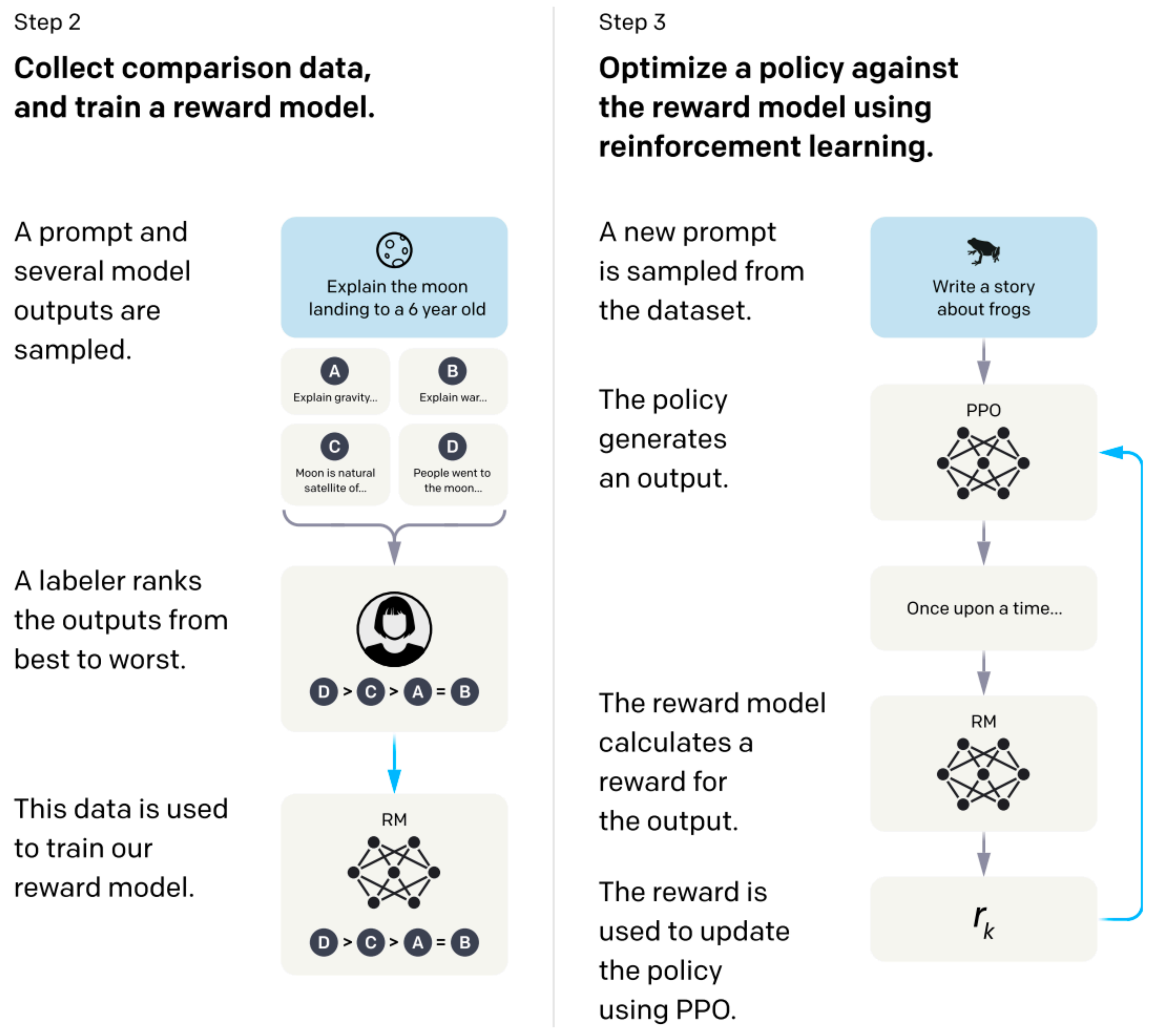
verl: an flexible and efficient RL framework for LLMs

