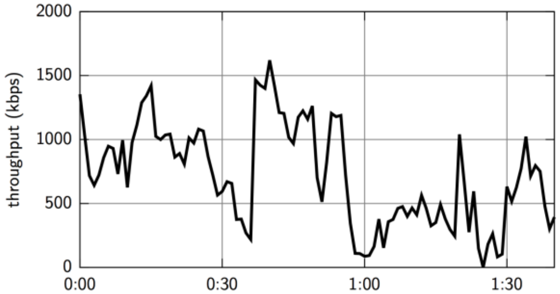
Towards Network Systems that Improve with Experience

Mohammad Alizadeh

CSAIL

Network Control Systems



Congestion Control

Traffic Engineering

Cluster Scheduling

Adaptive Video Streaming

Internet Telephony

……

Two Paradigms in Network Control

Classical Paradigm (1960 – now)
[Specify & Build]

1. Specify operating environment & low-level design goals

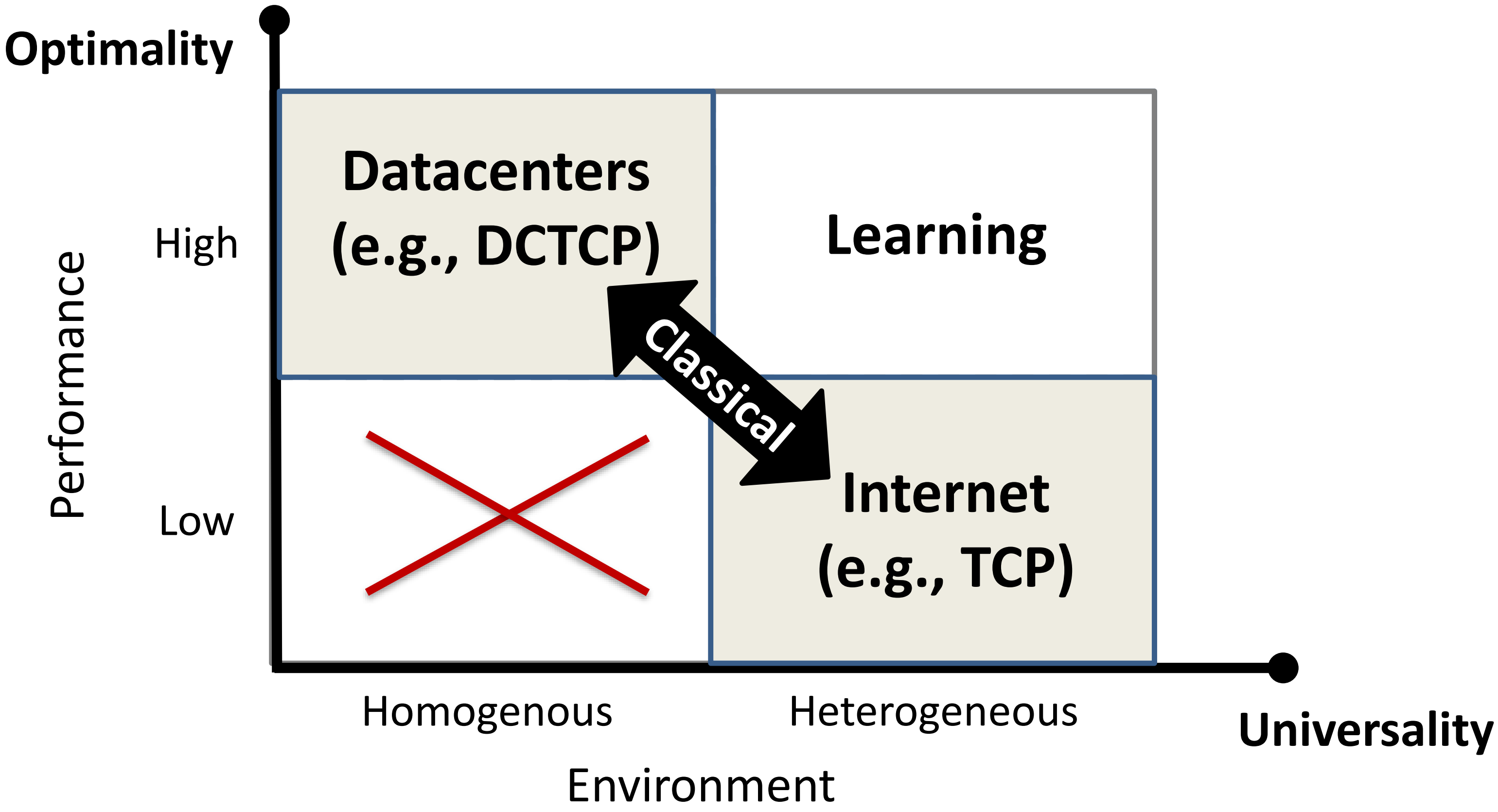
2. Build one algorithm that achieves goals in (most) cases of interest

Learning Paradigm
[Learn & Adapt]

1. Learn operating environment & low-level goals

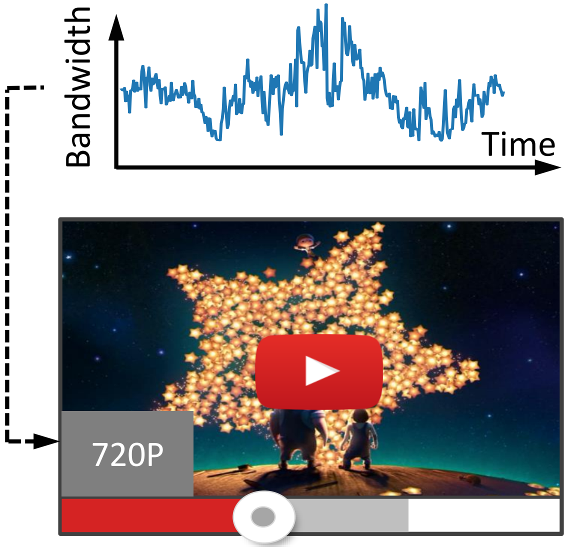
2. Learn algorithms that can adapt to network/workload conditions

Two Paradigms in Network Control

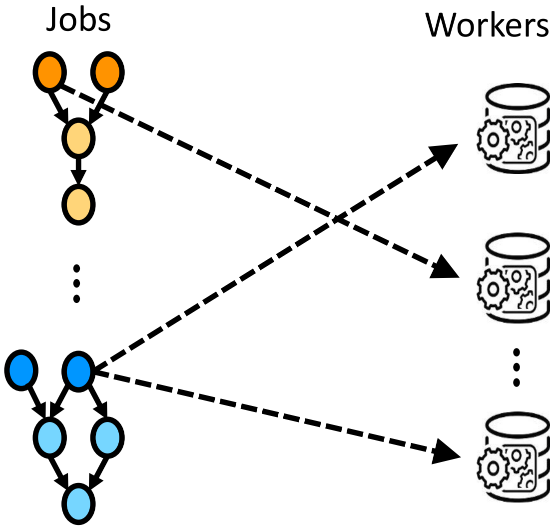


Our Work on Learning-Based Network Systems

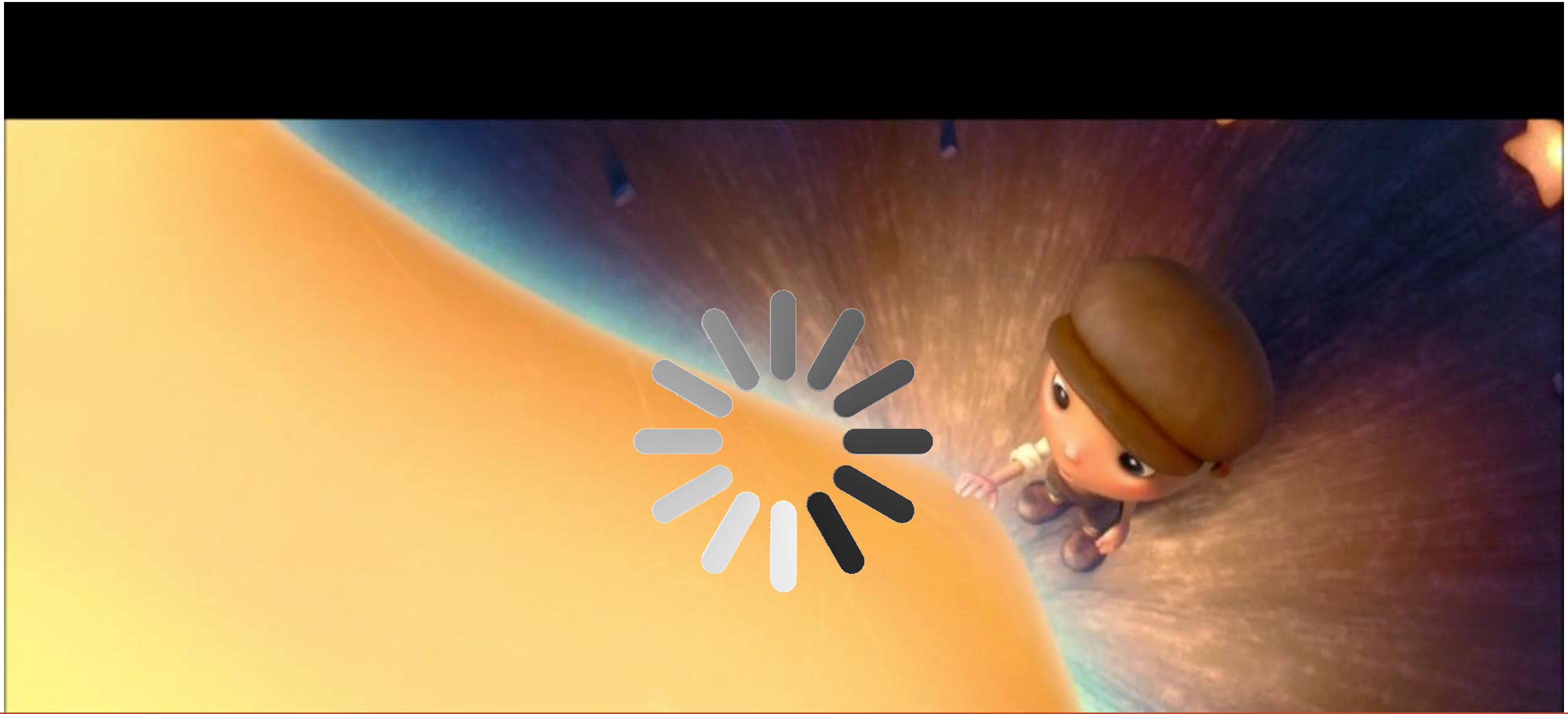
Adaptive video streaming
(Pensieve)



Cluster Scheduling
(Decima)



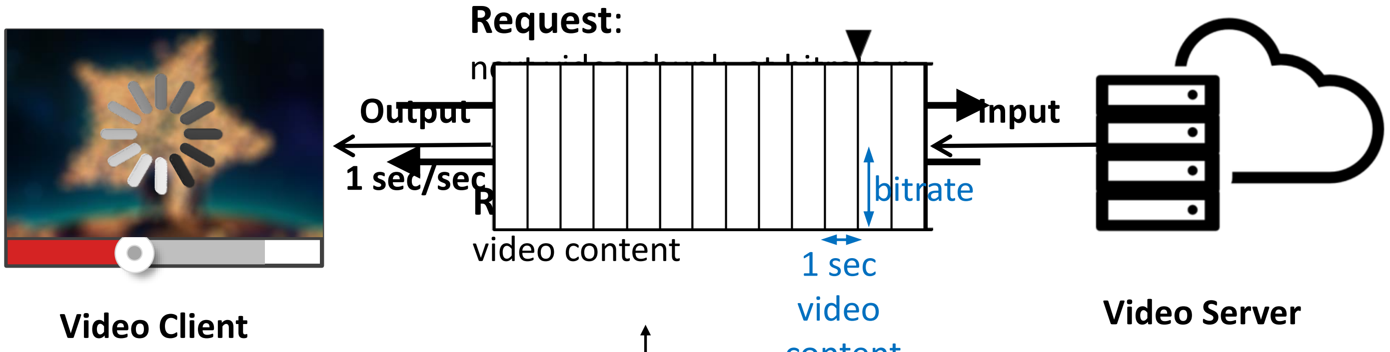
RL Techniques



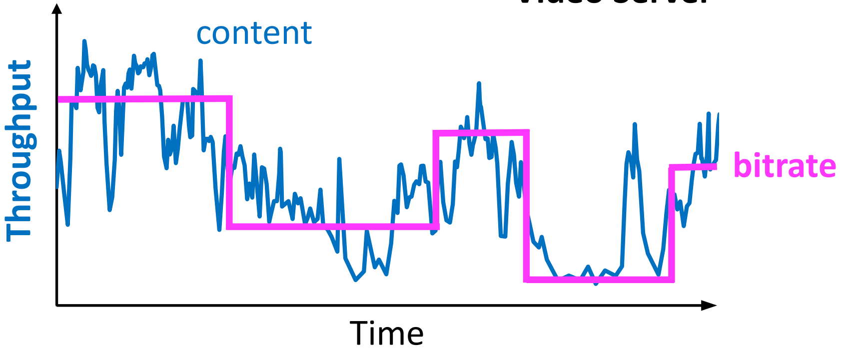
Users start leaving if video doesn't play in 2 seconds

https://gigaom.com/2012/11/09/online-viewers-start-leaving-if-video-doesnt-play-in-2-seconds-says-study/
Video: La Luna (Pixar 2011)

Dynamic Streaming over HTTP (DASH)

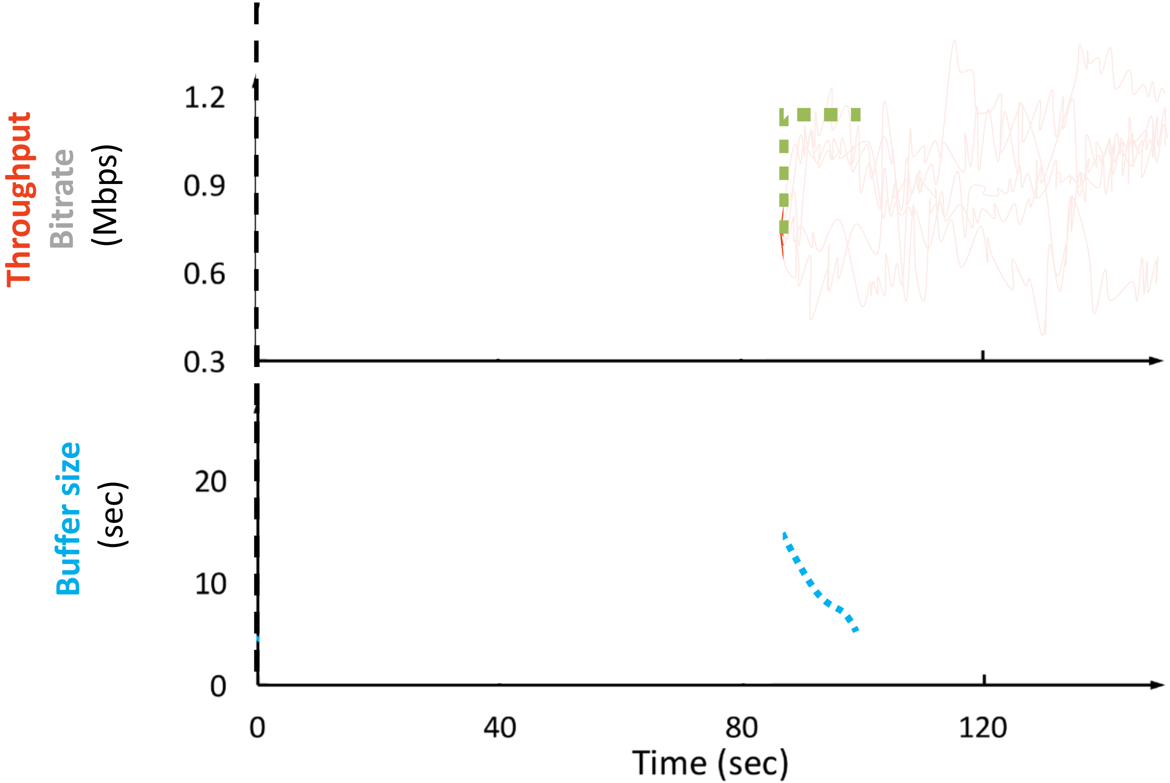


Adaptive Bitrate (ABR) Algorithms



Animation borrowed from Te-Yuan Huang (SIGCOMM '14) http://conferences.sigcomm.org/sigcomm/2014/doc/slides/38.pdf

Why is ABR Challenging?



