Peter the Great St.Petersburg Polytechnic University

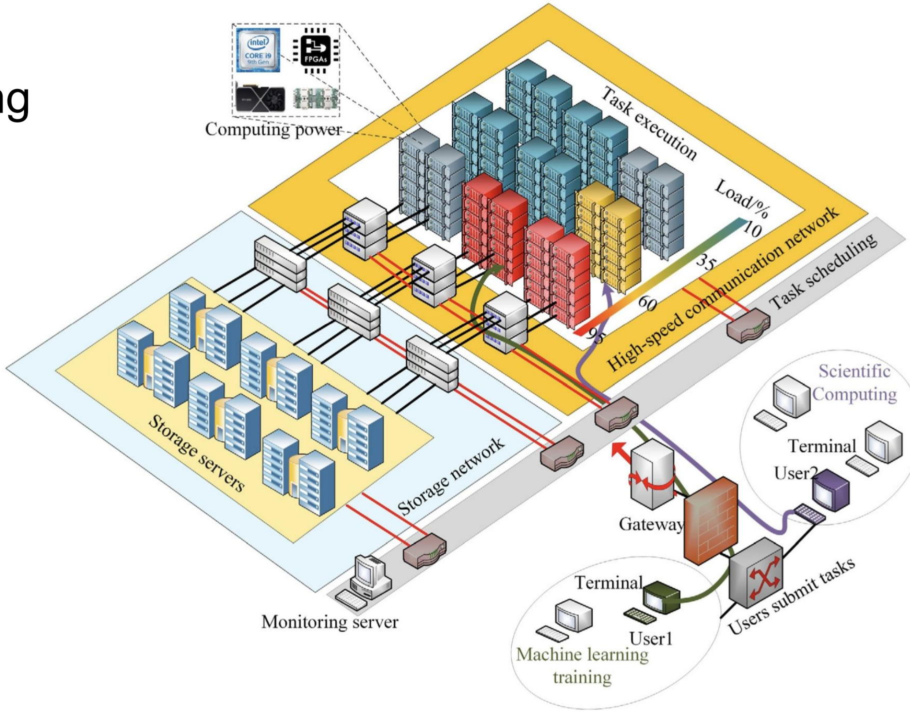
GARLSched: Generative adversarial deep reinforcement learning task scheduling optimization for large-scale high performance computing systems

Daria Yashnova, 5120301/20102

2024

Introduction

Problem statement : develop scheduling system which will be more efficient and adaptive than existing solutions and will improve the performance of large distributed HPC systems by integrating expertise and new learning methods.



Task scheduling for large-scale HPC system

A Rule-based planning algorithm

Principle:
* HPC experts develop rules for task allocation based on the principles of the system

Advantages:
* Ease of implementation

Disadvantages:
* Low flexibility

The DRL-based algorithm

Principle:
* Uses machine learning (reinforcement learning)
* Looks for the optimal task allocation strategy to achieve maximum efficiency

Advantages:
* Flexibility and adaptability

Disadvantages:
* Requires a lot of time and data for training

GARLSched algorithm

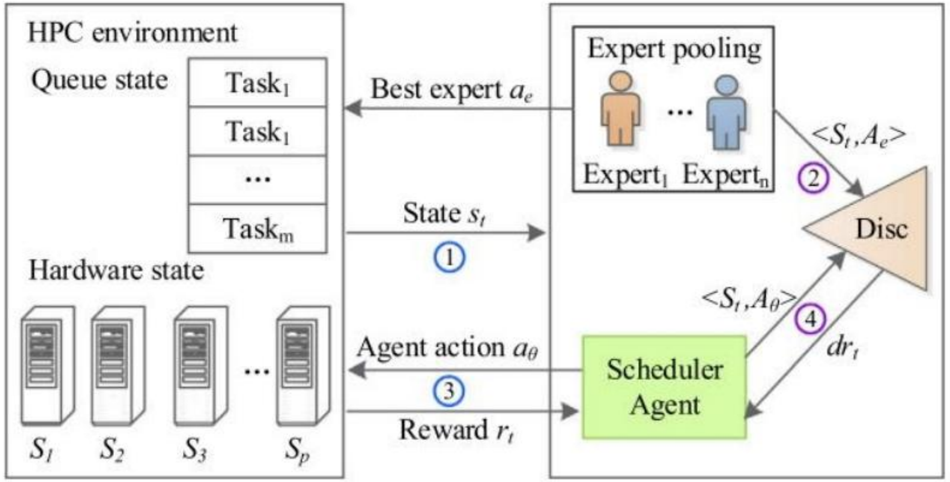
The problem of dynamic planning:

