SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning

Di Zhang[1], Dong Dai[1], Bing Xie[2]

[1]University of North Carolina at Charlotte

[2]Oak Ridge National Laboratory

CHARLOTTE
COLLEGE OF COMPUTING AND INFORMATICS

OAK RIDGE
National Laboratory

Motivation & Background

SchedInspector Design

Evaluation & Analysis

Conclusion

Motivation & Background

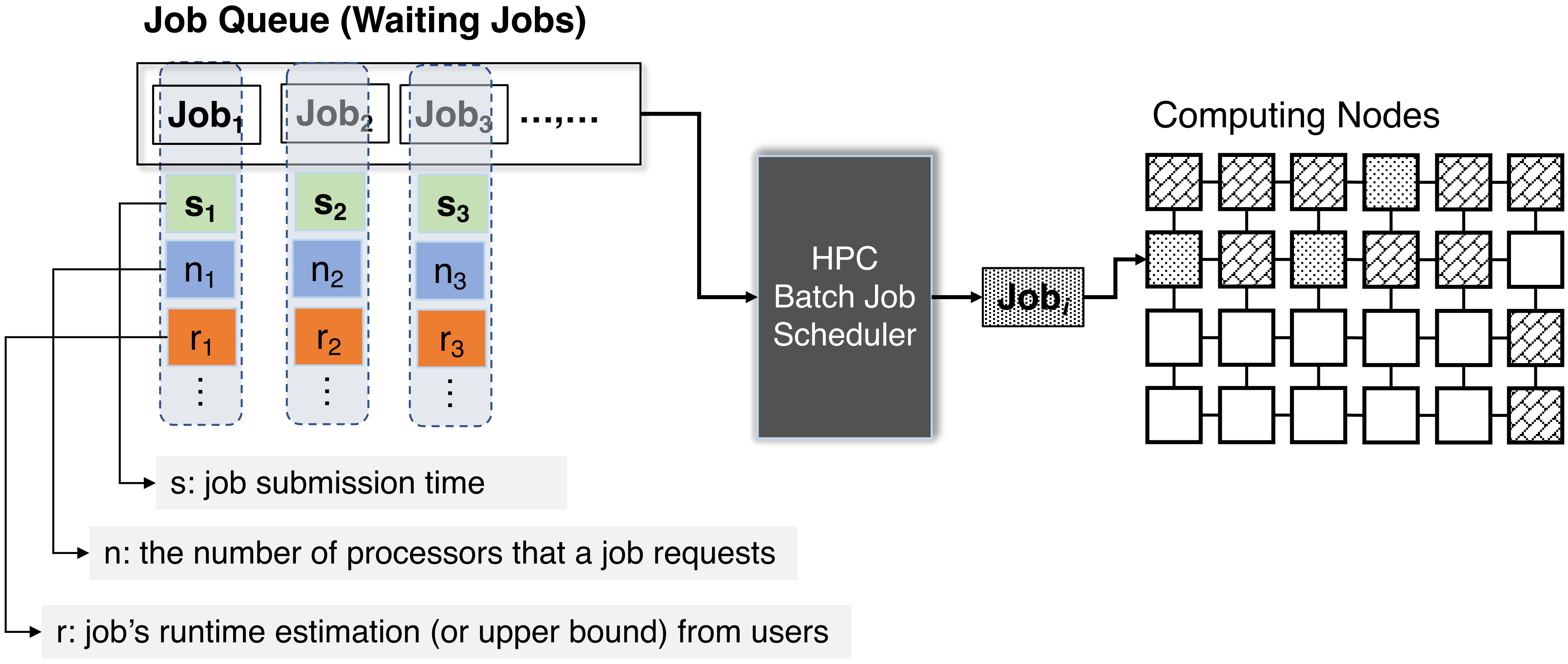
SchedInspector Design

Evaluation & Analysis

Conclusion

- Introduction of HPC batch job schedulers
- Challenges of existing schedulers
- Background of Reinforcement Learning

HPC Batch Job Scheduler

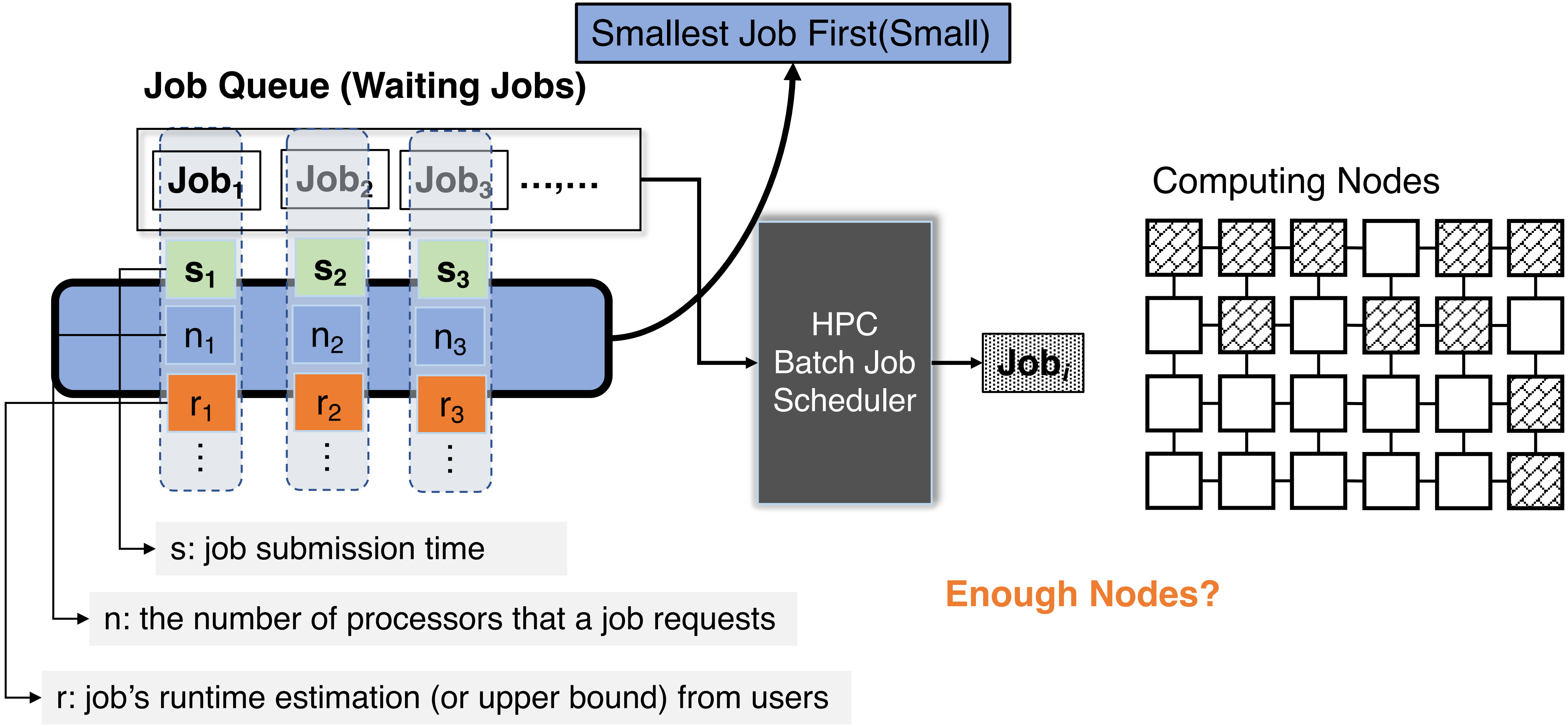


s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

HPC Batch Job Scheduler



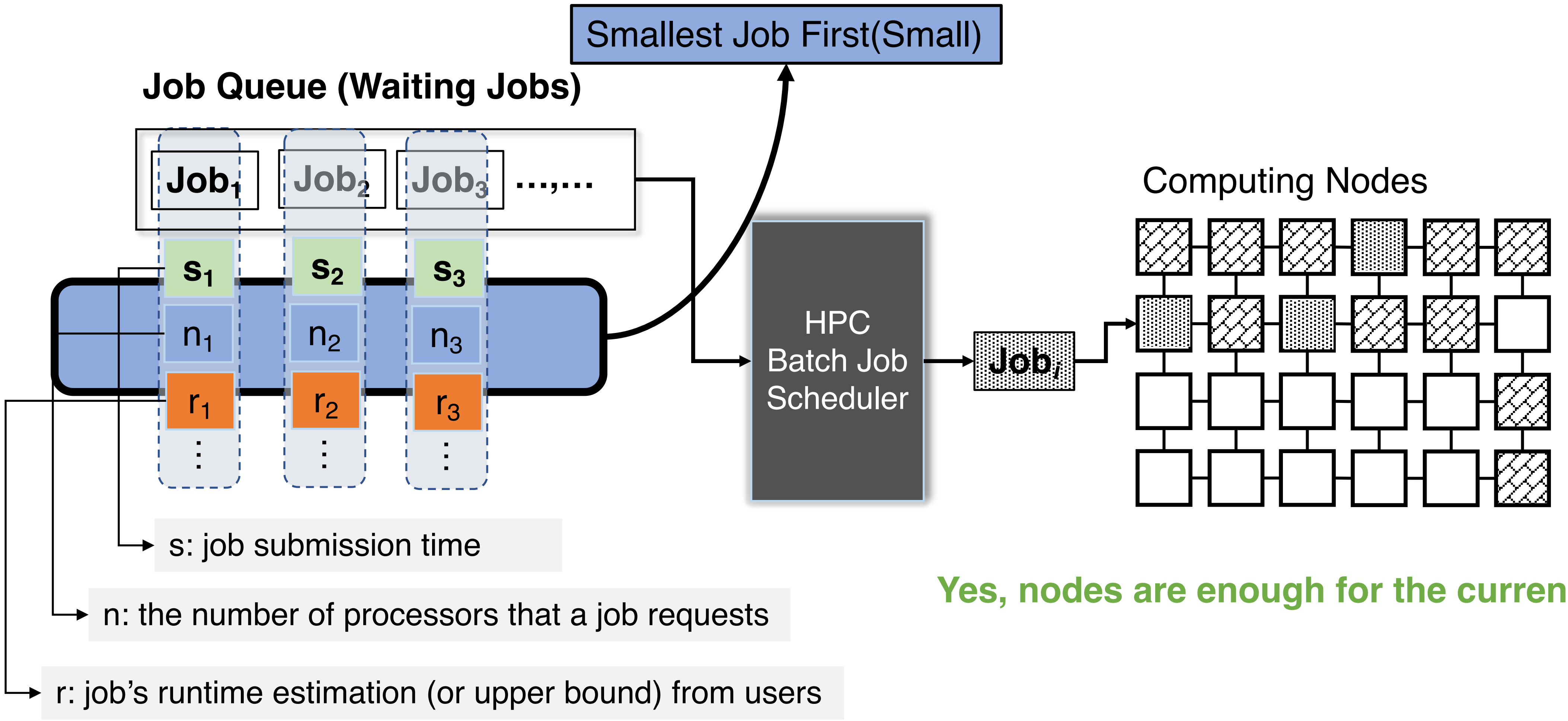
s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

Enough Nodes?

