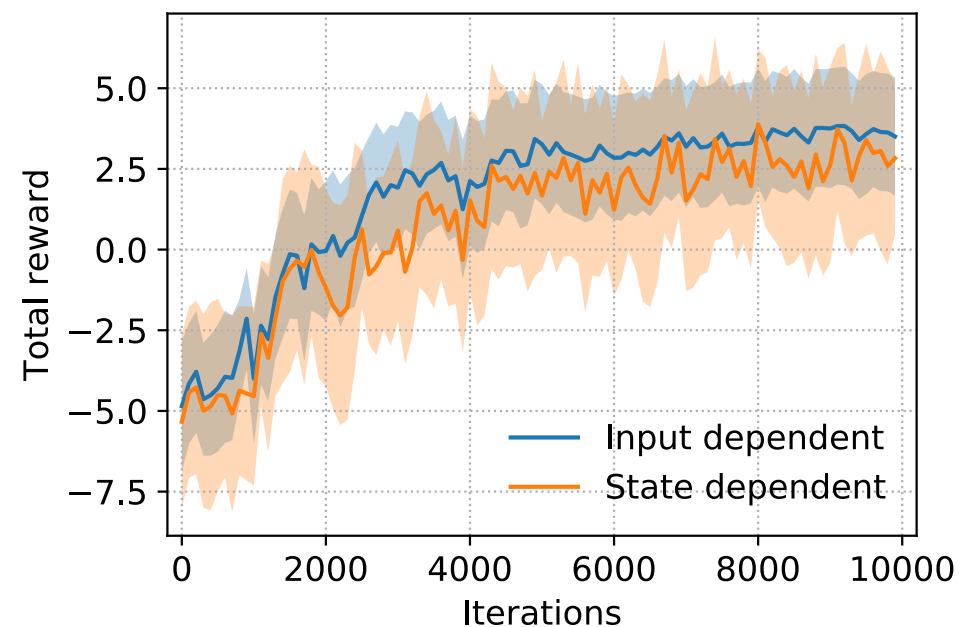
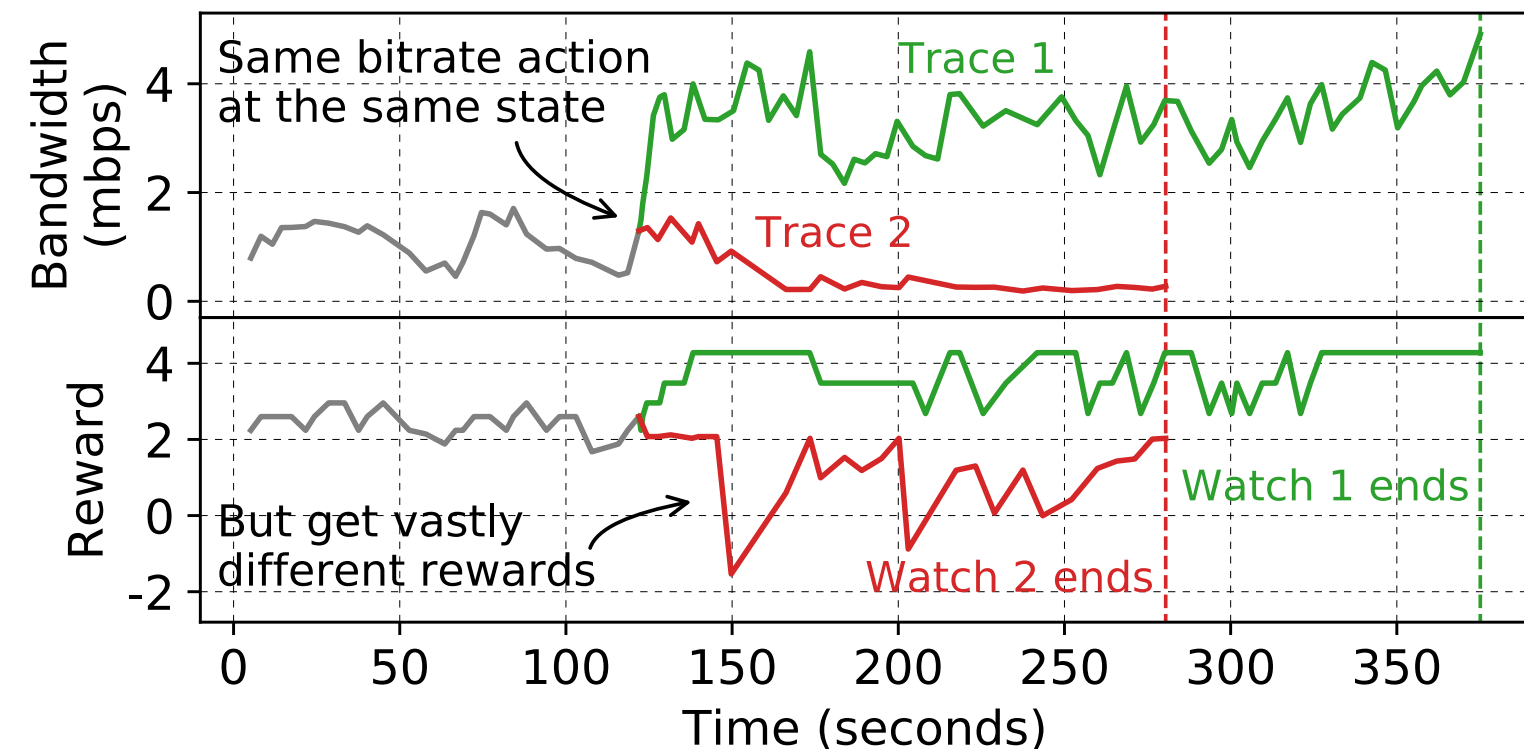
- **Real-world Video Adaptation with Reinforcement Learning.** Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, Eytan Bakshy. *ICML Workshop*, 2019.

