Brandeis
UNIVERSITY

AtlasTune

Multi-Agent Reinforcement Learning for Database Optimization

James Petullo

COSI PhD PhooD Seminar Lecture, November 13, 2024

Database Optimization Overview

- For optimal performance on intended workloads, a database management system (DBMS) needs to be properly configured.
- The configuration task can be broken down into four broad components:

  - Query optimizing
  - Query scheduling
  - Index selection
  - Knob tuning

- The primary emphasis of AtlasTune is the latter-two components.

Index Selection Overview

- Example: find order priority frequencies for the last three months.

orderkey	custkey	orderstatus	totalprice	orderdate	orderpriority
1	53424	F	2323	2024-05-10	LOW
2	23423	O	233	2023-09-18	MEDIUM
3	43422	F	443	2024-10-20	HIGH
4	32322	O	2323	2022-11-22	LOW

```
select o_orderpriority, count(*) as order_count
from orders
where o_orderdate > now() - interval '3' month
            and o_orderdate <= now()
group by o_orderpriority
```

Index Selection Overview

- A possible candidate for indexing:

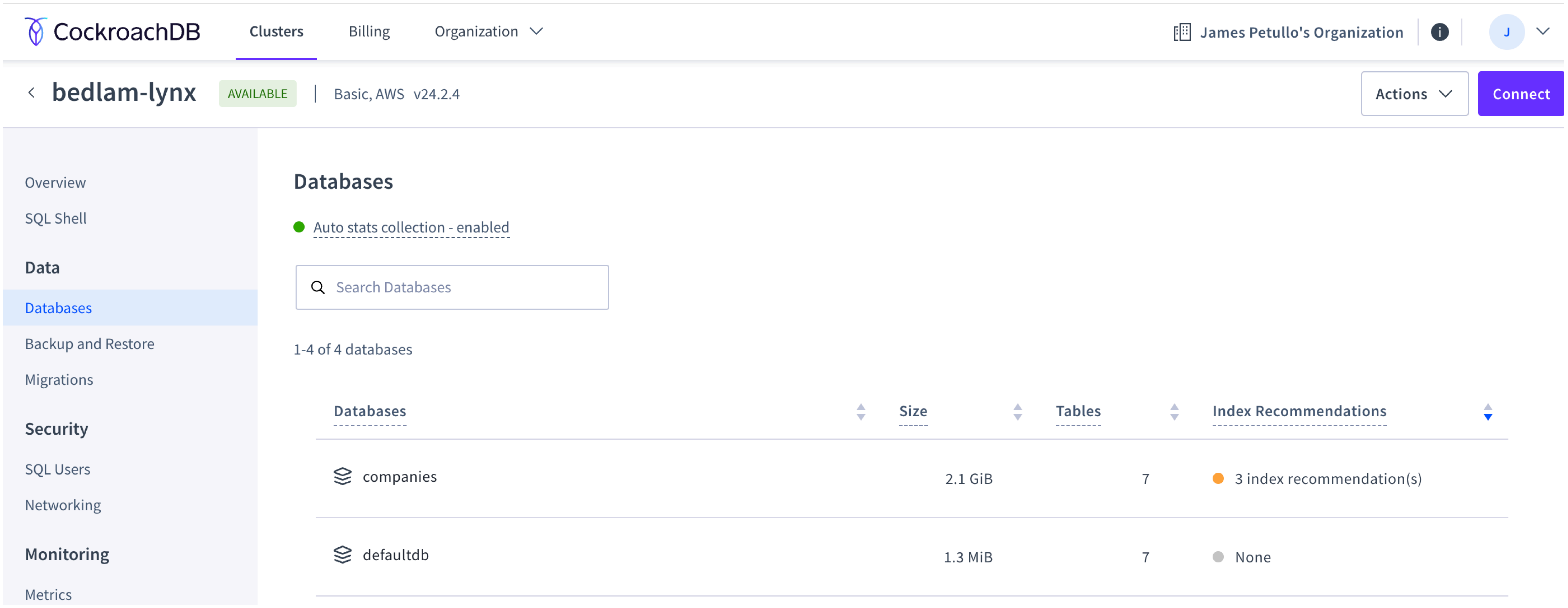
orderkey	custkey	orderstatus	totalprice	orderdate	orderpriority
1	53424	F	2323	2024-05-10	LOW
2	23423	O	233	2023-09-18	MEDIUM
3	43422	F	443	2024-10-20	HIGH
4	32322	O	2323	2022-11-22	LOW

```
select o_orderpriority, count(*) as order_count
from orders
where o_orderdate > now() - interval ‘3’ month
            and o_orderdate <= now()
group by o_orderpriority
```

Index Selection, cont.

- Choosing a subset of columns in a database’s table to index is essential to improving query performance.
- Finding the optimal column subset to index is challenging, as the selector must balance tradeoffs between increasing the speed of read operations, memory usage, and the impact the chosen indexes have on write operations.
- Automatic index selection tools range from simple heuristics to reinforcement learning.