Markov decision process of discrete events: $\langle S, A, P, r, \gamma \rangle$

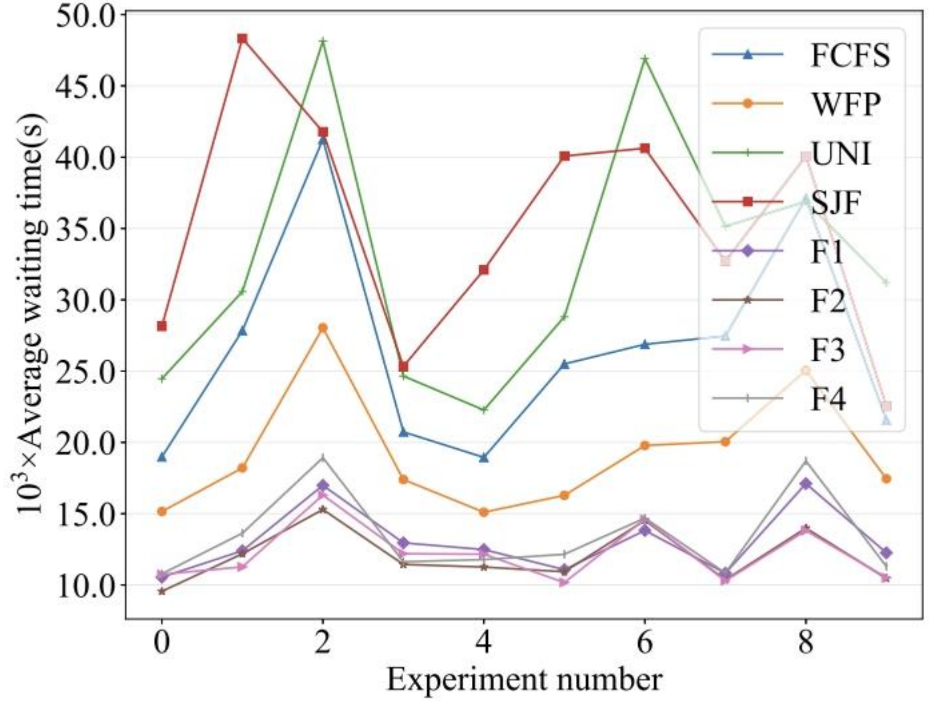
- $S \subseteq R^k$ — *k-dimensional task state*
- $A \subseteq R^q$ — *q-dimensional action* A= $[a_1, a_2, \dots, a_q]$
- $P_s(s_{t+1}|s_t, a_t)$ — *state transition probability*
- $r_s^a = E[r_{t+1}|s_t = s, a_t = a]$ — *reward function*
- $\gamma \in (0, 1)$ — *discount factor for future reward*

GARLSched overview

1. Send state
2. Expert Advice
3. Agent Training
4. Evaluation of actions
5. Agent Improvement
6. Continuous learning



Adversarial generative learning based expert guidance algorithm



The performance comparison of different scheduling policies in expert pooling.

Adversarial generative learning based expert guidance algorithm

$$AVGwt = \frac{1}{M}\sum_{v} t_{wait}^{v}$$

— average waiting time

$$E(d) = max_{\pi_e} min_{\pi_\theta} V_D(\tau_e, \tau_\theta)$$

— ability to classify experts and scheduler agent policies

$$L(d) = -\frac{1}{|\tau_e|}\sum_{(s,a)\in\tau_e} logD(S_t, A_e) - \frac{1}{|\tau_\theta|}\sum_{(s,a)\in\tau_\theta} log(1 - D(S_t, A_\theta))$$