HPC Batch Job Scheduler



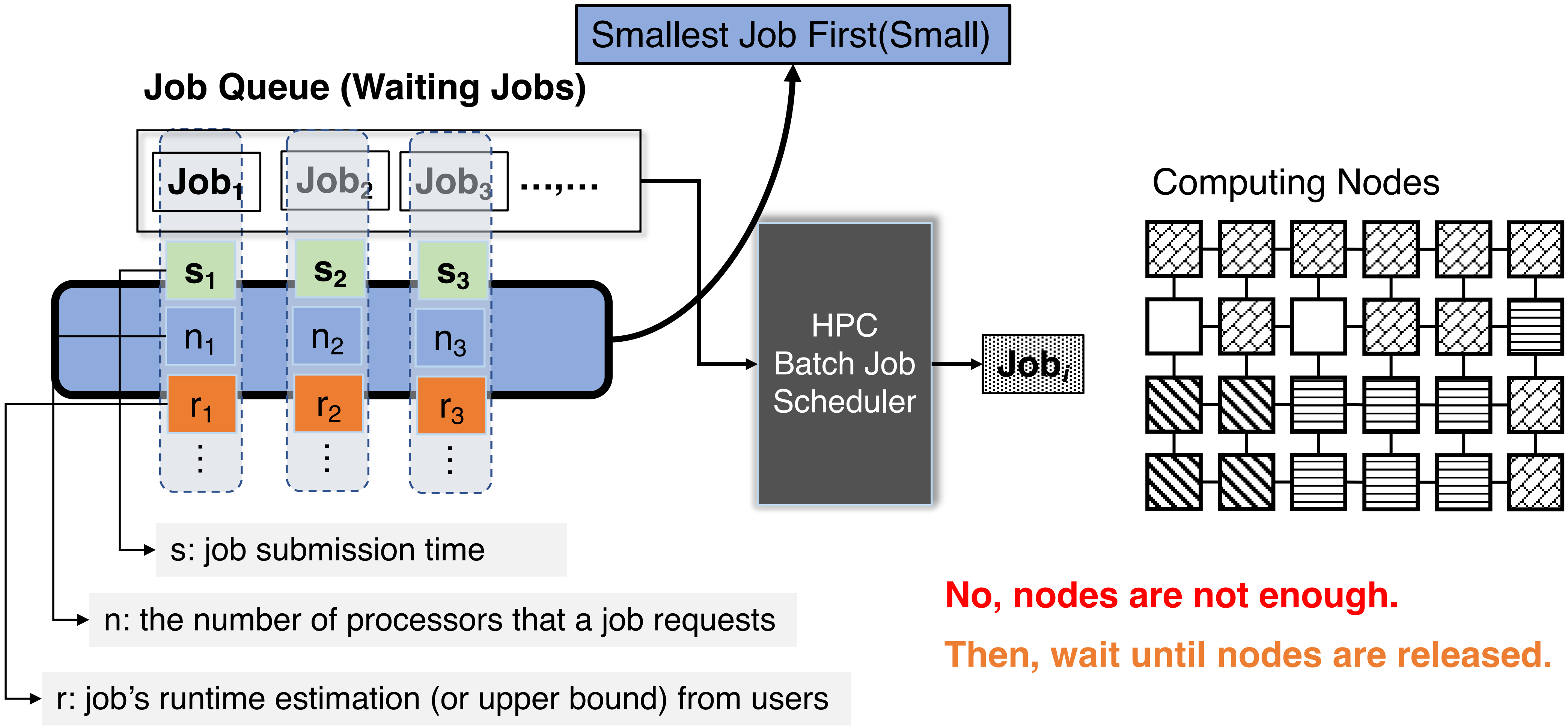
s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

Yes, nodes are enough for the current job.

HPC Batch Job Scheduler



s: job submission time

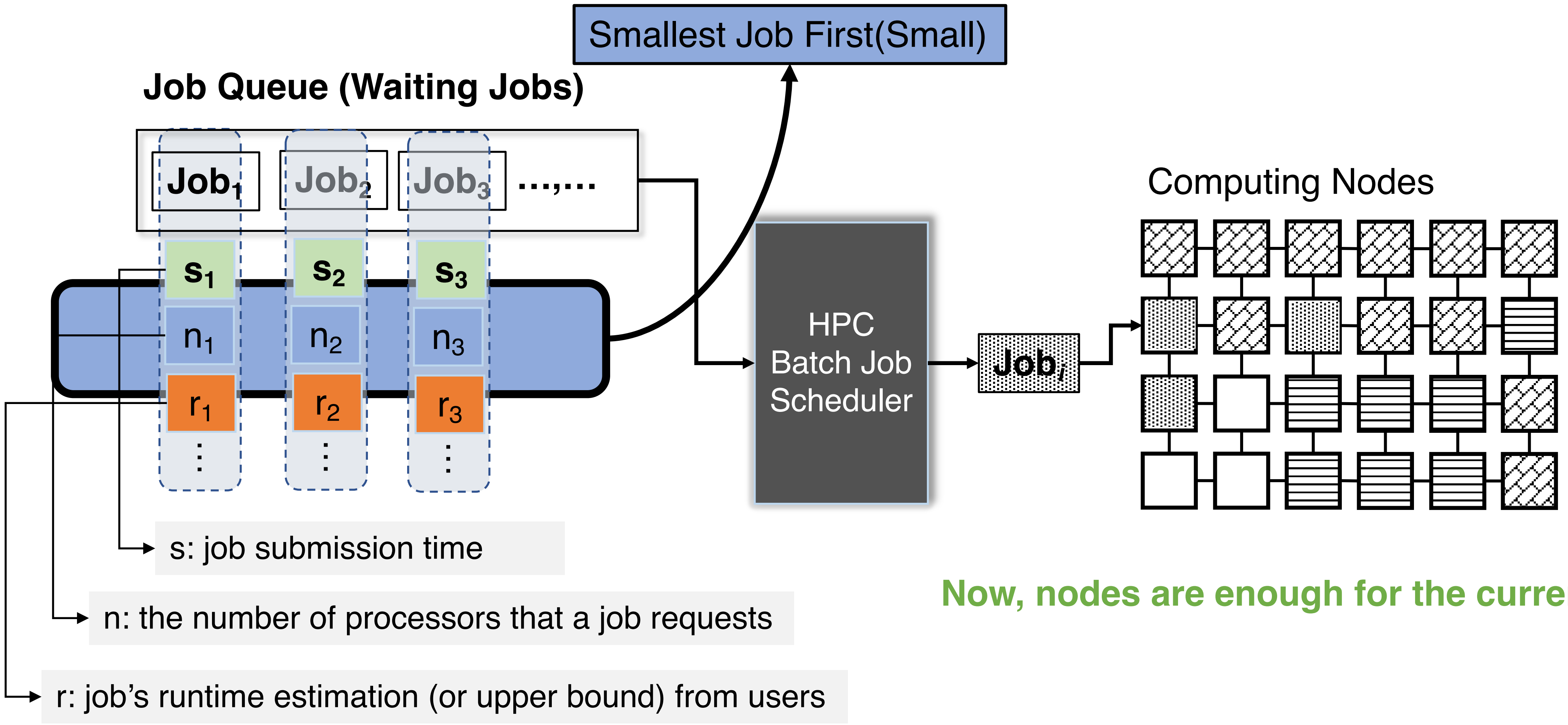
n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

No, nodes are not enough.

Then, wait until nodes are released.

HPC Batch Job Scheduler



s: job submission time

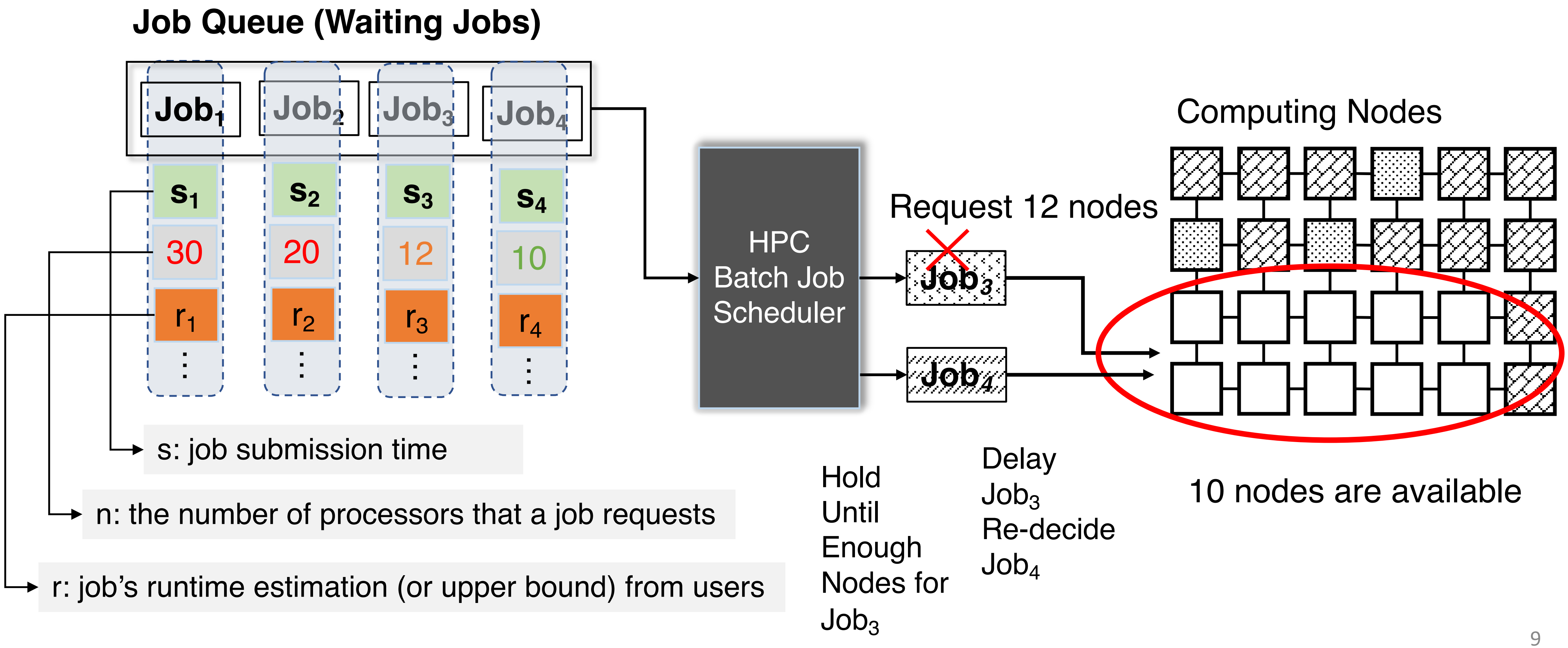
n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

Now, nodes are enough for the current job.