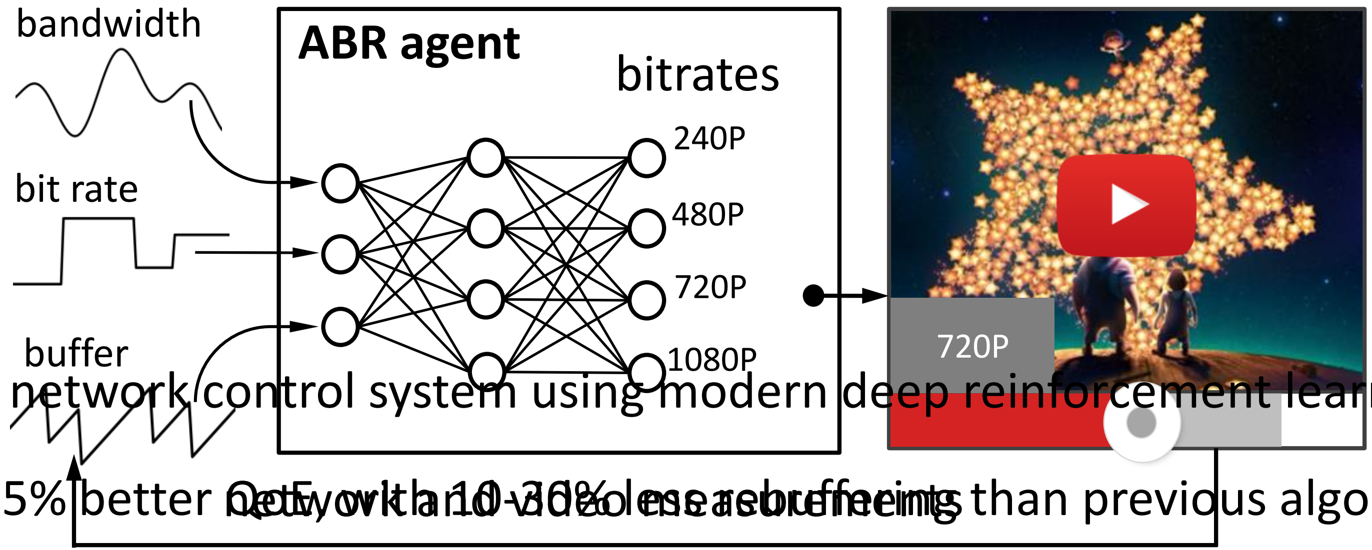
Network throughput is variable & uncertain

Conflicting QoE goals

- Bitrate
- Rebuffering time
- Smoothness

Cascading effects of decisions

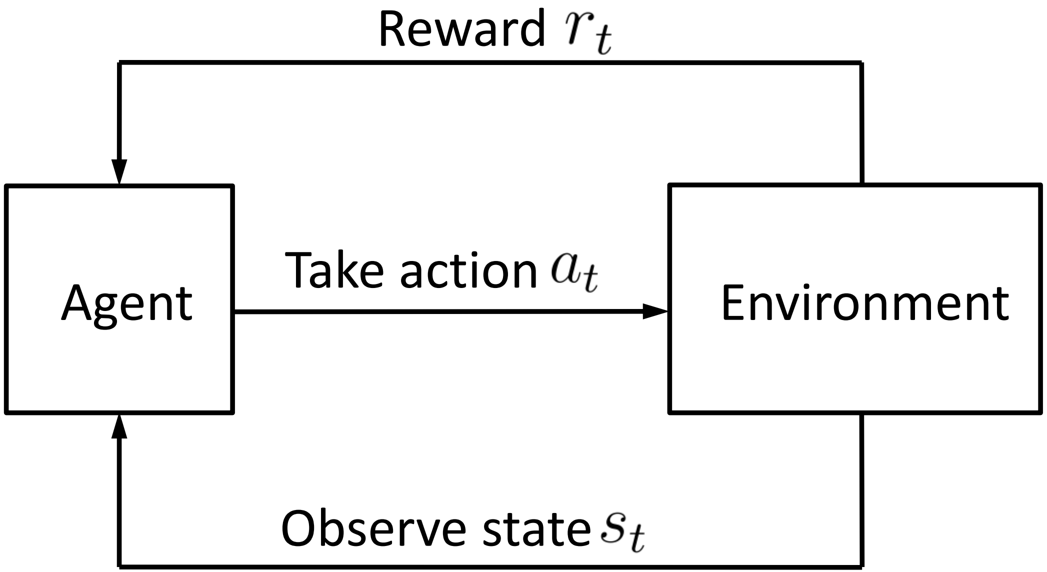
Our System: Pensieve



1. First network control system using modern deep reinforcement learning
2. 12-25% better QoE with 10-30% less rebuffering than previous algorithms
3. Tailors ABR decisions specifically for different network conditions

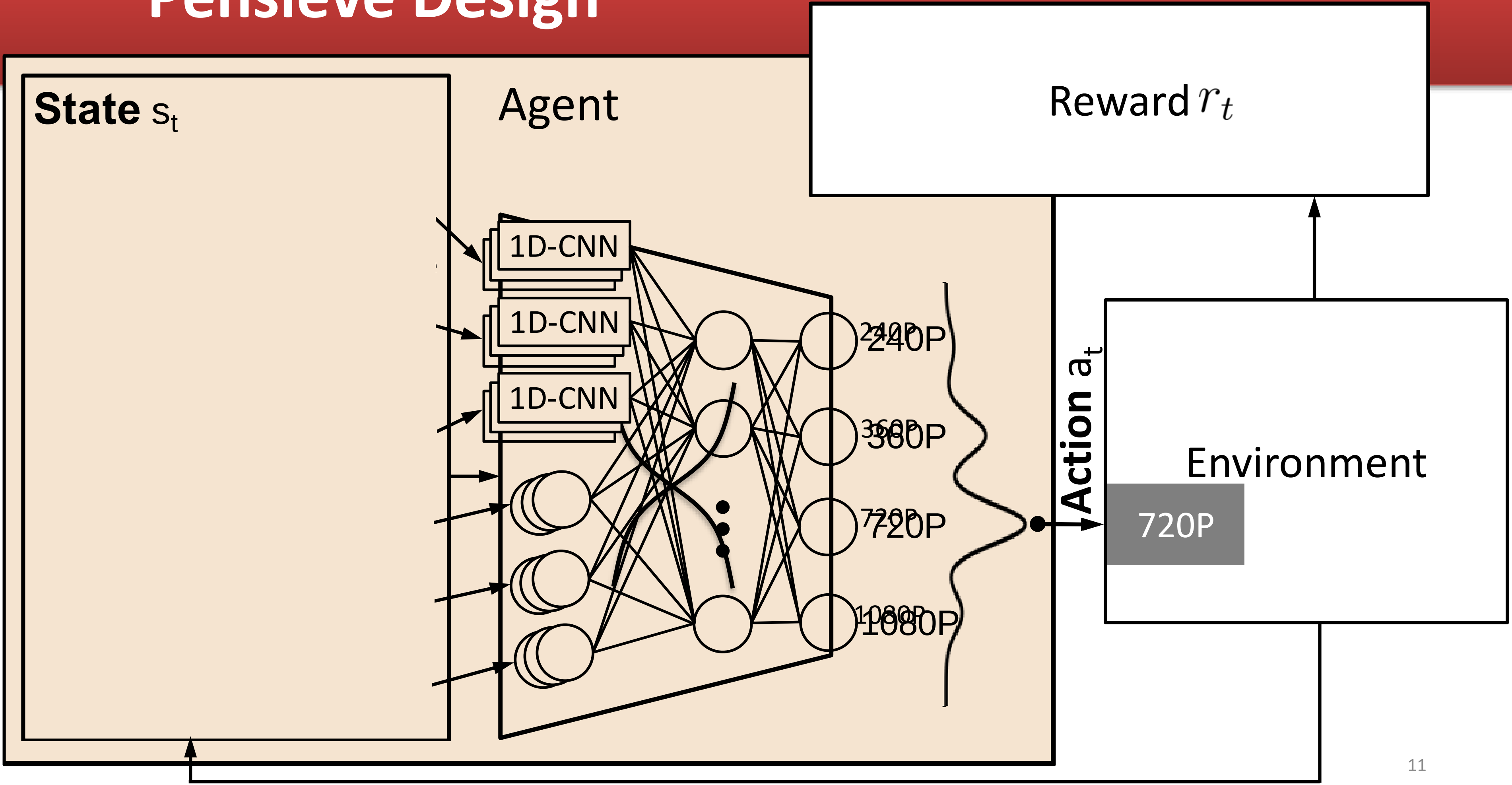
Pensieve learns ABR algorithms automatically through experience

Reinforcement Learning



Goal: maximize expected total future reward $\mathbb{E}\left[\sum_t r_t\right]$

Pensieve Design

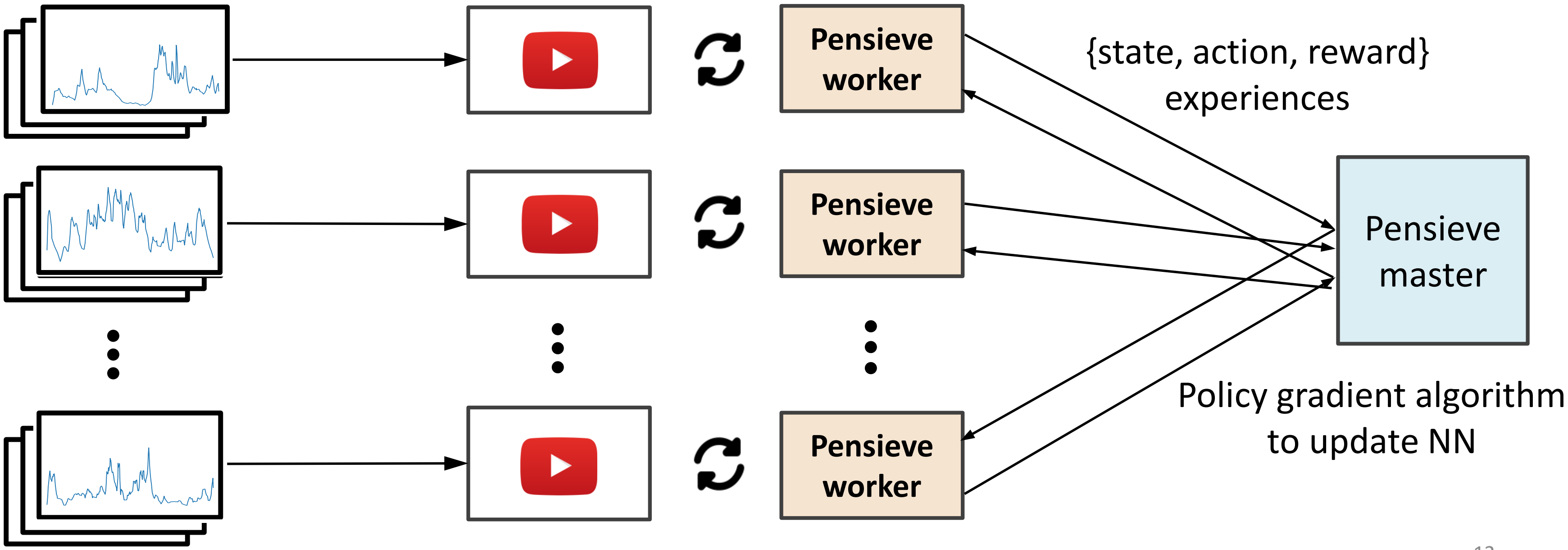


Pensieve Training System

Large corpus of network traces
cellular, broadband, synthetic

Video playback
Fast chunk-level simulator

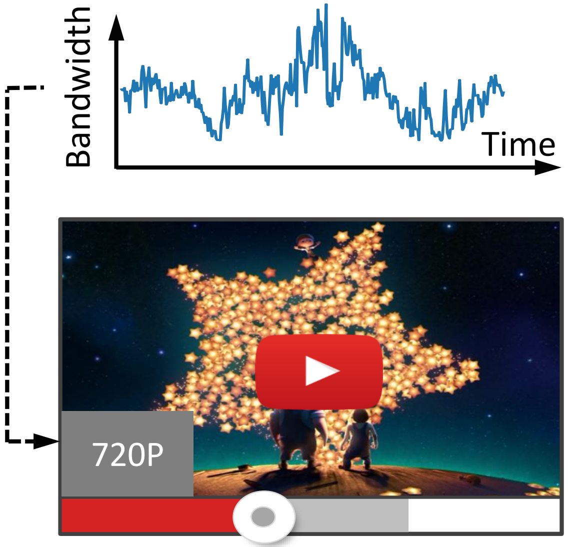
Model update



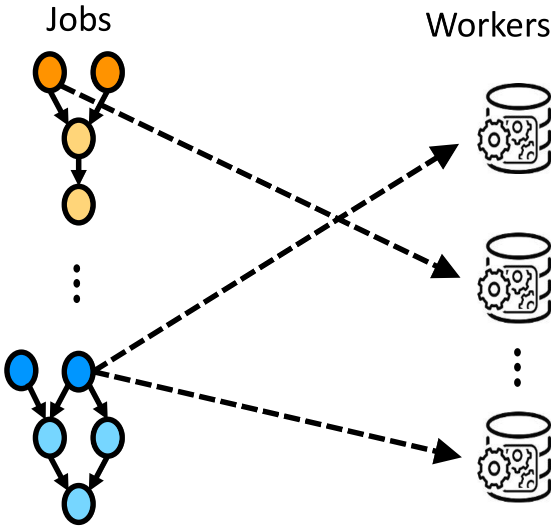
Pensieve
MPC
240P
240P
Pensieve
buffer (sec)
Throughput
(mbps)
MPC
buffer (sec)
chances of outage
Rebuffering
Time