— the expert scheduling behaviors loss function

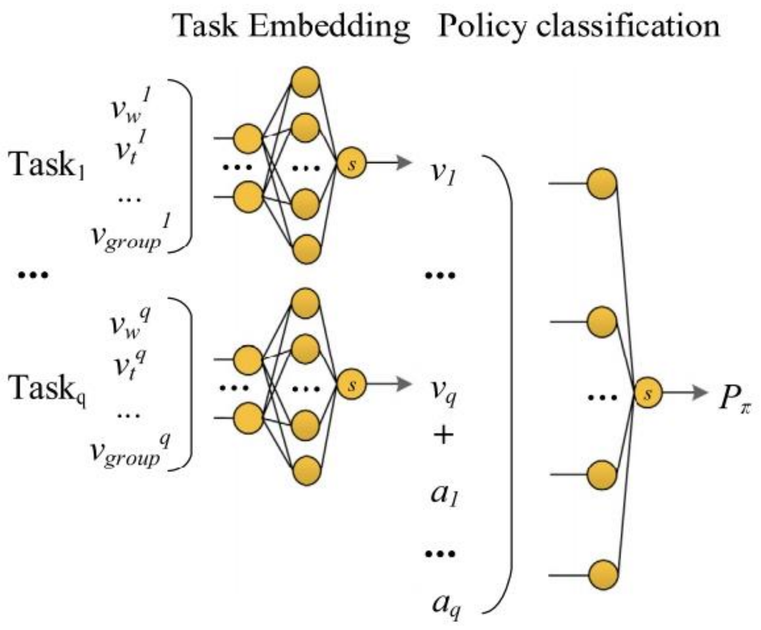
$$\Psi_{GA} = \frac{1}{|\tau_\theta|}\sum_{s_t}\left(v_{GA} - \sum_{(S_t,A_\theta)\in\tau_\theta}\gamma^i(r_t + kdr_t)\right)^2$$

— the loss function of the scheduler agent value network, where k is the weighting factor of the expert guidance algorithm for DRL

$$\Psi_{RL} = \frac{1}{|\tau_\theta|}\sum_{s}\left(v_{RL} - \sum_{(s_t,a_\theta)\in\tau_\theta}\gamma^i r(s_t, a_\theta)\right)^2$$

— the scheduler agent value network loss function

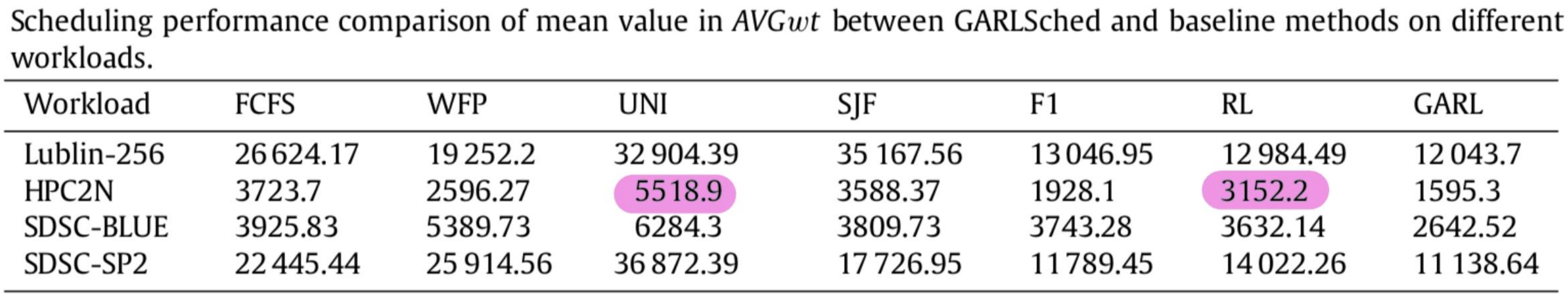
Task embedding based discriminator network



Discriminant network architecture with task embedding and policy classification.

- process timeouts, execution time, server numbers, required resourses
- combine data and actions
- sigmoid function in output

Experimental results and analysis



Scheduling performance comparison of mean value in *AVGwt* between GARLSched and baseline methods on different workloads.