Haibin Lin, Bytedance MLSys & LMSYS.org

Reinforcement learning

- Supervised fine-tuning
  - Learning from labeled examples
- Reinforcement learning
  - Optimization based on rewards
  - Preference alignment
    - Human feedbacks
  - Reasoning with automated feedbacks
    - Coding: unit-tests
    - Math: ground truth graders
  - Agentic tasks: operator, deep research



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

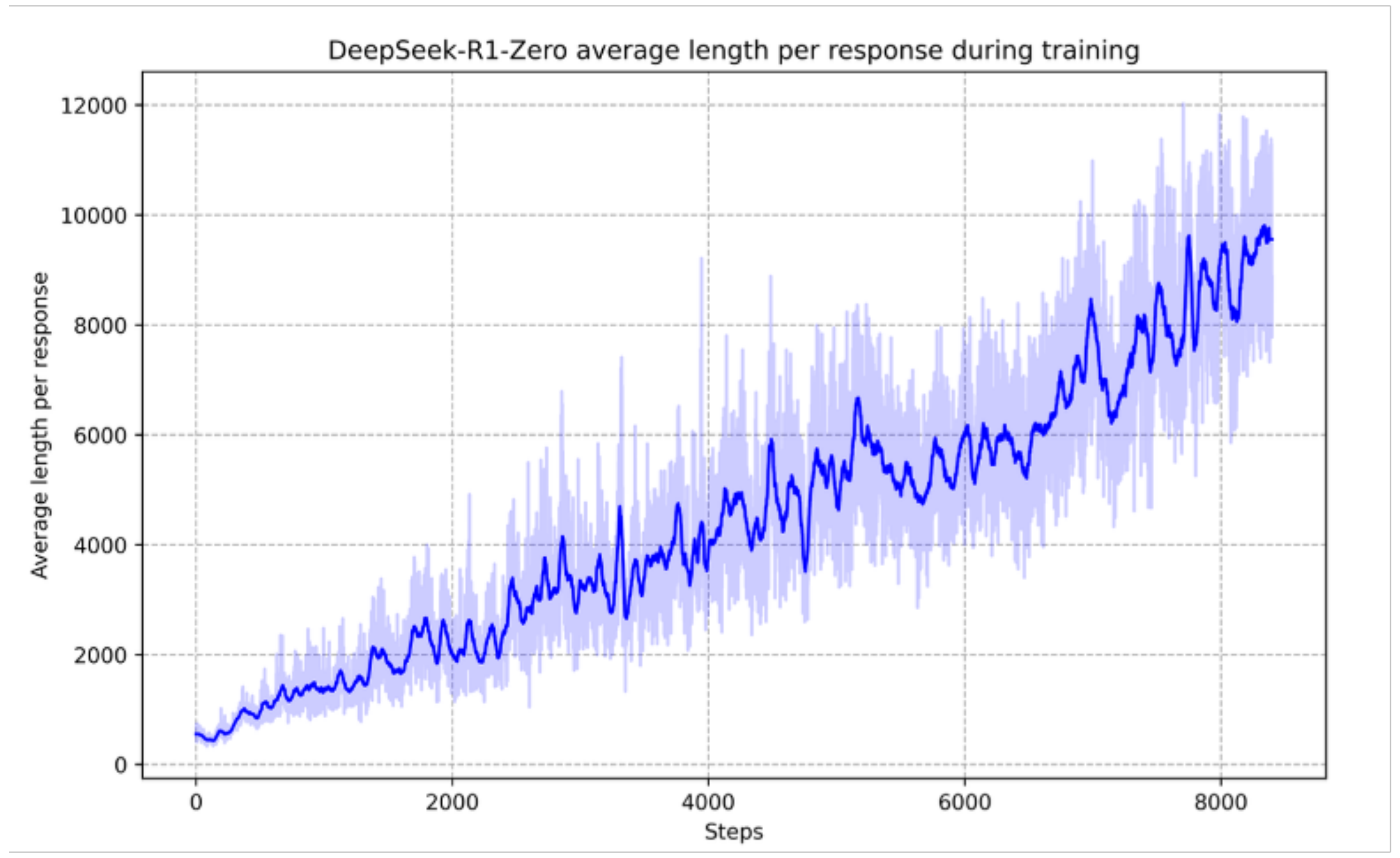
The reward model calculates a reward for the output.

