LLMs for Enhanced Code Review

&

Optimizing Compute Cluster Efficiency with Reinforcement Learning

Alexey Rybalchenko
Software Development for Experiments (SDE) group, GSI Helmholtz Centre for Heavy Ion Research

GSI/FAIR AI Workshop
GSI, October 29, 2024

2 AI Projects

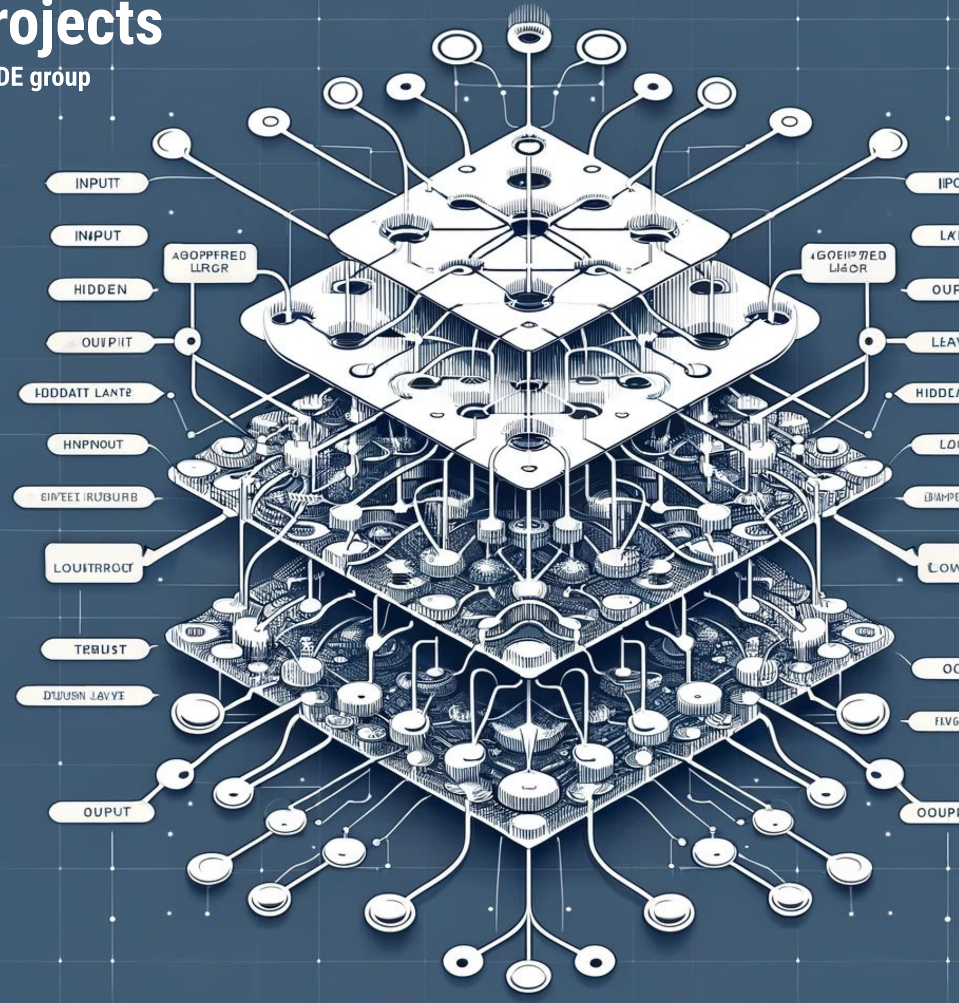
in the SDE group

1. PR code review assistance with CodeRabbit.ai (GPT-4/3.5) & Pearbot (local models).

   → **Focus on LLM usage & tooling.**

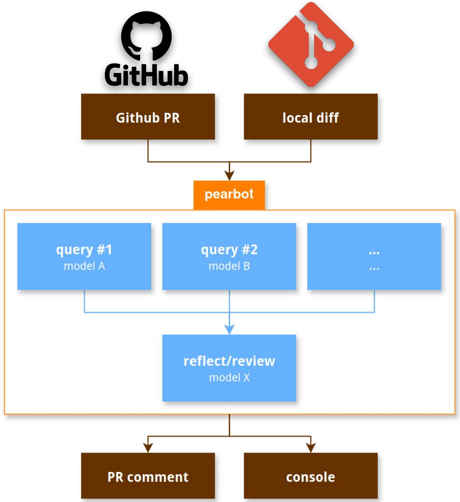
2. Train a model with Reinforcement Learning to suggest cluster job scheduling based on the electricity price predictions.

