The Alan Turing Institute

MPI Part II: Practical MPI

Baskerville Training 2025

29-30 January 2025 – David Llewellyn-Jones

Message Passing Interface

1. MPI: Message Passing Interface
2. Provides a standard for *distributed memory parallelisation*
3. Gavin already explained the theory and practice
4. Now let’s apply it to some machine learning

Motivation

GPUs are great for accelerated training and inference, but processing is bound by:

1. Available memory
2. Speed of computation

How to make AI better?

1. Improved algorithms
2. Larger models
3. More training data

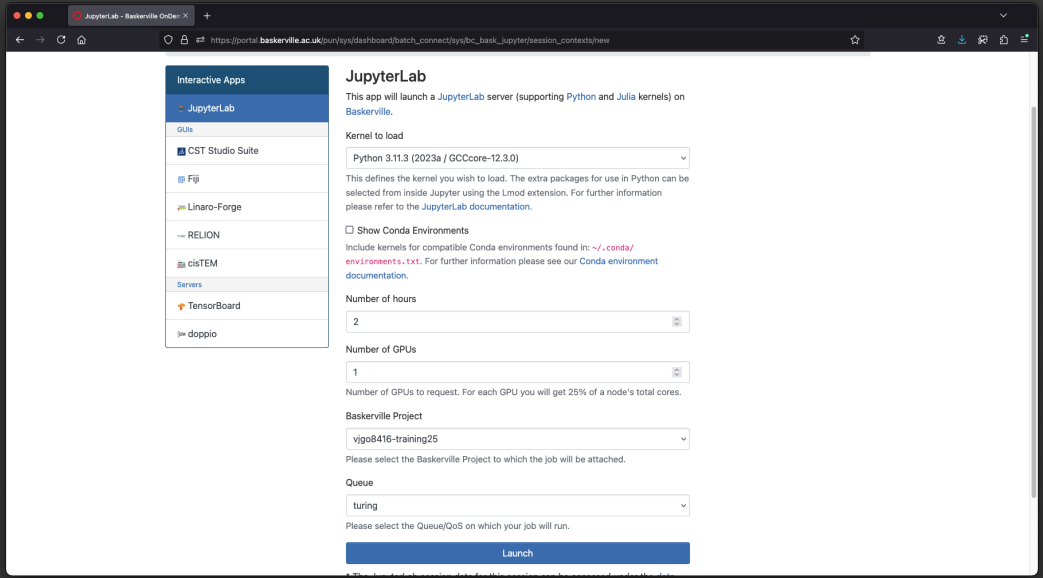
The last two require *more compute*

Scaling up

1. Strong motivation for scaling across GPUs
2. Single device: 4 or 8 GPUs maximum
3. Eventually want to scale across nodes
4. Get this right... the sky’s the limit

Preparation

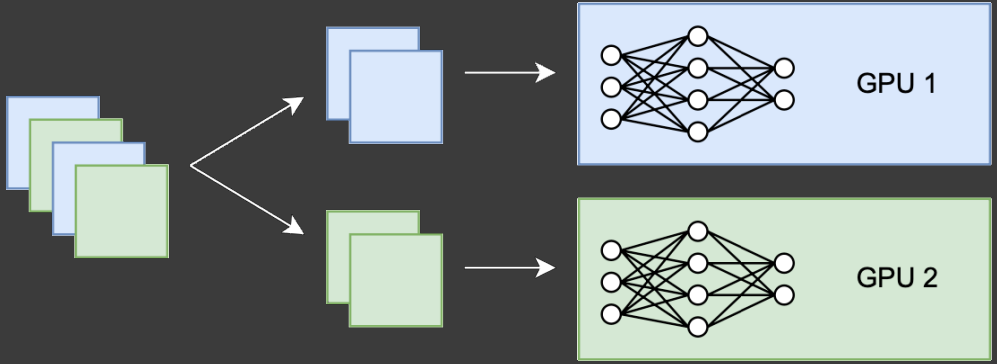
1. Open `https://docs.baskerville.ac.uk/`
2. Select Baserville Portal
3. Login if necessary
4. Select JupyterLab from the Interactive Apps list
5. Configure a 2 hour session with 1 GPU on the project `vjgo8416-training25` and the `turing` queue
6. Launch


JupyterLab - Baskerville OnDem
https://portal.baskerville.ac.uk/pun/sys/dashboard/batch_connect/sys/bc_bask_jupyter/session_contexts/new
Interactive Apps
JupyterLab
GUIs
CST Studio Suite
Fiji
Linaro-Forge
RELION
cisTEM
Servers
TensorBoard
doppio
JupyterLab
This app will launch a JupyterLab server (supporting Python and Julia kernels) on Baskerville.
Kernel to load
Python 3.11.3 (2023a / GCCcore-12.3.0)
This defines the kernel you wish to load. The extra packages for use in Python can be selected from inside Jupyter using the Lmod extension. For further information please refer to the JupyterLab documentation.
Show Conda Environments
Include kernels for compatible Conda environments found in: ~/.conda/environments.txt. For further information please see our Conda environment documentation.
Number of hours
2
Number of GPUs
1
Number of GPUs to request. For each GPU you will get 25% of a node's total cores.
Baskerville Project
vjgo8416-training25
Please select the Baskerville Project to which the job will be attached.
Queue
turing
Please select the Queue/QoS on which your job will run.
Launch

