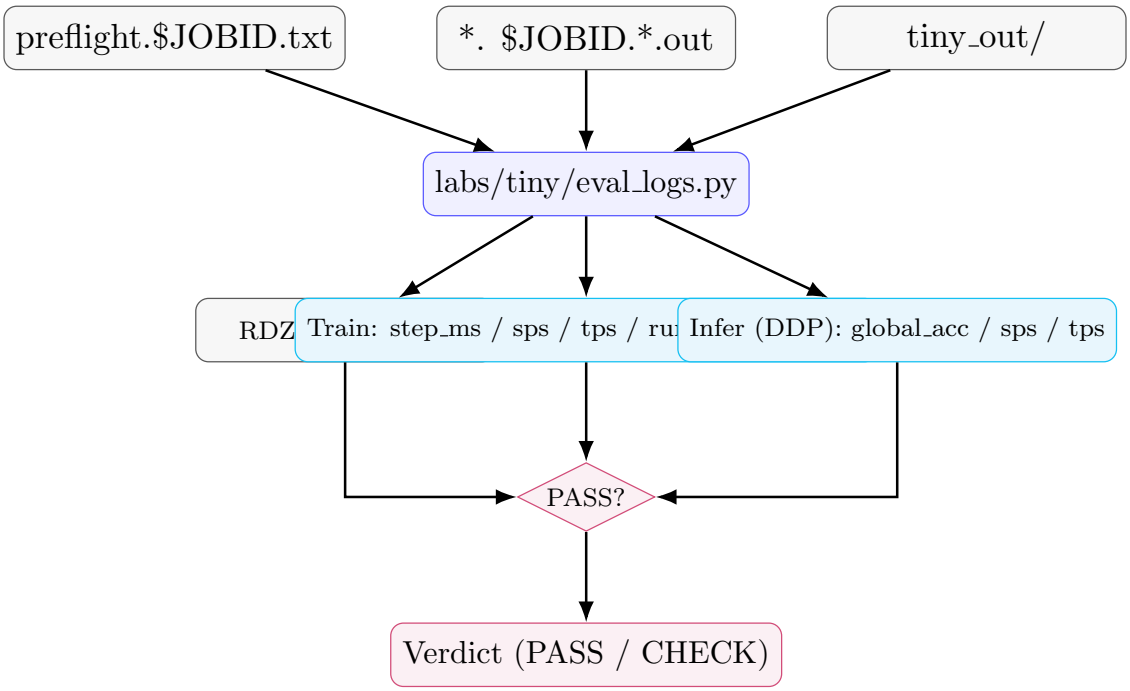
Mini Lab: Evaluate DDP Training & Distributed Inference from SLURM Logs

Ahmed Métwalli — September 3, 2025

Goal. Using only SLURM logs, verify rendezvous & rank coverage, read *training science metrics* (step_ms, samples/s, tokens/s, runtime, eval accuracy), and read *distributed inference* metrics (global accuracy & throughput). Our tools:

- `/launch_tiny.sh` (`ACTION={train,infer}` + `EXTRA_ARGS`)
- `labs/tiny/train_tiny.py` (evaluation runs on *all* ranks)
- `labs/tiny/infer_ddp.py` (rank 0 prints global metrics)
- `labs/tiny/eval_logs.py` (parses everything)



1. Run & Evaluate (3 commands)

Train (multi-node, with signals). Keep `--exclude=hpc42` handy.

```
SALLOC_OPTS="--exclude=hpc42" ACTION=train \
  EXTRA_ARGS="--subset 2000 --epochs 1 --batch 8" \
  bash ~/launch_tiny.sh
```

Distributed inference (reads checkpoint `tiny_out/`).

```
SALLOC_OPTS="--exclude=hpc42" ACTION=infer \
  EXTRA_ARGS="--ckpt tiny_out --batch 64 --max_test 2048" \
  bash ~/launch_tiny.sh
```

Evaluate from logs (works for train or infer jobs).

```
python3.11 ~/mrmito/project/labs/tiny/eval_logs.py --job $SLURM_JOB_ID
# or reuse a past JOBID, e.g.:
python3.11 ~/mrmito/project/labs/tiny/eval_logs.py --job 7765
```