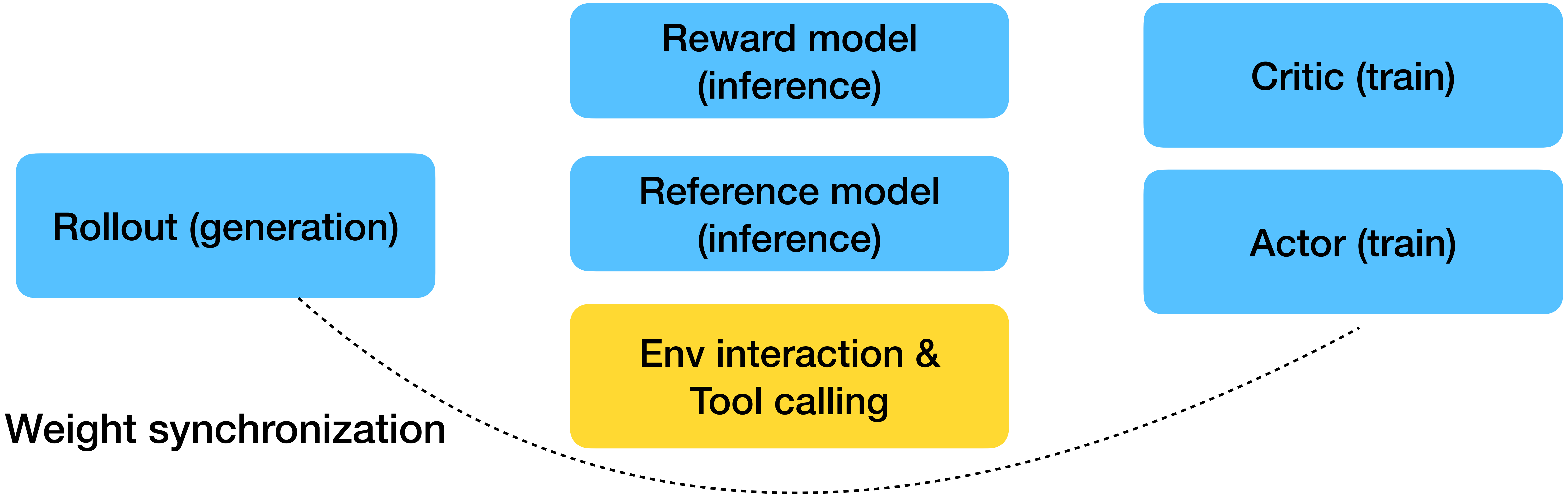
The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

Infra challenges for RL on LLMs

- the need for nD parallelisms (Megatron-LM)
  - Growing model size: llama 70b, Deepseek 671B
  - Growing sequence length: 8k -> 1M
- the need for programming abstractions





Reward model (inference)

Critic (train)

Rollout (generation)

Reference model (inference)

Actor (train)