Index Recommendation in a DBMS



Databases

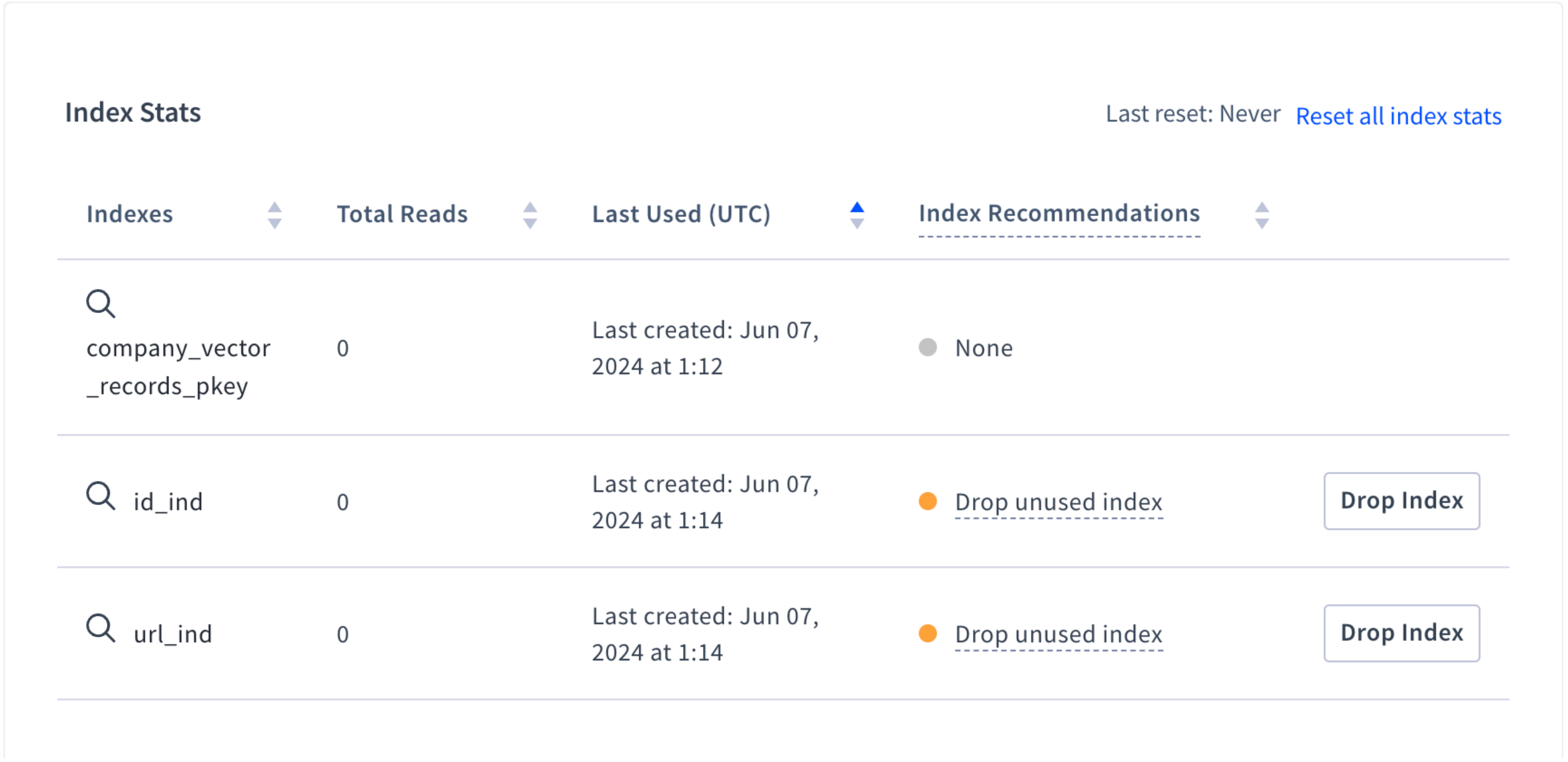
● Auto stats collection - enabled

Search Databases

1-4 of 4 databases

Databases	Size	Tables	Index Recommendations
companies	2.1 GiB	7	● 3 index recommendation(s)
defaultdb	1.3 MiB	7	● None

Index Recommendation in a DBMS



Index Stats

Last reset: Never Reset all index stats

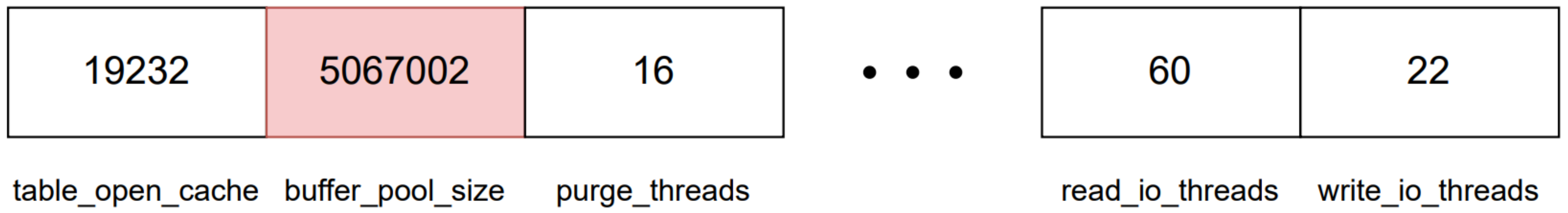
Indexes	Total Reads	Last Used (UTC)	Index Recommendations	
company_vector_records_pkey	0	Last created: Jun 07, 2024 at 1:12	● None	
id_ind	0	Last created: Jun 07, 2024 at 1:14	● Drop unused index	Drop Index
url_ind	0	Last created: Jun 07, 2024 at 1:14	● Drop unused index	Drop Index

Knob Tuning

- DBMSs have hundreds of configuration parameters (PostgreSQL has over 350 knobs).
- Proper configuration values are vital for database performance, particularly on large systems with complex workloads.
- Knobs have many intricate interdependences in their own right, and it is non-trivial to tune by hand, especially considering the fact that the range of values many knobs can be set to are very large (1 to 100000, in some cases).

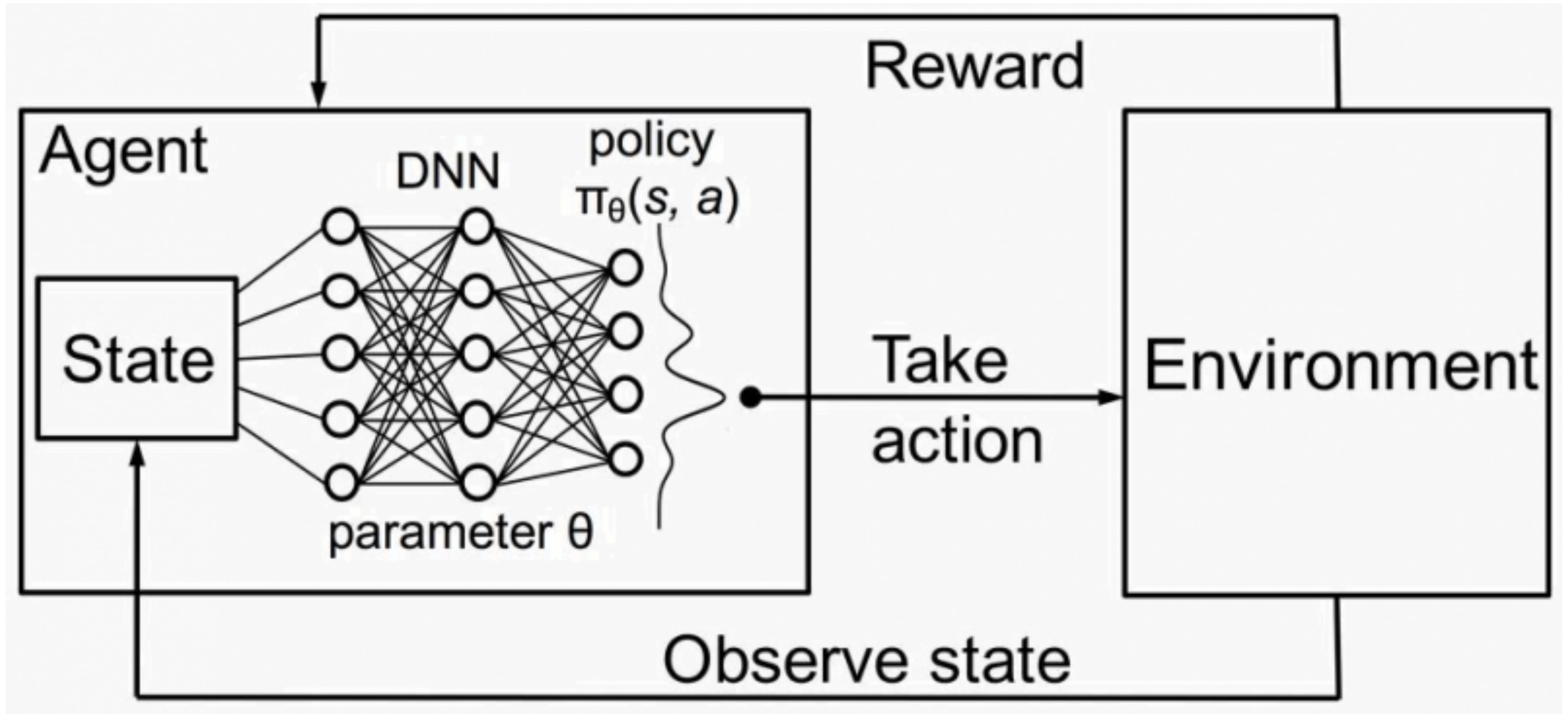
Knob Tuning

- Sample knob value configuration:

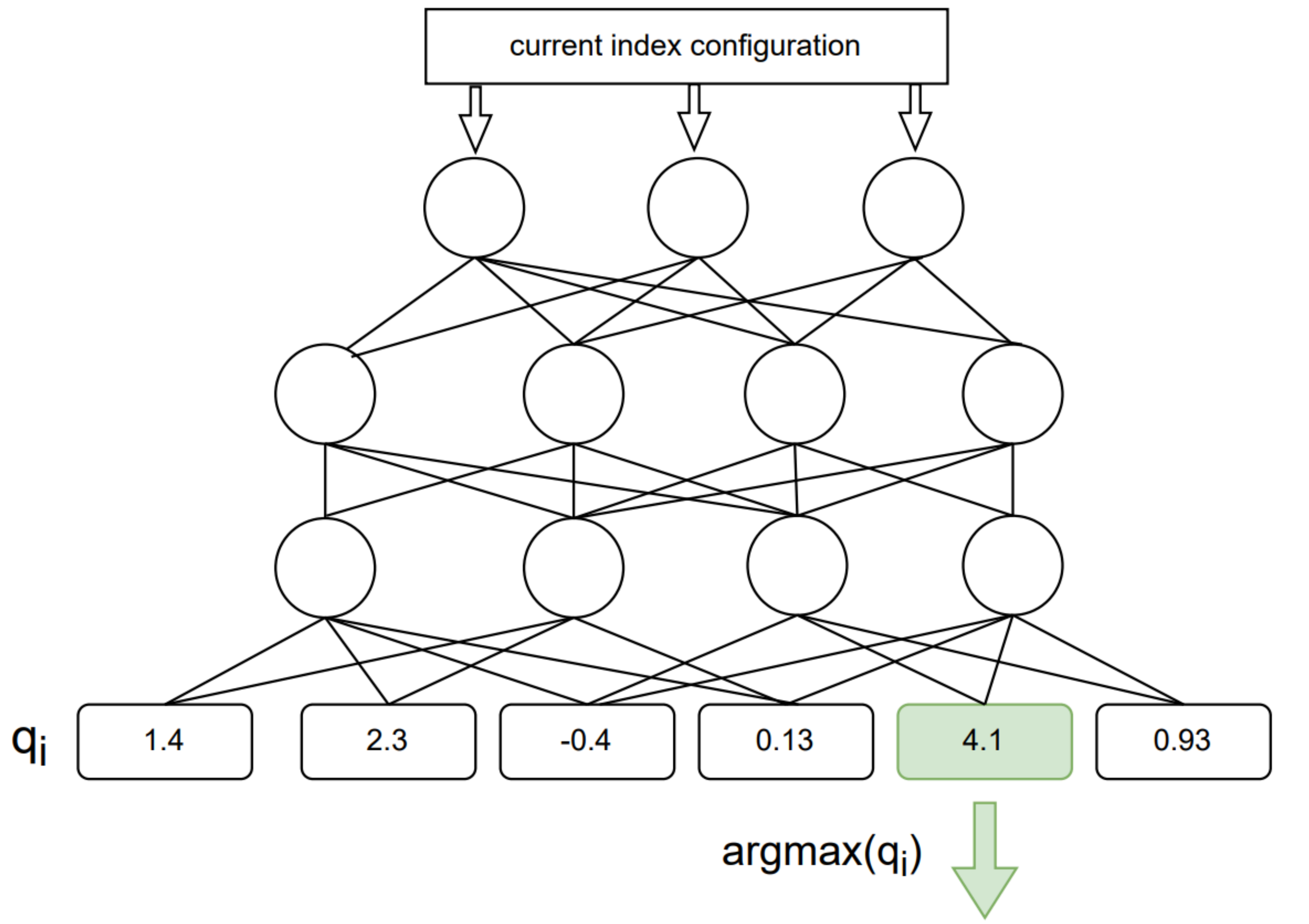


Component Tuners

- Components are tuned using deep reinforcement learning:

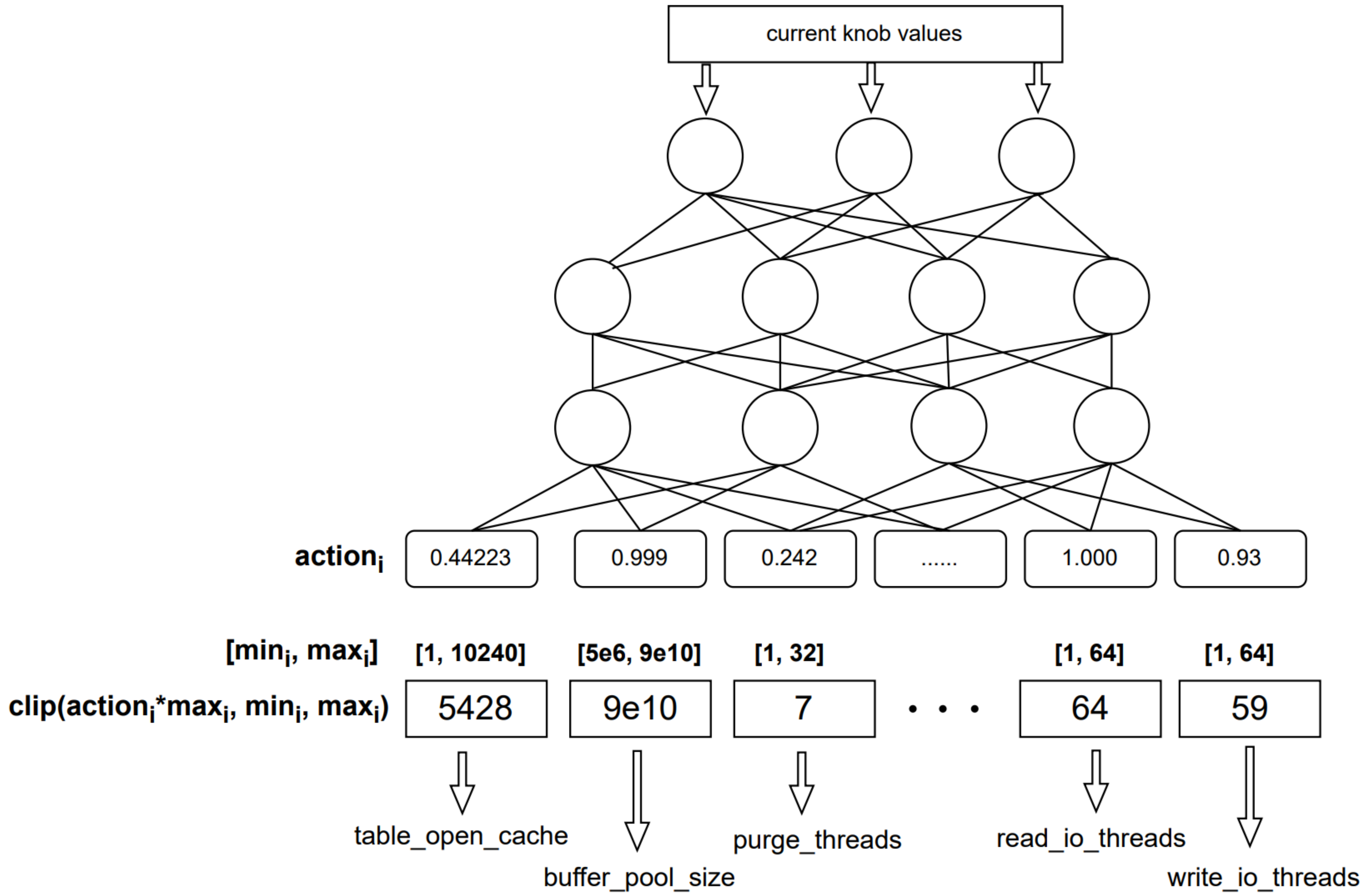


RL for Index Selection



orderkey	custkey	orderstatus	totalprice	orderdate	orderpriority
1	53424	F	2323	2024-05-10	LOW
2	23423	O	233	2023-09-18	MEDIUM

RL for Knob Tuning



Component Tuners

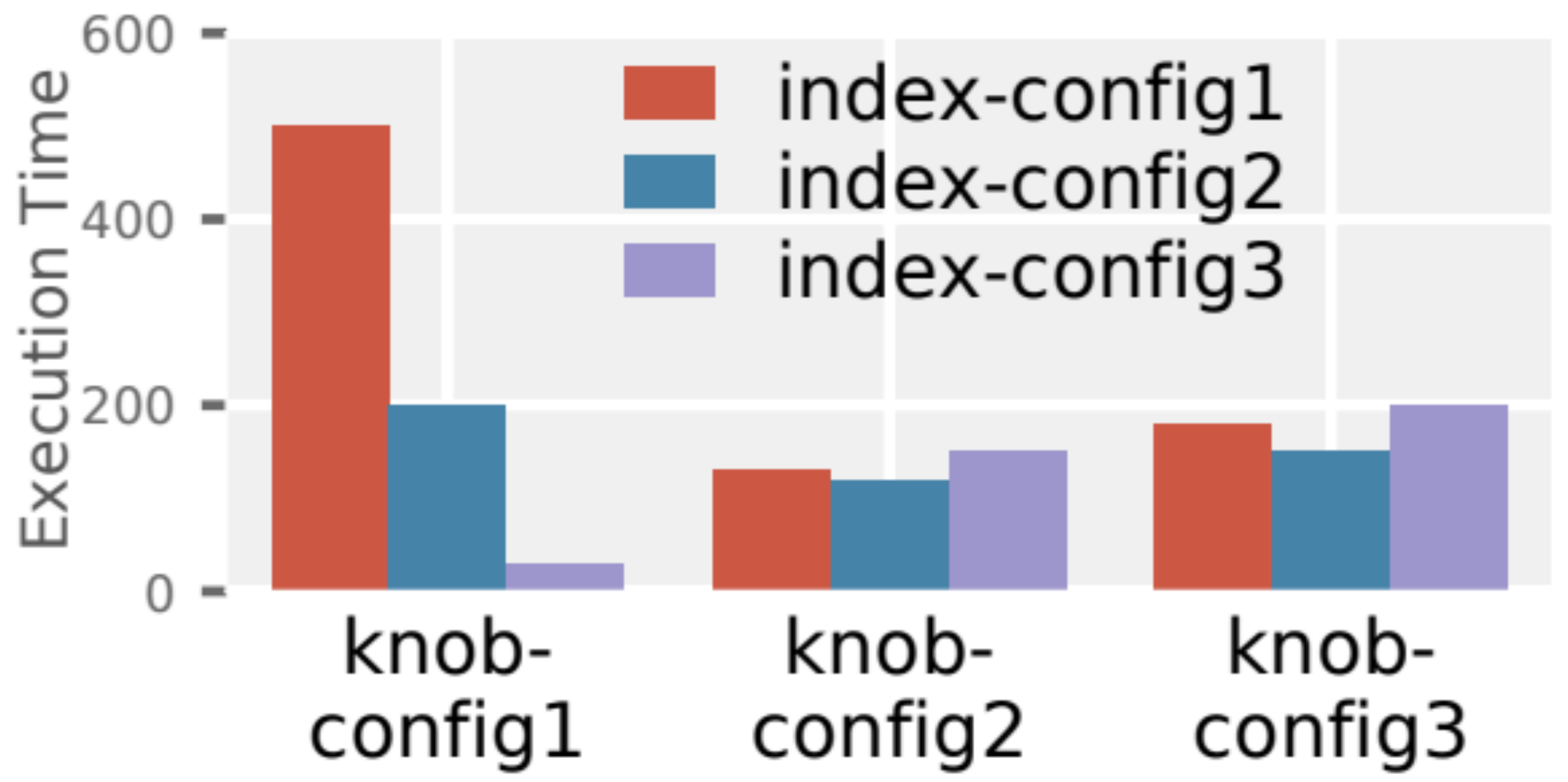
- The tuners explore the search space for a time by randomly generating configurations and observing the change in database performance. Gradually, exploration decreases and the model itself is used to output a target configuration (exploitation).
- Each iteration's state, selected action, reward, and new state are saved to a memory buffer.
- At each iteration, a random subset from the buffer is taken and the model is trained on that chosen sample.

Component Interdependencies

- The standard optimization approach is to run each individual tuner on every component sequentially.
- However, this process does not take into account the interdependencies that exist between components, whereby the optimal configuration of one component depends on the configuration of the other components.
  - For example, a large query cache and small buffer size (knob tuner) is best when no indexes are built (index selection) while a smaller cache size and larger buffer pool are more suitable when indexes are built.
- If these interdependencies are not accounted for, tuners will ultimately suggest suboptimal configurations that can degrade overall DBMS performance

Component Interdependencies, cont.

- Example of workload execution times under different knob and index configurations. Index-config3 and knob-config1 together perform best, as opposed to the others.



(b) Dependency Between DBMS Components.