Real-world Video Bitrate Adaptation with RL



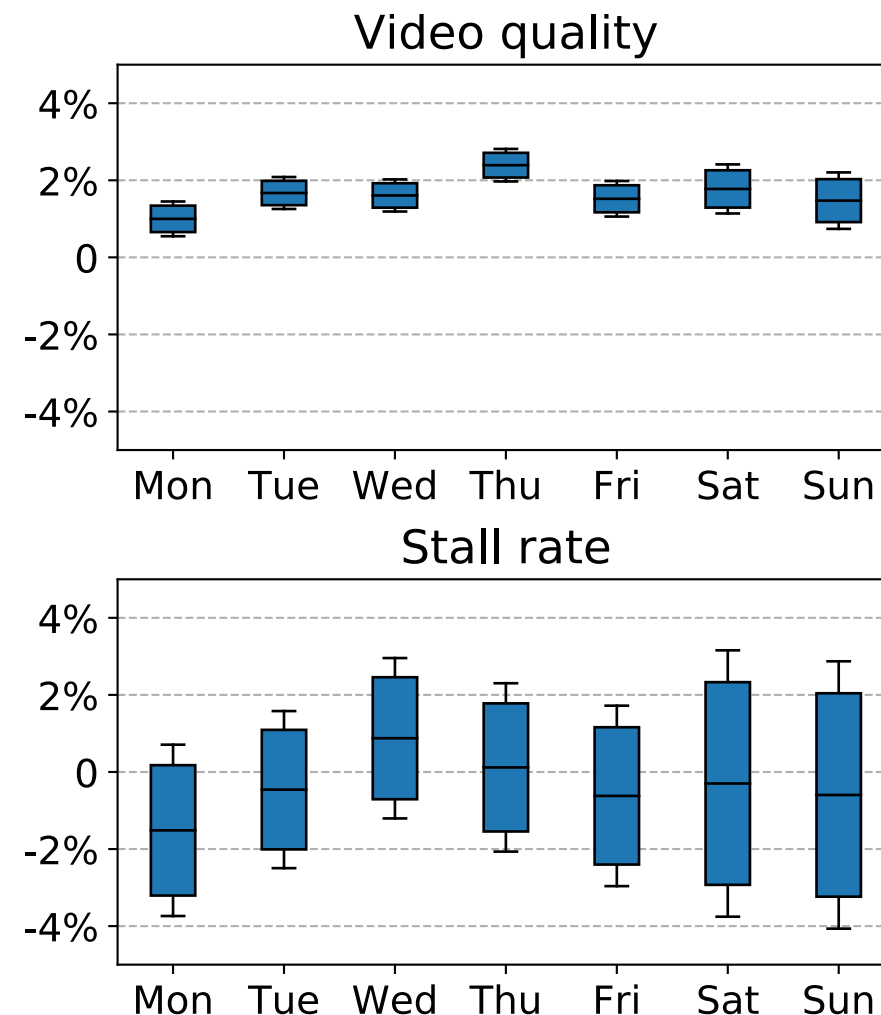
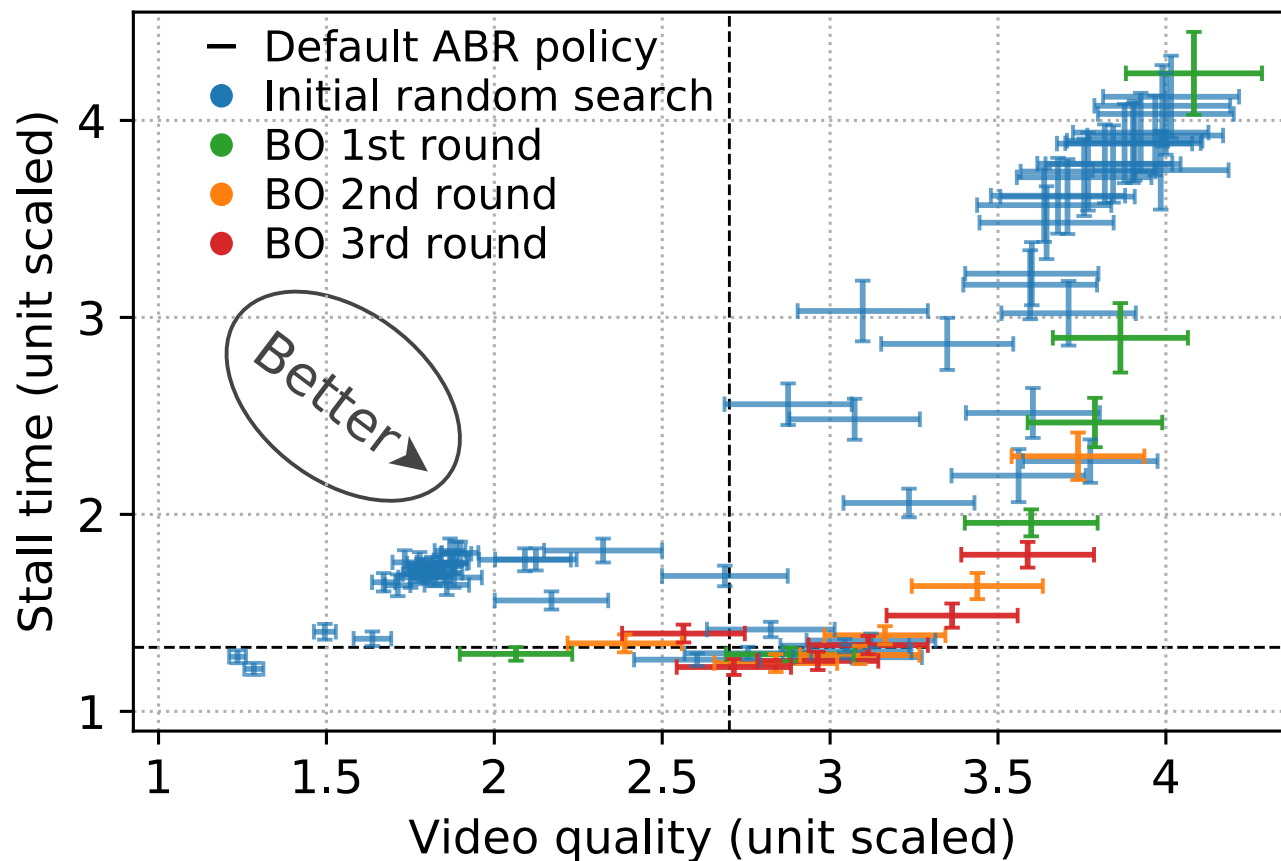
- **Real-world Video Adaptation with Reinforcement Learning.** Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, Eytan Bakshy. *ICML Workshop*, 2019.