   → **Focus on model training with Reinforcement Learning.**



LLMs for Enhanced Code Review

Pearbot:



GitHub

Github PR

local diff

pearbot

query #1
model A

query #2
model B

...
...

reflect/review
model X

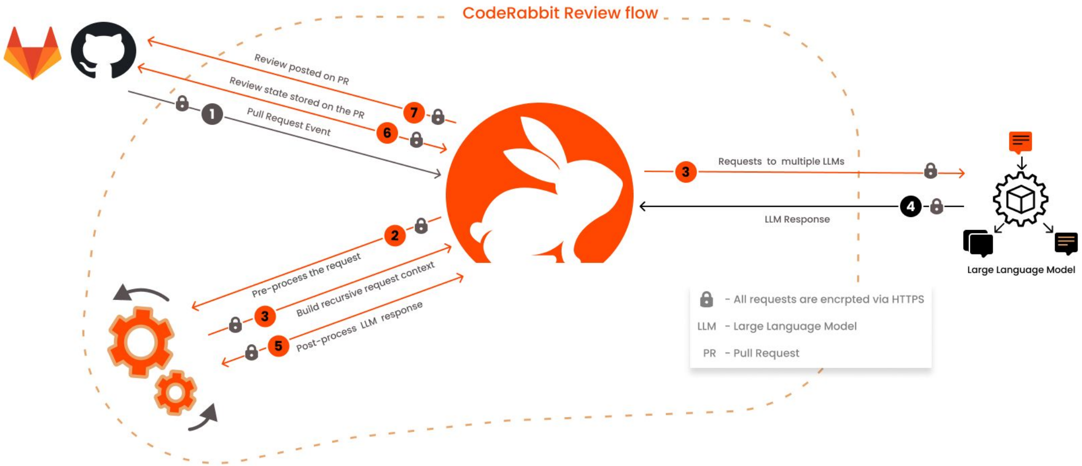
PR comment

console

https://github.com/GSI-HPC/pearbot

- GitHub App for reviewing Pull Requests.
- Local execution mode for diffs or annotated commits.
- Agent ensemble approach for improved results.
- Customizable model(s) via the ollama setup.
- Execution on low-end hardware and/or without GPU.
- Customizable prompt(s).

CodeRabbit:



CodeRabbit Review flow

Review posted on PR

Review state stored on the PR

Pull Request Event

1

7

6

2 Pre-process the request

3 Build recursive request context

5 Post-process LLM response

3 Requests to multiple LLMs

4 LLM Response

Large Language Model

🔒 - All requests are encrpted via HTTPS

LLM - Large Language Model

PR - Pull Request

https://coderabbit.ai/

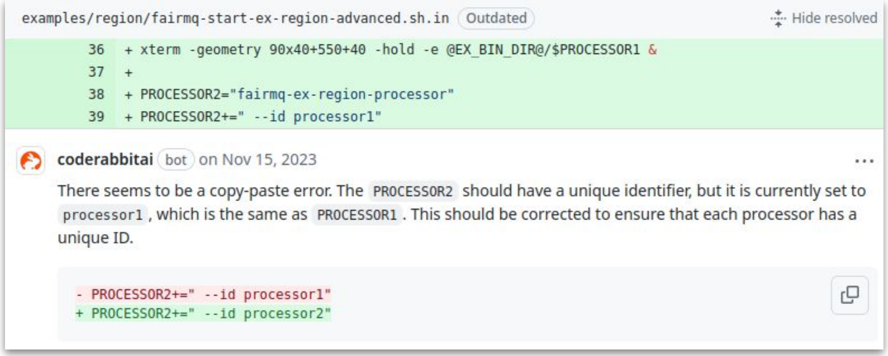
- AI-based Pull Request reviewer via a Github/Gitlab App.
- Combination of various OpenAI models.
- Previously open source, now a closed project.
- Free to use for open source projects.

