TEXAS TECH UNIVERSITY SYSTEM

Data Centers Job Scheduling with Deep Reinforcement Learning

Sisheng Liang, Zhou Yang, Fang Jin, and Yong Chen
Computer Science Department, Texas Tech University, USA

Presented by Sisheng Liang
May 14th, 2020

Table of Contents

- Background & Motivation
- Reinforcement Learning
- Proposed Work
- Experiments
- Conclusion

Background

- The HPC job scheduler is a global queuing system
- Jobs are submitted to a centralized job queue
- Jobs are ordered in the queue based on their priorities
- Scheduler picks jobs to run based on available resources
- Re-claim resources once a job finishes

Background

- Conventional HPC Job Scheduling Policies
  - FCFS
  - Backfiling
  - Preemption (w/o resumption)
  - List scheduling
  - Fairness Sharing
- Real world HPC job schedulers
  - Slurm
  - SGE(UGE)
  - PBS

Motivation

- Design a method to learn the policy automatically
- Adaptive to the workload or optimization objects
- Save time and effort for system administrators
- No job profile information is needed

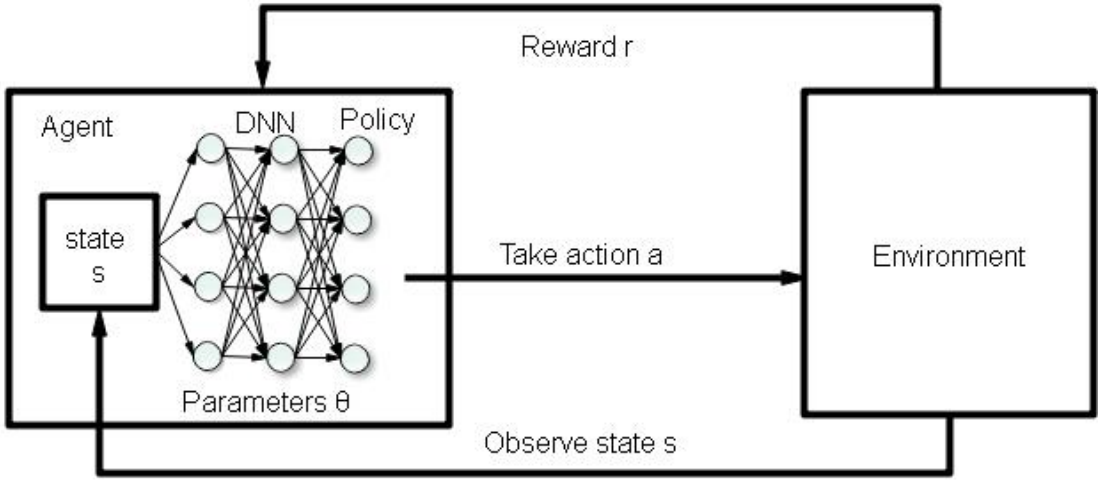
Reinforcement Learning

- Make sequential decisions
- Learn by interacting with the environment

[2] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529.7587 (2016): 484.

Reinforcement Learning

- Deep Reinforcement Learning with Policy gradient method



[1] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource Management with Deep Reinforcement Learning,” in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks - HotNets ’16*, Atlanta, GA, USA, 2016, pp. 50–56.

