北京大学

PEKING UNIVERSITY

A Dual-Agent Scheduler for Distributed Deep Learning Jobs on Public Cloud via Reinforcement Learning

Mingzhe Xing, Hangyu Mao, Shenglin Yin, Lichen Pan, Zhengchao Zhang, Zhen Xiao*, Jieyi Long

Background

- Training distributed deep learning (DDL) jobs usually requires powerful and expensive GPUs, and it gradually becomes infeasible to fit them into a private cluster
- Public cloud service providers, e.g., Azure and Alibaba Cloud, have built GPU clusters to gain monetary benefits by training DDL jobs for users
- Two metrics related to user experiences:
  - job completion time
  - training fee

Challenge

北京大学 PEKING UNIVERSITY

- As a typical **online bin-packing problem**, job scheduling is known to be NP-hard
- The mainstream solutions divide this problem into two easier sub-tasks, i.e., **ordering task** and **placement task**
- Heuristic rules:
  - First-in-first-out
  - Shortest-job-first
  - First-fit
  - Best-fit
  - Load-balance
  - ……

Solution

- We analyze and abstract existing ordering and placement methods as a dual-agent structure, and employ reinforcement learning to learn the two agent policies
- Why cooperative dual-agent?
  - The potential **cooperation between the two task policies** can be exploited to further improve the performance
- Why using RL to optimize the dual-agent?
  - System uncertainty:
    - when and what kind of DDL jobs will come.
    - uncertain performance fluctuation caused by future arrived and co-located jobs.
  - RL can naturally adapt to the uncertainty by exploration and exploitation

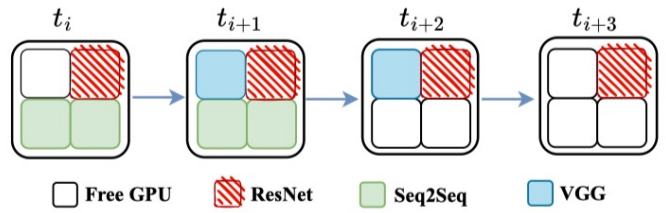


Figure 1: Illustrative example of uncertain performance fluctuation caused by future arrived and co-located jobs in a machine equipped with four GPUs during four timesteps.

Solution

- Can the exploration in Gaussian distribution space in native continuous-space RL well tackle the system uncertainty?
  - We propose a Random Walk Gaussian Process, which can well model the performance similarity and uncertain performance fluctuation