CodeRabbit

Interesting Results

+ Unbiased, 24/7, scalable, multi-lingual, customizable, with a broad knowledge base, can be kept up-to date with new data.
+ Good at **identifying logical errors** early, that other automated tools or even human review may overlook.
+ Can potentially reduce the time and resources needed for code reviews.

- May produce unnecessary output, when no actionable changes are necessary or such are not deduced by the model.
- Weaker models easily dive into hallucinations.
- Limited understanding of complex projects.
- Potential for false positives, flagging issues that aren't actually problematic.



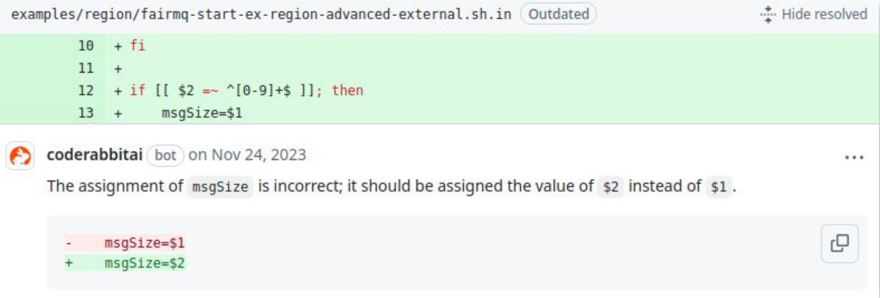
examples/region/fairmq-start-ex-region-advanced.sh.in

```
+ xterm -geometry 90x40+550+40 -hold -e @EX_BIN_DIR@/$PROCESSOR1 &
+
+ PROCESSOR2="fairmq-ex-region-processor"
+ PROCESSOR2+=" --id processor1"
```

coderabbitai bot on Nov 15, 2023

There seems to be a copy-paste error. The `PROCESSOR2` should have a unique identifier, but it is currently set to `processor1`, which is the same as `PROCESSOR1`. This should be corrected to ensure that each processor has a unique ID.

```
- PROCESSOR2+=" --id processor1"
+ PROCESSOR2+=" --id processor2"
```



examples/region/fairmq-start-ex-region-advanced-external.sh.in

```
+ fi
+
+ if [[ $2 =~ ^[0-9]+$ ]]; then
+     msgSize=$1
```

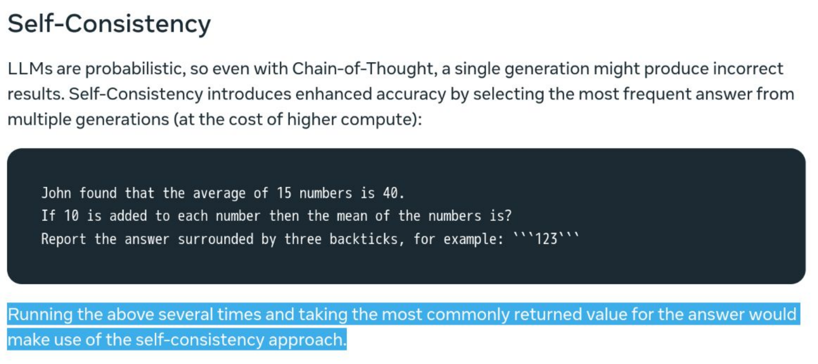
coderabbitai bot on Nov 24, 2023

The assignment of `msgSize` is incorrect; it should be assigned the value of `$2` instead of `$1`.

```
-     msgSize=$1
+     msgSize=$2
```