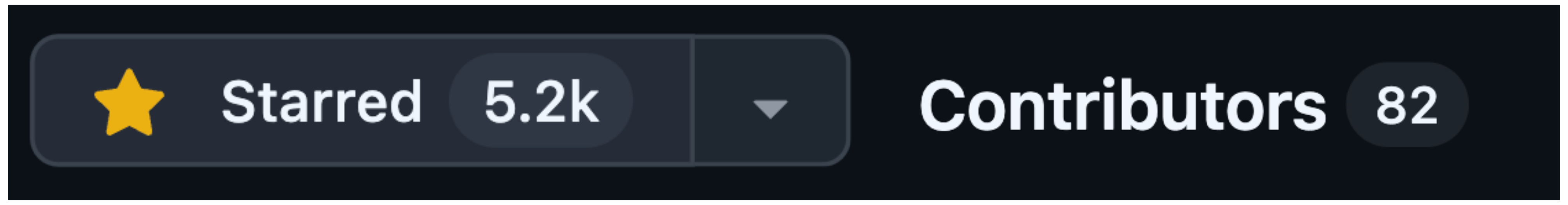
Env interaction & Tool calling

Weight synchronization

verl: open framework for RL on LLMs

History & community

- Developed & adopted internally since 2023/9 for research*, open sourced on 2024/10
- Reinforcement learning at Bytedance
  - Reasoning: O1-level performance on math benchmarks
  - RLHF, Image generation, music generation
  - Desktop operator, coding assistant…
- Users and contributors:
  - PKU, THU, UIUC, UCB, UCLA, HKU, Stanford, Northwestern, MIT…
  - Amazon, NVIDIA, LMSys, Alibaba, StepFun, Anyscale, OpenHands, …