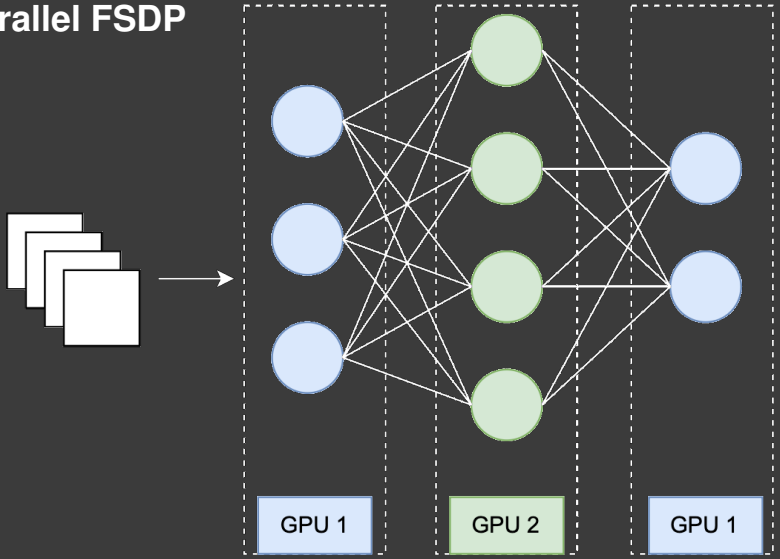
Types of model parallelism

1. DDP: Distributed Data Parallel
2. FSDP: Fully Sharded Data Parallel
3. DeepSpeed ZeRO: Zero Redundancy Optimiser

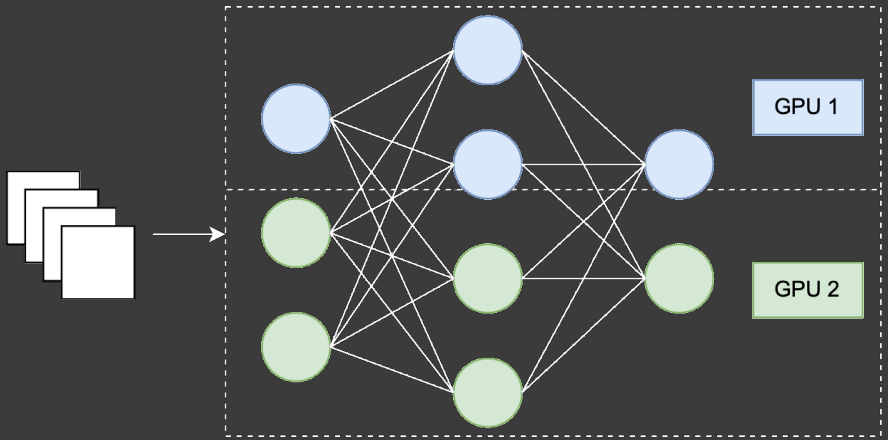
Data parallel



Model parallel FSDP



Model parallel DeepSpeed

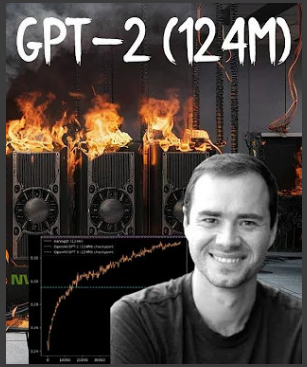


DeepSpeed

1. DeepSpeed Stage 1: optimiser state partitioning
2. DeepSpeed Stage 2: gradient partitioning
3. DeepSpeed Stage 3: model parameter partitioning

The plan

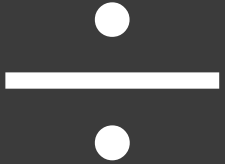
1. No more theory
2. Simplified GPT2 nano code for one GPU
3. Andrej Karpathy:
   `https://youtu.be/l8pRSuU81PU`
4. Extend to support Distributed Data Parallel
5. Extend using PyTorch Lightning



The intention

1. Not to care too much about the model implementation
2. Understand changes needed to add MPI functionality
3. For this, we'll rely heavily on `diff`s
4. Using JupyterLab and SLURM

Your most important tools



$(*, /)$

Terminology

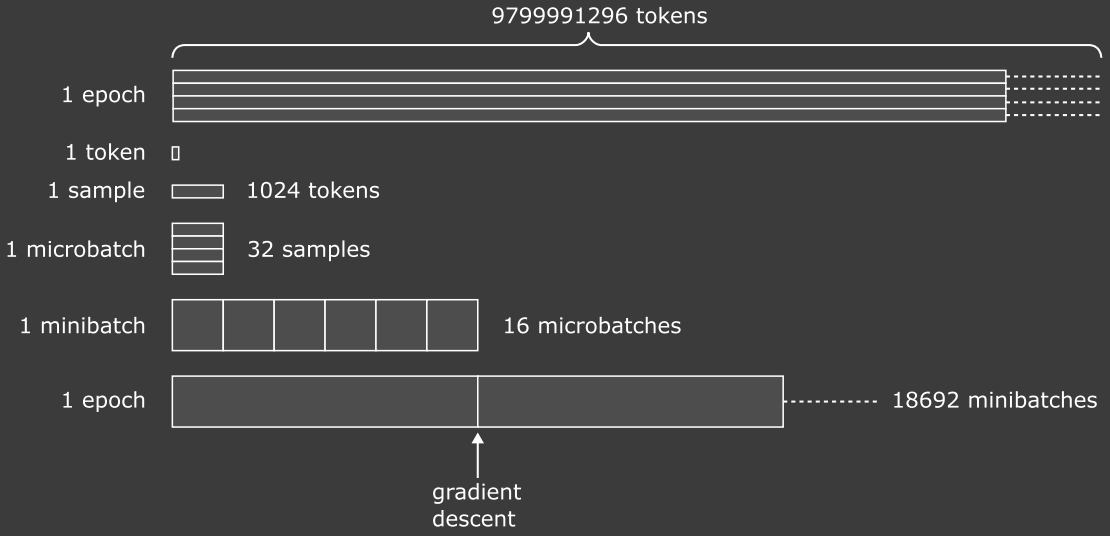
1. Vocabulary: 50257 embeddings
2. Dataset: 9799991296 FineWeb tokens, our training data
3. Sample: a sequence of 1024 tokens from the dataset
4. Microbatch: 32 samples
5. Minibatch (also called a batch): 16 microbatches, gradient accumulation performed afterwards
6. Step: the process for training on one minibatch
7. Epoch: one training sweep of the entire dataset, 18692 steps