Our Work on Learning-Based Network Systems

Adaptive video streaming
(Pensieve)



Cluster Scheduling
(Decima)

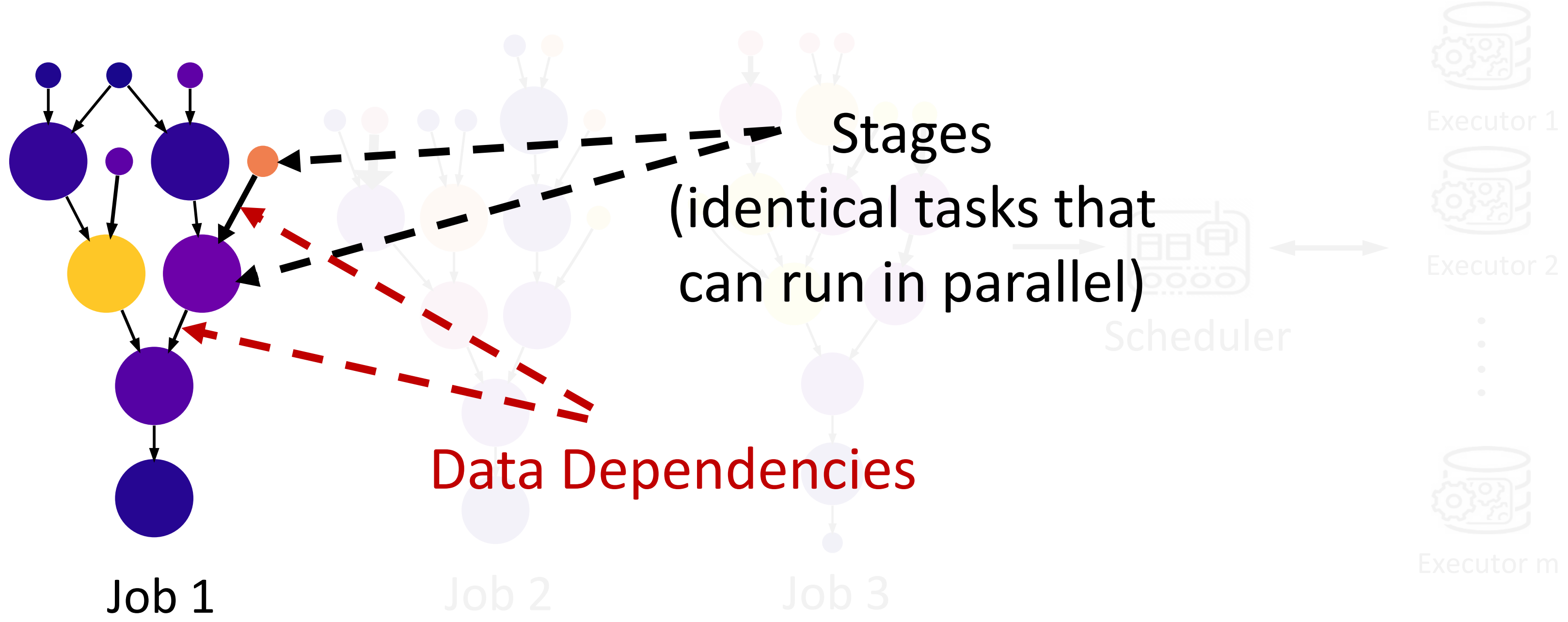


RL Techniques

Scheduling Graph-Structured Workloads

Many systems encode jobs as directed acyclic graphs (DAGs)

- Data processing (e.g., Spark, Hadoop), ML training (e.g., TensorFlow), ...



Designing Optimal Schedulers is Intractable

A lot of considerations for optimal performance:

- Job dependency structure
- Degree of parallelism
- Scheduling order
- Locality
- …

No "one-size-fits-all" solution:

Best algorithm depends on specific workload and system

Decima: Technical Challenges

Challenge: Huge state and action space

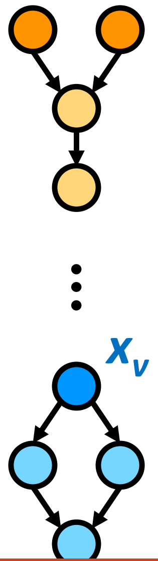
→ Scalable Graph CNN to process any number of job DAGs

Challenge: Variance caused by stochastic job arrival process

→ Variance reduction technique for RL in input-driven systems

Decima Design

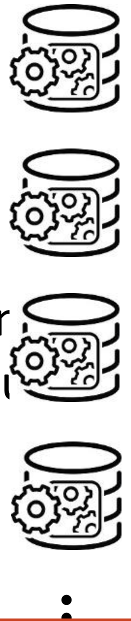
Job DAG 1



Job DAG n

Node features:

- # of tasks
- avg. task duratior
- # of current execı



Executor 1

Executor 2

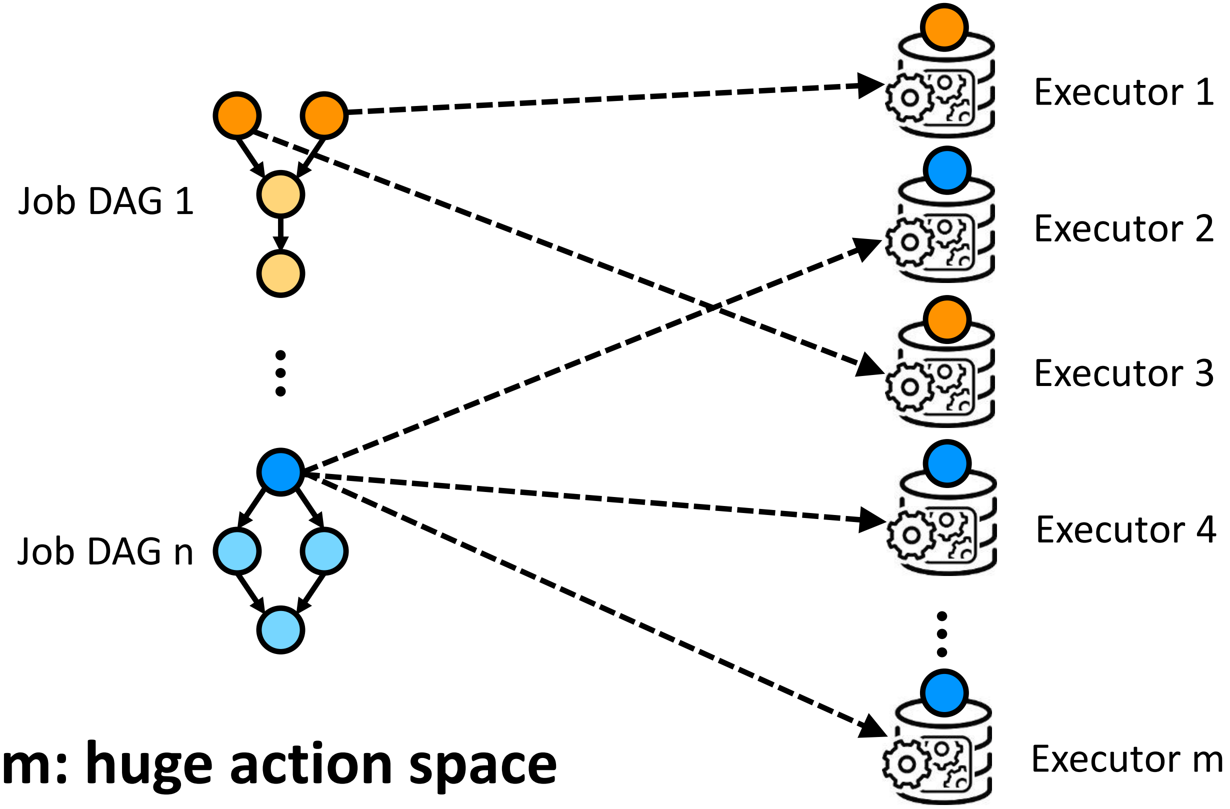
Executor 3

Executor 4

Set of identical free executors

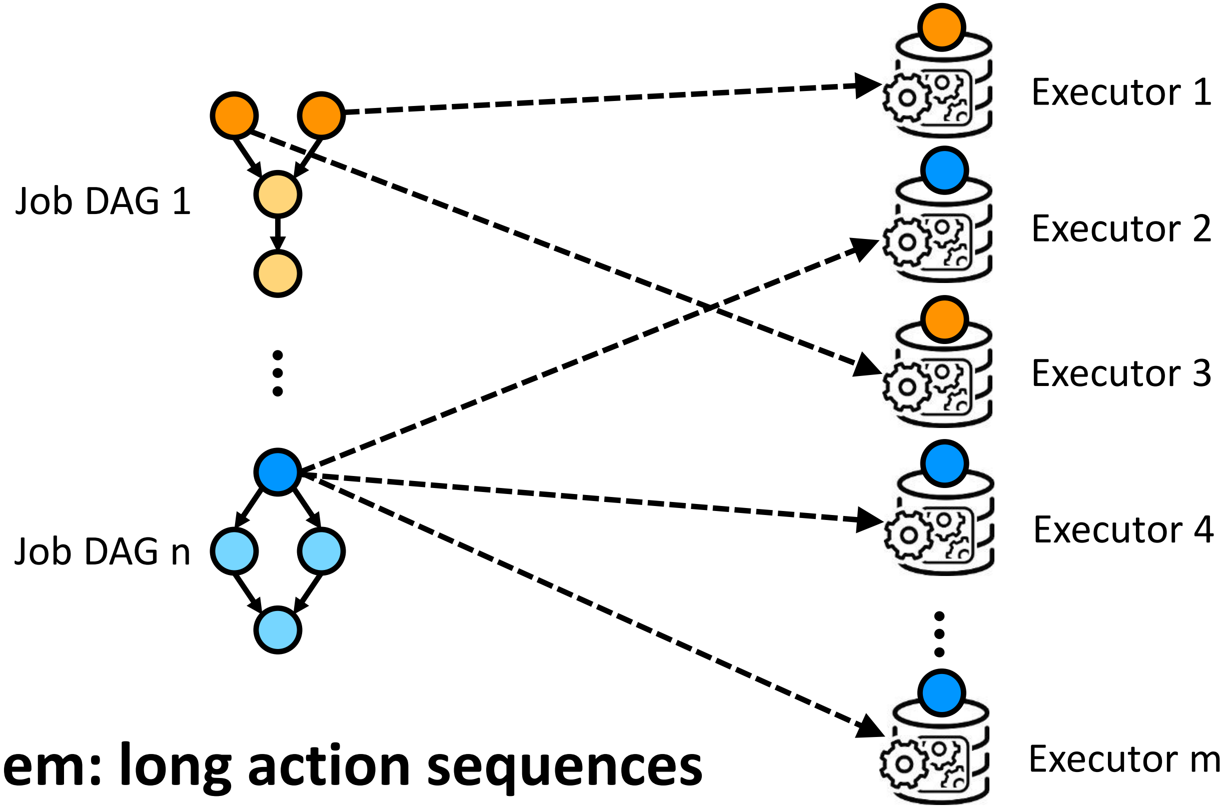
How to encode scheduling decisions as actions?