Motivation Example

Smallest Job First(Small)



Job Queue (Waiting Jobs)

Job_1 Job_2 Job_3 Job_4

s_1 s_2 s_3 s_4

30 20 12 10

r_1 r_2 r_3 r_4

s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

HPC Batch Job Scheduler

Request 12 nodes

Job_3

Job_4

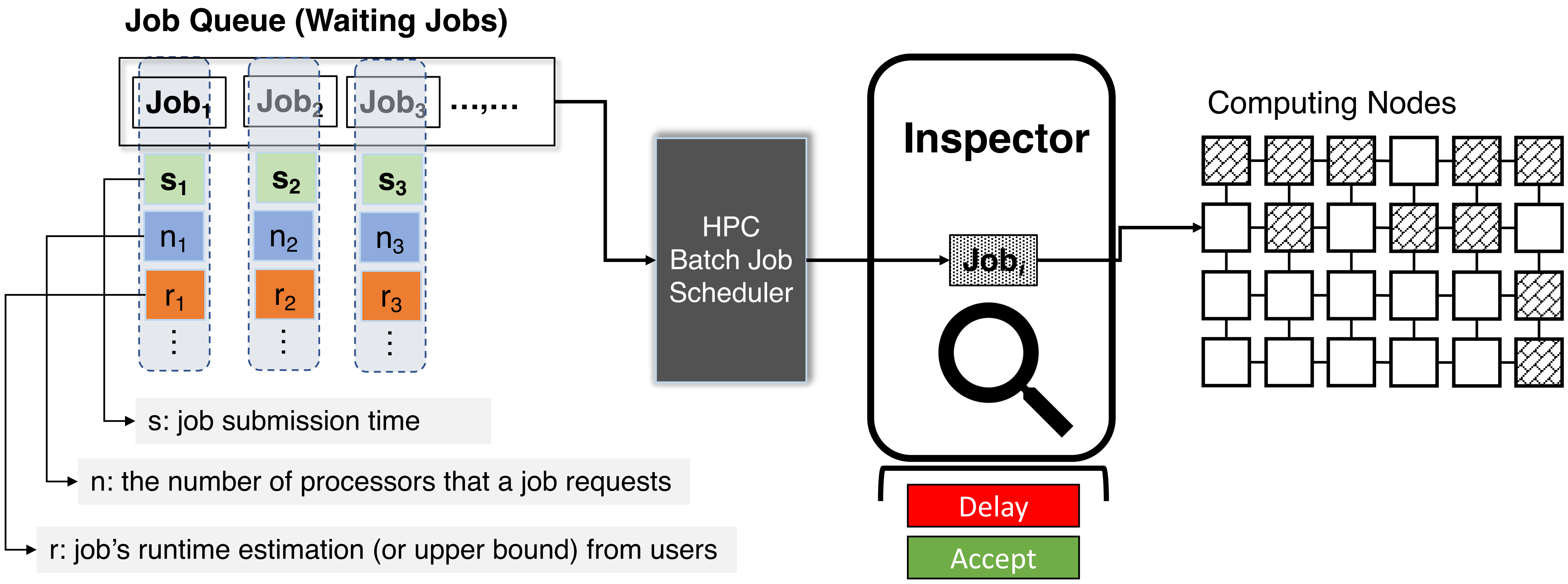
Computing Nodes

Hold Until Enough Nodes for Job_3

Delay Job_3 Re-decide Job_4

10 nodes are available

Motivation



Job Queue (Waiting Jobs)

s: job submission time

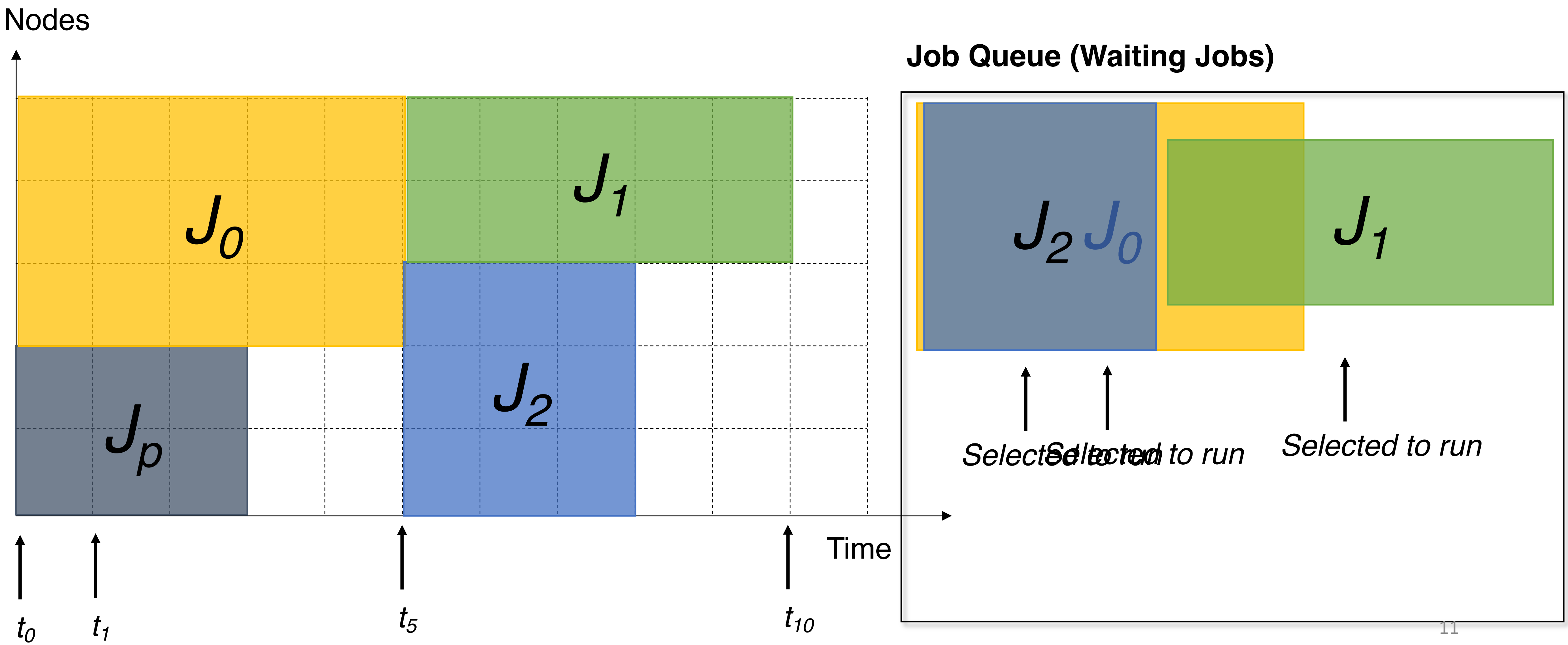
n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

Shortest Job First(SJF)

Without Inspector

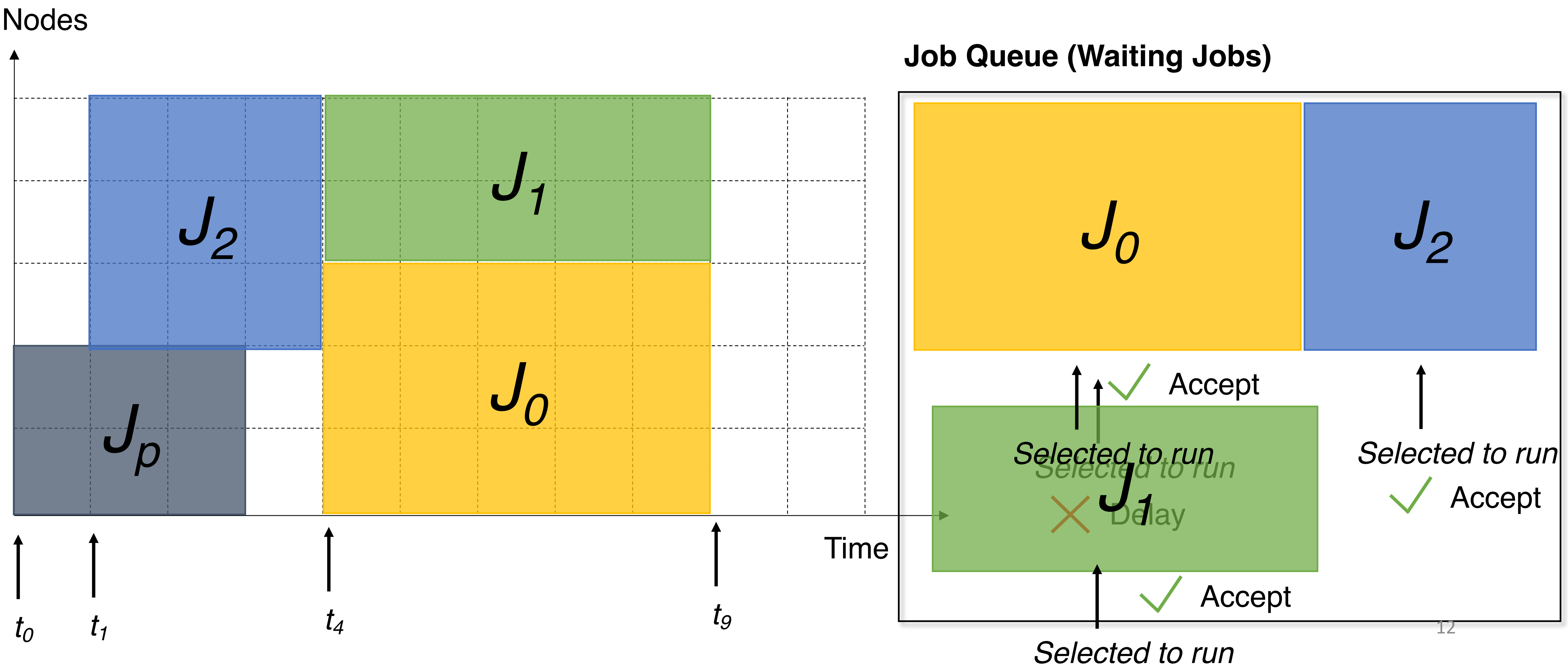
Completion Time : 10



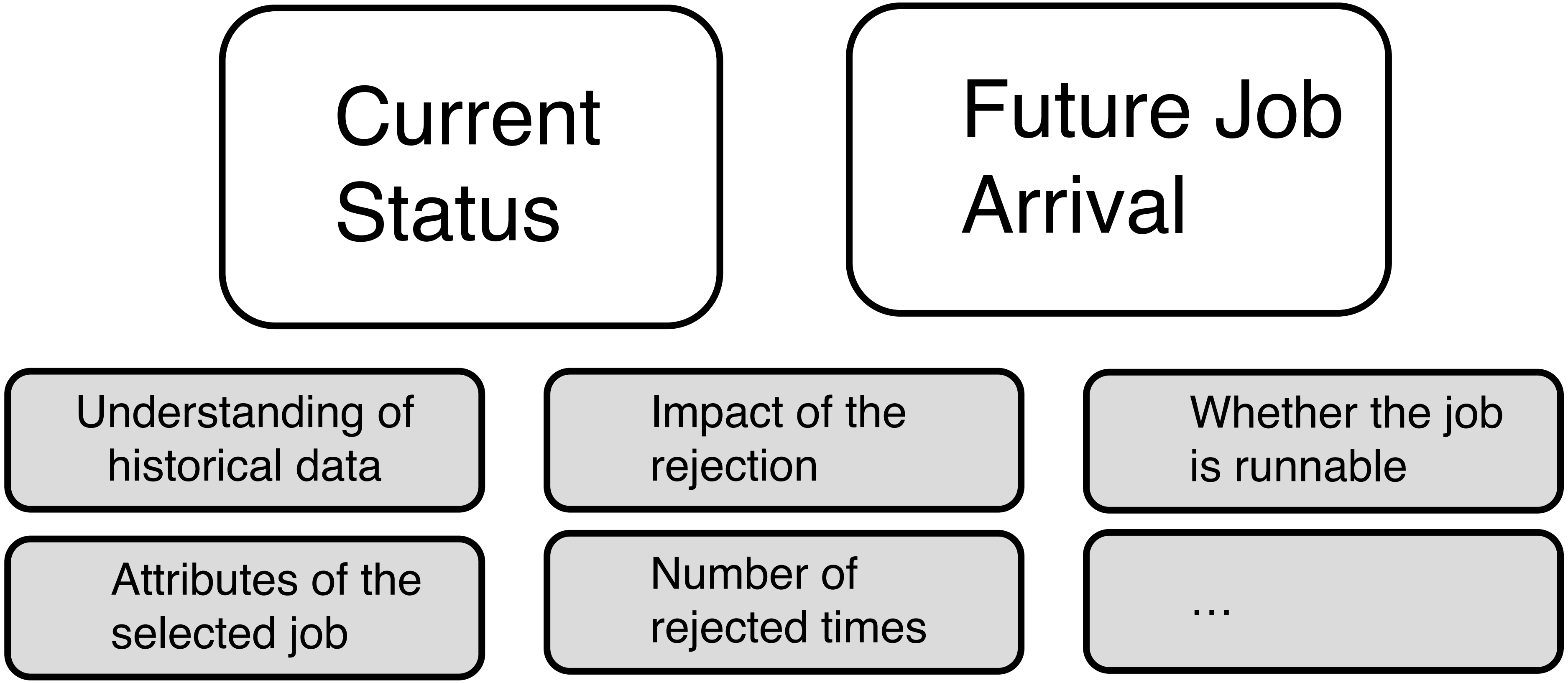
Shortest Job First(SJF)

