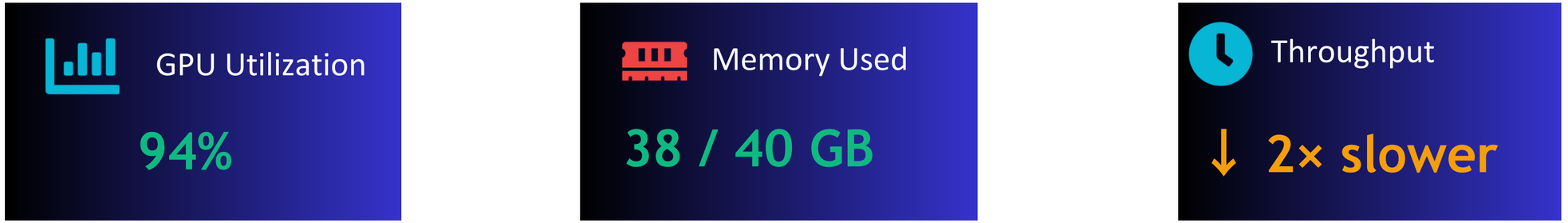
Step-Aligned Telemetry for PyTorch Distributed Training

Time & Memory Attribution Across Ranks

Abhinav Srivastav | TraceOpt

Pytorch Conference 2026, Paris | April 8, 2026

Healthy dashboard but slower training



- System-level metrics hide training-level failures.
- System metrics are sampled by time whereas training progresses by steps

Why the dashboard can look healthy

Time-sampled telemetry

- Samples every few seconds
- Blends multiple training phases together
- Averages across step boundaries
- Hides rank-level skew

Step-aligned telemetry

- One completed step at a time
- Phase attribution
- Compares worst rank vs median rank
- Preserves step jitter and drift

Training happens step by step. In distributed runs, overall speed is limited by the slowest rank.

Failure modes standard metrics miss

Rank stragglers

One slow rank delays the whole job.

Dataloader Stalls

GPUs wait between steps for input.

Step-Time Jitter

Latency varies step to step.
Even though average looks stable.

Memory Creep

Memory rises gradually across steps.
Late failure or throughput drops.

Step-aligned, rank-aware aggregation (TraceML)

Completed-step aggregation

Aggregate only when the full step is complete across ranks.

Worst-rank vs median-rank

Compare the slowest rank to the median for each step.

Semantic attribution

Tie time and peak memory back to training phases

This is the view used in the next demos.