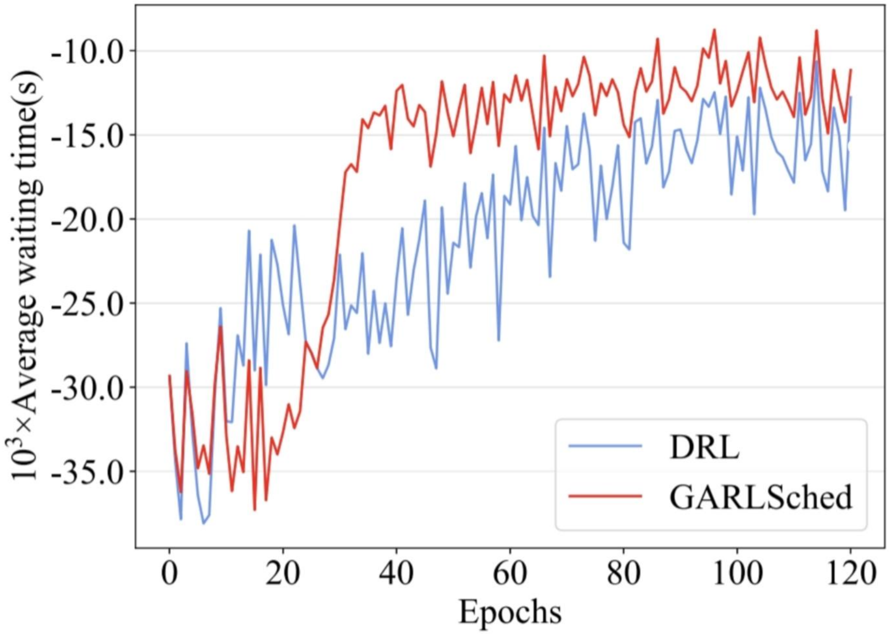
Workload	FCFS	WFP	UNI	SJF	F1	RL	GARL
Lublin-256	26 624.17	19 252.2	32 904.39	35 167.56	13 046.95	12 984.49	12 043.7
HPC2N	3723.7	2596.27	5518.9	3588.37	1928.1	3152.2	1595.3
SDSC-BLUE	3925.83	5389.73	6284.3	3809.73	3743.28	3632.14	2642.52
SDSC-SP2	22 445.44	25 914.56	36 872.39	17 726.95	11 789.45	14 022.26	11 138.64

Average percentage gain: 41.4%

The least improvement in UNI: 71.1%

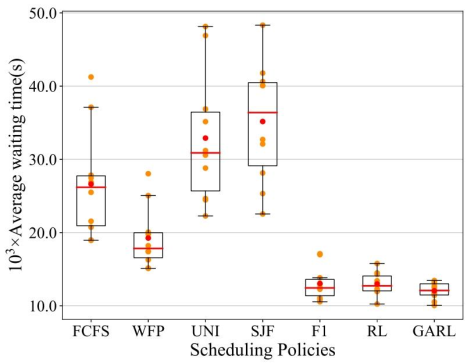
The least improvement in RL: 49.4%

Experimental results and analysis

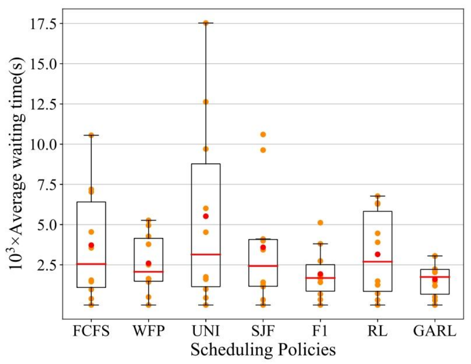


The learning ability comparison between GARLSched proposed in this paper and DRL on the average waiting time.

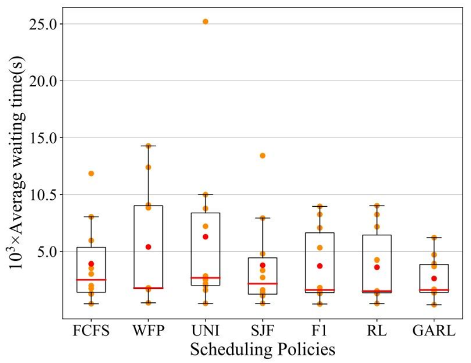
Experimental results and analysis



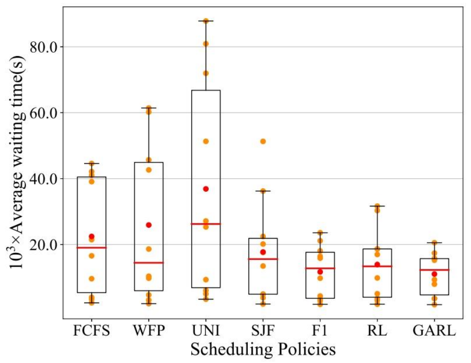
Scheduling for Lublin-256



Scheduling for HPC2N-2002-2.2-cln



Scheduling for SDSC-BLUE-2000-4.2-cln



Scheduling for SDSC-SP2-1998-4.2-cln

Conclusion

This algorithm demonstrates high efficiency, learning from some data and successfully applying to other previously unseen systems. This makes GARLSched a practical tool for real HPC systems.
In the future, it is planned to improve the algorithm by adding a more complex reward function that takes into account energy consumption and quality of service, as well as add support for dependencies between tasks for more precise management.

Thank you for your attention