Pearbot

Quality Improvements



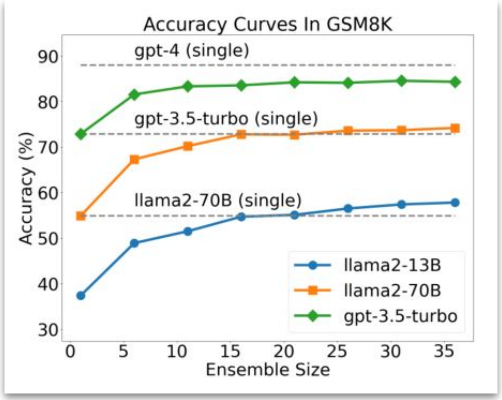
Self-Consistency

LLMs are probabilistic, so even with Chain-of-Thought, a single generation might produce incorrect results. Self-Consistency introduces enhanced accuracy by selecting the most frequent answer from multiple generations (at the cost of higher compute):

```
John found that the average of 15 numbers is 40.
If 10 is added to each number then the mean of the numbers is?
Report the answer surrounded by three backticks, for example: ```123```
```

Running the above several times and taking the most commonly returned value for the answer would make use of the self-consistency approach.

https://www.llama.com/docs/how-to-guides/prompting/



Li, Junyou, et al. "More agents is all you need." arXiv preprint arXiv:2402.05120 (2024). https://doi.org/10.48550/arXiv.2402.05120

1. **Multi-Agent Initial Reviews:**
   - Multiple AI models generate initial code reviews.
2. **Reflection[1][2] by a “decider” Agent:**
   - A separate, potentially more advanced model analyzes the initial reviews.
   - This agent synthesizes and refines the feedback from multiple sources, rejects potentially less impactful comments.
   - It prioritizes the most important issues and suggestions.
   - The reflection step helps in producing a more comprehensive and coherent final review.
3. **Prompt improvements:**
   - Specific & useful code review examples.
   - Examples include Chain-of-Thought[3] type of reviews, that include some reasoning why the suggestions would be good.

[1] Madaan, Aman, et al. "Self-refine: Iterative refinement with self-feedback." Advances in Neural Information Processing Systems 36 (2024). https://doi.org/10.48550/arXiv.2303.17651

[2] Shinn, Noah, et al. "Reflexion: Language agents with verbal reinforcement learning." Advances in Neural Information Processing Systems 36 (2024). https://doi.org/10.48550/arXiv.2303.11366

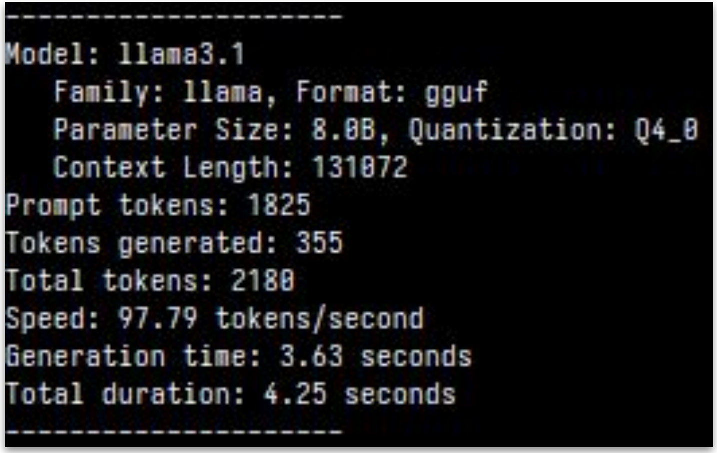
[3] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022). https://doi.org/10.48550/arXiv.2201.11903

Pearbot

Backend

ollama[3] (via python lib and HTTP request): open-source large language model server, written in Go, backed by **llama.cpp**[4] (C++):

- Efficient serving of large language models
- CPU/GPU/CPU+GPU hybrid inference to partially accelerate models larger than the total VRAM capacity
- Supports many model architectures: llama, gemma2, qwen2, ...
- Support for multitude of model quantization techniques and precisions for faster inference and reduced memory use
- Usage Metrics



```
--------------------
Model: llama3.1
   Family: llama, Format: gguf
   Parameter Size: 8.0B, Quantization: Q4_0
   Context Length: 131072
Prompt tokens: 1825
Tokens generated: 355
Total tokens: 2180
Speed: 97.79 tokens/second
Generation time: 3.63 seconds
Total duration: 4.25 seconds
--------------------
```