Real-world Video Bitrate Adaptation with RL



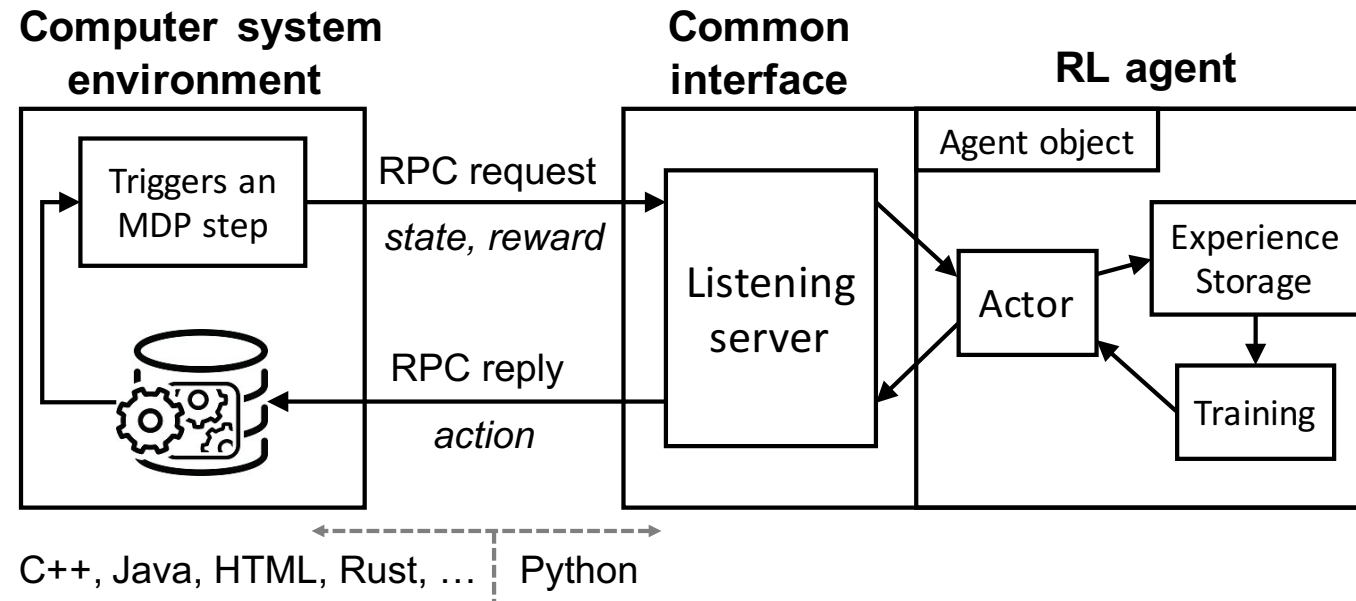
- **Real-world Video Adaptation with Reinforcement Learning.** Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, Eytan Bakshy. *ICML Workshop*, 2019.