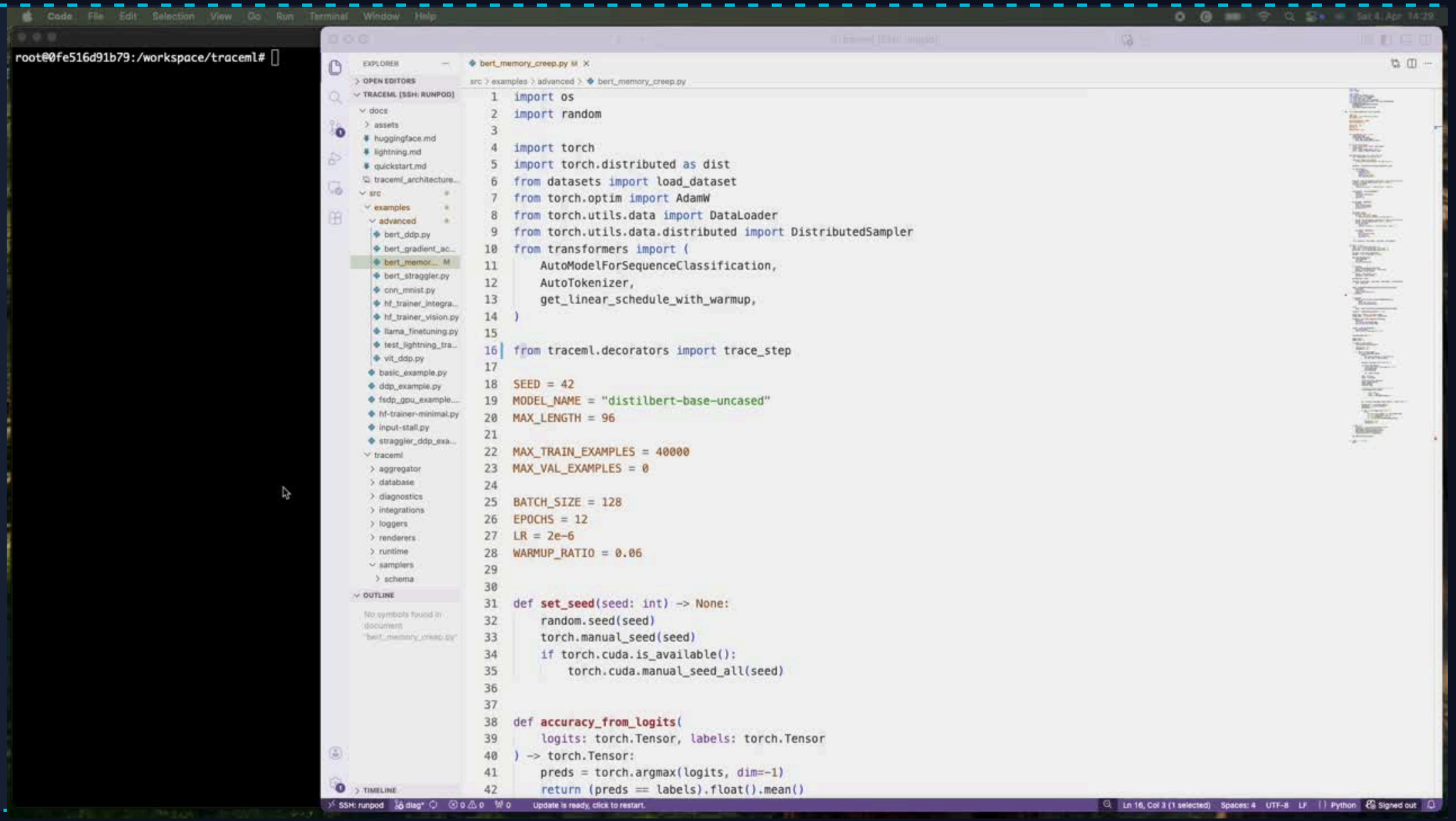
Demo 1: Catching a memory creep

Workload: DistilBERT fine-tuning on AG News on a single A40

Injected issue: Retain loss, logits, and hidden_states every step

Expected behaviour: Peak memory rises gradually across steps

Recorded demo — 150 seconds


root@0fe516d91b79:/workspace/traceml#
bert_memory_creep.py
src > examples > advanced > bert_memory_creep.py
import os
import random
import torch
import torch.distributed as dist
from datasets import load_dataset
from torch.optim import AdamW
from torch.utils.data import DataLoader
from torch.utils.data.distributed import DistributedSampler
from transformers import (
AutoModelForSequenceClassification,
AutoTokenizer,
get_linear_schedule_with_warmup,
)
from traceml.decorators import trace_step
SEED = 42
MODEL_NAME = "distilbert-base-uncased"
MAX_LENGTH = 96
MAX_TRAIN_EXAMPLES = 40000
MAX_VAL_EXAMPLES = 0
BATCH_SIZE = 128
EPOCHS = 12
LR = 2e-6
WARMUP_RATIO = 0.06
def set_seed(seed: int) -> None:
random.seed(seed)
torch.manual_seed(seed)
if torch.cuda.is_available():
torch.cuda.manual_seed_all(seed)
def accuracy_from_logits(
logits: torch.Tensor, labels: torch.Tensor
) -> torch.Tensor:
preds = torch.argmax(logits, dim=-1)
return (preds == labels).float().mean()