$$18692 \times 16 \times 32 \times 1024 = 9799991296$$

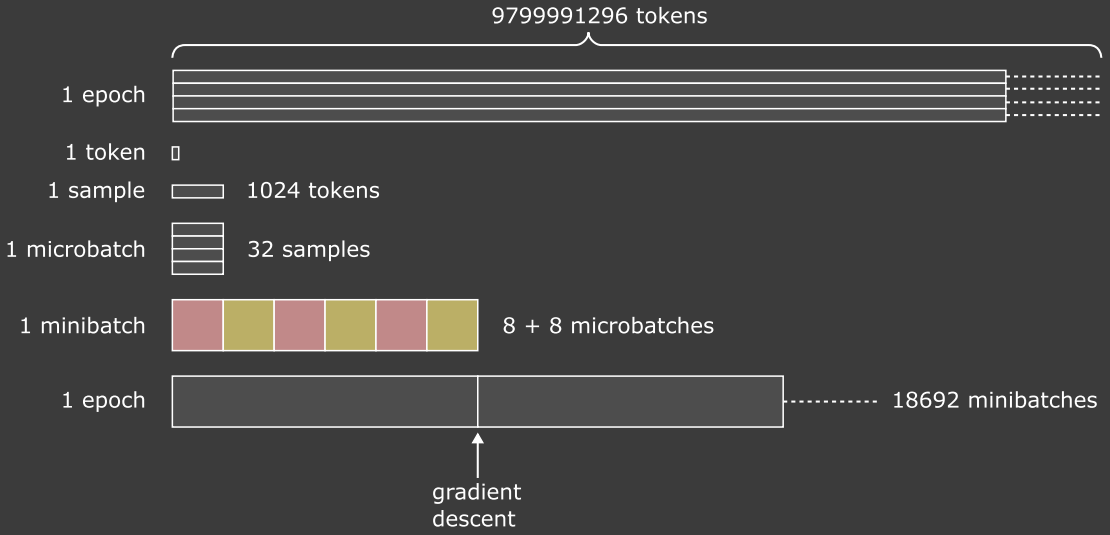
Distributed training

1. Training on n nodes, each with m GPUs
2. Total $n \times m$ GPUs
3. World size: $w = m \times n$
4. Global rank r_G is a unique index $r_G \in \{0, \ldots, w-1\}$
5. Local rank r_L is unique per device $r_L \in \{0, \ldots, m-1\}$
6. No node index, but we do have *hostnames*, *e.g.* `bask-pg0309u05a`, `bask-pg0309u06a`
7. We may also have multiple CPU *workers* for each node