Markov Decision Process Formulation

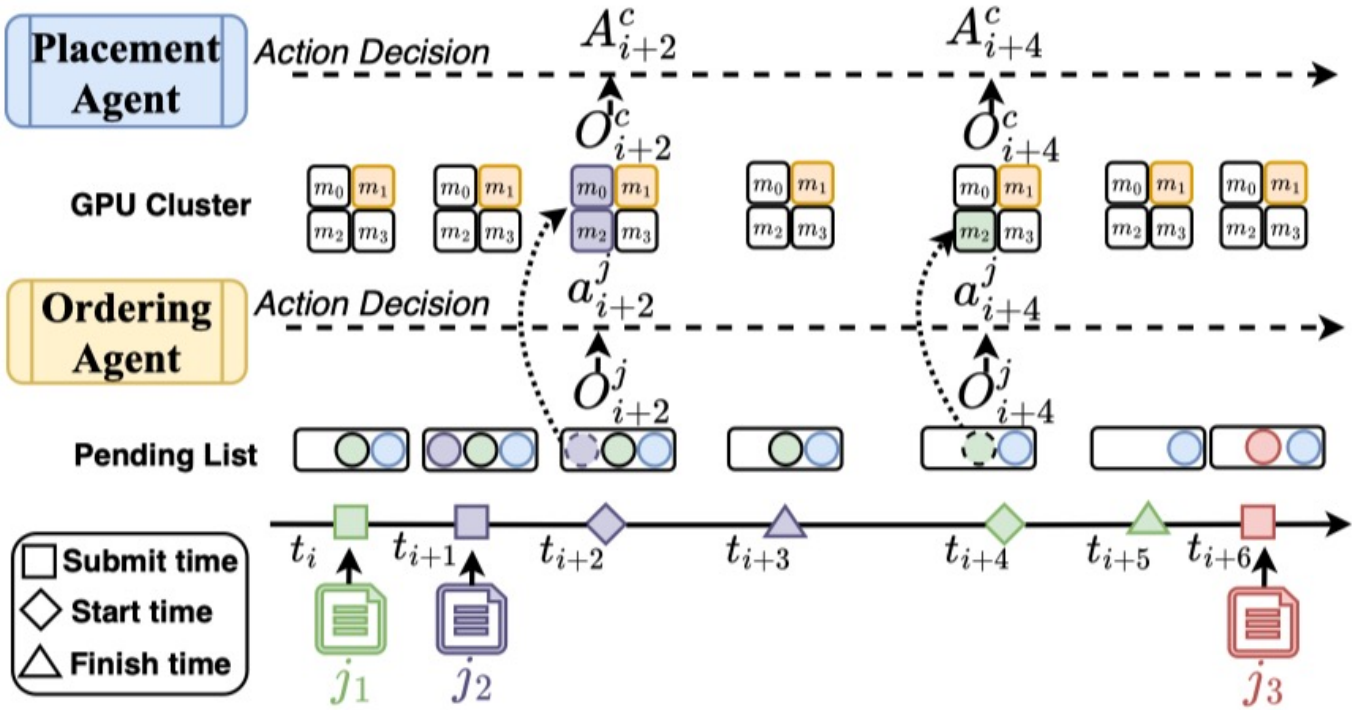


Figure 2: The Dec-MDP formulation of DDL job scheduling. Job j_2 (in purple) is scheduled at t_{i+2} by the ordering agent, and is placed on machines m_0 and m_2 by the placement agent.

- Action:
  - ordering decision
    - ✓ Which job should be scheduled first
  - placement decision
    - ✓ Which machines are suitable for placing a specific job
- Observation:
  - job states
    - ✓ Resource requirements, pending time
  - cluster states
    - ✓ Free resources, utilization ratio
- Reward:

$$r_t = \sum_{i=0}^{N'} \left(\frac{\zeta_i}{r_i^{fee} \times r_i^{jct}} - (1-\zeta_i)\alpha(C_i \times n_i^{GPU}) \right) / N',$$