Proposed A2C Job Scheduling framework

- Job scheduling with Deep reinforcement Learning

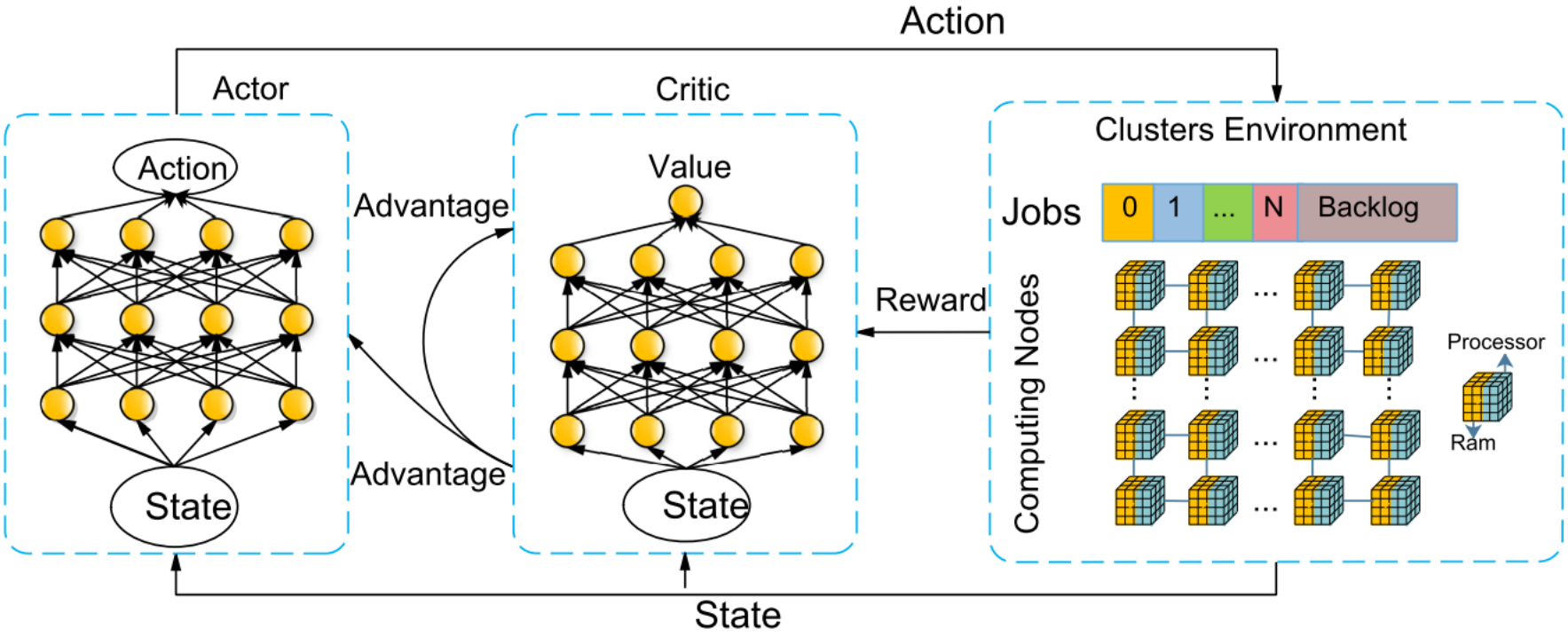


Figure 2: **A2C job scheduling framework.**

A2C Problem Statement

- Agents
- Actor: generate actions, to choose jobs from the queue
- Critic: to evaluate actions of the actor
- Agent Action space: $\{0, 1, 2, \ldots N\}$, $a = j$ means j-th job in queue will be allocated; $a = 0$, void action

A2C Problem Statement

- Reward Function
- Individual reward for *the actor*: $r_t = -\frac{1}{T_j}$
- Expected discount reward: $E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$
- The actor attempts to maximize its expected discounted reward.

Experiment Settings

- Hardware:

  48 GB ram and

  two Nvidia RTX-2080 GPUs, 2944 cuda cores

- Software:

  Tensorflow

- Parameters

  Learning rates: actor 0.0001, critic 0.001

  Actor : 2 layers of CNN, 1 flatten layer, 1 dense layer

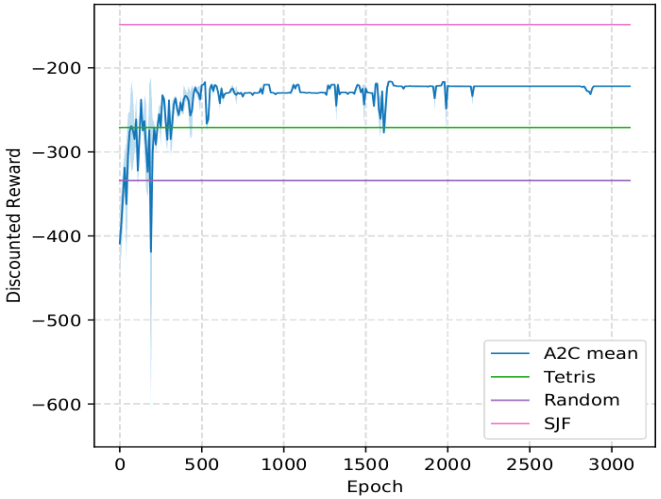
  Critic : 2 layers of CNN, 1 flatten layer, 1 dense layer