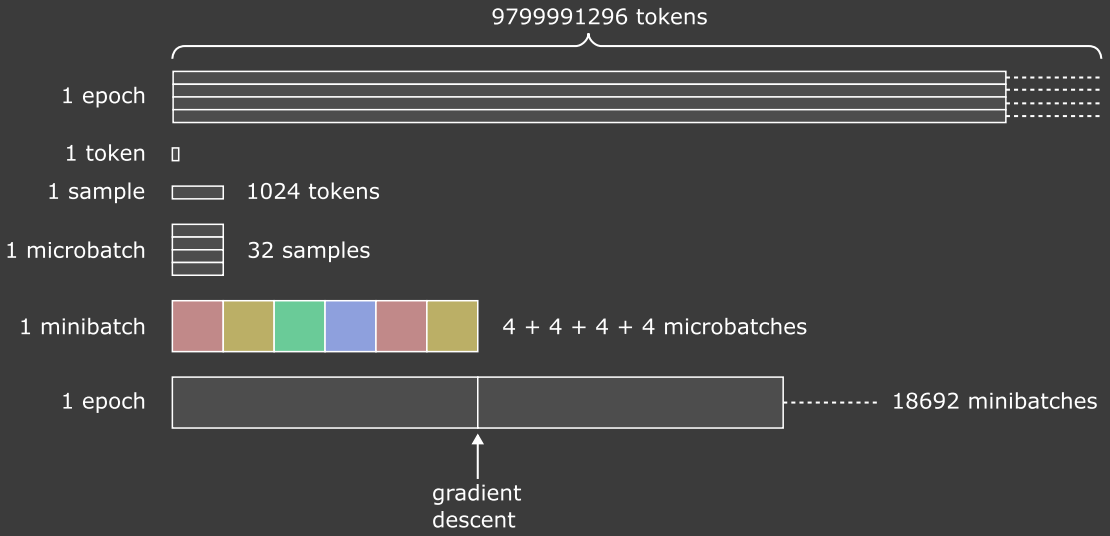
Training with 1 GPU



Training with 2 GPUs

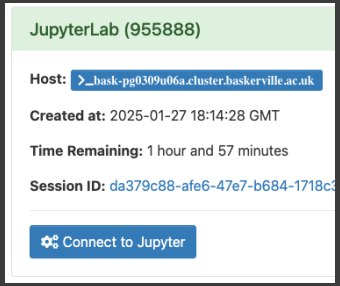


Training with 4 GPUs



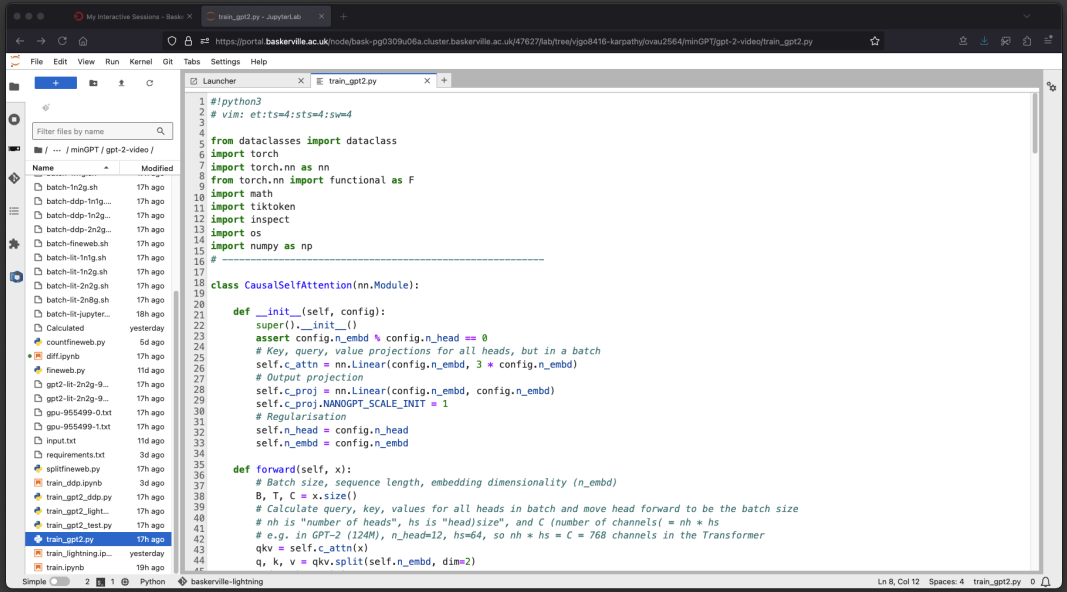
Time to look at the code

1. Open your JupyterLab page
2. Connect to Jupyter
3. Clone the repository
4. Open the `train_gpt2.py` file




```
# Move into your user directory
cd /bask/projects/v/vjgo8416-training25/$USER/

# Clone the repository
git clone \
    https://github.com/alan-turing-institute/practical-mpi.git
```


My Interactive Sessions - Baskı
train_gpt2.py - JupyterLab
https://portal.baskerville.ac.uk/node/bask-pg0309u06a.cluster.baskerville.ac.uk/47627/lab/tree/vjgo8416-karpathy/ovau2564/minGPT/gpt-2-video/train_gpt2.py
File Edit View Run Kernel Git Tabs Settings Help
Launcher
train_gpt2.py
Filter files by name
/ ··· / minGPT / gpt-2-video /
Name
Modified
batch-1n2g.sh 17h ago
batch-ddp-1n1g.... 17h ago
batch-ddp-1n2g... 17h ago
batch-ddp-2n2g... 17h ago
batch-fineweb.sh 17h ago
batch-lit-1n1g.sh 17h ago
batch-lit-1n2g.sh 17h ago
batch-lit-2n2g.sh 17h ago
batch-lit-2n8g.sh 17h ago
batch-lit-jupyter... 18h ago
Calculated yesterday
countfineweb.py 5d ago
diff.ipynb 17h ago
fineweb.py 11d ago
gpt2-lit-2n2g-9... 17h ago
gpt2-lit-2n2g-9... 17h ago
gpu-955499-0.txt 17h ago
gpu-955499-1.txt 17h ago
input.txt 11d ago
requirements.txt 3d ago
splitfineweb.py 17h ago
train_ddp.ipynb 3d ago
train_gpt2_ddp.py 17h ago
train_gpt2_light... 17h ago
train_gpt2_test.py 17h ago
train_gpt2.py 17h ago
train_lightning.ip... yesterday
train.ipynb 19h ago
#!python3
# vim: et:ts=4:sts=4:sw=4
from dataclasses import dataclass
import torch
import torch.nn as nn
from torch.nn import functional as F
import math
import tiktoken
import inspect
import os
import numpy as np
# ---
class CausalSelfAttention(nn.Module):
def __init__(self, config):
super().__init__()
assert config.n_embd % config.n_head == 0
# Key, query, value projections for all heads, but in a batch
self.c_attn = nn.Linear(config.n_embd, 3 * config.n_embd)
# Output projection
self.c_proj = nn.Linear(config.n_embd, config.n_embd)
self.c_proj.NANOGPT_SCALE_INIT = 1
# Regularisation
self.n_head = config.n_head
self.n_embd = config.n_embd
def forward(self, x):
# Batch size, sequence length, embedding dimensionality (n_embd)
B, T, C = x.size()
# Calculate query, key, values for all heads in batch and move head forward to be the batch size
# nh is "number of heads", hs is "head)size", and C (number of channels(= nh * hs
# e.g. in GPT-2 (124M), n_head=12, hs=64, so nh * hs = C = 768 channels in the Transformer
qkv = self.c_attn(x)
q, k, v = qkv.split(self.n_embd, dim=2)
Simple
Python
baskerville-lightning
Ln 8, Col 12
Spaces: 4
train_gpt2.py