*HybridFlow: A Flexible and Efficient RLHF Framework (Eurosys 25')

verl: open framework for RL on LLMs

Features…

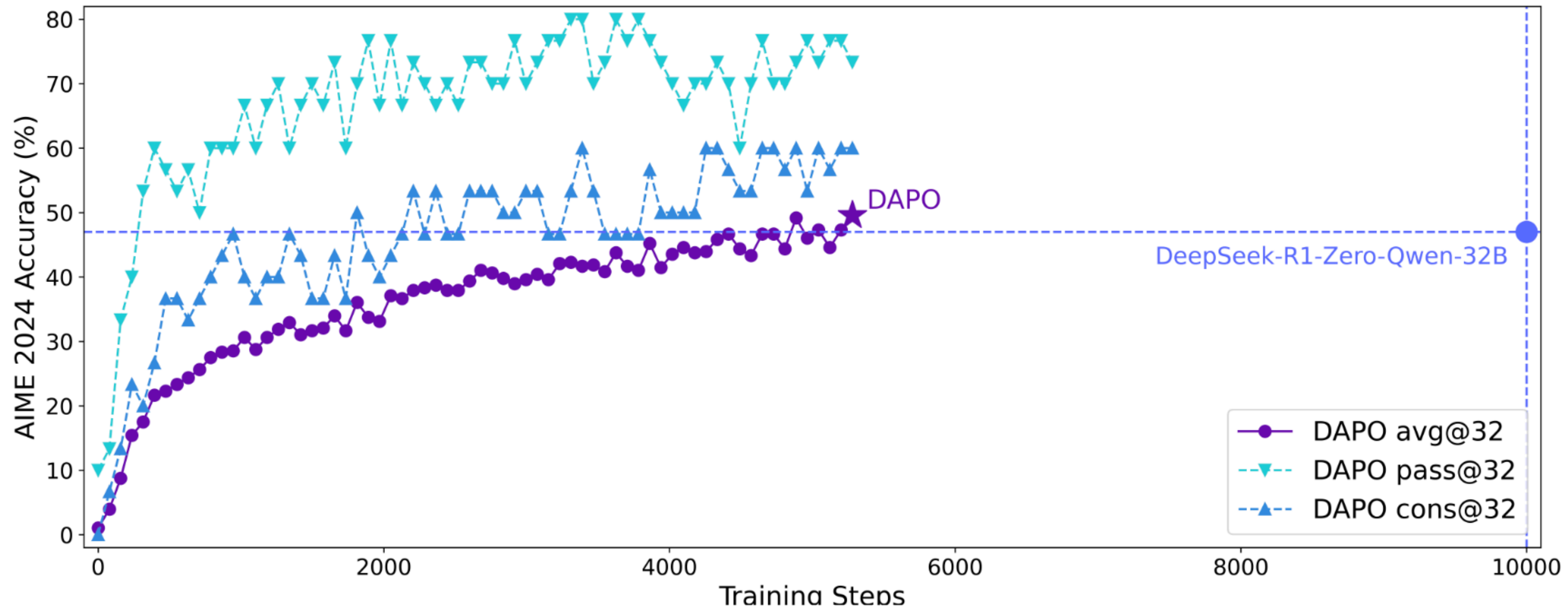
- RL recipes: PPO, GRPO, RLOO, reinforce++, DAPO
- Transformers integration: deepseek, llama, qwen, gemma, etc
- Inference engine: vllm, sglang**
- Distributed training engine: FSDP, Megatron
- System optimizations: seq packing, seq parallelism, fused entropy kernels
- Hardware support: NVIDIA GPU, AMD GPU*, Huawei NPU**
- Hybrid controller programming