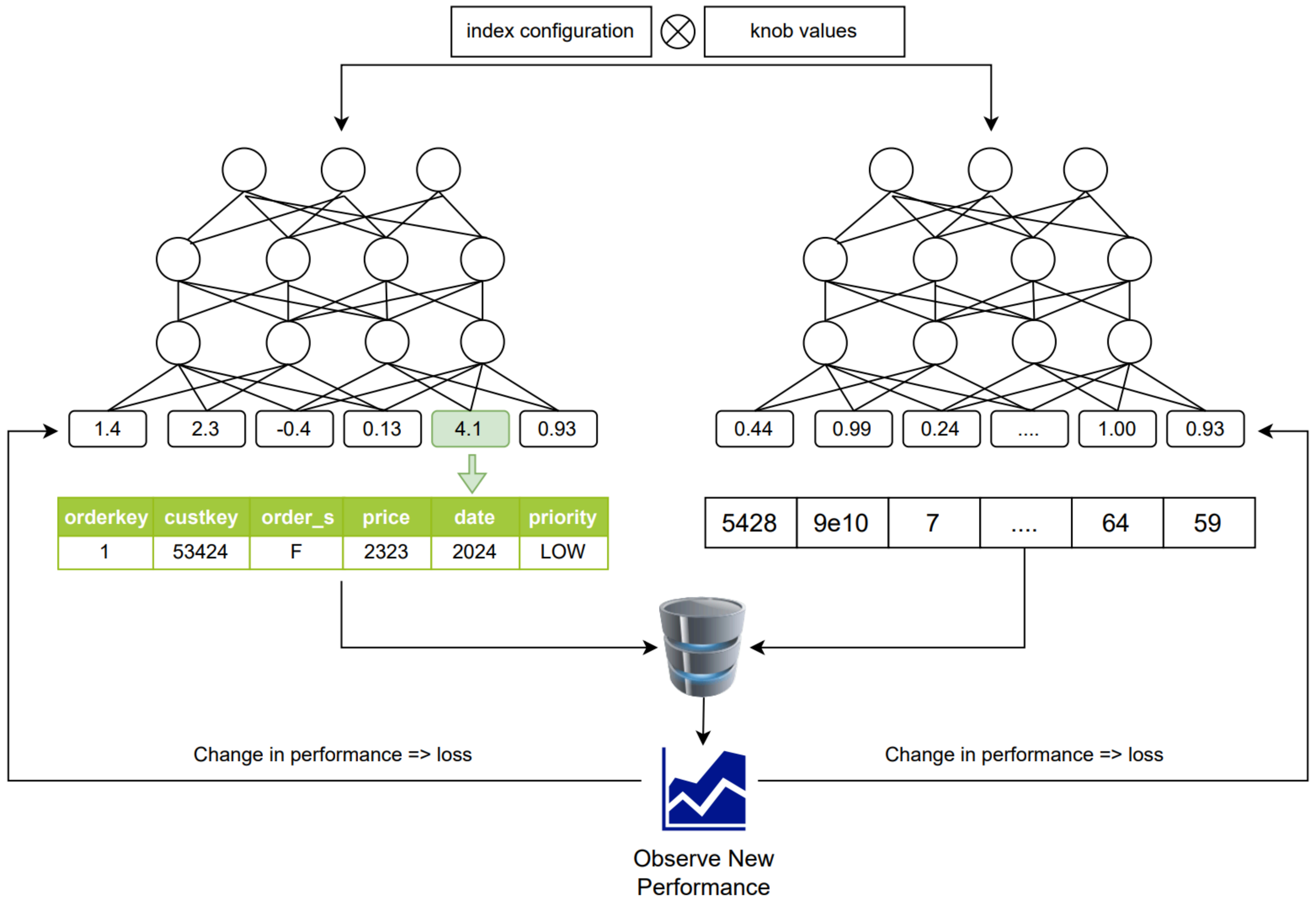
Previous Work and Existing Solutions

- The literature on this problem has been very sparse.
- Prior work has consisted of alternating training schedules, state sharing, tuning budget allocation, workload classification (i.e what kind of queries are being used), and more.
- In most cases, the workload queries are encoded and fed into prediction models, while considerable assumptions are made with regards to what cross-component actions have the strongest interdependencies.
- Ideal goal: an intelligent multi-component tuner should be workload-agnostic, modular, and robust to changes in workload composition and reward signals.

Proposed Solution Using MARL

- MARL: multi-agent reinforcement learning.
- State sharing: each tuning component's state includes not just its own default state but that of the other agent.
- Team reward: both the knob tuner and the index selector have the same reward function. Each iteration, a set of queries is run to produce latency and throughput readings, which are then used as a reward signal.
- Interleaved tuning: instead of running one agent to convergence and then deploying the next agent, we alternate between tuners, pausing the execution of one after a certain number of iterations, fixing the best configuration found, and then switching to the next agent. This way, each agent has the chance to adapt to into intermediate decisions and compensate for any suboptimal solutions the other makes.

Proposed Agent Schema

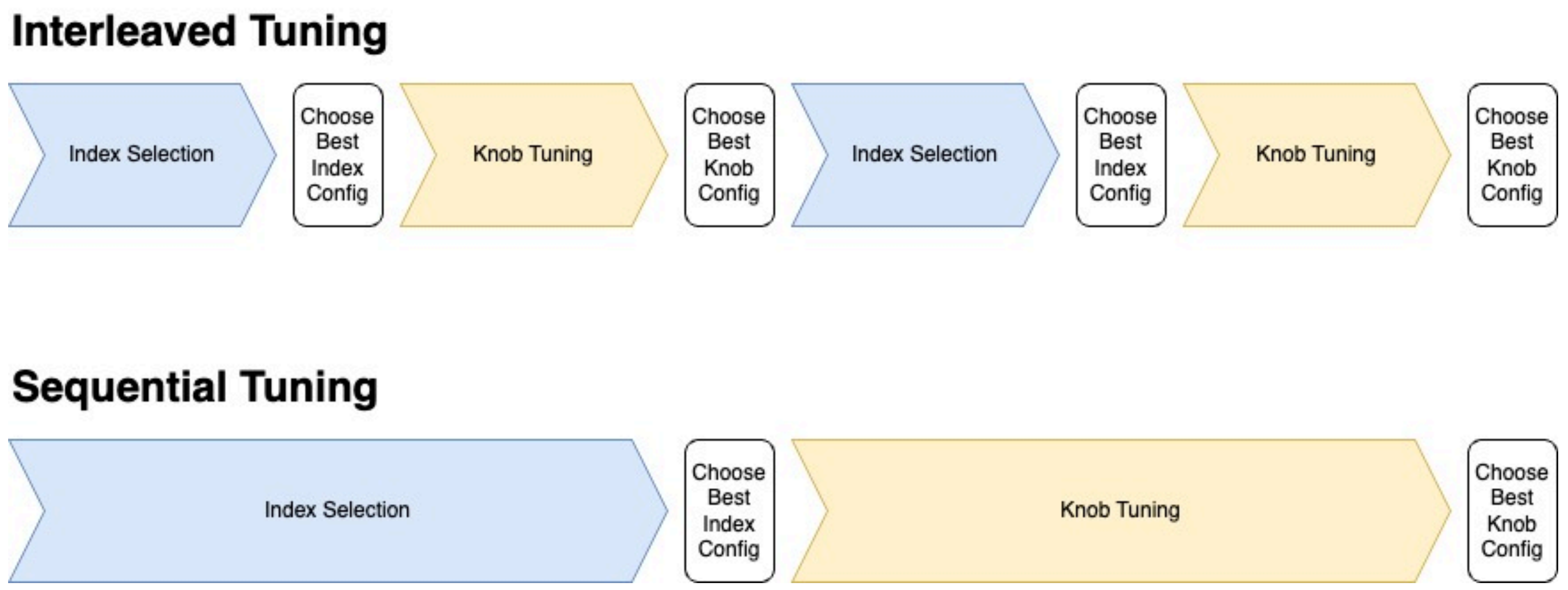


orderkey	custkey	order_s	price	date	priority
1	53424	F	2323	2024	LOW

5428	9e10	7	....	64	59

Proposed Solution, cont.

