Running a Query Optimizer Advisor in Production

What we Learned (and What the Model didn't)

Matteo Interlandi

HPTS 2024

Crash Course on **Steering Query Optimizers using ML**

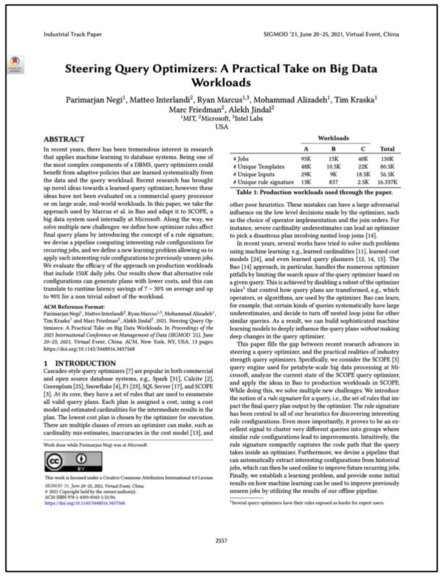
Goal

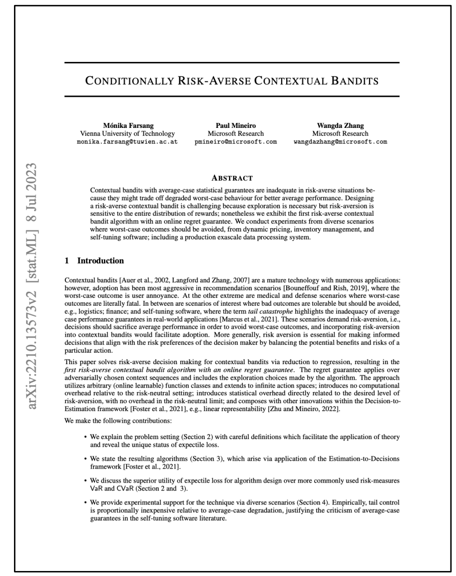
Share our experience on running a ML-based Query Optimizer Advisor at scale

What you will learn (Outline)

- How query hints are used to steer query optimizers toward better plans
- How to automate hint generation using ML
- How to scale it over hundreds of thousands of jobs
- Key lessons learned

For more Details





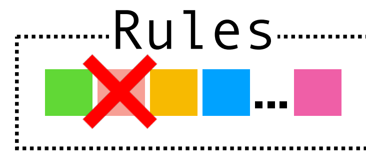
Steering Query Optimizers

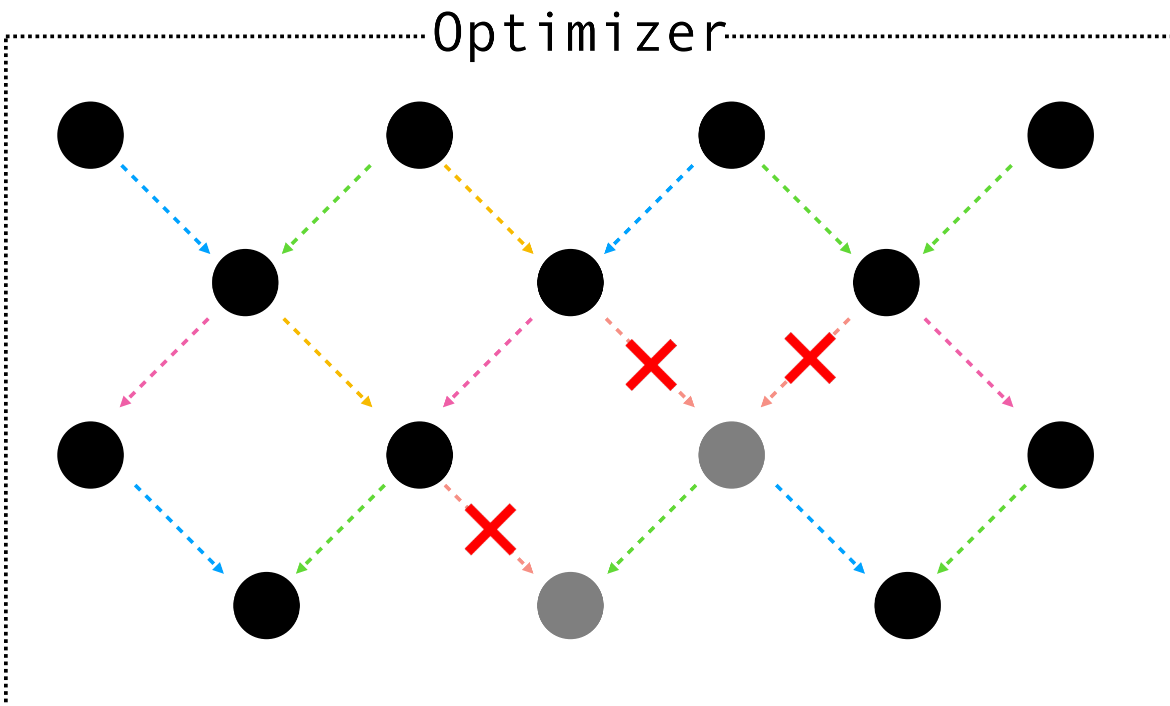
The Intuition

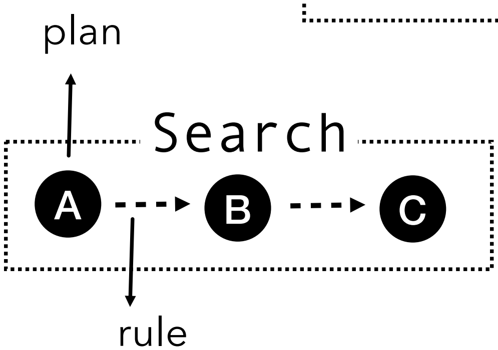
Intuition **Rule Based Optimizer**

Query (job)

```
in0 = SELECT * FROM sales WHERE ...;
in1 = SELECT * FROM product WHERE ...;
SELECT A, COUNT(*) FROM in0 JOIN in1
GROUP BY ...;
```







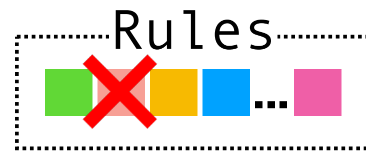
Optimizer 10,000 feet POV:

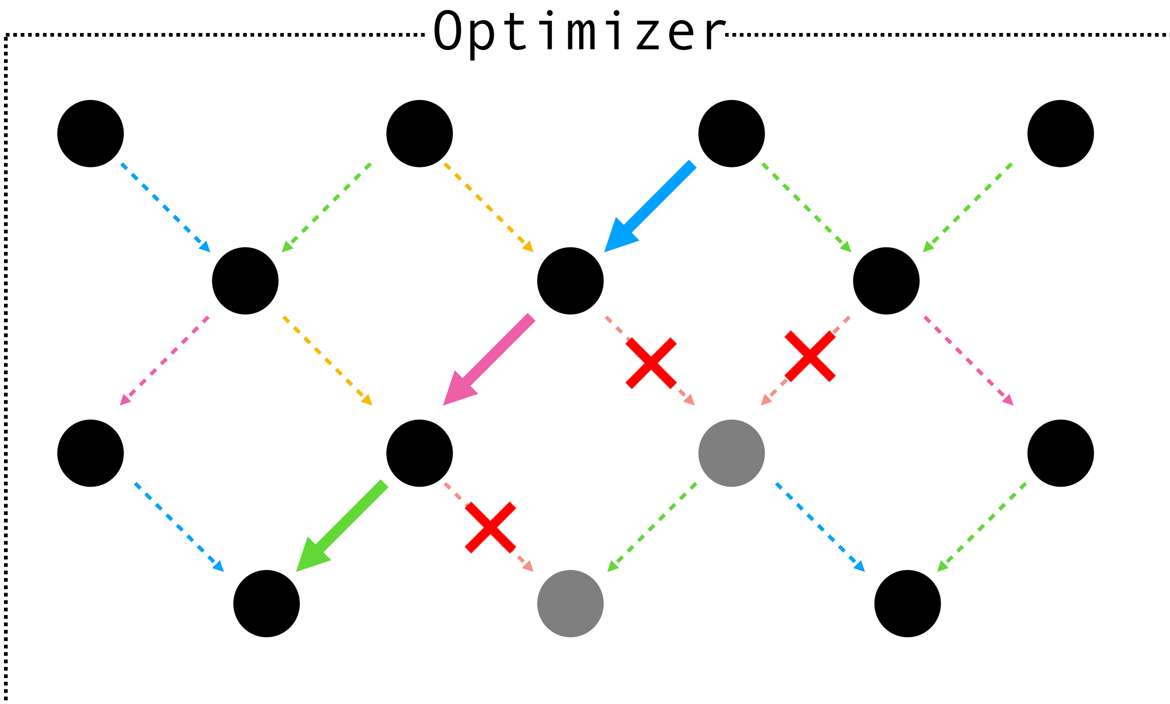
- *Rules used to transform query graph*
- *Rule configuration dictates the search space*

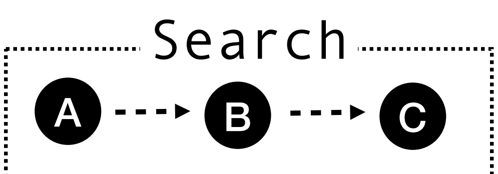
Intuition **Cheapest Plan**

Query (job)

```
in0 = SELECT * FROM sales WHERE ...;
in1 = SELECT * FROM product WHERE ...;
SELECT A, COUNT(*) FROM in0 JOIN in1
GROUP BY ...;
```







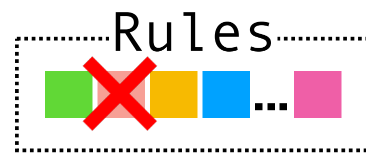
Optimizer 10,000 feet POV:

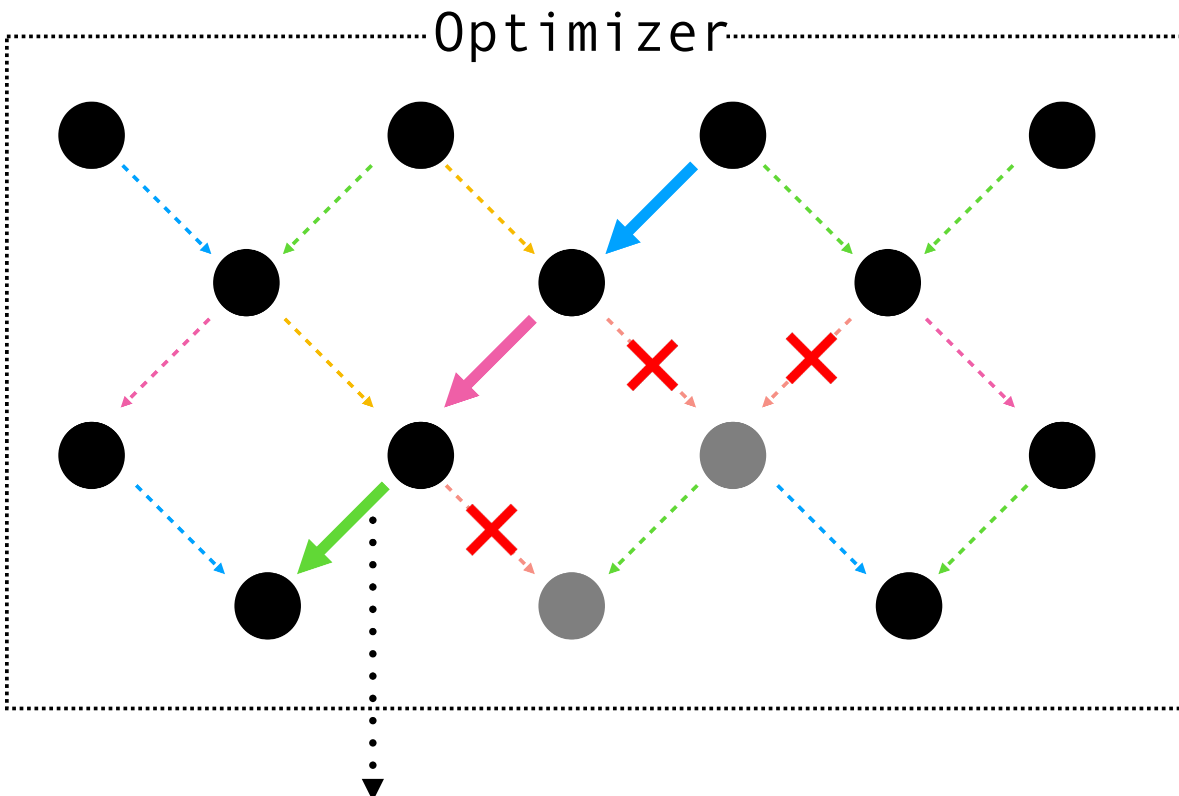
- *Rules used to transform query graph*
- *Rule configuration dictates the search space*