Option 1: Assign all Executors in 1 Action



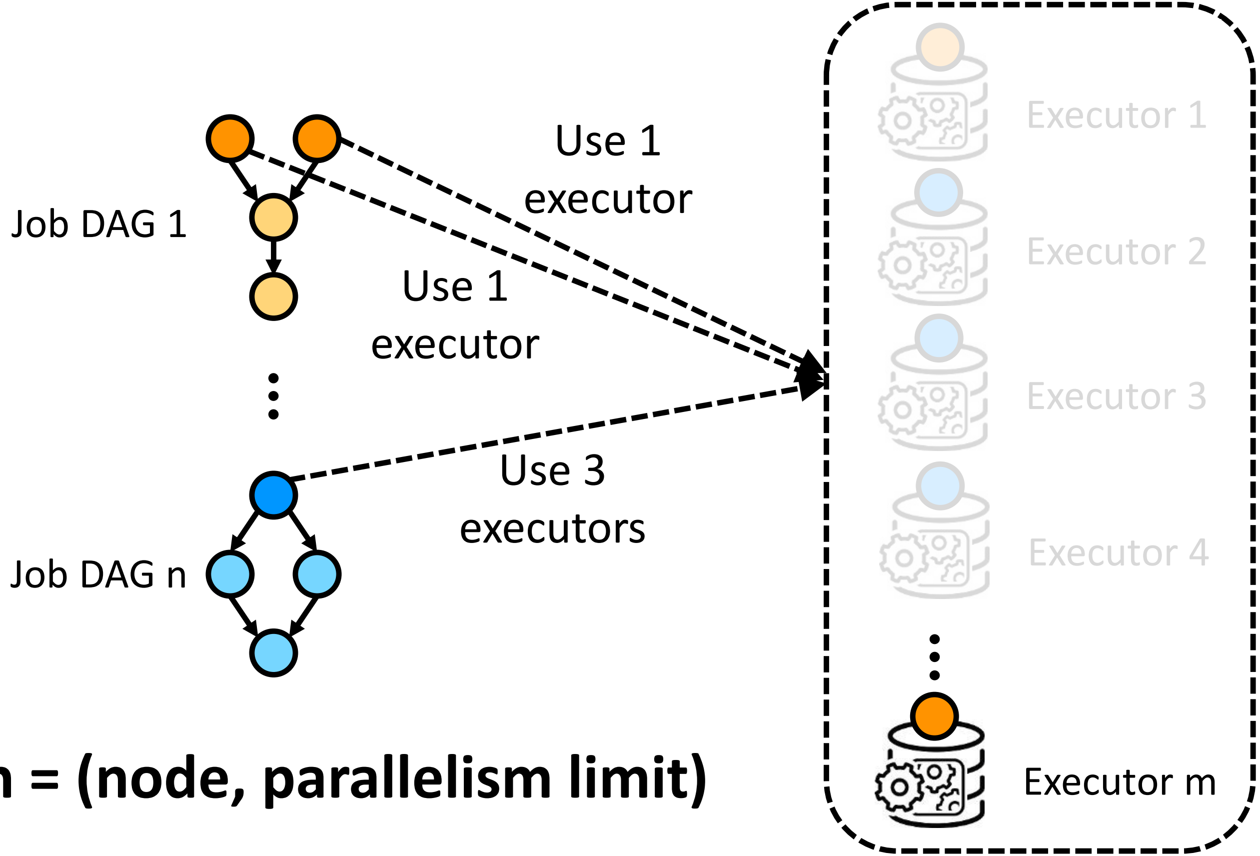
Problem: huge action space

Option 2: Assign 1 Executor per Action



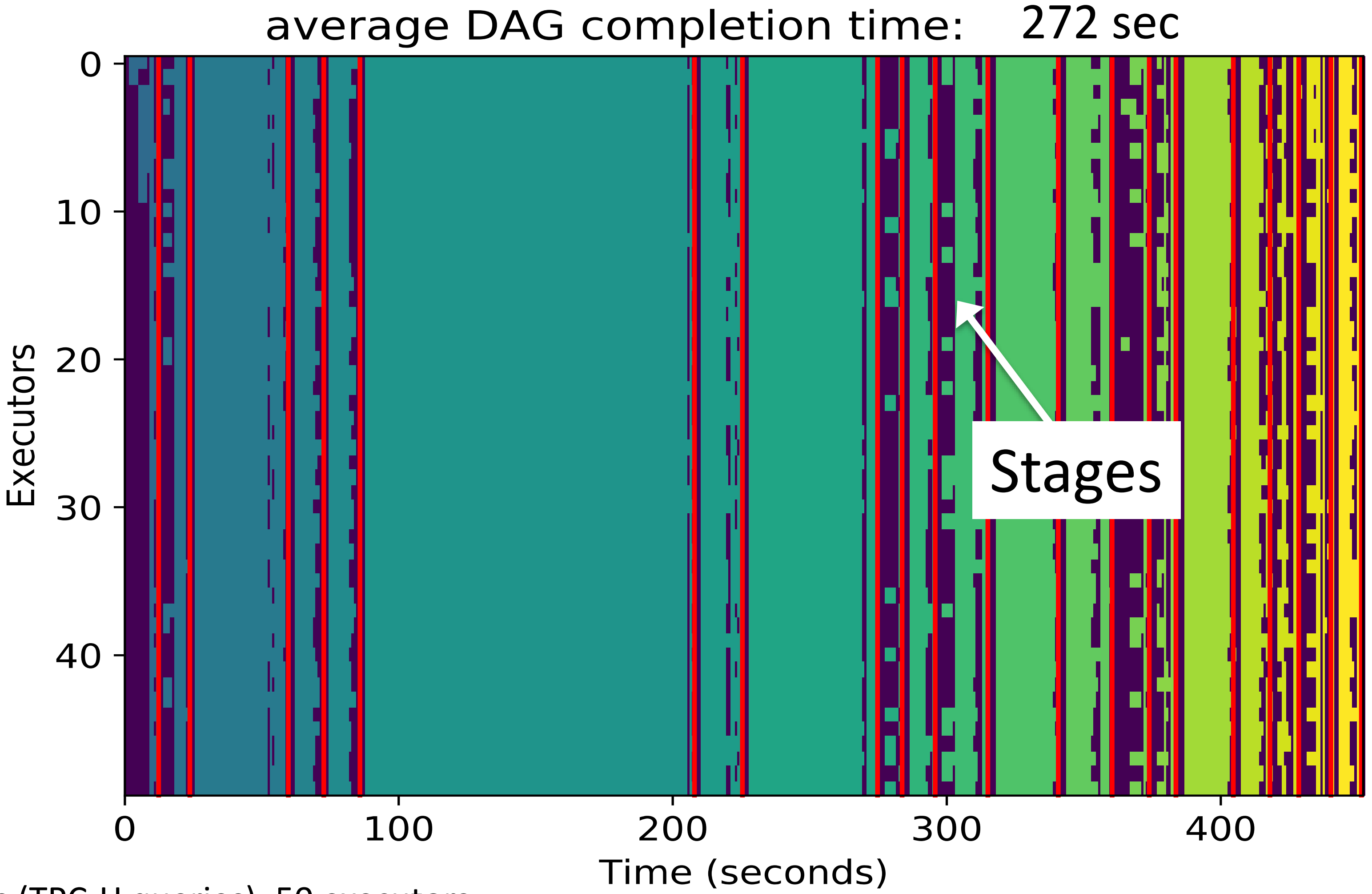
Problem: long action sequences

Decima: Assign Groups of Executors per Action



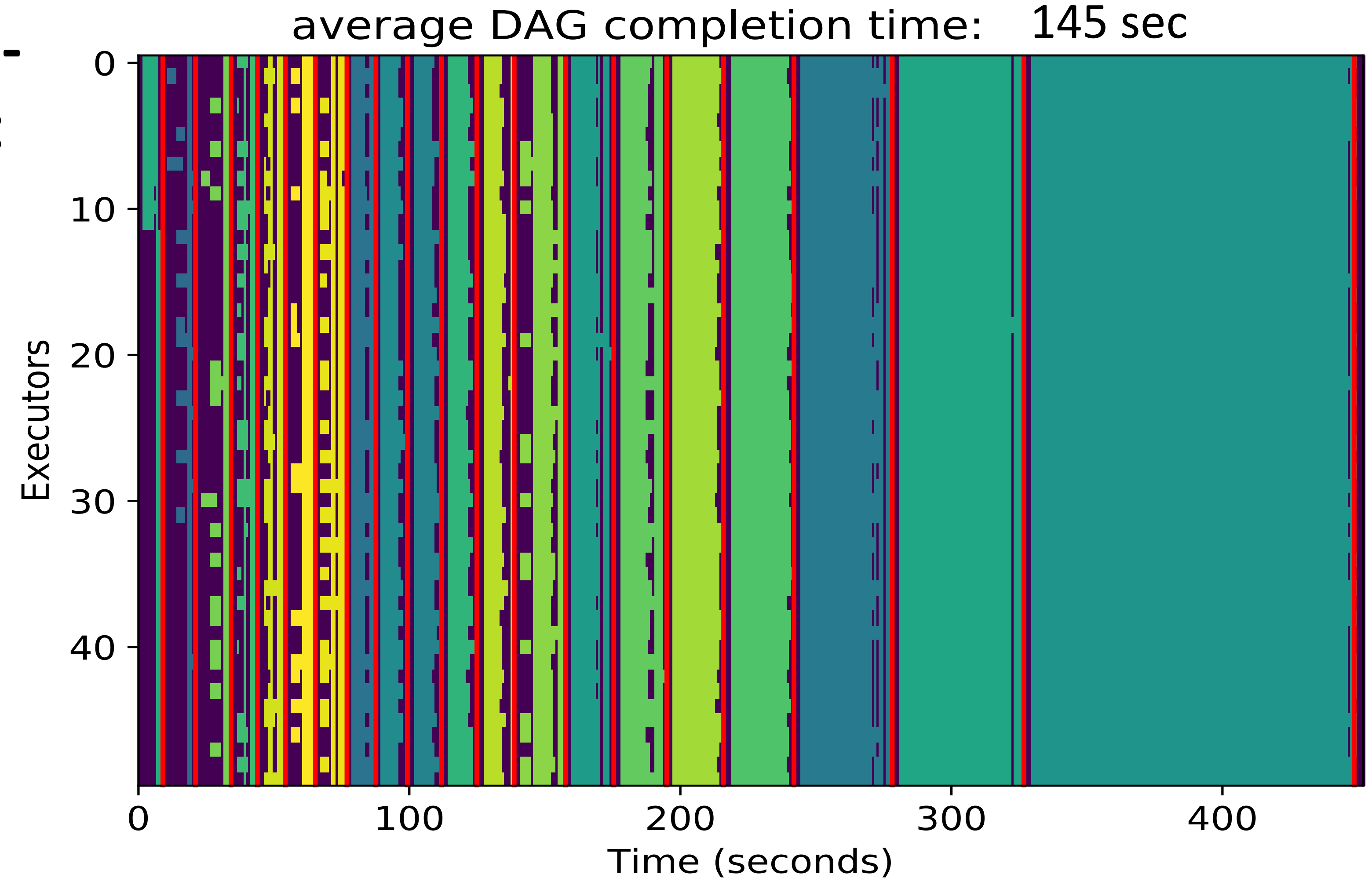
Action = (node, parallelism limit)

FIFO

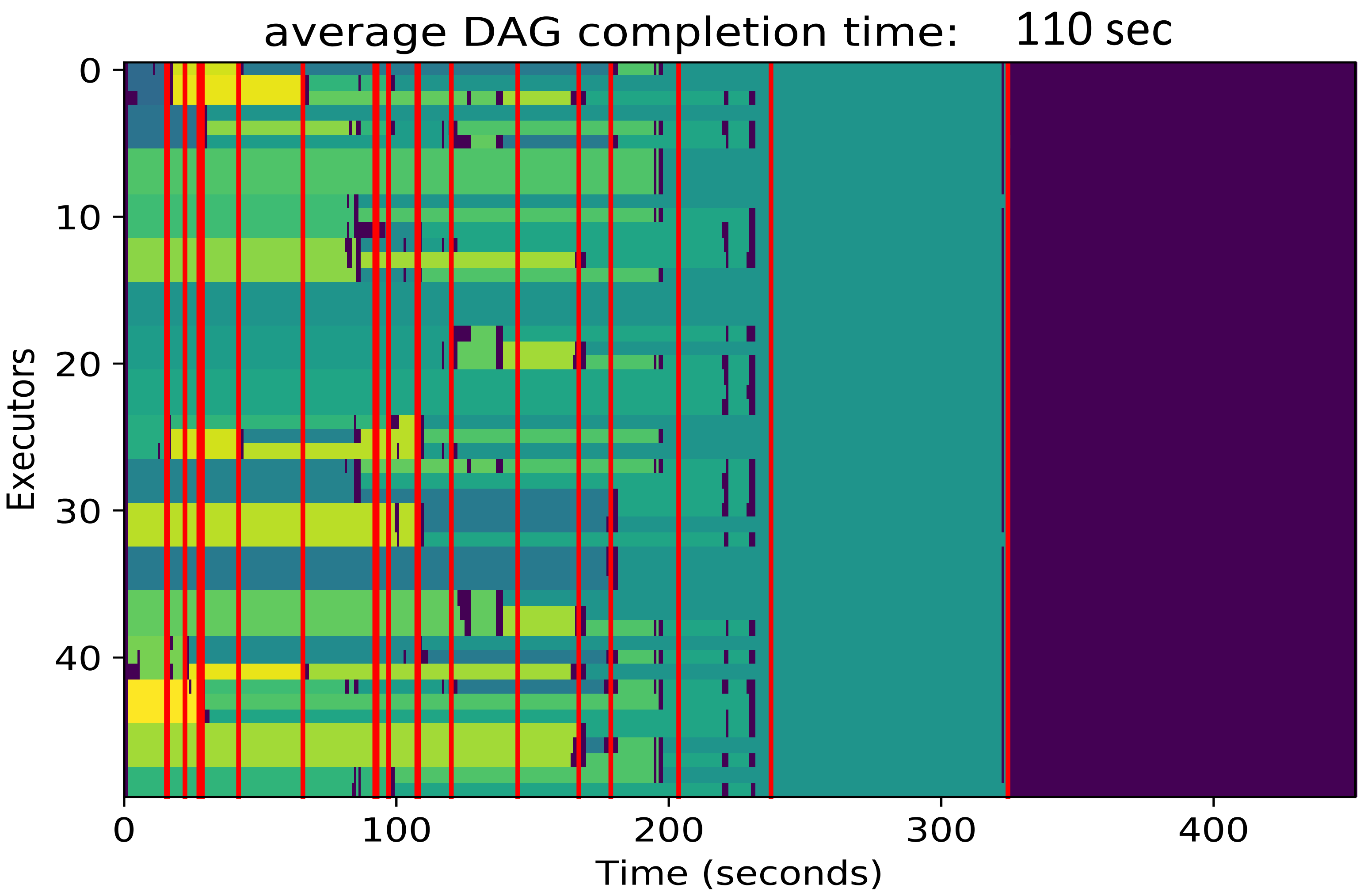


✧ 20 Spark jobs (TPC-H queries), 50 executors

Shortest-Job-First

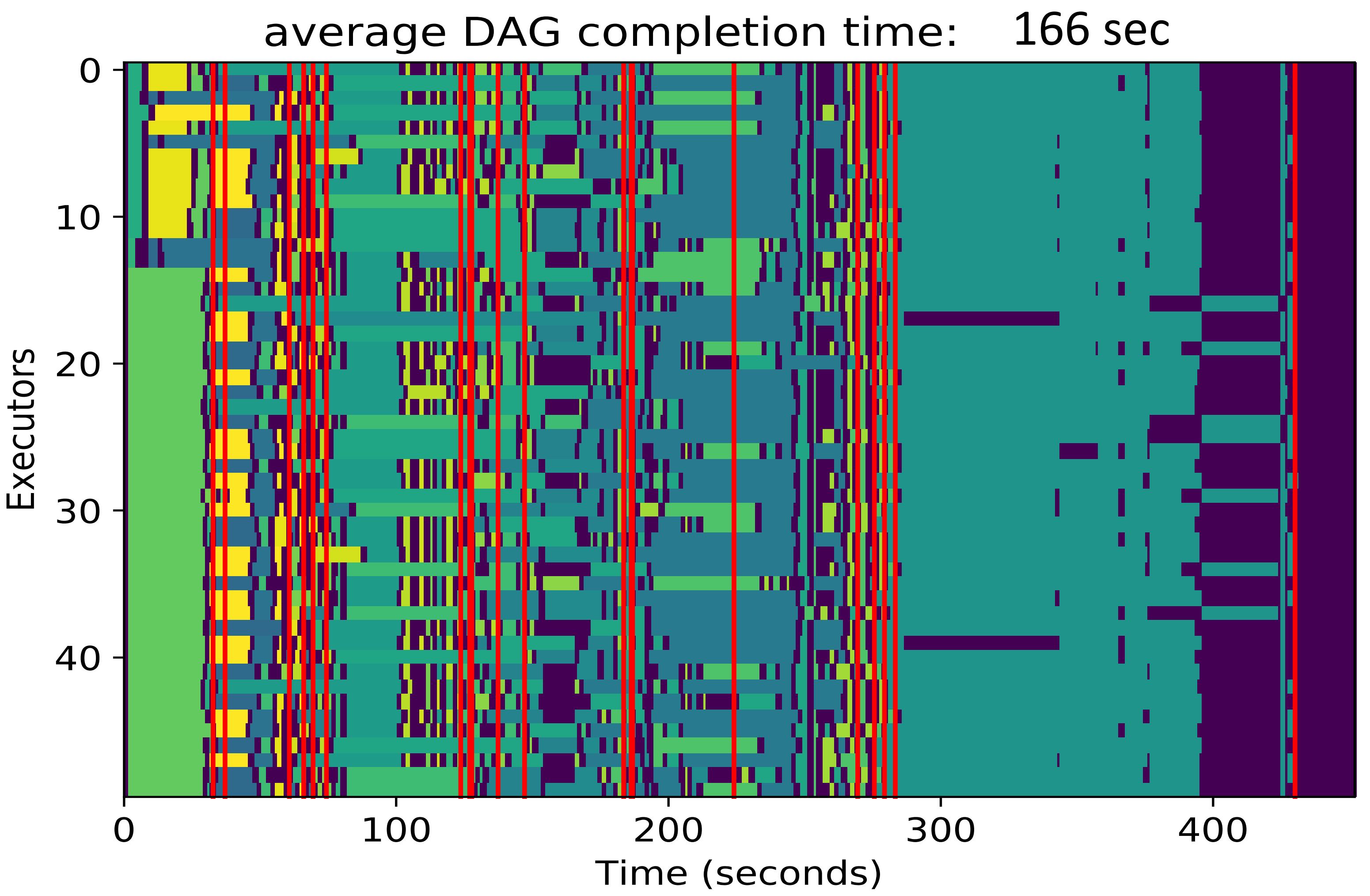


Fair


average DAG completion time: 110 sec
Executors
0
10
20
30
40
0
100
200
300
400
Time (seconds)