With Inspector

Completion Time : 9



Challenges



Current Status

Future Job Arrival

Understanding of historical data

Impact of the rejection

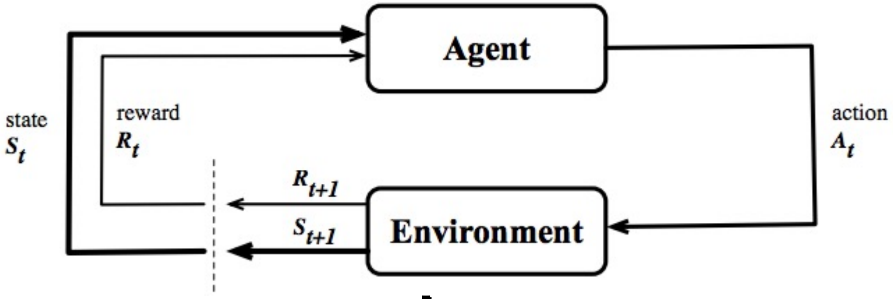
Whether the job is runnable

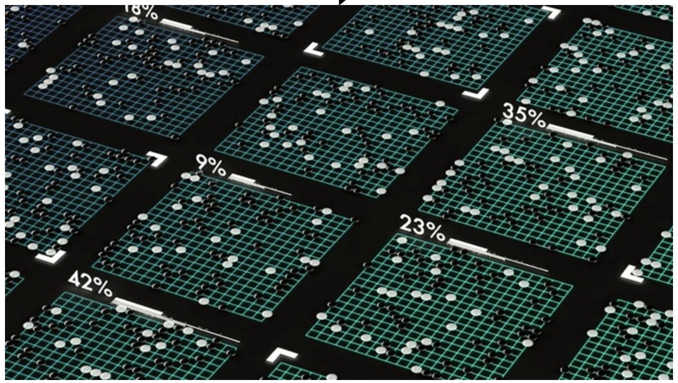
Attributes of the selected job

Number of rejected times

…

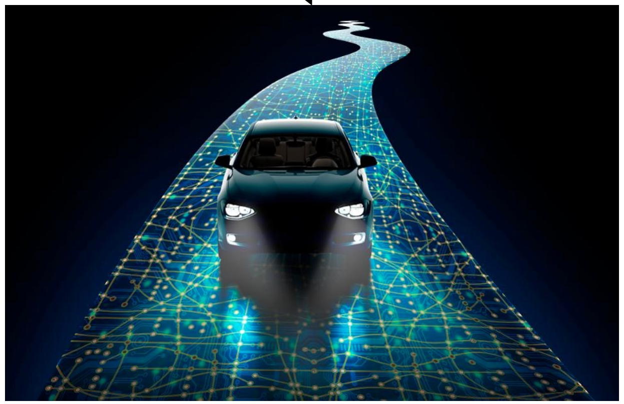
Reinforcement Learning





David Silver, et. al. Mastering the game of Go with deep neural networks and tree search, Nature vol. 529 (2016)

Volodymyr Mnih, et. al. Playing Atari with Deep Reinforcement Learning arXiv:1312.5602 (cs)



From https://www.selfdrivingcars360.com/how-autonomous-vehicles-fit-into-our-ai-enabled-future/

Motivation & Background

SchedInspector Design

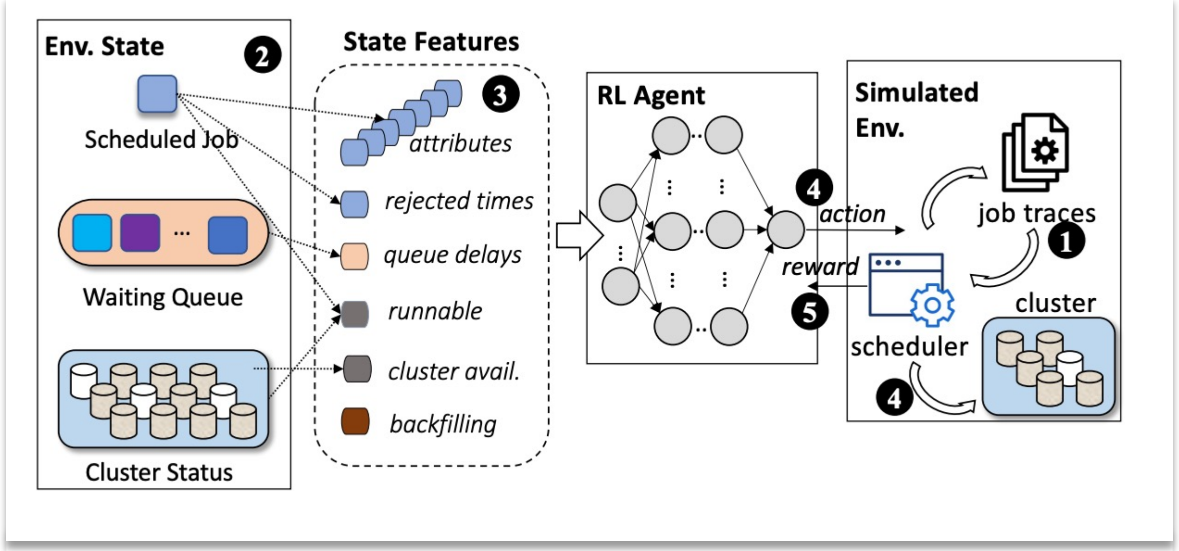
Evaluation & Analysis

Conclusion

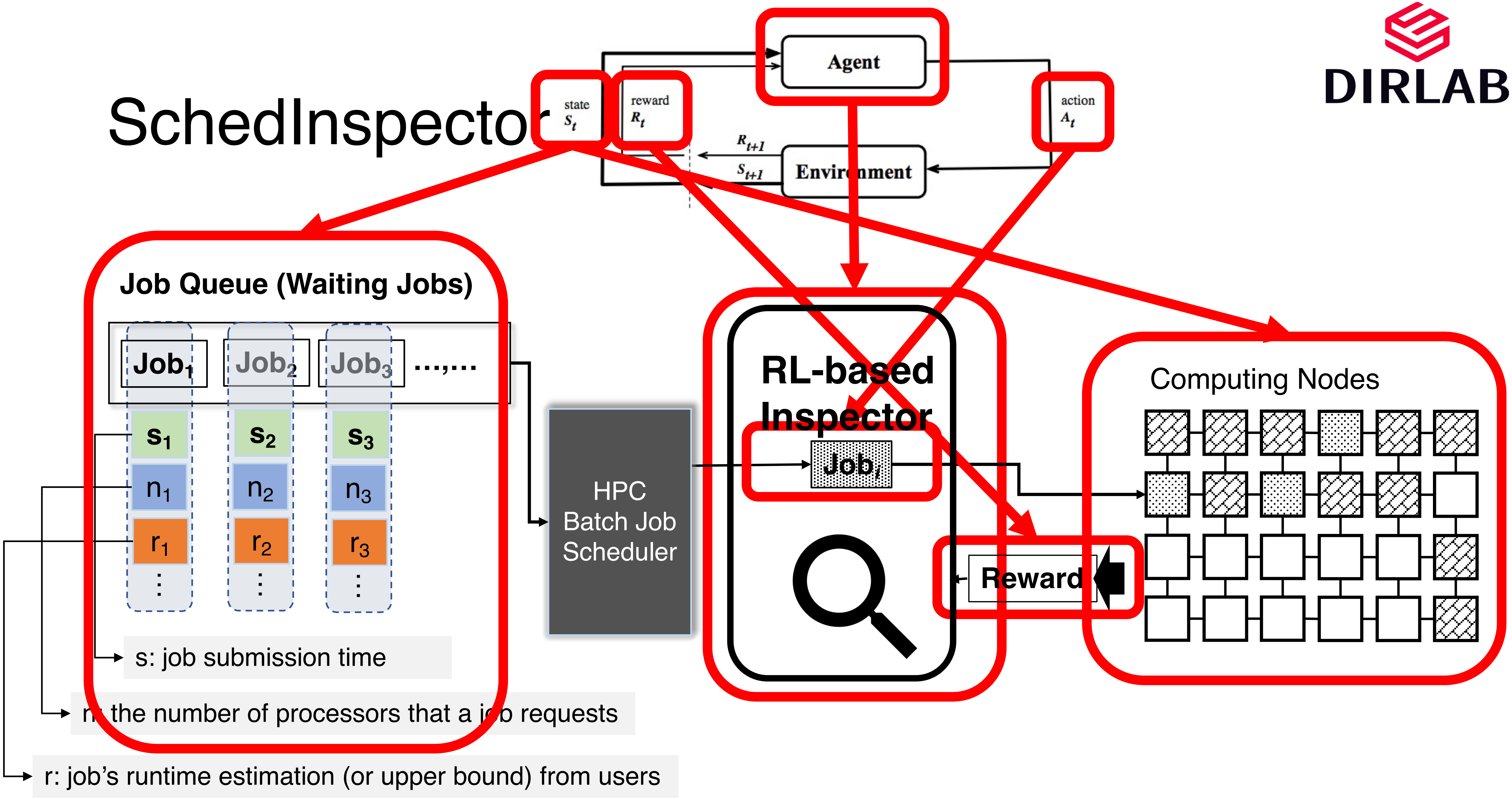
- Overview of SchedInspector
- Design of State and Reward

Our Contribution

- The first scheduling inspector for HPC systems.
- New optimizations of the state and reward to enable efficient RL training.
- Extensively evaluations on efficiency, stability and interpretability of SchedInspector.