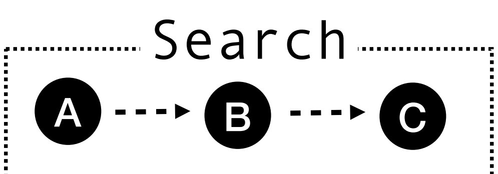
Intuition **What if cheapest plan is not optimal?**

Query (job)

```
in0 = SELECT * FROM sales WHERE ...;
in1 = SELECT * FROM product WHERE ...;
SELECT A, COUNT(*) FROM in0 JOIN in1
GROUP BY ...;
```

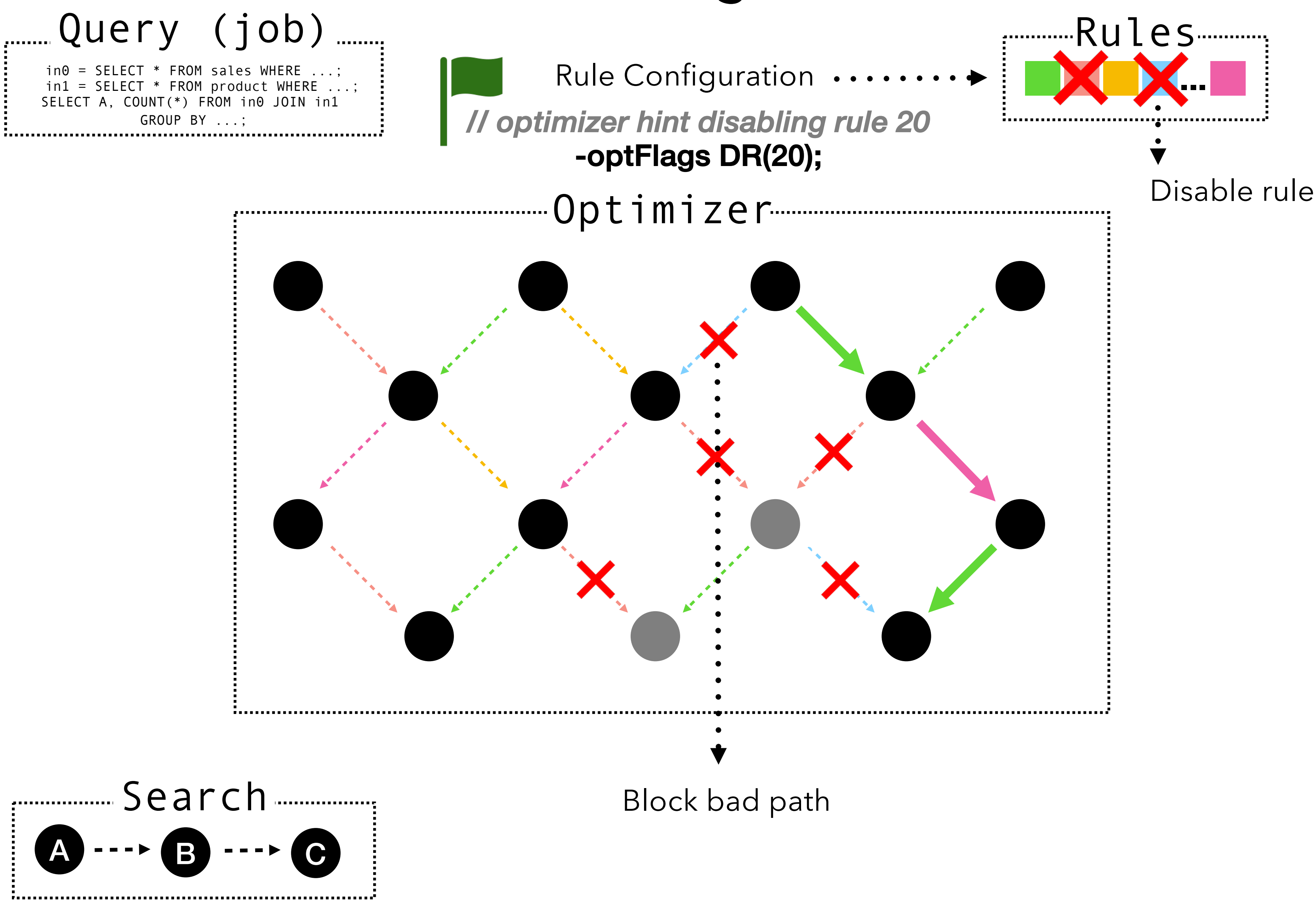




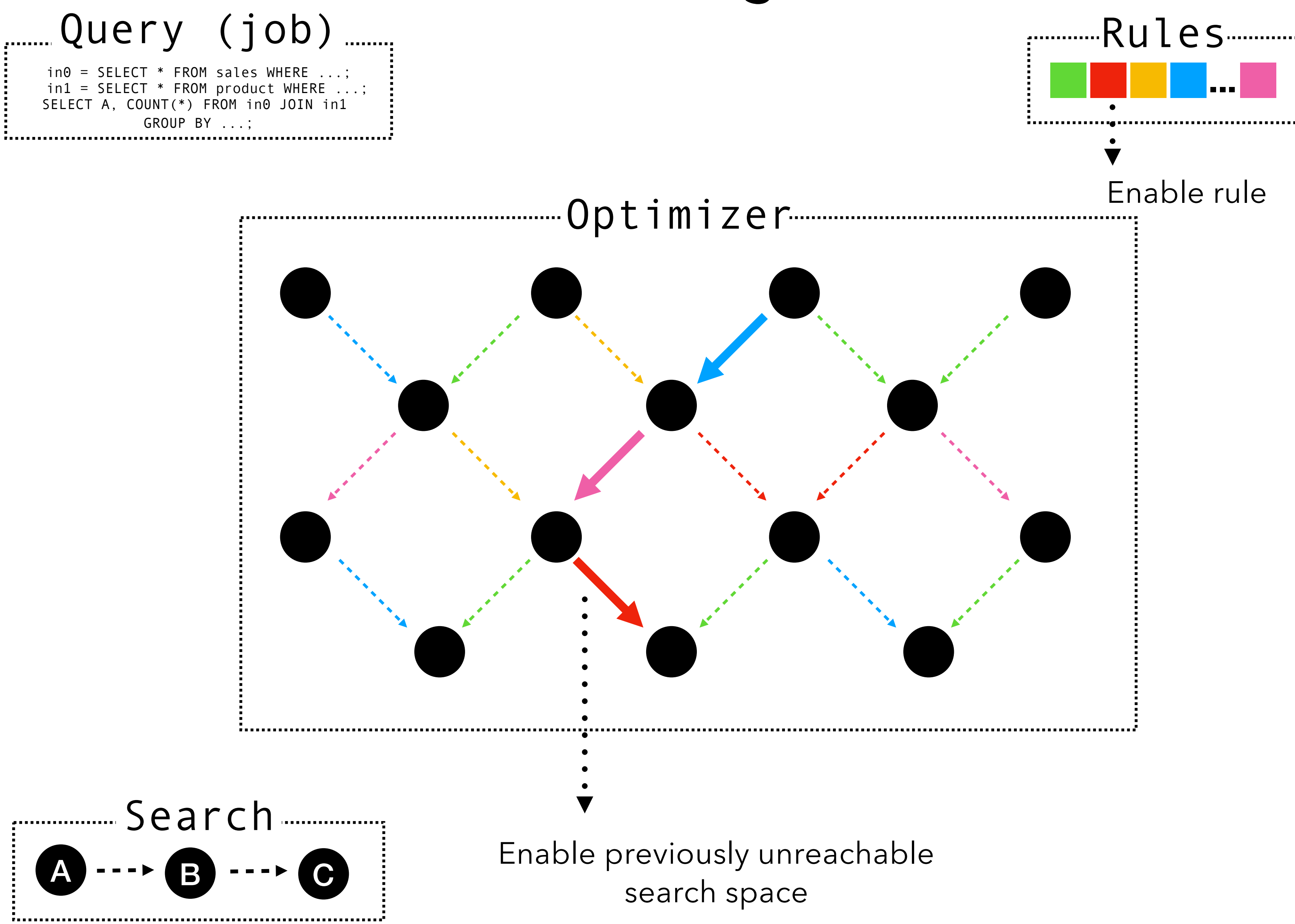


Not optimal due to mistakes in the *heuristics*, *cardinalities*, *costs*, other assumptions

Intuition **Disabling Rules**



Intuition **Enabling Rules**



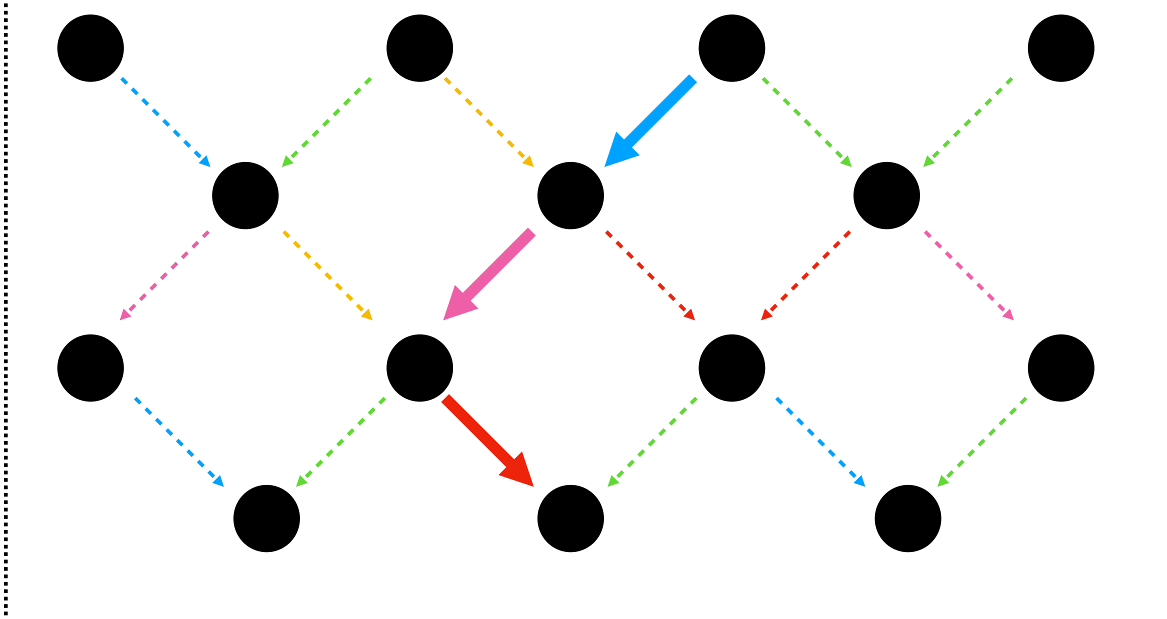
Intuition **Steering Query Optimizer**

Query (job)