Decima

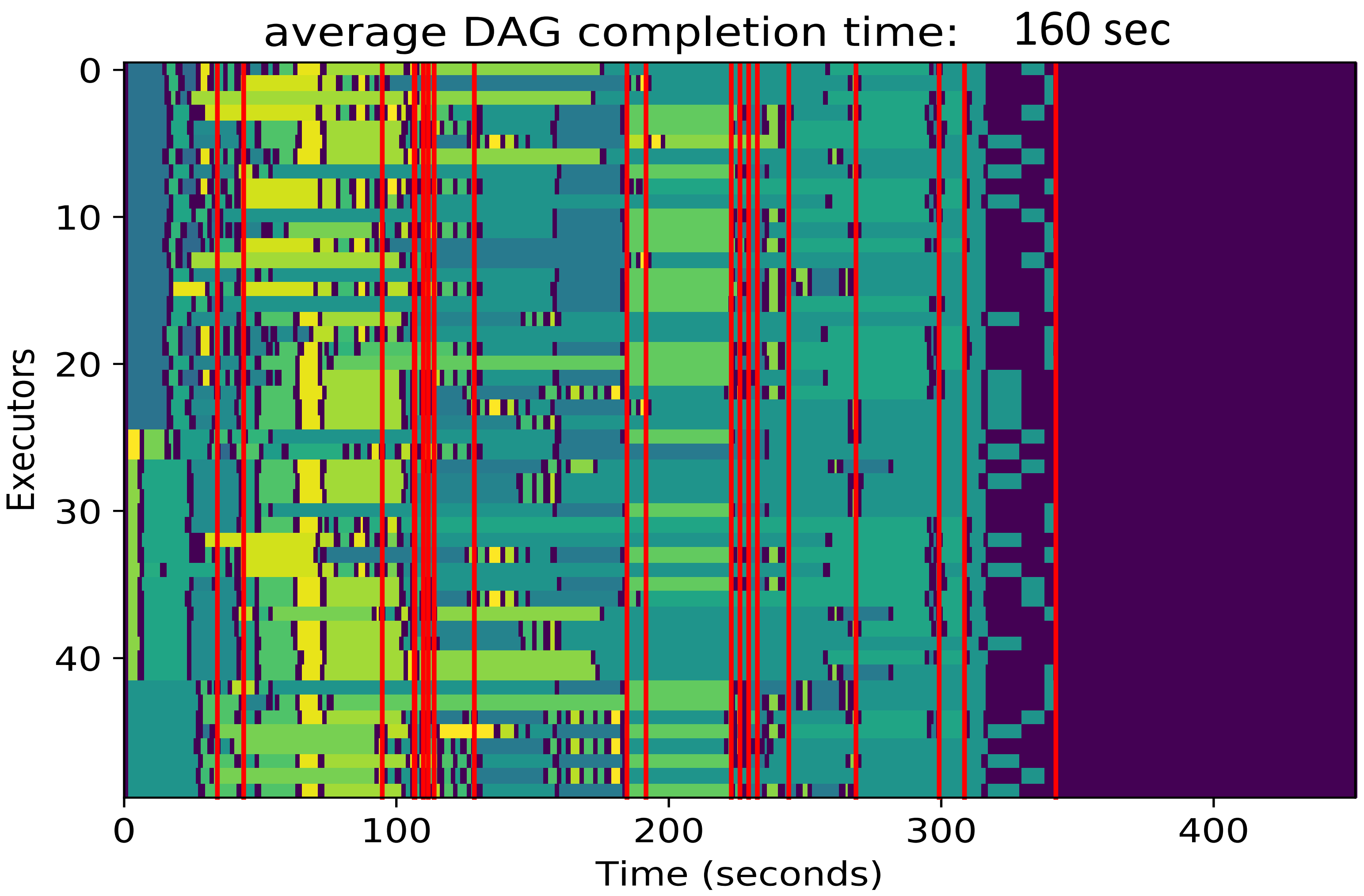
it=0



average DAG completion time: 166 sec
Executors
Time (seconds)
0
10
20
30
40
0
100
200
300
400