** experimental

verl for LLM RL: why is it special?

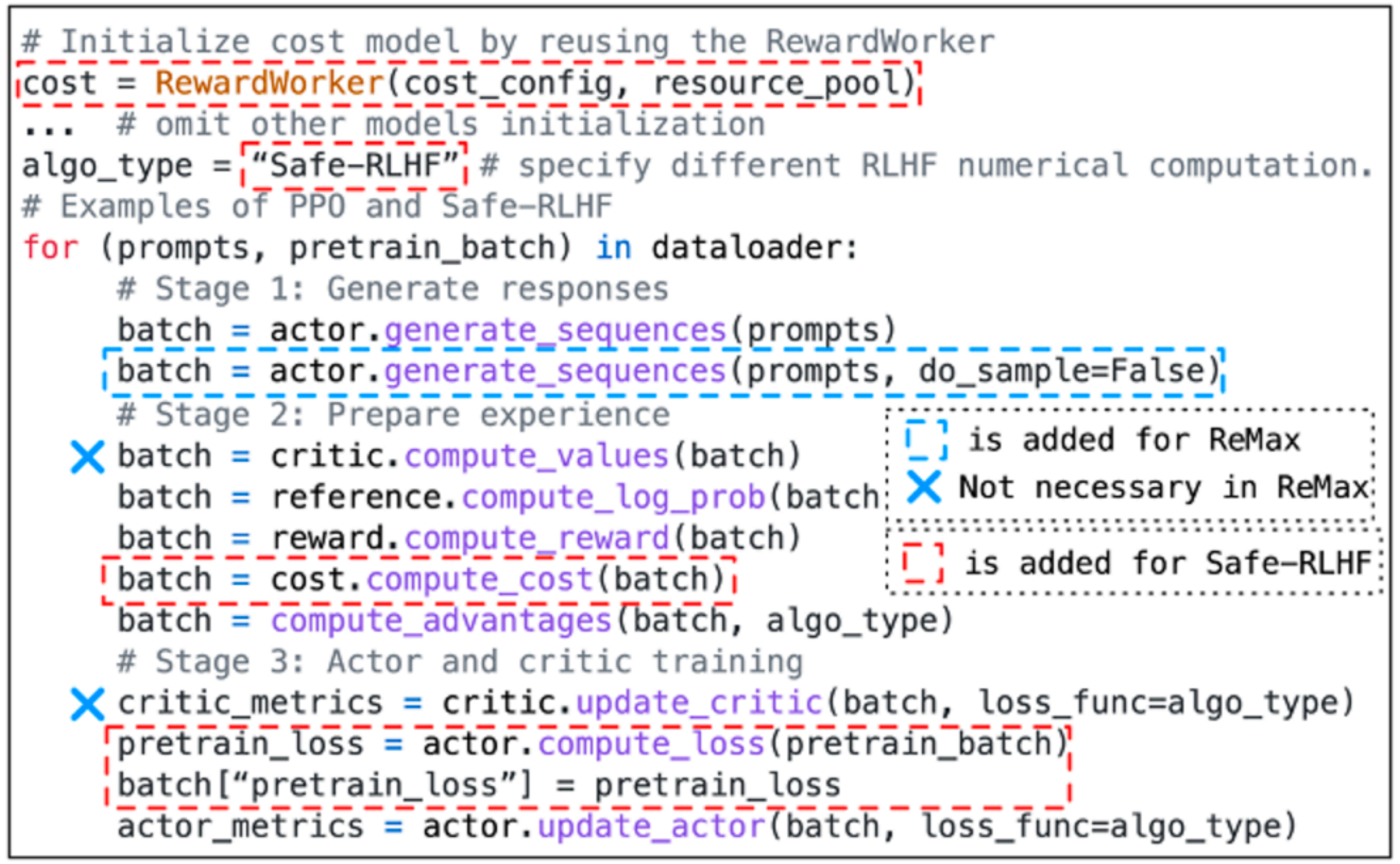
flexible, efficient, battle-tested

- **DAPO algorithm**: improvements on top of GRPO
- beats DeepSeek-R1-zero-32B with fewer steps
- Fully open source recipe (dataset, code, logs, model)



AIME 2024 Performance

- **hybrid controller programming model***
- program distributed RL algorithm like a single threaded program



```
# Initialize cost model by reusing the RewardWorker
cost = RewardWorker(cost_config, resource_pool)
...  # omit other models initialization
algo_type = "Safe-RLHF" # specify different RLHF numerical computation.
# Examples of PPO and Safe-RLHF
for (prompts, pretrain_batch) in dataloader:
    # Stage 1: Generate responses
    batch = actor.generate_sequences(prompts)
    batch = actor.generate_sequences(prompts, do_sample=False)
    # Stage 2: Prepare experience
  ✕ batch = critic.compute_values(batch)
    batch = reference.compute_log_prob(batch)
    batch = reward.compute_reward(batch)
    batch = cost.compute_cost(batch)
    batch = compute_advantages(batch, algo_type)
    # Stage 3: Actor and critic training
  ✕ critic_metrics = critic.update_critic(batch, loss_func=algo_type)
    pretrain_loss = actor.compute_loss(pretrain_batch)
    batch["pretrain_loss"] = pretrain_loss
    actor_metrics = actor.update_actor(batch, loss_func=algo_type)
```