GPT2 nano classes and functions

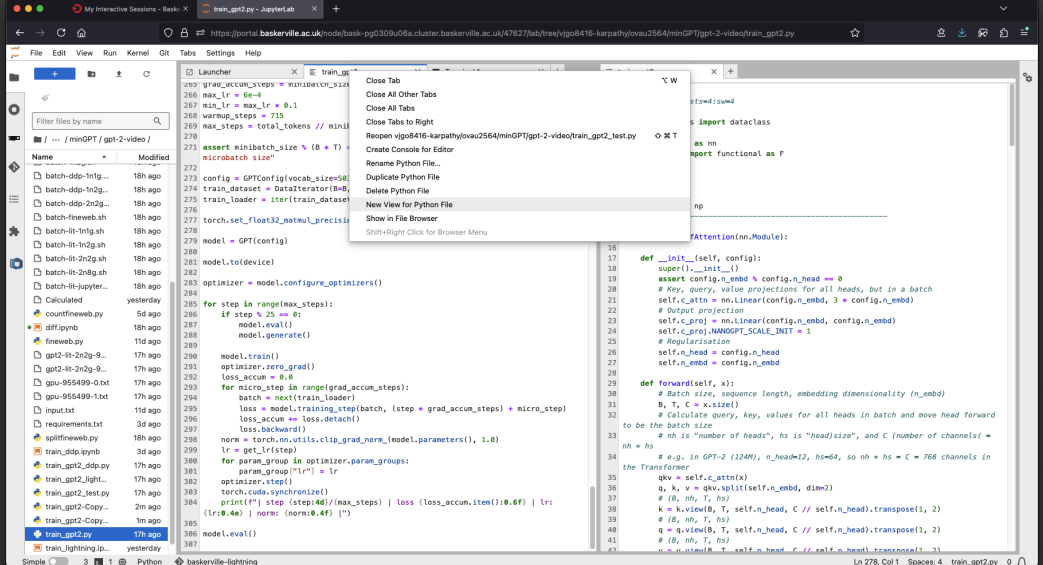
1. `CausalSelfAttention`, `MLP`, `Block`: model components
2. `GPTConfig`: model configuration
3. `GPT`: the model
4. `generate()`: inference
5. `configure_optimizers()`: training configuration
6. `training_step()`: one training step
7. `load_tokens()`: load a single FineWeb shard
8. `get_shards()`: find the FineWeb shards on disk
9. `DataIterator`: our dataset and data loader
10. `get_lr()`: calculate the learning rate

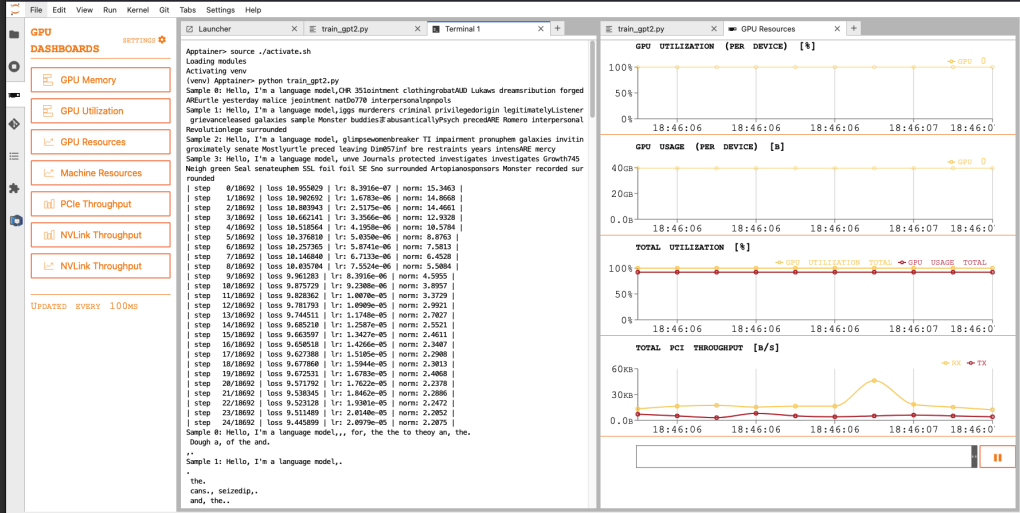
GPT2 nano execution

1. Set hyperparameters
2. Create model
3. Perform training loop

Training time!

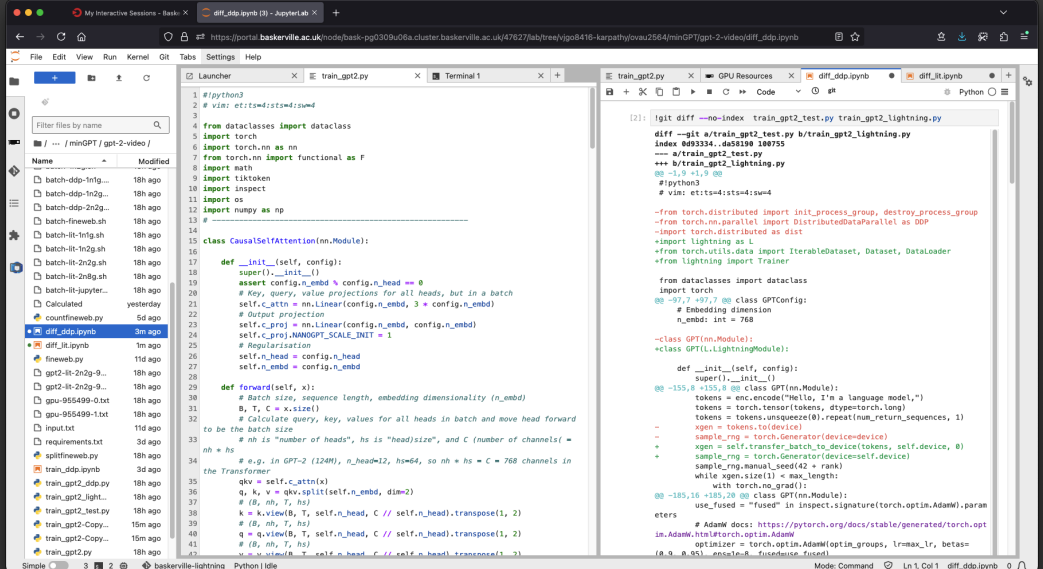
1. Right click on the `train_gpt2.py` tab
2. Select **New View for Python File**
3. Open a **GPU Resources** pane from the GPU Dashboard
4. Drag the pane to the tab space on the right
5. Open a terminal in the left tab space
6. `source activate.sh`
7. `python train_gpt2.py`