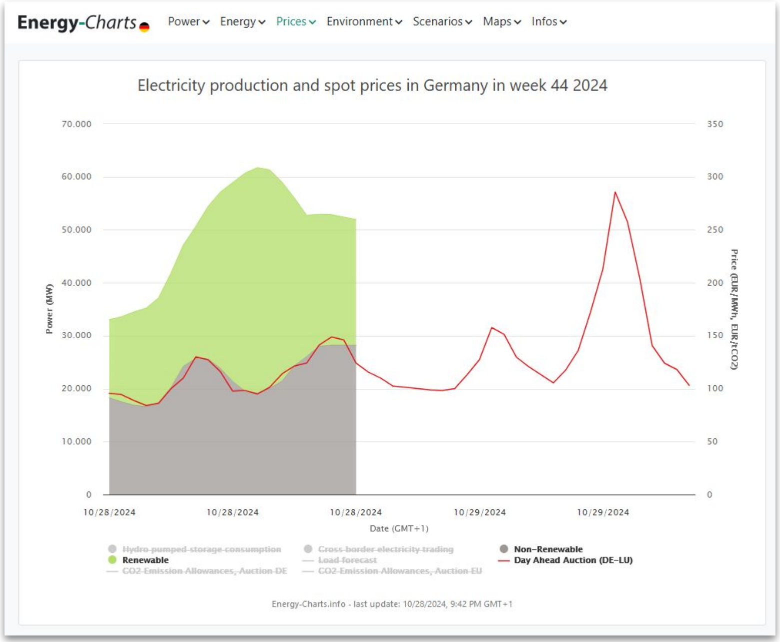
Cluster Job Scheduling with Reinforcement Learning (RL)

Idea

Schedule jobs when electricity prices are lower and/or electricity is “greener”.

The RL approach could be beneficial if:

- There are complex job dependencies/patterns.
- Electricity prices are volatile.
- Need to optimize for many variables simultaneously.
- Lots of historical data.



https://energy-charts.info/charts/price_spot_market/chart.htm?l=en&c=DE

Cluster Job Scheduling with Reinforcement Learning

Concepts

Agent:

- Scheduling advisor

Environment:

- Cluster

Observations:

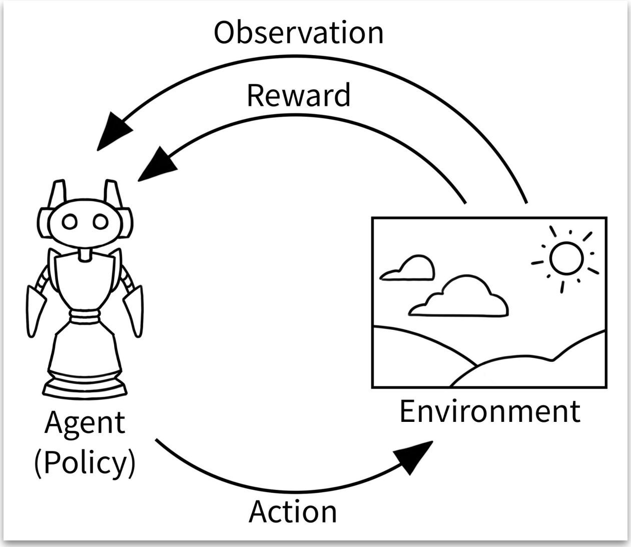
- Nodes & their states
- Job queue
- Predicted prices

Actions:

- Turn nodes on / off.