- Interleaved tuning: at successive intervals, an agent's best configuration is fixed and tuning switches to the next agent:



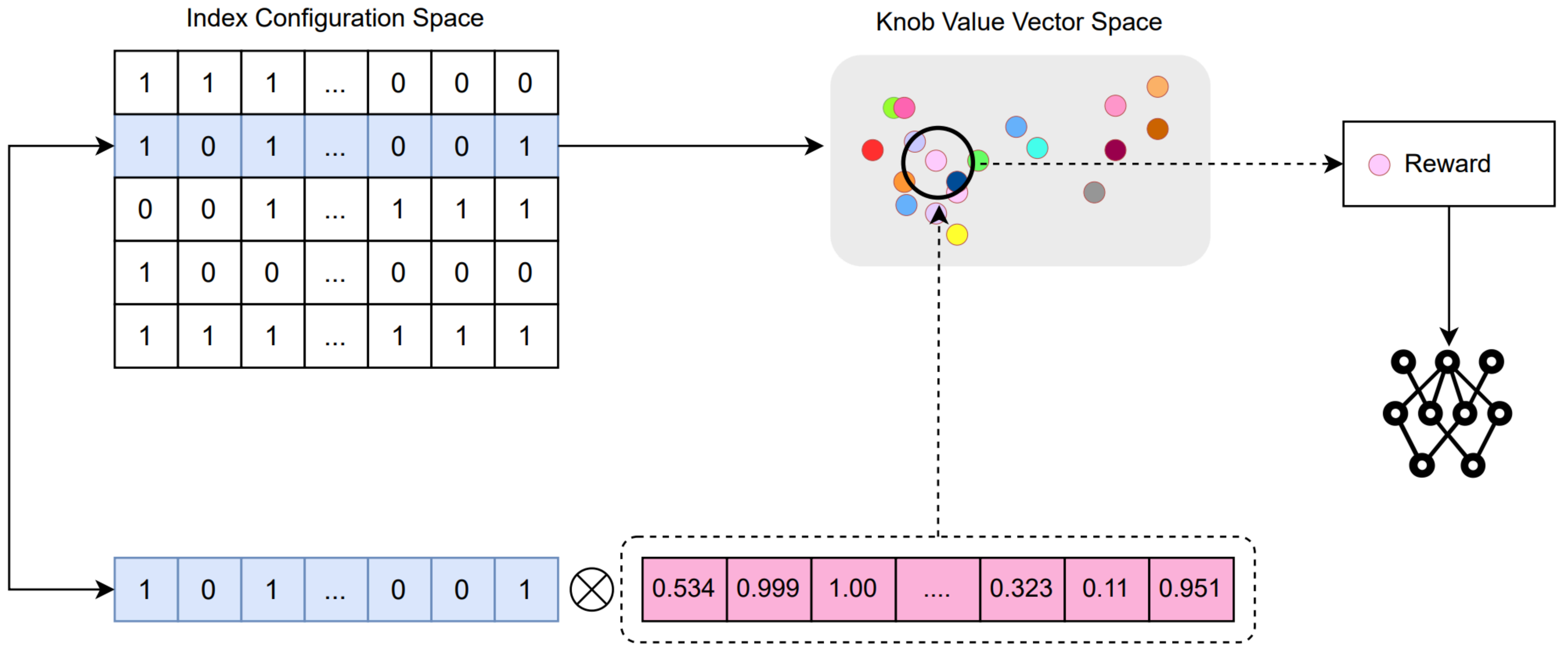
Rewards and Performance

- When a new configuration is deployed, a set of queries (the workload) is run to gauge the throughput and latency. The reward function takes those two values and produces a scalar value.
- However, if a configuration *C* and a workload *W* are applied and run at two separate iterations *i* and *j*, the reward at *j* is often not the same as the reward at *i*. This is due to the composition of the workload, DBMS background processes, etc.
- This implies that for the same input states, there are two or more different outputs that the model has to account for, making learning much more difficult.
- Solution: save prior configuration-reward pairs to a cache and perform a lookup when a new action configuration is produced.

Action Space Search

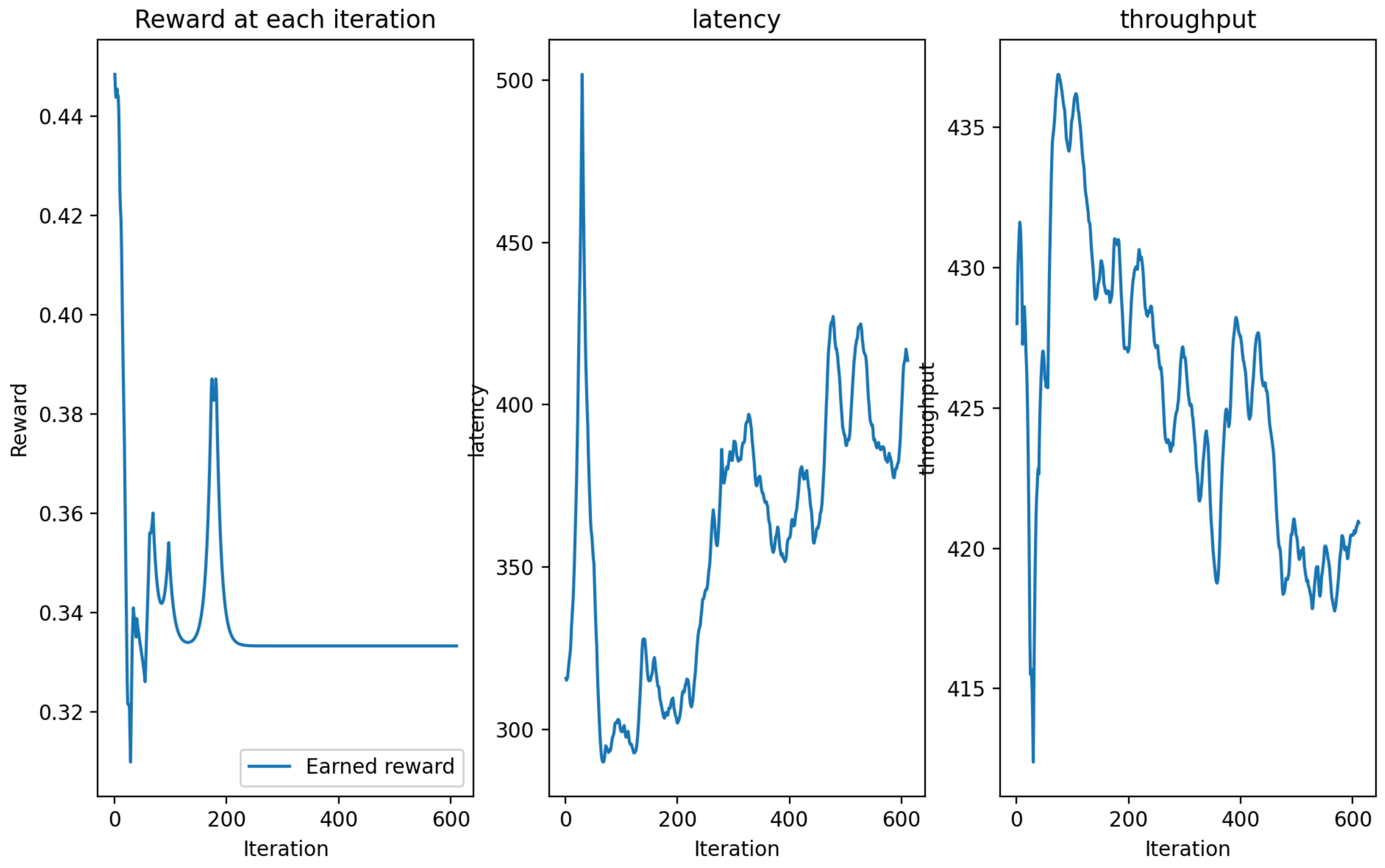
- Action configurations are vectors storing knob and index values. Given a new configuration, the vector space of actions can be searched to find similar action observations and their associated reward.
- The search consists of a two step process:
  - First, the new action's index configuration is used to filter candidate solutions from the space via direct comparison with each candidate action's index configuration. Index configurations are considered discrete, as their relative proximity in the search space is not an accurate gauge of the effect that their individual indexes have on the workload runtimes.
  - From the filtered candidate action space, cosine similarity is used to find the closest neighbor to the action's knob value vector, given a distance threshold. The closest neighbor's reward value is then returned.