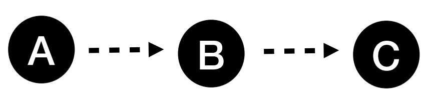
```
in0 = SELECT * FROM sales WHERE ...;
in1 = SELECT * FROM product WHERE ...;
SELECT A, COUNT(*) FROM in0 JOIN in1
GROUP BY ...;
```

Rules

Optimizer



Search



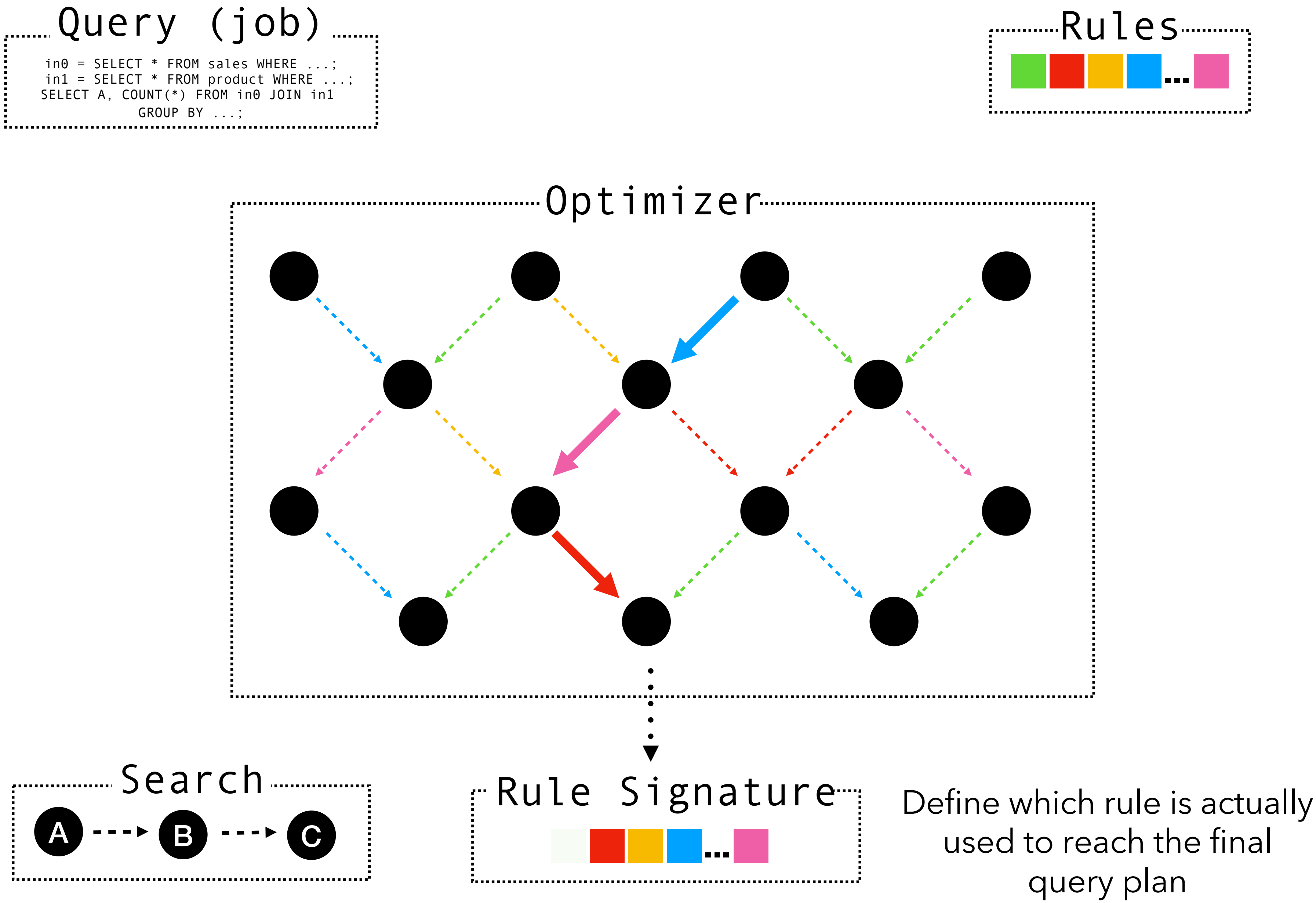
Scope users already use rule hints to fine tune queries:

- Up to ~9% of jobs contain user-provided rule hints
- Difficult to tune, requiring a lot of expertise and experimentation
- *Can we automate this at scale?*

Steering Query Optimizers

Core Techniques

Core Techniques **Rule Signature**

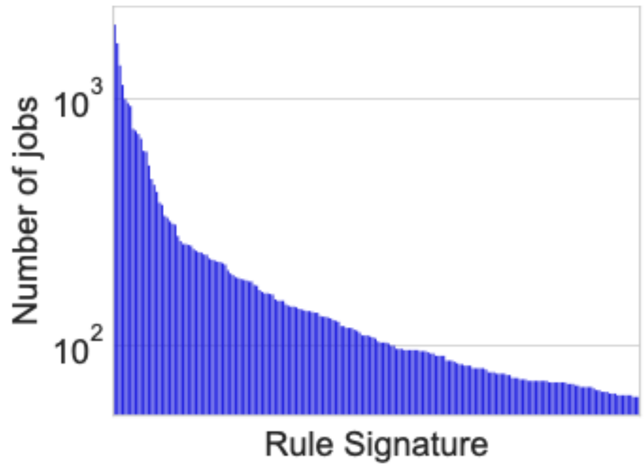


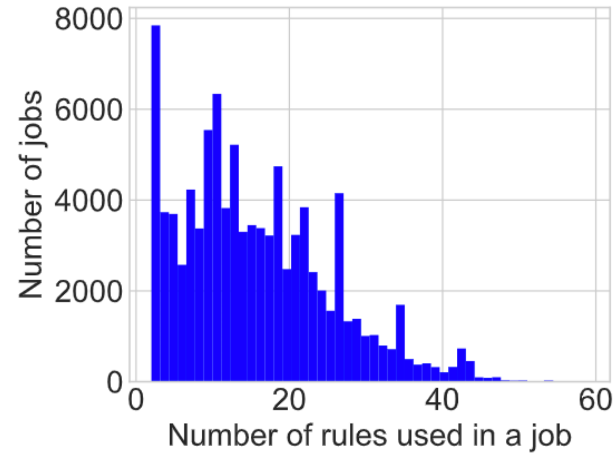
Core Techniques **Rule Signature**

Query (job)

```
in0 = SELECT * FROM sales WHERE ...;
in1 = SELECT * FROM product WHERE ...;
SELECT A, COUNT(*) FROM in0 JOIN in1
GROUP BY ...;
```

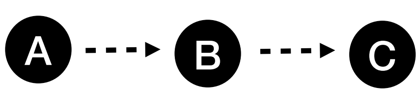
Rules





2^256 potential rule signatures. But it has a lot of structure!

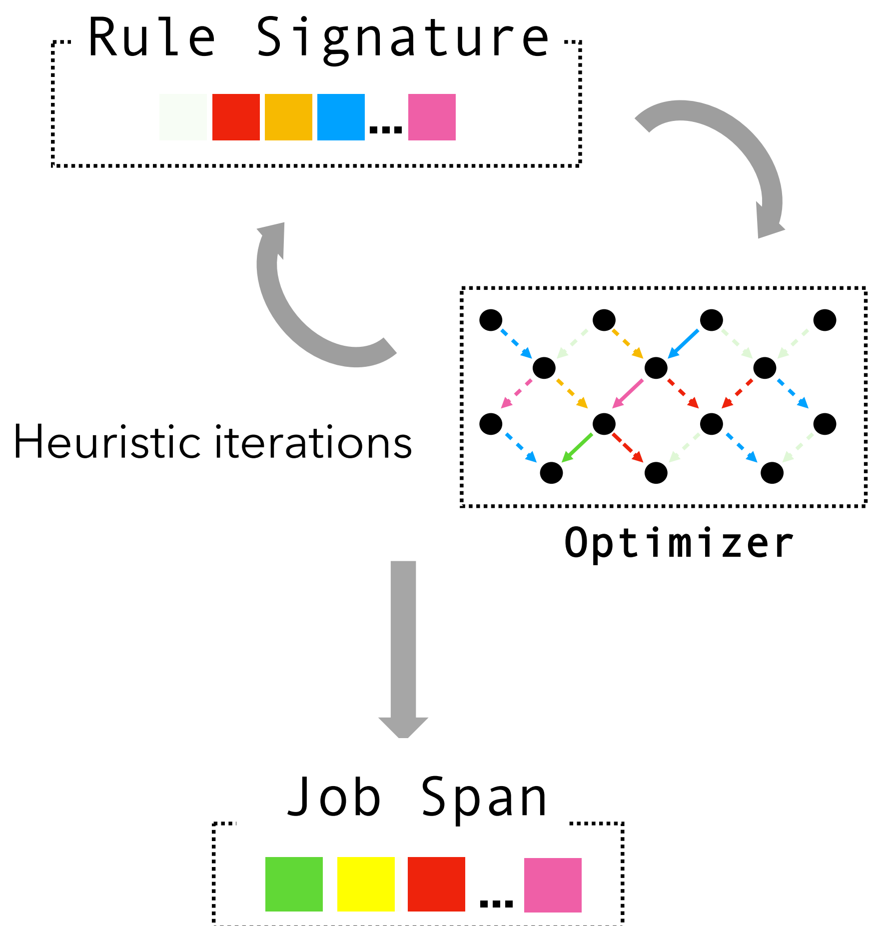
Search



Rule Signature

Define which rule is actually used to reach the final query plan

Core Techniques **Job Span**

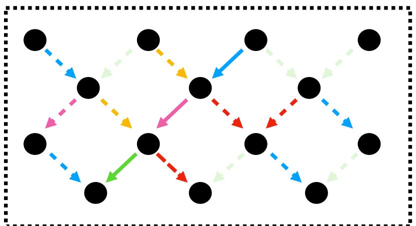


Set of rules that can change the final query plan
(i.e., the interesting set of rules)

