Clase 172 — Entrenamiento multi-dispositivo, tf.distribute

Parte: 2 — Deep Learning · Fuente: Géron, cap. 19 § Training Models Across Multiple Devices. Duración estimada: 75 min.

Clase 172 — Entrenamiento multi-dispositivo, tf.distribute

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 19 § Training Models Across Multiple Devices. Duración estimada: 75 min.*

Objetivo

Escalar el training a múltiples GPUs (y multi-nodos). Conocer las 3 estrategias TF: MirroredStrategy (1 nodo, varias GPUs), MultiWorkerMirroredStrategy (varios nodos), TPUStrategy. Conocer equivalentes PyTorch (DDP, FSDP) que son estándar para LLMs grandes.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Aplicar MirroredStrategy en un único nodo con N GPUs.
- Entender data parallelism: cada GPU procesa su mini-batch, gradients se promedian (all-reduce).
- Diferenciar de model parallelism (modelo dividido entre GPUs) y pipeline parallelism.
- Conocer FSDP (Fully Sharded Data Parallel) para modelos demasiado grandes para 1 GPU (LLMs).
- Saber que PyTorch Lightning abstrae todo esto cambiando un kwarg.

Temas

- Data parallelism: misma red replicada, distinto batch por GPU.
- Model parallelism: red dividida (tensor parallel, pipeline parallel).
- FSDP / DeepSpeed ZeRO: shard de parámetros, gradientes, optimizer states.
- TF: MirroredStrategy, MultiWorkerMirroredStrategy, TPUStrategy.
- PyTorch: DistributedDataParallel, FullyShardedDataParallel, DeepSpeed.
- Lightning / Accelerate: abstracciones de alto nivel.

Definiciones y características

- Data Parallelism: cada device procesa un slice del batch.
- All-reduce: operación colectiva que promedia gradients entre devices.
- MirroredStrategy: data parallelism single-node.
- MultiWorkerMirroredStrategy: data parallelism multi-node.
- FSDP: shards de parámetros entre devices — para modelos demasiado grandes.
- Gradient accumulation: simular batch grande acumulando gradients en GPUs chicas.

Dataset / recursos

- Acceso a ≥ 2 GPUs (Colab Pro+, cloud).
- Librerías: tensorflow, opcional torch.distributed.

Ejercicios

1. MirroredStrategy: strategy = tf.distribute.MirroredStrategy(). with strategy.scope(): model = build_model(); model.compile(...). Entrenar.
2. Batch effective: si tenés 4 GPUs y batch_size=32, el batch global es 128. Adjust LR (LR scales linearly with batch).
3. Gradient accumulation: simular batch=512 acumulando 4 mini-batches de 128. Manual con custom training loop.
4. PyTorch DDP: comando torchrun --nproc_per_node=4 train.py con DistributedDataParallel(model).
5. PyTorch Lightning: trainer = L.Trainer(strategy='ddp', devices=4). Listo.

Homework verificable

Si tenés ≥ 2 GPUs:

1. Mismo modelo en single-GPU vs MirroredStrategy(2 GPUs).
2. Reportar wall-time por epoch.
3. Verificar que la accuracy final es similar.

Criterio de aceptación: speedup wall-clock ≈ 1.5-1.9× con 2 GPUs (no es 2× porque hay overhead de comm); accuracy similar.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
MirroredStrategy no acelera	Bottleneck en data loading. Fix: optimizar
LR no ajustado con batch global mayor	Convergencia subóptima. Fix: lr_new = lr_o
OOM en multi-GPU con LLM grande	Data parallelism replica el modelo. Si el
All-reduce muy lento	Network bandwidth limitado en multi-nodo.
Mismatch en BatchNorm stats entre replicas	Sync BN: tf.keras.layers.experimental.Sync

Preguntas frecuentes

¿Multi-GPU cuándo necesario?

Dataset masivo (días en 1 GPU) o modelo grande (no entra en 1 GPU). Para experimentos chicos, 1 GPU.

¿FSDP o DeepSpeed?

Ambas implementan shard de pesos. FSDP es PyTorch nativo. DeepSpeed (Microsoft) tiene más features (ZeRO-Offload, CPU offloading).

¿Train LLM 70B en 1 nodo?

8× H100 (640 GB VRAM total) con FSDP es factible para Llama 70B. Cluster necesario para más grande.

¿Cloud para multi-GPU?

AWS p4d (8× A100), GCP A3 (8× H100). Caro ($30-60/hora).

¿MultiWorkerMirroredStrategy real-world?

Sí, con TF_CONFIG env var y orquestación (K8s, Slurm, Vertex AI). Vertex AI hace esto transparente.

Referencias

- Géron, cap. 19 — Training Models Across Multiple Devices.
- tf.distribute guide.
- PyTorch DDP, FSDP.
- DeepSpeed.
- Rajbhandari et al. (2020), ZeRO: Memory Optimizations Toward Training Trillion Parameter Models.

Siguiente clase

Clase 173 — JAX y Flax: el stack moderno de Google para DL

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb