INTELLIGENT JOB SCHEDULING FOR NEXT GENERATION HPC SYSTEMS

Yuping Fan‡, Zhiling Lan (advisor)‡, Michael E. Papka (advisor)⋆

‡Illinois Institute of Technology, ⋆Argonne National Laboratory

Motivation & Challenges

- ♢ Next generation HPC systems are equipped with **heterogeneous global and local resources**, e.g., burst buffer and GPU.
- ♢ **Diverse applications** such as data- and memory-intensive applications are emerging in HPC systems.
- ♢ HPC job scheduler plays a crucial role in efficient use of resources.
- ♢ Existing scheduling solutions are mainly **heuristic methods**, leading to degraded system and job performance.
- ♢ More **intelligent** job scheduling solutions are needed to keep up with the pace of rapid HPC systems and applications evolution.

Key Contributions

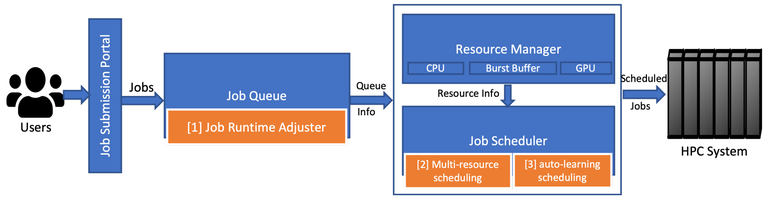


Fig. 1: Framework Overview

✓ **Improving job runtime estimates by exploiting machine learning algorithm** [1]

- Utilize historical job and user information to predict job runtime estimates
- Leverage Tobit model to largely reduce the prediction underestimation rate
- Improve system utilization and average job wait time

✓ **Optimizing multi-resource scheduling** [2]

- Formulate the multi-resource scheduling problem into a multi-objective optimization (MOO) problem
- Rapidly solve the MOO problem through multi-objective genetic algorithm
- Significantly improve user- and system-level performance when CPU and burst buffer need to be scheduled
- Easily embrace other types of global and local resources

✓ **Automatically learning customized scheduling policies via reinforcement learning** [3]

- Leverage reinforcement learning algorithms to learn scheduling policies customized to specific systems and workloads
- Explore two types of reinforcement learning algorithms, policy gradient and deep Q-learning, are explored
- Design three-phase learning process largely improves learning efficiency
- Dynamically adjust the auto-learned policies to handle dramatic workload changes

Primary Publications

[1] Y. Fan, Z. Lan, T. Childers, P. Rich, W. Allcock, and M. Papka. "Trade-Off Between Prediction Accuracy and Underestimation Rate in Job Runtime Estimates". In: *CLUSTER*. 2017.

[2] Y. Fan, Z. Lan, P. Rich, W. E. Allcock, M. E. Papka, B. Austin, and D. Paul. "Scheduling Beyond CPUs for HPC". In: *HPDC*. 2019.

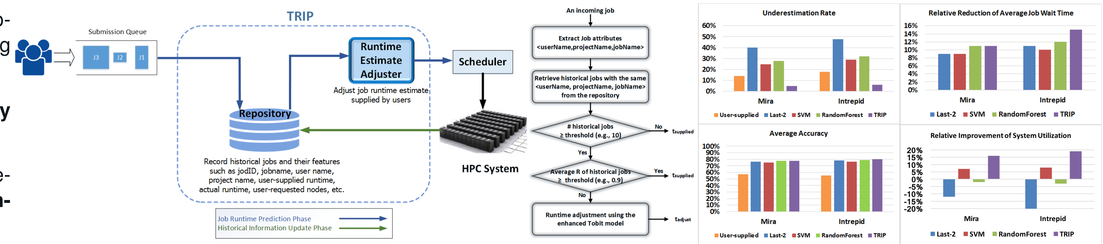
[3] Y. Fan, P. Rich, W. Allcock, M. Papka, and Z. Lan. "Deep Reinforcement Agent for Scheduling in HPC". In: *IPDPS*. 2021.

Machine Learning based Adjustment on Job Runtime Estiamtes

✗ Existing HPC runtime estimate adjuster only focuses on improving prediction accuracy, resulting in **high underestimation rate**.

✗ Underestimated jobs are killed, which **adversely affect scheduling performance**.

✓ We present TRIP, leveraging Tobit model, to predict job runtime with **high accuracy and low underestimation rate**.



Multi-Resource Scheduling

✗ Existing HPC schedulers are mainly **CPU-centric**, making scheduling decisions solely based on the application's processor footprint.

✗ Such a CPU-centric scheduling can easily result in **poor application performance** and waste of system resources.