SchedInspector



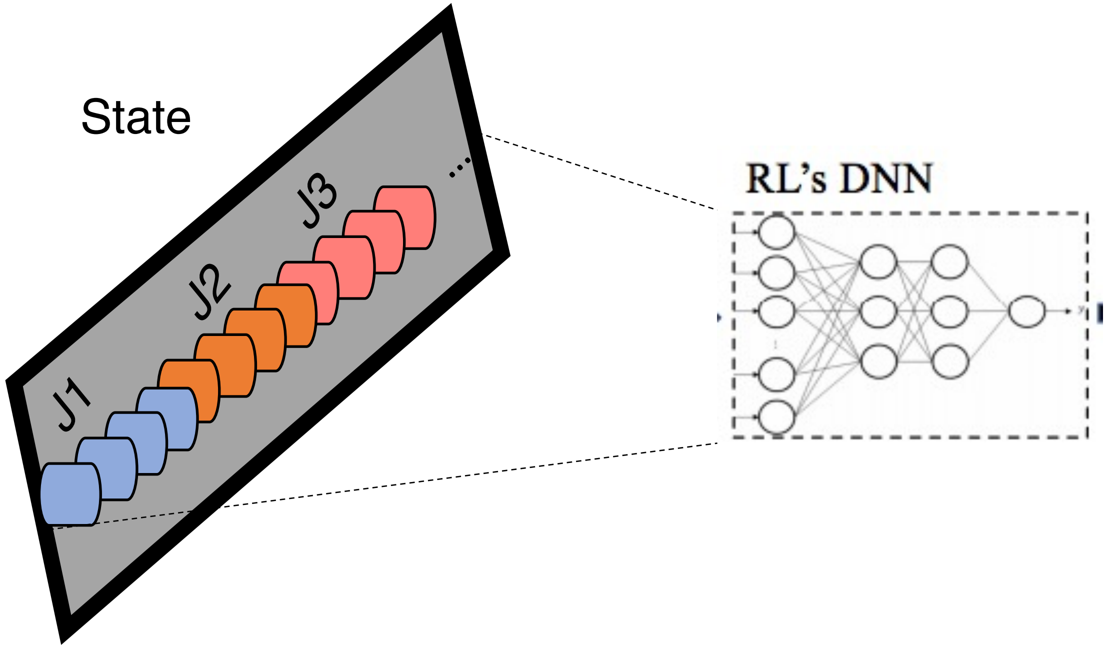
s: job submission time

n: the number of processors that a job requests

r: job's runtime estimation (or upper bound) from users

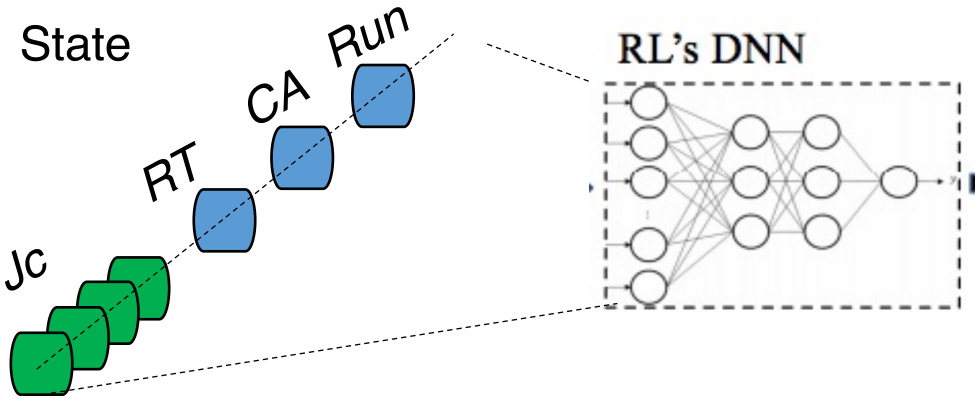
Design of State

Naïve Features



Compacted Features

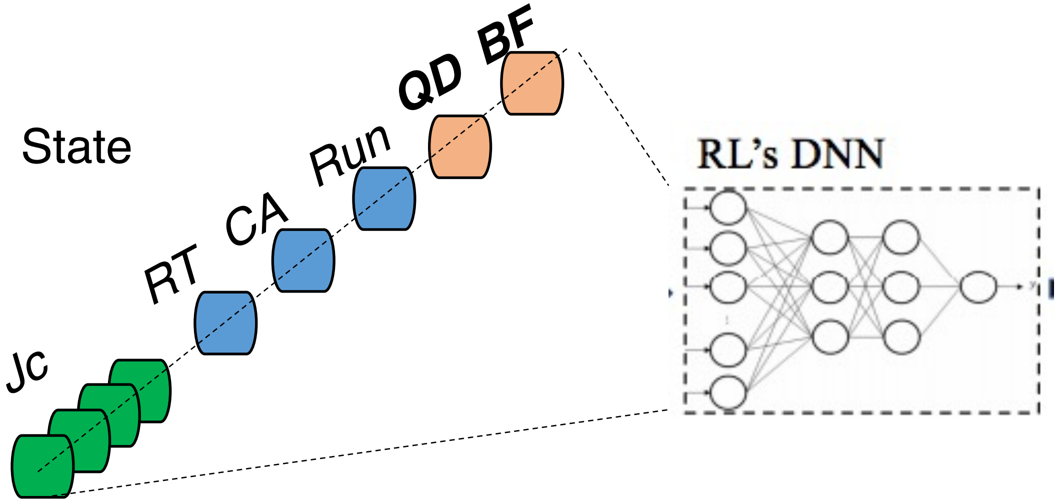
Jc: Scheduled Job
RT: Rejected Times
CA: Cluster Avail.
Run: Runnable

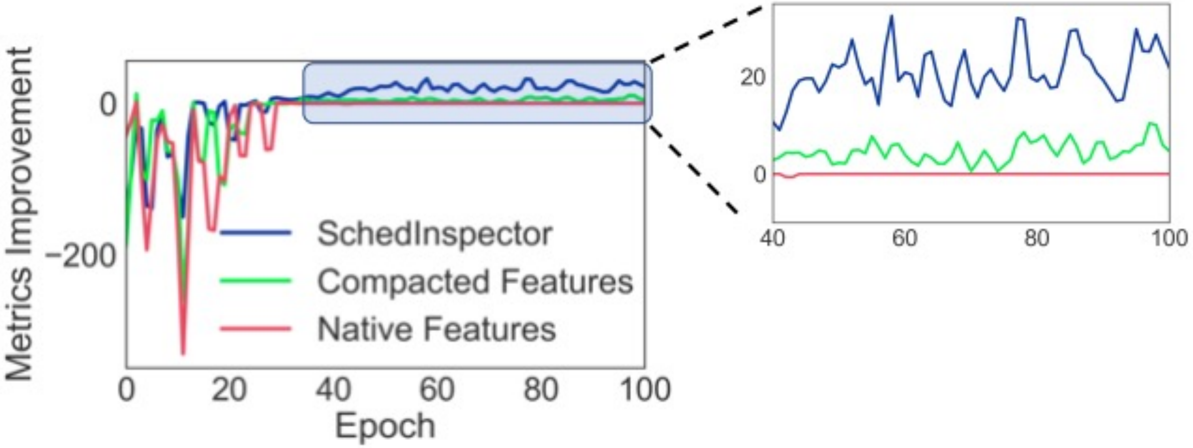


Design of State

SchedInspector

QD: Queue Delay
BF: Backfilling



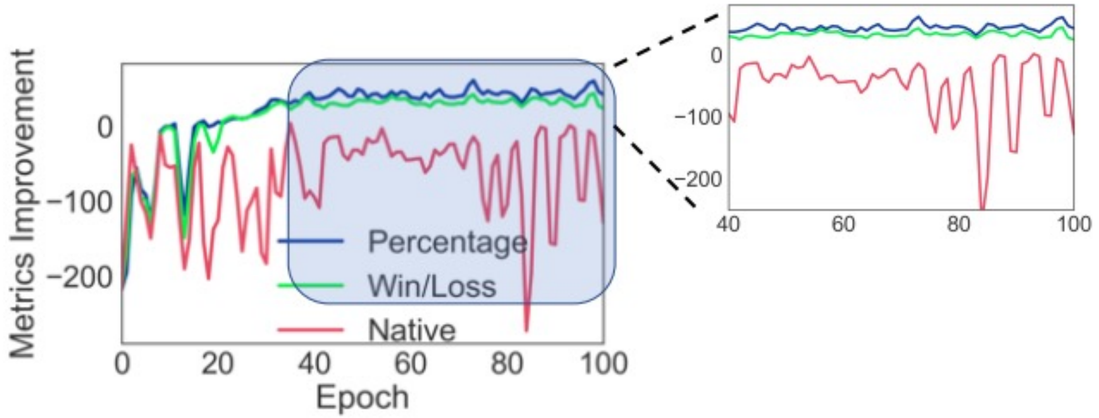


Design of Reward

Naïve: $Metric_{inspect} - Metric_{orig}$

Win/Loss: $Integer(Metric_{inspect} > Metric_{orig})$

✓ Percentage: $(Metric_{inspect} - Metric_{orig}) / Metric_{orig}$



DIRLAB

Motivation & Background

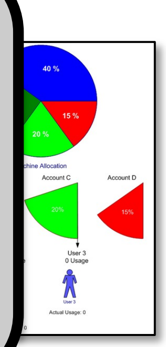
RLScheduler Design

Evaluation & Analysis

- Usability
- Efficiency
- Interpretability

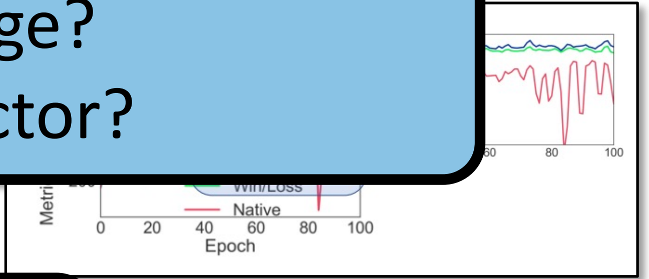
Conclusion

- How is the performance on **various job traces**?
- How is the performance for **different scheduling policies**?
- How is the performance of **different metrics**?

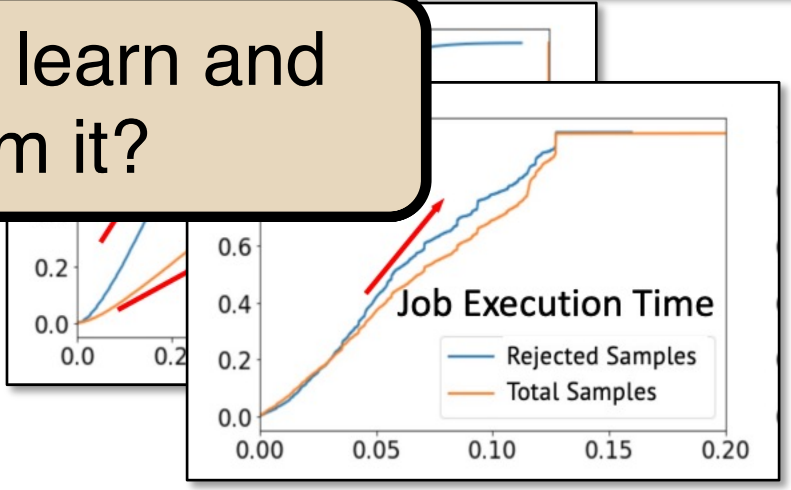


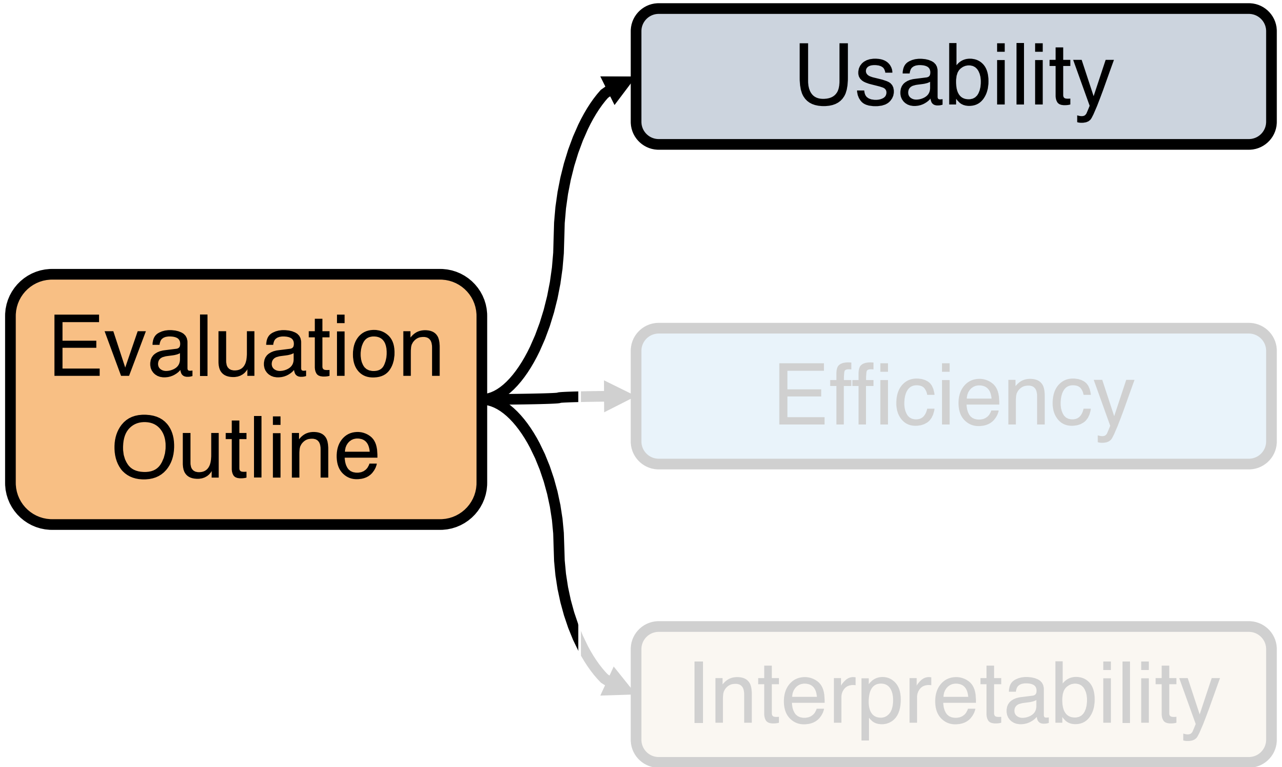
Ev

- How fast and stable can SchedInspector converge?
- What pattern it is in the training of SchedInspector?