My Interactive Sessions - Bask
train_gpt2.py - JupyterLab
https://portal.baskerville.ac.uk/node/bask-pg0309u06a.cluster.baskerville.ac.uk/47627/lab/tree/vjgo8416-karpathy/ovau2564/minGPT/gpt-2-video/train_gpt2.py
File Edit View Run Kernel Git Tabs Settings Help
Launcher
Filter files by name
/ ... / minGPT / gpt-2-video /
Name
Modified
batch-ddp-1n1g....
18h ago
batch-ddp-1n2g...
18h ago
batch-ddp-2n2g...
18h ago
batch-fineweb.sh
18h ago
batch-lit-1n1g.sh
18h ago
batch-lit-1n2g.sh
18h ago
batch-lit-2n2g.sh
18h ago
batch-lit-2n8g.sh
18h ago
batch-lit-jupyter...
18h ago
Calculated
yesterday
countfineweb.py
5d ago
diff.ipynb
18h ago
fineweb.py
11d ago
gpt2-lit-2n2g-9...
17h ago
gpt2-lit-2n2g-9...
17h ago
gpu-955499-0.txt
17h ago
gpu-955499-1.txt
17h ago
input.txt
11d ago
requirements.txt
3d ago
splitfineweb.py
18h ago
train_ddp.ipynb
3d ago
train_gpt2_ddp.py
17h ago
train_gpt2_light...
17h ago
train_gpt2_test.py
17h ago
train_gpt2-Copy...
2m ago
train_gpt2-Copy...
1m ago
train_gpt2.py
17h ago
train_lightning.ip...
yesterday
Close Tab ⌥ W
Close All Other Tabs
Close All Tabs
Close Tabs to Right
Reopen vjgo8416-karpathy/ovau2564/minGPT/gpt-2-video/train_gpt2_test.py ⇧⌘T
Create Console for Editor
Rename Python File...
Duplicate Python File
Delete Python File
New View for Python File
Show in File Browser
Shift+Right Click for Browser Menu
grad_accum_steps = minibatch_size
max_lr = 6e-4
min_lr = max_lr * 0.1
warmup_steps = 715
max_steps = total_tokens // minib
assert minibatch_size % (B * T) =
microbatch size"
config = GPTConfig(vocab_size=50
train_dataset = DataIterator(B=B,
train_loader = iter(train_dataset
torch.set_float32_matmul_precisi
model = GPT(config)
model.to(device)
optimizer = model.configure_optimizers()
for step in range(max_steps):
if step % 25 == 0:
model.eval()
model.generate()
model.train()
optimizer.zero_grad()
loss_accum = 0.0
for micro_step in range(grad_accum_steps):
batch = next(train_loader)
loss = model.training_step(batch, (step * grad_accum_steps) + micro_step)
loss_accum += loss.detach()
loss.backward()
norm = torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
lr = get_lr(step)
for param_group in optimizer.param_groups:
param_group["lr"] = lr
optimizer.step()
torch.cuda.synchronize()
print(f"| step {step:4d}/{max_steps} | loss {loss_accum.item():0.6f} | lr:
{lr:0.4e} | norm: {norm:0.4f} |")
model.eval()
sts=d;sw=4
s import dataclass
as nn
mport functional as F
np
fAttention(nn.Module):
def __init__(self, config):
super().__init__()
assert config.n_embd % config.n_head == 0
# Key, query, value projections for all heads, but in a batch
self.c_attn = nn.Linear(config.n_embd, 3 * config.n_embd)
# Output projection
self.c_proj = nn.Linear(config.n_embd, config.n_embd)
self.c_proj.NANOGPT_SCALE_INIT = 1
# Regularisation
self.n_head = config.n_head
self.n_embd = config.n_embd
def forward(self, x):
# Batch size, sequence length, embedding dimensionality (n_embd)
B, T, C = x.size()
Calculate query, key, values for all heads in batch and move head forward
to be the batch size
nh is "number of heads", hs is "head)size", and C (number of channels(=
nh * hs
e.g. in GPT-2 (124M), n_head=12, hs=64, so nh * hs = C = 768 channels in
the Transformer
qkv = self.c_attn(x)
q, k, v = qkv.split(self.n_embd, dim=2)
# (B, nh, T, hs)
k = k.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)
# (B, nh, T, hs)
q = q.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)
# (B, nh, T, hs)
Simple
3
1
Python
baskerville-lightning
Ln 278, Col 1
Spaces: 4
train_gpt2.py
0