✓ We present **BBSched** for HPC scheduling under multiple resource constraints.

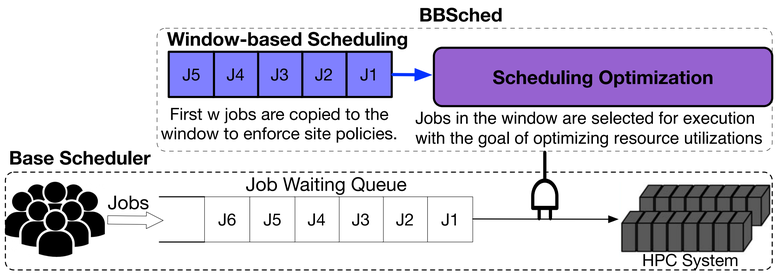


Fig. 6: BBSched design

✓ Formulate the multi-resource problem into **multi-objective optimization problem**

✓ Solve the MOO problem via multi-objective **genetic algorithm**

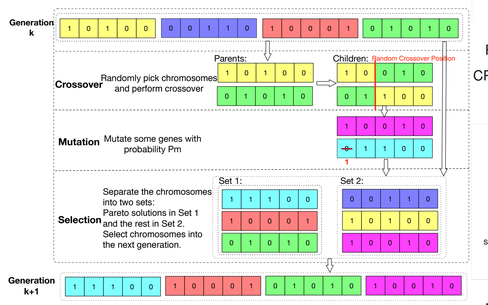


Fig. 7: An example of solving scheduling problem via genetic algorithm.

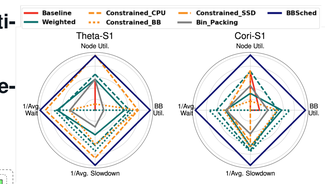


Fig. 8: The overall performance on scheduling CPU and burst buffer. BBSched achieves the best performance.

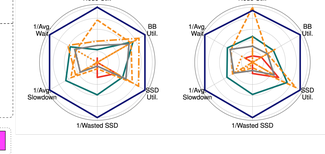


Fig. 9: The overall performance on scheduling CPU, burst buffer and local SSD.

Deep Reinforcement Agent for HPC Job Scheduling

✗ Traditionally, system managers manually design and tune scheduling policies based on their **experiences on specific systems and workloads**.

✗ The rapidly changing workloads and systems makes the **manual policy design impossible**.

✓ We design **DRAS, an reinforcement learning agent** to automatically learn high-quality and customized policies for HPC jobs

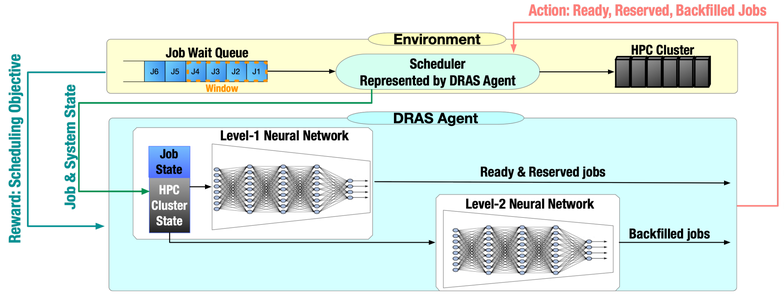


Fig. 10: DRAS design

✓ Build **hierarchical neural network** to incorporate reservation and backfilling decisions

✓ **Prevent large jobs from starvation**

✓ The **three phase learning process** allows DRAS to reach high-quality policies faster

✓ **Adapt to workload changes** by continuously learning

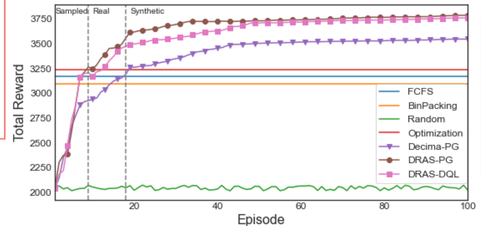


Fig. 11: DRAS achieves the best learning efficiency.

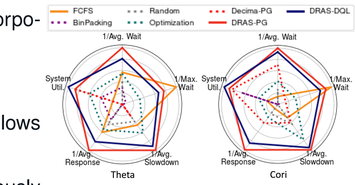


Fig. 12: Compare overall performance. The larger the area, the better the performance. DRAS yields the best performance.

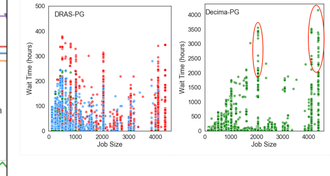


Fig. 13: Comparing with the standard reinforcement learning agent, DRAS agent prevents large jobs from starvation.