2. Quick Checklist (what "good" looks like)

- **RDZV:** one unique tuple (`nnodes,nproc_per_node,endpoint`).
- **Ranks:** min=0, max=`WORLD_SIZE-1`, no missing ranks.
- **Training science:** P50/P95 `step_ms`; P50/P95 samples/s, tokens/s; `TRAIN_RUNTIME_SEC`; best/last `eval_accuracy`.
- **Inference:** rank 0 prints `INFER global_accuracy`, `global_samples_per_sec`, `global_tokens_per_sec`.
- **Preflight:** all nodes `proj=ok pkgs=ok` or staged path noted.

Speedup & efficiency (optional):

$$S(n) = T(1)T(n), \qquad E(n) = S(n)n \quad (usesteady - stateepochtimes).$$

3. Tips & Common Gotchas

- **Eval must run on *all* ranks.** In `train_tiny.py` we call `trainer.evaluate()` on every rank and only *print* on rank 0. If you guard eval with rank 0, you'll see *exit-barrier / gloo connection closed* errors after training.
- **Node issues.** Use `SALLOC_OPTS="--exclude=hpc42"` (add more if needed). If RDZV mismatches appear, restart with a fresh job.
- **CPU friendliness.** We set `OMP_NUM_THREADS=2`, disable tokenizers parallelism, and use `gloo`. Keep batches modest on CPU.
- **Reading logs fast.**

```
# RDZV + ranks
grep -hE "torchrun:|^\[RANK " ~/slurm_logs/*.$SLURM_JOB_ID.*.out | sort -u | tail

# Training signals + eval
grep -h "step_ms=" ~/slurm_logs/*.$SLURM_JOB_ID.*.out | head
grep -h "TRAIN_RUNTIME_SEC=" ~/slurm_logs/*.$SLURM_JOB_ID.*.out
grep -h "eval_accuracy" ~/slurm_logs/*.$SLURM_JOB_ID.*.out

# Inference metrics
grep -h "INFER global_accuracy" ~/slurm_logs/*.$SLURM_JOB_ID.*.out
```

4. What to Submit (short)

- First ∼50 lines of `eval_logs.py` for your job.
- RDZV line + highest `[RANK r]` proving full rank coverage.

- P50/P95 step_ms, samples/s, tokens/s, train runtime, and best/last eval accuracy.
- Inference: global accuracy and throughput.
- (Optional) A 3–5 line note on bottlenecks (comms vs. tokenization/Python).

Appendix. Minimal evaluator patterns

```
RANK_RE = r"\[RANK\s+(\d+)\]\s+WORLD_SIZE=(\d+)"
RDZV_RE = r"torchrun:\s+nnodes=(\d+)\s+nproc_per_node=(\d+)\s+node_rank=(\d+)\s+rdzv=([^\
    s]+)"
STEP_RE = r"\[rank\s+(\d+)\s+\|\s+step\s+(\d+)\].*?step_ms=([\d.]+)\s+samples_per_sec=([\
    d.]+)\s+tokens_per_sec=([\d.]+)"
RT_RE = r"TRAIN_RUNTIME_SEC=([\d.]+)"
EVAL_RE = r"(?:eval_accuracy|EVAL accuracy)=\s*([0-9]*\.?[0-9]+)"
INFER_RE= r"\[RANK 0\]\s+INFER.*global_accuracy=\s*([0-9]*\.?[0-9]+).*?
    global_samples_per_sec=([\d.]+).*?global_tokens_per_sec=([\d.]+)"
```

Cheat sheet:

```
SALLOC_OPTS="--exclude=hpc42" ACTION=train EXTRA_ARGS="--subset 2000 --epochs 1 --batch 8" bash  /launch_tiny.sh
SALLOC_OPTS="--exclude=hpc42" ACTION=infer EXTRA_ARGS="--ckpt tiny_out --batch 64 --max_test 2048" bash
/launch_tiny.sh
python3.11  /mrmito/project/labs/tiny/eval_logs.py --job $SLURM_JOB_ID
```