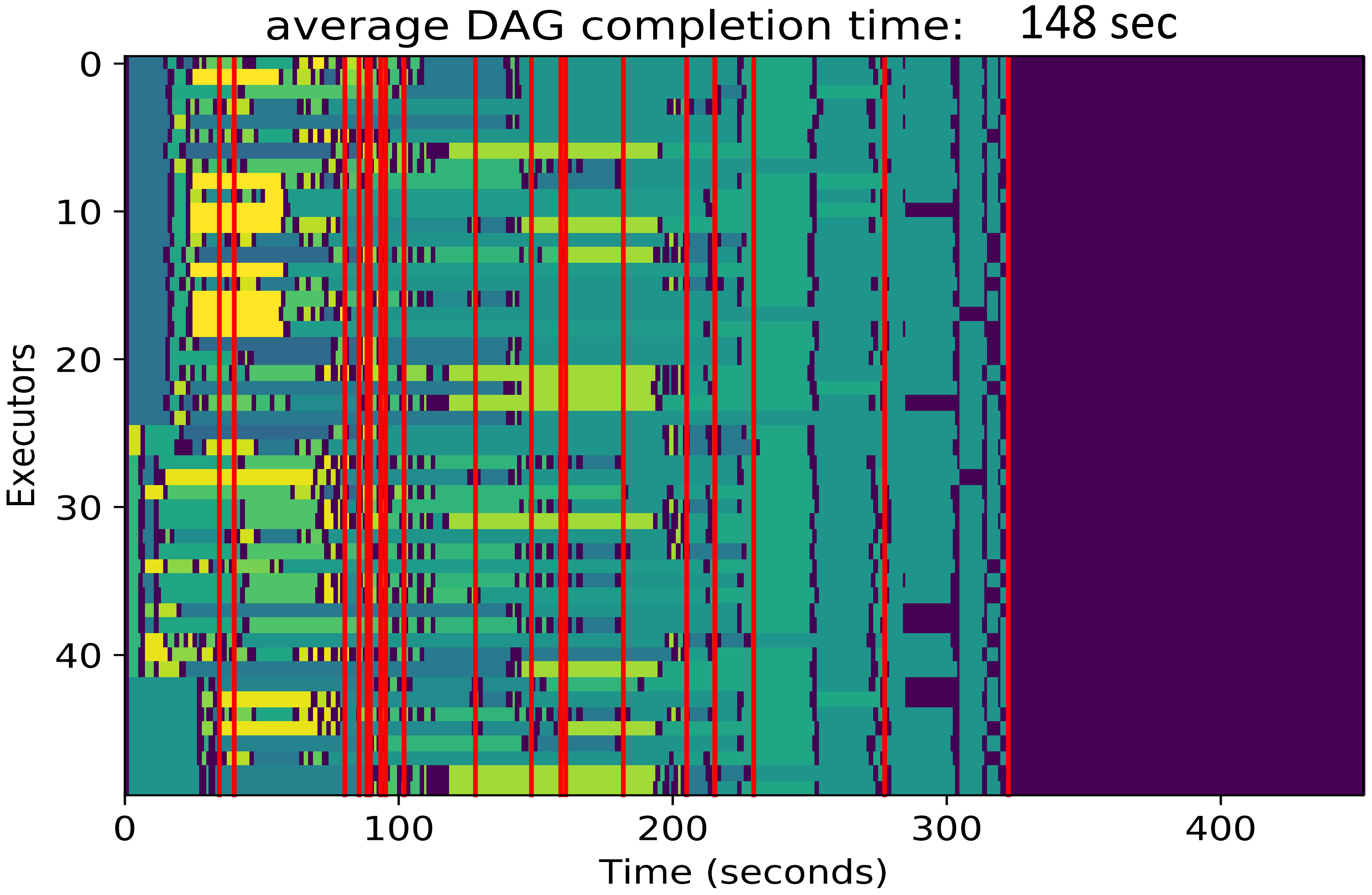
Decima

it=3000



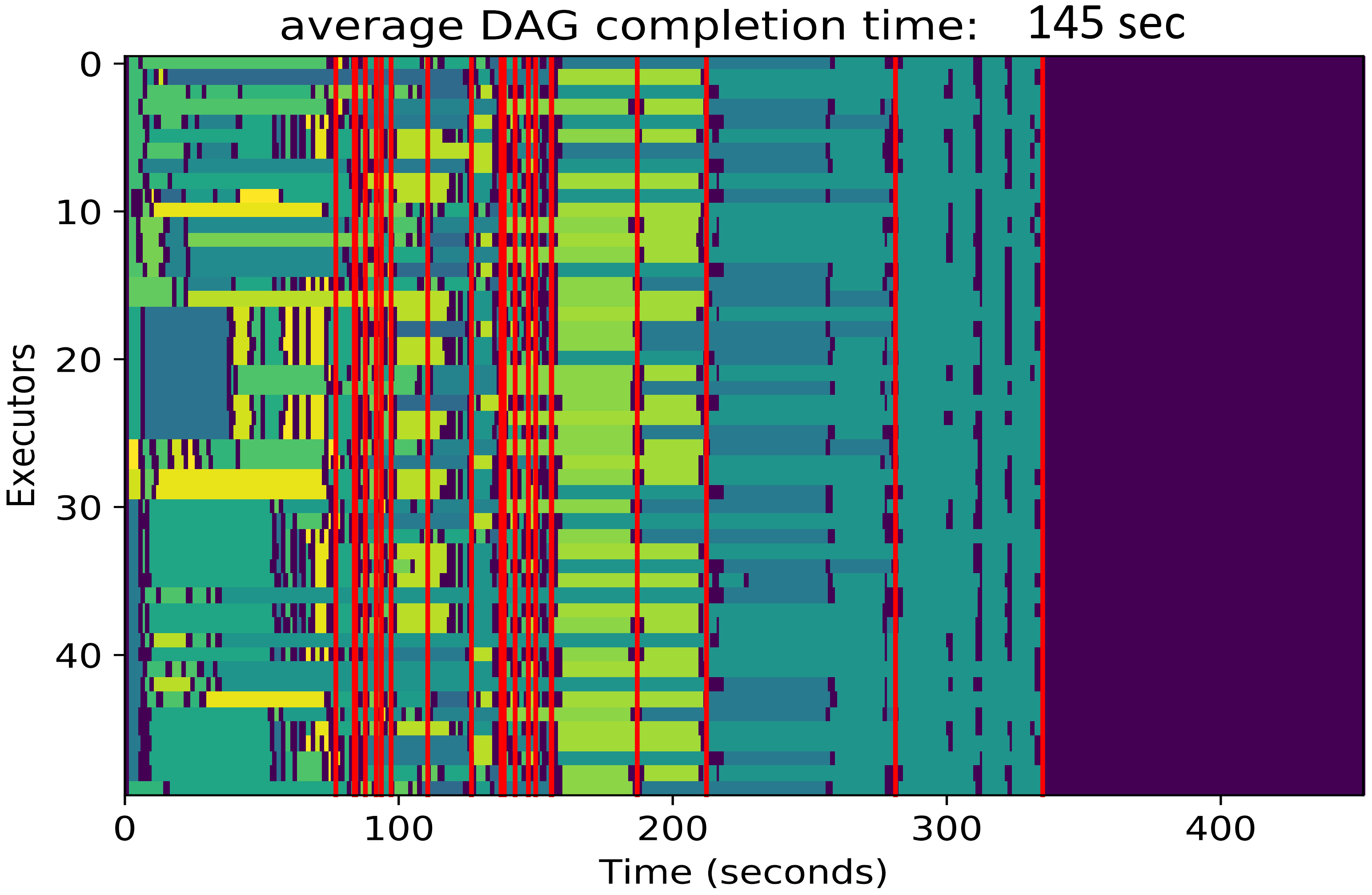
Decima

it=6000



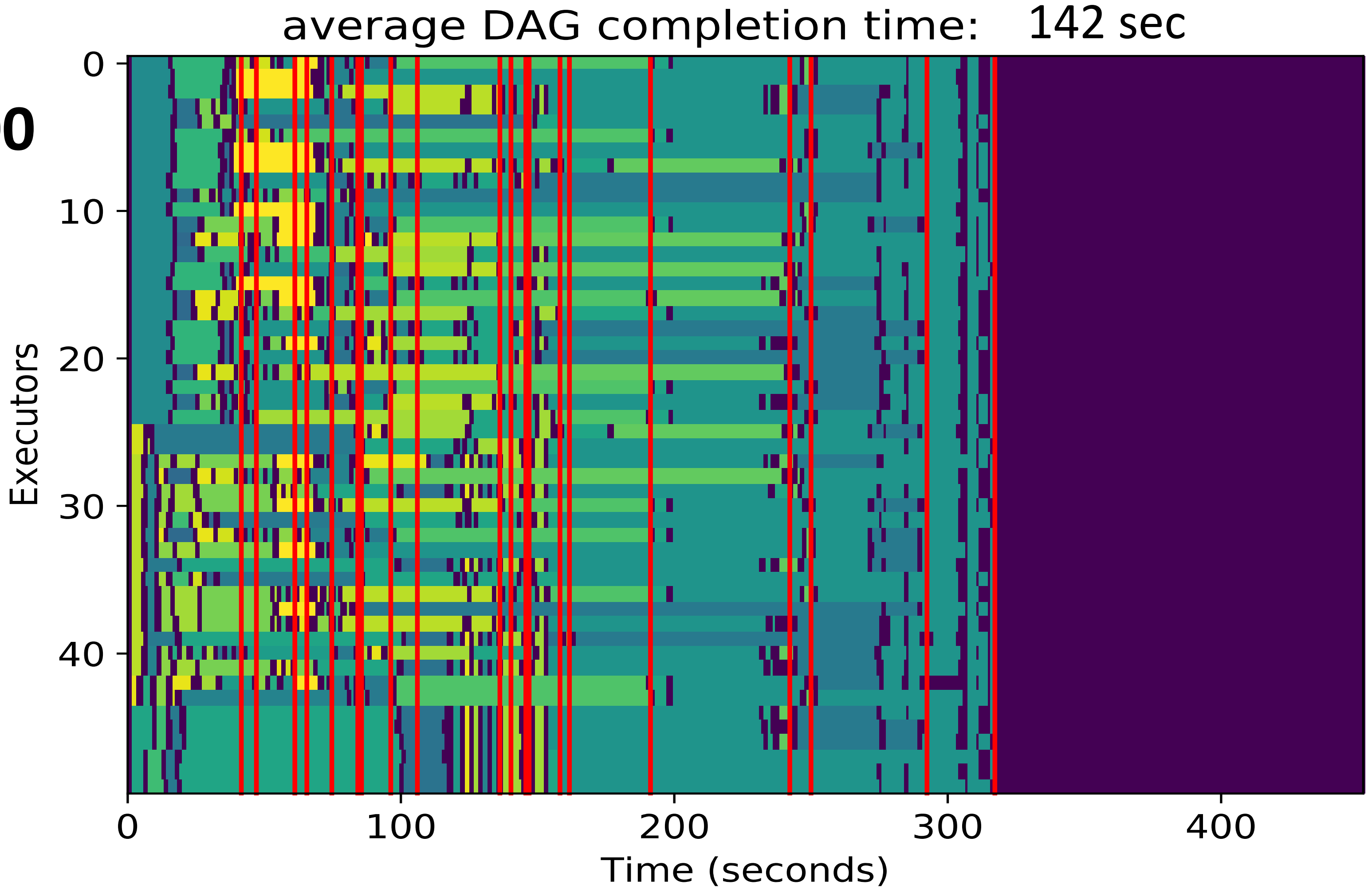
Decima

it=9000

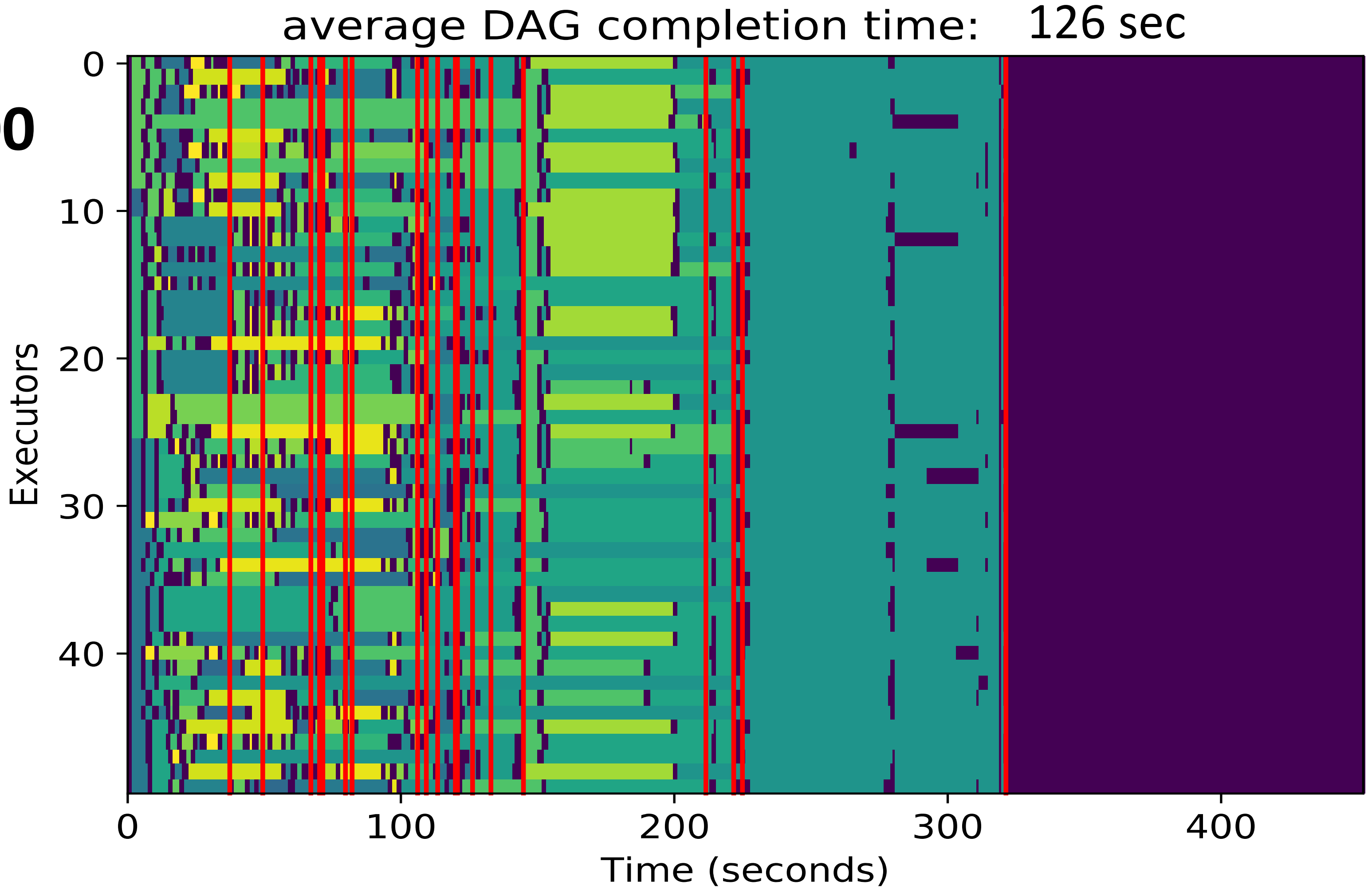


Decima

it=12000

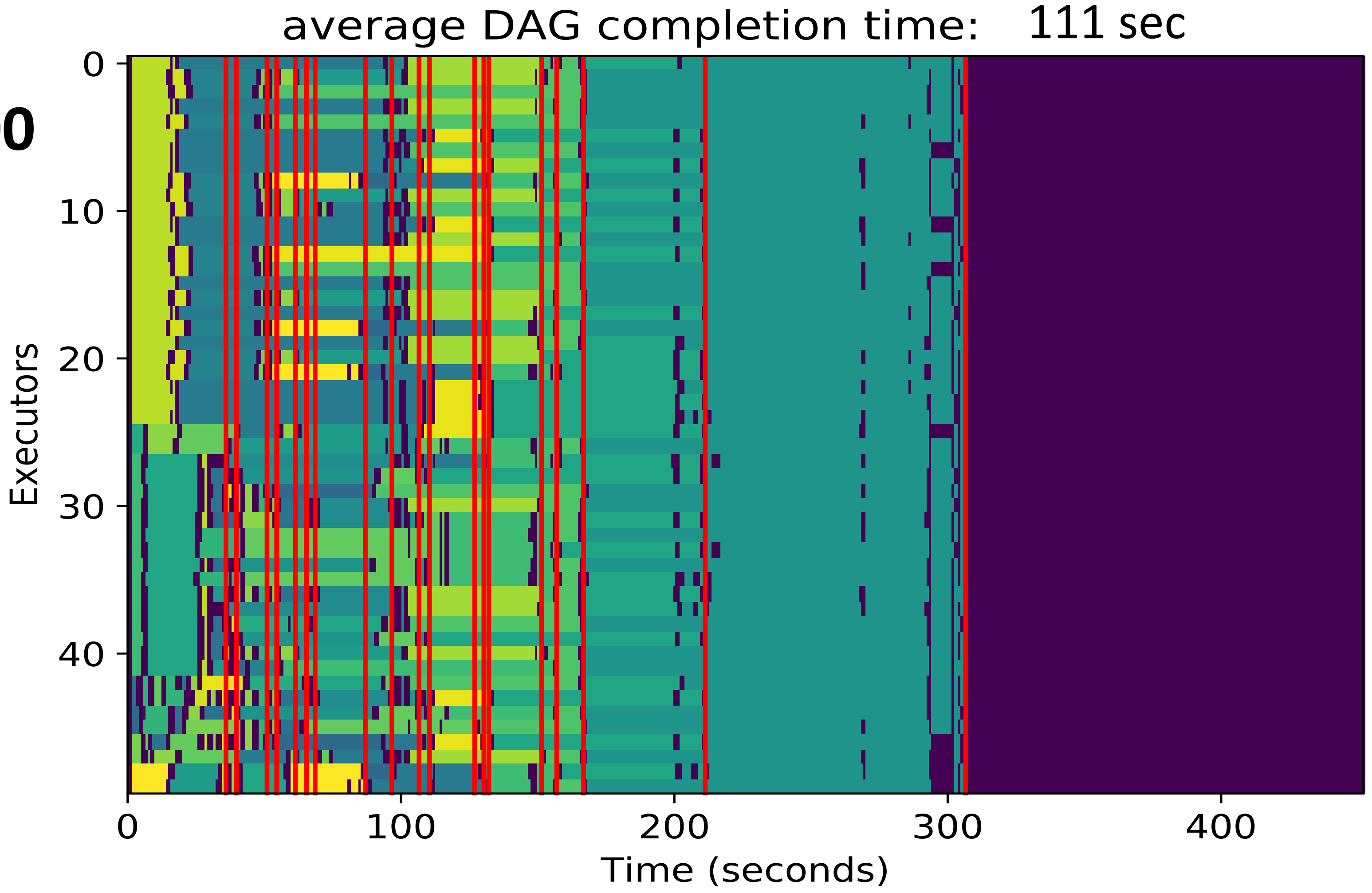


Decima
it=15000



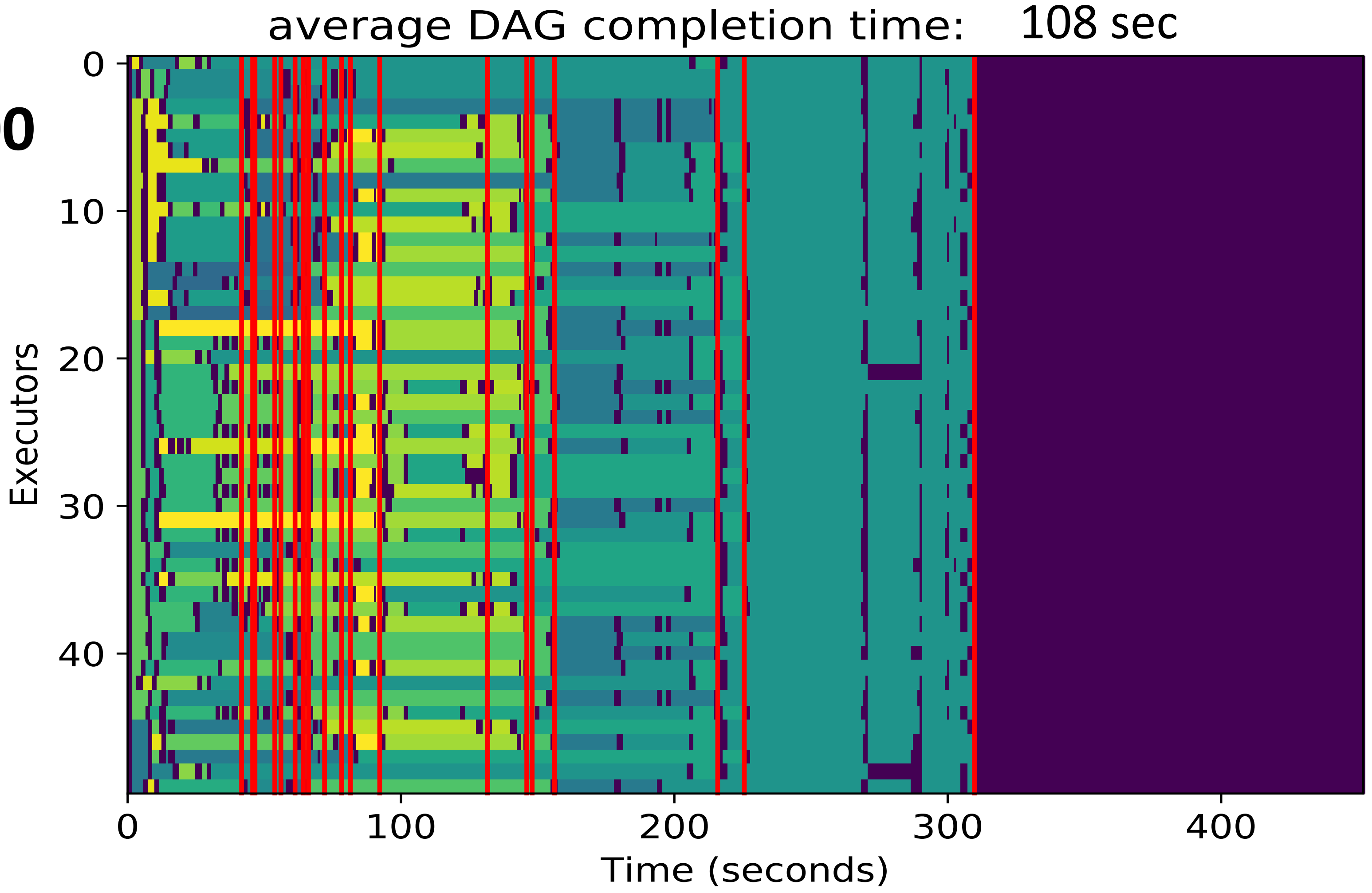
Decima

it=18000



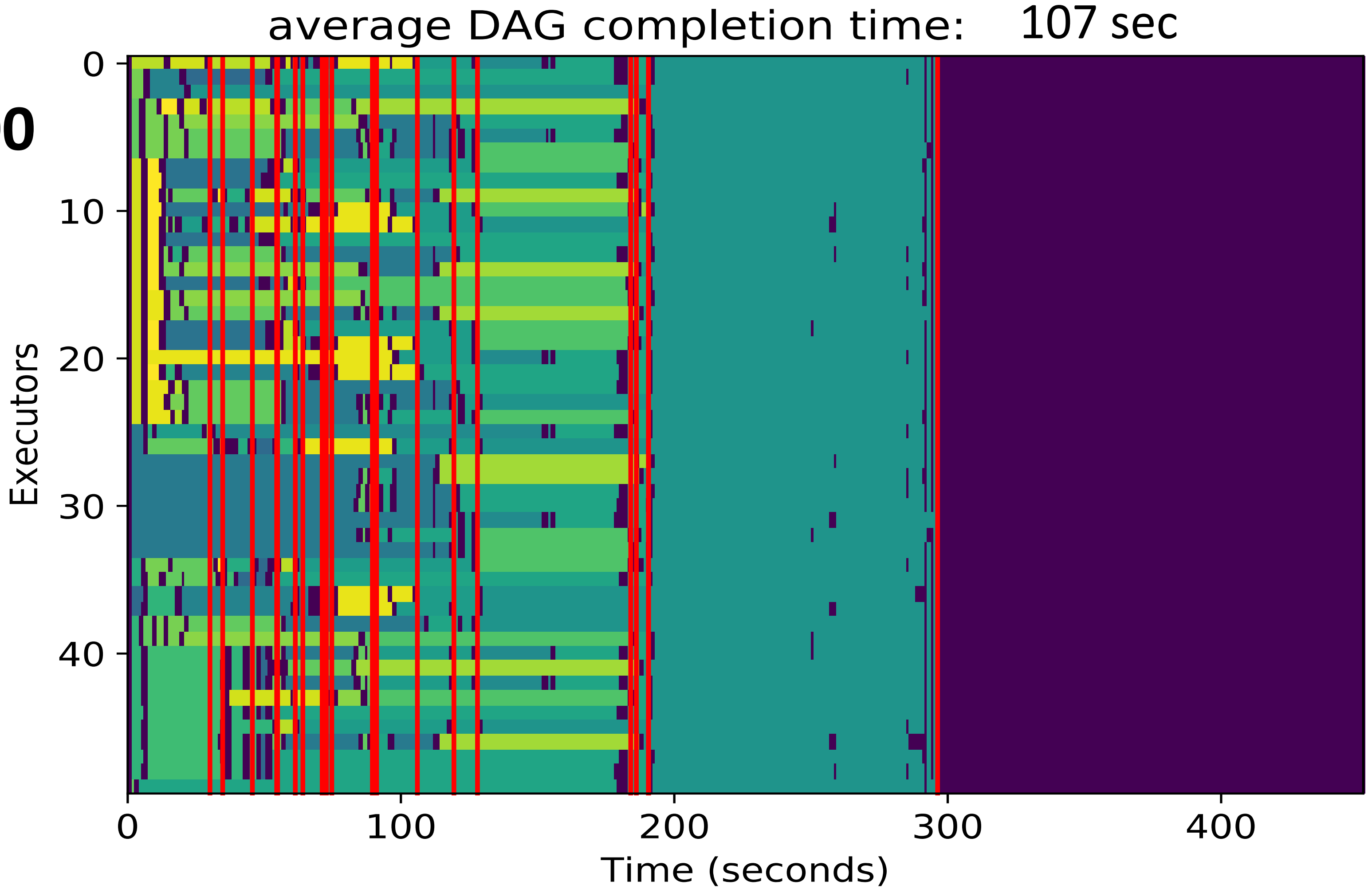
Decima

it=21000



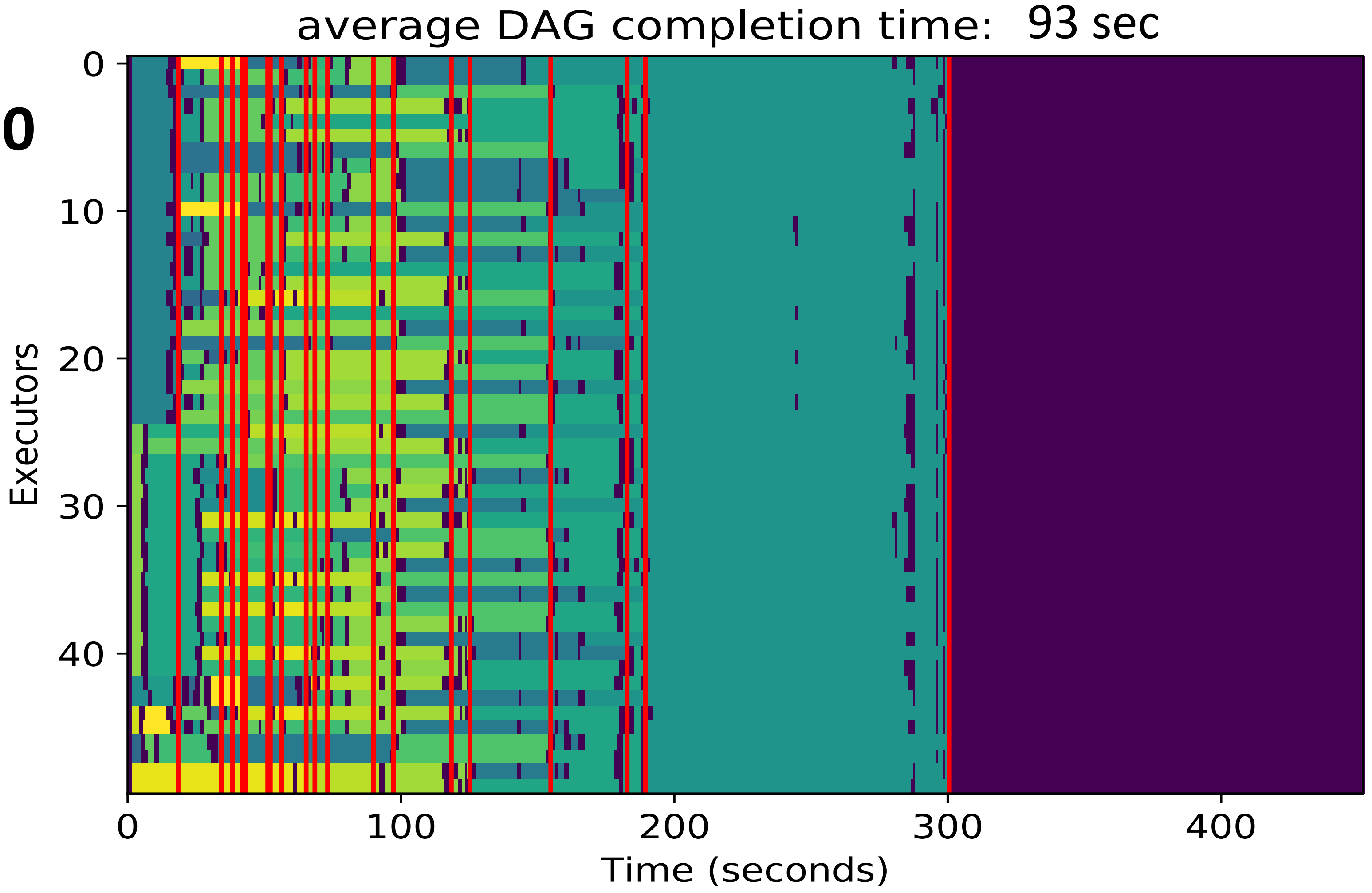
Decima

it=24000

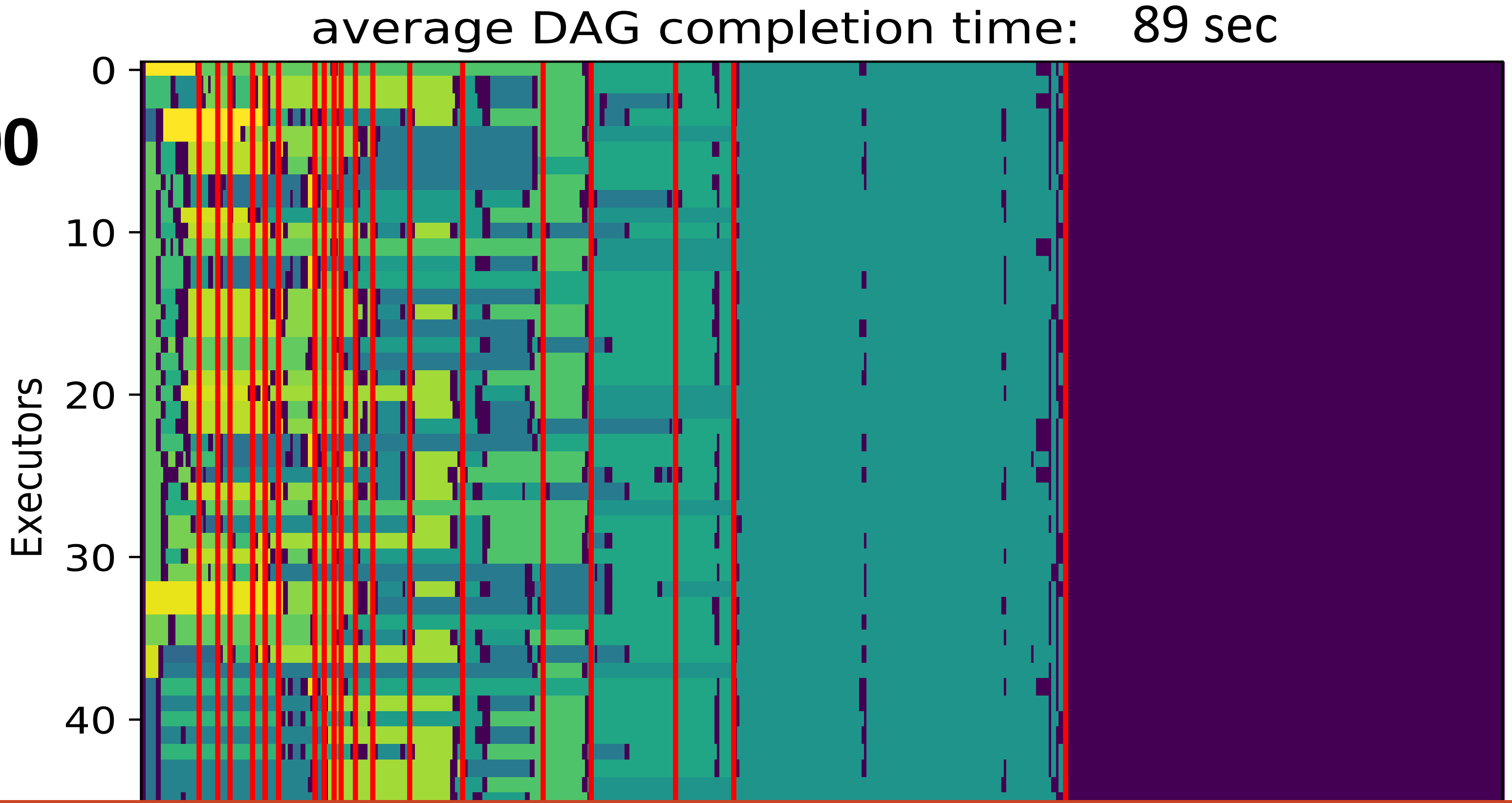


Decima

it=27000