What does Schedinspector learn and what we can learn from it?



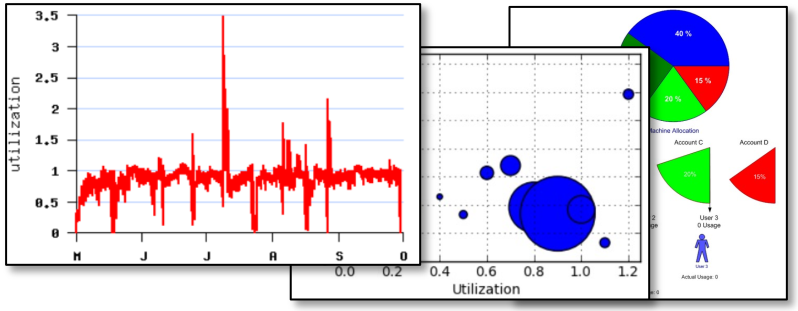


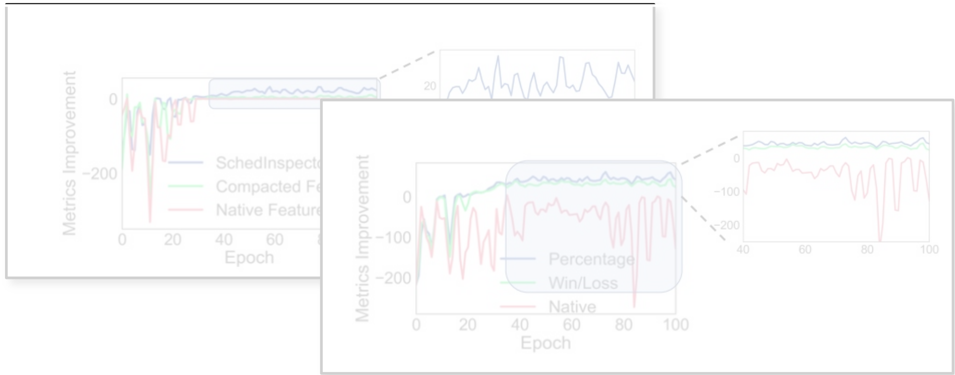
Evaluation Outline

Usability

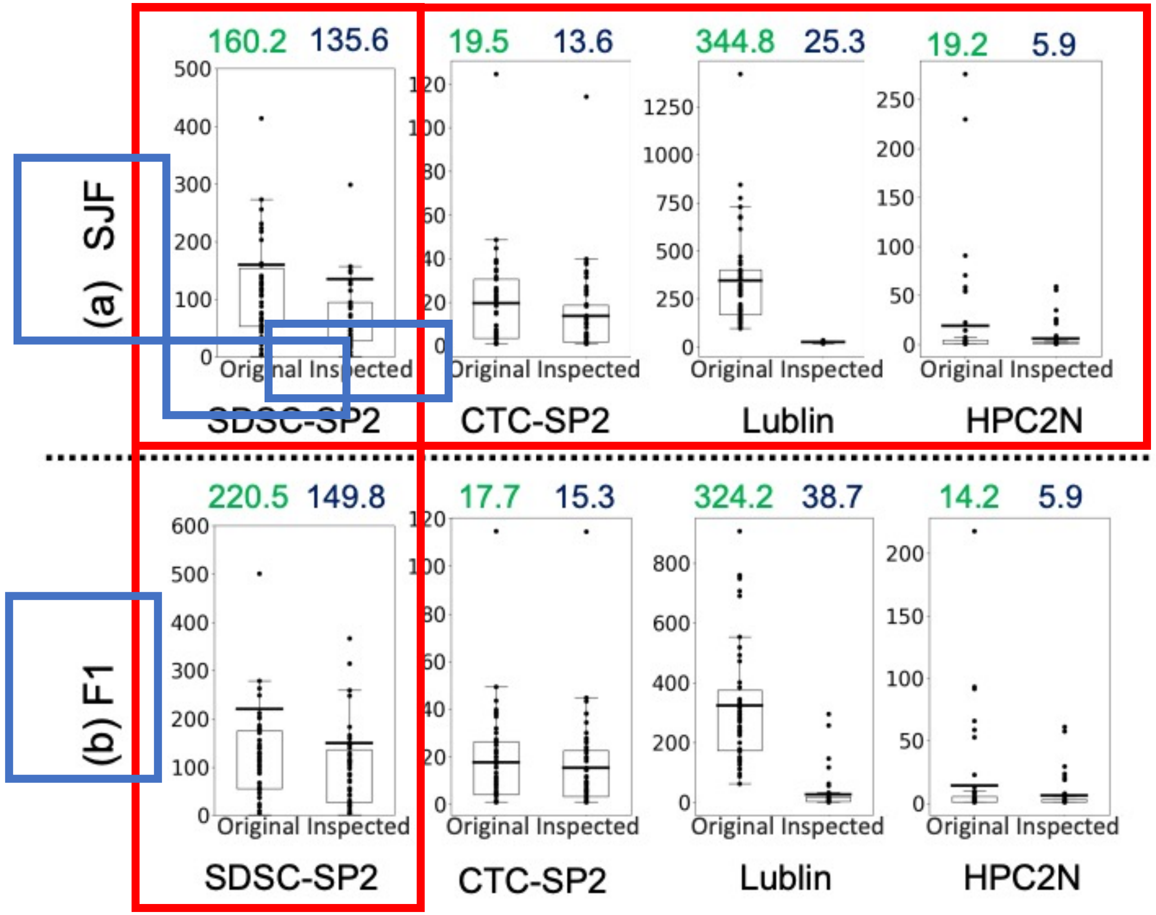
Efficiency

Interpretability



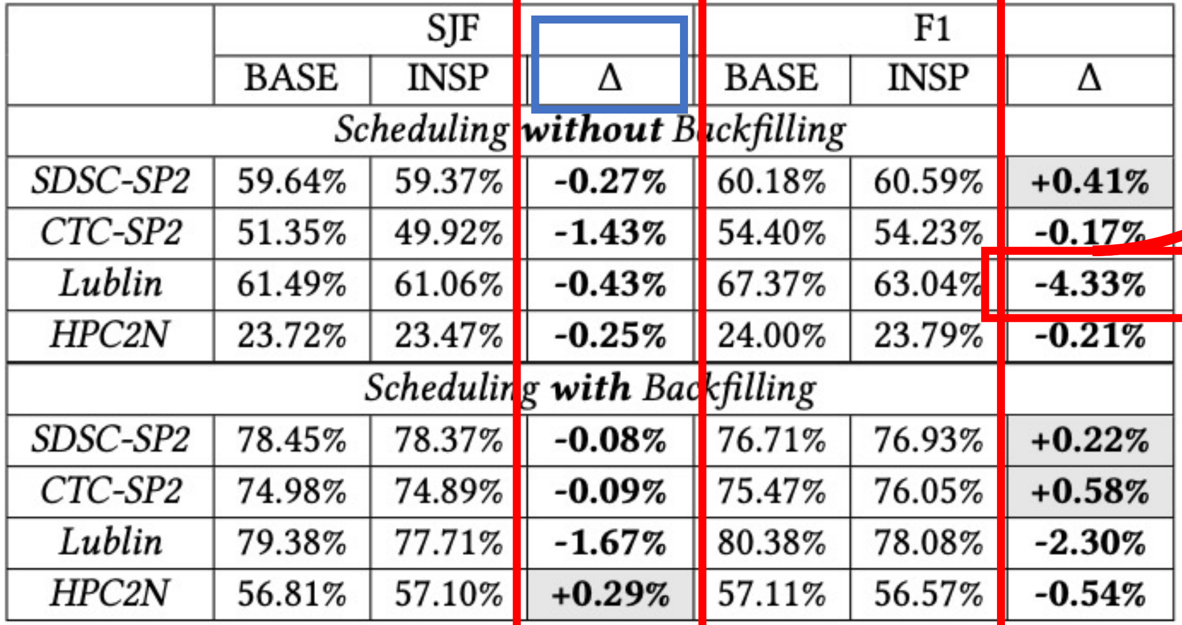


Testing for **Different Job Traces and Policies**

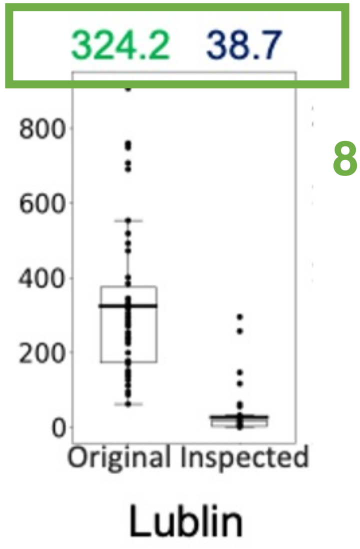


SchedInspector has significant improvement for the two scheduling policies on all job traces.

