DaisyRose-27B explained: long-horizon agent RL on 1,200 curated tasks and a faster trainer

25 November 2026 · 8 min read · blogs.iictp.org/daisyrose-27b-agentic-rl-post-training-efficiency-explained

Short answer: DaisyRose-27B is an open-weight model released by RadixArk on 18 November 2026, built on Qwen3.8-27B and trained with reinforcement learning on long-horizon agent tasks. RadixArk reports raising Terminal-Bench 4.1 pass@1 from 5.6% to 12.0% using 1,200 curated tasks, 18 days and 144 H200 GPUs, and says its training backend is 57% cheaper than the open-source trainer. The checkpoints are under Apache 2.0.

TL;DR The short version

- DaisyRose-27B is RadixArk's first model, a Qwen3.8-27B trained with reinforcement learning on about 1,200 curated long-horizon agent tasks.
- RadixArk reports Terminal-Bench pass@1 rising from 5.6% to 12.0%, and says its internal training backend reached 37% FLOPs utilisation against 16% for the open-source trainer.
- The figures are the vendor's own and 12% is still low, but curated tasks aimed at known mistakes and an efficient training stack both cut cost.

Read online, listen to it, or change the text size: https://blogs.iictp.org/daisyrose-27b-agentic-rl-post-training-efficiency-explained

What is DaisyRose-27B?

RadixArk's first model. It keeps Qwen3.8-27B's architecture, tokenizer, chat template, image understanding and 256K-token context window, according to the post. It thinks before answering by default, returning reasoning separately from the final answer. RadixArk recommends serving it with SGLang and DSpark speculative decoding.

How it works

- **Curated tasks.** Built with Eigent AI from thousands of candidates; about 1,200 were kept. Each targets a mistake the base model makes, needs several interdependent steps, and must reward actually solving the problem. Each attempt gets up to 1.5 hours in a sandbox, with more than 100 tool calls and more than 256K output tokens.
- **Asynchronous RL.** RadixArk's open-source framework Miles runs in fully asynchronous mode: 128 H200s run agent inference while 16 run the trainer.
- **Long sessions.** Sessions reach up to 1M tokens, using harness summarisation to span trajectories.
- **Sandboxes.** Every attempt runs in a fresh environment copy where the agent works and is graded.

What is new compared with earlier approaches

The post argues that task design and training-system efficiency, not just model size, limit agent post-training. It reports higher GPU utilisation on its internal backend than on the open-source Megatron-based trainer.

Results reported by the source

- Terminal-Bench 4.1 pass@1 rises from 5.6% to 12.0%; the post says it also gains on reasoning, instruction-following and coding benchmarks.
- Model FLOPs utilisation of 37% on the internal backend against 16% for the open-source trainer, which RadixArk says means 57% less GPU time.
- About 16,800 tokens per second across 144 GPUs against about 8,300 for the open-source trainer.
- The release run used 63,000 GPU-hours over weeks and held steady speed.
- The scores use the higher of BF16 and FP8 runs, which differ by less than a point where both were evaluated, per the post. All figures are RadixArk's own.

Limitations and open questions

- A 12.0% pass rate is still low in absolute terms.
- The efficiency numbers compare an internal backend with an open-source one; the internal backend is not public.
- The post links to a model card, which holds the comparison sources and settings, and I did not verify them independently. Sources were read on 25 November 2026. Products, rules and standards change, so check the linked originals before you rely on any detail.

How to try or evaluate it

1. Download the checkpoint (Apache 2.0) and serve it with SGLang using the cookbook configuration for your GPU.
2. Run your own agent tasks and compare against the base Qwen3.8-27B.
3. Measure pass rate, tokens used and wall-clock time per task.
4. Review the licence terms of the base model and the dataset before commercial use.

What it means for engineers working with open models

Two lessons stand out: curated tasks that target a model's known mistakes can matter more than volume, and utilisation of the training stack changes cost materially. For teams fine-tuning agents, see harness adapters and RL with LoRA.

Frequently asked questions

- **What is DaisyRose-27B?** An open-weight 27B model from RadixArk, built on Qwen3.8-27B and trained with RL on agent tasks.
- **What improvement does it report?** Terminal-Bench 4.1 pass@1 from 5.6% to 12.0%.

- **What is its licence?** Apache 2.0 for the checkpoints, per the post.
- **How was it trained?** About 1,200 curated tasks, 18 days, 144 H200 GPUs with the Miles framework.

Related reading

- Harness adapters: RL and LoRA for agent harnesses
- Agentic evaluation: how to build and maintain one
- Local LLM Practitioner certification

> Related credential: Local LLM Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. RadixArk. DaisyRose-27B: Pushing the Frontier of Post-Training Efficiency. Hugging Face, 18 November 2026. huggingface.co/blog/RadixArk/daisyrose-27b-pushing-the-frontier-of-post-trainin

Published by IICTP, an independent technical credential body, on 25 November 2026. The current version of this post is at https://blogs.iictp.org/daisyrose-27b-agentic-rl-post-training-efficiency-explained. More guides: blogs.iictp.org.