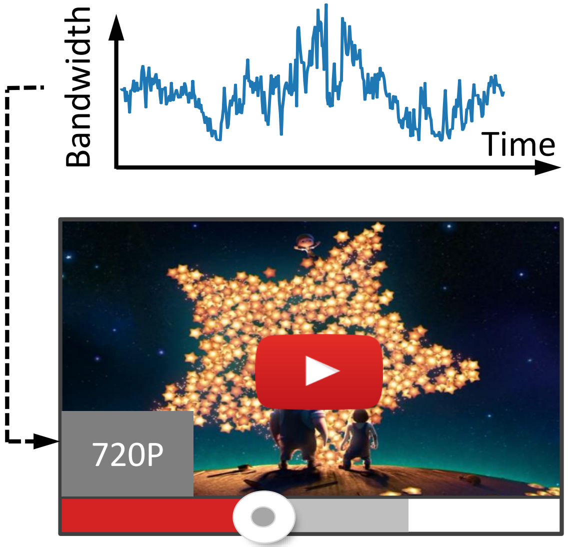
Decima
it=30000



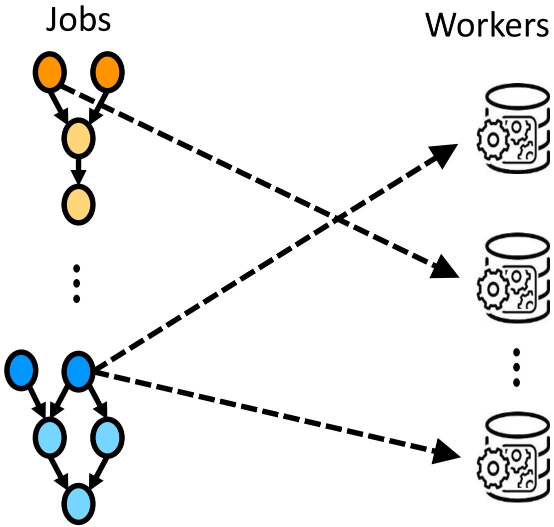
Decima improves average job completion time by >2x over existing schedulers, and 19-31% over best hand-crafted heuristics

Our Work on Learning-Based Network Systems

Adaptive video streaming
(Pensieve)



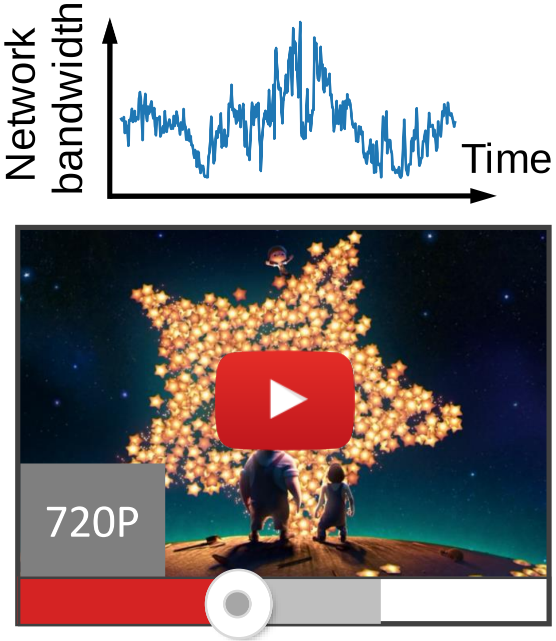
Cluster Scheduling
(Decima)

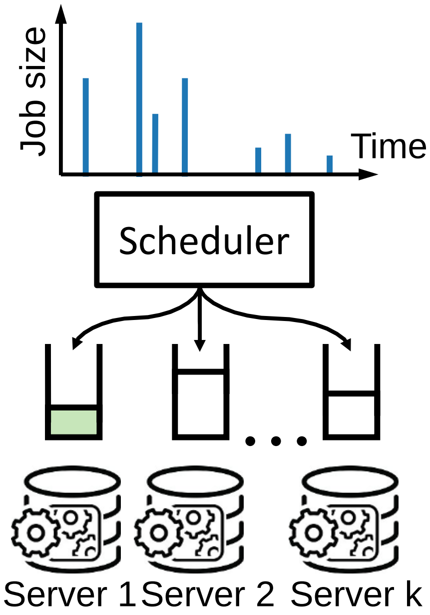


RL Techniques

Input-Driven Environments

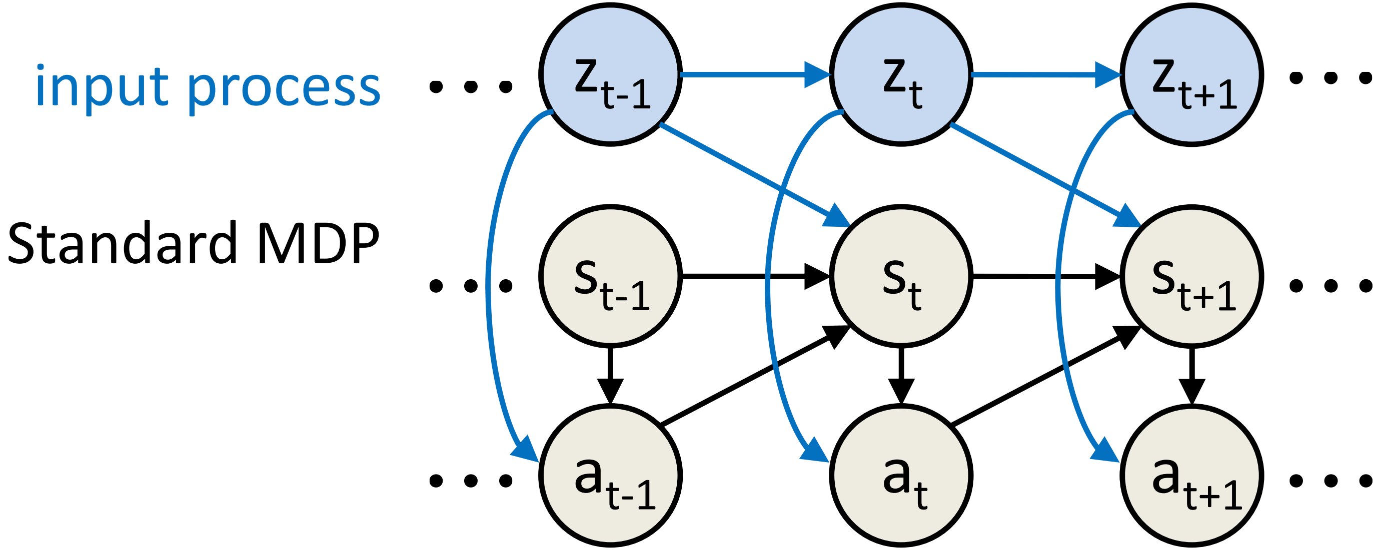
Dynamics driven by an exogenous stochastic **input process**