Model Overview

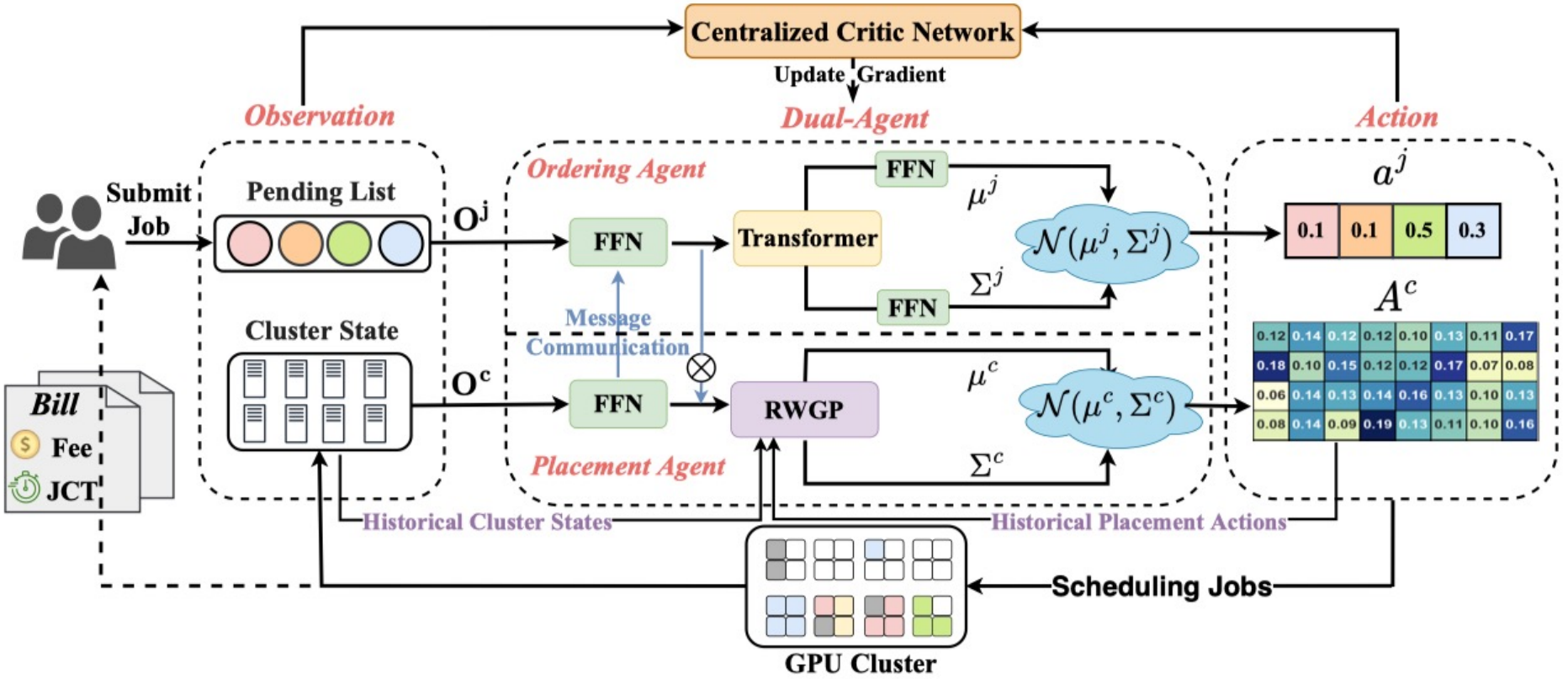


Figure 3: The overall architecture of our proposed DAS model. In this example, four jobs submitted by users are ready to be scheduled on the cluster with eight machines, where each machine is equipped with four GPUs.

Ordering Agent with Communication

北京大学 PEKING UNIVERSITY

- Ordering agent
  - assigning priority scores for jobs
  - the job with a higher priority indicates that it will gain more benefits if scheduled first
- Infer the most proper scheduling order under the current cluster state
  - deliver cluster state to the ordering agent
  - machine number is varying and may be large
  - squeeze-and-communicate $\widetilde{\mathbf{s}^c} = \mathrm{AvgPool}(\mathbf{S}^c)$

$$\widetilde{\mathbf{S}^j} = \delta \cdot f(\widetilde{\mathbf{s}^c}) + (1-\delta) \cdot \mathbf{S}^j$$

$$\delta = \sigma\left(\mathbf{W}_3[\mathbf{S}^j \oplus f(\widetilde{\mathbf{s}^c})] + \boldsymbol{b}_3\right)$$

- Position-aware and self-attentive job ordering
  - jobs arrive at the pending list according to their submission order, which is a key factor for job scheduling
  - Transformer encoder

Placement Agent with Random Walk Gaussian Process

北京大学 PEKING UNIVERSITY

- Placement agent
  - assigning **affinity scores**
  - a higher a_{ik} denotes that job j_i is more suitable to be placed on machine m_k
- Two intuitive placement principles
  1. job instances should first be placed on the machines with **similar performances** to ensure that no machine could become the performance bottleneck
  2. should be aware of the **uncertain future performance fluctuation** caused by resource contention

Placement Agent with Random Walk Gaussian Process

- **Principle 1:** Random walk kernel in non-Euclidean space
  - abstract the cluster as a graph G_t at timestep t
    - node: machine
    - edge: communication topology
  - a random walker selects the next visited machine node according to transition matrix $P(i)$
  - a larger $P(i)_{kl}$ denotes that machines m_k and m_l are more similar for placing job j_i and there is a higher probability to transit from m_k to m_l

Placement Agent with Random Walk Gaussian Process

- **Principle 2:** Random walk kernel in Gaussian Process
  - give a posterior distribution rather than point estimation
  - random walk kernel is symmetric and positive semi-definite
  - uncertainty from topological view

$$k_{\mathrm{RW}}(G_1, G_2) = \sum_{i,k=1}^{|V_\times|} \left[\sum_{\ell=0}^{\infty} \lambda^\ell \mathbf{P}_\times^\ell \right]_{ik}$$