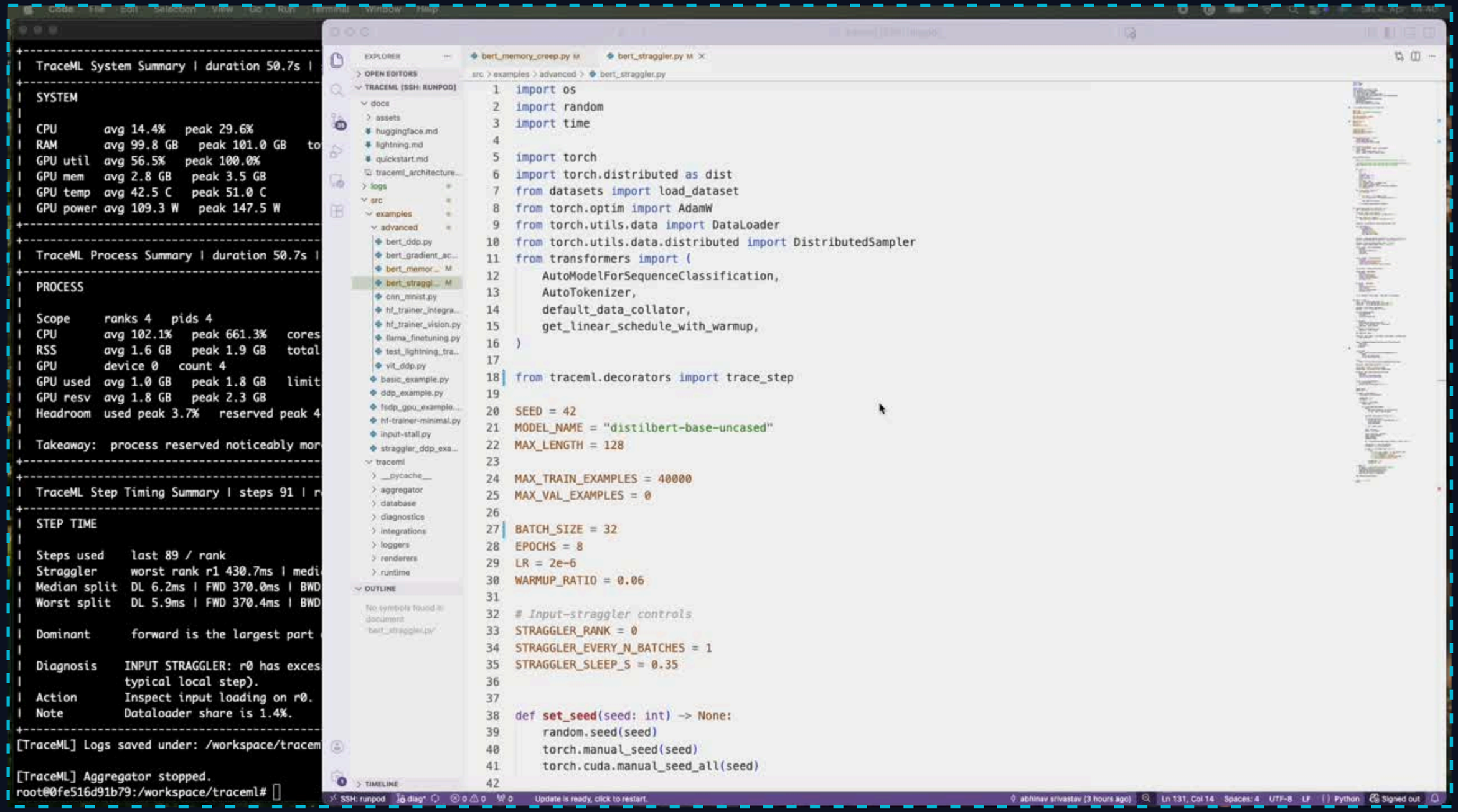
Demo 2: Spotting a DDP rank straggler

Workload: DistilBERT fine-tuning on AG News on 4 × A40, single-node DDP

Injected issue: One rank has slower data loading due to time.sleep

Expected behaviour: Worst-rank vs median-rank latency diverges immediately

Recorded demo — 90 seconds


TraceML System Summary | duration 50.7s |
SYSTEM
CPU avg 14.4% peak 29.6%
RAM avg 99.8 GB peak 101.0 GB
GPU util avg 56.5% peak 100.0%
GPU mem avg 2.8 GB peak 3.5 GB
GPU temp avg 42.5 C peak 51.0 C
GPU power avg 109.3 W peak 147.5 W
TraceML Process Summary | duration 50.7s |
PROCESS
Scope ranks 4 pids 4
CPU avg 102.1% peak 661.3%
RSS avg 1.6 GB peak 1.9 GB
GPU device 0 count 4
GPU used avg 1.0 GB peak 1.8 GB
GPU resv avg 1.8 GB peak 2.3 GB
Headroom used peak 3.7% reserved peak 4
Takeaway: process reserved noticeably mor
TraceML Step Timing Summary | steps 91 |
STEP TIME
Steps used last 89 / rank
Straggler worst rank r1 430.7ms | medi
Median split DL 6.2ms | FWD 370.0ms | BWD
Worst split DL 5.9ms | FWD 370.4ms | BWD
Dominant forward is the largest part
Diagnosis INPUT STRAGGLER: r0 has exces
typical local step).
Action Inspect input loading on r0.
Note Dataloader share is 1.4%.
[TraceML] Logs saved under: /workspace/tracem
[TraceML] Aggregator stopped.
root@0fe516d91b79:/workspace/traceml#
EXPLORER
OPEN EDITORS
TRACEML [SSH: RUNPOD]
docs
assets
huggingface.md
lightning.md
quickstart.md
traceml_architecture...
logs
src
examples
advanced
bert_ddp.py
bert_gradient_ac...
bert_memor... M
bert_straggl... M
cnn_mnist.py
hf_trainer_integra...
hf_trainer_vision.py
llama_finetuning.py
test_lightning_tra...
vit_ddp.py
basic_example.py
ddp_example.py
fsdp_gpu_example...
hf-trainer-minimal.py
input-stall.py
straggler_ddp_exa...
traceml
__pycache__
aggregator
database
diagnostics
integrations
loggers
renderers
runtime
OUTLINE