Reward:

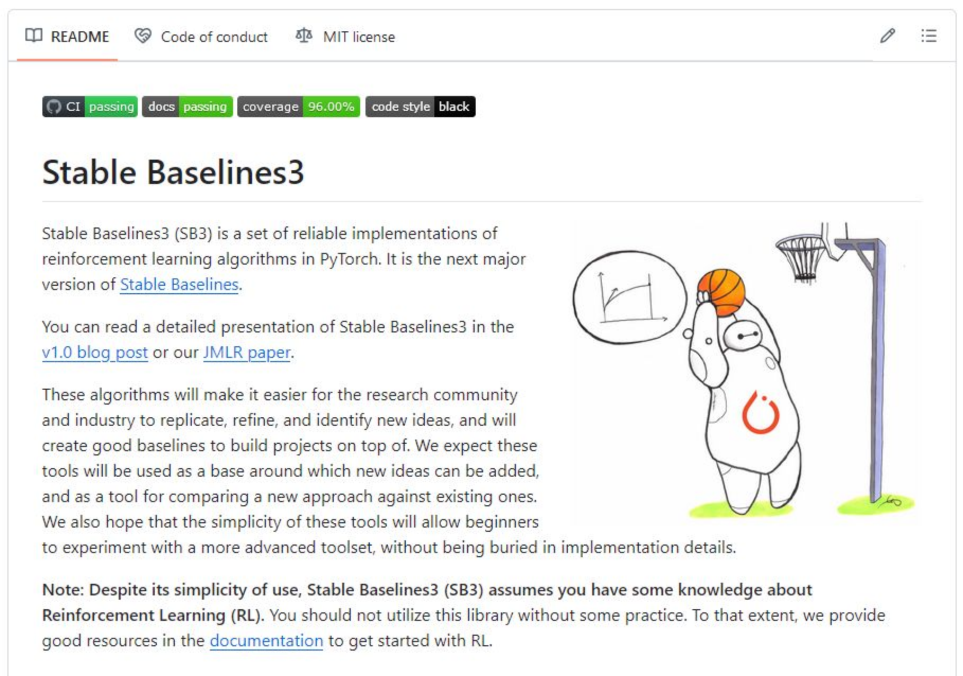
- Processing jobs during favorable times.



https://gymnasium.farama.org/introduction/basic_usage/

Cluster Job Scheduling with Reinforcement Learning

Tools



Stable Baselines3

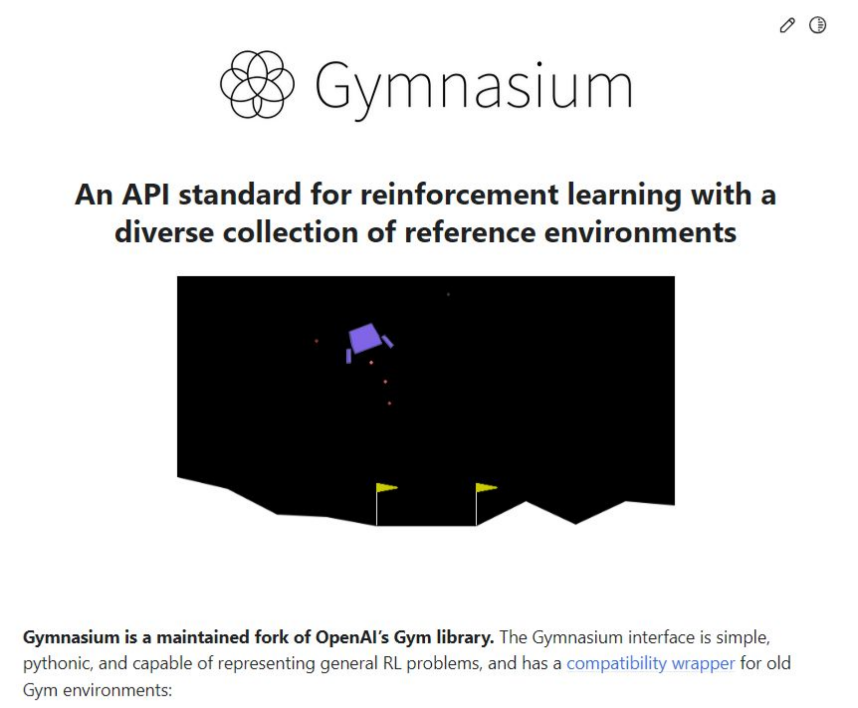
Stable Baselines3 (SB3) is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It is the next major version of Stable Baselines.

