Artifact Evaluation Guidelines for NSDI'26 Paper #677

Testbed Environment and Experiments Description

We provide you with scripts and configurations to reproduce key results in our paper, including:

- End-to-end evaluation results in Figure 8(a), Figure 9(a) and Figure 10(Qwen2.5-7B version).
- Sensitive analysis in Figure 11.
- Micro-benchmarks in Figure 13(b with Qwen2.5-7B), Figure 13(c) and Table 3.

Note 1: The evaluation results shown in the paper were conducted on NVIDIA H800 clusters with 16-128 GPUs. Providing such infrastructure is extremely expensive, so we only provide you with limited resources to train a **Qwen2.5-7B** model with 8K response length, which is the minimal model size exhibited in the paper. Micro-benchmarks in Figure 12, as well as the large-scale analysis in Figure 14, are not provided, since they require larger model sizes and more resources. Reviewers with access to large-scale resources may replicate these experiments.

Note 2: The minimal scale of the evaluation results in our paper is performed with 16 H800 GPUs, while we can only provide limited GPUs; the absolute performance will not be an exact match, but they provide approximately the same trends.

Note 3: For detailed information of the resources provided, please refer to our repo.

Environment Setup

Option #1: Using our provided Docker image.

We stronly recommend using our provided docker images, thus no additional environment setup is required. For detailed information please refer to the provided repo.

You will find the source code located in `/root/code/RollPacker` of the Docker image.

Option #2: Build your own environment based on ROLL.

`RollPacker` is build on ROLL. So you can follow ROLL's set up instructions to build your own environment for the **RLVR Pipeline**.

Get Start

- Install necessary python packages and wait for several minutes to complete.

```
bash install.sh
```

Run Experiments

To visualize your results, please refer to our provided repo.

- End-to-end Evaluation (Figure 8(a), 9(a), 10).

```
bash examples/e2e_performance/run_pipe.sh
```

By default, this pipeline runs for 40 iterations for illustration, you may run more steps to reproduce the reported calidation score in Figure 8(a).

- Sensitive Analysis (Figure 11).

```
bash examples/sensitive_analysis_fig11/run_8k.sh
```

This example will only invoke the rollout stage without actually train the model, since the sensitive analysis in Figure 11 only consider rollout time with different *P* and *R*

- Micro-benchmarks of Reward Scheduler (Figure 13(b-c)).

Figure 13 (b) **without** pipeline execution:

```
bash examples/benchmark_fig13/run_7B_nopipeline.sh
```

Figure 13 (b) **with** pipeline execution:

```
bash examples/benchmark_fig13/run_7B_pipeline.sh
```

Figure 13 (c) **without** adaptive timeout:

```bash
bash examples/benchmark_fig13/run_7B_adpt_notimeout.sh
```

Figure 13 (c) **with** adaptive timeout:

```bash
bash examples/benchmark_fig13/run_7B_adpt_timeout.sh
```

- Stream Trainer (Table 3).

```bash
bash examples/stream_trainer_table3/run_stream_trainer.sh
```

This example provides a fixed setting of `infer_scaling_down_progress_ratio=0.40` and `scaling_down_train_batch_size=64` **w/o tail batching**, you may compare the performance with the **Long Rounds** in end-to-end performance. The iteration repeats for 5 times by default.