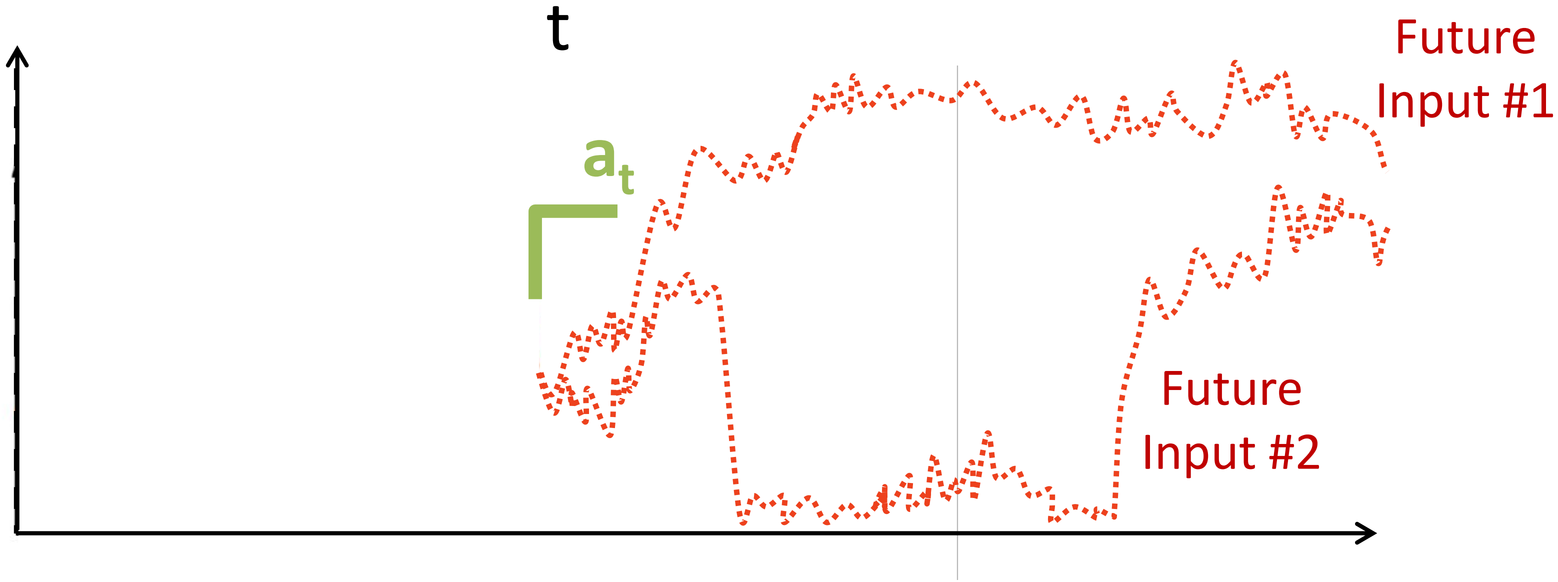
Input-Driven Environments

Dynamics driven by an exogenous stochastic **input process**



The input does not depend on the states and actions

Reinforcement Learning for Input-Driven MDPs



Must take the future input into account when evaluating actions

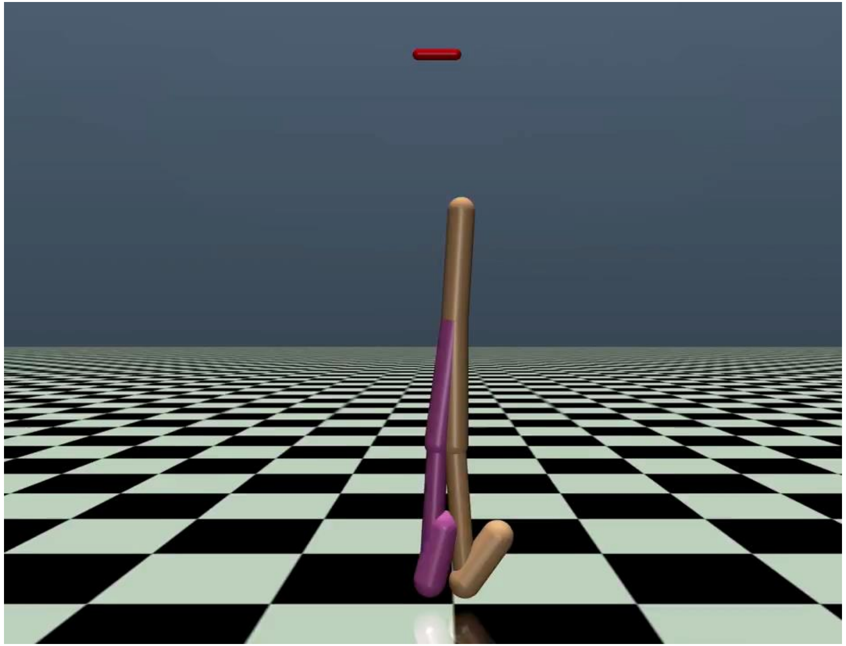
Input-Dependent Baseline

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

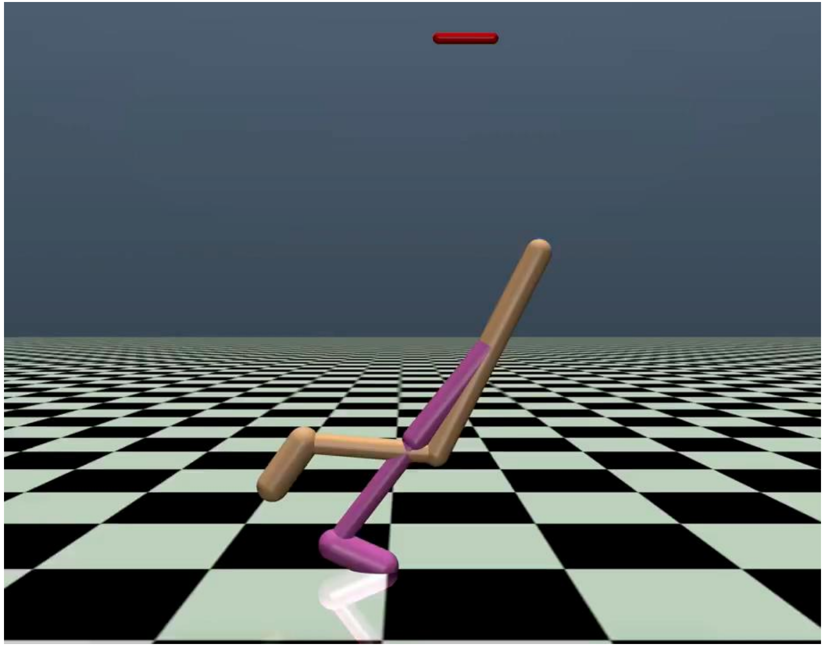
Expected return for trajectories from state s_t
with input sequence z_t, z_{t+1}, …

- Input-dependent baselines reduce variance without bias
- We use meta-learning to learn baseline efficiently

Walker2d with Wind

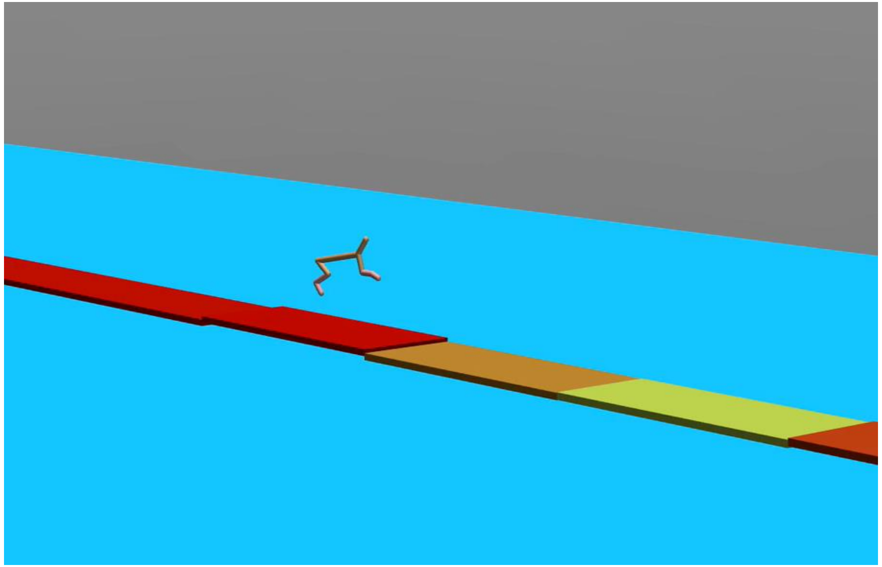


TRPO with standard baseline

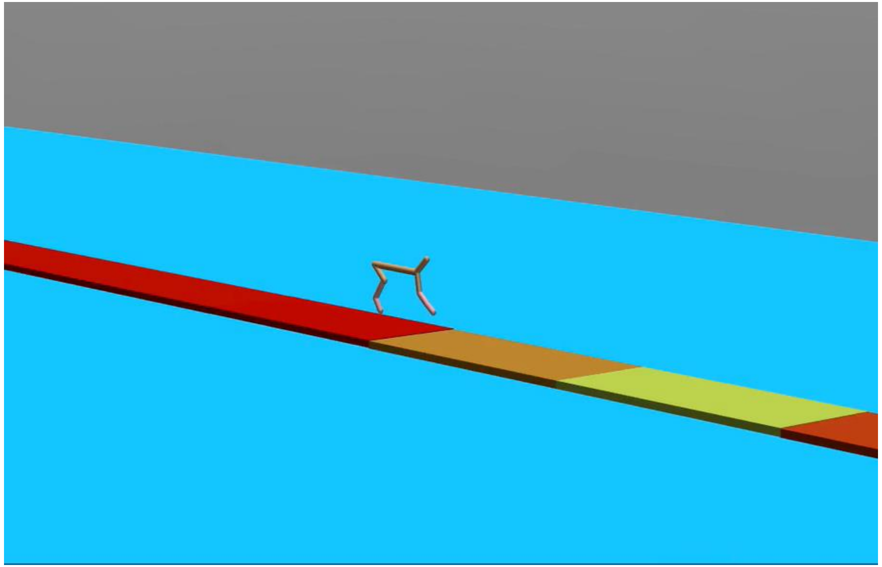


TRPO with input-dependent baseline

Half-Cheetah on Floating Tiles

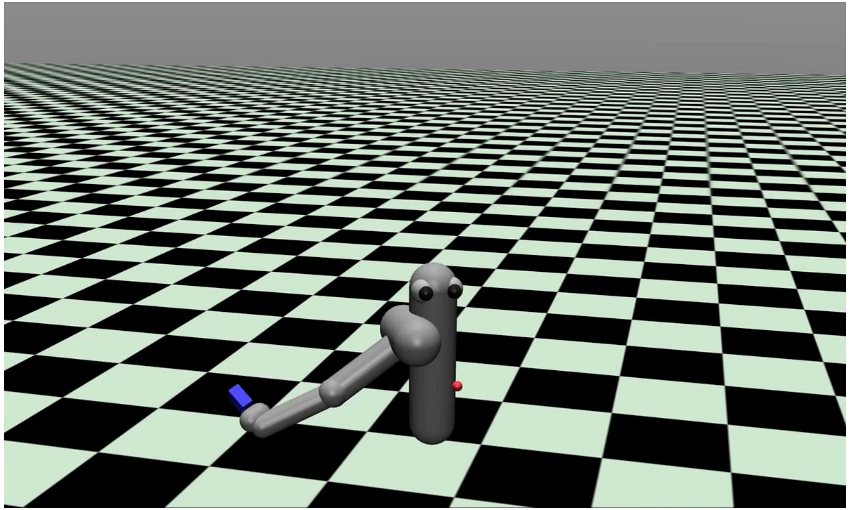


TRPO with standard baseline

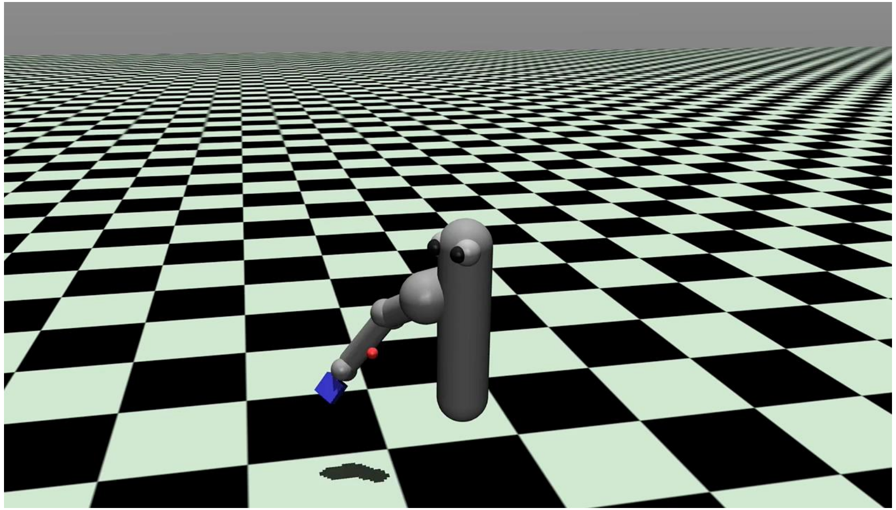


TRPO with input-dependent baseline

7-DoF Arm Tracking Moving Object



TRPO with standard baseline



TRPO with input-dependent baseline

Summary

- Significant opportunity to build smarter networks that can adapt to workload and environment
- Network systems are an exciting domain for ML/AI; can lead to new techniques with broad applications
- Many challenges remain for learning-based systems
  - Safe training & exploration
  - Interpretability
  - …