Steering Query Optimizers

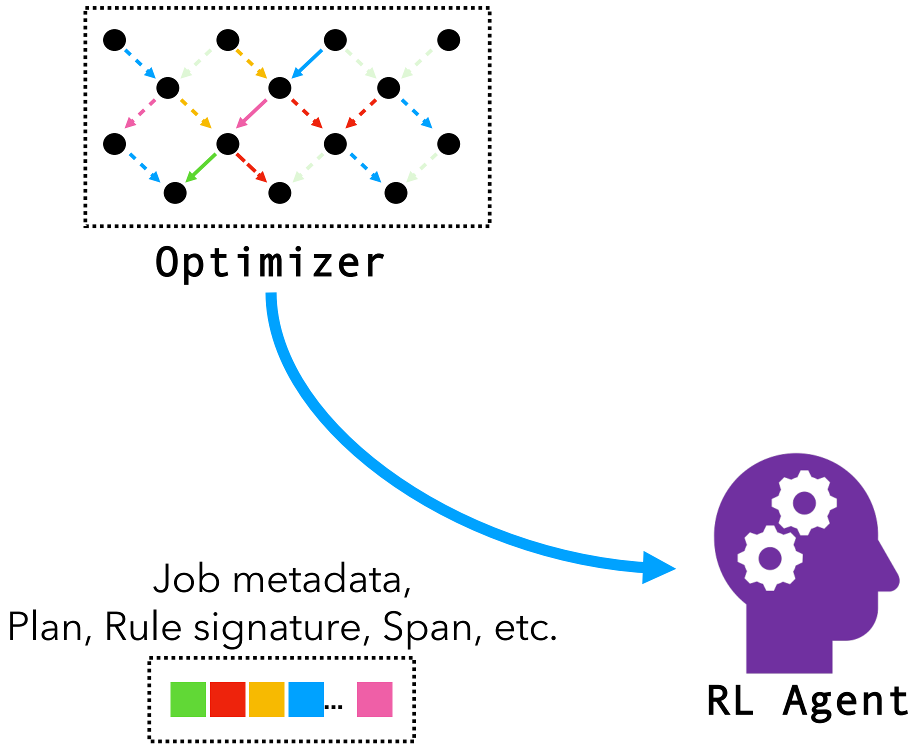
The Automation

Automation **As RL Problem**

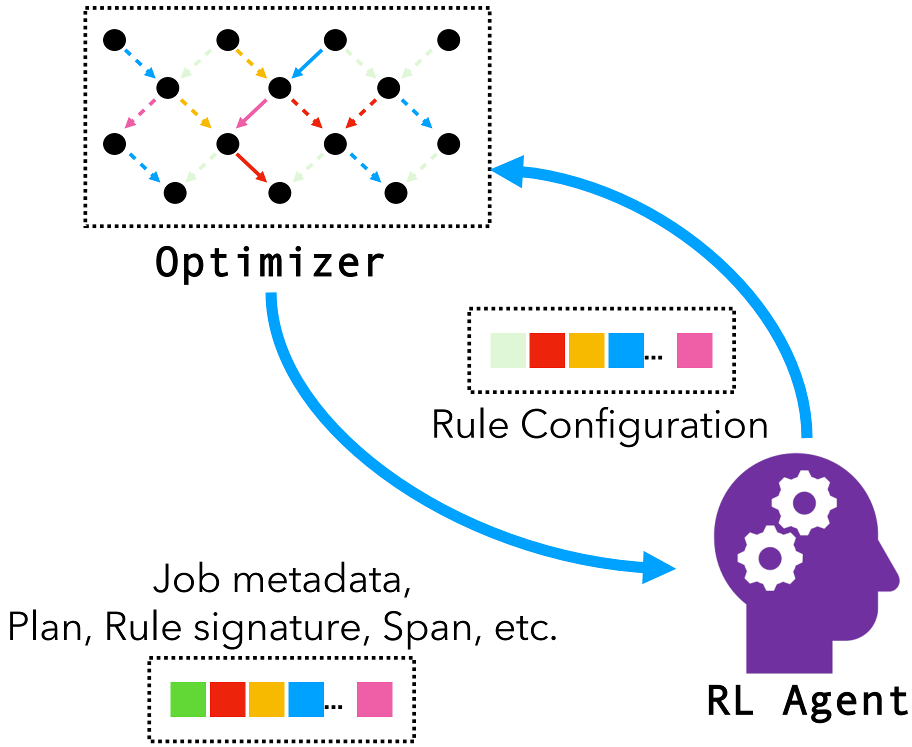


Optimizer

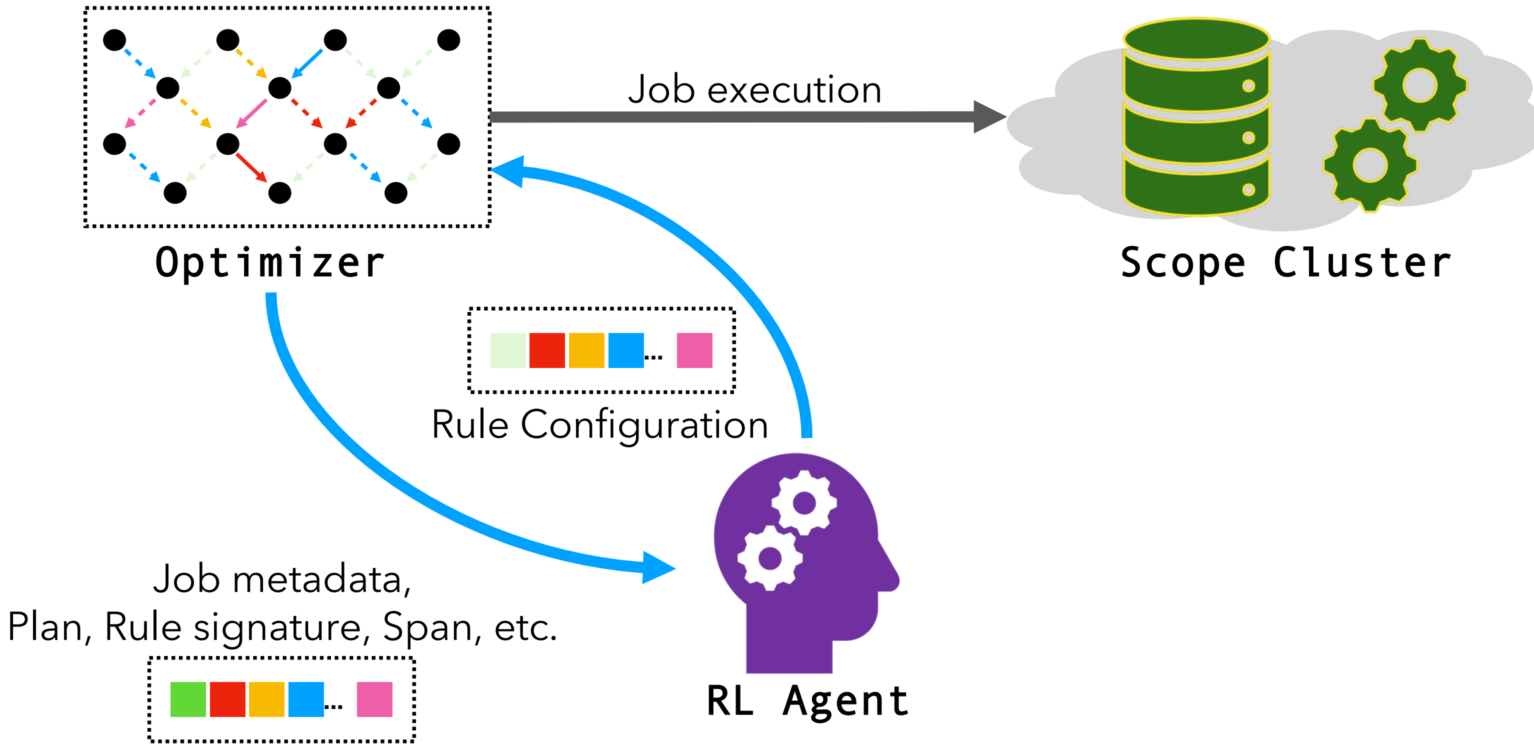
Automation **As RL Problem**



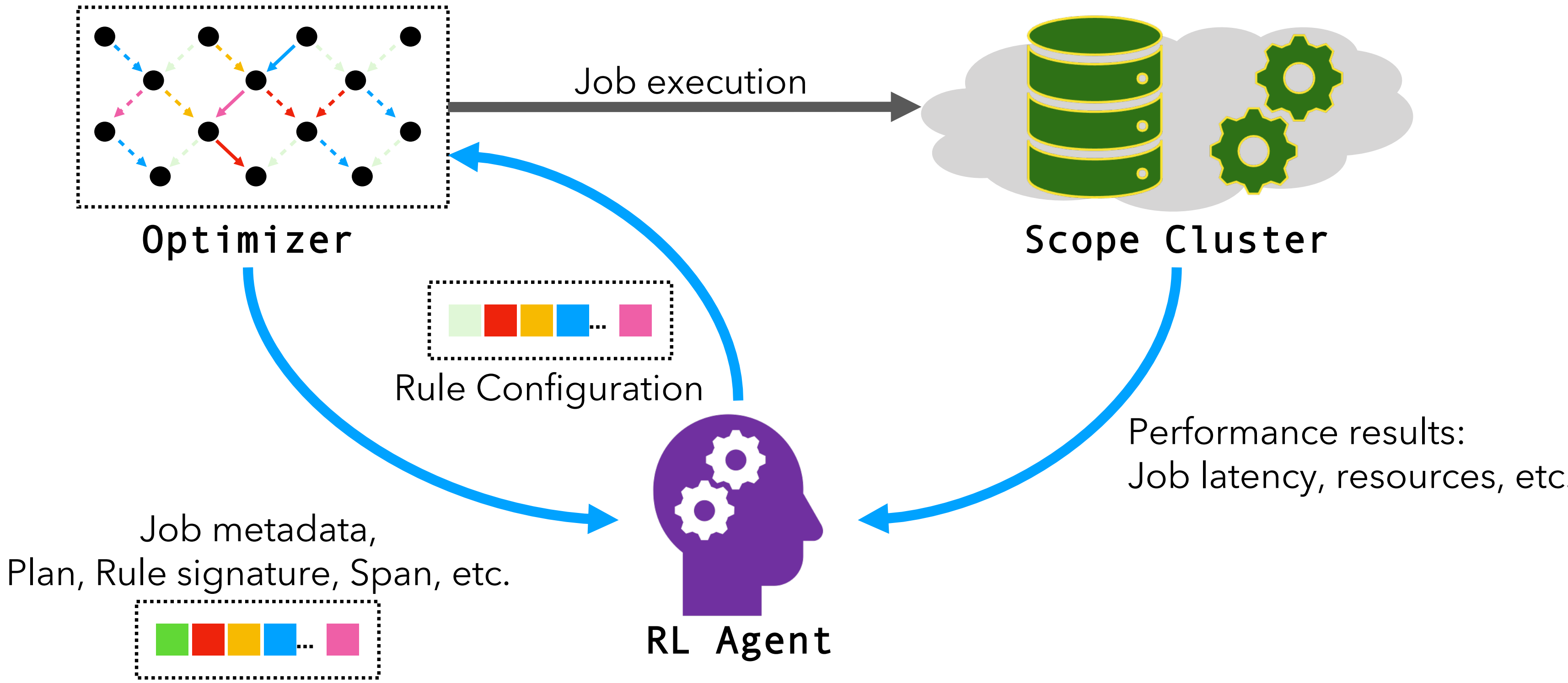
Automation **As RL Problem**