- Posterior distribution of RWGP

$$\boldsymbol{\mu}_t^c = k_{\mathrm{RW}}(\widehat{\mathbf{S}}_t^c, \mathbf{X}) \left[k_{\mathrm{RW}}(\mathbf{X}_t, \mathbf{X}_t) + \sigma^2 \mathbf{I} \right]^{-1} \mathbf{Y}_t \tag{13}$$

$$\boldsymbol{\Sigma}_t^c = k_{\mathrm{RW}}(\widehat{\mathbf{S}}_t^c, \widehat{\mathbf{S}}_t^c) - k_{\mathrm{RW}}(\widehat{\mathbf{S}}_t^c, \mathbf{X}_t) \left[k_{\mathrm{RW}}(\mathbf{X}_t, \mathbf{X}_t) + \sigma^2 \mathbf{I} \right]^{-1} k_{\mathrm{RW}}(\mathbf{X}_t, \widehat{\mathbf{S}}_t^c). \tag{14}$$

- Action Sampling

$$\mathbf{A}_t^c \sim \mathcal{N}(\boldsymbol{\mu}_t^c, \boldsymbol{\Sigma}_t^c).$$

Optimization with MAPPO

- Temporal-Difference Error

$$\mathcal{L}_t^{TD} = \left(Q_\phi(\mathbf{O}_t^j, \mathbf{O}_t^c, \boldsymbol{a}_t^j, \mathbf{A}_t^c) - y_t\right)^2$$

$$y_t = r_t + \gamma Q_{\phi_{old}}\left(\mathbf{O}_{t+1}^j, \mathbf{O}_{t+1}^c, \pi_{\theta_{old}}(\mathbf{O}_{t+1}^j, \mathbf{O}_{t+1}^c)\right),$$

- Policy gradient

$$\mathcal{L}^{Actor} = -\left(\mathcal{L}^{PG} + \lambda H(\pi_\theta)\right)$$

$$\mathcal{L}^{PG} = \mathbb{E}_t\left[\min\left(\text{ratio}, \text{clip}(\text{ratio}, 1-\epsilon, 1+\epsilon)\right)B_t\right]$$

$$\text{ratio} = \mathbb{E}_t\left[\exp\left(\log(\pi_\theta) - \log(\pi_{\theta_{old}})\right)\right],$$

$$\log(\pi_{\theta^j}) = -\frac{1}{2}\left[(\boldsymbol{a}^j - \boldsymbol{\mu}^j)^2 + N\log(2\pi)\right] - \log(|\Sigma^j|)$$

Experiments

- Simulator: a well-established simulation environment used in many DDL scheduling research work
- Dataset: Alibaba GPU Cluster Trace
- Baselines:
  - heuristic-based schedulers
  - meta-heuristic-based schedulers
  - RL-based scheduler
- Evaluation metrics

$$JCT = \sum_{i=0}^{|\mathcal{J}|} \frac{r_i^{jct}}{|\mathcal{J}|}, \qquad Fee = \sum_{i=0}^{|\mathcal{J}|} \frac{r_i^{fee}}{|\mathcal{J}|}$$

Experiments (Performance Comparisons & Ablation Study)

北京大学 PEKING UNIVERSITY

Table 1: Performance comparisons with baselines and ablation study for four variants of our method. The JCT is in minutes and Fee is in dollars. The best and second best are in bold and underline, respectively. Please note that the metrics are presented at job level, and even marginal improvements can lead to significant benefits regarding all the clusters running on the cloud.