Impact on **System Utilization**

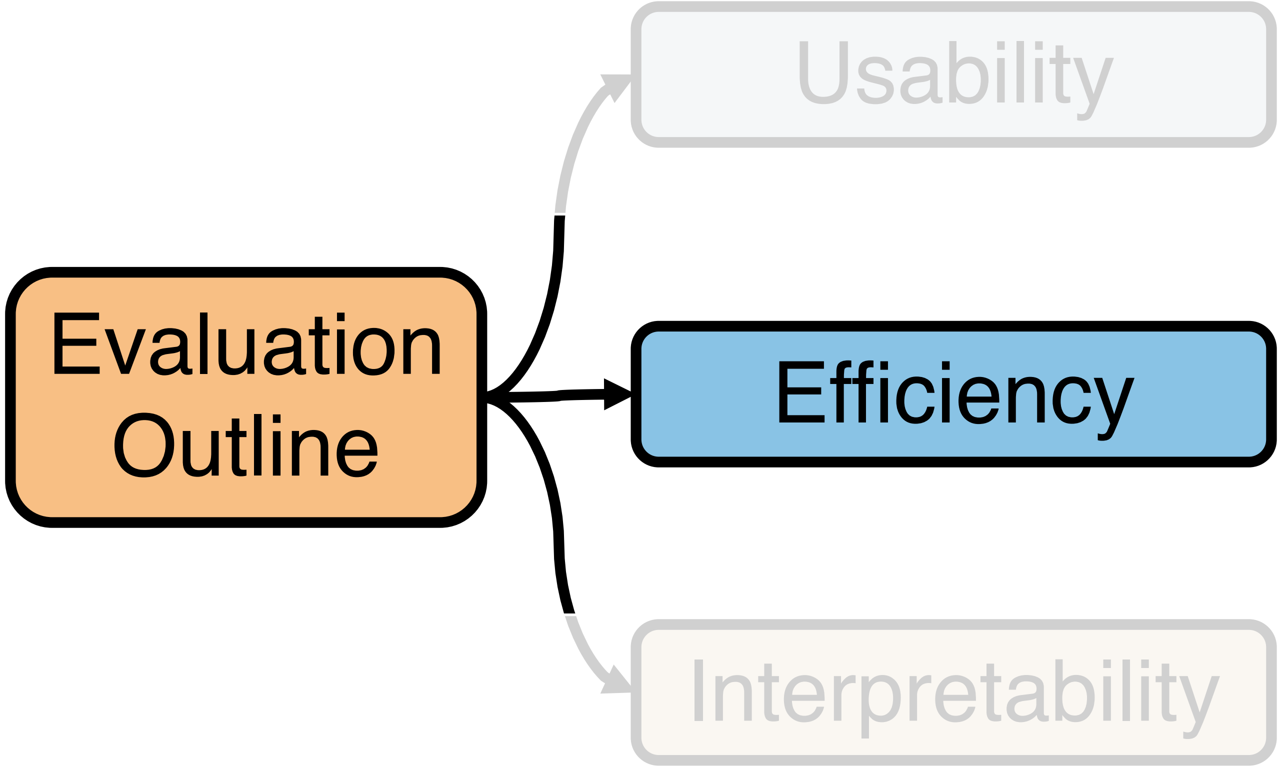


	SJF			F1		
	BASE	INSP	Δ	BASE	INSP	Δ
*Scheduling **without** Backfilling*						
SDSC-SP2	59.64%	59.37%	**-0.27%**	60.18%	60.59%	**+0.41%**
CTC-SP2	51.35%	49.92%	**-1.43%**	54.40%	54.23%	**-0.17%**
Lublin	61.49%	61.06%	**-0.43%**	67.37%	63.04%	**-4.33%**
HPC2N	23.72%	23.47%	**-0.25%**	24.00%	23.79%	**-0.21%**
*Scheduling **with** Backfilling*						
SDSC-SP2	78.45%	78.37%	**-0.08%**	76.71%	76.93%	**+0.22%**
CTC-SP2	74.98%	74.89%	**-0.09%**	75.47%	76.05%	**+0.58%**
Lublin	79.38%	77.71%	**-1.67%**	80.38%	78.08%	**-2.30%**
HPC2N	56.81%	57.10%	**+0.29%**	57.11%	56.57%	**-0.54%**



88% Improvement

SchedInspector has barely noticeable reduction (1% difference) on system utilization

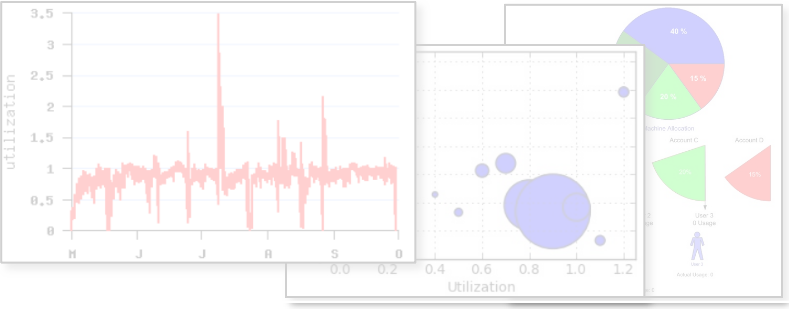


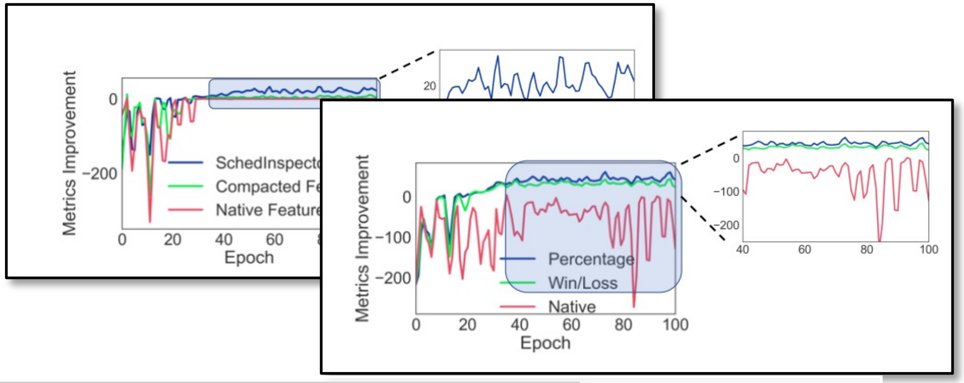
Evaluation Outline

Usability

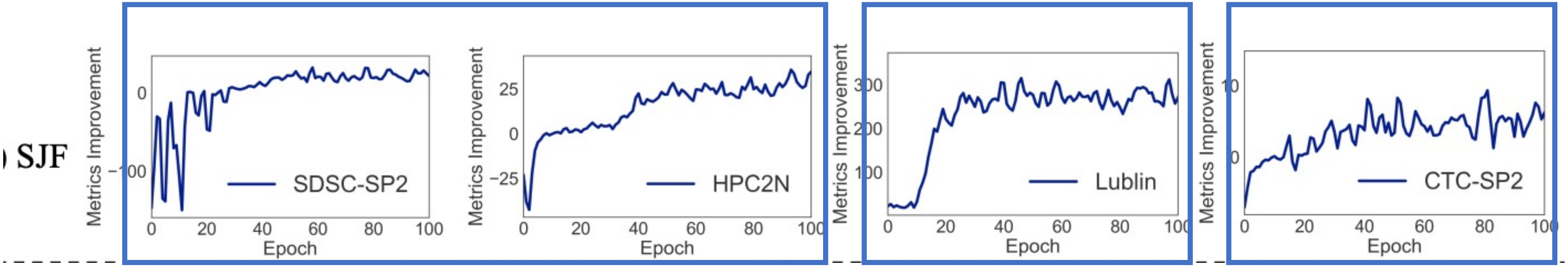
Efficiency

Interpretability



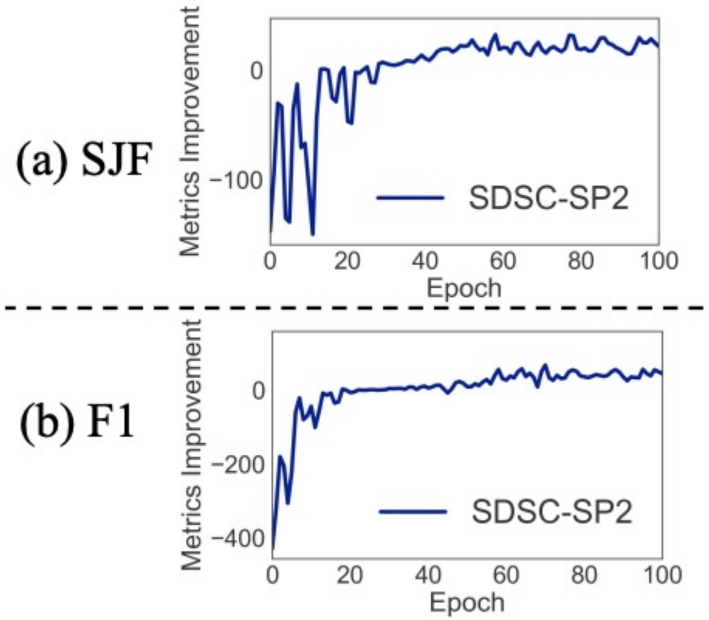


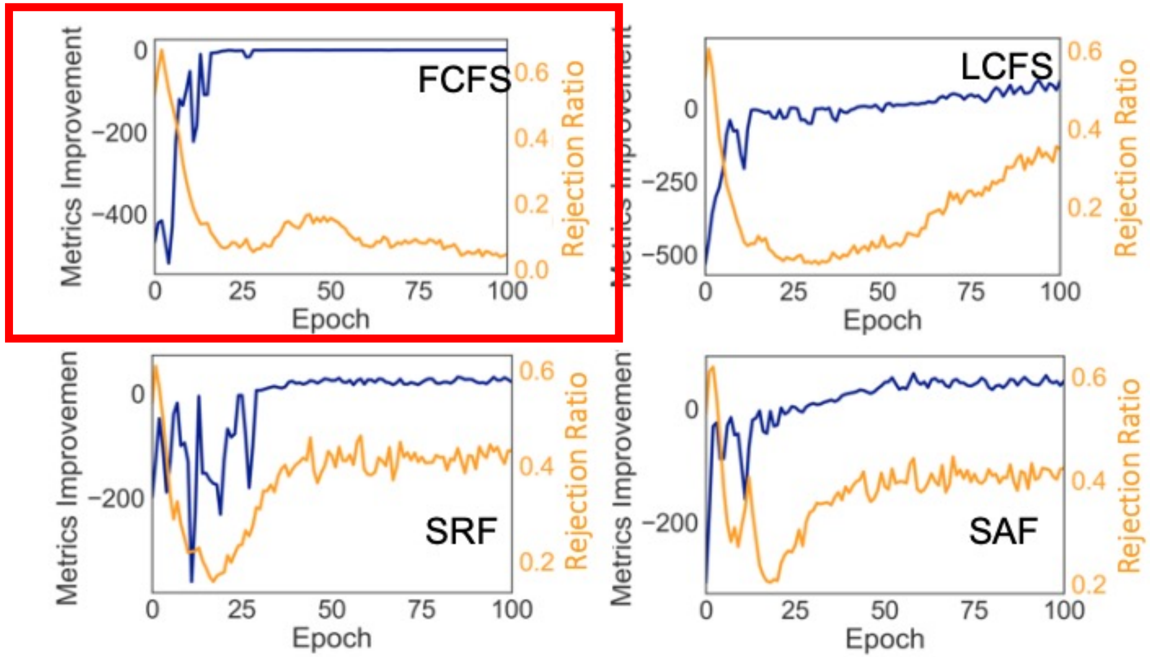
Training on **Different Job Traces**



SchedInspector converges in all of the workloads within 100 training epochs and different job traces have different converge pattern.