My Interactive Sessions - Baske
lab - JupyterLab
https://portal.baskerville.ac.uk/node/bask-pg0309u06a.cluster.baskerville.ac.uk/47627/lab
File Edit View Run Kernel Git Tabs Settings Help
GPU DASHBOARDS
SETTINGS
GPU Memory
GPU Utilization
GPU Resources
Machine Resources
PCIe Throughput
NVLink Throughput
NVLink Throughput
UPDATED EVERY 100MS
Launcher
train_gpt2.py
Terminal 1
Apptainer> source ./activate.sh
Loading modules
Activating venv
(venv) Apptainer> python train_gpt2.py
Sample 0: Hello, I'm a language model,CHR 351ointment clothingrobatAUD Lukaws dreamsribution forged AREurtle yesterday malice jeointment natDo770 interpersonalnpnpols
Sample 1: Hello, I'm a language model,ggo murderers criminal privilegedorigin legitimatelyListener grievanceleased galaxies sample Monster buddies‡abusantiallyPsych precedARE Romero interpersonal Revolutionlege surrounded
Sample 2: Hello, I'm a language model, glimpsewomenbreaker TI impairment pronuphem galaxies invitin groximately senate Mostlyurtle preced leaving Dim057inf bre restraints years intensARE mercy
Sample 3: Hello, I'm a language model, unve Journals protected investigates investigates Growth745 Neigh green Seal senateuphem SSL foil foil SE Sno surrounded Artopianosponsors Monster recorded surrounded
step 0/18692	loss 10.955029	lr: 8.3916e-07	norm: 15.3463
step 1/18692	loss 10.902692	lr: 1.6783e-06	norm: 14.8668
step 2/18692	loss 10.803943	lr: 2.5175e-06	norm: 14.4661
step 3/18692	loss 10.662141	lr: 3.3566e-06	norm: 12.9328
step 4/18692	loss 10.518564	lr: 4.1958e-06	norm: 10.5784
step 5/18692	loss 10.376810	lr: 5.0350e-06	norm: 8.8763
step 6/18692	loss 10.257365	lr: 5.8741e-06	norm: 7.5813
step 7/18692	loss 10.146840	lr: 6.7133e-06	norm: 6.4528
step 8/18692	loss 10.035704	lr: 7.5524e-06	norm: 5.5004
step 9/18692	loss 9.961283	lr: 8.3916e-06	norm: 4.5955
step 10/18692	loss 9.875729	lr: 9.2308e-06	norm: 3.8957
step 11/18692	loss 9.828362	lr: 1.0070e-05	norm: 3.3729
step 12/18692	loss 9.781793	lr: 1.0909e-05	norm: 2.9921
step 13/18692	loss 9.744511	lr: 1.1748e-05	norm: 2.7027
step 14/18692	loss 9.685210	lr: 1.2587e-05	norm: 2.5521
step 15/18692	loss 9.663597	lr: 1.3427e-05	norm: 2.4611
step 16/18692	loss 9.650518	lr: 1.4266e-05	norm: 2.3407
step 17/18692	loss 9.627388	lr: 1.5105e-05	norm: 2.2908
step 18/18692	loss 9.677860	lr: 1.5944e-05	norm: 2.3013
step 19/18692	loss 9.672531	lr: 1.6783e-05	norm: 2.4068
step 20/18692	loss 9.571792	lr: 1.7622e-05	norm: 2.2378
step 21/18692	loss 9.538345	lr: 1.8462e-05	norm: 2.2886
step 22/18692	loss 9.523128	lr: 1.9301e-05	norm: 2.2472
step 23/18692	loss 9.511489	lr: 2.0140e-05	norm: 2.2052
step 24/18692	loss 9.445899	lr: 2.0979e-05	norm: 2.2075
Sample 0: Hello, I'm a language model,,, for, the the to theoy an, the. Dough a, of the and.
,,
Sample 1: Hello, I'm a language model,.
.
the.
cans., seizedip,.
and, the..
train_gpt2.py
GPU Resources
GPU UTILIZATION (PER DEVICE) [%]
GPU 0
100%
50%
0%
18:46:06 18:46:06 18:46:06 18:46:07 18:46:0
GPU USAGE (PER DEVICE) [B]
GPU 0
40GB
20GB
0.0B
18:46:06 18:46:06 18:46:06 18:46:07 18:46:0
TOTAL UTILIZATION [%]
GPU UTILIZATION TOTAL GPU USAGE TOTAL
100%
50%
0%
18:46:06 18:46:06 18:46:06 18:46:07 18:46:0
TOTAL PCI THROUGHPUT [B/S]
RX TX
60KB
30KB
0.0B
18:46:06 18:46:06 18:46:06 18:46:07 18:46:0
Simple
baskerville-lightning
Terminal 1

Upgrade to DDP

1. Open `train_gpt2.py` in the left hand tab space
2. Open `diff_ddp.ipynb` in the right hand tab space
3. Execute the first cell of `diff_ddp.ipynb`


My Interactive Sessions - Baske
diff_ddp.ipynb (3) - JupyterLab
https://portal.baskerville.ac.uk/node/bask-pg0309u06a.cluster.baskerville.ac.uk/47627/lab/tree/vjgo8416-karpathy/ovau2564/minGPT/gpt-2-video/diff_ddp.ipynb
File Edit View Run Kernel Git Tabs Settings Help
Launcher
train_gpt2.py
Terminal 1
train_gpt2.py
GPU Resources
diff_ddp.ipynb
diff_lit.ipynb
Filter files by name
/ ... / minGPT / gpt-2-video /
Name
Modified
batch-ddp-1n1g....
batch-ddp-1n2g...
batch-ddp-2n2g...
batch-fineweb.sh
batch-lit-1n1g.sh
batch-lit-1n2g.sh
batch-lit-2n2g.sh
batch-lit-2n8g.sh
batch-lit-jupyter...
Calculated
countfineweb.py
diff_ddp.ipynb
diff_lit.ipynb
fineweb.py
gpt2-lit-2n2g-9...
gpt2-lit-2n2g-9...
gpu-955499-0.txt
gpu-955499-1.txt
input.txt
requirements.txt
splitfineweb.py
train_ddp.ipynb
train_gpt2_ddp.py
train_gpt2_light...
train_gpt2_test.py
train_gpt2-Copy...
train_gpt2-Copy...
train_gpt2.py
Code
Python
[2]:
!git diff --no-index train_gpt2_test.py train_gpt2_lightning.py
Simple
3
2
baskerville-lightning
Python | Idle
Mode: Command
Ln 1, Col 1
diff_ddp.ipynb
0

Understanding unified diffs

1. `@@` prefix indicates line numbers
   `@@ -before,len +after,len @@`
2. `+` prefix in green indicates lines added
3. `-` prefix in red indicates lines removed
4. Replay the cell to update the diff after making changes

```
@@ -145,7 +149,7 @@ class GPT(nn.Module):
                loss = F.cross_entropy(logits.view
s.view(-1))
        return logits, loss

-    def generate(self):
+    def generate(self, rank):
        num_return_sequences = 4
        max_length = 32
        tokens = enc.encode("Hello, I'm a lang
@@ -153,7 +157,7 @@ class GPT(nn.Module):
        tokens = tokens.unsqueeze(0).repeat(nu
        xgen = tokens.to(device)
        sample_rng = torch.Generator(device=de
-       sample_rng.manual_seed(42)
+       sample_rng.manual_seed(42 + rank)
        while xgen.size(1) < max_length:
            with torch.no_grad():
                with torch.autocast(device_typ
6):
@@ -207,22 +211,24 @@ def get_shards(split):

 class DataIterator:

-    def __init__(self, B, T):
+    def __init__(self, B, T, num_processes, pr
```

Manually apply the diff – part I

1. Imports
2. One process per GPU
3. We *generate* on every process
4. One `DataIterator` per GPU
5. We must shard the data appropriately
6. Initialise the process group
7. Harvest **world size**, **rank** and **local rank** from the environment
8. Where do these come from?