Algorithms		Small		Medium		Large		Average	
		JCT	Fee	JCT	Fee	JCT	Fee	JCT	Fee
Heuristic-based	FIFO-FirstFit	720.228	51.685	171.282	51.684	110.316	51.219	333.942	51.529
	FIFO-Load Balance	715.776	51.559	162.102	50.900	112.290	51.112	330.054	51.189
	DRF-FirstFit	862.242	52.055	164.778	51.354	109.098	52.325	378.708	51.911
	DRF-Load Balance	852.984	51.798	173.106	51.908	113.556	51.924	379.884	51.877
	Tetris	813.480	51.668	176.688	52.567	113.364	53.109	367.842	52.448
Meta-heuristic-based	MALO	708.858	51.675	182.862	51.129	111.660	51.233	334.458	51.346
	MOSA	759.222	51.783	173.400	51.998	109.422	51.933	347.346	51.905
RL-based	DL^2	721.083	51.166	160.100	50.970	110.215	51.080	330.468	51.072
	RIFLING	717.576	51.160	176.676	50.980	109.872	51.039	334.708	51.060
Ours	DA	697.732	51.106	173.354	51.097	108.821	51.025	326.636	51.076
	DA+MC	680.861	50.952	163.456	50.941	109.854	50.701	318.057	50.865
	DA+RWGP	686.894	51.029	157.071	50.856	**107.896**	50.439	317.287	50.684
	DA+MC+RWGP (DAS)	**638.046**	**50.484**	**155.003**	**50.019**	108.700	**50.354**	**300.584**	**50.286**

Parameter Tuning

- Two important parameters in the ordering and placement agents:
  - the number of attention head h
  - the length of historical data τ
- The JCT and Fee of our model consistently outperform RIFLING