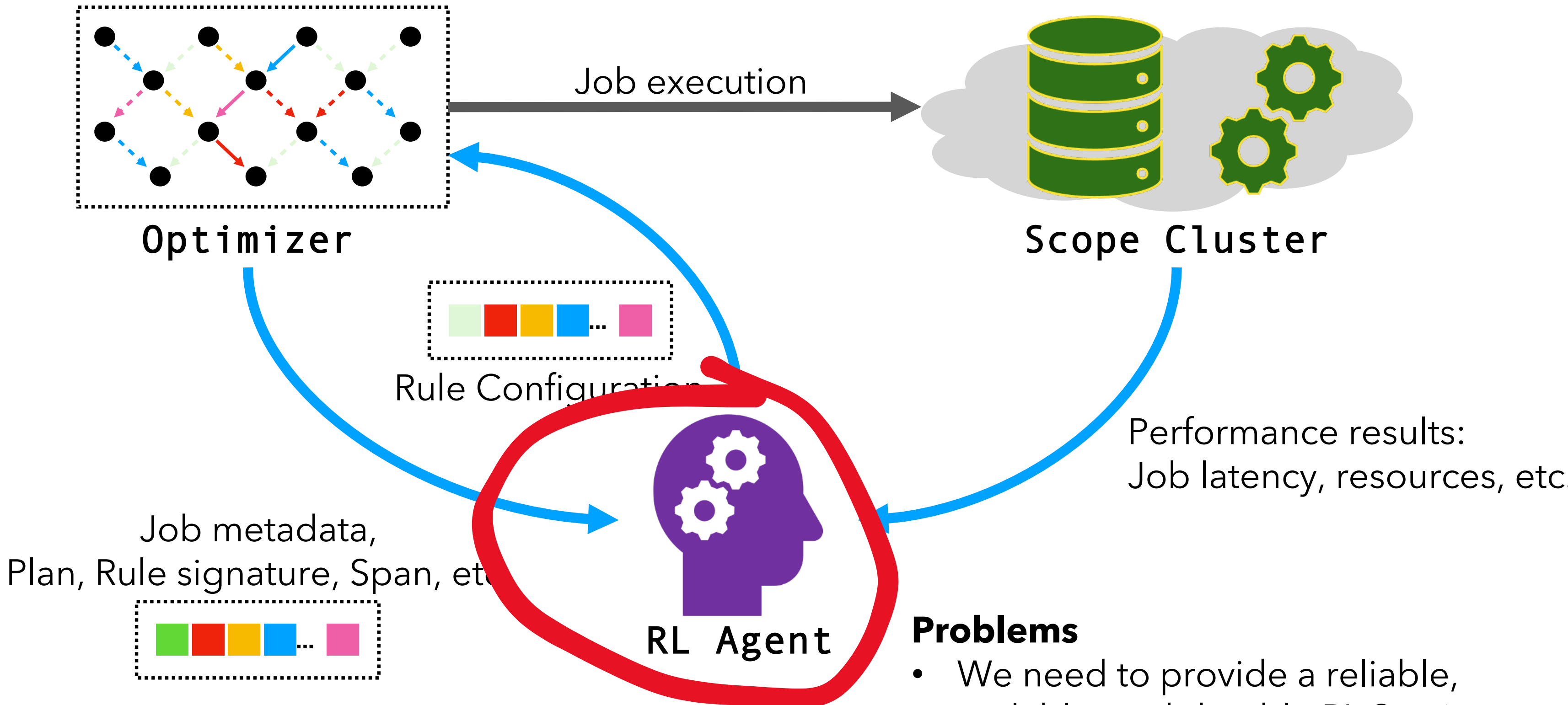
Automation **As RL Problem**



Automation **As RL Problem**



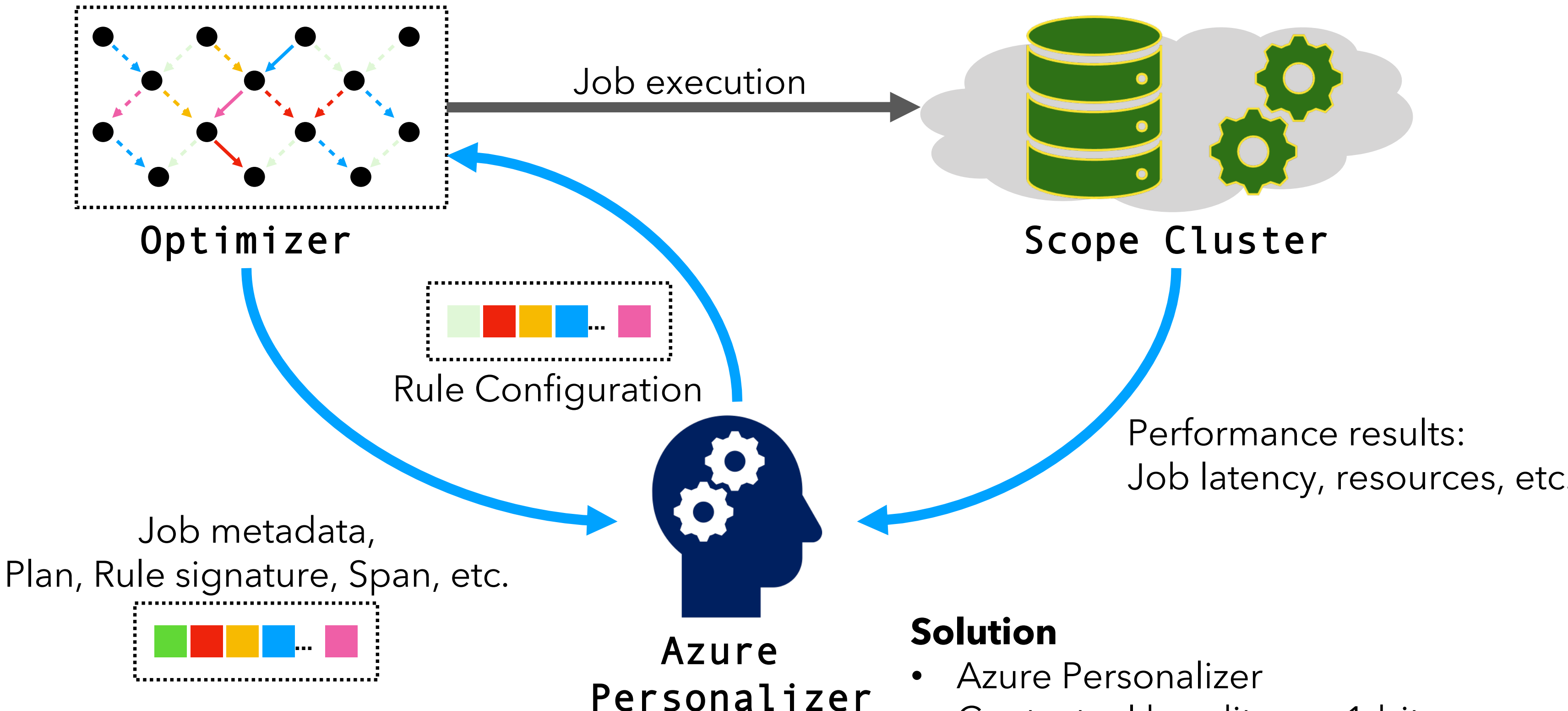
Automation **As RL Problem**



Problems

- We need to provide a reliable, scalable, and durable RL Service
- 2^20 is still a large space to explore