No symbols found in document 'bert_straggler.py'
TIMELINE
bert_memory_creep.py M
bert_straggler.py M
src > examples > advanced > bert_straggler.py
1 import os
2 import random
3 import time
4
5 import torch
6 import torch.distributed as dist
7 from datasets import load_dataset
8 from torch.optim import AdamW
9 from torch.utils.data import DataLoader
10 from torch.utils.data.distributed import DistributedSampler
11 from transformers import (
12 AutoModelForSequenceClassification,
13 AutoTokenizer,
14 default_data_collator,
15 get_linear_schedule_with_warmup,
16)
17
18 from traceml.decorators import trace_step
19
20 SEED = 42
21 MODEL_NAME = "distilbert-base-uncased"
22 MAX_LENGTH = 128
23
24 MAX_TRAIN_EXAMPLES = 40000
25 MAX_VAL_EXAMPLES = 0
26
27 BATCH_SIZE = 32
28 EPOCHS = 8
29 LR = 2e-6
30 WARMUP_RATIO = 0.06
31
32 # Input-straggler controls
33 STRAGGLER_RANK = 0
34 STRAGGLER_EVERY_N_BATCHES = 1
35 STRAGGLER_SLEEP_S = 0.35
36
37
38 def set_seed(seed: int) -> None:
39 random.seed(seed)
40 torch.manual_seed(seed)
41 torch.cuda.manual_seed_all(seed)
42
SSH: runpod
Update is ready, click to restart.
abhinav srivastav (3 hours ago)
Ln 131, Col 14
Spaces: 4
UTF-8
LF
Python
Signed out

Minimal Integration

Works with your existing stack. No rewrites.

PyTorch

```
from traceml.decorators  import trace_step

for batch in dataloader:
    with trace_step(model):
        # your training loop
        ...
```

Hugging Face Trainer

```
from traceml.hf_decorators import TraceMLTrainer

trainer = TraceMLTrainer(
    model=model,
    args=training_args,
    traceml_enabled=True
)
```

PyTorch Lightning

```
from traceml.utils.lightning import TraceMLCallback
trainer = L.Trainer(callbacks=[TraceMLCallback()])
```

Try TraceML on your next training run

```
pip install traceml-ai
```

github.com/traceopt-ai/traceml

SCAN ME

Abhinav Srivastav | abhinav@traceopt.ai

Apache 2.0 | Feedback welcome