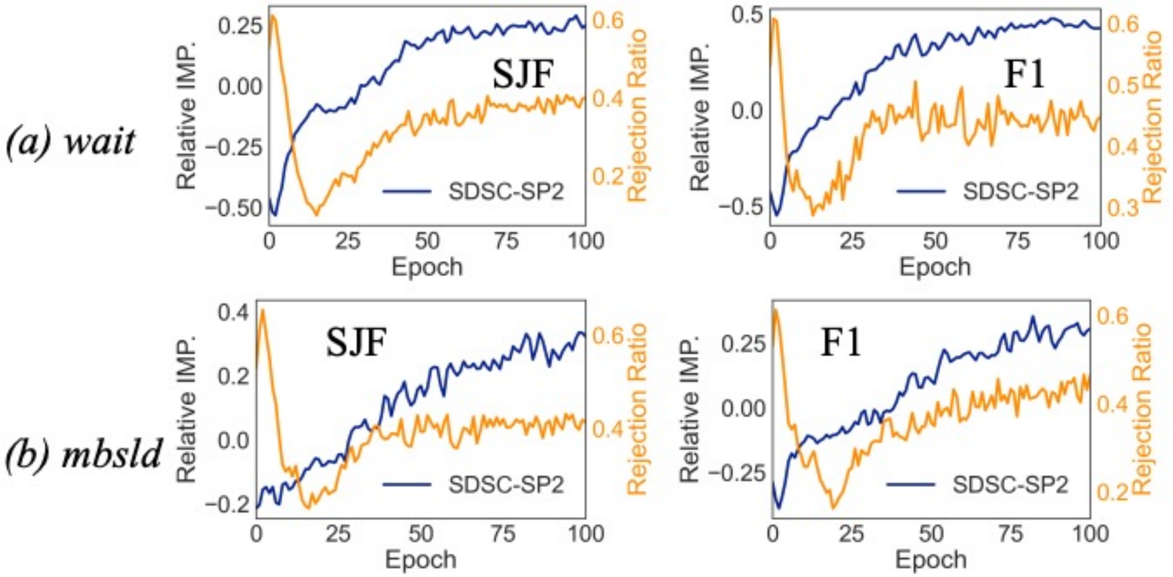
Training on **Different Scheduling Policies**



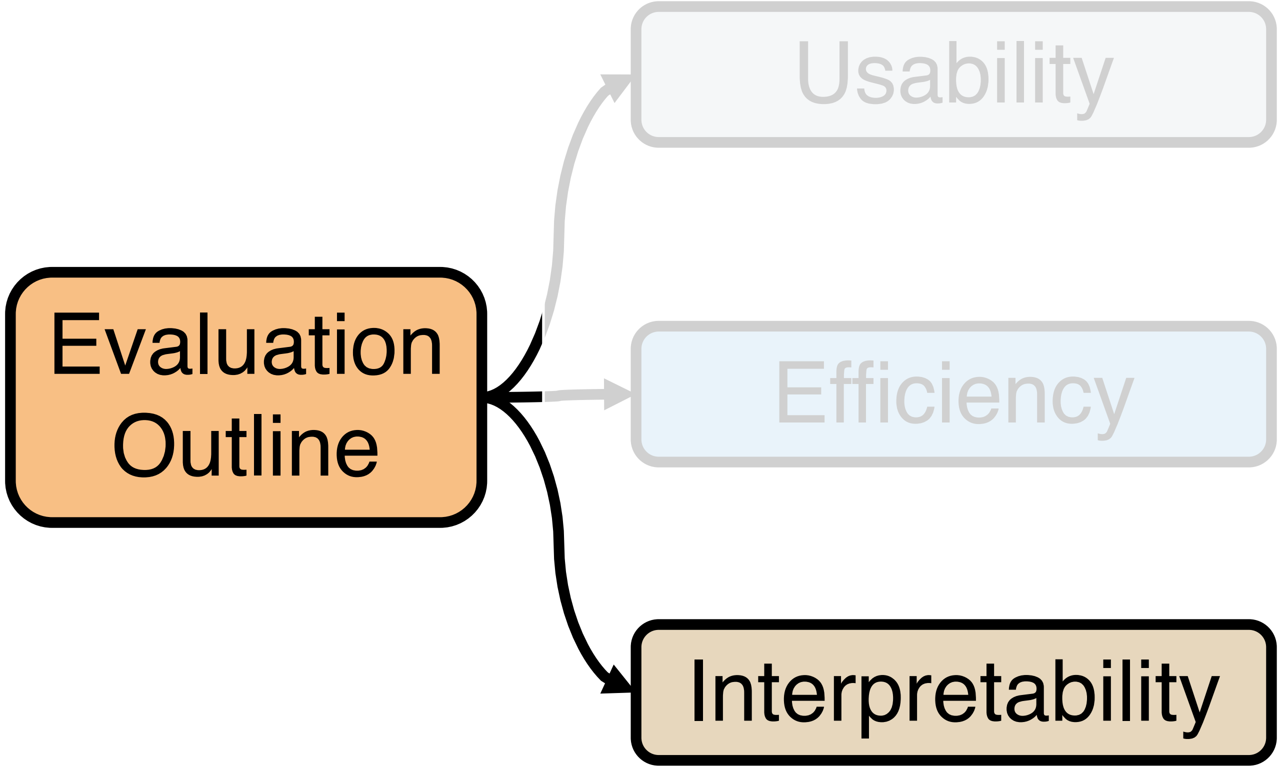


SchedInspector converges in all scheduling policies.
For some scheduling policies, the converged value is near 0 and the rejection ratio is low.

Training for **Different Metrics**



SchedInspector converges towards two new metrics but with different patterns.

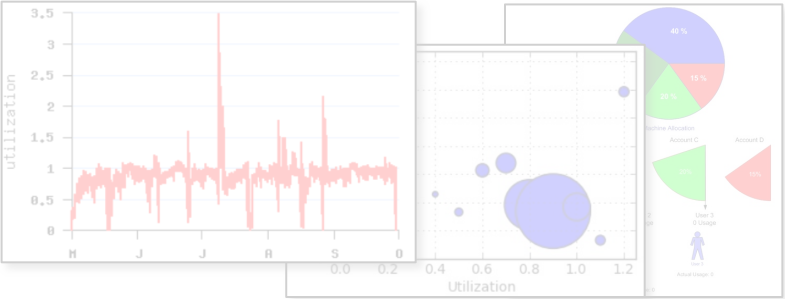


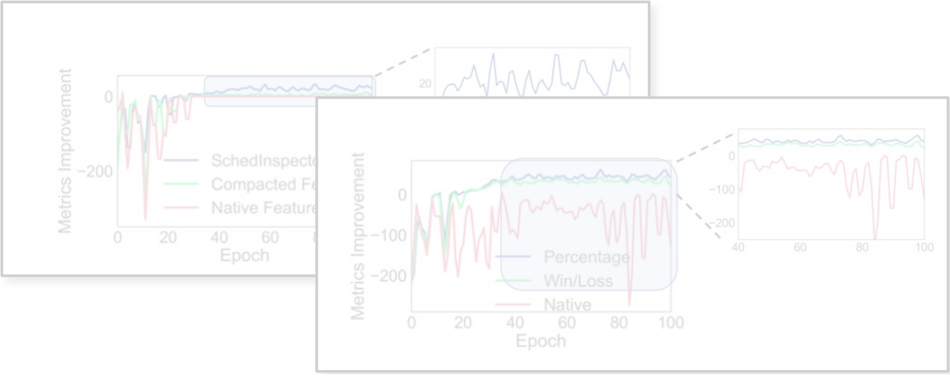
Evaluation Outline

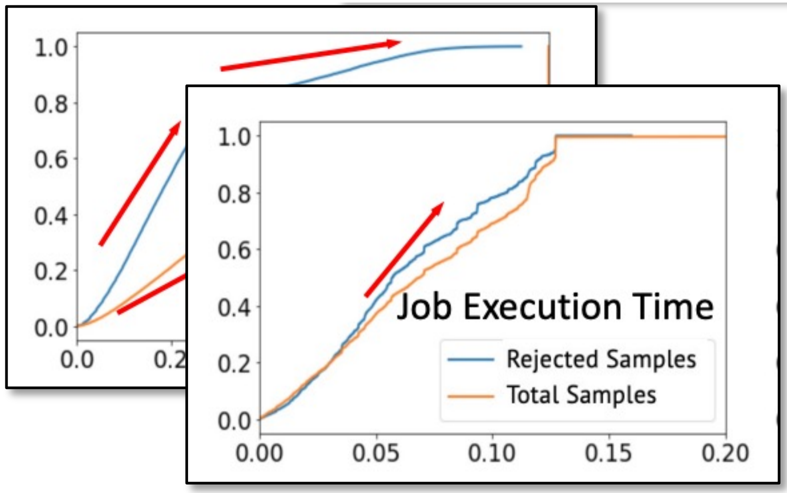
Usability

Efficiency

Interpretability

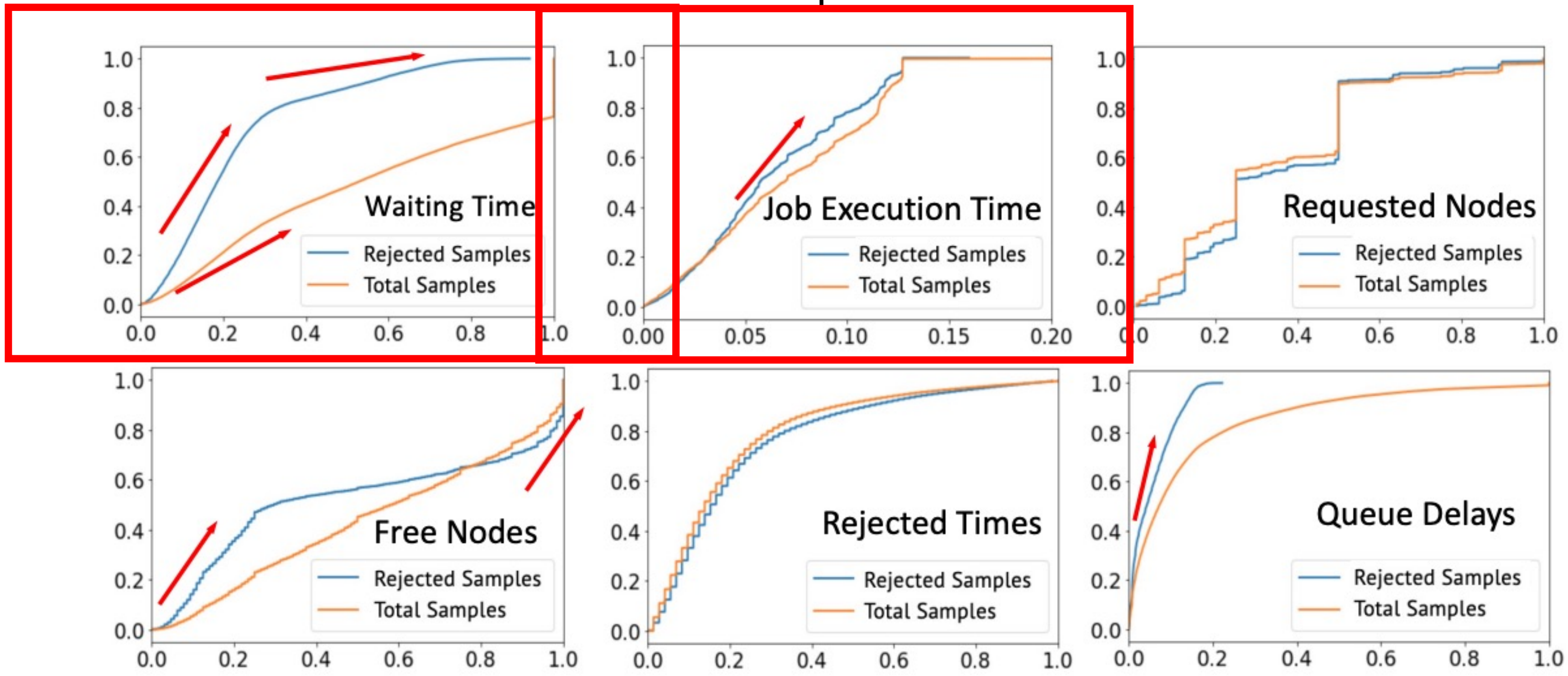






What SchedInspector Learns

CDF of input features.



SchedInspector has obvious patterns for different features which indicates the effectiveness of feature selection

DIRLAB

Motivation & Background

SchedInspector Design

Evaluation & Analysis

Conclusion

Summary

- We introduces scheduling inspector to integrate runtime factor into existing batch job scheduling.
  - https://github.com/DIR-LAB/SchedInspector
- We conducted extensive evaluations to show how SchedInspector performs on various job scheduling policies under various workloads.
- We carefully analyze and summarize the statistical rules learned by SchedInspector.

CHARLOTTE
COLLEGE OF COMPUTING AND INFORMATICS

OAK RIDGE
National Laboratory

Thank you! & Questions?