Real-world Video Bitrate Adaptation with RL



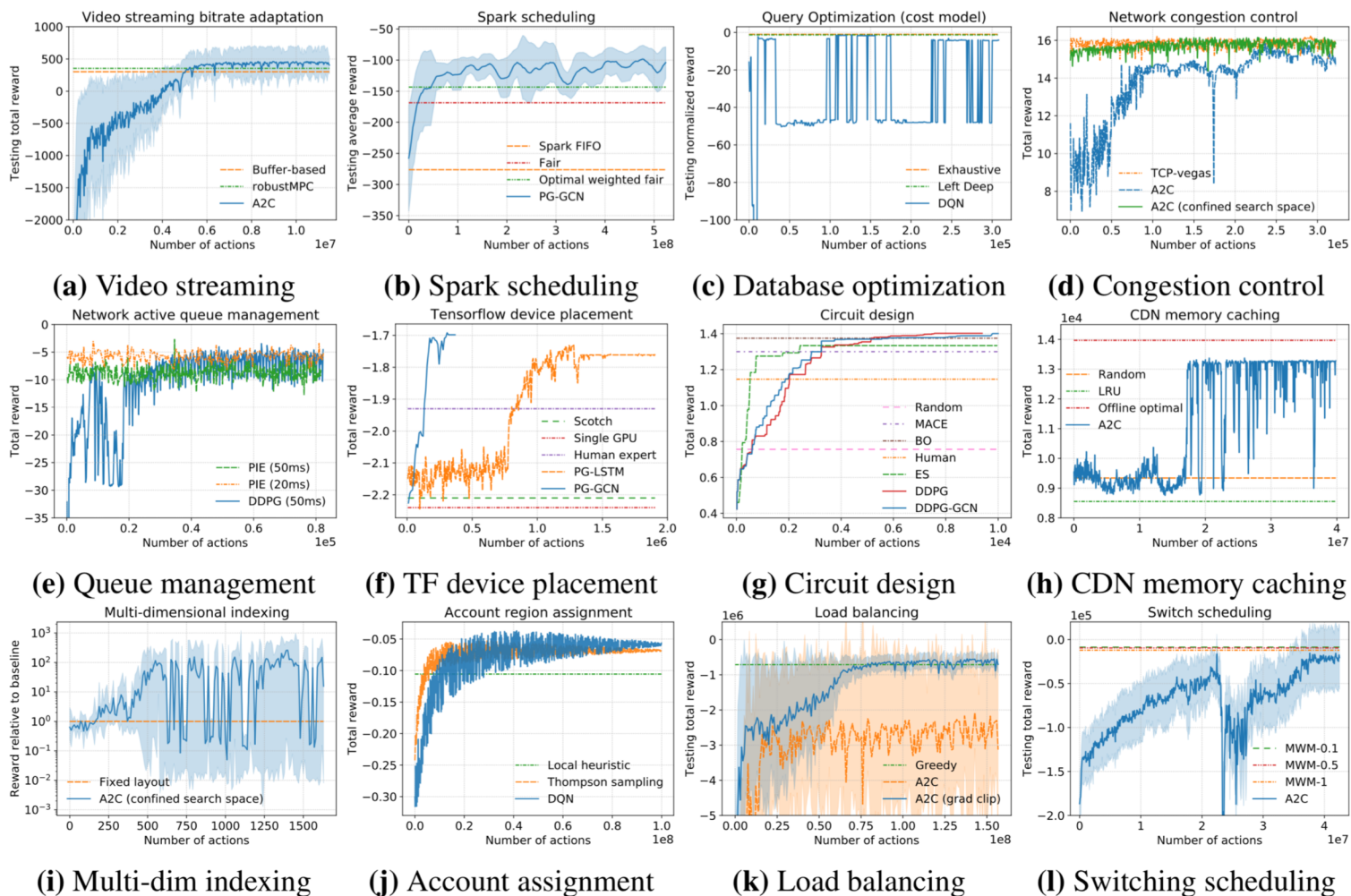
- **Real-world Video Adaptation with Reinforcement Learning.** Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, Eytan Bakshy. *ICML Workshop*, 2019.

Park: An Open Platform for Learning-Augmented Systems



- 12 system environments for networking, databases, distributed systems, ...
- Contains real system and simulation
- Interact with the system with a standard API

Park: An Open Platform for Learning-Augmented Systems



- **Park: An Open Platform for Learning-Augmented Computer Systems.** H. Mao, P. Negi, A. Narayan, H. Wang, J. Yang, H. Wang, R. Marcus, R. Addanki, M. Khani, S. He, V. Nathan, F. Cangialosi, S. Venkatakrisnan, W. Weng, S. Han, T. Kraska, M. Alizadeh. *Neural Information Processing Systems (NeurIPS)*, 2019.

Park: An Open Platform for Learning-Augmented Systems

Some example challenges:

- Infinite horizon
- Representation of the states and actions
- Simulation-reality gap
- Needle-in-the-haystack problem
- ...

Summary

- Decima develops new RL algorithms to learn workload-specific cluster scheduling algorithms
<http://web.mit.edu/decima/>
- ABRL conducts large-scale production experiment on applying RL to video bitrate adaptation
<https://openreview.net/forum?id=SJlCkwN8iV>
- Park open sources a platform for RL research in computer systems
<https://github.com/park-project/park>