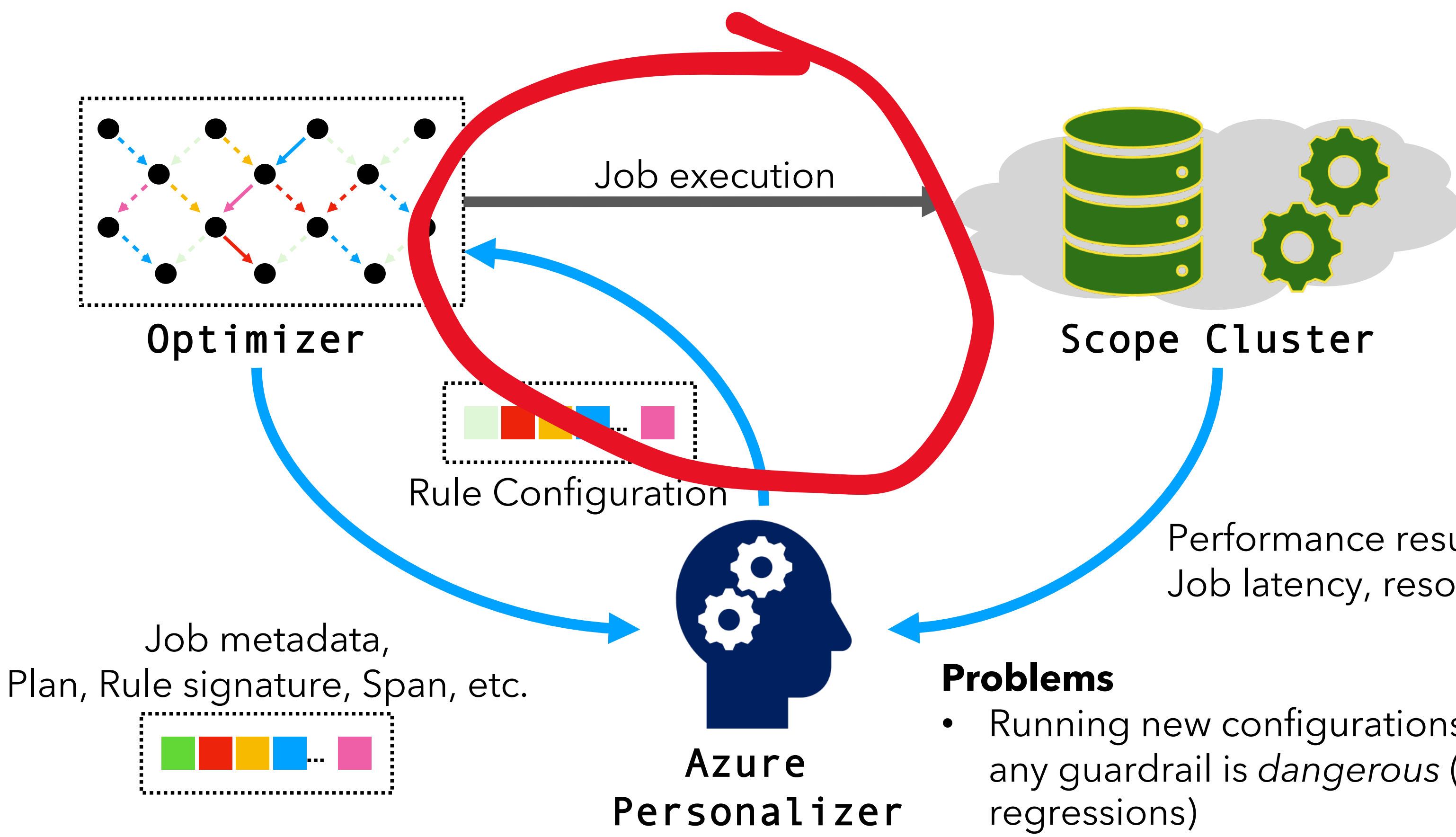
Automation **As RL Problem**



Solution

- Azure Personalizer
- Contextual bandit over 1-bit neighborhood of the span

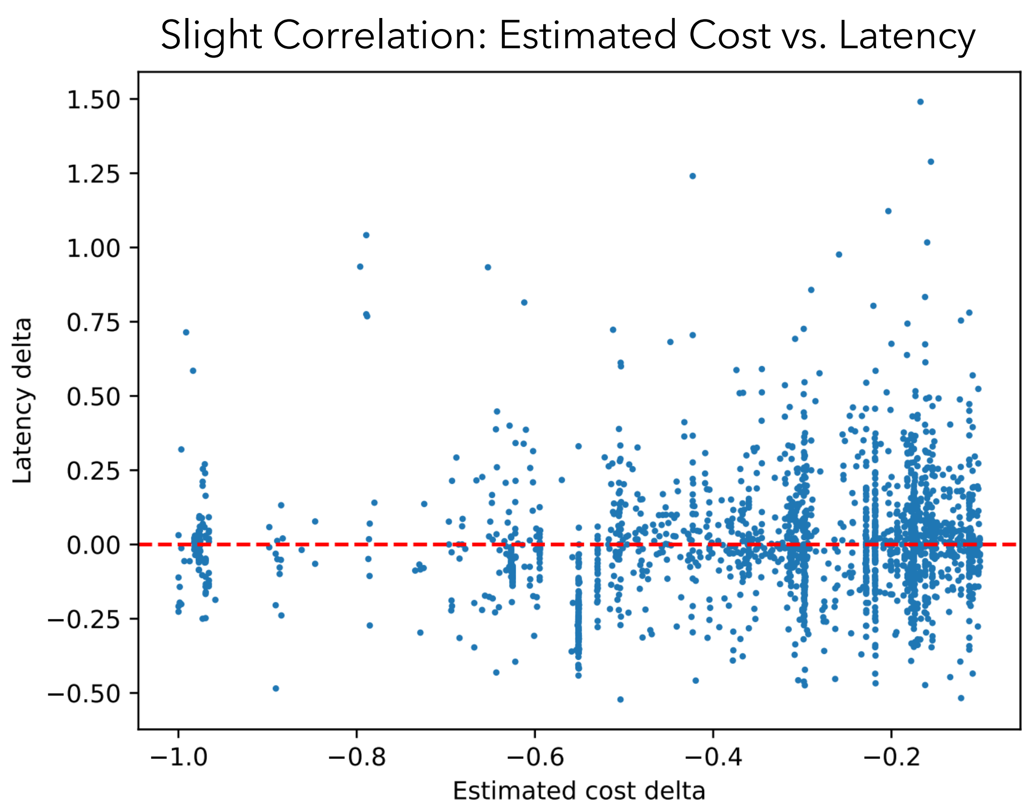
Automation **As RL Problem**



Problems

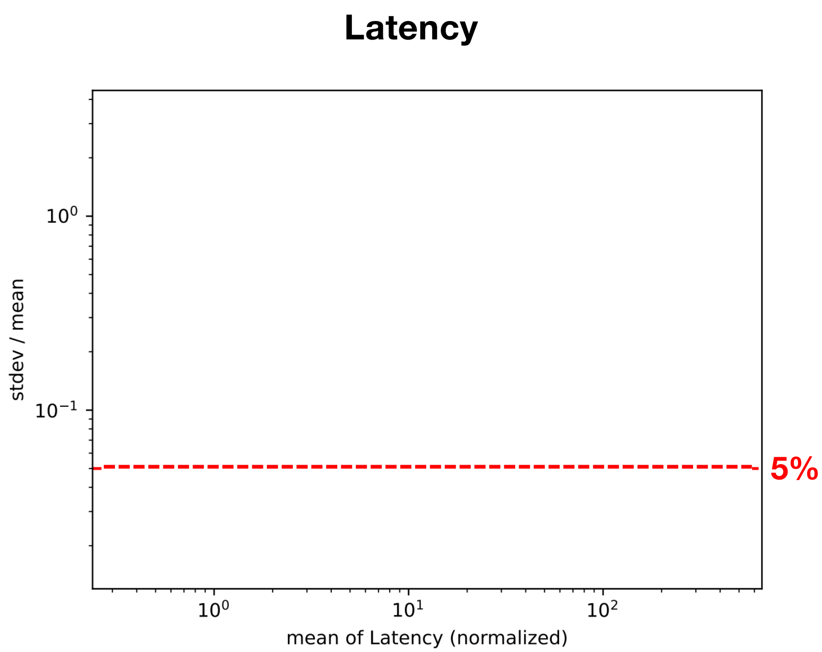
- Running new configurations without any guardrail is *dangerous* (e.g., regressions)
- *Assumption* that the optimizer will find a new *plan with lower cost* **and** *the lower cost plan will execute faster*

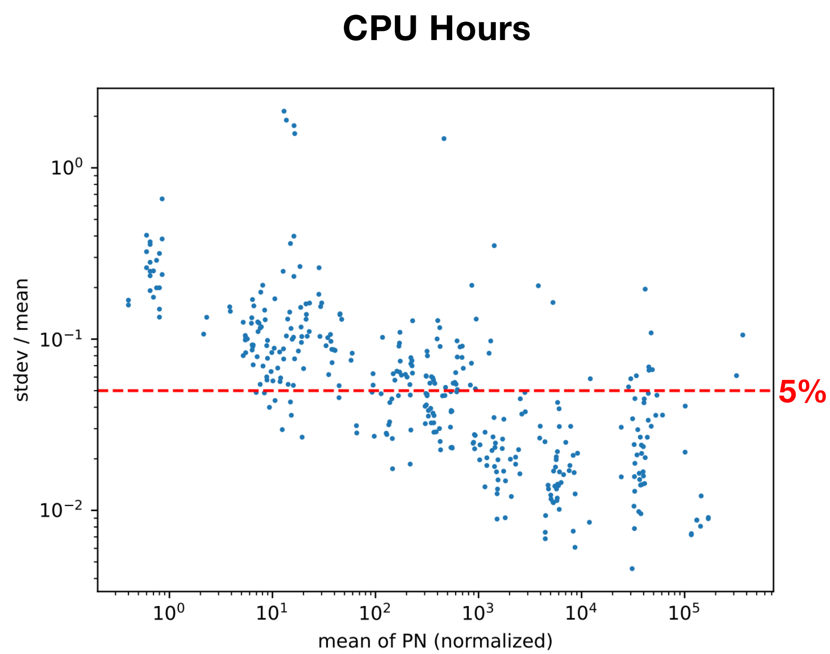
Automation **Lower Costs are not enough**



Conclusion: We need runtime information to avoid regressions. **But** …

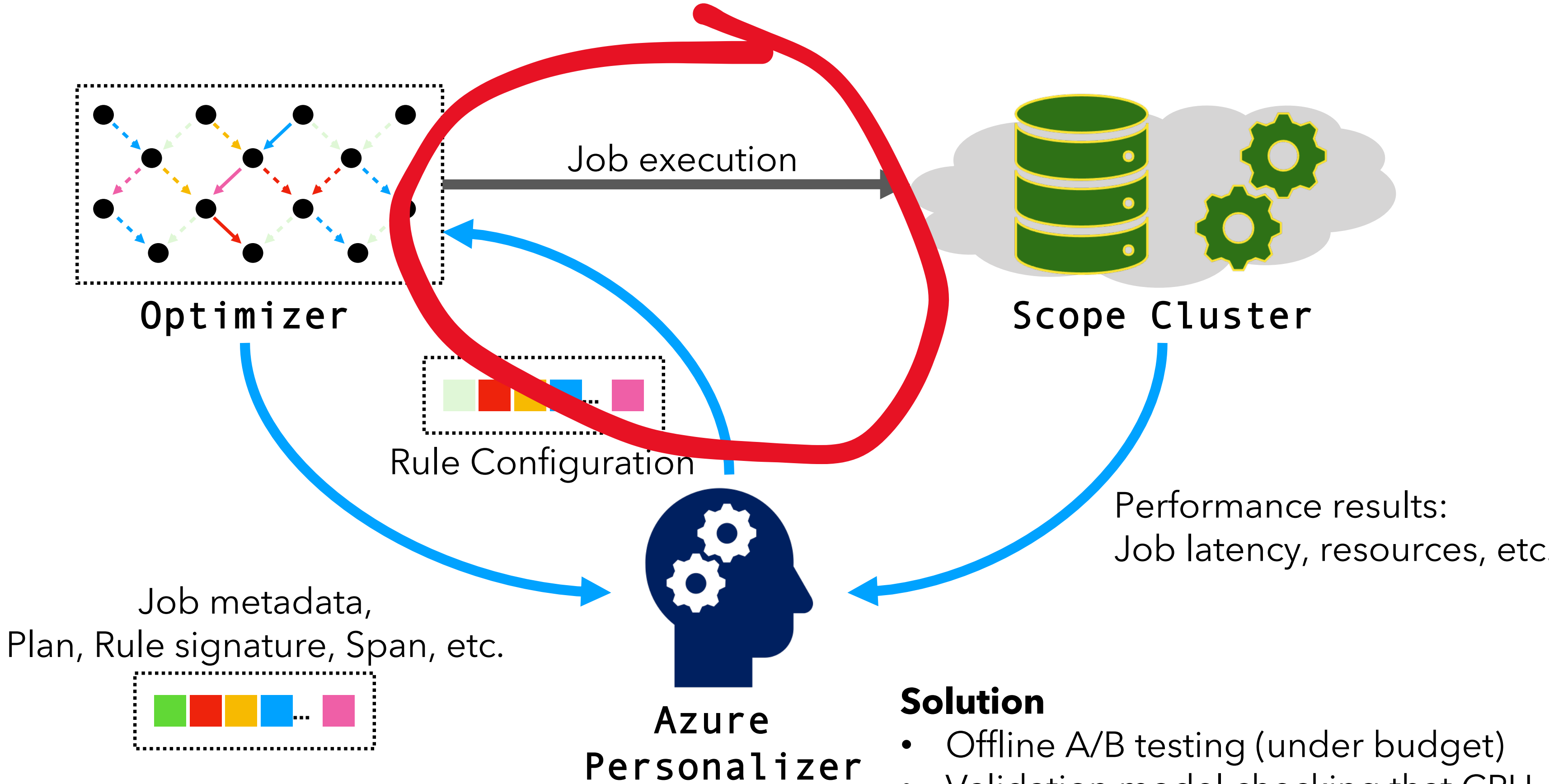
Automation **High Variance makes Learning Difficult**