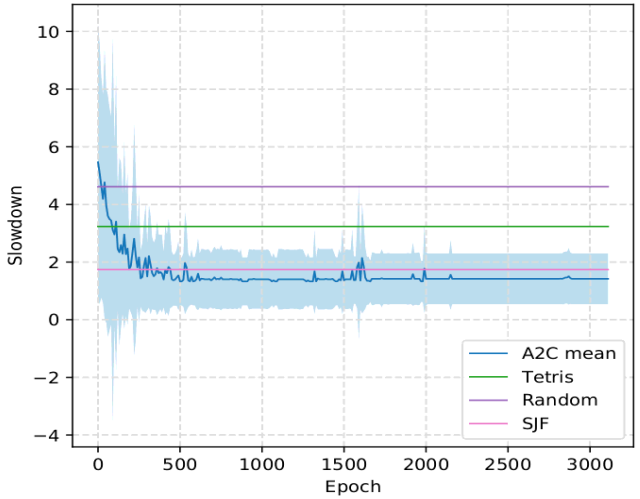
Simulation Results

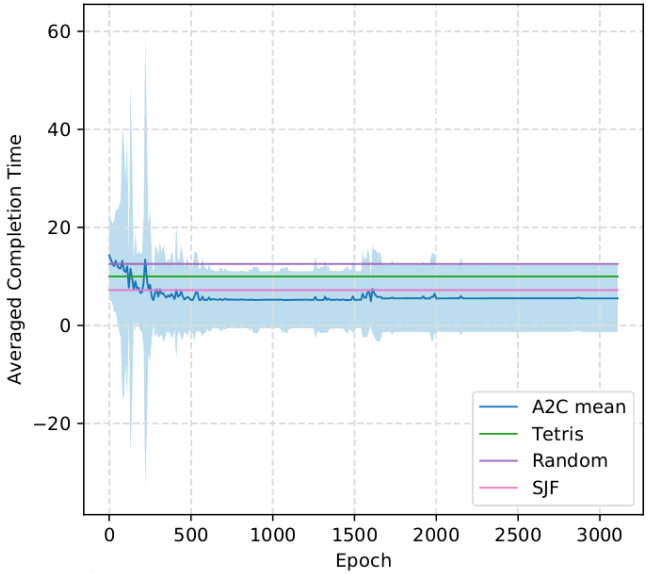
- Simulated workloads

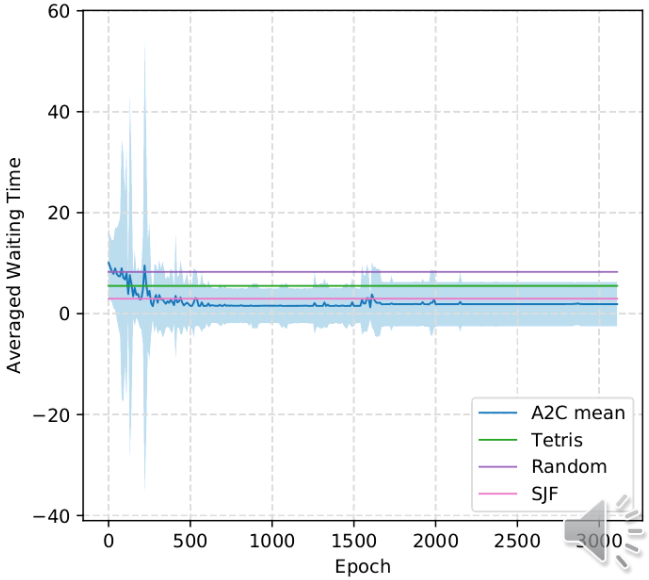
Metrics

- Slowdown:
- Turnaround time:
- Waiting time:

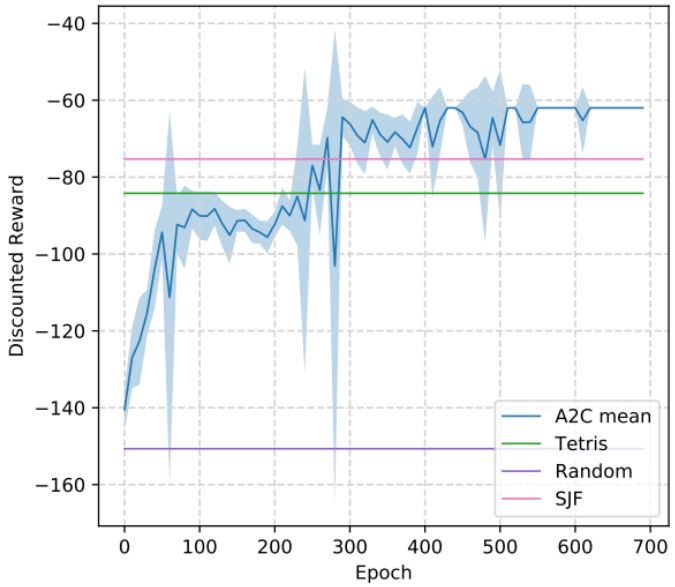


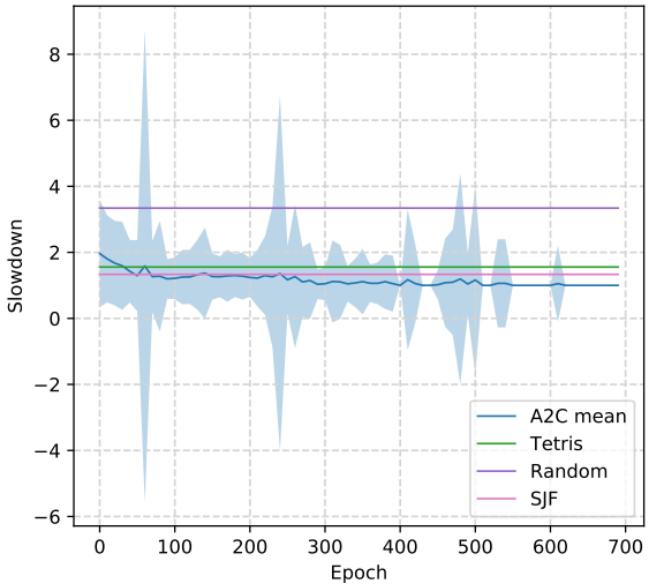


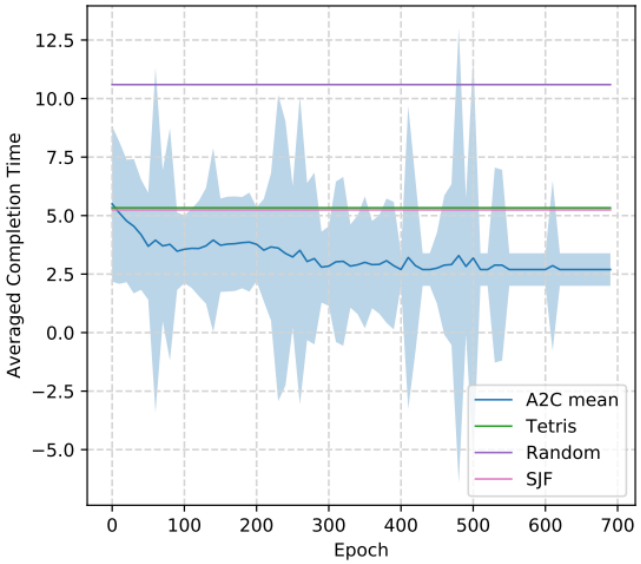


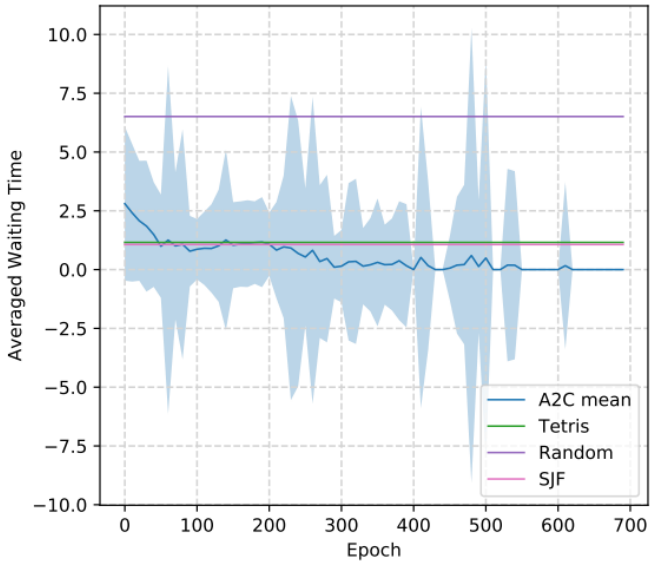


Simulation Results from Job Traces Data









Simulation Results from Job Traces Data

Table 4: **Results of Job Traces**

Type	Random	Tetris	SJF	A2C
Slowdown	3.52±0.00	1.82±0.00	1.61±0.00	**1.01±0.02**
CT^*	10.2±0.00	5.55±0.00	5.51±0.00	**2.58±0.01**
WT^*	6.32±0.00	1.25±0.00	1.21±0.00	**0.01±0.02**

Conclusion

- A2C scheduler could find the effective scheduling policy automatically
- Can save time and effort of the system administrators.
- Our experiments on both simulated data and real jobs traces of HPC systems show that this proposed method performs better than the widely used heuristics.

Thank you !

Q & A