Action Space Search, Visualized



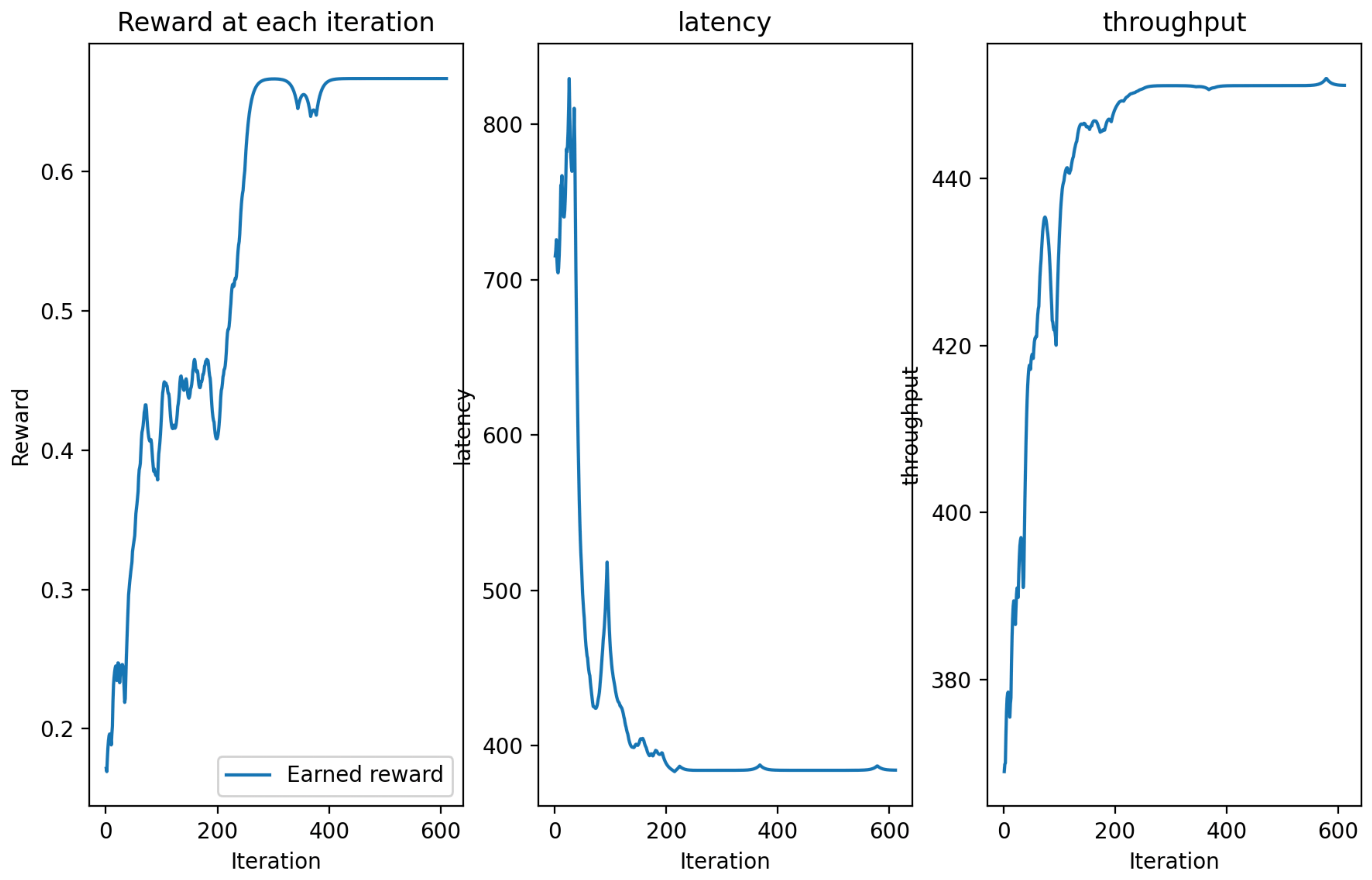
Knob Tuning Without Caching

- Workload execution time = 60 seconds/iteration, training time = 10 hours



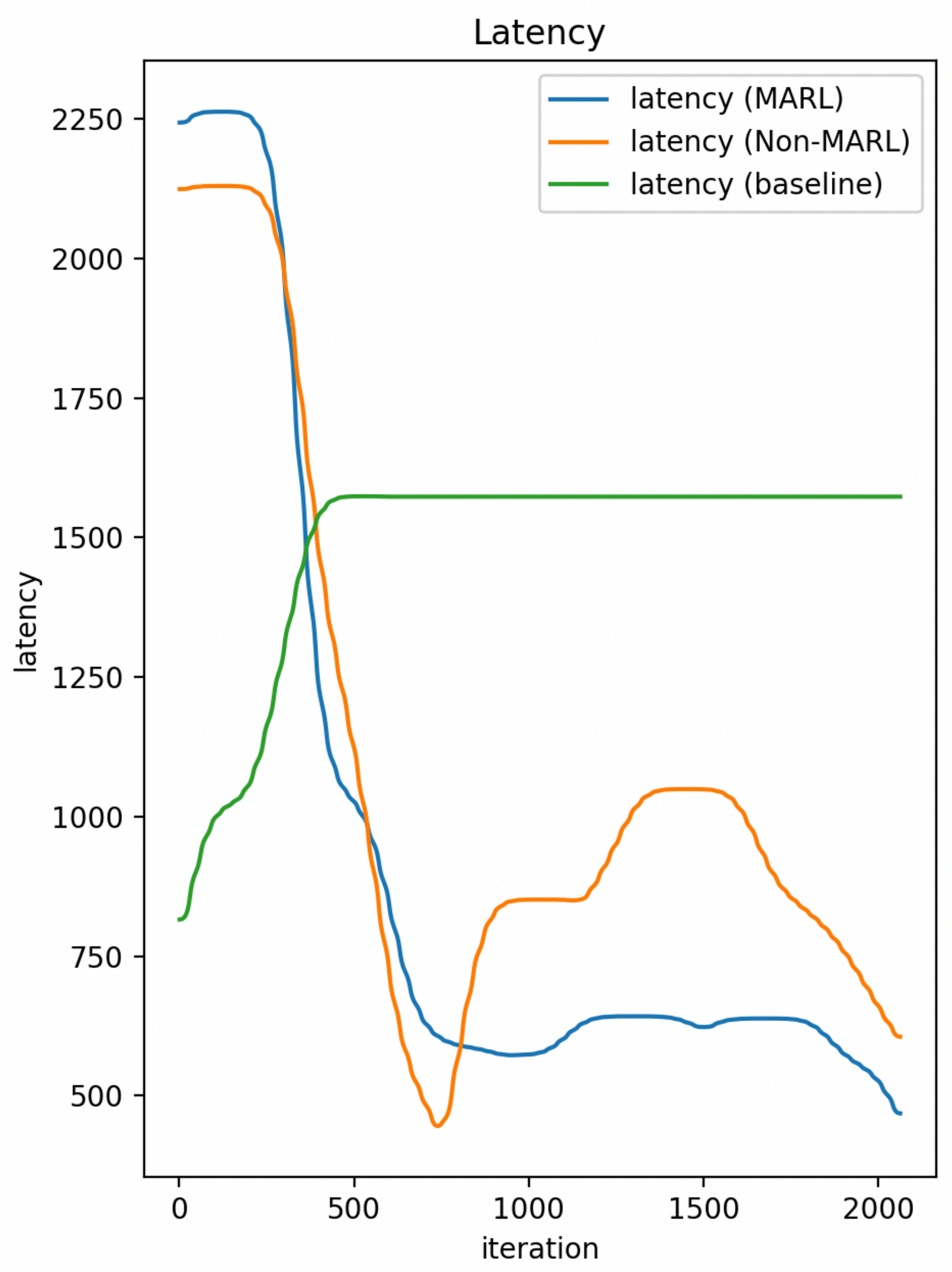
Knob Tuning With Caching

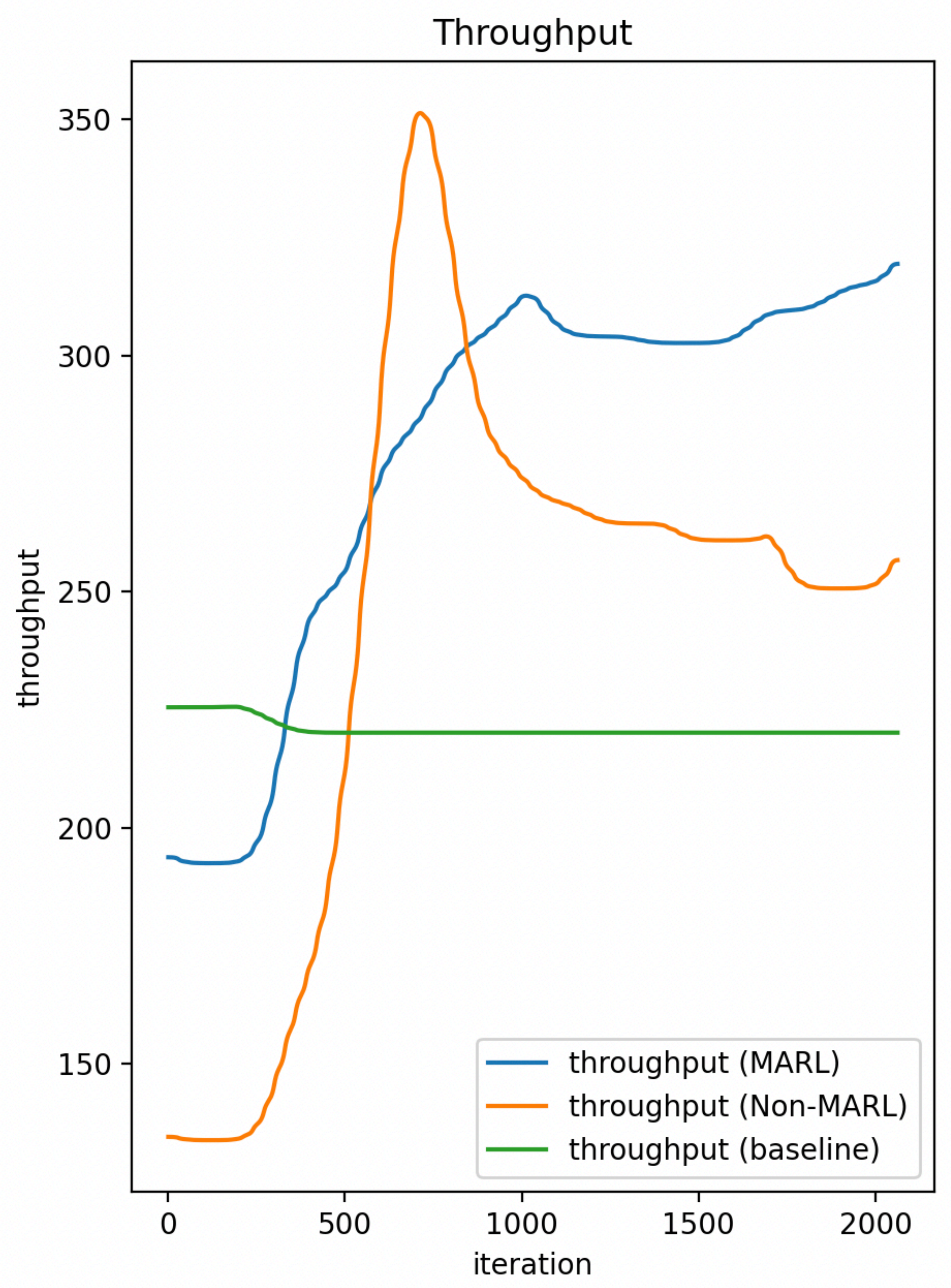
- Workload execution time = 60 seconds/iteration, training time = 2.5 hours



Preliminary Results

- Sysbench OLTP workload on 3.5 GB database:





Analysis

- As expected, RL-based tuners outperform the DBMS's baseline configurations.
- In particular, the MARL tuning schema outperforms the naive, sequential deployment of individual component tuners.