Run the same job on same data 10 times

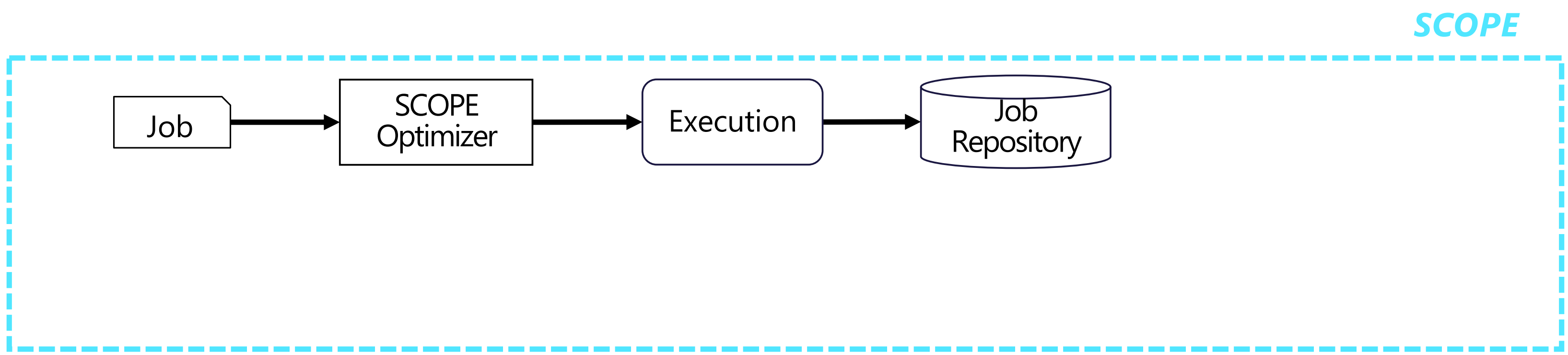
Automation **As RL Problem**



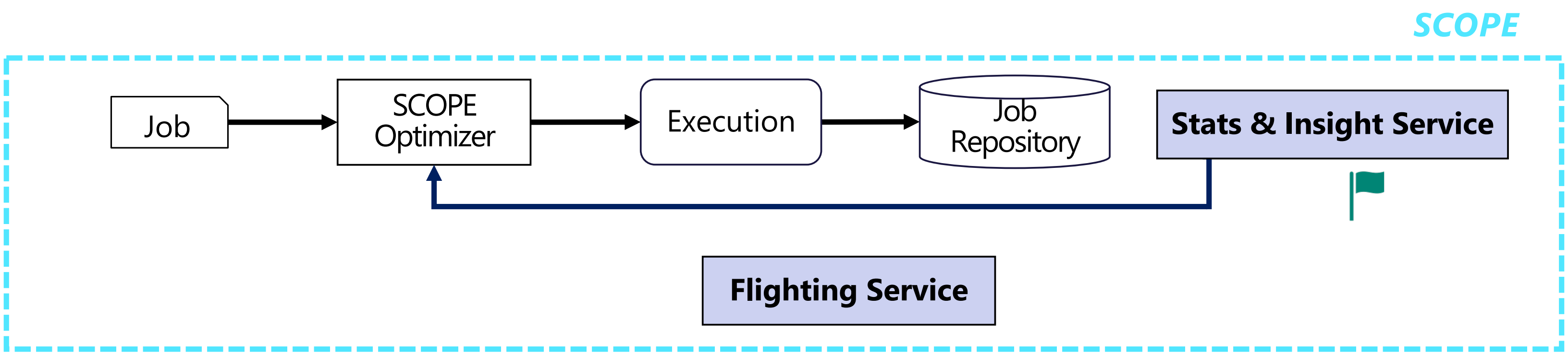
Solution

- Offline A/B testing (under budget)
- Validation model checking that CPU Hours + data read and data written do not increase substantially

Automation **Let's make it Concrete**



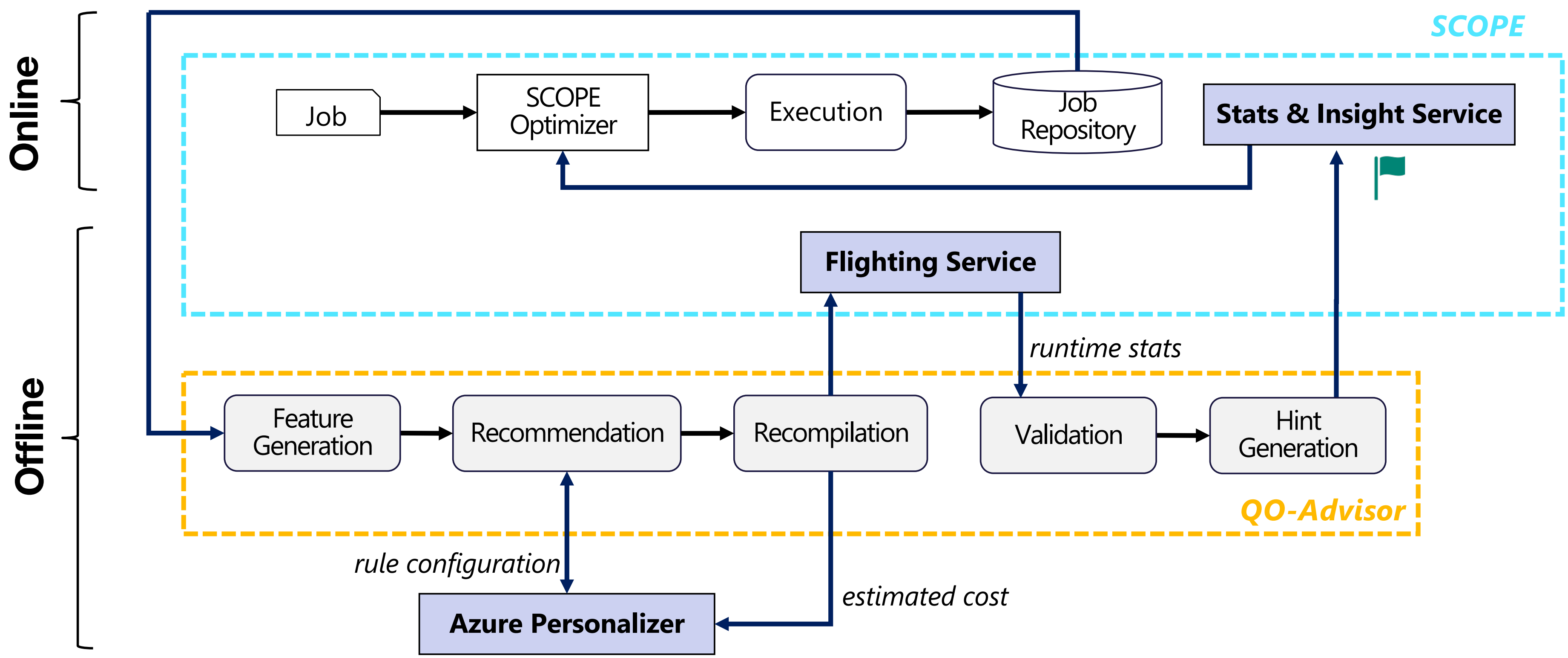
Automation **Let's make it Concrete**



Flighting Service: A/B testing of different query plans

Stats & Insight Service: Serving machine learning models and hints **(e.g.,** **vs** **)**

Automation **Let's make it Concrete**

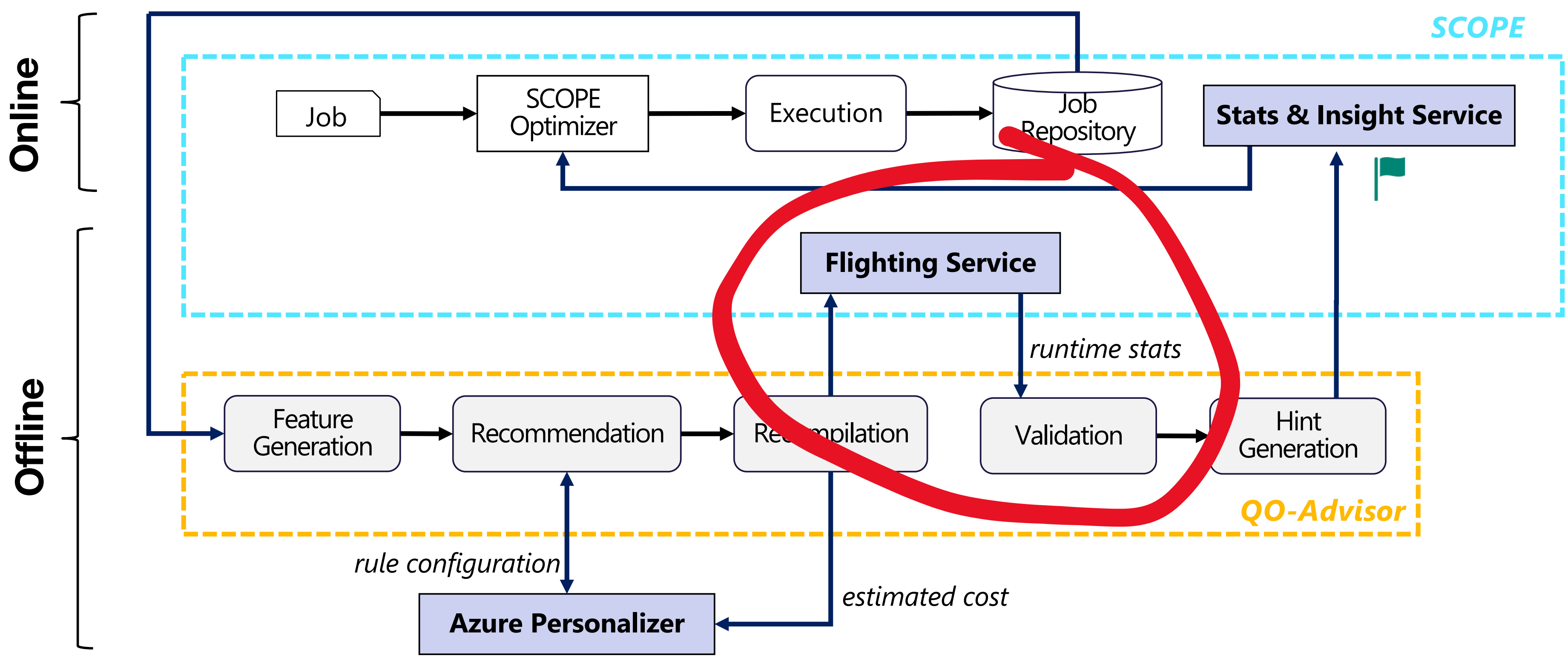


- Pipeline trained daily on ~5% of Scope jobs
- A pipeline run takes about 24h and about 500 vcores
- For jobs using QO-Advisor, 14% overall improvement in CPU Hours
- Generate ~200k events per day for Azure Personalizer