Manually apply the diff – part II

9. Fix the gradient accumulation steps
10. Batch sizes must align
11. Create a `DDP` model
12. Use the right model variable at the right time
13. Trigger backward gradient sync
14. Perform all reduce
15. Destroy the process group

Observations

1. Most of this is boilerplate
2. The hardest part is sharding the data correctly
3. Because this is *data parallel*
4. MPI is also hard

. . . but it's done for us by `torch.distributed` and `DDP`

Training time!

1. See `batch-ddp-1n1g.sh` for single GPU
2. See `batch-ddp-2n3g.sh` for dual-node dual GPU

```
# Single GPU
python -m torch.distributed.launch \
    --standalone --nproc_per_node=1 \
    train_gpt2.py

# Multi-node
python -m torch.distributed.launch \
    --nproc_per_node=${SLURM_GPUS_PER_NODE} \
    --nnodes=${SLURM_NNODES} \
    --master-port=${MASTER_PORT} --master-addr=${MASTER_ADDR} \
    train_gpt2.py
```

Upgrade to Lightning

1. Lightning is conceptually different
2. Code is organised in `LightningModule`: init, train step, validation step, test step, optimisers
3. Code outside `LightningModule` is automated by `Trainer`
4. Remove code moving data to the GPU

Upgrade to Lightning

1. Open `train_gpt2.py` in the left hand tab space
2. Open `diff_lit.ipynb` in the right hand tab space
3. Execute the first cell of `diff_lit.ipynb`

Manually apply the diff – part I

1. Imports
2. Switch from `Module` to `LightningModule`
3. Fix the `generate()` function
4. Use the built-in `OneCycleLR` learning rate scheduler
5. Remove CUDA code from `training_step()`
6. Generate examples periodically
7. Initialise `DataIterator` using a `worker_init_fn()`
8. Remove our custom learning rate scheduler code

Manually apply the diff – part II

9. Remove the process group code
10. Lose our DDP configuration
11. Simplify the random seeding
12. Fix the steps: minibatches are now implicit
13. Use a `DataLoader` to sequence data loading
14. Remove our training loop
15. Add the `Trainer` code

Observations

1. The training code is greatly simplified
2. Lightning is SLURM-aware
3. So MPI is even easier
4. We can now switch to other strategies (*in theory*)

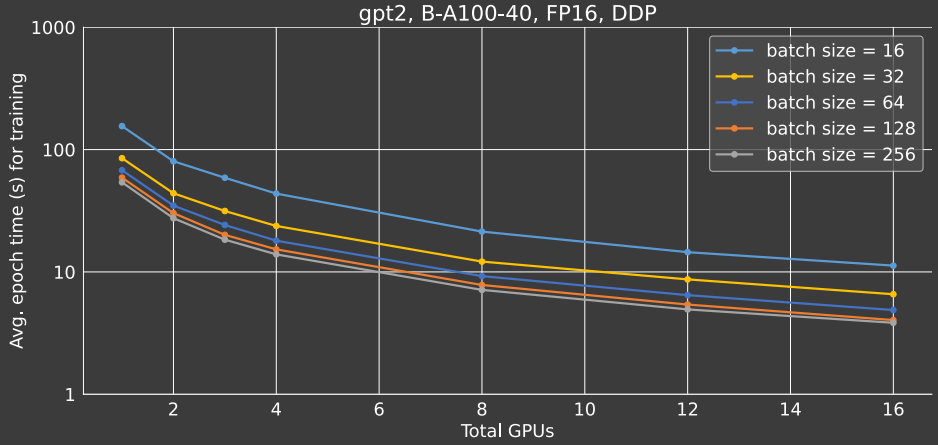
Training time!

1. See `batch-lit-2n8g.sh` for dual-node eight GPU
2. See `batch-lit-jupyter.sh` to queue from Jupyter

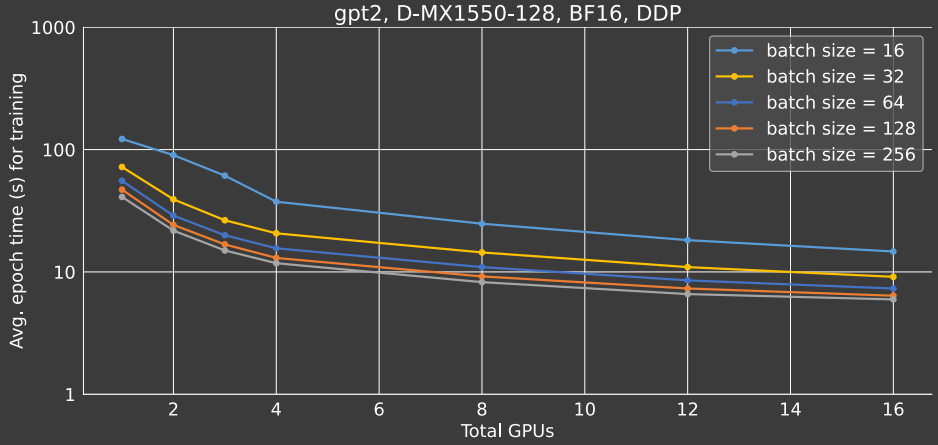
```
# From inside a SLURM batch script
srun python train_gpt2.py

# Queue execution from inside JupyterLab
sbatch batch-lit-jupyter.sh
```

Scaling across nodes on A100 (40 GiB) GPUs



Scaling across nodes on MX1550 (128 GiB) GPUs



Wrapping up

1. Converting code for distributed training is doable
2. Review and try out the batch scripts
3. More examples in the `hpc-landscaptes` repository

`https://github.com/alan-turing-institute/hpc-landscape`