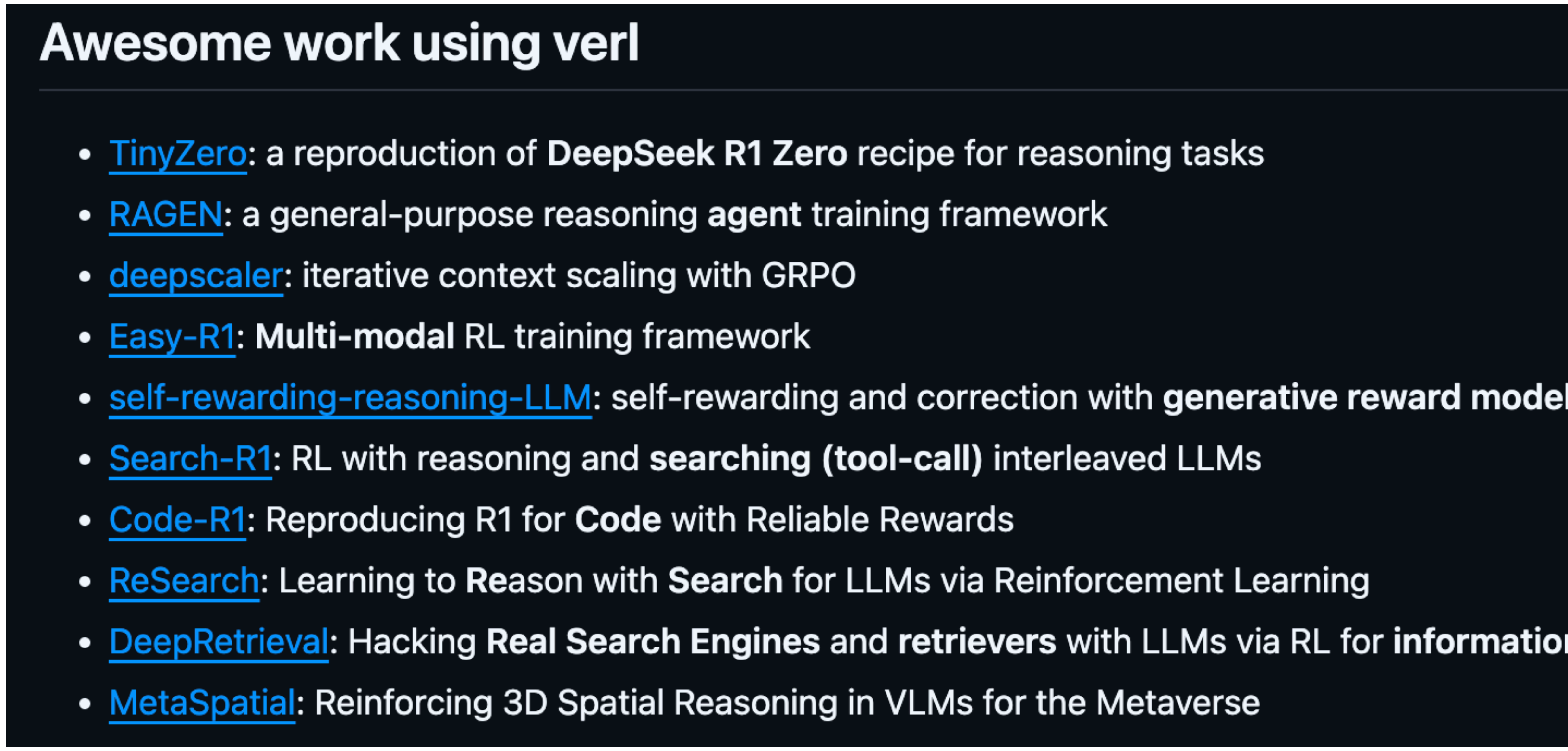
⬚ is added for ReMax
✕ Not necessary in ReMax
⬚ is added for Safe-RLHF

*DAPO: an Open-Source LLM Reinforcement Learning System at Scale

*HybridFlow: A Flexible and Efficient RLHF Framework (Eurosys 25')

verl Roadmap

- megatron v0.11 for deepseek-v3 MOE
- multi-turn optimizations
- agentic environment & tool calling
- stable sglang integration
- stable hardware support: AMD & NPU
- contributions are welcome!!



Awesome work using verl

- TinyZero: a reproduction of **DeepSeek R1 Zero** recipe for reasoning tasks
- RAGEN: a general-purpose reasoning **agent** training framework
- deepscaler: iterative context scaling with GRPO
- Easy-R1: **Multi-modal** RL training framework
- self-rewarding-reasoning-LLM: self-rewarding and correction with **generative reward mode**
- Search-R1: RL with reasoning and **searching (tool-call)** interleaved LLMs
- Code-R1: Reproducing R1 for **Code** with Reliable Rewards
- ReSearch: Learning to **Reason** with **Search** for LLMs via Reinforcement Learning
- DeepRetrieval: Hacking **Real Search Engines** and **retrievers** with LLMs via RL for **informatio**
- MetaSpatial: Reinforcing 3D Spatial Reasoning in VLMs for the Metaverse

Let’s build together!