Steering Query Optimizers

Scaling

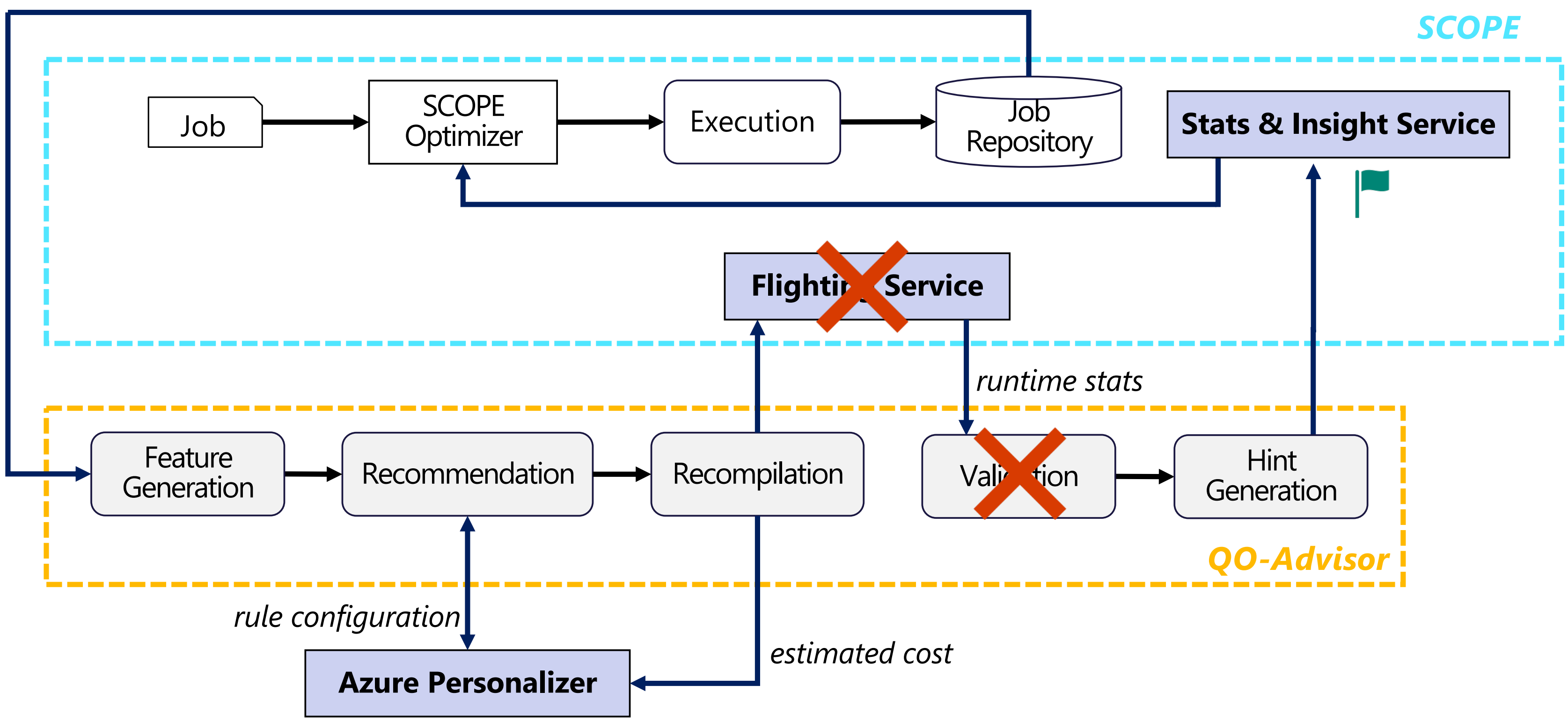
Scaling **Flighting Limitation**



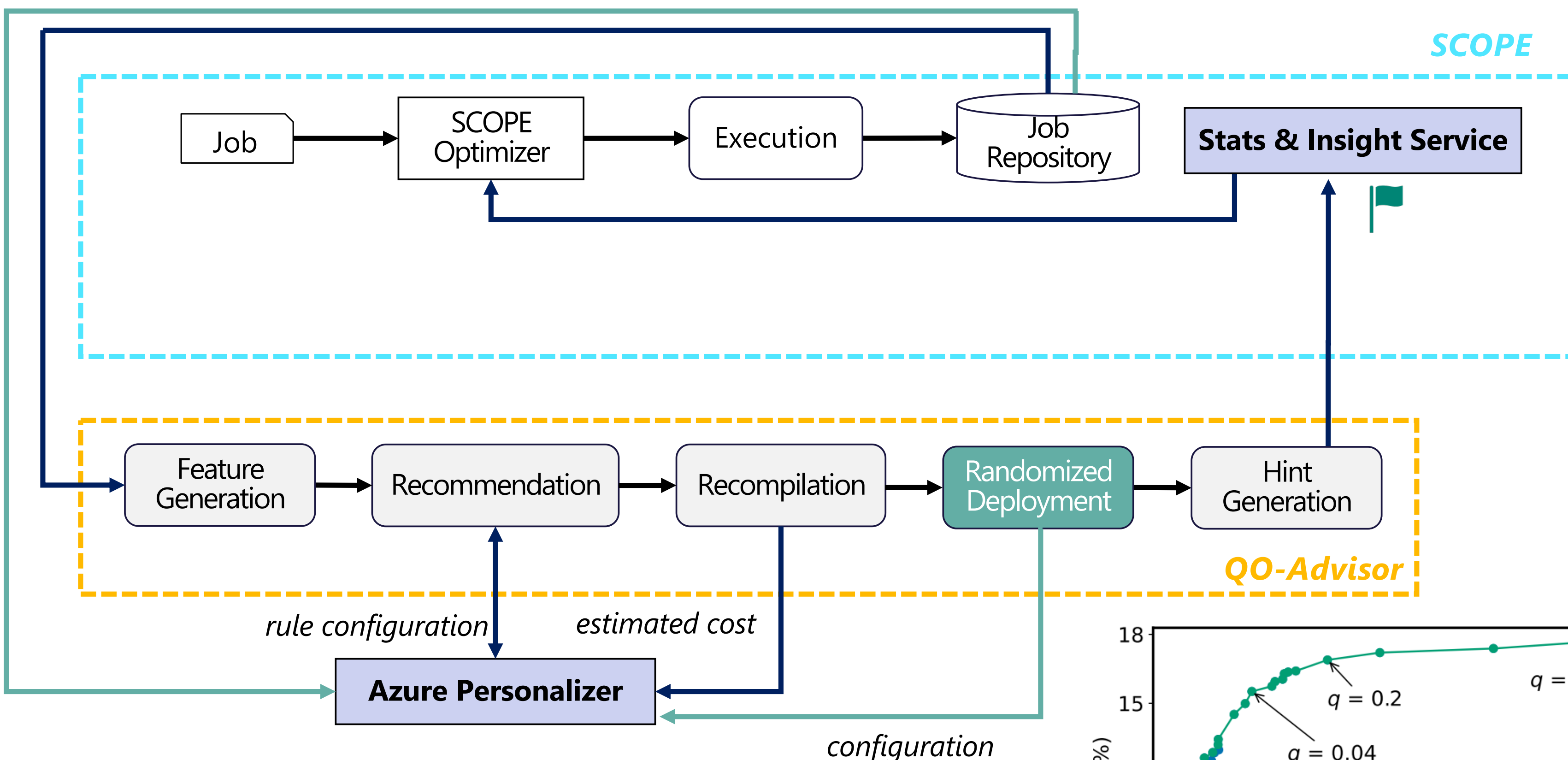
Flighting is limiting the number of jobs we can optimize

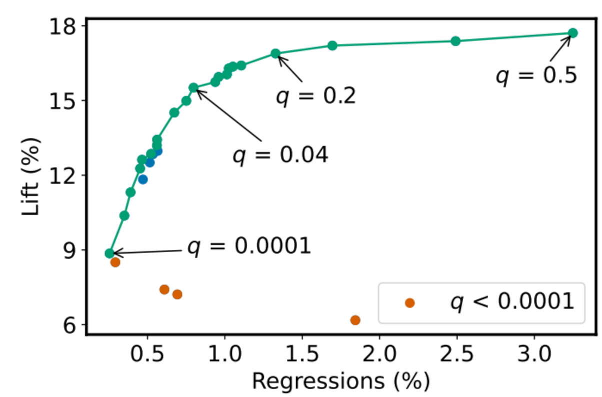
Can we replace offline validation with ***online randomized A/B testing****?*

Scaling **Learning over Repeated Runs**



Scaling **Learning over Repeated Runs**

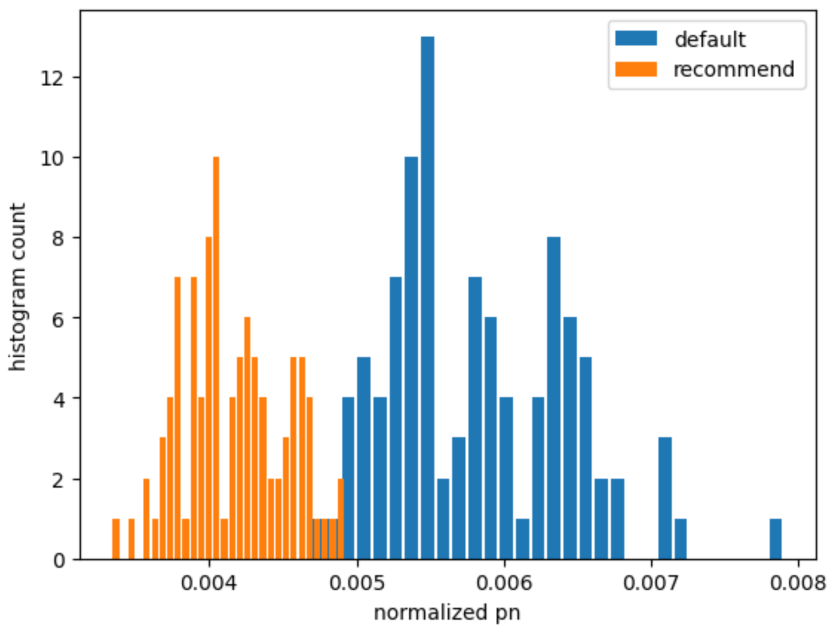




- Informed randomized deployment of configurations
- Risk-adverse contextual bandit

Scaling **Randomized Online A/B Testing Results**

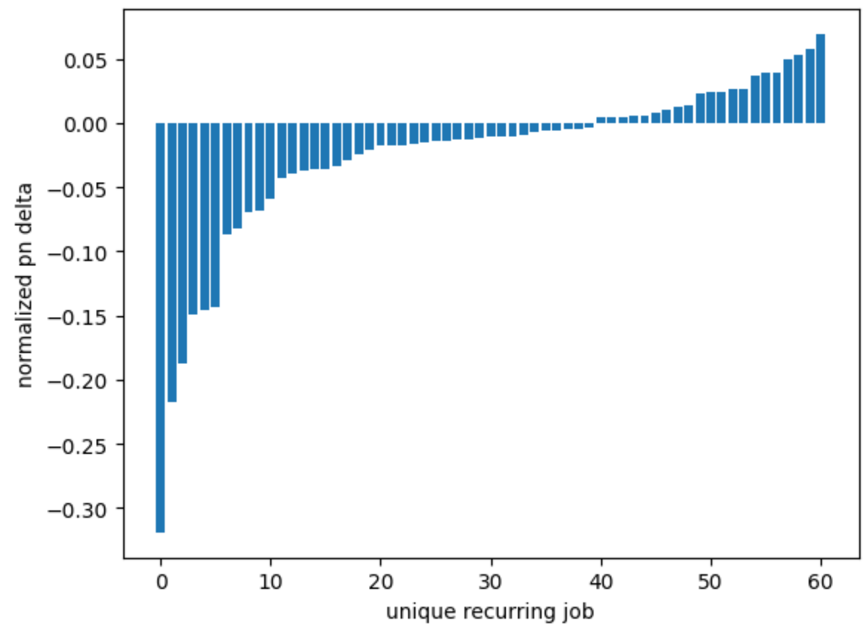
Performance of a cherry-picked job



normalized_pn = CPU_hours / input_data_size

Hint improves normalized PN
by ~25% for this particular job

Performance for each unique recurring job



“delta”: lower is better (PN_recommend/PN_default – 1)

Informed randomization + risk-adverse
algorithm lowers the chances of regressions

Scaling **Summary**

- *Pipeline runtime and resource utilizations*
  - From 24h to ~3h
  - Only uses vcores for recompilation and generating estimates

- *Job Coverage:*
  - We train over all Scope jobs
  - On average, apply hint on 4.84% of recurring jobs

Not as high as before due to risk-adverse algorithm

- *CPU Hours Improvement:*
  - For jobs using QO-Advisor, overall improvement: 7.29%
  - Small regressions for ~30% jobs

More regressions due to online exploration after removing offline validation step

- *We decided to* **shut down the project after ~3 years**
  - Total impact on Scope workload was less than expected
  - Regressions were a major source of concern
  - Azure Personalizer got discontinued

Conclusion/Key Takeaways

1. **This is a hard problem:** state-of-the-art optimizers are hard to beat on average
2. **Inaccurate cost estimates:** can be solved with validation (but it's expensive)
3. **But cannot really ignore cost estimates:** ML models make mistakes
4. **Noisy performance numbers:** ML models have hard time converging
5. **Sparse data:** This is the cost of safety
6. **Online exploration does not work well if you must avoid regressions:** Offline validation is better (but it's expensive)
7. **Featurization of database jobs:** The embedding-based approaches might have a great potential, but we didn't have the time to investigate this path

Thank You!

Wangda Zhang, Paul Mineiro, Shi Qiao, Nasim Ghazanfari, Karlen Lie, Marc Friedman, Rafah Hosn, Hiren Patel, Alekh Jindal, Parimarjan Negi, Ryan Marcus, Mohammad Alizadeh, Tim Kraska