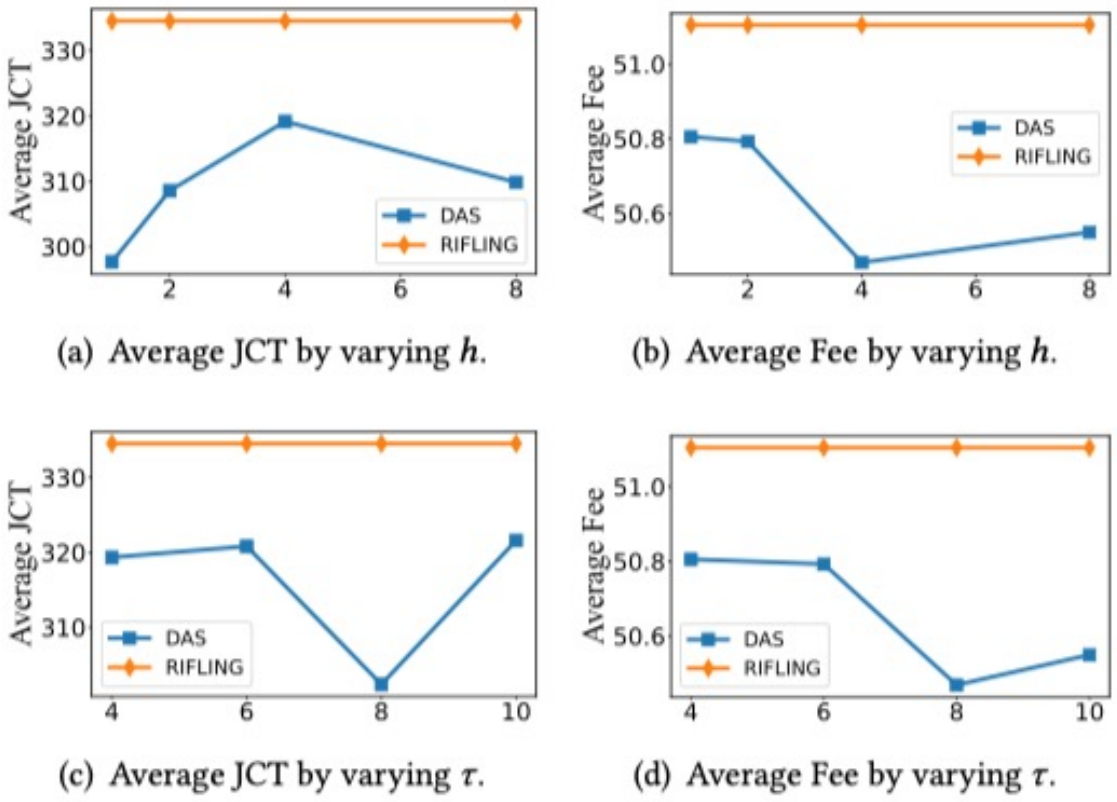
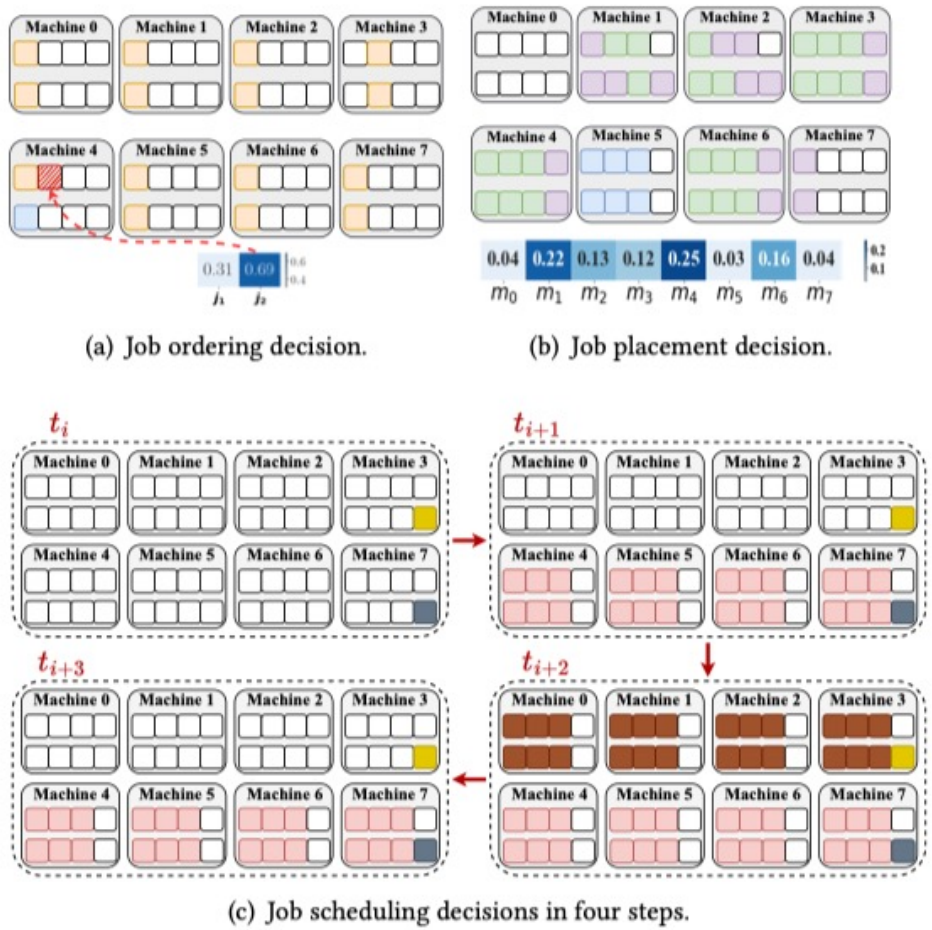


Figure 5: Performance tuning by varying h and τ. Our method consistently outperforms RIFLING on both metrics.

Case Study

- The communication mechanism is able to convey the cluster state to the ordering agent, so as to make proper ordering decisions
- The RWGP is able to detect the performance similarities of machines and the potential resource contention



(a) Job ordering decision. (b) Job placement decision.

(c) Job scheduling decisions in four steps.

Figure 6: Case study on the Small cluster to show the effectiveness of the ordering and placement policies of our model.

Conclusion

北京大学 PEKING UNIVERSITY

- A distributed deep learning job scheduler is responsible for reducing both job completion time and training cost.
- Existing heuristic and reinforcement learning based methods either ignore or cannot handle the system uncertainty and policy cooperation.
- Contributions:
  - We propose a dual-agent structure to learn the ordering and placement policies.
  - We design an ordering agent with efficient communication mechanism.
  - For the placement agent, we propose a novel Random Walk Gaussian Process to model the performance similarities and performance uncertainty from topological view.

北京大学

PEKING UNIVERSITY

Thanks