Serving DeepSeek-R1 on 2x8 H100 SLURM nodes with SGLang

1. Set up the environment (adjust for your cuda version):

```
conda create -n sglang124 python=3.11
conda activate sglang124

pip install torch==2.5.1 --index-url https://download.pytorch.org/whl/cu124

pip install sgl-kernel --force-reinstall --no-deps
pip install "sglang[all]>=0.4.2.post4" --find-links https://flashinfer.ai/whl/cu124/torch2.5/flashinfer/
```

2. Run the server and wait for the model to load:

```
sbatch slurm/serve_r1.slurm -m "/fsx/deepseek-r1-checkpoint" -e "sglang124"
```

3. Run the data generation script:

```
python scripts/generate_reasoning.py \
    --dataset-name "AI-MO/NuminaMath-1.5" \
    --output-file "numinamath_r1_generations.jsonl" \
    --prompt-column "problem" \
    --uuid-column "problem" \
    --api-addr "<SGLANG_SERVER_ADDRESS>:39877" \
    --num-generations 2 \
    --max-tokens 16384 \
    --max-concurrent 200
```

————————————————————————————————————

```
#!/bin/bash
#SBATCH --ntasks-per-node=1
#SBATCH --gres=gpu:8
#SBATCH --partition=hopper-prod
#SBATCH --output=./logs/%x-%j.out
#SBATCH --err=./logs/%x-%j.err
#SBATCH --requeue
```

```
# Specific configuration optimized for the Hugging Face Compute
Cluster
# Be ye warned this may not work on other clusters!
module load cuda/12.4

set -x -e

source ~/.bashrc
source openr1/bin/activate

TASK_NAME=$1
TASKS=$2
MODEL_ID=$3
MODEL_REVISION=$4
# Optional args
[ -z "$5"] && TENSOR_PARALLEL=False || TENSOR_PARALLEL=$5
[ -z "$6"] && TRUST_REMOTE_CODE=False ||
TRUST_REMOTE_CODE=$6
# $7 is reserved for system_prompt, see line 51
NUM_GPUS=$(nvidia-smi -L | wc -l)

# Set Whether to use tensor parallelism or data parallelism
if [ "$TENSOR_PARALLEL" = "True" ]; then
    # use TP to shard model across NUM_GPUS
    export VLLM_WORKER_MULTIPROC_METHOD=spawn
    # FIXME: lighteval now requires us to manually pass the generation
params

MODEL_ARGS="pretrained=$MODEL_ID,revision=$MODEL_REVISIO
N,trust_remote_code=$TRUST_REMOTE_CODE,dtype=bfloat16,tenso
r_parallel_size=$NUM_GPUS,max_model_length=32768,max_num_b
atched_tokens=32768,gpu_memory_utilization=0.8,generation_para
meters={max_new_tokens:32768,temperature:0.6,top_p:0.95}"
else

MODEL_ARGS="pretrained=$MODEL_ID,revision=$MODEL_REVISIO
N,trust_remote_code=$TRUST_REMOTE_CODE,dtype=bfloat16,data
_parallel_size=$NUM_GPUS,max_model_length=32768,max_num_b
atched_tokens=32768,gpu_memory_utilization=0.8,generation_para
meters={max_new_tokens:32768,temperature:0.6,top_p:0.95}"
```

```
fi

LM_EVAL_REPO_ID="open-r1/open-r1-eval-leaderboard"
MODEL_NAME=$(echo $MODEL_ID | sed 's/\//_/g') # replaces / with _
DETAILS_REPO_ID="open-r1/details-$MODEL_NAME"
OUTPUT_DIR="eval_results/$MODEL_ID/$MODEL_REVISION/
$TASK_NAME"
# We need this flag since we run this script from training jobs that use
DeepSpeed and the env vars get progated which causes errors
during evaluation
ACCELERATE_USE_DEEPSPEED=false
# Enable fast downloads
HF_HUB_ENABLE_HF_TRANSFER=1

echo "Running lighteval script ..."
echo "Eval results will be saved to $OUTPUT_DIR"
lighteval vllm "$MODEL_ARGS" $TASKS \
    --use-chat-template \
    --output-dir $OUTPUT_DIR \
    --save-details \
    ${7:+--system-prompt "$(echo "$7" | base64 --decode)"}

OUTPUT_FILEPATHS=$(find $OUTPUT_DIR/results/ -type f \( -name
"*.json" \))
for filepath in $OUTPUT_FILEPATHS; do
    echo "Uploading $filepath to Hugging Face Hub..."
    filename=$(basename -- "$filepath")
    for attempt in {1..20}; do
        if huggingface-cli upload --repo-type space --private
$LM_EVAL_REPO_ID $filepath $OUTPUT_DIR/$filename; then
            echo "Upload succeeded for $filepath"
            break
        else
            echo "Upload failed for $filepath. Attempt $attempt of 20.
Retrying in 5 seconds..."
            sleep 5
        fi
    done
done

echo "Uploading details to Hugging Face Hub..."
DETAILS_FILEPATHS=$(find $OUTPUT_DIR/details/ -type f \( -name
```

```
"*.parquet" \))
echo "DETAILS_FILEPATHS: $DETAILS_FILEPATHS"
TIMESTAMP=$(date +"%Y-%m-%dT%H-%M-%S")
python scripts/upload_details.py --data_files $DETAILS_FILEPATHS
--hub_repo_id $DETAILS_REPO_ID --config_name
$MODEL_REVISION.$TASK_NAME.$TIMESTAMP

echo "Cleaning up ..."
rm -rf $OUTPUT_DIR

echo "Done!"
```