You can read a detailed presentation of Stable Baselines3 in the v1.0 blog post or our JMLR paper.

These algorithms will make it easier for the research community and industry to replicate, refine, and identify new ideas, and will create good baselines to build projects on top of. We expect these tools will be used as a base around which new ideas can be added, and as a tool for comparing a new approach against existing ones. We also hope that the simplicity of these tools will allow beginners to experiment with a more advanced toolset, without being buried in implementation details.

Note: Despite its simplicity of use, Stable Baselines3 (SB3) assumes you have some knowledge about Reinforcement Learning (RL). You should not utilize this library without some practice. To that extent, we provide good resources in the documentation to get started with RL.

https://github.com/DLR-RM/stable-baselines3



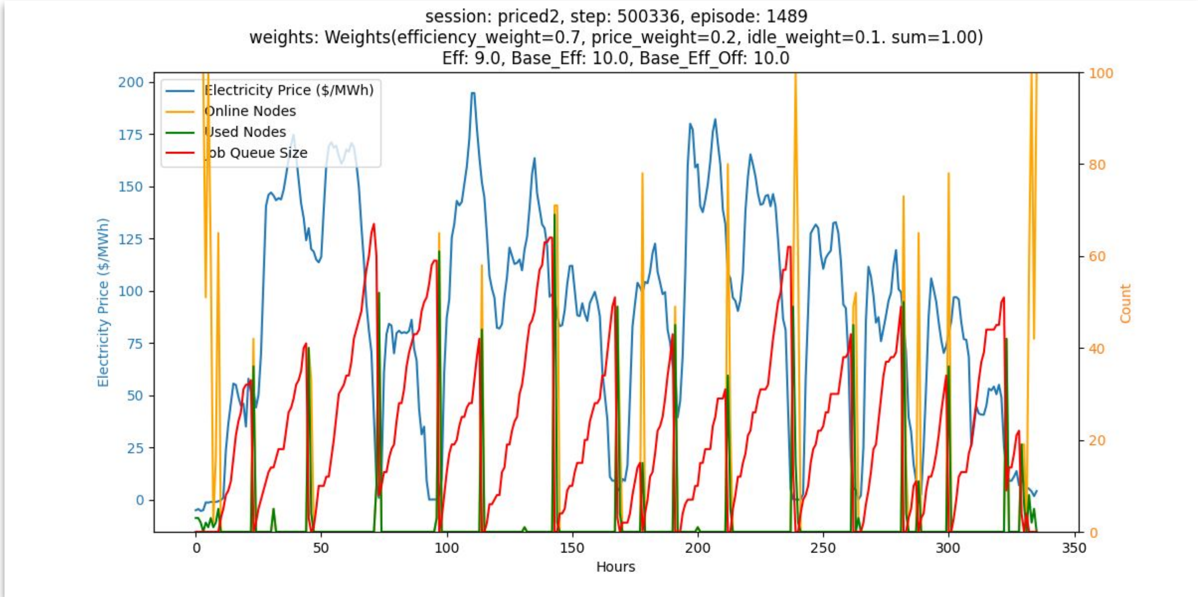
An API standard for reinforcement learning with a diverse collection of reference environments

Gymnasium is a maintained fork of OpenAI's Gym library. The Gymnasium interface is simple, pythonic, and capable of representing general RL problems, and has a compatibility wrapper for old Gym environments:

https://gymnasium.farama.org/

Cluster Job Scheduling with Reinforcement Learning

Some early results WORK IN PROGRESS



session: priced2, step: 500336, episode: 1489
weights: Weights(efficiency_weight=0.7, price_weight=0.2, idle_weight=0.1. sum=1.00)
Eff: 9.0, Base_Eff: 10.0, Base_Eff_Off: 10.0